Fall 10-28-2010

A Single-Chip Ultra-Wideband Based Wireless Sensor Network Node

Nathan R. Schemm
University of Nebraska at Lincoln, nathan.schemm@huskers.unl.edu

A SINGLE-CHIP ULTRA-WIDEBAND BASED WIRELESS SENSOR NETWORK NODE

by

Nathan Schemm

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Engineering

(Electrical Engineering)

Under the Supervision of Professor Sina Balkır and Professor Michael W. Hoffman

Lincoln, Nebraska

December, 2010

A SINGLE-CHIP ULTRA-WIDEBAND BASED WIRELESS SENSOR NETWORK NODE

Nathan Schemm, Ph. D.
University of Nebraska, 2010

Advisers: Sina Balkır and Michael W. Hoffman

This dissertation presents the design of a next-generation wireless sensor network node. The node incorporates many new and innovative technologies such as an ultra-wideband radio which allows very low-energy communication, a low-power radiation detection front end, and an efficient implementation of dynamic voltage scaling which improves the energy efficiency of the integrated processor. The complete design is integrated on a single chip for maximum power savings and minimal size.

The ultra-wideband transceiver includes many novel techniques to produce a receiver with low power consumption and fast and accurate packet acquisition and reception. These include the use of a standard symmetric inductor in a non-standard way as a T-coil load which extends the bandwidth of RF amplifiers, and a novel baseband architecture which relies on parallel analog processing to achieve both low power and fast and accurate signal acquisition and tracking.

Other features include a very low power interface to radiation detection detectors which uses an architecture in which the ADC only samples when an event occurs. This decreases the power over a conventional continuous-sampling scheme. A low-power charge sensitive amplifier is designed for the front end while the peak detection is done in the analog domain before an ADC converts the peak value to digital.

An efficient implementation of dynamic voltage scaling allows for unmatched computational power consumption combined with good sleep mode power. This is achieved through the integration of two buck converters which have high efficiency even at light loads.

The complete design was integrated on a single chip using a 0.18 μm CMOS process with a total die area of 3×4.5 mm. The fabricated chip was tested and the measured results show that the chip does indeed achieve low power consumption. The measured results validate the approaches chosen, and field tests show the system is useful when interfaced to a variety of sensors for detecting various types of radiation.

ACKNOWLEDGMENTS

There are many people who without their assistance, this dissertation would not be possible. I would like to thank Professor Sina Balkır from whom I learned much about analog VLSI design and other life topics such as human relations and middle east politics, and to Professor Michael W. Hoffman who was in invaluable resource for any non-circuits knowledge. I would like to thank Professor Mark Bauer for his friendship and advice in many areas and for assisting me with some chip testing.

I would like to thank my friends and fellow graduate students for their support and encouragement during my tenure at Lincoln. Your support and friendship made my life at Lincoln infinitely more enjoyable.

Finally, I would like to thank my parents, Richard and Sue Schemm, and my sisters Hannah, Naomi, and Rachel for their loving support over the years. I would particularly like to thank my Dad for "teaching me everything he knew about electricity in five minutes" when I was very young (to his credit, it took him many years), and for always allowing me to add one more piece of broken-down, taken-apart junk to his basement workshop whenever I was able to acquire one. Without such an upbringing, I would not be where I am today.

GRANT INFORMATION

This work has been funded by a DOE grant entitled “All Solid-State Wireless Sensor Network for Nuclear Proliferation Detection.”

Contents

List of Figures

List of Tables

Nomenclature

ADC	Analog to digital converter
AGC	Automatic gain control
ALU	Arithmetic and logic unit
AWGN	Additive white Gaussian noise
ECC	Error correcting code
CDMA	Code division multiple access
CMOS	Complementary metal-oxide semiconductor
COTS	Commercial off the shelf
CRC	Cyclic redundancy check
CSA	Charge sensitive amplifier
CTS	Clock tree synthesis
DAC	Digital to analog converter
DCO	Digitally controlled oscillator
DNL	Differential non-linearity
DRC	Design rule check
DVS	Dynamic voltage scaling
FFT	Fast Fourier transform
FIFO	First in first out
FIR	Finite impulse response
GPS	Global positioning system
I^2C	Inter-IC bus

IIR	Infinite impulse response
IP	Intellectual property
INL	Integral non-linearity
ISM	Industrial medical scientific
ISR	Interrupt service routine
I/O	Input/Output
LDO	Low-dropout voltage regulator
LNA	Low noise amplifier
LO	Local oscillator
LSB	Least significant byte/bit
LVS	Layout versus schematic
MSB	Most significant byte/bit
NMI	Non-maskable interrupt
OFDM	Orthogonal frequency division modulation
OOK	On-off keying
OTA	Operational transconductance amplifier
p-cell	Parameterized Cell
PFD	Phase-frequency detector
PFM	Pulse-frequency modulation
PLL	Phase-locked loop
PMT	Photomultiplier tube
PN	Pseudorandom noise
PPM	Pulse-position modulation
PSK	Phase-sift keying
PVT	Process voltage temperature
PWM	Pulse-width modulation
QAM	Quadrature amplitude modulation
RAM	Random access memory

RF	Radio frequency
RMS	Root mean squared
ROM	Read only memory
RTL	Register transfer level
SAW	Surface acoustic wave
SNR	Signal to noise ratio
SPI	Serial Peripheral interface
SRAM	Static RAM
UART	Universal asynchronous receiver/transmitter
UWB	Ultra-wideband
VCO	Voltage controlled oscillator
VHDL	VHSIC hardware description language; VHSIC—very-high-speed integrated circuit
WSN	Wireless sensor network

Throughout the dissertation references may be made to registers and specific bits within the registers. These registers define the user interface which a program uses to interact with the hardware. When referenced, the register names and bit names are denoted with an different font, for example the HFEn bit in the OSC_CR register enables the crystal oscillator.

For full definitions of all the registers and bits within them, see the supporting chip datasheet.

Assembly instructions for the integrated processor are denoted with a fixed-width font, for example `mov r4,r5`.

Part I

Introduction

Chapter 1

Overview

The miniaturization of electronics and sensors in recent years had led to the development of wireless sensor networks (WSN). These consist of a number of small nodes, called motes, which have one or more sensors of some type and wireless connectivity to other motes. The motes may be deployed over an area to monitor a physical quantity of interest. Wireless sensor networks have many exciting applications in diverse fields ranging from habitat monitoring to medical applications to defense systems. The one theme in WSNs have in common is extremely stringent energy requirements. Simultaneously, many new applications are not feasible with today's power consumption, but with lower energy requirements, a vast array of exciting applications could be enabled.

The power consumption of today's motes allow battery lifetimes in the order of weeks to months, however many applications only become feasible with much longer battery lifetimes of years. To achieve this the average power consumption of a mote must be dramatically decreased from the 3-20 mW consumed on average by today's motes to 100-500 μW for battery lifetimes of up to 5 years from two AA batteries.

To achieve this power consumption, revolutionary new hardware is required which decreases the power consumed in all areas of the system. This dissertation presents the design of an ultra-wideband (UWB) based single-chip wireless sensor network node. The presented sensor node has many advanced features which allows a substantial improvement over present state-of-the-art designs in virtually every aspect. The presented hardware will allow the permeation of wireless sensor networks into a variety of new applications that are not practical with today's technology.

The architecture for the design is shown in Figure 1.1. This is a standard architecture with the exception of the dual RF transceivers present on this design. Each of the parts will be discussed in later chapters. The power consumption of each must be minimized to minimize total system power consumption.

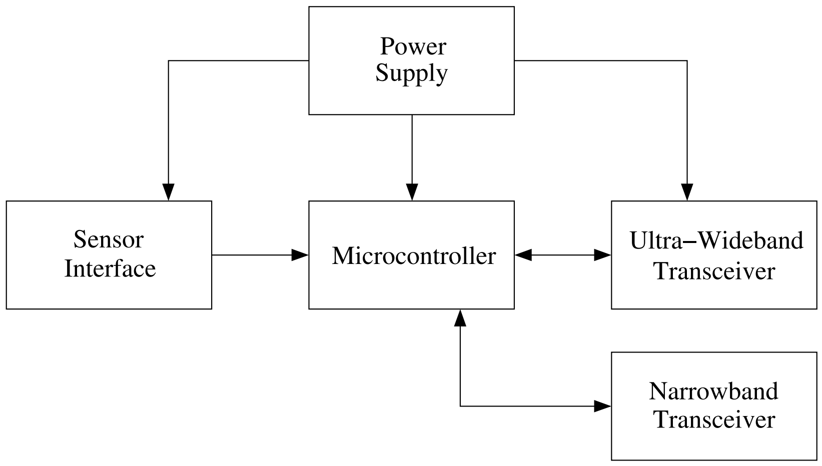


Figure 1.1 – WSN node architecture presented in this dissertation.

1.1 Contributions

The sensor network node designed has many improvements over existing nodes in each of the major sub-systems shown in Figure 1.1. The main contributions of this work include a number of low-power innovations targeted at many different sub-systems within a wireless sensor network. To dramatically decrease the system power consumption, the power of each of the individual blocks within a WSN must be decreased. Combining these numerous low power innovations can decrease the power consumption of future wireless sensor networks. These innovations include:

- A low-power event driven charge-sensitive amplifier front end for sensor interface applications. The front end measures the size of the charge pulse. The presented front end has much lower power consumption than any in the literature. This is achieved through two methods. One is by utilizing a low-power architecture for the initial charge sensitive amplifier. The novel architecture allows the bias currents in the amplifier to be scaled to very low values while still maintaining good performance. Further power savings is achieved through the use of an event-driven front end where the charge pulse amplitude is measured in the analog domain before being converted to digital. This decreases the sample rate of the ADC to simply the count rate as opposed to the Nyquist rate for a conventional front end, thus saving power. Overall, the CSA power consumption is as low as $4+0.075R\,\mu$W where R is the particle rate in thousands of particles per second.

- A low-power digital core for processing the sensor data and mote control functions. The low power consumption is achieved through the use of dynamic voltage scaling which is in turn implemented with extremely low-power integrated buck power converters. The buck converters also support a sleep voltage which is lower than the operating voltage to save leakage power. The buck converters are designed to have a high efficiency even when the output current is very small by using a pulse-frequency modulation scheme. Using buck converters instead of the alternative linear regulators allows high efficiency, even with a low output voltage. The digital sleep power consumption of the system can be as low as 3 μW while the computational power can be as low as 140 μW/MHz.

- A low-power UWB transceiver for use in short range wireless sensor networks. The simple analog architecture consumes much less power than competing designs, and is one of the few complete UWB transceivers in the published literature. The design includes a novel wideband amplifier in the front end which uses a standard symmetrical inductor as a transformer to extend the bandwidth of an amplifier while still only requiring the area of one inductor. The baseband architecture contains many parallel analog correlators which simultaneously allow both low power and accurate and reliable signal acquisition, even in poor channel conditions. Furthermore, unlike other designs, it includes a variable rate hardware error control code to simultaneously maximize reliability, range, and datarate. The UWB transmitter uses 0.46-2.3 nJ/bit to transmit the data while the UWB receiver consumes 3.52-17.6 nJ/bit to receive the data. This is much lower than any narrowband communication power by at least an order of magnitude.

- A simple low-power narrowband transceiver. This can can be used in applications where power consumption is critical and the associated negative aspects of the receiver can be accommodated.

- A single-chip solution. The single chip solution allows lower cost, higher performance, and lower power over multi-chip solutions.

Both the UWB and narrowband transceivers have a much lower energy/bit than conventional transceivers. The UWB transceiver can be low power due to the wide bandwidth usage, while the narrowband receiver can be low power simply due to its simple architecture.

All the required active circuitry, except for the flash memory for program storage, are integrated on a single die. This provides many benefits for sensor networks. First, with sufficient volume, it can decrease the price of a sensor network node. Furthermore, it can also decrease the physical size of the electronics. Finally, the high integration allows for power savings which can be gleaned from the efficiency that comes with high integration. As an example, consider a COTS sensor node such as the MicaZ mote, where there are separate processor and radio transceiver chips connected by a serial bus. The serial bus connecting the two operates at a couple mega-baud. This means that when a radio packet needs to be transferred between the two chips, it takes some time for this to occur. Not only does this slow down the maximum throughput of the transceiver, it also requires much processing by the microcontroller to individually send or receive the bytes to/from the radio chip. This requires the processor to execute many more instructions than in the presented design where the overhead of communicating with the transceiver is negligible.

Four separate chips were designed for the the dissertation, one on a 0.35 μm process, and the other three on a 0.18 μm process. The final chip is the culmination of all previous chips, the design of which will be discussed in Part II of this dissertation. The final chip area is $3{\times}4.5$ mm^2 and has many impressive specifications. The dynamic digital power consumption consumes up to an order of magnitude less power than the widely available commercial-off-the-shelf (COTS) solutions, the UWB transmitter consumes two orders of magnitude less power consumption, while the UWB receiver consumes one order of magnitude less power. Furthermore, the processing frequency can be many times greater than similar COTS products. Combined, this should allow wireless sensor networks to find uses in many new applications where previously not possible.

The design was funded by the DOE grant titled “All Solid-State Wireless Sensor Network for Nuclear Proliferation Detection” In this grant, it was specified that the target topology is a simple star, where all nodes have direct communication with a less-power-constrained sink. Furthermore, it was also specified that the communication from the sink to the nodes use a narrowband technique, due to the possibility of a simple low-power receiver at the node, and the communication from the nodes to the sink use a UWB technology due to the possibility of a simple low-power transmitter. This dictated the requirement for both a UWB and narrowband functionality.

Per the grant, the node only needs to integrate the UWB transmitter and the narrowband receiver, while the sink may implement the UWB receiver and narrowband transmitter however it likes, even with high power components. However, when deciding how to make the UWB receiver, it was determined to fully integrate this on the chip, and the same was done for the narrowband transmitter. Therefore, the fabricated sensor node has two complete transceivers and thus could support a multi-hop sensor network with the addition of a suitable protocol.

1.2 Organization

This dissertation is divided into three parts. Part I which contains chapters 1 and 2 is the introduction and overview of the dissertation. This presents some background information and outlines the major contributions of this work.

Part II which contains chapters 3-7 presents the design of the individual parts of the presented sensor network. Chapter 3 presents the charge sensitive amplifier front end and the event-driven ADC. Chapter 4 presents the digital core. This is a standard MSP430X microcontroller with a few small modifications. Chapter 5 presents the ultra-wideband transceiver. This includes the convolutional code used, the transmitter, and the receiver. Chapter 6 presents the the low-power narrowband transceiver. Finally, chapter 7 presents some supporting circuitry including the power supply and clock system.

Part III which contains chapters 8-12 presents the measured results of the various chips. Chapter 8 presents the test boards and software used to test the chips. Chapter 10 presents the measured results from the UWB and narrowband transceivers. Chapter 11 presents the measured results for the processor core and provides a comparison of the complete system with other WSN platforms. Finally, chapter 12 concludes the dissertation.

Chapter 2

Motivation

This chapter gives some background information and motivation for this work. Section 2.1 discusses the applications, requirements, and principles behind wireless sensor networks and the specific hardware requirements which need to be addressed. Section 2.2 gives background information to the problem of power consumption in digital circuits and what approaches have been taken to address it. Section 2.3 discusses a number of techniques which are used to detect radiation and the benefits of each. This section also discusses the conventional detection electronics used with these detectors. Section 2.4 discusses UWB technology and the presently used designs. Section 2.5 presents the merits of a single-chip solution and how this can provide better performance than a multi-chip solution. Section 2.6 presents a brief overview of the various chips where were designed and fabricated for the project.

2.1 Sensor Networks

Recent advances in microelectronics and sensors have enabled the development of miniaturized low-power sensor systems which can be networked in a wireless manner for long-duration wide area monitoring. They cover a vast array of applications with each application having different requirements [1, 2]. Each application generally has different constraints on power, weight, communication range, battery life, and cost, so many different architectures exist to fit many different needs.

2.1.1 Applications

Wireless sensor networks have been used for many different applications, each with different requirements. Nonetheless, the hardware for many different applications uses a similar base unit with different sensors attached.

One important application is habitat and wildlife monitoring. In [3] a wireless sensor network was used to monitor the presence of petrel birds on Great Duck Island off the coast of Maine. In this application the wireless sensor network allows for monitoring of the bird's burrows without disturbing them. A similar application is presented in [4] except they are monitoring rare or endangered plants instead of animals. In [5] the sensor networks were carried on the animals, Zebras in this case, and the authors studied how to handle the mobile and constantly changing nature of the network. In this case each sensor has GPS capabilities to record the movement pattern of the individual Zebras. A similar, but more advanced work is also being carried on by researchers here at UNL on tracking whooping cranes [6]. This is more challenging due to the very severe weight constraints.

Wireless sensor networks can also be used to monitor environments which may pose a danger to prolonged human presence such as hurricanes or volcanoes. In [7] a wireless sensor network was designed to monitor infrasonic signals during volcano eruptions.

Wireless sensor networks have also found application in agricultural contexts. In [8] a wireless sensor network was designed to determine what parasites may be present in a vineyard and determine the need for an insecticide. In this case the focus was on the sensor node processing the local data and making its decision. [9–11] investigates the properties and feasibility of placing the sensors underground and the effect this has on the wireless channel.

Wireless networks also have many applications in the medical field. In [12] a system was made to monitor patients' vital signs and report to medical health professionals for assistance in diagnosis.

A multitude of military applications also exist for sensor networks. In [13] a WSN-based system was built to determine the location of a sniper based on the time of arrival.

Though there exist a number of different applications, some unifying themes can be found in them. First, and most importantly, the sensors are generally small, cheap and power constrained. Due to their small size and cost sensitivity, large batteries cannot be used, so power consumption is of the utmost concern in sensor networks. This one factor is the main driving force for much of the research being done in the field, and is also the main factor impeding the deployment of sensor networks to a much larger set of applications than they are used for today.

Other defining characteristics of wireless sensor networks include a much higher deployment density than traditional ad hoc wireless networks, nodes which may be prone to failure, and many times the network connectivity is not static but can change with time due to either node mobility, node failure, or wireless channel conditions.

2.1.2 Lifetime

The lifetime of a sensor network is dependent upon the power it consumes and the size of the battery. To achieve a low-power system, the power must be decreased at all levels in both hardware and software [14]. The energy consumed in the wireless sensor network depends somewhat on the application, but in many applications it is dominated by the power consumed by the radio. The computational power and the leakage power are also significant in some applications. Finally, the sensing power varies greatly from application to application. In some applications it is negligible, however in others it can be dominating depending on the sensor modality [15].

The power consumption of the radio must be attacked from many angles to successfully decrease the power. This can include circuit-level design improvements, transceiver architecture level improvements, modulation and coding changes, radio technology and band used, and higher software-level functions such as medium-access protocol.

One method of decreasing the power is using a flexible modulation scheme which changes depending on the channel conditions [16]. For example, using a larger constellation size when the channel is good (i.e. high SNR) results in lower power consumption. Changing the code strength on the link depending on the channel conditions can also save power.

Various radio architectures can be used, but narrowband communication architectures presently dominate the field. However ultra-wideband radios can theoretically allow much higher data rate at much lower power when compared to conventional radio because the transmission bandwidth can be traded off for power and data rate.

At a higher network-level, the energy consumption can be decreased by a number of factors [17]. Some factors which can be optimized are area coverage, request spreading and data aggregation. Some methods focus on either decreasing the coverage or providing the same coverage with fewer nodes to allow some nodes to sleep and save power [18]. The authors in [19] analyze the precision and resolution requirements and use data aggregation at nodes to decrease the amount of data to transmit and save energy. Finally, the nodes within a network must attempt to spread out the network traffic between the nodes otherwise certain commonly-used nodes will run out of battery more quickly which may bring down the complete network [20].

WSNs generally use batteries, however energy scavenging can also be used [21]. Solar is the most common method, and can be used indoors with much lower power levels [22]. Solar generally requires some low-power maximum power point tracking circuit [23] to maximize the energy harvested. Mechanical energy harvesting focuses on kinetic and vibrational energy and can be harvested either by piezoelectric capacitors or a mass-spring-magnet system. In [24] a system was built with piezoelectric vibration harvesters. For worn sensors, energy can be harvested from the environment as in [25] where energy from walking was used.

The power consumed in the digital part of the chip includes the static digital leakage power and the dynamic power. Both of which may be significant depending on the application. The following section discusses the digital power consumption in more detail.

2.2 Digital Power Consumption

Both leakage power and dynamic power may be important depending on the application. A number of different methods can be used to decrease both static and/or dynamic power.

Leakage power is the power consumed when no digital switching activity is taking place. At any given instant, the vast majority of the logic in the system is not switching, so it is only dissipating leakage power. This power does not depend on switching frequency. It originates as sub-threshold leakage current across a transistor which is nominally off, but may leak a small amount. The leakage current for the classic long-channel approximation is [26]

$$I_{ds} \approx \frac{W}{L} I_s e^{\frac{V_{gs}-V_{th}}{a(kT/q)}}. \tag{2.1}$$

The amount of leakage is determined by a number of factors. First, it is an exponential function of $V_{gs} - V_{th}$. If we assume $V_{gs} = 0$, then this becomes a exponential function of V_{th}. As technology is scaled, the supply voltage is also generally scaled, and to maintain performance, the V_{th} must also be scaled. This means that smaller processes have higher leakage currents. The leakage also depends on the W/L of the transistor as well as process parameters I_s and a.

The long-channel approximation does not mention V_{ds} however for short-channel devices, the leakage also scales with V_{ds} due to a number of reasons [27]. For a static CMOS inverter, V_{ds} is the same voltage as V_{dd}. The most significant leakage is drain-induced barrier lowering (DIBL) in which the source injects carriers into the channel due to the high E-field present in the channel. A lower drain voltage decreases the injection. Other less-significant short-channel leakage effects include gate-oxide tunneling, hot-carrier injection, gate-induced drain leakage (GIDL), and punchthrough leakage [27].

One way of combating leakage power consumption is to use a larger process with a higher threshold and supply voltage, however this negatively affects dynamic power. A second way is to use high threshold transistors in a smaller process, however the higher-threshold transistors are slower than their normal-threshold counterparts. With an advanced digital design flow, it is possible to mix regular V_t and high V_t transistors so that the logic on the critical path is regular V_t while the remaining logic is high V_t. Another technique is using back-biasing of the substrate to dynamically change the threshold voltage allowing higher threshold voltages when the circuit is sleeping and lower threshold voltages when the circuit is active. The main drawback of this is the need for specialized IP which supports the separate substrate biasing. This IP requires more area for the same circuitry and is not generally readily available. Another drawback is the need for the circuit to generate the back-bias voltage.

Another way to decrease leakage power is to change the states of the sleeping circuit to a configuration with less leakage. This works because each gate in the system leaks a different amount depending on the input voltage levels. Therefore, a circuit in one state may leak more than the same circuit in another state. Finding the state with the minimal leakage and entering it when the system is in sleep mode can save power, however there is some overhead power to transition to and from the lower-power state as well as the extra circuitry needed to implement this switch.

Another effective way of decreasing power is to switch off the power of the inactive circuits with large power gating MOSFET. This is very effective, however it is difficult to implement at a fine granularity due to the many power domains necessary. The power MOSFETS also require much area and the gated circuits suffer some performance hit due to their on resistance.

Dynamic power is the power consumed when the logic is switching states. It occurs from the repeated charging and discharging of the parasitic capacitance at each node in the circuit. This power is approximated as [28]

$$P_d = \frac{1}{2} F \alpha C_{\text{eff}} V_{dd}^2 \tag{2.2}$$

where C_{eff} is the effective capacitance of the digital circuit which is proportional to circuit area. α is the activity factor of the circuit which is defined as the fraction of the total nodes in the circuit which switch in an average clock cycle. V_{dd} is the supply voltage, and F is the clock frequency. If we only concern ourself with energy consumption, the F term drops out. Glitches in the system also increase power consumption, glitches effectively cause more than one switching transition per cycle on a node, so effectively increase α. Therefore to decrease the energy consumption we can target any and all of the remaining factors. Scaling the process reduces C_{eff} and V_{dd}, so this is effective at reducing the power–until the static leakage power begins to dominate. α can be reduced using certain digital design techniques which either decrease glitches or disable switching in parts of the circuit when not being used. Finally, the voltage can be reduced at the expense of maximum frequency.

Other design styles can also be used to decrease power such as pass-transistor logic or dynamic logic such as domino logic [29]. These can lower power by using fewer transistors and thereby decreasing capacitance and increasing speed, nonetheless, these design styles have their own problems such as dynamic logic being susceptible to noise. Furthermore, these are not supported in the standard static CMOS design flow.

After a process is chosen and the circuit complexity and activity factor are minimized, the most powerful tool to reduce both the static and dynamic power in digital circuits is scaling the voltage, and by extension the frequency. This dynamic voltage scaling (DVS) can ideally reduce both the static and dynamic power quadratically. However, to date this promising technique has not been widely adopted in wireless sensor networks. The main reasons for this include:

- Quiescent power of the regulators required generate the smaller voltage add to the sleep current and can dramatically affect power consumption.
- If using simple linear regulators, the efficiency of the regulator decreases linearly with decreasing output voltage, thereby reducing the effectiveness of DVS.
- Extra complexity and cost associated with external regulators and advanced power management circuits add to system cost and size.

- DVS complicates software development and may require complex software algorithms to manage the supply voltage intelligently. It may also add to interrupt latency causing missed deadlines in a real-time system.

The first three constraints can be addressed with the addition of an extremely small and simple buck regulator on the sensor-network chip. The regulator must be designed so it consumes an absolute minimum amount of quiescent power and can maintain high efficiency at light loads. Using a buck regulator as opposed to a linear regulator allows it to maintain a constant high efficiency, even with very low output voltages. Finally, integrating it on the sensor-network chip means minimal extra board-level components will be required, furthermore, the tight integration between the processor and power supply allows for further savings of sleep current.

Conventional DVS schemes work by allowing the processor under program control to change the supply voltage. This works well at reducing the dynamic power consumption, however to get the full benefits of DVS when in a sleep state, there should be some type of fully-automatic switch between a lower voltage and higher voltage state as the processor enters and exits from sleep state. This is because the voltage can usually be scaled further in sleep state than is possible in active state. So, the processor hardware must implement some way of automatically increasing the voltage from the sleep-state voltage to the minimum active state voltage before it begins executing instructions. This requires either having the processor and power supply on the same chip, or at least have the processor designed with the DVS power-up sequencing capability built-in. This is not present in most COTS microcontrollers, so this means that no presently available sensor networks have this capability.

The architecture of the digital processor as well as the implementation details has a major effect on the power consumption. In particular, the activity factor of the digital logic is greatly affected by the implementation architecture and the use of power-saving methods such as clock gating. The processor architecture also has some effect on power consumption. However, for this work the main focus was not on low-power processor architecture, so an off-the-shelf processor architecture was used. This has many advantages over creating a new architecture, chiefly the ability to use off-the-shelf development tools such as a compiler and assembler. The off-the-shelf processor architectures are already highly optimized, so little may be saved by making your own.

With an architecture chosen, the implementation must be done with minimal power consumption. Standard low-power design techniques such as aggressive clock gating and operand isolation are used to decrease the activity factor of the logic. Using pipelining can save power because it allows for a faster clock frequency which in turn can be traded off for power savings with DVS. It also adds circuit complexity and makes the implementation much more time consuming and error-prone due to the very much expanded testing space.

Most of the low-power processor architectures available today use very limited if any pipelining for simplicity; this is the approach used for this work as well.

2.3 Radiation Detection Methods

This section discusses differing ways to detect radiation—both the detector technologies and the interface electronics needed for them. Detection of neutrons and gamma rays will be examined, though other detector technologies have similar electronic interface requirements.

2.3.1 Neutron Detection Methods

Neutrons—having no charge—are difficult to detect directly, but instead the detection of neutrons is done by observing the secondary effects neutrons have when they collide with matter [30]. Neutrons can be broadly classified as thermal neutrons or fast neutrons. Thermal neutrons are neutrons which are traveling at or near speeds which are in thermal equilibrium with the surrounding matter, while fast neutrons may have much more energy.

When a neutron interacts with matter, it may either undergo scattering or absorption. In scattering the neutron "bounces" off the atom, losing some energy to the recoil of the atom and thus ending with a slower speed than it started, assuming the neutron was a fast neutron before the collision. If the neutron is absorbed, the impacted atom, which has now gained a neutron, may be unstable and may undergo a nuclear reaction possibly releasing other types of radiation which can then be detected.

The probability that the neutron will interact with matter is called the neutron cross-section and is measured in barns. The neutron cross-section is highly dependent on the energy of the neutron. This also varies greatly with different elements. There is no convenient way to calculate the neutron cross-section of an element and it is determined empirically.

Thermal or near-thermal neutrons in the range of 10 keV-0.025 eV are much easier to detect than fast neutrons due to their much larger neutron cross section. Many times, to detect fast neutrons, they are first thermalized by passing them through a material with much hydrogen in it such as paraffin. The fast neutrons scatter off the light hydrogen nuclei and lose energy with each scattering interaction. The neutrons lose more energy through recoil when scattering off light nuclei such as hydrogen than they do heavy nuclei because they impart more energy to a light nuclei when bouncing off it than to a heavy nuclei.

Neutrons are generally detected either by detecting the recoil of light atoms when a neutron strikes them, or by relying on neutron absorption and the nuclear decay (usually fission) of the resulting unstable nuclei. This decay releases much energy which is present in released gamma rays and/or high kinetic energy of the decay products.

In ^{3}He and BF_3 tubes, the incoming neutron interacts with the ^{3}He or ^{10}B creating charged particles which ionize other gas particles. ^{3}He tubes rely on scattering while BF_3 tubes rely on absorption. These ions are collected, potentially after being multiplied by an avalanche factor, to produce a charge pulse. These detectors have high efficiency at thermal neutrons, but poor efficiency with fast neutrons.

^{4}He or CH_4 detectors can be used to detect fast neutrons with greater efficiency. These detectors rely on the recoil of light nuclei to ionize the gas. Fast neutron detectors have a much lower efficiency of around 1%.

A fission chamber has a lower efficiency than the ^{3}He detector, but is highly gamma-blind. In this detector, incoming neutrons interact with a thin layer of fissionable material such as ^{235}U which is coated on the inner surface of the tube. The fission creates charged particles which enter the gas tube and ionize the gas.

Another class of neutron detectors are scintillator based detectors, where the neutron interacts with atoms through elastic scattering, which produces photons. These photons are then amplified by a photomultiplier tube (PMT) to create a charge pulse. These are fast detectors with good efficiency, however they are also sensitive to gamma rays.

A solid-state detector works by using a neutron absorbing layer, such as ^{10}B, in conjunction with a diode. The fission of the boron produces energetic particles which go into the diode and produces electron-hole pairs and produce a charge pulse. The diode may be either a standard silicon diode, or may itself be made out of boron to improve efficiency.

A short summary of some properties of some neutron detectors is shown in Table 2.1.

2.3.2 Gamma Ray Detection Methods

Gamma rays are high-energy electromagnetic radiation released from nuclear decay. They can be thought of as charged particles, so they interact with matter more readily than neutrons. Gamma detection generally involves a scintillation based detector where the gamma

Table 2.1 – Detection Technologies

Technology	**Detection Probability**			**Power**
	Thermal Neutron	**Fast Neutron**	**γ-ray**	
BF_3 Tubes	high	low	moderate	moderate
^{3}He Tubes	very high	low	low	moderate
^{4}He or CH_4	low	high	moderate	moderate
Fission Chamber	low	low	very low	moderate
Plastic Scintillator	high	very high	very high	high
Solid-State	high	low	moderate	low

ray excites a number of atoms within a crystal. The atoms then fall back to their not excited state producing photons. The photons are then measured with a photomultiplier tube or other photon sensitive device. This detection architecture is shown in Figure 2.1

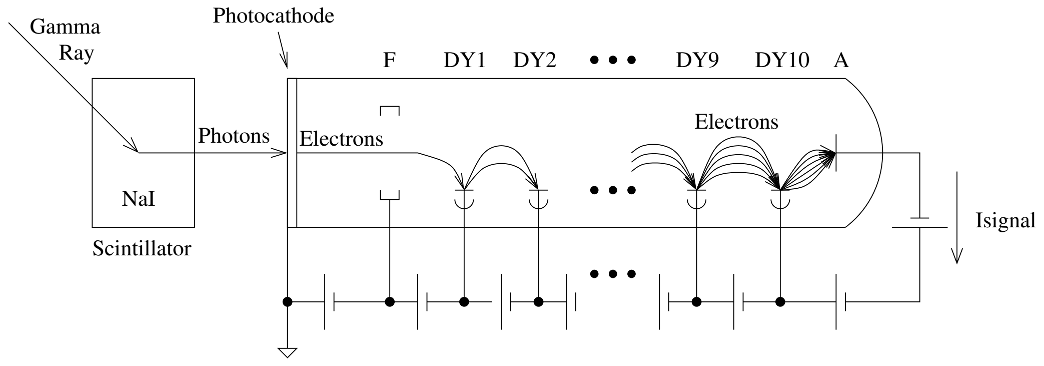


Figure 2.1 – Scintillation-based gamma detection

The figure shows an incoming gamma ray interacting with a scintillator which produces a burst of photons. The photons then are converted to photoelectrons at the photocathode of the PMT. These electrons are amplified as they pass through the dynode stages of the PMT to create a large charge pulse on the output.

Gamma rays may interact with matter in three ways, or it may not interact at all. The probability that the gamma will not interact at all, but will instead travel through the material unchanged increases as the gamma ray energy increases.

The gamma ray may undergo Compton scattering where the gamma ray interacts with an electron and imparts some energy to the electron, but the gamma ray continues on at a lower energy. This interaction occurs with weakly bonded electrons, and as a result the probability of this interaction is relatively constant with respect to atomic number and does not vary much with respect to gamma energy.

Secondly, the gamma ray may be fully absorbed via photoelectric absorption and all the energy transferred to a single electron. This is the most important interaction for detection. This interaction occurs with tightly bonded electrons, and as a result the probability of interaction is greater with higher atomic number.

Finally, if the gamma energy is over 1.022 MeV, it may undergo pair production where the gamma ray produces a positron and electron pair. This production requires at least 1.022 MeV, so this cannot occur at lower gamma energies. If the incident gamma ray has more energy, it will be divided between the electron and positron as kinetic energy. The positron and electron are quickly slowed, and the positron combines with an electron in an annihilation process. This releases two 511 keV gamma rays, which may interact with the detector. If both gamma rays are fully absorbed in the detector, a full energy photopeak is realized, but if one escapes, a peak which is 511 keV less than the full energy photopeak will be created. If both escape, the peak will be 1.022 MeV less than the full photopeak. In practice, this interaction is not important as it is rare that the gamma rays of interest have energies greater than 1.022 MeV.

The photo-sensitive device, in this case a PMT, can also be a photodiode if the photon production of the scintillator is enough. The photodiode is much more noisy than a PMT, but it is much lower power.

Other detectors exist such as avalanche diodes which are simple and low power, however they have poor sensitivity and no spectral resolution.

2.3.3 Charge Sensitive Amplifier Design

Most radiation detection sensors require a charge sensitive front end. This usually consists of a leaky integrator with optional pulse shaping circuits as well as an ADC and digital back end as shown in Figure 2.2

The leaky integrator produces a pulse in response to an input charge. The time constant of the integrator must be tuned so it does not attenuate the signal from the detector. If the detector produces the charge over a long period of time, the time constant must be increased otherwise most of the charge will be lost through the resistor. The pulse shaping circuits are present for two reasons, one to decrease the noise bandwidth to improve the noise per-

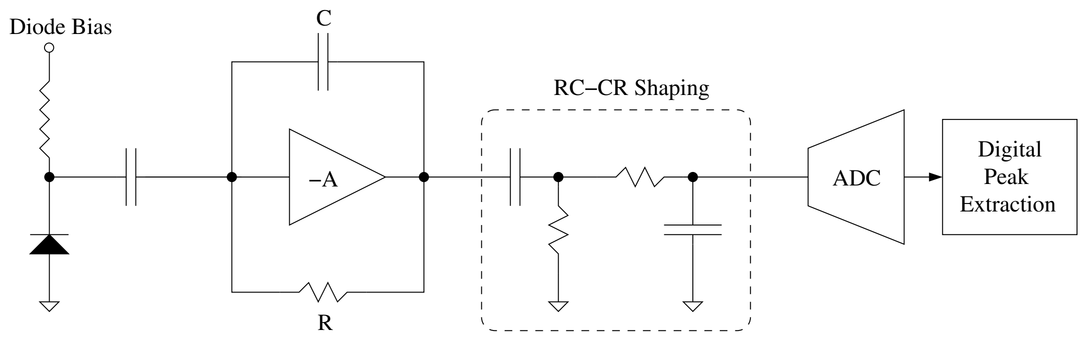


Figure 2.2 – Conventional charge sensitive signal chain

formance, and the second as an anti-aliasing filter for the ADC. The exact configuration of the charge shaping circuits varies greatly, with the circuit shown only a representative example. In most practical circuits the shaping circuits are higher-order, and include active components. The time constant of the pulse shaper should also be tuned to match the detector characteristics. Finally conventionally there is an ADC sampling at over the Nyquist rate and a digital processing section to extract the charge pulse amplitudes.

The problem with this architecture is the power consumption of the Nyquist-rate ADC, the power of the digital processing, and the potentially high power charge amplifier. The ADC power consumption can be dramatically reduced by extracting the peak amplitude in the analog domain and using one ADC conversion just on the peak height, while the CSA power can be reduced with creative analog circuits. Using a peak detection in the analog domain means that the charge shaping circuits are not as necessary, because there is no Nyquist-rate ADC and therefore no aliasing. The limiting of the noise bandwidth is not that important when the input signal is larger as in this case.

Many charge sensitive amplifiers are based on a folded-cascode front end which is shown in Figure 2.3. Present designs in the literature consume between 150 μW and 12 mW per channel [31–41]. A much lower power consumption is desired for extended operation from a small battery. If the bias currents in the conventional designs are scaled to reduce power consumption, a limit is reached when the bias current becomes comparable to the peak current produced by a given particle detector. At this point, slewing occurs during a particle detection event. In a standard folded cascode CSA, the positive-going slew rate is limited by the bias current and the feedback capacitor. This places a lower limit on the bias current scaling which can be performed with a conventional CSA. If the CSA goes into a slewing mode, some charge may be lost through the diode biasing circuits. This leads to a requirement for a CSA circuit which allows the peak current from the CSA to exceed the

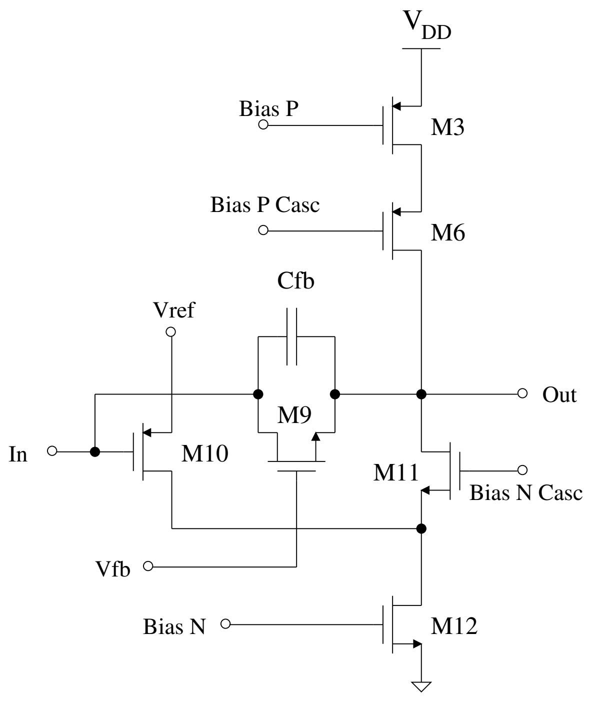


Figure 2.3 – Conventional folded cascode CSA

bias current. The presented design meets this requirement and allows a very low power consumption.

2.4 Radio

The radio technology used in a wireless sensor network dramatically affects the overall system power consumption of the WSN. This is because the radio power consumption is a large percentage of the total network power consumption. Many different radio technologies can be used in wireless sensor networks.

Narrowband communication is the most common type of radio. However, numerous different modulation forms may be used, each of which results in different tradeoffs among power, bandwidth, data-rate, required SNR, interference resilience, and circuit complexity.

UWB communication is defined by the FCC as a signal which occupies a bandwidth of over 500 MHz, or a bandwidth of greater than 20% of the center frequency. This type of radio can theoretically provide a much higher data-rate with a much lower power consumption than conventional narrowband communication by trading off occupied bandwidth for other desirable traits such as data-rate or transmit power. The drawbacks of this type of radio include short range and sensitivity to interference.

2.4.1 UWB Communication

UWB communication promises great benefits, but it is not without its challenges. First, and most importantly, the maximum allowable transmit power is limited to a value such that communication may only take place over a relatively short distance. Secondly, narrowband interference can cause great problems for the UWB receiver. The narrowband interferer can be much more powerful than the desired UWB signal, and may be in the frequency band of the UWB signal. It is theoretically possible to notch-filter the signal to eliminate the narrowband interferer while still keeping most of the UWB signal power. This will also eliminate some of the desired UWB signal, but there should be enough UWB signal remaining at other frequencies to decode the UWB signal. In practice, the frequency of the narrowband interferer is rarely known at design time, so to use this approach, an adaptive filter must be used. However, some antenna and receiver designs have fixed-frequency notch filters to filter out likely interference-causing bands such as the 2.4 GHz ISM band.

Another more common approach is to simply choose a frequency range which avoids a number of the anticipated narrowband interferer frequencies. For example, one may choose the UWB frequency range to avoid the cluttered 2.4 GHz ISM band as well as the 2.3, 2.5, and 3.5 GHz Wi-max frequencies.

2.4.1.1 MB-OFDM vs. Impulse Radio

Two broad categories of UWB modulation schemes are commonly seen, multi-band orthogonal frequency division modulation (MB-OFDM) and impulse radio. A third category exits, Frequency division UWB (FM-UWB), which uses frequency modulation which can be quite simple, but supports very low data rates of around 50 kbps [42].

MB-OFDM is a straightforward extension of the conventional OFDM modulation format used in conventional narrowband radios such as IEEE 802.11.4. In this type of modulation the data is encoded with a slow symbol rate on a large number of carriers. This has many advantages including simple channel equalization, simple intersymbol interference mitigation, high spectral efficiency, and simple implementation using the FFT at the receiver and inverse FFT at the transmitter. To adapt this to UWB, it is simply extended to work with many more sub-carriers over a wider frequency range. This type of architecture promises the highest data rates but is also the most complex to implement and requires the most amount of power. Due to the complexity, this method is not being pursued for wireless sensor networks and is instead finding application in high-data-rate cable-replacement standards such as wireless-USB.

Impulse radio on the other hand is a radically different type of radio where very short duration signals are sent. The short duration of the signals means they have a very wide bandwidth. The information may be encoded in numerous ways including the polarity of the transmitted signal, the presence or absence of the signal, or the position of the signal. For some modulation schemes, it may be beneficial to send a reference pulse before the encoded pulse to aid in decoding.

The shape of the transmitted UWB pulse determines the spectral bandwidth. Short moncycles do not satisfy the FCC bandwidth requirement, but higher-order derivatives of a Gaussian do. The 5th derivative of a Gaussian pulse satisfies the FCC mask for indoor requirements while the 7th derivative works for outdoor environments [43]. Some work in the literature on UWB transmitters focuses on producing a pulse with the correct pulse shape directly from the active electronics [44], while other work uses a simple digital pulse and then filters the pulse to fit the spectrum mask of the FCC [45, 46]. Both methods perform well, however the latter method is somewhat simpler and lower power and will be used in this design. Furthermore, the thrust of this design is not to design an FCC compliant transceiver, but is instead to design a novel transceiver whose architecture could be used in different frequency bands with different spectral masks.

Impulse radio receivers fall into two broad categories, coherent and non-coherent designs. Coherent designs are a more conventional design where the incoming pulse is correlated with a basis function to produce a signal-space representation and a decision is made. Non-coherent designs simply utilize the energy of the incoming signal to decide on the presence or absence of a pulse. The coherent design gives higher performance but with many implementation difficulties.

2.4.1.2 Coherent Designs

In a coherent design, the incoming signal is correlated with a basis function to produce a signal-space representation of the signal. This is the optimal receiver architecture, however it has some implementation difficulties. This is usually implemented by correlating the incoming signal to a template signal. The difficulty is in generating the reference signal. Two major problems are encountered, one is the timing of the template signal must be very good, and secondly the shape of the optimal reference signal is dependent upon channel conditions and therefore unknown at design time. The precise timing requirements stipulate the use of very high-power low-jitter oscillators which can position the signal to within pico-seconds, furthermore it is difficult to determine the exact position of the signal given the corrupting noise. Modifying the template pulse from its optimal shape can help decrease the jitter sensitivity [47], but at the expense of performance. Furthermore, over-sampling the data and using multiple antennas can reduce performance penalty associated with jitter [48].

In [49] an apparently simple and low-power coherent design is presented, however it does not address the timing constraints, and neither does it have any way to find the start of a packet or stay synchronized to the packet once found. This is true of many of the designs presented in the literature, with very few implementing a complete system.

One way of alleviating both the unknown template pulse shape and the jitter problem is to use a transmitted reference pulse before the UWB symbol as the template signal. In the simplest form of this receiver, the signal is delayed enough so the reference pulse can be correlated with the modulated pulse. This solves the problem of a channel-dependent template pulse shape and most timing problems, however, this architecture suffers from poor noise performance because of the noisy reference signal. This can be improved with template pulse averaging where the reference pulse shape is averaged over many received symbols, but this generally always requires a complex digital implementation.

2.4.1.3 Non-Coherent Designs

In non-coherent designs, the RF signal is squared to find the energy of the signal and the energy is integrated to find the presence or absence of a pulse. This is a much simpler architecture to implement but is less optimal than a coherent design.

For best performance, the signal should first be passed through a matched filter which selects only the frequency region of interest [50].

Many UWB non-coherent receiver architectures have been proposed through the years. They fall into two broad categories, depending on if they do self-mixing and process the energy of the signal or direct conversion where the signal is multiplied by a quadrature LO. Within each category, they can be further divided as performing the baseband functions in the analog or digital domain. Doing a direct digitization of the either self-mixed or quadrature components and then using a fully digital baseband implementation can require much power due to the high-speed ADCs required. Some authors propose to use 1-bit

ADCs [51], but this degrades performance in the presence of an interferer. The digital type of architecture can do very quick signal acquisition and processing in the digital domain. Another type of receiver does the correlation in the analog domain and uses a much lower speed ADC [52], usually at the symbol rate instead of the Nyquist rate. But with this architecture, it can be difficult to quickly perform signal acquisition and tracking as well as more difficult channel compensation. The slow signal acquisition is because of the small number of analog domain correlator(s) so a highly parallel architecture cannot be implemented as can be done with the digital architecture.

In [51] a 1-bit ADC is used to directly quantize the energy envelope signal and do processing in the digital domain, however in the presence of a narrowband interferer, this architecture suffers greatly. In [53] a bank of integrators in the analog domain is used before quantization which allows for high speed signal acquisition, however it still requires a fast ADC (or many slower ADCs) to convert each of the correlator's output. In [54] the signal is directly quantized at 1.2 GHz at the front end and all processing is done in the digital domain. In [55] a direct-conversion architecture is used and the baseband signal is then directly quantized. These all require considerable power consumption for the ADC and digital sections with high frequency clocks to attain the necessary time resolution.

In [52] a direct-conversion architecture is used with the correlation taking place in the analog domain. This makes for a simple and low-power receiver, however due to the fact there is only one analog domain correlator per I and Q channel, signal acquisition is slow and unreliable and symbol synchronization is difficult. [56] presents a front-end with a similar architecture, but the authors only present the front end, up through the mixer and energy detection and do not present a complete system.

In [57] a circuit is presented to do the bit detection in the analog domain, but the other functions of symbol synchronization and packet detection are not addressed. A similar approach is used in [58] with the ability to do packet detection, but no ability to stay synchronized once a packet has been detected.

In all the present papers, a tradeoff exists between doing the quantization early in the signal chain and then using highly parallel digital circuits to do the signal acquisition quickly, versus doing the correlation in the analog domain, and then they are not able to use a parallel signal acquisition architecture to quickly acquire the signal. Furthremore, both receivers can suffer from poor sensitivity to packets but for different reasons. For the directly digitizing receivers, the low number of bits in the ADC can cause this problem for certain channel conditions, and for the analog type, the serial nature of the signal acquisition means that a robust signal acquisition algorithm cannot be implemented. Furthermore, tracking the incoming data stream can be difficult with the architecture where the correlation is done in the analog domain.

This dissertation proposes an architecture with highly parallel analog correlators which allow both fast and reliable signal acquisition while only requiring low-rate one-bit ADCs. The large numbers of parallel analog correlators do not greatly affect power consumption because they operate at the lower symbol rate and are simple and low power.

2.4.1.4 Coding

Very few if any impulse-radio designs in the literature present a code on the UWB link. A pulse-repetition code is used in some designs [59] to assist in multiple access schemes. In OFDM schemes some type of error-control coding is usually included such as turbo codes [60]. Others use a similar architecture where each bit is mapped to a PN code [61], however this has the same performance as a repetition code. The pulse repetition code and similar is used as a simple way to increase the E_b/N_o of each bit because it cannot be increased by increasing the transmitted power (due to FCC constraints). In some implementations, this pulse repetition code is done in the analog domain by not making a hard decision on the bit until all the pulses have been integrated. This has its problems when dealing with UWB and other interference which produces impulse-like signals, as a single interference pulse can corrupt an entire bit if a hard decision is not made after each pulse. A convolutional code with hard decision after each bit can produce much better results than a pulse repetition code, so this is used in the presented architecture.

2.4.1.5 Design Considerations

Power consumption and performance were the two driving factors with this design. Other facets of a transceiver design were only treated in a cursory manner because they were out of the scope of this dissertation. For example, the designed transceiver is not FCC compliant because it operates over the range of 500 MHz to 2.5 GHz. This is not compliant with either the 3-10 GHz range or sub-960 MHz range specified by the FCC. The lower-frequency range was chosen to implement this proof of concept design because to attain the higher frequency range, a newer process with a smaller feature size is generally used. Due to budget constraints, this was not possible, so a lower frequency range was settled upon. Nonetheless, the design concepts are directly applicable to high-frequency designs. The low-frequency band was not used due to antenna size constraints.

2.4.2 Narrowband Communication

There are a multitude of narrowband modulation schemes and transceiver architectures in the literature. The modulation formats can be broadly characterized as amplitude modulation, phase modulation, frequency modulation, or a combination such as quadrature amplitude modulation (QAM). Amplitude modulation by itself is rarely used due to its poor performance and implementation issues when using M-ary modulation.

Phase modulation has the advantage of using a constant-envelope signal allowing a higher-efficiency power amplifier to be used at the transmitter, and for small constellation sizes, it is quite near optimal. It suffers from some receiver implementation issues such as the requirement for an absolute phase reference. However, for a small performance penalty, differential PSK can be used which does not require an absolute phase reference.

Frequency modulation is also a constant-envelope modulation. This has worse performance than PSK and QAM, but some simple receiver architectures exist which make this attractive for some wireless sensor networks.

QAM is a combination of amplitude and phase modulation. It has good performance but is complex and both the transmitter and receiver require much power.

The IEEE 802.15.4 standard uses a variant of 4-PSK called offset quadrature phase-shift keying where the I and Q components are shifted with respect to each other. This decreases the maximum phase discontinuity which can occur at symbol transitions from 180^{o} to 90^{o}. The smaller phase discontinuity produces better spectral properties and decreases the peak amplitude of the transmitted waveform after it is passed through a low-pass filter. Furthermore the I and Q components are shaped as half-sine waves which improves the spectral properties and is equivalent to minimum-shift keying (MSK) modulation. IEEE 802.15.4 also uses direct-sequence-spread-spectrum (DSSS) which spreads the spectrum of the transmitted data and makes it more resistant to noise and other users in the channel.

For this application, a very low power and simple receiver is desired. For this reason a simple on-off keying modulation scheme is used which has simple transmit and receive circuitry.

2.5 Single-Chip Solution

Putting the complete sensor network on a single chip as a system on a chip (SoC) has many advantages over a conventional multi-chip solution. These advantages include cost, size, throughput, and power. However, a small penalty is paid in terms of flexibility.

Integrating all the parts of the wireless sensor network on a single chip can allow lower costs with volume production. This lower cost is mainly due to the decreased packaging, assembly, and overhead costs. Though it is slightly cheaper to manufacture two chips of half the area than it is to manufacture one large chip with twice the area. So at a manufacturing level, it is cheaper to make two chips, one being the processor and the other being the RF chip. However, the manufacturing difference should be small because of the small die size. However, the packaging, board assembly, logistics, and profit costs will be much greater for the multi-chip solution than for the single-chip solution. This allows for a smaller form factor and lower cost sensor network node.

When using a multi-chip solution, the most common division is to put the processor and radio transceiver on separate chips. However, much data must flow over this division, so this causes inefficiencies in the sensor network node. These manifest themselves as two problems. First, the responsiveness and throughput of the radio link is decreased. This is due to the delay incurred while a packet is transferred over the serial bus between the processor and transceiver. For example, after a packet is received, it first must interrupt the processor, the processor must read the packet—one byte at a time over the serial link—and then the processor must compose a response (possibly an acknowledge) which then must be sent over the serial link to the transmitter before the transmitter can begin sending a response. This affects the maximum utilization of the channel as well as causing excess power to be dissipated in both the local node as well as the remote node.

The remote node dissipates extra power because while the local node is transferring the packet to and from the processor, the remote node has turned on its receiver and is waiting for an acknowledge. The extra delay causes the the receiver to be on for longer than necessary, wasting power. The local node dissipates more power because the processor must individually transfer each of the bytes from the serial interface to the internal RAM. Practically this is achieved by having the serial module interrupt the processor after every byte is received/transmitted at which time the processor either stores the received byte or transmits the next one. These continual interruptions cause extra power to be dissipated.

For a single-chip solution, when the packet is done being received it is immediately available in RAM for the processor to use. For the transmit interface, the processor simply puts the packet in a special location in RAM and the hardware can automatically send it. This also greatly simplifies driver software development.

One final power saving feature of a single-chip solution is the fact that transmitting a bit across the board consumes much more power than transmitting a bit across a chip. The board is much more highly capacitive and therefore wastes much more power when sending bits across it.

2.6 Chip Generations

For this dissertation, numerous chip designs were fabricated, with each chip adding features and testing new designs. The later parts of the dissertation will focus on the final, fourth chip, however references to earlier chips and earlier designs are also included. Therefore, it is beneficial to introduce the chip generations and their associated functionality.

- **First chip:** The first Austriamicrosystems 0.35 μm chip contained the processor, serial peripherals, timers, manually designed SRAM and ROM, and the low-power front end. This was a test vehicle mainly for the radiation detection front end, while also accruing some some valuable experience in digital design. No working RF part was present on this chip, though there was a non-working UWB transmitter.

- **Second chip:** The second and all later chips use a 0.18 μm process. The second chip was somewhat rushed because it was a free fabrication run with a hard deadline attached to it. Therefore not everything was included in this one which was included in the previous one, namely, the CSA and front end were not included. However the UWB transmitter and narrowband receiver were included. Furthermore the digital design experience gained from designing the first chip allowed a much better digital section to be included. The smaller process allowed much more memory to be included. This is the first chip which could perform wireless (narrowband) communication.

- **Third chip:** The third chip was a vast increase in complexity and functionality over any previous chip. At this point the chip transformed from being part of a system but still requiring much support circuitry to begin the complete system with only minimal mostly passive support components necessary.

For the third chip some incremental optimizations were performed on the CPU to increase the processor clock speed, and more peripherals were added. The low-power front ends were included, but with much lower power consumption than on the AMS 0.35 μm chip. A clock generator with a crystal oscillator and digitally-controlled oscillator was added to make the microcontroller a complete system. The narrowband receiver was re-designed to its final form, and the UWB receiver was implemented, though it had an oscillation in the front end which did not allow it to work. Finally the DVS enabled buck power supplies were added.

- **Fourth chip:** The fourth and final chip is the culmination of all previous chips. In this chip the digital processor was upgraded to the MSP430X architecture to support a larger memory of 96 kB. Many bugs in the previous chip were fixed, while some new ones were added! The main functional addition over the third chip was the complete integration of the narrowband transmitter. On previous chips the RF section of the narrowband transceiver was not included on the chip, but it was included on this version. The UWB receiver was also improved, adding error control coding on the link.

Test boards were developed to test each chip. Functional tests were performed on the individual parts of the chips. Software was developed as necessary to test each part of the chip individually. More details of the individual chips and the test boards can be found in chapter 8.

2.7 Conclusion

This chapter presented the motivations behind many of the design choices which are used in the wireless sensor network node and presented the main contributions of this work. Further background information about the individual sections of the chip will be introduced in later chapters as needed.

Part II

Design

Chapter 3

Charge Sensitive Amplifier Front End

This chapter details the sensor interface. Figure 3.1 shows the section of the overall node architecture this chapter describes. The sensor interface is designed to work with radiation detectors, however the radiation detector specific front end can be bypassed which allows the front end to work as a general purpose ADC.

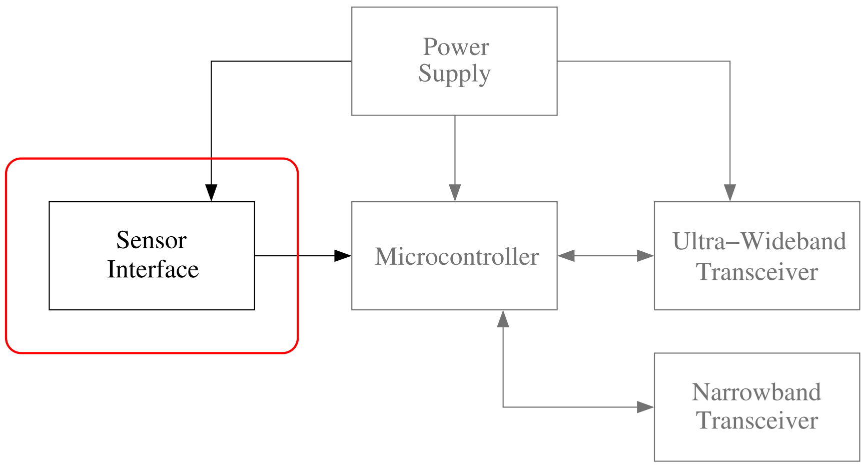


Figure 3.1 – Sensor interface section enclosed in red box.

Many detectors, particularly radiation detectors, produce small charge pulses as their output. Generally we are interested in measuring the size of the charge produced. However, different detectors can produce varying amounts of charge and over different time scales. Therefore a programmable front end which can accommodate differing charge amplitudes with different time durations is desired.

The front end design integrates four independent analog channels designed to interface to external radiation detectors. The intended sensor is a solid-state detection diode; though due to the programmable nature of the front end other sensors may be interfaced as well. These detector diodes produce a small pulse of current when excited by incoming radiation. The designed front end amplifies the current pulse, measures the amplitude and reports this to the microcontroller. The signal chain of a single channel is shown in Figure 3.2.

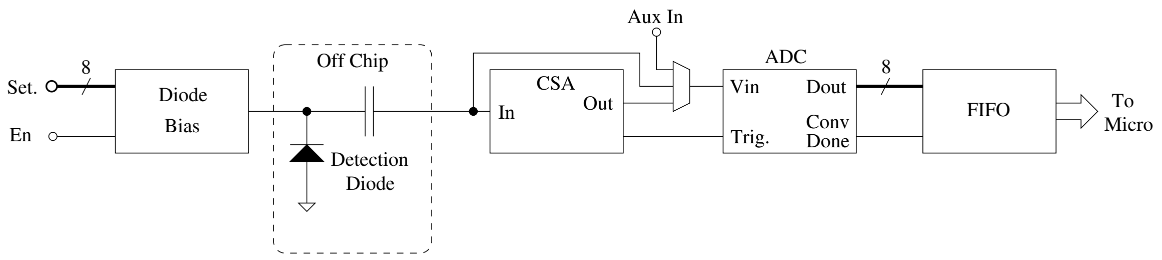


Figure 3.2 – Detection signal chain showing the diode bias, CSA, ADC and microcontroller interface.

An adjustable DAC within the diode bias circuit applies a reverse bias voltage to the diode, but presents a high impedance to the signal currents. The signal currents are capacitively coupled to the CSA which amplifies the signal and finds the peak value. The ADC produces a digital codeword from the peak value and the FIFO buffer acts as the microcontroller interface.

If the CSA is not necessary, the channel's ADC can sample an external signal so the ADC can be used as a general purpose ADC for other external sensors. In this case the ADC is triggered by system software.

The rest of this chapter is organized as follows: Section 3.1 presents the design of the diode biasing circuitry which produces the reverse bias for the detection diode. Section 3.2 discusses the design of the charge-sensitive amplifier core as well as the peak detector. Section 3.3 presents the design and implementation of the event-driven 8-bit ADC while section 3.4 presents the microcontroller interface to the ADC. Section 3.5 presents the design of the DACs which are used in the front end for maximum configurability. Finally section 3.6 concludes the chapter.

3.1 Diode-Biasing DAC

The biasing of the diode can greatly affect the detection performance. In particular, the diode biasing system should ideally place a constant reverse bias on the diode, not absorb any signal current, and not add much electronic noise. In a small-signal model of the front end, the signal current of the diode is in parallel with the input impedance of the amplifier, the impedance of the diode biasing circuitry, and the diode capacitance. The signal which goes through the input impedance of the amplifier is what is detected, therefore it is best to maximize the impedance of the bias circuitry and the diode capacitance and minimize the input impedance of the amplifier to assure that the maximum amount of current flows through the amplifier and the minimum is lost in the biasing circuits.

A standard way of biasing a diode is using a voltage source connected in series with a resistor as shown in Figure 3.3. In this configuration the diode biasing impedance is just the bias resistor value, R_b. However, if the resistor is made too large, two problems occur. First, the stability of the applied reverse bias suffers, and secondly the voltage source which the resistor is connected to must become large. These two effects are caused by the reverse leakage current of the diode and its variability with environmental factors. The value of the reverse bias leakage current sets the voltage across the bias resistor and thus the reverse bias voltage of the diode as $V_b - I_{leak}R_b$. As this reverse leakage current varies with temperature and—in some diodes—light, the reverse bias diode voltage changes.

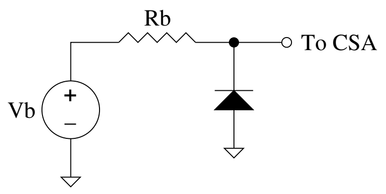


Figure 3.3 – Standard diode biasing scheme

What is desired is a circuit which will present a low impedance to the DC reverse leakage currents but a high-impedance to signal currents, all while introducing minimal noise. The chip includes four independent channels which do just that. Each channel consists of an 8-bit DAC and a buffer. There are two types of buffers implemented, and application software can select which buffer to enable. A block diagram of a channel is shown in Figure 3.4.

The reverse bias voltage on the diode can be set via an 8-bit DAC. The DAC is a very low power DAC. The output of the DAC is buffered either by a conventional op-amp to

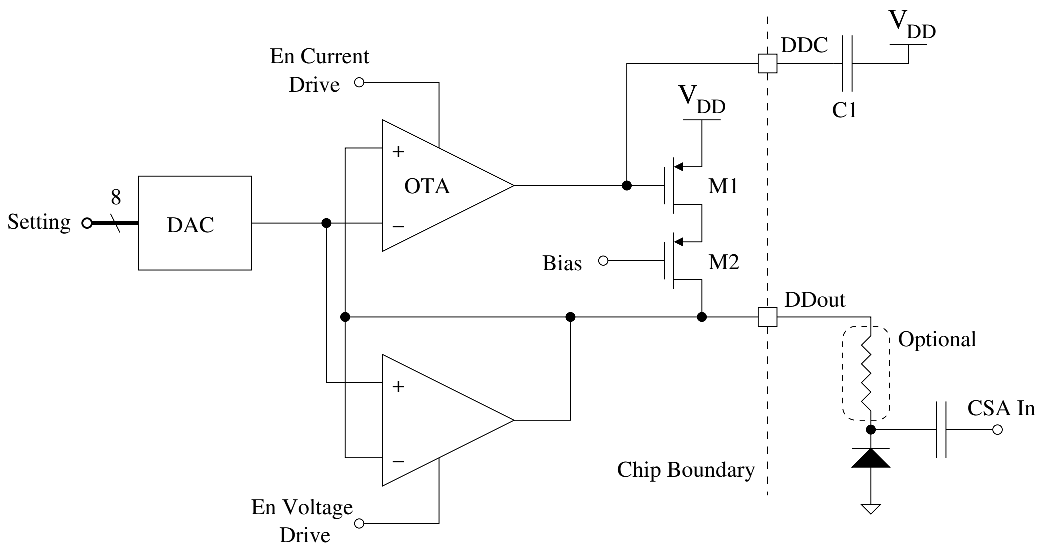


Figure 3.4 – Diode DAC architecture

produce a constant voltage output, or a modified op-amp to produce an output which has a high impedance at the signal frequencies but a low impedance at DC. The current output is beneficial as it decreases the amount of the signal current absorbed in the bias network and therefore forces the entire signal to flow through the CSA and associated circuitry.

The voltage drive op-amp, which is the lower op-amp in Figure 3.4, is a relatively conventional op-amp. The output is stable over a very wide range of capacitive loading, as many diodes will look like large capacitive loads. The maximum output voltage is about 1 V less than the supply voltage. If this driver is used, the external resistor must be used to increase the impedance the signal current sees looking back into the bias network. This is the conventional way of biasing the diode. The open-loop gain of the op-amp is 100 dB, and the 3 dB frequency is 10 Hz. The unity gain frequency is 1.4 MHz. When disabled, the output is in high impedance so it does not interfere with the other op-amp.

When using the current drive configuration, the output should be directly connected to the diode without a series resistor. In this configuration, the bias circuits have an impedance of larger than 1 MΩ at signal frequencies, so very little signal is lost to the bias circuits. The circuit can be unstable under certain operating conditions. There are two low frequency poles. One at the output, where the large diode capacitance in parallel with the large output impedance leads to a very low frequency pole, and one at the compensation point. The diode compensation point pole should be at a frequency a couple decades lower than than the output pole. With the recommended 10 nF capacitor as C1, the output pole is at 1.5 mHz. The unity gain frequency of the op-amp is about 1 Hz. The pole at the output should stay beyond 10 Hz to maintain stability. If this is not met, C1 can be increased to further decrease that pole. C1 should be a capacitor type with low leakage (< 2 nA). The start-up time of the circuit may be large. With a 10 nF capacitor as C1, it will require about 500 ms to start.

The current drive configuration can supply up to 20 μA of diode current while still maintaining the same output voltage and high impedance bias network. This driver configuration allows for much better performance for many diodes under a variety of environmental conditions.

The open-loop transfer function of the diode drive circuitry has two poles and no zeros in the low frequencies of interest. One pole, ω_1, is the dominant pole which is created by C1 and the output impedance of the OTA driving that node. The other pole, ω_2, is created by the substantial detection diode capacitance and the output resistance of the current source in parallel with the small signal impedance of the diode. The closed-loop expression for the output impedance as seen by the diode current is given by:

$$R_{oc}(s) = \frac{R_o(s-\omega_1)(s-\omega_2)}{A_{DC} + (s-\omega_1)(s-\omega_2)} \tag{3.1}$$

where R_o is the open-loop output impedance which is in the 10 MΩ range, and A_{DC} is the DC open-loop voltage gain which is about 100 dB. Near DC, the output impedance is approximately $R_o\omega_1\omega_2/(A_{DC} + \omega_1\omega_2)$, which is small, particularly when ω_1 and ω_2 are small. At signal frequencies, which are above both pole frequencies, the output impedance approaches R_o.

The two poles in the loop may create a stability problem. Creating a zero to cancel the detection-diode-created low-frequency pole is not optimal or practical as the location of the pole is highly diode dependent as well as being sensitive to environmental conditions. The zero may also decrease the output impedance at signal frequencies. Therefore, the method chosen to overcome this is a simple brute-force lowering of the added dominant pole until the loop is stabilized. This can be achieved by using an OTA with a very high output impedance, and a large 10-100 nF off-chip compensating capacitance which produces a dominant pole near DC. This size of external capacitor can compensate for a diode-created pole of 100 Hz or larger. This pole corresponds to a diode capacitance of up to 100 pF and a small signal impedance of 16 MΩ. The external capacitor C1 can be increased or decreased depending on the properties of the specific diode. The diode bias circuit operates on 3.3 V to allow a higher reverse bias diode voltage. The noise introduced by this portion of the circuit is negligible. The only noise contributors are M1 and M2 in Figure 3.4, as the rest of the circuit has a very low noise bandwidth.

3.2 Charge Sensitive Amplifier

The charge sensitive amplifier (CSA) is responsible for presenting a low-impedance to the signal current and producing an output swing proportional to the input charge. The CSA circuit is a leaky integrator where the charge is integrated for a short time and then slowly the charge dissipates through the integrator.

3.2.1 CSA Core

The schematic of the presented charge sensitive amplifier is shown in Figure 3.5. It is based on the standard folded cascode charge sensitive amplifier. The DC bias current through

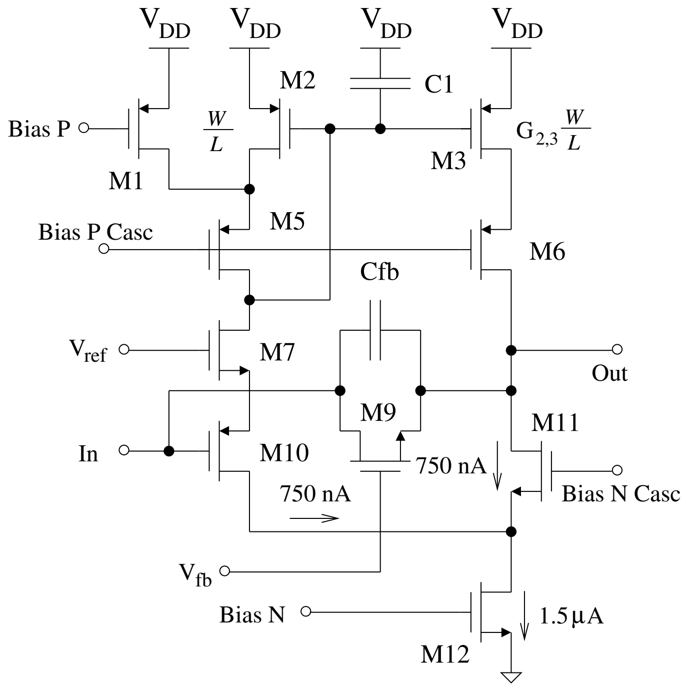


Figure 3.5 – CSA schematic.

M12 is split into currents I_{M10} and I_{M11}. The current through M10 is given by equation (3.2)

$$I_{\mathrm{M10}} = \frac{I_{\mathrm{M12}} + G_{2,3} I_{\mathrm{M1}}}{1 + G_{2,3}} \tag{3.2}$$

where $G_{2,3}$ is the mirror gain from transistor M2 to transistor M3. If $G_{2,3}$ is large, the current through M10 is approximately I_{M1}, with the rest flowing through M11. For the presented design $G_{2,3}$ is 30. The DC operating currents are set at 1.5 μA through M12 and 0.75 μA through both transistors M10 and M11. If M1 were not present, the current through M10 would be $I_{\mathrm{M12}}/(1+G_{2,3})$ which is unacceptably small. Therefore, M1 is present exclusively to increase the DC current through the input transistor M10 and therefore decrease its noise contribution.

The amplifier has a wide output swing capability. The swing can be as large as $V_{DD} - 4V_{D_{sat}}$, where $4V_{D_{sat}}$ is the overdrive voltage of the transistors. The DC output level is controlled by $\mathrm{V_{ref}}$. Though the output of the amplifier has a large swing range, even in a low supply voltage process, the input side of the amplifier has very limited headroom. The biases must be chosen carefully to keep all transistors in saturation. At DC, the most constrained path from power supply to ground consists of three $V_{D_{sat}}$ drops through M2, M5, and M7, then the V_{GS} of M10, and finally two more $V_{D_{sat}}$ drops through M11 and M12. Because of the low bias currents, all transistors are operating at or near the sub-threshold regime, hence the $V_{D_{sat}}$ of each transistor is kept at a minimum.

A particle detection event causes a decrease in the input voltage, and an increase in current through M10. This signal current is mirrored from M2 to M3 and multiplied by the mirroring gain before being applied to the output. Also, as in a conventional folded-cascode design, the signal current is applied to the output via M11. The increase in current through M3 allows the positive going slew rate to be maximized. If a conventional CSA is used where the current through M3 is fixed, the maximum possible slew rate for the conventional CSA is a function of the the bias current through M3 given by $SR = I_{M3}/C_{fb}$. This limits the amount to which the biases can be scaled to achieve low-power operation. In this design, the maximum positive slew rate (SR_{pos}) is given by equation (3.3) if the r_o of the transistors is neglected.

$$SR_{pos} = \frac{(I_{M12} - I_{M1})G_{2,3}}{C_{fb}} \tag{3.3}$$

From this equation, it is clear that using a larger mirroring gain can further increase the maximum slew rate. The open loop gain of the presented design is also improved from a conventional design. Conventionally, the open loop voltage gain is given by $A_{vo} = -g_{m10}((r_{o3,6}^2\, g_{m6})||(r_{o11,12}^2\, g_{m11}))$. The open loop gain of the presented design is given by:

$$A_{vo} = -\frac{g_{m10}\, g_{m7}}{g_{m10}+g_{m7}}(G_{2,3}+1)((r_{o3,6}^2\, g_{m6})||(r_{o11,12}^2\, g_{m11})) \tag{3.4}$$

If it is assumed that g_{m10} and g_{m7} are equal, then the presented design has a higher open loop gain by a factor of $(G_{2,3}+1)/2$. The closed loop charge-to-voltage conversion gain can be approximated as $1/C_{fb}$, where C_{fb} is adjustable in this design to range from 0.2 pF to 15.2 pF in 1 pF increments.

The circuit uses an internal current feedback loop for operation. The gain is created by the mirroring gain from transistor M2 to M3. Without compensation, this loop may become unstable. To correct this, capacitor C1 is added to force a dominant pole response and stabilize the circuit. The circuit was analyzed by carefully breaking the loop at the drain of M6 and injecting an AC current through M11. The output is then measured as the current through M6. This analysis confirms a single dominant pole present at the gate of M3. A graph of the magnitude and phase of the open-loop response is shown in Figure 3.6.

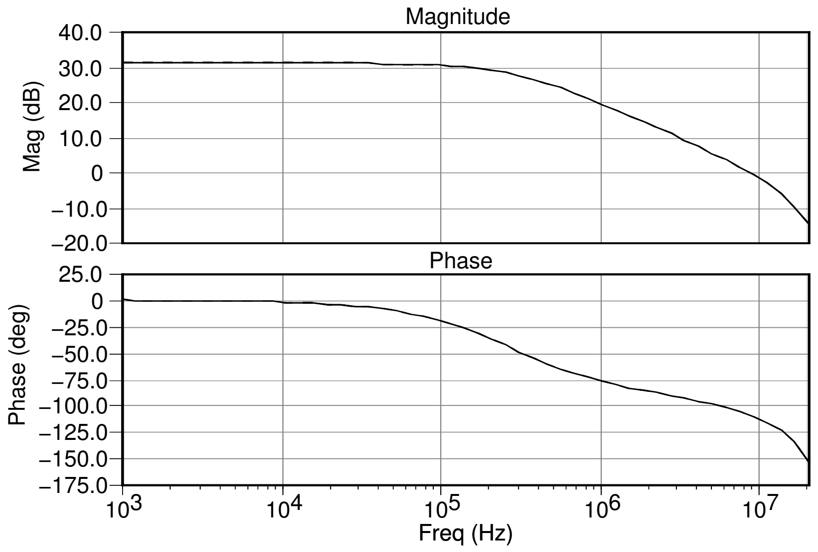


Figure 3.6 – Open-loop AC response of the feedback circuit.

The noise performance of the presented circuit is similar to that of previous work in open literature [31–41]. The squared output noise is given in Figure 3.7. The total integrated output noise, V_n, is 420 μV. The equivalent noise charge (ENC) of the circuit

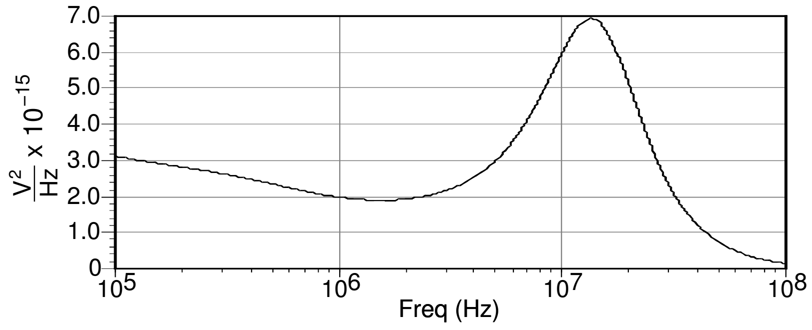


Figure 3.7 – Output noise spectral density in V^2/Hz.

depends on the feedback capacitor. For a setting of C_{fb}=200 fF, ENC equals 525 e$^-$ with the detector disconnected. The ENC increases as the feedback capacitor increases because the output noise voltage is constant, but the circuit gain decreases. The noise in the circuit is not dominated by a particular device. The three largest noise contributors in the circuit are M1, M12 and M7 in that order. M1, though a p-channel device, experiences a large gain to the output which is $G_{2,3}$ times the cascode gain of the output section, whereas M12 experiences the cascode gain of the output section. M7 experiences half the cascode gain to the output and therefore produces less noise.

The negative-going slew rate (SR_{neg}) is limited to a value much less the positive-going slew rate because it does not get amplified by the mirroring gain. The negative-going slew rate affects how fast the amplifier can recover from a particle detection event, which may be a limiting factor in some high-count rate applications. The maximum negative going slew rate is given by:

$$SR_{neg} = I_{M12}/C_{fb} \tag{3.5}$$

Comparing this to the maximum positive slew rate, if $G_{2,3}$ is large, this slew rate can be many times smaller than the positive going slew rate. This constraint sets a maximum recovery speed for the amplifier. However, for low to moderate particle detection rates of up to 100 thousand particles per second, this is not a limiting factor.

Figure 3.8 shows a circuit simulation demonstrating the operation of the CSA for multiple detection events. This simulation shows the operation of the complete front end with the peak detector and ADC interface. The top graph is the output of the CSA before the peak detector, the second graph is the ADC trigger, the third graph is the input to the ADC after the peak detector circuit, and the bottom graph is the current through M3 from Figure 3.5. Not shown, a digital codeword is also generated by the ADC with a conversion time of

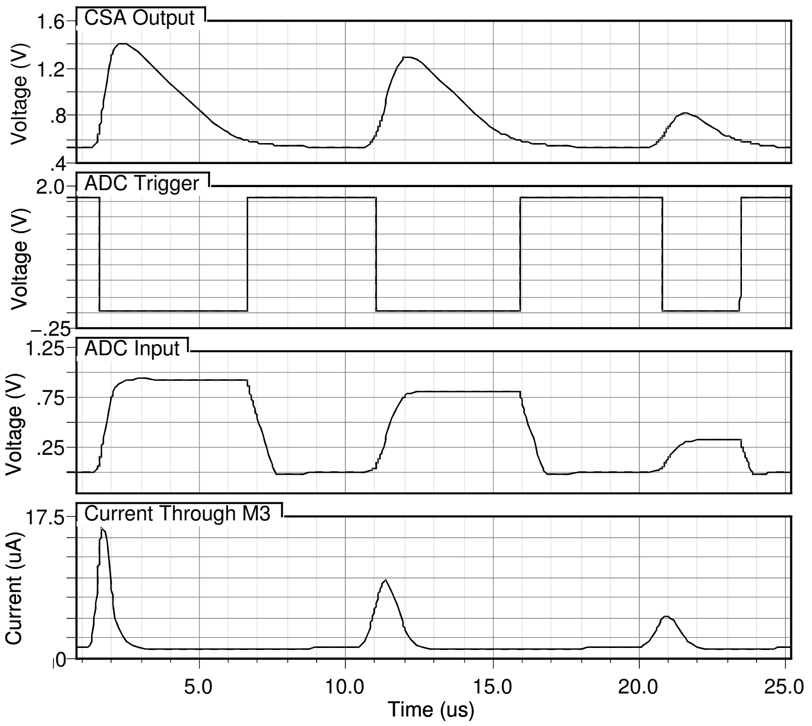


Figure 3.8 – Front end simulation. From top to bottom:
CSA Output, ADC Trigger, ADC Input, Current through M3.

one microsecond after the rising edge of the ADC trigger. Three distinct capture events were modeled as a total of 5.5 pC, 4.8 pC, and 1.8 pC of charge introduced to the front end in 400 ns, with peak currents of 25 μA, 15 μA, and 8 μA, respectively. The feedback capacitor used in the charge-to-voltage conversion is set at 6.2 pF. Without the adaptive biasing scheme, a minimum of 25 μA of DC bias current through M3 would be required. In the presented design, there is 750 nA quiescent current through M3 demonstrating the significant power savings.

For the simulations, the detector diode was modeled as shown in Figure 3.9. V1 and R1 model the external diode biasing circuitry required to apply the proper reverse bias across the diode. C1 models the parasitic capacitance of the diode and wiring, while current source I1 is a pulsed current source modeling the particle detection events which generate a charge pulse in the diode [62]. Capacitor C2 is a coupling capacitor with an arbitrarily large magnitude.

For lab tests, Figure 3.10 shows the circuit used to generate small charge pulses. A sawtooth wave is differentiated to produce small charge pulses with a charge content of *CV*

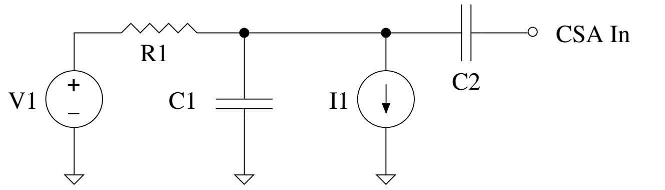


Figure 3.9 – Simulation diode model

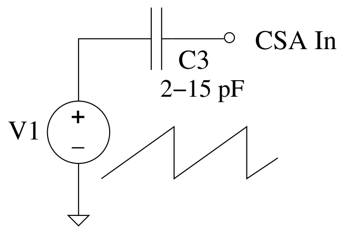


Figure 3.10 – Differentiator circuit

where V is the amplitude of the sawtooth wave.

3.2.2 Peak Detector and ADC Control Functions

A diagram of the CSA and surrounding circuits is shown in Figure 3.11. The overall system consists of a diode biasing circuit, the CSA, a peak detector to extract the peak value, and an ADC. A simplified diode biasing circuit is shown in Figure 3.11. If the charge sensitive amplifier is not required, the input to the ADC is available externally. In this mode, the ADC conversions can be triggered in software.

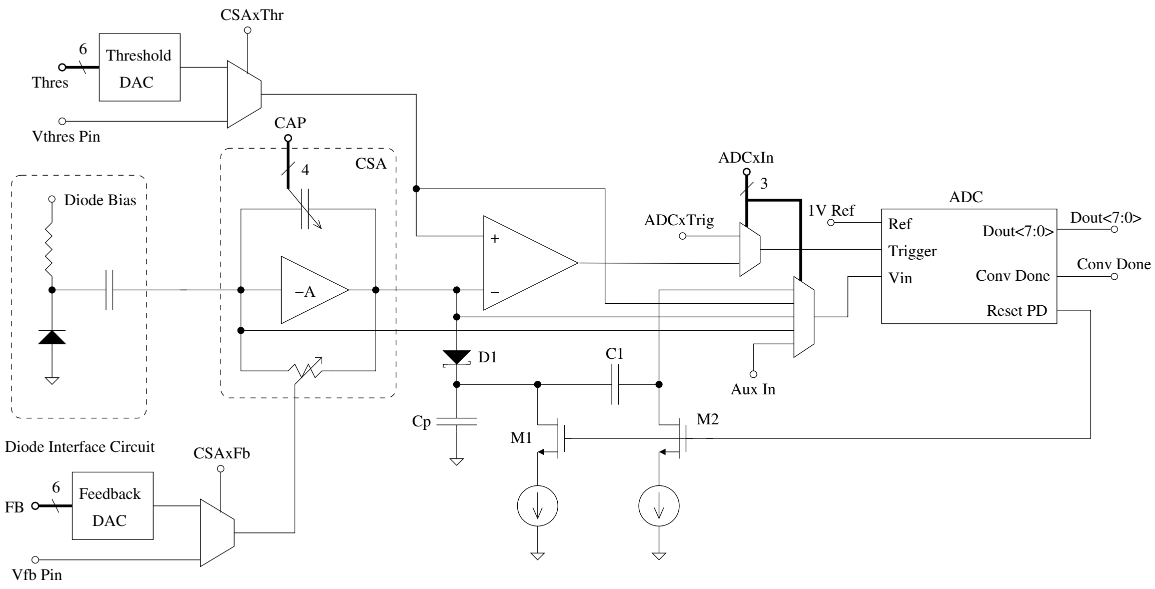


Figure 3.11 – Front end architecture implemented in a 0.18 μm CMOS process with a Schottky diode option.

The initial stage consists of a charge sensitive amplifier (CSA). The amplifier is a lossy integrator. The gain of the integrator is set by $1/C_{\text{fb}}$. C_{fb} is digitally adjustable from between 0.2 pF and 15.2 pF in 1 pF increments via the CAP bits in the associated CSA control adjustment register. The leakiness of the integrator is set by a feedback resistor element. In the actual implementation, the resistor is implemented as an n-channel FET operating in triode region. The gate drive voltage of the FET is adjustable, which adjusts the leakiness of the integrator. The gate drive voltage can either be supplied externally on the V_{feedback} pin or generated internally via the 6-bit DAC. The CSAxFB bit in the CSA control register selects which is used. It is recommended to use the internal DAC. The gate drive voltage should shift with processor and temperature, and the DAC has been designed to automatically shift as it should. If the voltage is applied externally, extra measures would have to be taken to make it shift with process and temperature. The output of the CSA can be read directly by the internal ADC, or is available via the low frequency analog test port.

The output of the CSA next goes to a peak detector circuit where the peak value is extracted. The peak detector works as follows. When there is no particle detection event, the output of the CSA is below the value of the threshold. Therefore, the ADC trigger line is high and the ADC also pulls the “Reset PD” high which turns on transistors M1 and M2. C1 is a DC blocking capacitor, so the input of the ADC settles to zero volts. The ADC is also completely shut off. During a particle detection event, the CSA output goes above the threshold and the ADC trigger line goes low. The ADC then turns off transistors M1 and M2 which creates a peak detector out of Cp and D1. C1 is large enough so it does not attenuate the signal, but instead just subtracts off the DC bias point. C1 is used to eliminate the DC bias because the ADC input is referenced from 0 to Ref. During this time, the internal ADC biases are enabled and begin to settle while the track and latch stage inside the ADC begins tracking the signal. The peak detector formed by Cp and D1 holds the peak value until the CSA output settles to a value below the threshold. At this point, the ADC latches the peak value and begins conversion. The trigger line goes high, the ADC drives the “Reset PD” line high and transistors M1 and M2 are turned on again allowing Cp and D1 to once again begin tracking the input. The ADC shuts off at the end of the conversion.

The “Reset PD” line generally follows the ADC trigger line except in two cases. One is when another ADC conversion is requested while the ADC is still processing the previous conversion, and the other is when the ADC trigger line goes low for less than the minimum sample time of the ADC which is 1.2 μs. In both of these cases, the ADC delays resetting the peak detector for some time to allow the ADC to possibly finish the present conversion, and then sample the input using the required 1.2 μs sampling time. After this is done, the peak detector can be reset. In this way the peak detector works as an analog memory to store the next sample while the previous one is being processed. In summary, the ADC has a maximum continuous rate of 500 kHz, however two events can be faster than that without loss of data if there is enough time after the events to process both of them.

The threshold to which the output of the ADC is compared can be supplied via an external pin or an internal DAC. This threshold should always be some small amount above the quiescent voltage of the CSA output. The quiescent voltage of the CSA output varies with process and temperature, so a fixed threshold voltage is not optimal. The internal DAC will vary in the same direction as the quiescent value, but not in precisely the same amount. Therefore, the microcontroller should periodically check the quiescent value of the CSA output and adjust the threshold accordingly. A mux on the input to the ADC allows the ADC to read both the output of the CSA and the threshold value. This will allow the microcontroller to adjust the threshold to keep the difference constant.

The multiplexers on the input lines of the ADC allow the ADC to read various signals. If the ADCxIn bits in the CSA ADC control register are “00," normal operation is selected for that ADC. If these bits are not “00," one of the other three inputs are selected. With those inputs selected, the ADCxTrig bit in the ADC control register controls the sampling and conversion of the ADC. To perform a conversion, the bit must be set and cleared. When set, the ADC is held in its sample mode, and it begins a conversion when the bit is cleared again. For all cases, the bit can be set one cycle and cleared the next. After the conversion is done, the result is placed in the FIFO associated with that ADC. For ADC1, the auxiliary input is connected to 1/4 of the analog 3.3 V supply. This can be used as a battery voltage monitor if it is directly connected to the battery. For ADC2, the auxiliary input is connected to an internal temperature sensor. For ADC3, the auxiliary input is connected to the low-frequency test port, and for ADC4 it is connected to the internal polarization pixel array. The ADCs can also sample input to the CSA line. An external signal can be driven on this line so the ADCs can be used as general purpose ADCs. However, the associated CSA must be disabled by clearing the CSAxEn bit in the CSA control register. It is not required for the CSA to be enabled via the CSAxEn bit in the CSA control register for the ADC to function.

The digital section of the ADC operates on the digital core supply. It will operate down to a 650 mV supply. The rest of the circuit operates on the analog 1.8 V supply. It requires the global biases, the 1 V reference, and the current reference.

Three internal digital signals are available for viewing on port B of the PIO controller

(PB0 - PB2). This is mainly for debugging. These signals are only associated with CSA1. The CSA ADC clock is a clock generated inside the ADC which clocks the ADC conversion. This is normally stopped, except during conversions when it operates at about 12 MHz (however with a wide variance). The CSA "conversion done" line is the output of the ADC which goes high when a conversion is complete. The ADC trigger line is the control input to the ADC. When the line is low, the ADC is in sample mode. When it transitions from low to high, it begins a conversion. After the conversion is done, the ADC shuts off.

3.3 ADC

The ADC used is an 8-bit successive approximation charge-redistribution type converter. It is a version of the one presented in [63, 64] modified to accept the desired input range and utilize a track and latch comparator. For the same power consumption, a track and latch comparator is faster and was used for that reason. The charge-redistribution successive approximation ADC uses a binary weighted array of capacitors to perform a binary tree search for the correct code. Figure 3.12 shows the analog section of the ADC. The capacitor array also integrates the sample and hold functionality, so a separate sample and hold stage is not necessary. The converter works as follows.

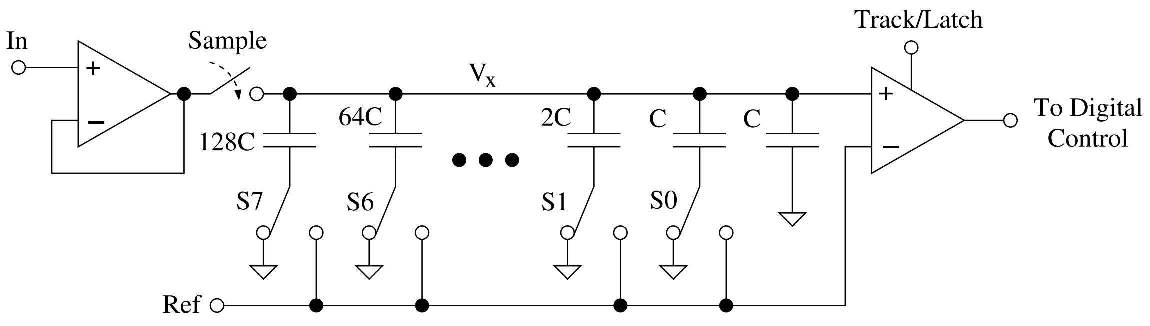


Figure 3.12 – ADC architecture.

First, the switches S0-S7 are in the grounded position and the sample switch is closed. After waiting some time for the input voltage to settle on the capacitor array, the sample switch is opened and the conversion begins. At this point, the op-amp before the sample switch is disabled to save power. Each conversion step consists of switching one switch to the reference position and checking if the voltage at V_x exceeds the reference voltage. If so, the switch is moved back to the grounded position before progressing to the next switch. This process starts with S7 and over 8 clock cycles progresses through each remaining switches. The final position of the switches is then the digital codeword.

The successive approximation type ADC is a medium speed medium accuracy converter. The accuracy is limited by the matching of the capacitor array. Converters over 9-10 bits are difficult to achieve. Mismatch in the capacitor array appears as systematic patterns in the INL measurements. The speed is limited by the speed of the comparator and the RC time constant of the switch-capacitor combinations. However, there is no large penalty paid to make the switches large, so this delay can be minimized. In this implementation, the main speed constraint is the time required to power-up the sampling op-amp and allow the voltage to settle on the capacitor array as the sample time is 1.2 μs and the conversion time is 0.75 μs.

3.3.1 Comparator

A track and latch comparator was used because these comparators have a faster response for a given power consumption than continuous time comparators. In a sampled data application, a clocked comparator is much lower-power. The comparator used in the first AMS 0.35 μm chip has an error which causes the ADC to have very poor INL and DNL performance. In fact, the INL is so bad that there are many missing codes. This is caused by the comparator having insufficient tracking bandwidth, or equivalently, the comparator not fully resetting itself in the tracking stage. This gives rise to hysteresis in the comparator, drastically affecting performance of some codes.

The defective comparator is shown in Figure 3.13. By itself it is not a particularly bad design, however it interacts with the voltage waveforms in the ADC to produce a hysteresis effect. In particular, the voltages at nodes “a" and “b” are not reset to be equal to each other during the tracking stage. This would not be a problem if the input signal was low-frequency band-limited, as the nodes “a" and “b” would follow the input; however, the ADC has fast switching transitions on the inputs which are above the bandwidth of the input differential pair. Therefore, the signals at “a" and “b” do not always follow the inputs, and erroneous decisions can be made.

This could be fixed in a number of ways. One is to extend the tracking time to allow the voltage nodes at “a" and “b” to settle, however this slows the conversion time. Another is to use a transistor to short “a" and “b” together during the tracking phase. However, the charge injection created by the added transistor interacts with the circuit in a complicated way to

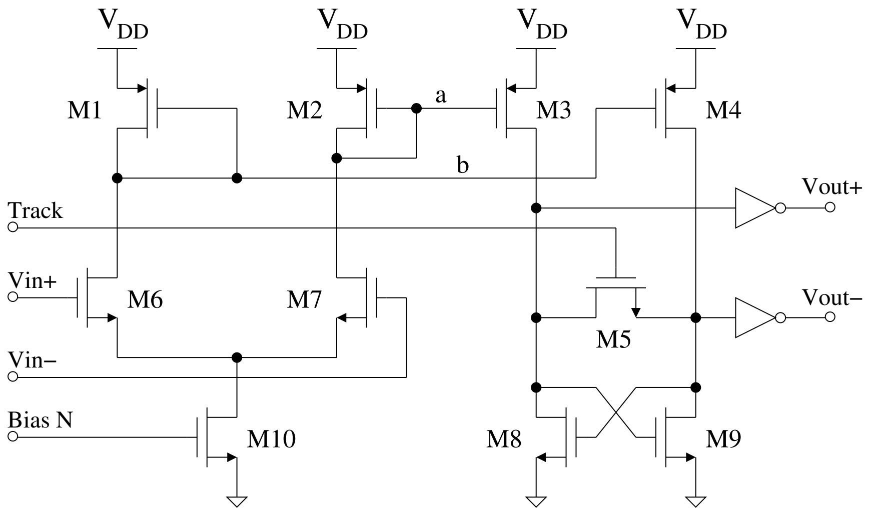


Figure 3.13 – Track and latch comparator present in the first chip.

make the circuit even more sensitive to parasitic capacitance mismatch between the two branches of the circuit—so much so that the circuit becomes unusable. Nevertheless, this can be avoided by creative timing of the gate signal of the added transistor, but generating this timing is not easy in this case.

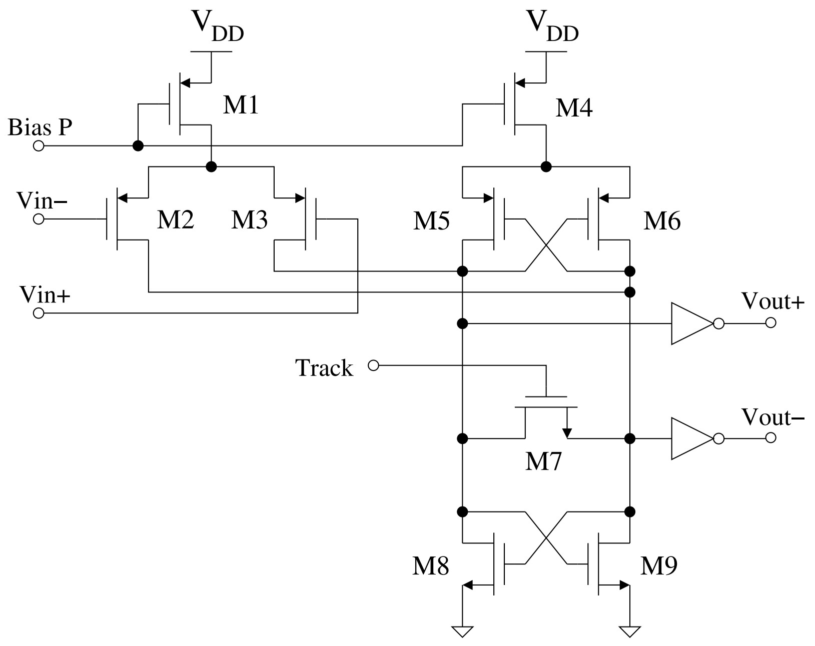


Figure 3.14 – Fixed track and latch comparator used in the third and fourth chip.

Instead, for the third and fourth chips the comparator was replaced with the design shown in Figure 3.14. This comparator has nearly infinite tracking bandwidth and is fully reset by the single transistor. Furthermore, the current which flows through the input differential pair also flows through the NMOS latching transistors increasing their g_m and decreasing the latching time making the circuit faster for a given power consumption. Its main disadvantages are the limited input swing range and the strong charge kickback from the output to the input during latching.

The swing range in this implementation is from 900 mV to 1.1 V. In this application the limited input swing range is not a problem because one input is always fixed at the reference voltage which is 1 V and is in the acceptable swing range.

The charge kickback from the output to the input is caused by the drain to gate capacitance of the input differential pair. It does not affect the operation of the ADC because the charge which is delivered to the capacitor bank during the latching phase is exactly canceled by the charge delivered during the tracking phase.

With this comparator, the INL of the third and fourth chips' ADCs were quite good. The capacitor mismatch was the limiting factor on performance.

3.3.2 Implementation

The ADC is implemented with circuitry in both the analog digital supply domains. The digital control operates on the digital supply and the analog section and the clock generator operates on the analog supply. All circuitry which operates on the digital supply is simulated to operate successfully down to a 650 mV supply voltage, so the ADC can operate when the digital supply is fully scaled for low-power sleeping. For this reason, the clock generator must be operated on the analog supply which is always nominally 1.8 V. Figure 3.15 is a block diagram of the parts of the ADC and their interconnections.

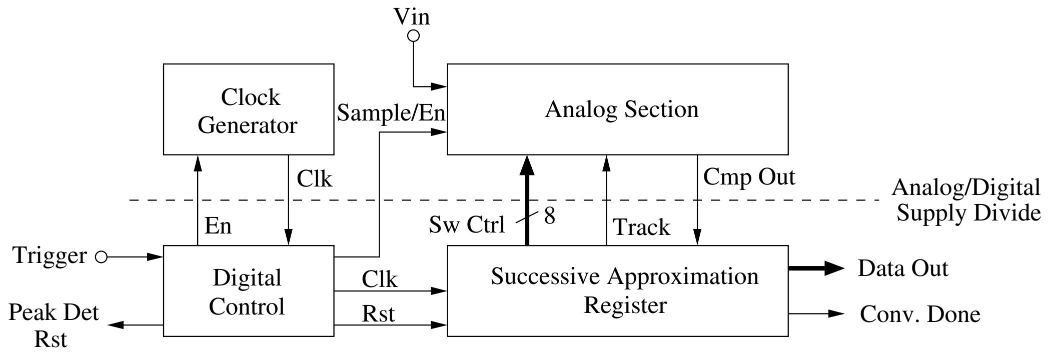


Figure 3.15 – Functional diagram of the ADC.

The digital control block is responsible for two functions. One, it enforces a minimum sampling time for the ADC. If the trigger line is pulsed for less than the minimum sampling time, the sampling time is extended to a minimum of 1.2 μs which is required by the op-amp to power-up and settle. Secondly, if a second ADC conversion is attempted while another conversion is ongoing, it waits till the end of the present conversion before beginning the second. The digital control is also responsible for disabling other modules when not in use to save power.

The clock generator is only present in the third and fourth chips. In the first chip an external clock had to be supplied continuously to the ADC, however this was replaced by a clock generator in the third and four chips. The clock generator concept is shown in Figure 3.16. It works by alternately charging capacitors to a threshold voltage by means of a constant current source. This oscillator architecture was also used for two different oscillators inside the clock generator module, and for more details on its implementation see subsection 7.1.4.

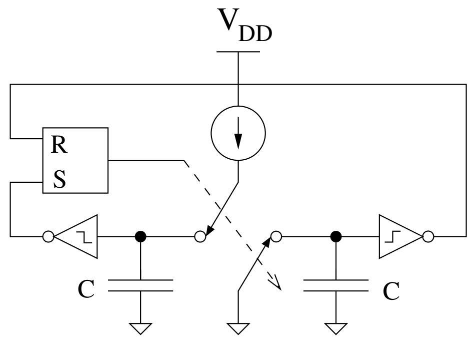


Figure 3.16 – Functional diagram of the ADC clock generator.

The clock operates at 12 MHz and the ADC conversion itself requires 9 cycles, or 750 ns total. However, the sample stage requires 1.2 μs which brings the total conversion time to 2 μs and the maximum conversion rate of 500 kHz. The timing of the 1.2 μs sample time is derived from the ADC clock via a counter in the digital control section.

The digital section was manually designed and laid out as was the analog section. The digital section was designed to operate on the fully-scaled 650 mV digital supply. so front end can process events when the processor is in deep-sleep.

The capacitor array is laid out in a common-centroid fashion to minimize the effect of oxide gradients across the chip. Furthermore a ring of dummy capacitor structures surround the array to improve matching. The other sensitive analog circuit is the comparator.

Extreme care was taken so the two halves of the circuit shown in Figure 3.14 have symmetric layout so they have equal parasitic capacitance. Any capacitance mismatch appears as an input offset voltage. A grounded Metal 3 shield was used above the comparator so Metal 4 and Metal 5 could be used for signal routing above the comparator; without the shield, Metal 4 and Metal 5 routing would cause asymmetric capacitance in the comparator circuit.

3.4 Microcontroller Interface

The ADC interface to the microcontroller is a 16 byte FIFO. Each ADC has its own FIFO. The FIFO can be used to improve the interrupt processing latency tolerance which may be substantial due to the use of DVS. If not all the channels are required, it is possible for one channel to use the FIFO of another adjacent unused channel and therefore double the effective FIFO size. If a FIFO fills up, it checks the status of the CSAxEn bit of the next higher channel. If the next higher channel is not enabled (and therefore assumed to be unused), the lower FIFO will overflow and begin to fill up the FIFO of the next higher channel. If that one fills up, it may continue to overflow to one FIFO after another until either all FIFOs are filled, or it runs into an enabled channel. the FIFO for ADC4 will overflow to the FIFO for ADC1. Figure 3.17 shows the internal structure of two of the FIFOs. When an ADC overflows into the adjacent FIFO, it will set the ADC overflow status bit in the CSA interrupt register but will not lose any data. The only caution when using this feature is if one is using the ADC for an external signal, but not using the CSA. In this case, the CSAxEn bit will be cleared for a particular channel, but the FIFO for that channel will be used. If the FIFO for the previous channel fills up, it will overflow into the used FIFO and data will be lost. It is the application software's responsibility to make sure the previous FIFO does not fill up when using the ADC in this mode! The previous FIFO must always have at least one free byte to guarantee no data will be lost. A channel which is using the ADC for an external signal, and not enabling the CSA, can still overflow into adjacent channels.

Each FIFO is implemented as circular buffer instead of 16 shift registers in order to decrease the number of flip-flops which are clocked on each cycle and decrease power consumption.

A flexible interrupt system was implemented to allow the FIFO buffers to adapt to differing requirements. Each channel has an independent interrupt enable and a FIFO overflow

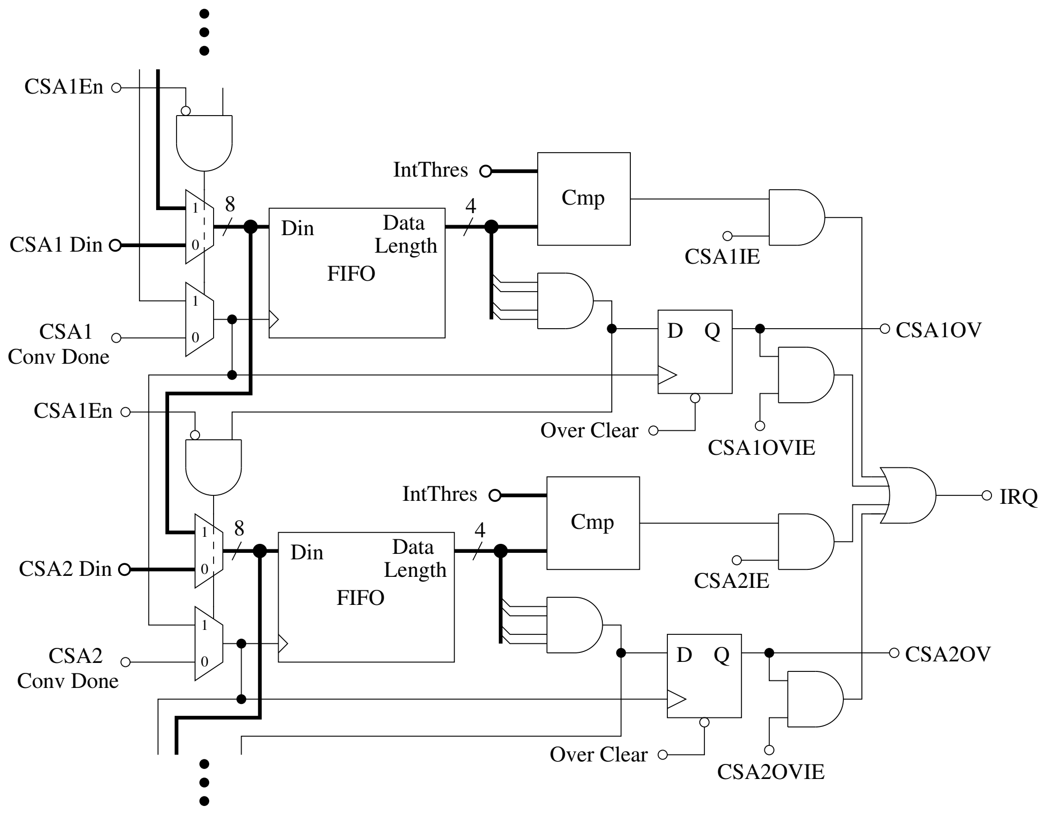


Figure 3.17 – Schematic of two FIFOs for the ADC interface.

interrupt enable. The normal channel interrupt is generated when the the number of bytes in the FIFO is larger than the threshold set in the IntThres field in the CSA interrupt enable register. This field can be zero, in which case the interrupt will be triggered whenever any new data is in the FIFO. Setting this value to 15 will disable all interrupts. The overflow interrupt is generated whenever an ADC conversion is finished with the normal channel FIFO full. If the next channel is used, the new data is lost; if the next FIFO is unused, the overflow bit will still be set, but the data will be present in the next FIFO.

3.5 DACs

Three low-power DACs are required for the design. They are the DAC required for the diode drive circuitry, the DAC required for the threshold adjustment, and the DAC required for the feedback voltage adjustment. Two are shown in Figure 3.11 and one in Figure 3.4. They share similar requirements and have a similar structure. Most importantly, they must be extremely low-power. They operate only at DC, and need to be small in area. The low-power requirement necessitates large resistance values, and the small area combined with the high resistance, requires that the resistors be very narrow. This in turn leads to the fact that the resistors will suffer greatly from process variations. Therefore, the accuracy of the DACs will be poor. For the diode DAC and the V_{fb} DAC, the accuracy does not need to be good, and they can both be calibrated for each chip to achieve acceptable performance. However, for the V_{thres} DAC the accuracy needs to be better because this voltage must be only a few tens of millivolts above the CSA quiescent value. To make matters worse, the CSA quiescent value is not constant with PVT variations. For this reason, the ADC in Figure 3.11 is able to sample both the CSA output and the threshold and then system software can adjust the threshold DAC so it is set to a small value above the CSA quiescent value. This check can be periodically performed, even when the sensor is online, to make certain the threshold value remains set correctly.

The DAC architecture consists of a constant current source driving a digitally adjustable resistor. This particularly simple architecture is very low power and has a small area, but is not fast, linear, or resistant to process variations. Fortunately, the systematic non-linearity caused by the non-zero switch resistance is proportional to the current, so for the low-power DACs used, the systematic non-linearity is pretty good. Each of the DAC designs are slightly different and are shown in Figure 3.18.

The current flowing through each DAC is on the order of 100-300 nA. The current sources for each is derived from the system current reference. The output voltage of each DAC is of the form

$$V_{\text{out}} = V_{\text{base}} + I_{\text{bias}} \times In \times R_{\text{base}} \tag{3.6}$$

where R_{base} is the value of the smallest resistance in the resistor chain and V_{base} is the voltage at the base of the resistor string. For the diode DAC this is the 1 V system reference, so this DAC ideally does not shift with process and temperature. For the other two DACs, the V_{base} value is either V_{gs} or $2V_{gs}$ and does shift with process and temperature. This is because the required output voltages from those two DACs need to shift with V_t and $2V_t$, respectively, for the operation of the CSA. This is one reason why it is recommended to use the internal DACs and not use externally applied voltages.

3.6 Conclusion

This chapter presents the design of a complete low-power charge-sensitive amplifier front end. Beyond just the design of an amplifier, the design is coupled with a peak detector

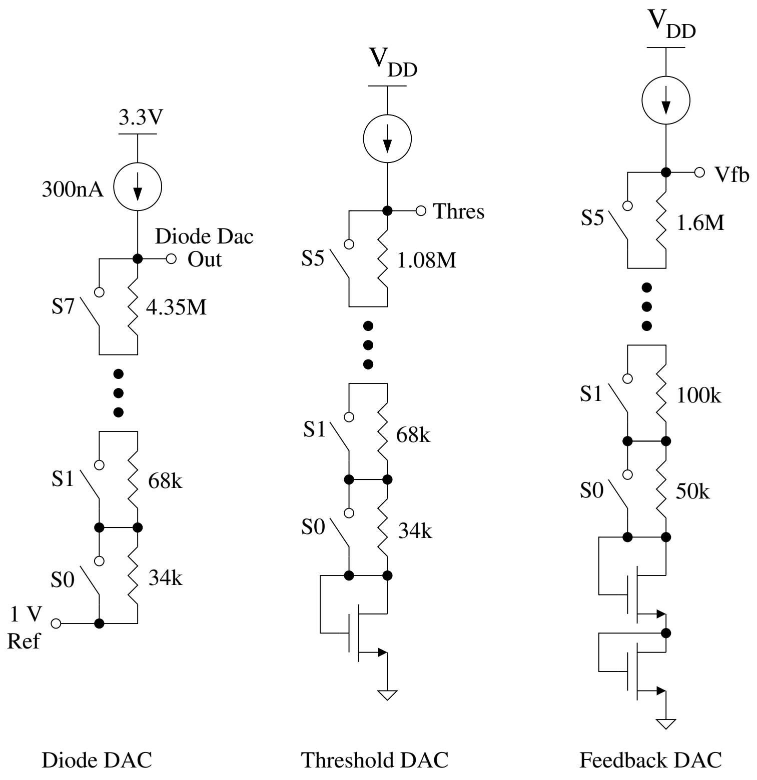


Figure 3.18 – Schematics for the three DACs.

and ADC to create the functionality of a multi-channel analyzer with a small amount of software. The total silicon area of the front end and ADC is 35,000 μm^2 with the IBM 0.18 μm process.

Chapter 4

Digital Core Design

The digital section of the design contains the processor, memory, peripherals, and many parts of the RF transceivers. It is a major portion of the design requiring a majority of the total die area. The digital section occupies over 6.3 mm^2 of silicon with over 36,000 individually placeable standard cells. It is a highly complex system which is integrated closely with the analog section of the design. Over 1,200 I/O pins enter or leave the digital section of the design to connect to other parts of the chip. 18 different clock domains are synthesized for different parts of the design with over 7,300 flip-flops. Figure 4.1 shows how the digital section of the design fits into the complete sensor node.

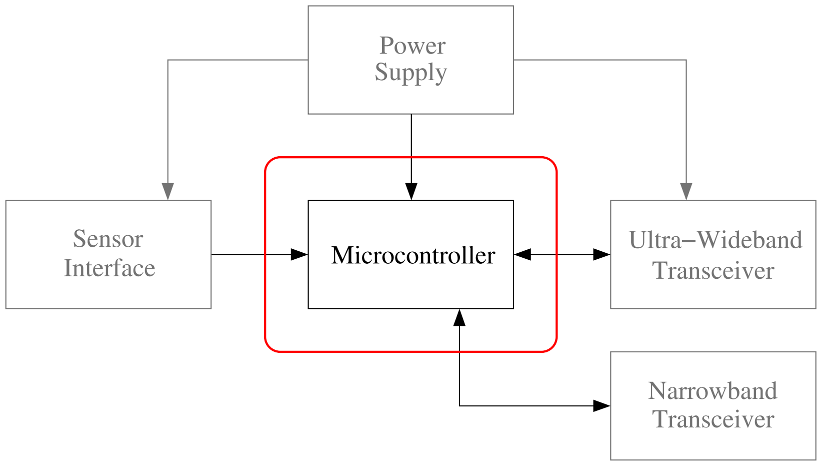


Figure 4.1 – Digital section enclosed in red box.

The processor is connected to various peripherals and memory through a shared bus. Physically, the memory accounts for most of the space with the UWB receiver requiring most of the remaining space. The processor and other peripherals share the remaining space. The floorplan of the digital section is shown in Figure 4.2.

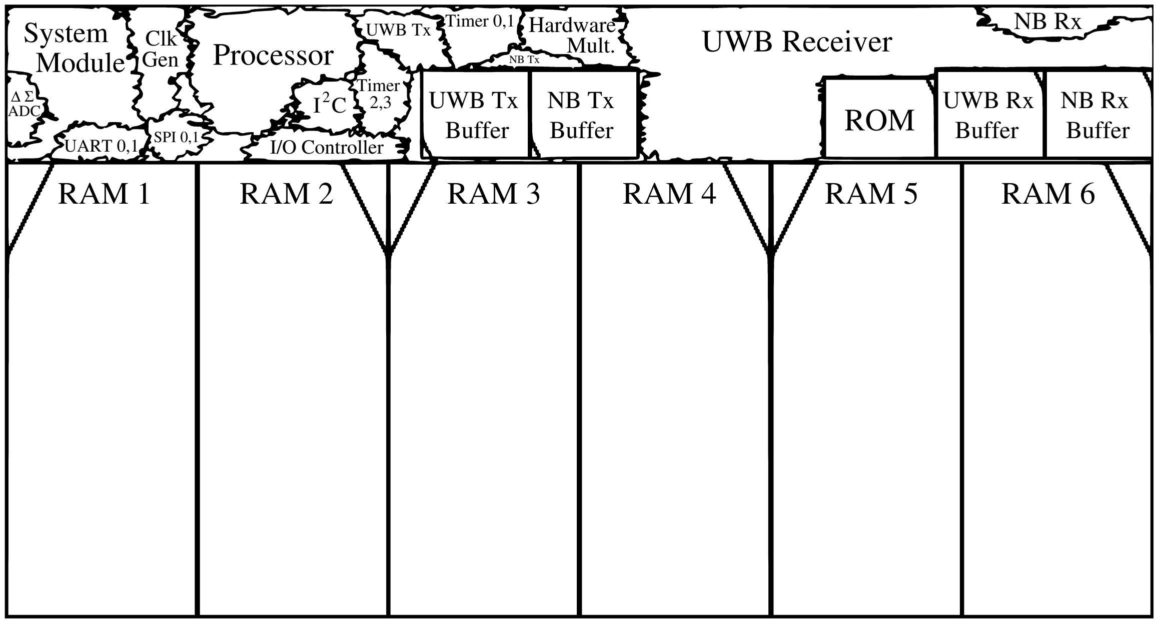


Figure 4.2 – Digital section final placement.

Most of the space is taken by 6 banks of SRAM; each is 16 kB. The RF transceiver buffers are dual port register files, and a 4 kB ROM complete the hard memory IP blocks. The ROM is mask-programmable ROM which contains the bootcode. The rest is synthesized from standard cells.

The most important factor driving the digital architecture is power consumption. Aggressive clock gating was used throughout the design to achieve small activity factors and decrease power consumption.

The rest of this chapter is organized as follows: section 4.1 presents the synthesis flow and the steps used to take the RTL-level VHDL and produce a GDS layout. Section 4.2 presents the format of the processor bus and how data is transferred on it. Section 4.3 presents an overview of the processor and its design and implementation. Sections 4.4-4.9 discuss the various all-digital peripherals which are used by the processor which include a CRC module, hardware multiplier, timers, serial peripheral interface module, UART module, and I^2C module. The other peripherals which are not all-digital are discussed in their respective chapters. Section 4.10 concludes the chapter

4.1 Design Flow

The design begins with behavioral simulation of RTL-level VHDL. The behavioral simulation contains no timing information, so all circuits operate infinitely fast. Many VHDL constructs cannot be synthesized but will simulate at the behavioral level, so care must be taken to avoid such constructs. Extensive functional simulation is performed at this level because it is easiest to debug at this level. After the design is fully synthesized, only a subset of the functional simulations are performed due to the difficulty of simulating at that level.

4.1.1 Source Files

The design flow starts with an RTL-level VHDL description and constraint files. The constraint files specify a number of important properties of the design. First, they specify the number and frequency of the clock trees in the design. Next they specify timing exceptions such as false paths in the circuit. False paths are circuit paths which can be found through the combinational logic in the design but will not actually be used in practice. Practically, we only need to concern ourselves with the paths which are longer or similar length to the actual critical path of the circuit. If a false path is left in the circuit which is longer than the critical path, the timing optimizer will focus all its effort on optimizing that path (possibly at the expense of other paths) which is not actually used in the functional operation of the circuit. Furthermore, the actual critical path will not be optimized. Therefore, it is important to enumerate the false paths in the circuit in the constraint file.

A second type of timing exception are multi-cycle paths. Multi-cycle paths are paths through a circuit which are used functionally, however the circuit is designed to capture the data from those paths two or more cycles after it was launched. By default the timing engine assumes all paths have only one clock cycle to settle to the correct value. However if the VHDL designer knew that a certain circuit would take longer than one cycle to compute the result, he could program the circuit to sample the data on the second clock to give the circuit more time. Multi-cycle paths specify which paths may be allowed more than one cycle produce the correct result. These paths are much more rare than false paths and are generally specifically known by the designer.

Finally, timing loops can exist in the design which must be broken. A timing loop is a combinational circuit which loops back to itself without going through a flip-flop. These are very common in the datapath section of the microcontroller due to the network of buses which can easily route a signal back to its starting point. An exception must be specified telling the timing engine to ignore those loops and not be confused by them.

Other technology and library information files are also necessary to inform the synthesis and place and route engine about the fabrication process used. Furthermore information is needed about the standard cells and memory IP blocks. The router needs the blockage information telling where the metal is used and where it is not so the router can route around existing metal, as well as antenna information. For more information about process antennas see subsubsection 4.1.5.1

4.1.1.1 Memory IP

Three different types of ARM memory IP are used in the design. Mask-programmable ROM is used for the bootcode, dual-port register file is used for the RF transceiver buffers, and single-port SRAM is used for the main memory.

On the first chip, no IP was used and the main memory was designed manually. On the second chip, dual-port SRAM was used instead of the register file for the RF transceiver buffers. However for later chips the dual-port register file was used because it is smaller and requires less power.

As is the case with the standard cell libraries, the layouts of the memory is not available to us. Instead, the generators provide front-end views which include the simulation model and place and route data. The actual layouts for the memory are generated by memory generators at MOSIS and inserted before the design is fabricated.

A specification file is required as the input to the memory generators. This specification file includes all the important parameters of the memory such as bus width, memory size, multiplexer width which controls the aspect ratio of the generated memory, power ring metal layers and the like. The memory generator produces a number of files, however only some are used in the simulation and place-and-route flow. These are the datatable, .vclef file, antenna .lef file, timing information (in both liberty format and cadence timing library format), and the verilog model.

The datatable is a human-readable text file which contains all the important performance parameters of the memory such as access time, active power consumption, standby power consumption, peak power consumption, area, and numerous others. The .vclef file contains layout information needed by the place and route engine. The layout information present is the location of all the pins, the location of the power ring, and large metal blockages over the core. The antenna .lef file contains information about the size of the gates connected to the input pins and the presence of diffusion connected to the pins. This information is needed to determine if any process antenna violations exist in the higher levels of the design. For information about process antennas, see subsubsection 4.1.5.1.

The timing information is used for simulation and for static timing analysis within the place and route engine. The synopsis liberty format is used in the first step of logic synthesis, while the cadence timing library format is used for subsequent steps. The verilog model is required for post-layout simulation. However, for high-level behavioral VHDL simulation, the verilog model is not suitable because it requires extensive information from the layout about timing delays. Therefore, a very simple behavioral-level VHDL model was written for the memories for high-level simulation.

4.1.2 Logic Synthesis

Next RTL Compiler from the cadence toolchain is used to create a gate-level netlist from the VHDL. This is done in two steps. In the first step RTL compiler parses the VHDL source and produces a technology-independent netlist. It is in this step that don't cares in the source files are mapped to actual logic. All further steps throughout the design flow cannot change the functionality of any digital circuit because they do not know where the don't cares are. The further steps will only re-map existing logic with equivalent logic.

From the technology independent netlist, RTL compiler maps the circuit to a particular technology and cell library. The cell library functions and timing information are loaded which RTL Compiler uses to determine the most efficient logic mapping. A simple wireload model is used at this point. The wireload model is used by RTL compiler to estimate how much wiring capacitance will be on the wire when the design is placed and therefore how much that will affect delay. RTL compiler will re-size the drivers of nets to decrease the fanout delay and wiring delay as required, however at this point the wiring capacitance information is very inaccurate. Later, the place and route engine will change the driver sizes to fit the actual wiring capacitance.

RTL compiler also inserts most of the clock-gating logic in the design. Some clock-gating logic is described in the VHDL source files, however the majority of the 370 clock gating cells are inserted automatically by RTL compiler.

RTL compiler does static timing analysis on the design and optimizes the design to meet timing requirements. Because RTL compiler is over-optimistic about the timing, it was configured so it tries to achieve a timing which is faster than the actual target. This way when the place and route engine does timing analysis, it is closer to the necessary target.

A couple hardware cores are not synthesized through RTL compiler. Instead they were drawn as a schematic in cadence using the cells available in the standard cell library and exported as a verilog netlist. RTL compiler does not synthesize these blocks because they are already mapped to the physical library. However, RTL compiler can re-size the drivers or insert or delete buffers in these blocks to meet timing requirements.

The two blocks which were included this way are the carry-select adder which is the core of the microcontroller ALU, and the booth-encoded 16x16 bit signed/unsigned multiply-accumulate block in the hardware multiplier. It was discovered that the synthesized multiplier was very inefficient, so the schematic was drawn manually to make it efficient. However, the place and route engine is still responsible for the layout.

RTL compiler then writes out the netlist as well as the constraints in an SDC file. These two files are the input files for the next step.

4.1.3 Placement

The placement, clock tree synthesis, routing and extraction are all performed within the cadence tool Encounter. The placement requires three steps, floorplaning, power planning/routing, and placement.

Floorplanning is the first step. First the size and aspect ratio of the design is chosen. An important factor is the cell utilization which is defined as the area available for placement divided by the total area of the standard cells. This is a measure of the free space in the design. Higher utilization is desired to save space, however routing congestion and other problems exist if the utilization is too high. Initially the utilization is set to 76%, however this will grow to 86% by the end due to buffer insertion and gate resizing. Utilizations of up to the mid-nineties are acceptable with this design. The lower utilization was the result of fixing the digital size early in the design cycle.

The overall dimensions of the digital design was set relatively early in the design cycle, before all the final digital blocks were designed. The rest of the chip was designed with this fixed size in mind, so changing the size is difficult. Therefore, some extra area was allocated originally for the planned but unfinished modules. Hence, when all the modules were finished, there was still some extra space leading to the somewhat lower utilization. However, utilizations in the mid-nineties are very difficult to work with and should be avoided if possible.

The I/O pins are placed on the periphery of the digital section, with their locations determined by where they will be routed to after leaving this section.

The hard IP macros are placed as well as some soft-placement guidelines for the placer to follow when placing the standard cells. The soft-placement guidelines are only guidelines, and the placer need not strictly adhere to them. Only a few modules are placed, the rest will be automatically placed. Figure 4.3 shows the placed hard IP macros and the soft placement guidelines.

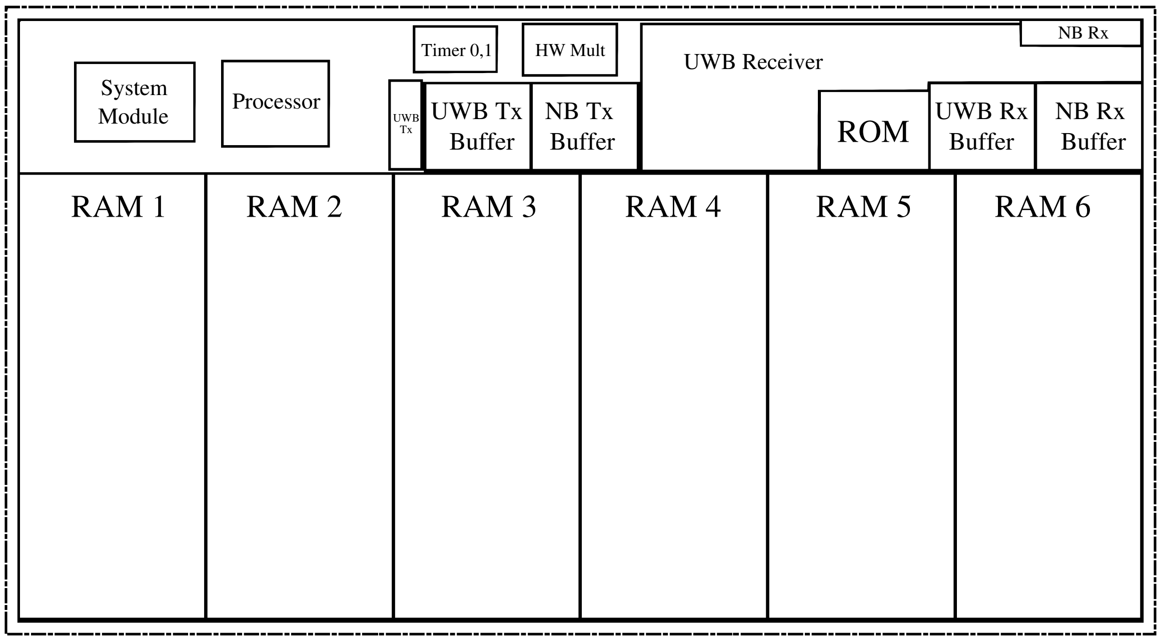


Figure 4.3 – Digital section floorplan showing memory placement and soft placement guidelines.

Next power planning and routing is performed. A power ring is placed around the edge of the design and vertical power straps are placed across the design at 90 μm increments. Both the ring and the straps are placed on the top layer (metal 6) because it is thick and can carry much current. Horizontal stripes are also placed over the memories on metal 5. The memories have their own internal power rings. Whenever a power stripe crosses one of the memory rings, vias are inserted to make a connection. In this way the power is supplied all around the power ring for each memory. For the standard cells, the horizontal power wires have vias to the vertical power stripes at every possible location. The ends of the standard cell power wires also connect to the power ring (followpin routing).

Finally placement can take place. Before placement, all buffers and inverters are removed from the netlist (the necessary inversions are of course remembered to be inserted later). The inverters and buffers will be re-placed as needed after final placement. This is because it produces better results to place buffers as needed after placement than to have the buffers in the netlist and attempt to place them as one would any standard cell. The placement engine is timing aware so it uses information about the critical paths to optimize placement. After many iterations, it then stops placement and begins the first round of logic optimization. This logic optimization will be performed many times in the design, though this first one is the most difficult and requires the most time.

Logic optimization first executes a trial route. Trial route does not make a DRC-free route, however it is a quick way to ascertain the approximate routing of the design. After trial route, a quick parasitic extraction is performed to get the wiring capacitance and wiring delays associated with the layout. Then static timing analysis is run to determine the critical paths. The logic optimization first fixes design rule violations such as maximum transition time violations and maximum capacitance violations by inserting buffers. Then it begins optimizing the critical path. It can insert or remove buffers, re-size drivers, re-map logic, move standard cells, or perform pinswapping to optimize the critical path. After doing some optimizations, it goes back to beginning and re-routes with trial route and repeats the process. It also attempts to reclaim area by shrinking gates or removing buffers.

After logic optimization, the design goes back to the placement engine which re-places most of the design. Then another optimization cycle is performed to optimize the placed design. Figure 4.2 shows the final placed design.

4.1.4 Clock Tree Synthesis

After placement clock tree synthesis (CTS) is performed. The goal of clock tree synthesis is to match the clock skew between flip-flops. It does this by inserting buffers in the fast paths to slow them down so all clocks arrive at the flip-flops at the same time. This job is complicated by clock gating. Clock gating introduces extra gates, and therefore delay, between the clock source and flip-flops which makes the clock tree synthesis problem harder. The added buffers also increase power consumption. Clock tree synthesis will insert many buffers (up to 15 in this design) to slow a clock down to match other clock signals which are buried very deep in clock gating cells. However the clock gating cells require special handling in the clock tree synthesis.

The clock gating cells must be specified and handled differently in clock tree synthesis. The top part of Figure 4.4 shows how the clock tree synthesis will handle the clock gating cell by default i.e. like it handles any latch/flip-flop. It will minimize the clock skew between the clock gating latch and the functional flip-flops by inserting buffers before the

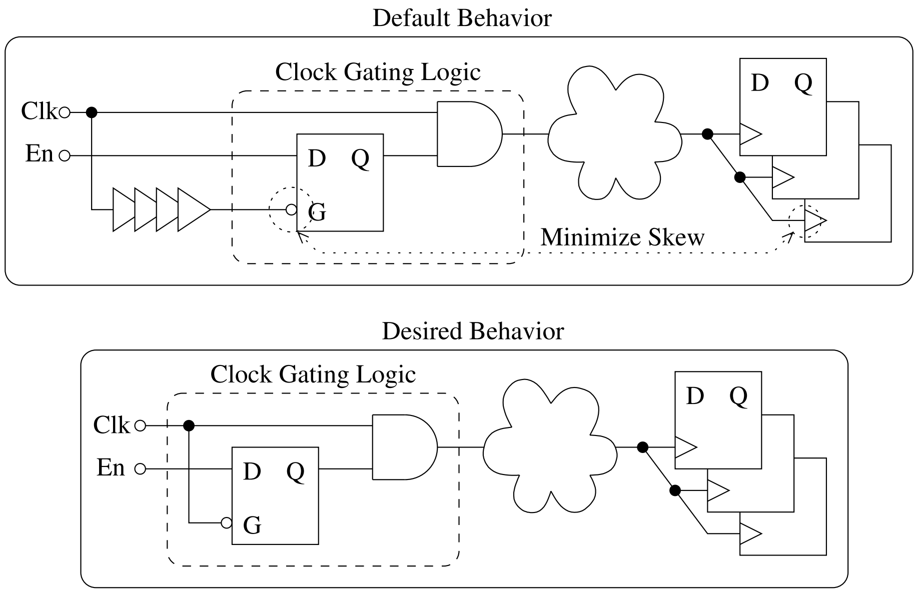


Figure 4.4 – Clock tree synthesis clock gating showing the default handling and the desired handling.

latch. However, the clock gating cell is a special case where we need the clock for the latch to arrive at the same time as it does at the AND gate. If this is not the case, the high-time of the clock from the gating cell may become distorted from the input clock and excess power is consumed in the added buffers.

The clock tree specification file lists all the clock domains and the clock gating cells which should be handled as shown in the bottom of Figure 4.4. It also specifies the details of each individual clock tree.

The clock tree synthesis will also route the clock tree, though no other signals are routed yet.

After clock tree synthesis, congestion optimization is run. In this step trial route is run with particular attention paid to the routing congestion in the design. Congestion optimization attempts to move cells to improve routing congestion. Finally, another post-CTS logic optimization iteration, as described above, is executed.

At this point the standard cell density for the design is 86%. Fill cells are added to completely fill all remaining space with dummy cells before routing.

Routing blockages are placed on the routing layers above the memory IP blocks. This is because routing signals on top of the memory can cause them to fail. In particular, signals which are routed on top of the memory modules can couple into the data bus lines of the memory array and disrupt operation of the memory. Therefore, all signal routing is blocked over the memory IP blocks. The only places the signals are allowed to route around memories is through the area where the power ring for the memory is. This is somewhat congested due to the power rings and the vias from the power stripes. These locations are the most congested areas in the design.

Routing blocks are also added on some metal layers in some locations around the outside power ring of the design. These routing blocks are added so that external power and ground supply connections can be made after the design is finished and imported into Cadence Virtuoso environment.

4.1.5 Routing

The design is then routed in two steps. In the first step—global routing—an overall plan for the routing is determined. The design is divided into a grid of g-cells which have a certain number of horizontal and vertical routing resources available. Each net which needs to be routed is then assigned space in one or more g-cells which it will traverse to reach its destination. The precise routing within the g-cell is not determined yet. A number of iterations are performed to decrease the number of g-cells which are congested (i.e. require more routing resources than are available). Then detail routing is performed where the individual wire connections are made following the plan created in global routing.

After all nets are connected, optimization begins. The three goals of optimization are to reduce the net length or the number of layer changes, to eliminate process antenna violations, and to insert double-cut vias.

4.1.5.1 Process Antenna Violations

A process antenna is a large area of metal which is exposed to ions during IC processing and is also connected to a CMOS gate. During processing in an ion-rich environment, the ions collect on the metal layer and induce a voltage on the gate of the transistor. If the gate area is too small or no other means of dissipating the charge are supplied, the voltage can damage the gate oxide leading to device failure. Figure 4.5 (a) shows an antenna violation. Antenna violations appear when a long route with a large metal area is connected to the input of a CMOS gate. However, if the wire is also connected to the output of a CMOS gate at the time the offending metal layer is processed, there usually will not be an antenna violation because the output of the CMOS gate is connected to diffusion and therefore a reversed-biased p-n diode. At the high-temperatures involved during processing, the diode will leak enough to safely dissipate the charge and keep the voltage safe.

If a process antenna violation exists, two methods are used to fix it. One is layer-hopping and the other is to introduce an antenna diode. Figure 4.5 (b) shows the introduction of an antenna diode. This fix simply consists of adding a reversed biased p-n diode to the layout which will leak and dissipate the charge. Figure 4.5 (c) and (d) shows the layer-hopping antenna fix. This consists of jumping up a layer and back down close to the input of the CMOS gate. In this example, during processing of M2, the long M2 wire will be isolated from the gate. Only during M3 processing are they connected, at which point the M2 wire does not collect ions because it is insulated with SiO_2 on top.

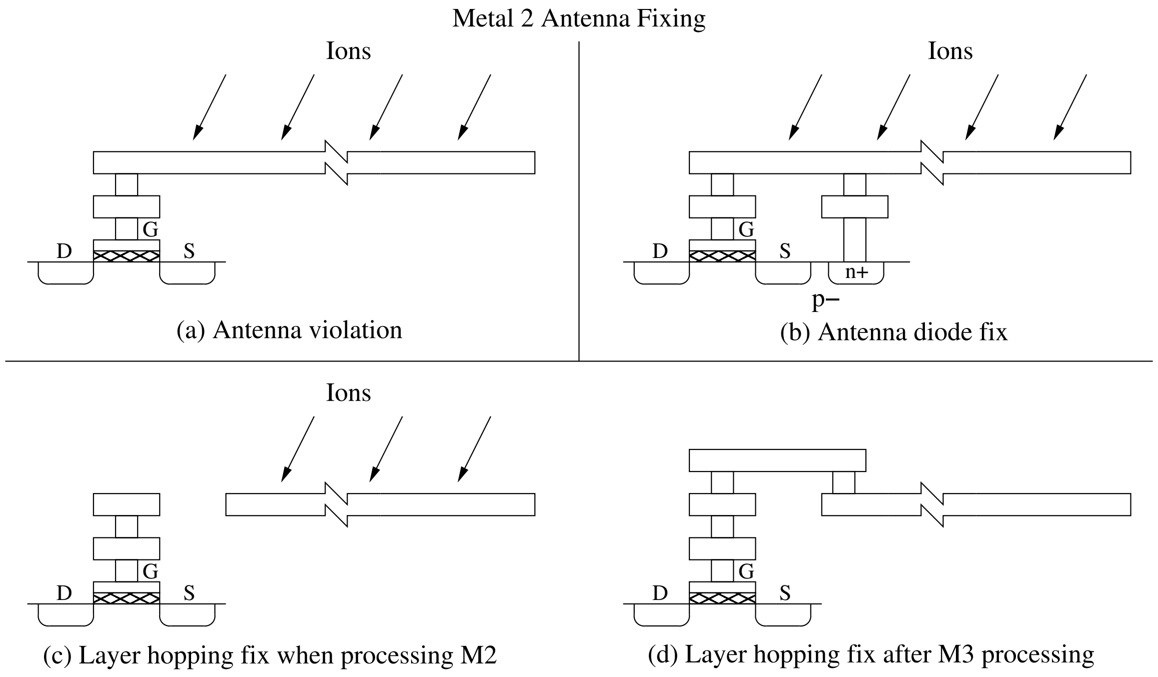


Figure 4.5 – (a) Antenna violation, (b) Antenna diode, (c) and (d) Layer hopping fix.

The router prefers to use layer hopping to fix antenna violations and only inserts antenna cells as a last resort. In this design, to start out with, there were 500 antenna violations with most being fixed via layer hopping and less than 10 required antenna diodes.

The optimization also attempts to insert double-cut vias in place of single-cut vias where space permits. Double-cut vias are just as they sound, two vias are placed in parallel from one layer to the other instead of a single via. Double-cut vias are desired somewhat to reduce delay, however their main purpose is to improve yield. Open vias are a major source of yield loss, so paralleling vias improves the yield. Generally 15-20% of vias can be made double-cut vias.

After optimization, DRC and LVS are performed on the layout to verify all the connections are made properly and the design passes manufacturability tests. The layout is written to a gds2 stream which can be imported into Cadence Virtuoso and integrated with the analog section and pads. The DRC in Encounter is not quite exhaustive of all checks, so in Cadence Virtuoso another more thorough DRC check is performed and a few errors are found and fixed.

4.1.6 Signoff Extraction and Simulation

After routing a much more detailed extractor is run. A simple extraction algorithm was used hundreds of times throughout the place and route steps to get an estimate of wiring capacitance and delay, however it is not very accurate. Now a much more complicated extraction is done to determine the final parasitic capacitance and wire delays.

After extraction, static timing analysis is run to report on any violating paths. Clock skew checking is performed to verify the clock skew is within limits. The individual wiring delays are calculated and written to an sdf file for post-layout simulation.

Post-layout simulation is performed with the nc-verilog and nc-simulator tools from Cadence. First the sdf file is compiled and back-annotated to the netlist. The netlist can then be simulated with realistic timing.

In conclusion, a number of steps are required to take a VHDL netlist to a gds layout. The entire process is fully automated with a tcl script and requires approximately 20 hours on the doppler server, at times using both of doppler's processors.

4.2 Bus Interface

The bus used in the system is a simple single-master bus. The CPU is the only master, and no DMA exists. The one bus is used for memory and peripherals. The architecture of the bus changed somewhat between the second and third chips. In the first and second chips the bus was modeled after a standard board-level embedded system bus where a common bus was shared between all components with each component either driving the bus or tri-stating it as directed by the chip-select logic. It was discovered that this architecture does not work well on an IC because the tri-state bus is a slow bottleneck which cannot be sped up. The self-loading of all the drivers guarantees the bus is slow, and gets slower with more components. There is also no way to add buffers to the bus. Starting on the third chip, all internal tri-state buses were eliminated with multiplexers being used instead. Besides being much faster, this also requires smaller area and allows buffering of the nets.

The bus architecture is shown in Figure 4.6. The data bus is split into two separate buses, one for data in each direction; however, only one is used in any given bus cycle. The bus is fully synchronous. The clock is disabled to modules which are unused to save power. In the figure, the AND gates are actually clock gating modules (so they can't glitch). The most area is required by the large 16 bit multiplexer feeding the data input line of the processor.

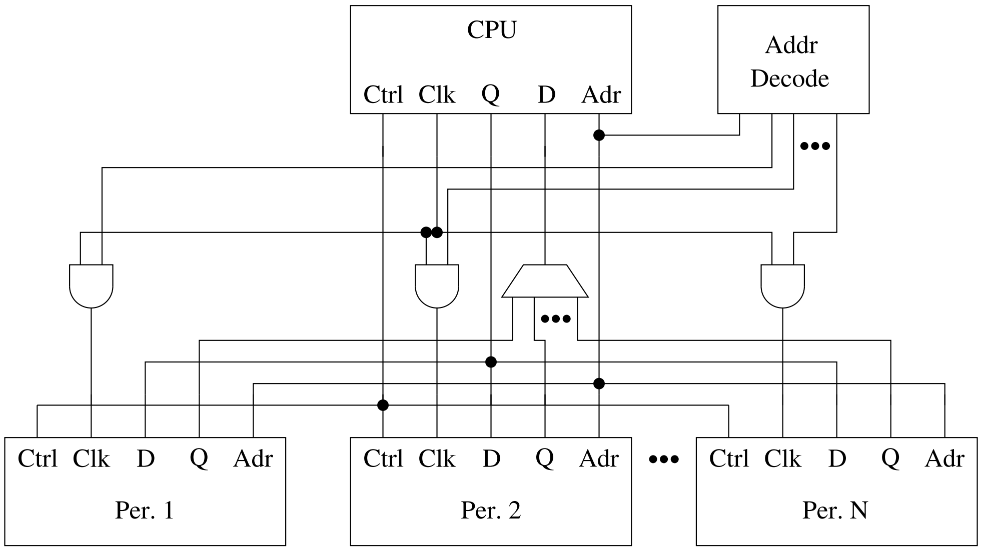


Figure 4.6 – Bus architecture

The bus read and write cycle is shown in Figure 4.7. All bus operations are synchronous. The clock to each peripheral is gated off when the peripheral is not accessed, however two clock cycles are included when accessing a peripheral. The second cycle is not always required, however it is present for two reasons. First, for the SRAM, the second clock, which occurs when the enable line is inactive, places the SRAM into the standby state and therefore saves power. For the peripherals, the second clock cycle is sometimes used to perform the requested operation. For example, in the hardware multiplier or CRC module, the actual multiply operation or CRC generation operation takes place in that second cycle.

In the read operation, the address and control signals are latched on the rising edge of

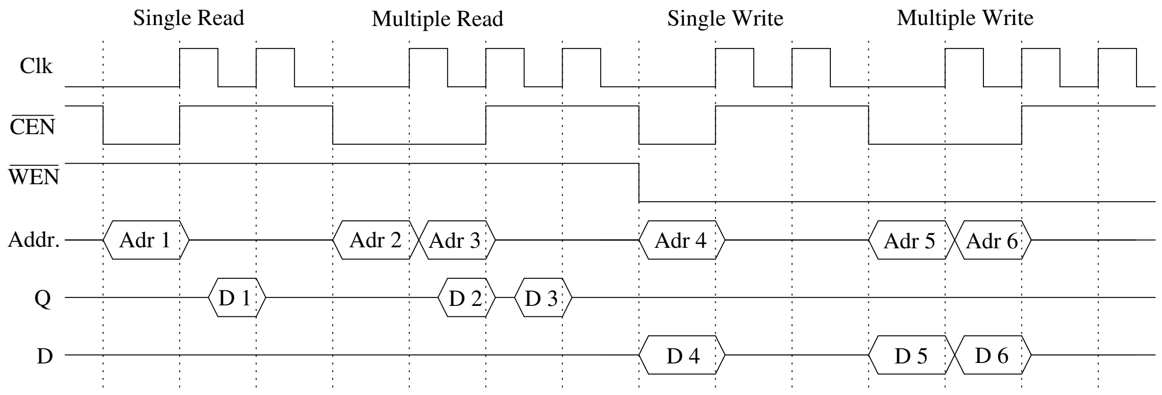


Figure 4.7 – Bus read and write.

one clock cycle, and the data is driven out in the next. Another address can also be driven out while reading the previous data which results in a throughput of one transaction per clock cycle.

In the data write operation the address, data, and control signals are latched on the rising edge of one clock, and the write is performed in the next cycle. Within the peripherals, the write operation takes place on the next falling edge of the clock. The buses labeled 'D' and 'Q' are labeled relative to the peripherals, so 'D' is the data from the processor and 'Q' is the data to the processor.

The control $\overline{\text{WEN}}$ indicates whether a read or write operation is taking place. It is actually a two-bit bus, with the two bits corresponding to the write-enable lines for the two bytes of the 16-bit data bus. In this way the processor may only write one byte of data at a time.

The address decoding logic determines the memory map for the device. The first three chips had a 16-bit address bus because the original MSP430 only supported 16-bit addresses. Therefore they were constrained to a 64 kB addressable memory range. However for the fourth chip the expanded MSP430X CPU architecture was implemented which allows for 20-bit addresses, or 1 MB addressing range. The lower 112 kB of memory is used, while the rest is unused. The memory map for the fourth chip is shown in Figure 4.8.

Any functionality not explicitly included in Figure 4.8 (such as the CSA interface and diode DAC interface) is included within the "System Module" area. Any write access to an unused area has no effect and any read access returns unpredictable data, however no interrupts or exceptions are generated by the invalid access.

Any data above the lower 64 kB boundary is not accessible using the original MSP430

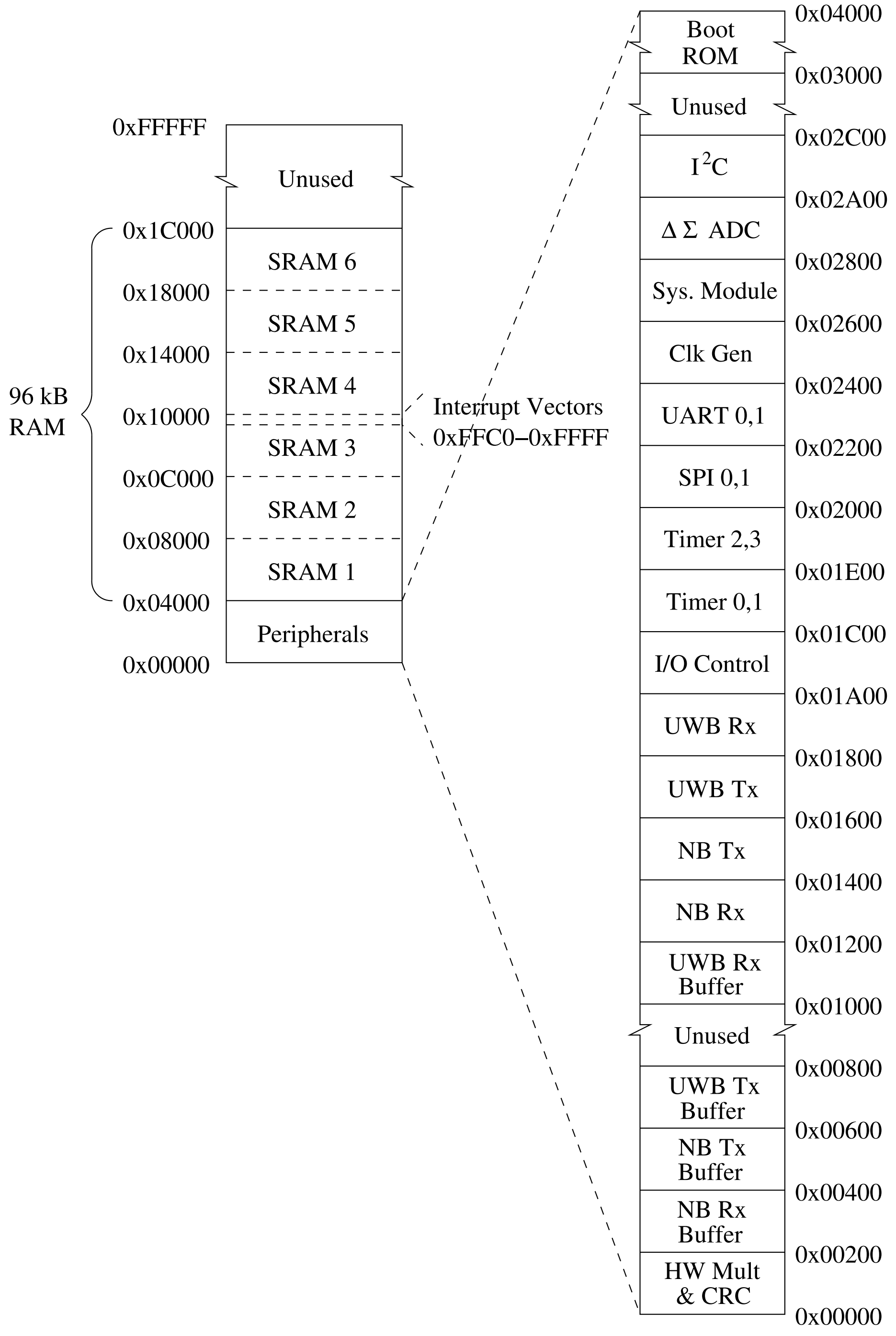


Figure 4.8 – Device memory map

architecture. Instead the extended MSP430X architecture is required to access data or execute code beyond the lower 64 kB of memory space.

The address decoding on the bus is incomplete. The most significant address lines are decoded to select which peripheral or memory, while the address lines in the middle are sometimes ignored, and the least significant address lines select the individual register within the peripheral. This means that the peripheral registers repeat a number of times within their allotted space. Each peripheral is allotted 512 bytes of space, but if fewer than 512 bytes are used, the registers repeat within the 512 byte space with a period of

$$Rep = 2^{\lceil \log_2 N \rceil} \tag{4.1}$$

where N is the number of registers. Generally the first repetition of the registers are used when the processor accesses the peripheral, however for the hardware multiplier, to make the register addresses match that of the original MSP430, the processor accesses one of the later repetitions of the registers.

4.3 Processor

Two different CPU processors were implemented. In the first, second, and third chips the MSP430 architecture was used. This is strictly a 16-bit architecture which has 16-bit address bus and therefore 64 kB directly accessible memory. In order to add more memory, in the fourth chip the MSP430X architecture was used. Both of these architectures are used in the popular MSP430 line of processors from TI [65]. The MSP430X architecture is used in some of TI's latest generation chips. It is backwards compatible, so machine code for the MSP430 will operate un-changed on the MSP430X, however it also supports 20-bit addresses to allow 1 MB addressing range. The implemented processors are generally compatible with the TI MSP430 and MSP430X. An extension not present in the TI parts was added to the MSP430X to better support single-instruction multiple-data (SIMD) instructions.

With the MSP430X upgrade, some extended instructions were added, mostly supporting the new 20-bit operand size, and a few new instruction functionalities were added. Furthermore, the architecture was further expanded when ported to this chip to support all four addressing modes for the destination operand for use with extended instructions. This addition, along with the existing single-instruction multiple-data (SIMD) support in the

MSP430X, greatly increases the power of the SIMD instructions present in the MSP430X. Besides this change, some bits in the status register were re-defined to support the specific clock architecture present in the chip as well as a bit to support DVS.

Only a brief overview of the processor will be provided. For complete information, see the TI documentation [65].

4.3.1 Registers

The MSP430/X has 16 registers. Four are special purpose, the other 12 are general purpose registers. R0 is the program counter, R1 is the stack pointer, R2 is the status register/constant generator and R3 is a constant generator. R4 through R15 are general purpose. The PC and SP are always aligned at even addresses and their least-significant bit will always read zero. If an odd address is written to them, the least-significant bit will be cleared.

The processor can manipulate three different sized data: byte, word, and address. When storing bytes, it uses a little-endian format whereby the address increases with increasing numerical significance. Word-sized operands must be aligned on even addresses, while address-sized operands must also be aligned on even addresses. The four MSBs in the address-sized operand when stored in memory are stored in the LSBs of the next 16-bit word. For example, to store 0x54321 at address 0x10, 0x4321 would be at 0x10 and 0x0005 would be stored at 0x12.

4.3.1.1 Status Register

The status register is generally compatible with the TI parts. The only bits which have been re-defined are the ones relating to the low-power states and clock source. A bit is also used to implement DVS functionality. The complete register definition is shown below.

15	14	13	12	11	10	9	8
Unused							V
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

7	6	5	4	3	2	1	0
SCG1	SCG0	DVS_ON	CPU_OFF	GIE	N	Z	C
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

V Bit 8 Overflow bit. This bit is set when the result of an arithmetic operation overflows the signed-variable range. This is the same as the original MSP430/X.

SCGx Bits 7-6 System clock generator control bits. These bits select which clock source is being used for MCLK.

SCG1	**SCG0**	**Clock Used**
0	0	VFOSC/CFOSC Clk
0	1	DCO Clk
1	0	SMCLK
1	1	Clock Disabled

Note: When disabling the main clock by setting both SCG1 and SCG0, also set the CPU_OFF bit. Otherwise undefined instruction execution may take place.

Unlike the rest of the bits in the status register which are cleared when an interrupt occurs, the system clock generator control bits are loaded with a user-defined value at the beginning of an interrupt. The value which these are set to at the beginning of an interrupt is defined in the IntSCGR field in the clock generator control register.

Continued on next page

Continued from previous page

DVS_ON	Bit 5	DVS On. Controls which DVS value is being used to control the power supply. 0 - Value in register `DVS_SET0` is used. 1 - Value in register `DVS_SET1` is used.
CPU_OFF	Bit 4	CPU off. This bit, when set, turns off the CPU. This is the same as the original MSP430/X.
GIE	Bit 3	General interrupt enable. This bit, when set, enables maskable interrupts. This is the same as the original MSP430/X.
N	Bit 2	Negative bit. This bit is set when the result of an operation is negative, and is cleared when the result is not negative. This is the same as the original MSP430/X.
Z	Bit 1	Zero bit. This bit is set if the result of an operation is zero, and is cleared if not. This is the same as the original MSP430/X.
C	Bit 0	Carry bit. This bit is set when the result on an operation produced a carry. This is the same as the original MSP430/X.

4.3.1.2 Constant Generator

The constant generator produces some commonly-used constants which the software uses to improve code density and execution time. Some instructions paired with the constant generator are enumerated as separate instructions. For example, the instruction `add #1,rn` can use the constant generator for the first operand instead of storing it in the code. This decreases the instruction size and improves performance for this commonly used function. TI defines a new instruction for this operand as `inc rn`, though it is not actually another instruction in hardware. Instead it is an emulated instruction using the constant generator.

The particular constant generated depends on the register accessed (R2 or R3) and the addressing mode. Table 4.2 gives which constant is generated for each register and addressing mode combination.

The column ‘As’ is the value of the addressing mode select bits in the instruction word.

Table 4.2 – Constant generator

Reg.	As	Constant	Remarks
R2	00	-	Status Register
R2	01	(0)	Absolute address mode
R2	10	0x0004	+4
R2	11	0x0008	+8
R3	00	0x0000	0
R3	01	0x0001	+1
R3	10	0x0002	+2
R3	11	0xFF,0xFFFF,0xFFFFF	-1

4.3.2 Addressing Modes

TI defines seven different addressing modes for the processor, however in the original MSP430 hardware there are actually only four. The other three utilize one of the four addressing modes paired with a special register to create an apparently different addressing mode. The seven addressing modes are listed in Table 4.3. In the MSP430X however, all seven addressing modes are handled in hardware instead of emulated for a very few instructions where there is not enough space in the op-code map to fit the general index addressing mode.

In the MSP430/X architecture from TI, all seven addressing modes may be used as the source operand, but only the first four may be used as an addressing mode for the destination. In particular, there is no register indirect or register indirect post-increment. The register indirect mode can be emulated as `0(rn)`, however the index 0 must be stored in the word following the instruction, thereby decreasing code density and performance. This is not particularly grievous; however, when using the newly-added instruction repetition field in the MSP430X architecture, the indirect and indirect post-increment destination addressing modes are highly desirable for SIMD instructions. So, in this implementation for extended instructions, the destination also supports indirect and indirect post-increment addressing modes if the source operand is either register, indirect, or indirect autoincrement. This greatly increases the capabilities of the SIMD instructions. See subsection 4.3.5 for

Table 4.3 – Addressing modes

As/Ad	Addr. Mode	Syntax	Description
00/0	Register	Rn	Register contents are operand
01/1	Indexed	X(Rn)	(Rn + X) points to the operand. X is stored in the next word or a combination of the preceding extension word and the next word.
01/1	Symbolic	Addr	(PC + X) points to the operand. This is the pc-relative addressing mode. Emulated as X(PC)
01/1	Absolute	&Addr	The word following, or a combination of the preceding extension word and the following word, contains the absolute address. Emulated as X(SR)
10/-	Indirect Register	@Rn	Rn is used as a pointer to the operand.
11/-	Indirect Autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards by 1 for .B instructions, by 2 for .W instructions, and by 4 for .A instructions.
11/-	Immediate	#N	N is the value of the operand. N is stored in the next word of a combination of the preceding extension word and the next word. Emulated as @PC+

more information. For information on the instruction encoding when using these addressing modes, see subsubsection 4.3.6.1.

4.3.3 Interrupts

The original MSP430 processor has 16 interrupt sources. The processor in the first, second, and third chip also has 16 interrupts, however in the fourth chip this was changed to 32 interrupt sources. The 32 interrupt sources are listed in Table 4.4 The processor supports a two-level interrupt structure. The normal interrupts are maskable, meaning they are disabled by the GIE bit in the status register. The non-maskable interrupts cannot be globally masked (except for disabling the individual interrupt sources) and always interrupt the processor, no matter what is executing. The NMI interrupt can interrupt other ISRs or even the NMI ISR itself.

Table 4.4 – Interrupt Sources

Interrupt Source	**Type**	**Vector Address**	**Interrupt Number**
Power-up External Reset Watchdog Reset	Reset	0xFFFE	0
External NMI Internally forced NMI interrupt	Non-Maskable	0xFFFC	1
External dedicated IRQ line	Maskable	0xFFFA	2
Narrowband TX	Maskable	0xFFF8	3
Narrowband RX	Maskable	0xFFF6	4
Parallel I/O Controller	Maskable	0xFFF4	5
UWB TX	Maskable	0xFFF2	6
Timer 0	Maskable	0xFFF0	7
Timer 1	Maskable	0xFFEE	8
SPI 0	Maskable	0xFFEC	9
UART 0	Maskable	0xFFEA	10
Clock Generator	Maskable	0xFFE8	11
Charge Sensitive Amplifier	Maskable	0xFFE6	12
System IRQ	Maskable	0xFFE4	13
UWB RX	Maskable	0xFFE2	14
Delta Sigma ADC	Maskable	0xFFE0	15
SPI 1	Maskable	0xFFDE	16
UART 1	Maskable	0xFFDC	17
Timer 2	Maskable	0xFFDA	18
Timer 3	Maskable	0xFFD8	19

Continued on next page

Continued from previous page

Interrupt Source	**Type**	**Vector Address**	**Interrupt Number**
I^2C Master	Maskable	0xFFD6	20
I^2C Slave	Maskable	0xFFD4	21
Unused	Maskable	0xFFC0-0xFFD2	22-31

The maskable interrupts cannot interrupt another ISR unless the executing ISR explicitly enables interrupts within itself. In the case of simultaneous interrupts, there is a fixed priority among the maskable interrupts with the interrupts with the lower interrupt number in Table 4.4 having a higher priority.

When an interrupt is taken, the following actions take place:

1. The present status register is saved on the system stack.
2. If DVS is enabled in sleep state, the `DVS_ON` bit in the SR is cleared which restores the supply voltage from the value in `DVS_SET1` to that in `DVS_SET0`. While the power supply is changing from the sleep state to the active state, no clocks are enabled and no instructions are executed. If DVS is not used, this step is skipped.
3. The `SCGR` bits in the status register are set to the value of the `IntSCGR` field in the clock generator control register. This configures which clock is used during interrupts. The clock is started. If this is the internal DCO or one of the two RC-oscillators, this requires less than one clock cycle, if it is a crystal oscillator, this may take much time.
4. The remaining bits in the status register are cleared. This clears the `CPU_OFF` bit which enables the processor, clears the `GIE` bit which disables interrupts, and clears the flags.
5. The value of the PC is pushed on the system stack.
6. The value at the interrupt vector location is fetched from memory and stored in the program counter.
7. Execution begins at the address pointed to by the interrupt vector.

The major factor contributing to long interrupt latency is the power supply power-up and the oscillator start-up time if using a crystal oscillator. If neither are used, the interrupt latency will be only a few cycles from the interrupt being accepted to the ISR execution beginning.

When the ISR has finished executing, it executes a return from interrupt instruction. This instruction does the following:

1. Pops from the system stack the value of the PC.
2. Pops from the stack the value of SR. This returns the clock status, the `CPU_OFF` bit, the `GIE` bit, and the `DVS_SET` bit to their values before the interrupt, thereby placing the processor back into a possible sleep state.

Generally after an interrupt, the processor returns back to the same sleep state from whence it came before the interrupt. However, the ISR can modify the value of the SR on the stack so that when it returns, the modified SR will be loaded to the SR and the processor will be in a different sleep state. This can be useful to allow an ISR to wake the processor from a sleep state.

The above description overlooks the fact that for the MSP430X architecture, the PC is a 20-bit value, not 16. The most-significant 4 bits of the PC are stored in the most-significant four bits of the SR on the stack.

4.3.4 Instruction Set

Only a brief introduction overview of the instruction set will be given, for complete information, see the TI documentation. Table 4.5 gives a complete list of all the instructions of the MSP430X architecture. The original MSP430 architecture uses one 16-bit word for the instruction with optional data words afterwards, depending on the addressing modes used in the instruction. The original instructions are denoted in Table 4.5 by a blank in the 'Type' column. The MSP430X added extended versions of virtually every instruction by appending an extension word before it. The extended versions of the instructions have similar functionality, with the main difference being their support for 20-bit operands (using the .A size code) and their support of 20-bit addresses and 20-bit index values. Furthermore, some instructions were added to the existing op-code map to create new extended instructions without the extension word. The extended instructions are denoted in Table 4.5 by an "Ext." in the type column, and the extended instructions which do not require the extension word are denoted with an *. Using the instruction word slows down code execution speed due to the extra instruction fetch cycle required, and decreases code density. To alleviate this, some of the most commonly used extended instructions with the most common addressing modes, were implemented without the extension word. These instructions always operate on 20-bit data size, and are denoted in Table 4.5 by a “Adr." in the type column. These instructions only work on a very limited set of addressing modes, always use 20-bit operand size, but execute faster than their normal extended instruction counterparts.

There are 56 base instructions, however many of these can be combined with the constant generator to produce emulated instructions. Together with the emulated instructions a total of 99 instructions are listed in Table 4.5. These emulated instructions in Table 4.5 and are denoted by a †. Some instructions operate on varying operand size, as selected by the .B, .W, or .A size extension. The .W extension is always optional, as the assembler assumes it is word size if no size extension is specified.

Table 4.5 – MSP430X Instructions

Mnemonic					Description		Type
`ADC`†	`.B`	`.W`		`dst`	Add C to destination	$dst + C \rightarrow dst$	
`ADCX`†	`.B`	`.W`	`.A`	`dst`	Add C to destination	$dst + C \rightarrow dst$	Ext.
`ADD`	`.B`	`.W`		`src,dst`	Add source to destination	$src + dst \rightarrow dst$	
`ADDA`				`src,dst`	Add source to destination	$src + dst \rightarrow dst$	Adr.
`ADDX`	`.B`	`.W`	`.A`	`src,dst`	Add source to destination	$src + dst \rightarrow dst$	Ext.
`ADDC`	`.B`	`.W`		`src,dst`	Add source and C to destination	$src + C + dst \rightarrow dst$	
`ADDCX`	`.B`	`.W`	`.A`	`src,dst`	Add source and C to destination	$src + C + dst \rightarrow dst$	Ext.
`AND`	`.B`	`.W`		`src,dst`	AND source and destination	$src\,.AND.\,dst \rightarrow dst$	
`ANDX`	`.B`	`.W`	`.A`	`src,dst`	AND source and destination	$src\,.AND.\,dst \rightarrow dst$	Ext.
`BIC`	`.B`	`.W`		`src,dst`	Clear bits in destination	$\overline{src}\,.AND.\,dst \rightarrow dst$	
`BICX`	`.B`	`.W`	`.A`	`src,dst`	Clear bits in destination	$\overline{src}\,.AND.\,dst \rightarrow dst$	Ext.
`BIS`	`.B`	`.W`		`src,dst`	Set bits in destination	$src\,.OR.\,dst \rightarrow dst$	
`BISX`	`.B`	`.W`	`.A`	`src,dst`	Set bits in destination	$src\,.OR.\,dst \rightarrow dst$	Ext.
`BIT`	`.B`	`.W`		`src,dst`	Test bits in destination	$src\,.AND.\,dst$	
`BITX`	`.B`	`.W`	`.A`	`src,dst`	Test bits in destination	$src\,.AND.\,dst$	Ext.
`BR`†				`dst`	Branch to destination	$dst \rightarrow PC$	

Continued on next page

Continued from previous page

Mnemonic			Description		Type
`BRA`[†]		`dst`	Branch to 20-bit destination	$dst \rightarrow PC$	Adr.
`CALL`		`dst`	Call destination	$PC+2 \rightarrow stack, dst \rightarrow PC$	
`CALLA`		`dst`	Call 20-bit destination	$PC+2 \rightarrow stack, dst \rightarrow PC$	Ext.[*]
`CLR`[†]	`.B .W`	`dst`	Clear destination	$0 \rightarrow dst$	
`CLRA`[†]		`Rdst`	Clear 20-bit Rdst	$0 \rightarrow dst$	Adr.
`CLRX`[†]	`.B .W .A`	`dst`	Clear destination	$0 \rightarrow dst$	Ext.
`CLRC`[†]			Clear C	$0 \rightarrow C$	
`CLRN`[†]			Clear N	$0 \rightarrow N$	
`CLRZ`[†]			Clear Z	$0 \rightarrow Z$	
`CMP`	`.B .W`	`src,dst`	Compare source and destination	$dst - src$	
`CMPA`		`src,dst`	Compare source and destination	$dst - src$	Adr.
`CMPX`	`.B .W .A`	`src,dst`	Compare source and destination	$dst - src$	Ext.
`DADC`[†]	`.B .W`	`dst`	Decimally add C to destination	$dst + C \rightarrow dst(dec.)$	
`DADCX`[†]	`.B .W .A`	`dst`	Decimally add C to destination	$dst + C \rightarrow dst(dec.)$	Ext.
`DADD`	`.B .W`	`src,dst`	Add src. and C decimally to dst.	$src + dst + C \rightarrow dst(dec.)$	
`DADDX`	`.B .W .A`	`src,dst`	Add src. and C decimally to dst.	$src + dst + C \rightarrow dst(dec.)$	Ext.
`DEC`[†]	`.B .W`	`dst`	Decrement destination	$dst - 1 \rightarrow dst$	
`DECX`[†]	`.B .W .A`	`dst`	Decrement destination	$dst - 1 \rightarrow dst$	Ext
`DECD`[†]	`.B .W`	`dst`	Double-decrement destination	$dst - 2 \rightarrow dst$	
`DECDA`[†]		`Rdst`	Double-decrement Rdst	$dst - 2 \rightarrow dst$	Adr.
`DECDX`[†]	`.B .W .A`	`dst`	Double-decrement destination	$dst - 2 \rightarrow dst$	Ext.
`DINT`[†]			Disable interrupts	$0 \rightarrow GIE$	
`EINT`[†]			Enable interrupts	$1 \rightarrow GIE$	
`INC`[†]	`.B .W`	`dst`	Increment destination	$dst + 1 \rightarrow dst$	
`INCX`[†]	`.B .W .A`	`dst`	Increment destination	$dst + 1 \rightarrow dst$	Ext.
`INCD`[†]	`.B .W`	`dst`	Double-increment destination	$dst + 2 \rightarrow dst$	
`INCDA`[†]		`Rdst`	Double-increment Rdst	$dst + 2 \rightarrow dst$	Adr.
`INCDX`[†]	`.B .W .A`	`dst`	Double-increment destination	$dst + 2 \rightarrow dst$	Ext.
`INV`[†]	`.B .W`	`dst`	Invert destination	$\overline{dst} \rightarrow dst$	
`INVX`[†]	`.B .W .A`	`dst`	Invert destination	$\overline{dst} \rightarrow dst$	Ext.

Continued on next page

Continued from previous page

Mnemonic			Description		Type
`JC/JHS`		`label`	Jump if C set/Jump if higher or same		
`JEQ/JZ`		`label`	Jump if equal/Jump if Z set		
`JGE`		`label`	Jump if greater or equal		
`JL`		`label`	Jump if less		
`JMP`		`label`	Unconditional Jump	$PC + 2 \times \text{Offset} \rightarrow PC$	
`JN`		`label`	Jump if N set		
`JNC/JLO`		`label`	Jump if C not set/Jump if lower		
`JNE/JNZ`		`label`	Jump if not equal/Jump if Z not set		
`MOV`	`.B .W`	`src,dst`	Move source to destination	$src \rightarrow dst$	
`MOVA`		`src,dst`	Move source to destination	$src \rightarrow dst$	Adr.
`MOVX`	`.B .W .A`	`src,dst`	Move source to destination	$src \rightarrow dst$	Ext.
`NOP`†			No operation		
`POP`†	`.B .W`	`dst`	Pop item from stack to dst.	$@SP \rightarrow dst, SP + 2 \rightarrow SP$	
`POPM`	`.W .A`	`#n,Rdst`	Pop n registers from stack	$@SP+ \rightarrow dst$	Ext.*
`POPX`†	`.B .W .A`	`dst`	Pop item from stack to dst.	$@SP+ \rightarrow dst$	Ext.
`PUSH`	`.B .W`	`src`	Push src onto stack	$SP - 2 \rightarrow SP, src \rightarrow @SP$	
`PUSHM`	`.W .A`	`#n,Rsrc`	Push n registers onto stack	$src \rightarrow @ - SP$	Ext.*
`PUSHX`	`.B .W .A`	`src`	Push src onto stack	$src \rightarrow @ - SP$	Ext.
`RET`†			Return from subroutine	$@SP \rightarrow PC, SP + 2 \rightarrow SP$	
`RETA`†			Return from subroutine	$@SP \rightarrow PC, SP + 4 \rightarrow SP$	Adr.
`RETI`			Return from interrupt		
`RLA`†	`.B .W`	`dst`	Rotate left arithmetically		
`RLAX`†	`.B .W .A`	`dst`	Rotate left arithmetically		Ext.
`RLAM`	`.W .A`	`#n,Rdst`	Rotate left Rdst n bits arithmetically (n=1-4)		Ext.*
`RLC`†	`.B .W`	`dst`	Rotate left through carry		
`RLCX`†	`.B .W .A`	`dst`	Rotate left through carry		Ext.
`RRA`	`.B .W`	`dst`	Rotate right arithmetically		
`RRAM`	`.W .A`	`#n,Rdst`	Rotate right Rdst n bits arithmetically (n=1-4)		Ext.*
`RRAX`	`.B .W .A`	`dst`	Rotate right arithmetically		Ext.
`RRC`	`.B .W`	`dst`	Rotate right through carry		

Continued on next page

Continued from previous page

Mnemonic			Description		Type
`RRCM`	`.W .A`	`#n,Rdst`	Rotate right Rdst n bits through carry (n=1-4)		Ext.*
`RRCX`	`.B .W .A`	`dst`	Rotate right through carry		Ext.
`RRUM`	`.W .A`	`#n,Rdst`	Rotate right Rdst n bits unsigned (n=1-4)		Ext.*
`RRUX`	`.B .W .A`	`dst`	Rotate right unsigned		Ext.
`SBC`†	`.B. W`	`dst`	Subtract $\overline{\text{C}}$ from destination	$dst + 0\text{xFFFF} + C \rightarrow dst$	
`SBCX`†	`.B .W .A`	`dst`	Subtract $\overline{\text{C}}$ from destination	$dst + 0\text{xFFFF} + C \rightarrow dst$	Ext.
`SETC`†			Set C	$1 \rightarrow C$	
`SETN`†			Set N	$1 \rightarrow N$	
`SETZ`†			Set Z	$1 \rightarrow Z$	
`SUB`	`.B .W`	`src,dst`	Subtract source from destination	$dst + \overline{src} + 1 \rightarrow dst$	
`SUBA`		`src,dst`	Subtract source from destination	$dst + \overline{src} + 1 \rightarrow dst$	Adr.
`SUBX`	`.B .W .A`	`src,dst`	Subtract source from destination	$dst + \overline{src} + 1 \rightarrow dst$	Ext.
`SUBC`	`.B .W`	`src,dst`	Subtract src. and $\overline{\text{C}}$ from dst.	$dst + \overline{src} + C \rightarrow dst$	
`SUBCX`	`.B .W .A`	`src,dst`	Subtract src. and $\overline{\text{C}}$ from dst.	$dst + \overline{src} + C \rightarrow dst$	Ext.
`SWPB`		`dst`	Swap bytes		
`SWPBX`	`.W .A`	`dst`	Swap bytes (.A version leaves bits 19:16)		Ext.
`SXT`		`dst`	Extend sign	$dst[7] \rightarrow dst[8:15]$	
`SXTX`	`.W .A`	`dst`	Extend sign	$dst[7] \rightarrow dst[8:MSB]$	Ext.
`TST`†	`.B .W`	`dst`	Test dst. (compare against zero)	$dst + 0\text{xFFFF} + 1$	
`TSTA`†		`Rdst`	Test destination register	$dst + 0\text{xFFFF} + 1$	Adr.
`TSTX`†	`.B .W .A`	`dst`	Test dst. (compare against zero)	$dst + 0\text{xFFFF} + 1$	Ext.
`XOR`	`.B .W`	`src,dst`	Exclusive OR source and dst.	$src\,.XOR.\,dst \rightarrow dst$	
`XORX`	`.B .W .A`	`src,dst`	Exclusive OR source and dst.	$src\,.XOR.\,dst \rightarrow dst$	Ext.

† Emulated instructions.

* Extended instruction not using the extension word.

Noticeably missing are any multiply or divide instructions. The multiply and multiply-accumulate can be done relatively quickly with the hardware multiplier module (described in section 4.5), but alas, there is no quick way of doing division.

4.3.5 Single-Instruction Multiple Data Instructions

The extension word used before the standard MSP430 instructions to make them the extended instructions has a instruction repetition field in some cases. There are actually two formats for the extension word. One format is used when one or both operands use either the indexed or immediate addressing modes. In this mode, the most significant four bits of either the immediate constant or the index value or both is stored in the extension word, and little space is available for other uses. However, if both the source and destination use a register or register indirect/postincrement, then there is much space available in the extension word. In the MSP430X architecture, an instruction repetition field was defined which allows the instruction to be repeated a number of times.

Therefore, almost any extended instruction can be repeated up to 16 times. The number of repetitions can either be fixed at compile time, or can be run-time adjustable by making the repetition count be set by the least-significant four bits of a CPU register.

However, the original MSP430X architecture does not allow the destination addressing mode to be register indirect post-increment. This particular addressing mode is very useful in this situation because most useful SIMD instructions need the destination register to automatically increment to point to the next address in memory after each operation. So a simple example could be `movx.w @Rn+, @Rm+, 16` which would move a block of 16 words from the location Rn to Rm. However, without the destination post-increment addressing mode, this is not possible.

For this reason, An extension was added to the processor to support indirect and indirect post-increment addressing modes for the destination operand when using the register extension word. This takes two bits in the extension word which were defined to always be zero by TI and re-defined them to be the destination addressing mode override. This unlocks many useful SIMD operations which allow higher performance.

On another note, a couple features which are present in the register mode extension word (the instruction repetition being the most important) are not available from the assembly syntax as published by TI. This means there is no way to use the instruction repetition field present in the MSP430X architecture without resorting to writing the extension word in machine code! This fact was confirmed with TI. The only way to use the repetition field is to define the relevant extension words and insert them before the desired instructions. This is not convenient when writing example instructions; therefore, in this document, whenever an assembly instruction is referenced with a repetition count, it will be included as a number after the list of normal arguments. This is not standard TI syntax and will produce an assembler error if compiled.

The other feature which is only available through this method is the use of the zero carry bit in the extension word. For information about the encoding of the extension word see subsubsection 4.3.6.1

Using the instruction repetition feature can increase interrupt latency due to the long duration of the repeated instructions. Interrupts cannot occur in between instruction repetitions. The longest possible instruction, the `addx.a @Rn+, @Rm+, 16` (or similar) requires two cycles to read the instruction, 32 cycles to read the 16 source operands, 32 to read the destination operands, and 32 to write the result for a total of 98 cycles. Of course, if a word-sized instruction is used, the number of cycles is only 50.

4.3.6 Instruction Encoding

For the original MSP430, the instruction encoding was very simple. However, for the MSP430X, the added instructions and the extension word were added somewhat haphazardly due to the fact that the extensions were not planned in the original MSP430. For the original MSP430, there are a total of 27 instructions (not including the emulated instructions) which are encoded in three formats. The three formats are shown in Table 4.6.

Table 4.6 – Instruction encoding.

Type	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Single	0	0	0	1	0	0	Opcode			Bs	Ad		Dst. reg			
Jump	0	0	1	Condition			10-bit PC Offset									
Double	Opcode				Src. reg				Ad	Bs	As		Dst. reg			

The field “Bs" is a one if a byte-sized operation is performed or a zero if a word-sized operation is performed. Ad and As are the destination and source addressing modes.

There are a total of seven single-operand instructions. Their op-codes are shown in Table 4.7.

There are eight conditions for the jumps. When taken, the PC-relative offset is doubled, sign-extended, and added to the PC to calculate the jump location. This gives a jumping range of -1024 to +1022. The jump conditions are given in Table 4.8.

Note, there is no jump if N=0.

Table 4.7 – Single-operand instruction op-codes.

Opcode	Instr.	Description
000	RRC	Rotate right through carry.
001	SWPB	Swap 8-bit register halves; no byte form.
010	RRA	Rotate right arithmetically (signed).
011	SXT	Sign extend 8-bit number to 16-bit. No byte form.
100	PUSH	Push value onto stack and de-increment stack pointer.
101	CALL	Push present PC onto stack, fetch operand and begin execution at that location. No byte form.
110	RETI	Pop SP then pop PC. No byte form.
111	None	None for the original MSP430; however, in the MSP430X this (as well as most of the space of the previous RETI instruction) is the CALLA instruction.

Table 4.8 – Jump conditions.

Condition	Instruction	Description
000	JNE/JNZ	Jump if Z=0, or if not equal.
001	JEQ/JZ	Jump if Z=1, or if equal.
010	JNC/JLO	Jump if C=0, or unsigned $<$.
011	JC/JHS	Jump if C=1, or unsigned $\geq$.
100	JN	Jump if N=1, or if negative.
101	JGE	Jump if N=V, or if signed $\leq$.
110	JL	Jump if N$\neq$V, or if signed $<$.
111	JMP	Jump unconditionally.

There are 12 double operand format instructions. These and their op-codes are listed in Table 4.9.

The preceding information completely describes the instruction encoding for the original MSP430 architecture which is used in the first, second, and third chips. For the

Table 4.9 – Double-operand instruction op-codes.

Opcode	Instr.	Description
0100	MOV	Move source to destination; status flags not set.
0101	ADD	Add source and destination.
0110	ADDC	Add source, destination, and carry.
0111	SUBC	Subtract source from destination with barrow.
1000	SUB	Subtract source from destination.
1001	CMP	Subtract source from destination, but do not write result anywhere, just update the status flags.
1010	DADD	Add source, destination, and carry decimally (BCD).
1011	BIT	AND source and destination, however do not write result anywhere, just update the status flags.
1100	BIC	AND $\overline{\text{source}}$ and destination; status flags not set.
1101	BIS	OR source and destination; status flags not set.
1110	XOR	XOR source and destination
1111	AND	AND source and destination

MSP430X architecture, extra instructions were added. One class of instructions are simply created by preceding the standard instruction with an extension word to create an extended instruction. Virtually every MSP430 instruction then has an associated extended instruction. Furthermore, more instructions were added without the use of the extension word by filling in the gaps in the original MSP430 op-code space. These instructions, which are the address instructions and the extended instructions denoted by an * in Table 4.5, are not systematic in their encoding. For further information about the encoding of these instructions, see [65].

4.3.6.1 Extension Word

The extension word which is placed before a standard instruction to make it an extended instruction has two formats. If one of the operands in the instruction is using either indexed addressing or immediate addressing, then the extension word contains the most-significant four bits of either the index or the immediate value. In this case, the instruction word format is shown in Table 4.10 where the A/L bit is the same as it is in the register mode extension word.

Table 4.10 – Non-register mode extension word.

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	1	1	Src bits [19:16]				A/L	0	0	Dst bits [19:16]			

If the instruction only uses register mode addressing modes (including indirect or indirect post-increment), the extended instructions also support an optional instruction repetition field, as well as a “Zero Carry" flag which zeros the carry before use. Another feature added in the chip but not present in the TI variant of the MSP430X architecture, is the ability to use the indirect and indirect post-increment addressing modes for the destination register. These two addressing modes only work when using the register mode extension word. These two extra addressing modes are particularly useful when used in conjunction with the instruction repeat function, thereby creating single-instruction multiple-data instructions. The register-mode extension word is given below.

15	14	13	12	11	10	9	8
0	0	0	1	1	Ad		ZC

7	6	5	4	3	2	1	0
#	A/L	0	0	(n-1)/Rn			

Continued on next page

Continued from previous page

Ad Bits 10-9 Destination addressing mode select. These bits are defined to be 00 by the TI documentation, but are re-defined by the chip to support all four addressing modes as the destination addressing mode. The chip re-defines them as follows:

Ad:	**Addressing Mode:**
00	Mode as defined in the instruction word bit 7
01	Undefined
10	Indirect register
11	Indirect autoincrement

ZC Bit 8 Zero carry bit. This is the same as the original MSP430X.

0 - The executed instruction uses the C bit.

1 - The C bit is cleared before instruction execution.

Bit 7 Repitition bit. This is the same as the original MSP430X.

0 - The number of instruction repetitions is set by the extension word bits 3:0.

1 - The number of instruction repetitions is set by bits 3:0 of the register Rn, where (n-1) is bits 3:0 of the extension word.

A/L Bit 6 Data length extension bit. Together with the B/W bit (bit 6 of the instruction word), this defines the size of the operand. This is the same as the original MSP430X.

A/L	**B/W**	**Size**
0	0	Reserved, undefined
0	1	20-bit operand
1	0	16-bit operand
1	1	8-bit operand

Continued on next page

Continued from previous page

(n-1)/Rn	Bits 3-0	Repitition count. This is the same as the original MSP430X. #=0 - These four bits set the repetition count n, they are n-1. #=1 - These four bits define the CPU register whose bits 3:0 set the repetition count. The LSBs of the CPU register are n-1.

However, even in the original MSP430X architecture, the instruction repetition field and zero carry field are not settable via the assembler syntax. To use these features, software must manually define the requisite extension words and insert them before normal MSP430 instructions. This is also true for the added addressing modes for the destination operand.

4.3.7 Implementation

The processor core consists of two main parts, the control section and the datapath. The datapath consists of all the registers, buses, and the ALU while the control section is responsible for instruction decoding and instruction execution. The vast majority of the area is required by the datapath. The datapath for the MSP430X architecture is shown in Figure 4.9. The datapath for the MSP430 is similar except it is only 16 bits wide and does not include the “Temp Read" and “Temp Write" registers and their associated multiplexer.

The “Temp Read" and “Temp Write" registers are used to read and write the 20-bit operands from the 16-bit memory. When reading, the “Temp Read" register stores the first 16 bits which are read from the memory first. In the following cycle, the most-significant 4 bits are combined and used as the complete operand. The “Temp Write” serves a similar function for writes.

The “Temp Src” and “Tmp Dst” registers are not programmer-accessible registers but are used for temporary storage during instruction execution. Mainly these are used for storage of the calculated address when using the indexed addressing mode.

The adder in the upper right corner is mainly used for incrementing the register addresses such as the PC, or any other register when using register indirect postincrement mode. It can add one, two, or negative two. It also calculates the new address to jump do when executing a jump.

The ALU can do all of the operations required by the op-codes such as add, decimally

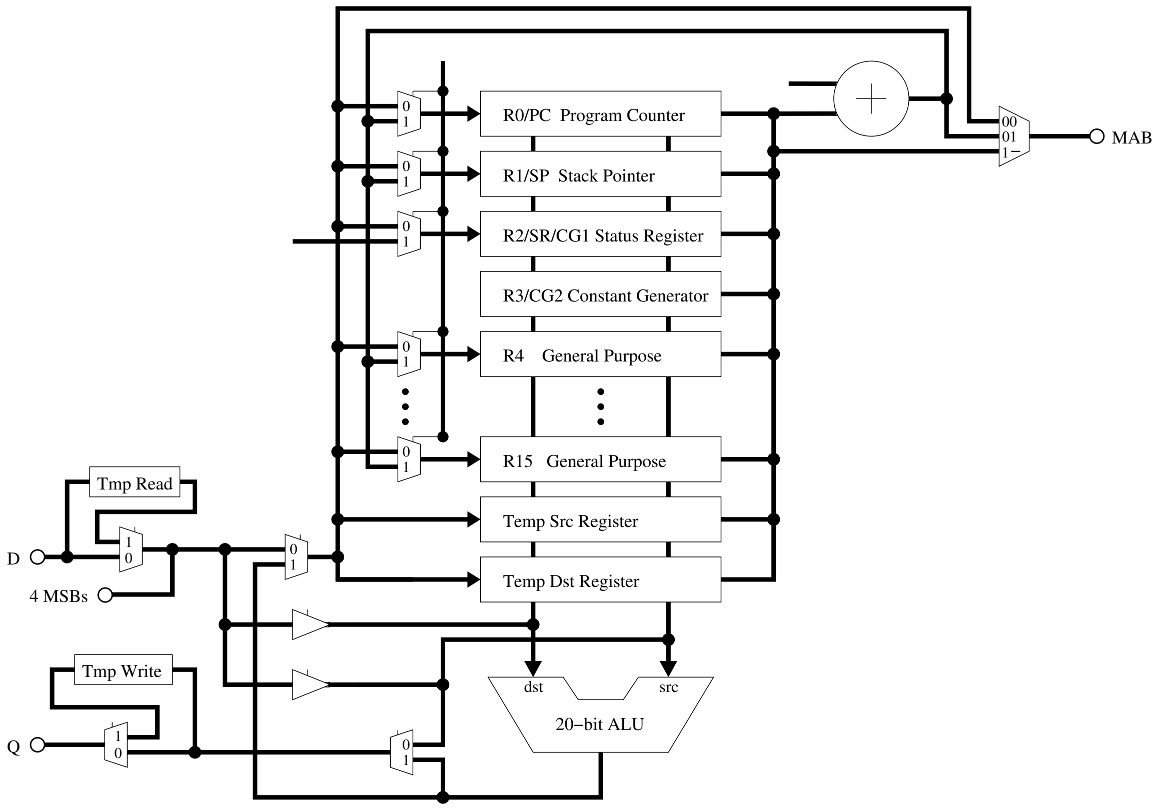


Figure 4.9 – MSP430X Datapath

add, subtract, AND, OR, XOR, rotates and shifts, and the like. There is also an internal bypass which drives out the "src" bus on the output.

Compared to the datapath which has stayed relatively consistent over the chip revisions, the control section has changed much. The control section consists of a large mealy finite state machine and the instruction register. In the first chip, the control section was a large 28-state state machine. On the re-design between the first and second chips, the number of states was greatly decreased so the second and third chips use an 11-state machine. With the MSP430X architecture, six extra states were added to support the extended instruction set.

Figure 4.10 shows the state diagram as present in the second and third chips. The MSP430X state diagram is similar, just with more states mainly to handle the reading and writing of the extra four bits when accessing memory.

The "start" state is where instruction execution begins. If it is a register-to-register operation which always complete in a single cycle, the next state is also the start state. In the start state, the instruction register is loaded. The "pre-start" state is also shared among

Table 4.12 – CPU control logic states.

State	Description
Reset	Inital reset dummy state, no operation is performed in this state. The FSM resets to this state.
Pre-start	Instruction finishing state. If the last instruction has a data write-back stage, it is performed while the address of the next instruction to be executed is driven on the bus in preparation for executing it in the next state.
Start	Main instruction execution state. Instruction execution may take place in this state.
Source Index Address Calculation	Fetches the index value from program, computes source operand's address and stores this in the temporary source register.
Fetch Source Operand From Memory	Fetches the source operand from memory. If the destination is a register, instruction execution takes place.
Dest. Index Address Calculation	Fetches index from program memory, computes the destination operand's address and stores this in the temporary destination register.
Fetch Dest. Operand From Memory	Fetches the destination operand from memory. Instruction execution takes place in this state.
Reti 0	Return from interrupt instruction state 0. Pop the SR from the stack
Reti 1	Return from interrupt instruction state 1. Pop the PC from the stack
Int 0	Enter interrupt state 0. Push the PC on the stack and set the PC to the value at the interrupt vector.
Int 1	Enter interrupt state 1. Push the SR on the stack and clear the SR.

many types of instructions. In this state, generally the data from the previous instruction is written back to memory while the address of the next instruction is driven on the address bus for execution in the next cycle.

Table 4.12 gives a brief description of each of the states.

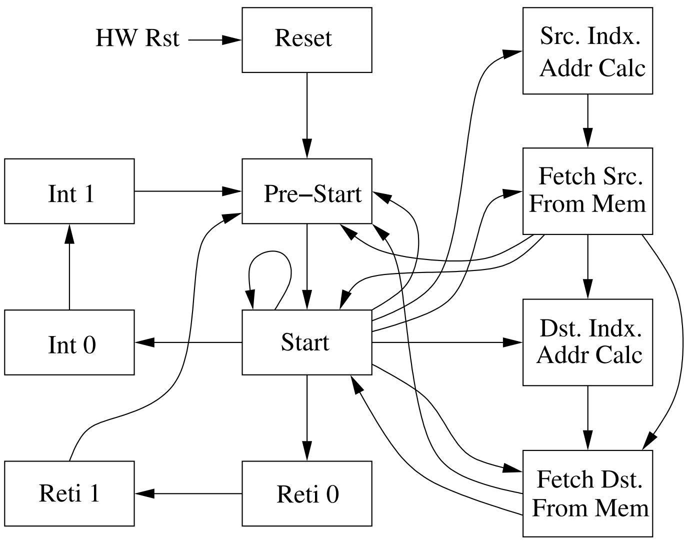


Figure 4.10 – CPU control section state diagram.

The state transitions are controlled by the type of instruction being executed and the addressing modes used. Table 4.13 shows the possible state transitions and the type of instructions which cause them. In the table, all double operand instructions are referred to as double instructions and single operand instructions are referred to as single instructions.

The control for the MSP430X is similar but more complicated with the additional states. The instruction repetition also complicates matters. For brevity the detailed FSM for the MSP430X will not be presented.

Table 4.13 – State Transitions

Start State	**End State**	**Condition**
Reset	Pre-Start	Always.
Pre-Start	Start	Always.

Continued on next page

Continued from previous page

Start State	**End State**	**Condition**
Start	Start	When all operands use the register addressing mode. Also for all jumps.
	Src. Indx. Addr. Calc.	Single or double instructions which use the indexed addressing mode for their source operand.
	Fetch Src. From Mem.	Single or double instructions which use register indirect or indirect post-increment addressing mode for their source operand.
	Dst. Indx. Addr. Calc.	Double instructions which have the source as register mode and the destination is indexed.
	Fetch Dst. From Mem.*	Double instructions which have the source as a register mode and the destination is indirect or indirect post-increment.
	Reti 0	Always with a return from interrupt instruction.
	Int 0	Always when an enabled interrupt request is detected.
	Pre-Start	Push or Call instruction, or a normal instruction with all operands as register mode and the instruction affects the PC. (i.e. `mov rn,PC`).
Src. Indx. Addr. Calc.	Fetch Src. From Mem.	Always.
Fetch Src. From Mem.	Pre-Start	Single instruction or double instruction and the destination is the PC.
	Start	Double instruction and the destination is a register but not the PC.
	Dst. Indx. Addr. Calc.	Double instruction and the destination is indexed.
	Fetch Dst. From Mem.*	Double instruction and the destination is indirect or indirect post-increment.

Continued on next page

Continued from previous page

Start State	End State	Condition
Dst. Indx. Addr. Calc	Fetch Dst. From Mem.	Always.
Fetch Dst. From Mem.	Pre-Start	All instructions except `bit` or `cmp`.
	Start	`bit` or `cmp` instructions.
Reti 0	Reti 1	Always
Reti 1	Pre-Start	Always
Int 0	Int 1	Always
Int 1	Pre-Start	Always

* Note: This transition is not possible in the original MSP430/X architecture, however it is used with the added extensions.

4.4 CRC Module

The chip incorporates a hardware CRC generation module using the CRC-CCITT 16-bit CRC. It easily allows the one to flip the order of the bits before generating the CRC to support the normal as well as the bit-reversed version. The output can also be read in a bit-reversed form. The initial state of the CRC module can be set to an arbitrary value. The CRC module is identical to the one found in some versions of the original MSP430, with the exception of the register addresses.

The CRC polynomial is given by

$$f(x) = x^{16} + x^{12} + x^5 + 1. \tag{4.2}$$

The implementation supports shifting the data in either LSB first or MSB first through the use of two registers which have their bits reversed.

As in the original MSP430, The CRC generation core is a large XOR tree which processes 8 bits of data at once. The processor can either write the data byte-wise using byte-sized instructions, or word-sized. If 16 bits of data are written at once, the first 8 bits are processed that cycle, while the second 8 bits are processed in the following cycle. Using the instruction repetition field in the MSP430X CPU architecture, it is possible to write a word of data to the CRC module every cycle which is faster than the CRC module can handle. For example, the instruction `movx rn,@rm 6` where the instruction is repeated six times and the register rm holds the address of the CRCINIRES register, will write a word of data (the data in rn) in six consecutive cycles, which is faster than the CRC module can process the data. However, the instruction `movx @rn+,@rm 6` will correctly compute the CRC over a block of six words stored in memory. Note, both of the aforementioned examples make use of the indirect addressing mode for the destination register, and are therefore not legal with the original MSP430X architecture. Furthermore, the assembler will not recognize them as valid syntax (neither the destination addressing mode, nor the instruction repetition field), so to actually use them, you must manually define the correct extension word and place it before a `mov rn,rm` or `mov @rn+,rm` instruction.

The CRC module is only present in the fourth chip and none of the previous chips.

4.5 Hardware Multiplier

The chip incorporates a hardware multiplier capable of doing 16×16 bit multiplication with a 32 bit result. It also supports a multiply accumulate function with a 32 bit accumulator size. The module supports both signed and unsigned multiplication and multiply accumulate. The hardware multiplier is code-compatible with the hardware multiplier implemented in some TI MSP430 parts. For a complete description of its operation, refer to the TI documentation [65].

The hardware multiplier module is one clock cycle faster than the one implemented in the original MSP430. The module can do a multiply in two cycles as opposed to three for the original one. As such, it is never necessary for software to delay one cycle before reading the result as is sometimes required with the original implementation.

To use the hardware multiplier, the first operand is moved to one of four registers which select the function. These registers are given in Table 4.14. All of these registers map to the same physical register, generically called OP1. After the mode is selected, the second operand is written to the OP2 register which also initiates the multiplication. The next instruction the processor can read the result from RESHI and RESLO. For the multiply-accumulate operation, the 32-bit accumulate input is the RESHI and RESLO registers. The processor does not have to reload OP1 if not desired, it only needs to put new data in OP2 to initiate another multiplication. The register addresses and definitions are the same as the original MSP430.

Table 4.14 – Hardware multiplier register function select.

Register	**Function**
MPY	Multiply unsigned.
MPYS	Multiply signed.
MAC	Multiply accumulate unsigned.
MACS	Multiply accumulate signed.

The input operand registers (`OP1` and `OP2`) are 16 bit registers. It is possible to write to these registers with a byte write operation. In this implementation, when a byte write operation is used to write to the register, the module will either sign-extend or zero-extend the value to 16 bits before writing the 16 bit value to the register. The selection of either zero extend or sign-extend is selected by weather a signed or unsigned operation is selected. A byte write operation can only be used to write to the low byte of the register; executing a byte write to the high byte of the register will result in unpredictable data in the low byte.

The result registers (`RESHI` and `RESLO`) can be written with new values. These new values will then be incorporated as the accumulate term in any further MAC operations.

As per the original MSP430, care must be taken when using the hardware multiplier in interrupt service routines. In particular, if an interrupt is executed which uses the hardware multiplier in between the loading of `OP1` and `OP2`, the resulting data will be wrong. For this reason, if the hardware multiplier is used in interrupts, interrupts must be disabled before using the hardware multiplier and re-enabled at the end. Compilers will generally perform this automatically. However, a special case exists with the non-maskable interrupts. These cannot be disabled via the simple `dint` instruction and the compiler will not know how to automatically disable the offending NMI, so this must be disabled manually. The best option however is not to use any multiplications in the NMI service routine which solves the problem.

The multiplier core is a booth-encoded hybrid multiply-add unit with a Dadda adder tree. It takes advantage of the optimized booth encoder and booth selector cells present in the ARM cell library produce the smallest area. The highly-parallel nature of the multiplier easily allows for single-cycle operation. The schematic was drawn manually using cells in the library, after which the verilog netlist was included in a VHDL wrapper file which includes the bus interface. Due to the possibility of human error in drawing the schematic, an automated VHDL testbench was created which randomly generates inputs and checks that the output is correct. Tens of thousands of multiplications were carried out in the various modes (signed and unsigned) to verify functionality before the netlist was incorporated into the design.

The hardware multiplier is included in the third and fourth chips only.

4.6 Timers

The chip integrates a number of independent timers. Each timer consists of a clock selector and prescaler, a 16-bit timer, three compare units, and two capture units. The three independent compare units can be utilized to create three independent software delays or periodic interrupts for each timer, or to generate two PWM waveforms. The timers may operate on the SMClk, the MClk, or either crystal oscillator clock. If operating on the low-frequency crystal oscillator clock, the frequency of the oscillator may be doubled by setting the TmrOscX2 bit in the clock generator control register. A functional diagram of a single timer module is shown in Figure 4.11.

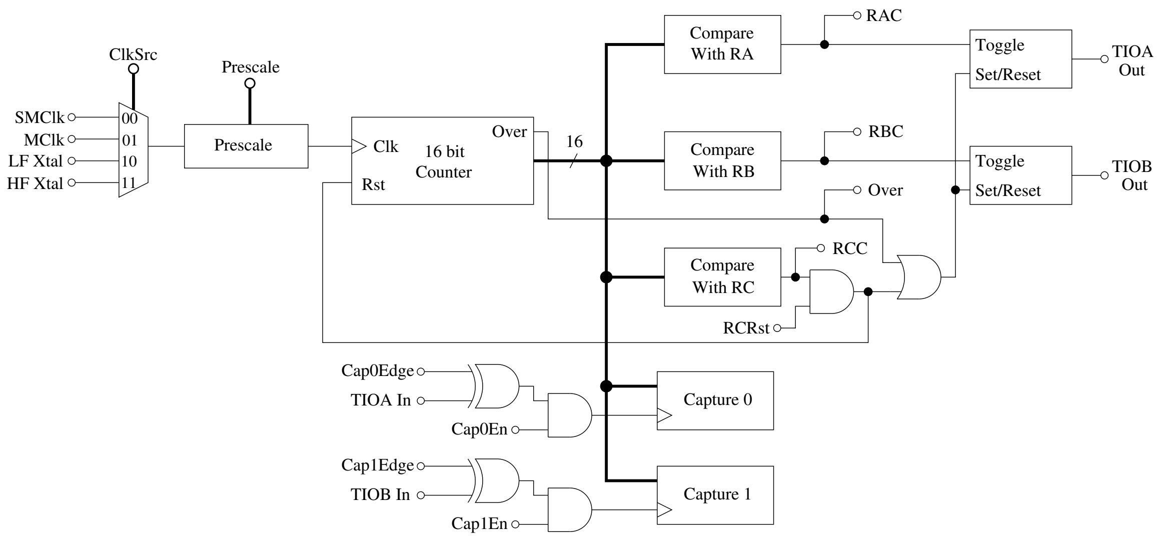


Figure 4.11 – Timer functional diagram

The input multiplexer selects the operating clock for the timer. The following prescaler can either leave the clock unchanged, or divide the clock by up to 2048. The value of the clock prescaler is chosen by the Prescale field in the timer status register. Clearing the timer enable bit in the timer control register disables the clock at the input to the prescaler. The 16 bit counter is a synchronous up-counter. The microcontroller has read and write access to the timer value via the timer counter register. The timer can either operate in a free-running mode where it counts to 0xFFFF, or instead reset on a compare with RC. The reset on RC compare gives the timer a count-to-n mode of operation.

The three compare modules can produce independent software delays, or can produce two PWM waveforms. The waveforms are synchronous to each other with no dead time. the frequency of the waveforms can be adjusted via the RC compare, while the duty cycle can be adjusted via the RA and RB registers. In this mode, the timer I/O pins are configured as outputs.

Two capture modules are also present which can be used for determining the frequency or period of external signals. On the programmed edge of an external signal, the counter value will be latched into a capture register. It is the responsibility of the microcontroller to read the latched data before another external event occurs and a new value is latched into the capture register. When operating in this mode, the timer I/O pins are configured as inputs. The capture module 0 is associated with TIOA and capture module 1 is associated with TIOB.

The functionality of the compare and capture modules are independent and can be used simultaneously if the compare module is only used for software interrupts. For example, capture module 0 can be enabled, therefore using the TIOA pin as an input, and the RA compare module can be used as a software timing-based interrupt. Furthermore, TIOB could be used in PWM mode. This allows some applications such as applying a stimulus to a circuit and measuring the response time. For example, an integrating type ADC could be made with an external op-amp and comparator.

The three compare outputs, the capture event status, the status of the timer I/O pins, and the overflow output are available in the status register. They are latched and are cleared by any write to the status register. Interrupts can be enabled for any event via the associated bits in the control register.

The count value for timer zero is driven to the UWB receiver where it is latched on the correct reception of a UWB packet preamble. This can be used to aid in synchronizing a transmitter and receiver.

The multiplexer to select the clock, as well as the prescaler multiplexer are not glitch-free. When changing clock sources or prescale values with the timer enabled, it is possible for a glitch prorogate to the timer and cause unpredictable values to get latched to the counter.

The timers for the first, second, and third chips did not have the capture module. They only had the compare module. Furthermore, they did not have the PWM generation capabilities. Therefore they were strictly for software interrupt generation without the ability to control an external pin. In those chips there were also only two timer modules.

4.7 Serial Peripheral Interface

The fourth chip integrates two independent SPI modules while the previous chips integrated one. This is a basic serial interface which sends and receives serial data simultaneously. It requires four wires, the clock, master in slave out (MISO), master out slave in (MOSI), and a dedicated chip select for each slave. All slaves share the other three lines. This module does not implement hardware support for the slave chip select lines, so they must be connected to a general purpose I/O line and the slaves must be selected and de-selected in software.

To initiate a bus transfer, data is written to the SPI_TDR register. If no bus transfer is ongoing, the data is immediately transferred to the transmit seralizer and shifted out one bit at time while shifting in data to the receive de-seralizer. At the end of a transfer, the data in the receive de-seralizer is transferred to the SPI_RDR and the RDRF flag is set in the SPI_SR. If the RDRF flag was already set then the OVER flag is set. The RDRF and OVER flags are cleared by any access to the status register.

If data is written to SPI_TDR while a bus transaction is ongoing, the data remains in the transmit data register until the completion of the ongoing bus transaction at which point the new data is loaded to the seralizer and transmitted.

The chip supports serial data rates of up to SMClk/2 and 8 and 16 bit transfer formats. The chip is always the master (i.e. supplying the clock) and cannot act as a slave. The device supports all SPI modes to interface with a variety of peripherals. The transfer modes are shown in Figure 4.12 and Figure 4.13.

The transfer modes are shown in Figure 4.12 and Figure 4.13 assume that there is no data to transmit immediately after the first byte of data. If another byte of data was written to the SPI_TDR register during transmission of this byte, the SPI clock would maintain its duty cycle and immediately begin transmitting the second byte after the first with no break

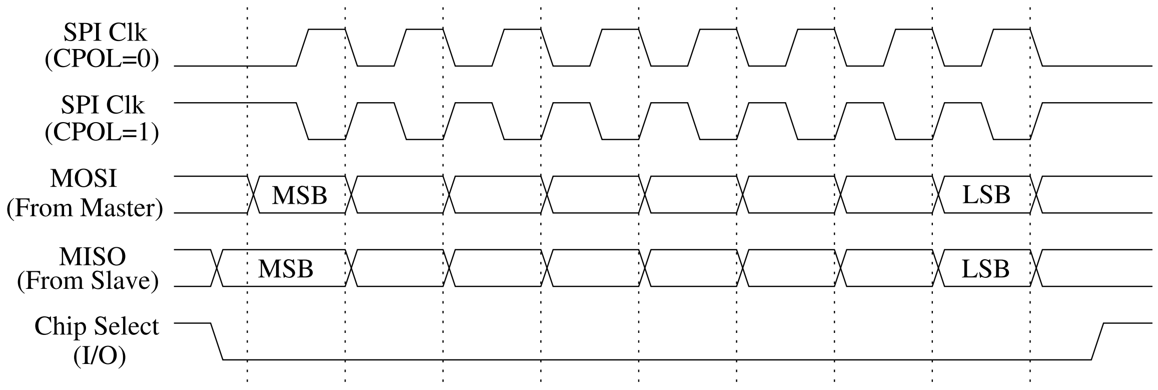


Figure 4.12 – SPI transfer format, CPHA=0

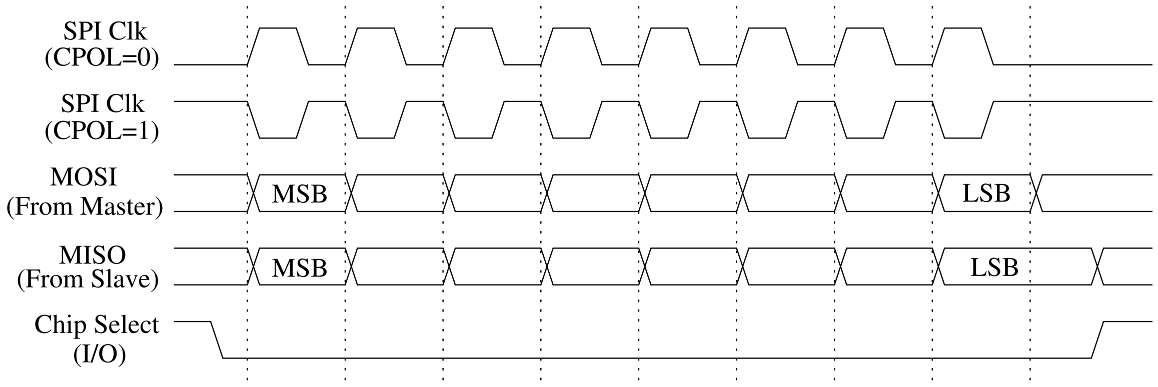


Figure 4.13 – SPI transfer format, CPHA=1

in the clock waveform. The slave select line shown in Figure 4.12 and Figure 4.13 is not generated by the SPI module and must be generated in software.

The SPI module always receives data when it is transmitting data. Therefore, if one is writing to a device, dummy data will be received which can be ignored. If reading from a device, dummy data must be transmitted to the slave in order to receive the data. Interrupts can be generated either on successful transmission or successful reception. A program can trigger on either of these two interrupts, and they will generally trigger simultaneously. The main difference between them is the receive-based interrupts can be cleared by reading from the status register, while the transmit interrupts are cleared by writing to the transmit data register (and therefore initiating another bus transaction).

4.8 Universal Asynchronous Receiver Transmitter

The fourth chip integrates two UART serial communication interfaces, while the previous chips integrated one. Each module supports full duplex asynchronous communication. They support hardware generation and checking of an optional parity bit.

The transmitter and receiver share a common baud rate. The baud rate generator consists of an 11 bit divide-by-n clock module, after which the transmit clock is divided by 16. The receive clock is 16 times the baud rate because the receiver samples the data line at 16 times in each bit interval and uses a majority vote to determine the actual value of the bit. The clock is disabled before the baud rate generator if neither the transmitter nor the receiver are active.

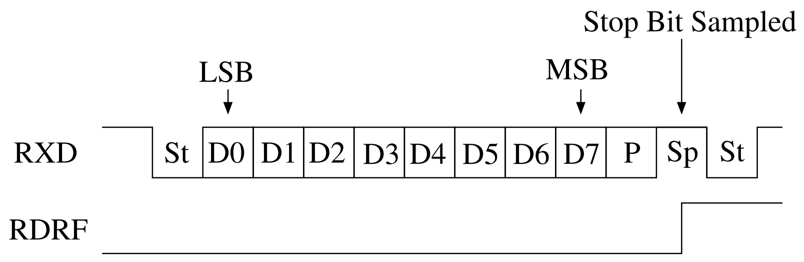


Figure 4.14 – UART receive byte

An example of the UART receiving a character is shown in Figure 4.14. The data format of the UART is shown. It consists of a start bit which is always low followed by 8 data bits, with the LSB first, followed by an optional parity bit and finally a stop bit which is always high.

Before reception, the UART receiver is sensitive to a falling edge of the RXD line. On a falling edge, it begins reception and after a single bit time it checks that the received bit is a zero, and therefore a valid start bit. If not, it terminates the reception. Otherwise, it receives the rest of the bits and when it half way through the stop bit it samples the RXD line to determine if a framing error has occurred. At this point, midway through the stop bit, it loads the receive data register (`UART_RDR`) with the received data, sets the flag (`RDRF`) and overflow flag (`OVER`) if necessary, and terminates the reception. Then it goes back to looking for an RXD falling edge.

A data transmission is started by writing data to the transmit data register (`UART_TDR`). If no transmission is ongoing, the data is immediately loaded from the TDR to the transmit data seralizer for transmission. After the start bit, data, optional parity, and stop bit are transmitted, the transmitter either immediately begins transmitting another byte if one was written to the TDR, or stops transmitting, setting the TXEMPTY bit.

Interrupts can be enabled for a number of different sources from UART_CR. The status of the module can be read from the UART_SR.

4.9 Inter-IC Bus (I^{2}C)

The I^{2}C module consists of two independent parts, each with the ability to transmit or receive data. These two parts, the master and the slave, operate independently. The master module acts as an I^{2}C bus master, and initiates all transactions and supplies the clock waveform. The slave module cannot initiate transactions and accepts the clock waveform from an external master. The I^{2}C master integrates all the functionality required by the I^{2}C bus specification.

The I^{2}C module can operate in one of the following 4 modes:

- Master mode, transmit data to slave.
- Master mode, receive data from salve.
- Slave mode, transmit data to master.
- Slave mode, receive data from master.

Each of the modes of operation will be described in the following sections. For the following descriptions, the following abbreviations are used:

- **A** - Acknowledge
- **NA** - Negative Acknowledge
- **P** - Stop
- **S** - Start
- **Sr** - Repeated Start
- **SADR** - Slave Address
- **ADR** - Any other address

- **R** - Read
- **W** - Write

4.9.1 Transfer Format

The I^{2}C bus uses two wires for communication, a clock line (SCL) and data line (SDA). Both are open-drain I/O pins with pull-up resistors required on each bus line. (The internal pull-up resistors in the pad logic may be enabled, so external pull-up resistors are not necessary.) For data transmission, the SDA line transitions to the correct voltage level with the SCL line low. This is followed by the master pulsing the SCL line high while keeping the SDA line constant. The start and end of each transaction is delaminated by a start and stop condition on the bus. These conditions are special conditions where the SDA line is allowed to transition when the SCL line is high. The transaction format is given in Figure 4.15. It begins by transmitting a start condition followed by the 7 bit slave address and the R/$\overline{\text{W}}$ bit. The R/$\overline{\text{W}}$ bit indicates the direction of the subsequent data transfer. If this bit high, the data transfer is from the slave to the master, if it is low it is from master to slave. In the subsequent data payload, the source of the data drives the bus during the 8 data bits and then releases the SDA during the acknowledge bit. The other party must drive the SDA line low during the acknowledge bit to acknowledge the correct reception of the data. However, the last acknowledge bit may be inverted in some situations. For example, when the master is receiving data from a slave, the final acknowledge bit will be inverted. Finally, in the example in Figure 4.15, a stop condition is transmitted; however, a repeated start condition could instead be transmitted thereby starting another transaction, possibly in the opposite direction.

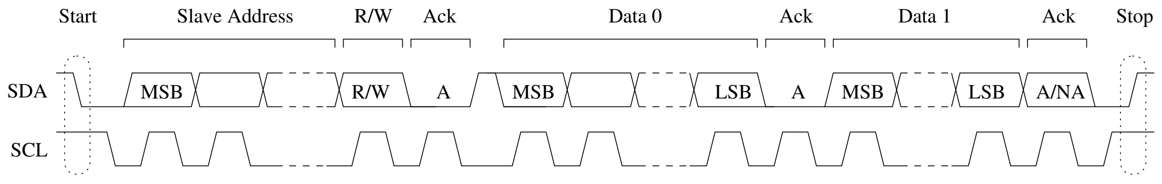


Figure 4.15 – I^{2}C transfer format

4.9.1.1 Clock Stretching/Flow Control

Clock stretching is a method for a slave to slow down the transfer of data. This works by the slave holding the SCL line low even after the master has released it. The master does not proceed with the transfer until the slave has released the SCL line. Therefore, the slave can slow the data transmission on a per-bit basis. Either the master or the slave can also hold the SCL line low for an undetermined amount of time between byte transmissions to indicate to the other device that it is either waiting for more data to transmit (if it is the transmitter), or that the buffer is full and it is waiting for the data to be processed (if it is the receiver).

The integrated I^{2}C master module allows the slave to do bit-wise clock stretching. It also will pull the SCL line low while it is waiting for data to transmit or if its receive buffer is full. See subsection 4.9.2 for more details. The integrated I^{2}C slave module will never attempt to perform bit-wise clock stretching, however it will pull the SCL line low between byte transmissions if the transmit buffer is empty or the receive buffer is full. See subsection 4.9.3 for more details.

4.9.2 I^{2}C Master

In master mode, the master generates the clock waveform, with the slave possibly stretching the clock to either slow down the transfer or to force the master to wait. The master can either transmit or receive data.

4.9.2.1 Bus Arbitration

As specified in the I^{2}C standard, the master must co-exist with other bus masters on the bus. Two features allow the bus to be shared with other masters. First, a circuit continuously monitors the bus for start and stop conditions to detect when the bus is idle. If the processor commands the I^{2}C module to start a transaction while the bus is busy, the module waits until a stop is received and the bus is idle before starting its transmission. In case two bus masters attempt to start transmitting at the same time, a bus arbitration scheme is used to select one master while not losing the integrity of any data transmission. Each bus master senses the state of the SDA line while transmitting, and if the transmitted SDA level does not match the actual one, then that master loses arbitration. So, for example if master one is sending the data “011" while master two is sending “001," the following will take place. First, both masters will drive out a zero onto SDA, and neither master will lose arbitration. Second, master 1 will drive a one, while master 2 will drive a 0. The external SDA line will go low, and master 0 will sense that the signal it is driving and the state of the SDA line is different and will lose arbitration and abort the transfer. Master 1 will go on to complete the transfer at which point master 0 will re-attempt its transfer.

The number of bits required before one of the masters lose arbitration depends on when the unique bit appears in the data stream of each transmission. If the unique bit appears in the first byte (the slave address and R/$\overline{\text{W}}$ bit), the master module will silently abort the transmission and retry when the bus becomes free. However, if the master loses arbitration at some point in the data payload, then the integrity of the re-transmitted packet cannot be guaranteed because the master may have already transmitted any number of bytes in the aborted transmission. Therefore, if this condition exits, the ArbLost bit in the I2CM_SR register is set so the software can attempt to re-transmit the entire data packet. This bit should only be set in a multi-master environment.

4.9.2.2 Transmit Mode

In this mode, the R/$\overline{\text{W}}$ bit is set to zero and the data payload goes from the master to the slave. This mode is selected by clearing the ReadEn bit in the I2CM_TCFG register. This bit is automatically cleared by any write to one of the transmit data registers (I2CM_TDR*). The slave address must be programmed in the I2CM_TCFG register. The transfer can be initiated either by setting the Start bit in the I2CM_TCFG register, or by writing to one of the I2CM_TDR* registers. When the data is written to a TDR register, the TDRE line goes low until the data can be transferred to the seralizer. The TXEmpty line goes high when the TDR and the seralizer are empty and a transmission is ongoing. It does not go high when a transmission is not ongoing. Figure 4.16 shows a sample transaction with three bytes being transferred to the slave. In the figure, after the first two data bytes are sent the transmitter runs out of data and suspends the transfer by pulling the SCL line low until the processor writes the last data byte. The end of a transmission is indicated by a write to the I2CM_TDR_STOP register. However, if instead of a stop condition a re-start is desired, this could be achieved in one of two ways. If the next transaction will also be from the master to the slave, writing the to the I2CM_TDR_START register will initiate a re-start condition and a transmission of the address before the data written to the I2CM_TDR_START register is also written as the first byte of the data payload. Otherwise, if the next transaction must be from the slave to host, writing the I2CM_TXCFG register with the ReadEn field, RxNum field, and TxAdr field set appropriate values as well as the Start bit set will initiate a re-start after the present byte is done transmitting.

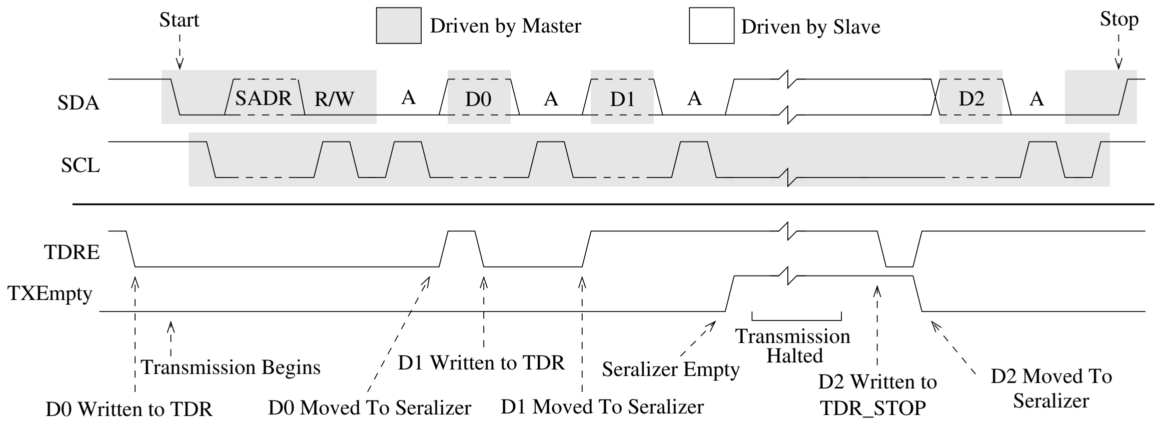


Figure 4.16 – I^{2}C master transmission

4.9.2.3 Receive Mode

In this mode, the R/$\overline{\text{W}}$ bit is set to one, and the data payload goes from the slave to the master. This mode is selected by setting the ReadEn bit in the I2CM_TCFG register. For this type of transmission, the I2CM_TCFG register is programmed with the number of bytes to transfer in RxNum. For the data payload, the master drives the acknowledge signal after each byte transmission, except for the final byte, when the RxNum is one. On the final byte it negative acknowledges after which the slave releases the bus and the master drives out a stop condition to terminate the transfer. Figure 4.17 shows a three byte transaction. In the figure, the processor does not read and process the first two data bytes immediately, so they cause the master to suspend the transfer when the receive buffer is full and wait until the processor reads a byte from the receive buffer. The slave device could also halt the transfer at any time by holding the clock line low.

A bug exists which makes it not possible to issue a re-start after a master reception, instead a stop condition is always issued.

4.9.3 I^{2}C Slave

The chip integrates a I^{2}C slave module which is completely independent of the master. The only circuitry which is shared between the two modules is the I/O lines and the BusActive line. This signal, which is available in both the I2CM_SR and the I2CS_SR, tells the state of

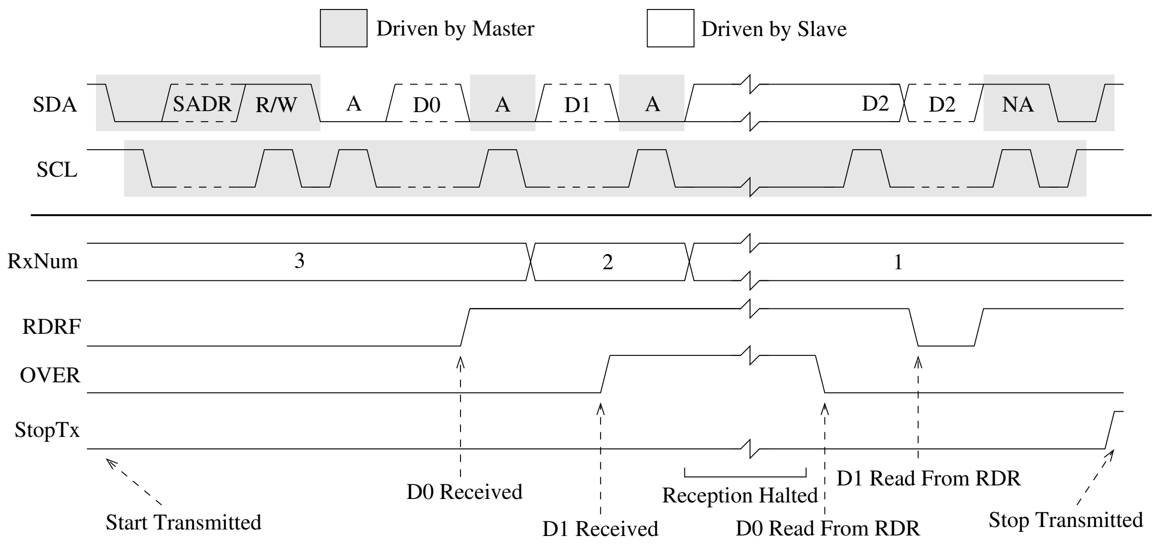


Figure 4.17 – I^{2}C master reception

the bus, regardless of the state of either module. The address which the slave will listen to is set in the control register. As the master and slave are independent, if the master initiates a transaction with the slave address the same as the address in the slave control register, the transaction will take place between the integrated master and slave module. This can be used for testing both modules.

The slave cannot initiate transactions, and does not supply the clock. It utilizes the clock supplied from the master. It is never appropriate for the slave module to negative acknowledge any data, so the slave will always acknowledge all data bytes it receives.

4.9.3.1 Transmit Mode

In this mode, the master initiates a transaction with the R/$\overline{\text{W}}$ bit set to one and the data payload is from the slave to the master. After the transmitted slave address is checked and it matches, the module begins transmitting data if data has been previously written to the transmit data register (I2CS_TDR). If the TDR is empty, it sets the TXEmpty flag, possibly causing an interrupt, and waits for data to be written to TDR. Whenever the TXEmpty bit is set, the SCL line is held low to suspend the transmission. When a NAck is received from the master which indicates the end of a transmission, the slave terminates the transmission. At this point, the master must issue a stop/start combination or a re-start (which due to a hardware bug, the integrated master cannot issue a re-start at this point) for transmission to continue. Figure 4.18 shows a three byte transmission by the slave to the master. In the figure, after the reception of the address, the slave suspends the transmission until the processor writes data to the TDR register. This could be avoided by having data already present in the TDR register.

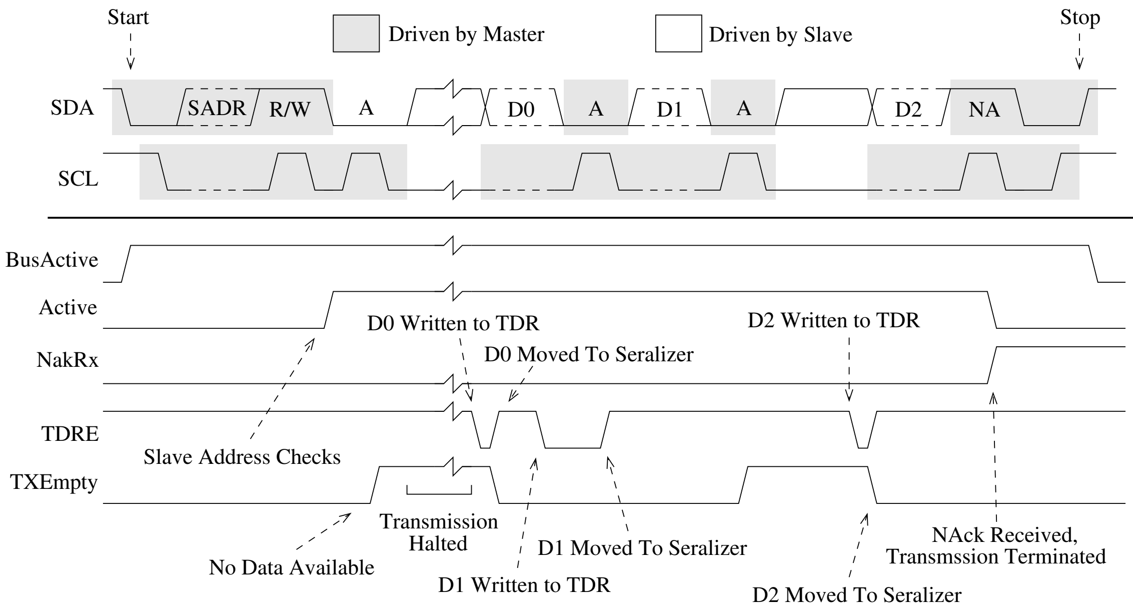


Figure 4.18 – I^{2}C slave transmission

4.9.3.2 Receive Mode

The slave can receive data from the master when the master initiates a transaction with the R/$\overline{\text{W}}$ bit cleared. Figure 4.19 shows the slave receiving three bytes from the master. After the transmitted address matches the slave address, the module receives the first byte from the master and sets the RDRF flag to signal the software. Reading the receive data register (I2CS_RDR) will clear this flag. If the flag has not been cleared and another byte is received, the OVER flag is set, and the module holds the SCL line low to stop the master from sending more data. Reading the RDR will clear the OVER flag, and reading it again will clear the RDRF flag. The slave never loses data due to buffer overrun, instead it stops the transmitter from continuing.

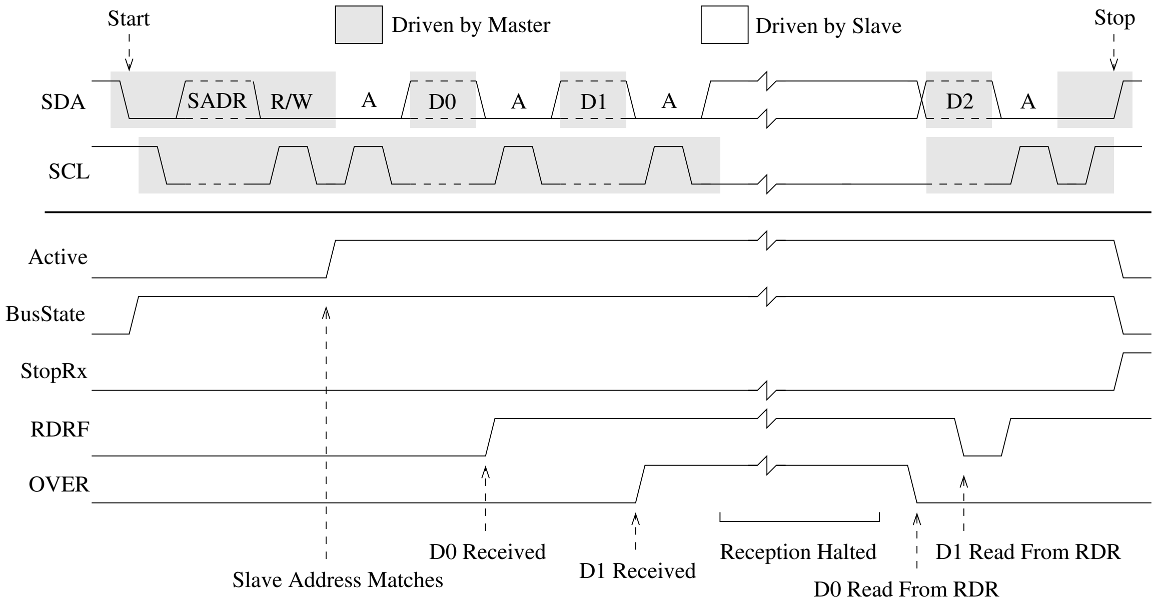


Figure 4.19 – I^{2}C slave reception

4.10 Conclusion

The digital section of the design is where most of the area of the chip is used. It is very complicated with a full microcontroller and its peripherals integrated. This chapter did not discuss all the digital peripherals as those relating to other design segments will be discussed in their appropriate chapters. Instead this chapter covered the digital peripherals and microcontroller not covered in other sections.

The microcontroller and peripherals is an important part of the design because it provides the programmability and computational power for the sensor node. The microcontroller is a low-power MSP430X clone with some extra extensions which more fully support single-instruction multiple-data instructions.

Chapter 5

UWB Transceiver Design

This chapter presents the design of a non-coherent UWB transceiver. The UWB transceiver is responsible for communication with neighboring network motes. Figure 5.1 shows how this fits in with the complete sensor network node.

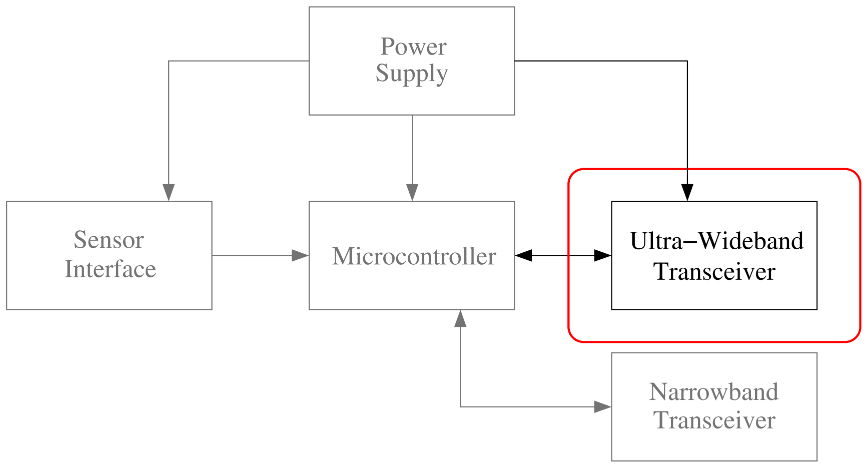


Figure 5.1 – UWB transceiver enclosed in red box.

The UWB receiver has many novel features such as heavy reliance on analog domain processing for the baseband to decrease power consumption and the presence of a variable-rate convolutional coder and interleaver to make it resistant to poor channel conditions. Furthermore, the front end of the receiver uses a novel architecture whereby a T-coil load is made on-chip through the use of a symmetric inductor. The transceiver has exceptional low power consumption and high data rate compared to other radios making it an attractive option for some applications. The all-digital UWB transmitter is simple yet effective as well as low power and re-configurable.

The rest of this chapter is organized as follows: Section 5.1 presents the architecture of the UWB transmitter and receiver including modulation formats and the UWB receiver architecture. Section 5.2 describes the convolutional code which was used on the link as well as the process used to search for the code, simulations of the code's performance, and hardware implementation of the encoder and decoder. Section 5.3 presents the random interleaver used and its implementation. Section 5.4 presents the 8-bit CRC used to verify the data integrity. Section 5.5 discusses the architecture and design of the UWB transmitter while sections 5.6-5.9 present the design of the UWB receiver. Section 5.11 concludes the chapter.

5.1 Architecture

The architecture of the complete UWB communication system is shown in Figure 5.2. The input message is first encoded with a variable-rate convolutional encoder. The rate varies from 1/2 to 1/10 and is set *a-priori* by the transmitter. This is followed by a 64-symbol random interleaver which combats brief burst errors which can be introduced by the de-modulator. Then two preambles are added and the symbols are mapped to waveforms in the modulator and then transmitted. The transmitter uses binary pulse-position modulation (PPM) whereby it transmits a UWB pulse in one of two locations. At the receiver side, the de-modulator uses the preambles to find the packet then determines the most likely symbols which were sent all while staying synchronized with the incoming data stream. The decoded symbols go to the inverse interleaver and then the Viterbi decoder decodes the original message.

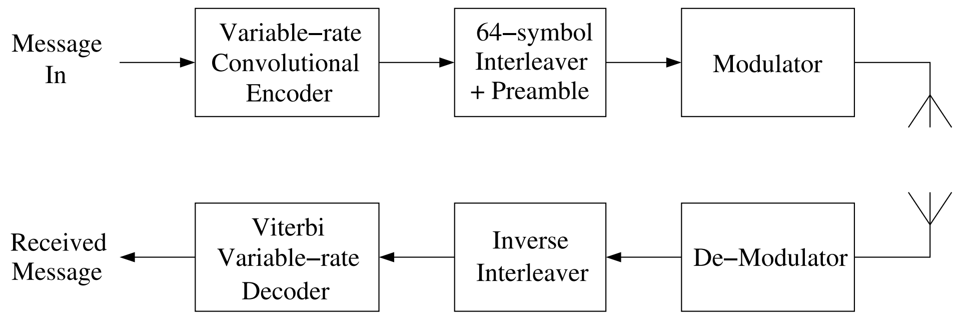

Message In
Variable−rate Convolutional Encoder
64−symbol Interleaver + Preamble
Modulator
Received Message
Viterbi Variable−rate Decoder
Inverse Interleaver
De−Modulator

Figure 5.2 – Overall UWB communication system architecture.

The variable-rate convolutional code allows one to tradeoff maximum data rate with received bit-error rate. This allows for successful communication with varying channel environments and SNR values.

For the transmitter side, the variable-rate convolutional encoder and interleaver are synthesized in the digital section of the design and are described in subsection 5.5.1. The modulator is described in subsection 5.5.2.

On the receiver side, the Viterbi decoder and inverse interleaver are synthesized in the digital section. The convolutional code is described in section 5.2 and the interleaver in section 5.3. The de-modulator is the most complicated part. A block diagram of this is shown in Figure 5.3.

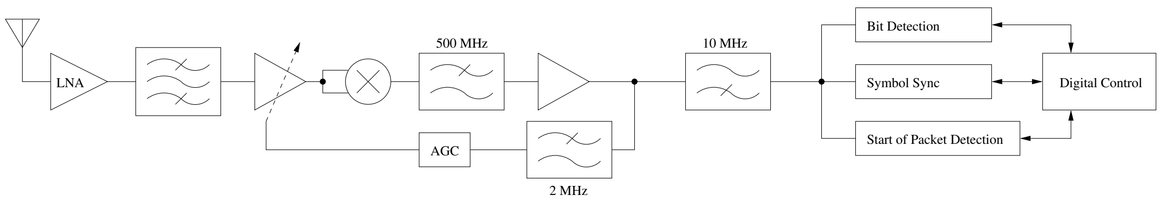


Figure 5.3 – Architecture of the de-modulator

This is an energy-detection type UWB receiver. It works by finding the power of the incoming signal, integrating this over each of the two possible pulse locations and selecting the one with the larger energy content as the most likely received symbol. The front end contains an LNA and a filter followed by a large amount of adjustable gain. Then a squarer circuit finds the power in the signal. The low-frequency component of the power is used to adjust the AGC loop to stop the receiver from saturating due to narrowband interference, and the high-frequency portion is used to determine the received bits. Three different circuits are used; one to find the presence of a packet, one to keep the receiver synchronized to the incoming UWB pulses, and a third to receive the actual data. The various parts of this receiver are covered in sections 5.6 to 5.9

5.1.1 Modulation Format

The modulation format used is binary pulse-position modulation (PPM). The two possible pulse positions are either at the beginning of the symbol or halfway through the symbol. For a maximum symbol rate of 10 MHz, this means there is a separation between the two position of 50 ns. The precise definition of which position is considered a ‘1’ and which is considered a ‘0’ is run-time selectable. Obviously, the transmitter and receiver must be in agreement on this for successful communication to take place.

5.1.2 Packet Format

The packet structure is shown in Figure 5.4. The packet has two preambles. The first one is used by the packet detection circuitry to confirm a packet is present and attain a rough estimate of the UWB symbol position. The first preamble consists of a repeating 7 bit PN code generated at the transmitter by a maximal-length 3-bit linear feedback shift register (LFSR). The second preamble is a single non-repeating PN code which allows the receiver to get an absolute position of the start of the packet—this one generated by an eight bit LFSR.

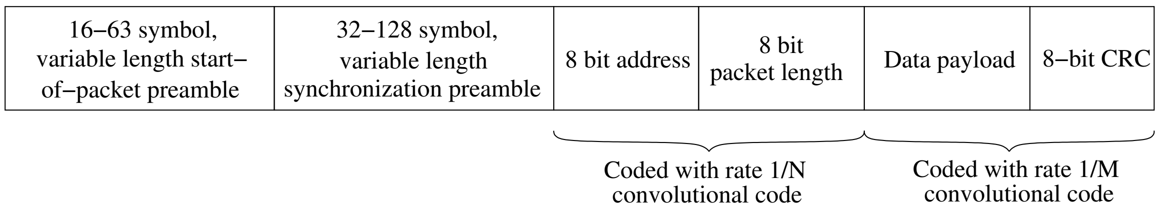

16−63 symbol, variable length start−of−packet preamble
32−128 symbol, variable length synchronization preamble
8 bit address
8 bit packet length
Data payload
8−bit CRC
Coded with rate 1/N convolutional code
Coded with rate 1/M convolutional code

Figure 5.4 – UWB packet structure, N and M are run-time adjustable integers from 2-10.

The transmitter may send more repetitions of the first preamble than the receiver is expecting. This is recommended as it will increase the probability the packet detection circuitry successfully detects the packet. Because the first preamble consists of many repetitions of a short code, there is no way for the packet detection circuitry to know absolutely where the beginning of the packet is once it recognizes the start of packet preamble. This is the job of the second preamble. After the packet detection circuitry detects a packet, the digital control disables the packet detection circuitry and enables the symbol synchronization and bit detection circuitry. The symbol synchronization circuit takes the rough estimate of the UWB symbol position generated by the packet detection circuit, refines it, and tracks the incoming UWB symbol stream to keep the bit detection circuit synchronized with the incoming data. After the packet detection circuit has detected a packet, the bit detection circuit correlates its output with the expected PN code found in the second preamble to determine the start of the packet. The second preamble is also responsible for detecting false alarms caused by the packet detect logic. A long PN code in the second preamble makes the overall false packet detection rate very low.

A simple scheme is employed to enable the communication link to interchangeably utilize two different convolutional code rates without prior negotiation with receiver. The transmitter may transmit the second preamble either true or inverted, thus signaling the receiver to use one of two pre-determined rates. This allows a transmitter with a better channel SNR to use a higher data rate to send its data than a transmitter with a poor channel SNR, and to do this without prior negotiation with the receiver. That is, the UWB receiver can be programmed to receive one of two code rates, and choose which one to use by the polarity of the received preamble.

After the two preambles, there is the address and the packet length bytes. The address byte is checked against the address of the receiver and is not received unless they match. Zero is the broadcast address which is received by all receivers. The packet length is the length of the data payload and the length byte, but not the CRC or address byte. This then sets a maximum data payload size of 254 bytes.

The address and length bytes are arguably more important than the rest of the payload, because an error in those bytes can cause problems in the receiver. For example, an error in the length byte could direct the receiver to receive a very long packet which is not present, thereby stopping the receiver from hearing a retry of the same packet. For this reason, it is possible to put a stronger, lower-rate convolutional code on these two bytes than on the rest of the packet. This gives extra assurance that these bytes will be received correctly. The encoder and Viterbi decoder are designed to automatically switch the code strength at this location without the need to terminate the first code to start the second code.

After the data payload, an 8-bit CRC is present. This is automatically generated and checked in hardware. The data payload and CRC are encoded with a convolutional code. Not shown in Figure 5.4, the convolutional code must be terminated with 5 zeros after the CRC is transmitted.

5.2 Convolutional Code

The UWB channel is inherently noisy, both due to the low SNR of the received signal, and also due to the prevalence of UWB-like signals in the environment. Some signals that will interfere with UWB signals include switching power supplies, OFDM modulated narrowband signals (due to their high peak to average ratio), brush motors, and household appliances such as dimmer switches. In fact, the integrated buck power supplies which power this chip can cause grievous interference to the UWB receiver during their switching events, so this is a form of self-generated interference. The noise will introduce bit errors, which are best fixed by adding an error control code to the link.

The convolutional code adds a certain amount of redundancy to the transmitted data so the symbol errors can be detected and corrected at the receiver. The amount of redundancy added affects the rate at which the code can correct for errors. However, it also adds more symbols to transmit for the same number of useful bits, therefore decreasing the data rate.

For the remainder of this chapter, the message bits will be referred to as bits, while a single PPM-coded UWB symbol will be referred to as a symbol. So one bit maps to N symbols before they are transmitted over the UWB channel where 1/N is the rate.

5.2.1 Fundamentals

Unlike a block code which operates over data of fixed length, the convolutional code is a continuous code. It processes the input and produces the output in a stream. The structure of the generator matrix is such that the the encoding operation can be viewed as a filter, or the convolution of the data with an impulse response—hence the name. In general any code is described as having a rate $R = k/n$ where, for a convolutional code, the encoder accepts k input bits and produces n output bits per cycle. The arithmetic can be done over any field, however $GF(2)$ is the most common and is used for the presented code.

Each convolutional code is described with a set of generator polynomials which can be arranged in a matrix to form a transfer function matrix. The generator polynomials can be a ratio of two polynomials. If the generator polynomials can be described as a simple polynomial, not a ratio, then the transfer function matrix is said to be feedforward. In this case the filter is an FIR filter and the encoder does not include feedback. A rational generator polynomial is said to be a feedback or IIR encoder. The same performance can be achieved with either FIR or IIR type codes for the same complexity, therefore FIR codes are used as they are simpler to analyze.

If one of the generator polynomials is strictly the input, then the code is called a systematic code. Systematic codes are simpler for people to decipher, however non-systematic convolutional codes always have superior performance to systemic convolutional codes and therefore systematic codes are rarely used. Furthermore, it is generally machines who decipher the code, not humans, so having a systematic code is of little benefit.

A code of rate k/n accepts k inputs and produces n outputs each cycle. For low rate codes, k is traditionally set to 1 and n is set to an integer to generate codes with rate $1/n$.

Setting k to a larger integer allows for high rate codes. However as we are dealing with low rate codes, the following discussion will be constrained to the case where k is 1.

The maximum degree of the generator polynomials is an important parameter called the constraint length υ. The constraint length is the total memory needed in the encoder in an optimal implementation. Furthermore, it sets the complexity of the decoder. The number of states of the encoder and decoder is given by 2^{υ}. Therefore the decoder complexity grows exponentially with constraint length which places a severe practical limit on the constraint length.

A code is said to be catastrophic if there exists a possibility of decoding infinite output errors with finite number of input errors. Mathematically, a code is not catastrophic if and only if it has a right inverse $\mathbf{G}(x)^{-1}$ with only polynomial entries where $\mathbf{G}(x)$ is the generator matrix. Because of the possibility of infinite output errors, catastrophic codes must not be used.

5.2.1.1 Performance Measurement

Conventionally, the performance of codes is measured by assuming the codes are used on an AWGN channel and then graphing the data bit error rate versus the SNR. When comparing codes of different rates, the energy per data bit is kept constant, so the energy per symbol is decreased at higher rate codes. For this application, the BER performance of the code is instead plotted versus the UWB symbol error probability. In this method the energy per symbol is kept constant and therefore the energy per bit is increased with decreasing rate. This method is used because it is directly applicable to the conditions of the UWB transmitter. The UWB transmitter has a fixed energy per symbol value set by the hardware, and we can easily determine the UWB symbol error rate as it is automatically calculated by the receiver. Therefore, for all code comparisons, the code performance is measured versus the probability of symbol error, not SNR.

5.2.2 Code Selection

The convolutional code selected for this application is not a single code, but instead it is a family of rate-compatible convolutional codes (RCCC). Or, said another way, the code consists of one code of rate 1/10 which may be heavily punctured to a rate 1/2 code. The motivation for using these codes is to allow the dynamic selection of the code based on the transmitter's knowledge of the channel. Rate-compatible codes are codes such that the polynomials for the rate $R = 1/(n-1)$ are a subset of the polynomials for the code of rate $R = 1/n$. This property greatly reduces the decoding complexity.

The constraint length υ of the code affects code performance and greatly affects decoding complexity. For this application the constraint length was set to 5 as a compromise between complexity and performance. This means the decoder has 32 states.

With the constraint length chosen, the rate of the codes could be determined. The rest of the UWB receiver is designed to work with symbol error rates as high as 0.2, so the lowest rate code should produce acceptable performance at that symbol error rate. A rate 1/10 code will produce a BER better than 10^{-4} so this was chosen as the lowest rate code. The highest rate code was chosen to be $R = 1/2$.

In the literature, a number of RCCC codes are presented for high rate codes (greater than 1/2) for use in standard narrowband communication systems [66–69], and for very low rate codes (less than 1/30) for use as a substitute for orthogonal codes in CDMA communications [70]. However, no RCCC codes were found in the rate 1/2 to 1/10 range. Therefore a computer search algorithm was developed to produce a set of RCCC codes.

The search proceeded as follows, first the best known rate 1/4 code was selected with the appropriate constraint length from [71,72]. The search proceeded to find the best codes with a higher rate, followed by another algorithm to find the codes of lower rate. The best known rate 1/4 code was chosen as the starting point due to the similarities between the best known 1/4 code and the best known 1/3 code, i.e. they share two polynomials. Also, the attempt at using the rate 1/3 code as the starting point resulted in very poor rate 1/2 code performance.

To find the codes of higher rate, i.e. to find the code of rate $R = 1/(n-1)$ given the code of rate $R = 1/n$, n simulations were performed, dropping a different polynomial for each simulation. The performance of each of the resulting codes was compared and the best code was selected as the rate $R = 1/(n-1)$ code. The resulting code was also checked to determine that it was not catastrophic.

To find the codes of a lower rate, i.e. to find the code of rate $R = 1/(n+1)$ given the code of rate $R = 1/n$, an extra polynomial must be added. The concept is to simulate all the possible codes and choose the best performance. For a code with a constraint length υ a total of $2^{\upsilon} - 1$ polynomials are possible, however, it is known that the best codes always include the x^{υ} term, so this eliminates half of the codes. Therefore, the simulator must compare the performance of $2^{\upsilon-1}$ codes and choose the best one. This code must also be checked to determine that it is not catastrophic.

To do the simulation, a program was written in C to simulate the UWB channel and the convolutional code performance. The channel was modeled as a binary symmetric channel. The random interleaver allows the channel to be modeled in such a way. A Matlab program was used at first, however, the simulation speed was unacceptable, so the program was re-written in C. Two versions were written, one with non-windowed decoding and the other with a 32-bit window. The non-windowed decoding was used to find the codes, and the windowed version was used to emulate the actual implementation.

5.2.3 Chosen Code

Table 5.1 gives the code polynomials for the variable rate code. As can be seen, each successive code uses all the previous polynomials and includes one more. This simplifies the decoding hardware of such codes with only a slight performance penalty. The rate 1/4 code is the best known code for that rate and constraint length [72], and the rate 1/3 and rate 1/2 codes have performance which is very close to the best known codes. For the lower rate codes, no information could be found on the best known codes. This is because very few, if any, applications use codes with that rate range, so little information is published on them.

Table 5.1 – Convolutional Code Polynomials

Rate	Polynomials (Octal)	d_{free}
1/2	53, 75	8
1/3	53, 75, 71	13
1/4	53, 75, 71, 67	18
1/5	53, 75, 71, 67, 43	21
1/6	53, 75, 71, 67, 43, 65	25
1/7	53, 75, 71, 67, 43, 65, 47	29
1/8	53, 75, 71, 67, 43, 65, 47, 55	33
1/9	53, 75, 71, 67, 43, 65, 47, 55, 77	39
1/10	53, 75, 71, 67, 43, 65, 47, 55, 77, 57	44

The generator polynomials are in octal and are interpreted as follows: Using the rate 1/2 code as an example, the two generator polynomials are 53 and 75. In binary notation, these are 101011 and 111101. Each bit in the binary notation maps to the coefficient of the delayed input, so the equations to generate the two output codes are given by:

$$\begin{aligned} c_t^{53} &= m_t + m_{t-2} + m_{t-4} + m_{t-5} \\ c_t^{75} &= m_t + m_{t-1} + m_{t-2} + m_{t-3} + m_{t-5} \end{aligned} \tag{5.1}$$

As is expected, all generator polynomials include the terms m_t and m_{t-5}.

Because of the hardware implementation of the Viterbi decoder, a windowed decoding scheme is necessary. A windowed Viterbi decoder forces a decision after N bits even if more than one path is still surviving where N is the window size. If the window is large enough, the probability of having more than one surviving path after N bits is very small so it does not affect performance much. However if the window is too short it will negatively affect the bit error rate. A 32 bit decoding window was chosen to minimize the performance penalty of window decoding. Figure 5.5 shows the performance of the rate 1/2, 1/3, and 1/4 codes with a 24 bit window, a 32 bit window, and the optimal decoding without a window. As can be seen, the 32 bit window is much closer to the optimal case than the 24 bit window. Therefore, a 32 bit window was chosen in the implementation.

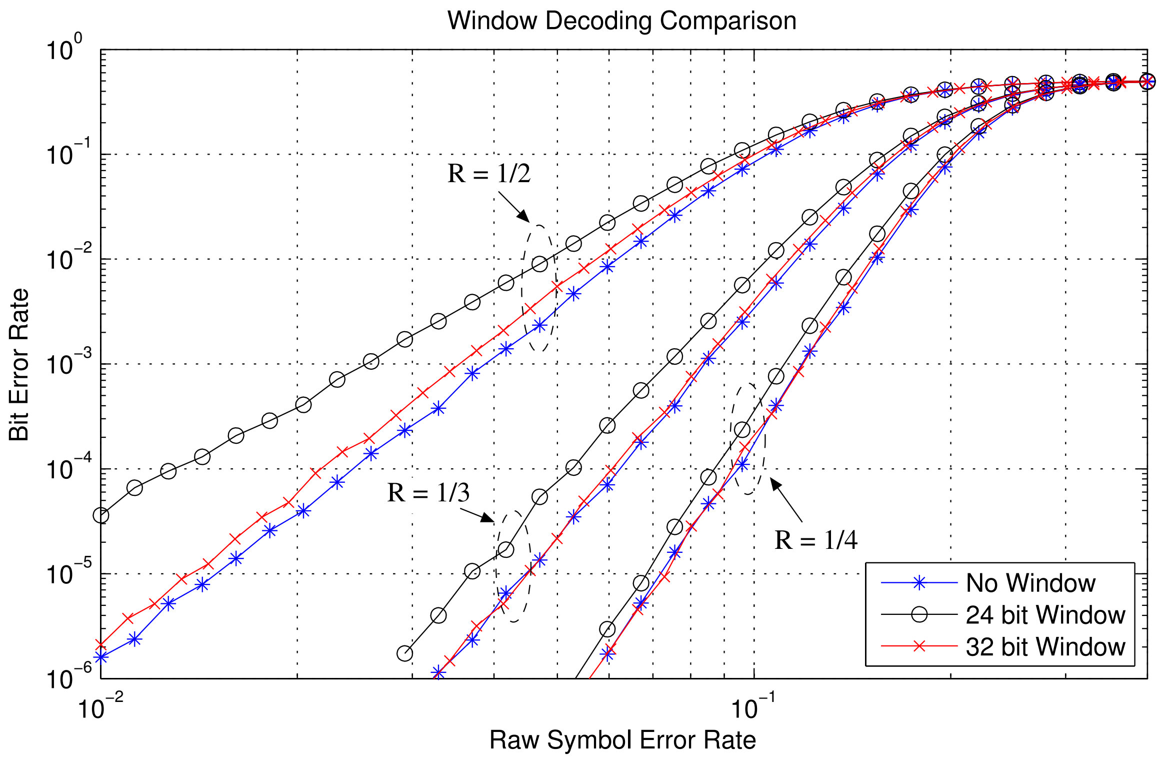


Figure 5.5 – Effect of window length on performance.

The window decoding algorithm used in the simulations is not optimal, even for the given window size. This was done to match the hardware implementation. In particular, the optimal window decoding algorithm selects the path to the state which has the least cumulative cost to use as the decoded bit. However, the implementation always selects the path leading to the zero state, regardless of whether it actually has the minimum cumulative cost. This avoids the requirement to find the global minimum of all the states which is a slow and difficult operation. However, if the decoding window is long enough, this non-optimum path selection has little effect on performance.

Figure 5.6 shows a comparison of the rate 1/2 and rate 1/3 codes with the best known codes from [71,72]. The d_{free} distances for the chosen codes are the same as the best known codes, however the distance spectra for the rate 1/3 code is somewhat worse than the best known code. However, the rate 1/2 code has virtually identical performance to the best known code. The chosen rate 1/4 code is the best known code. For codes of rate 1/5 to 1/10, no information could be found about good codes as these are generally never used in conventional narrowband communication systems.

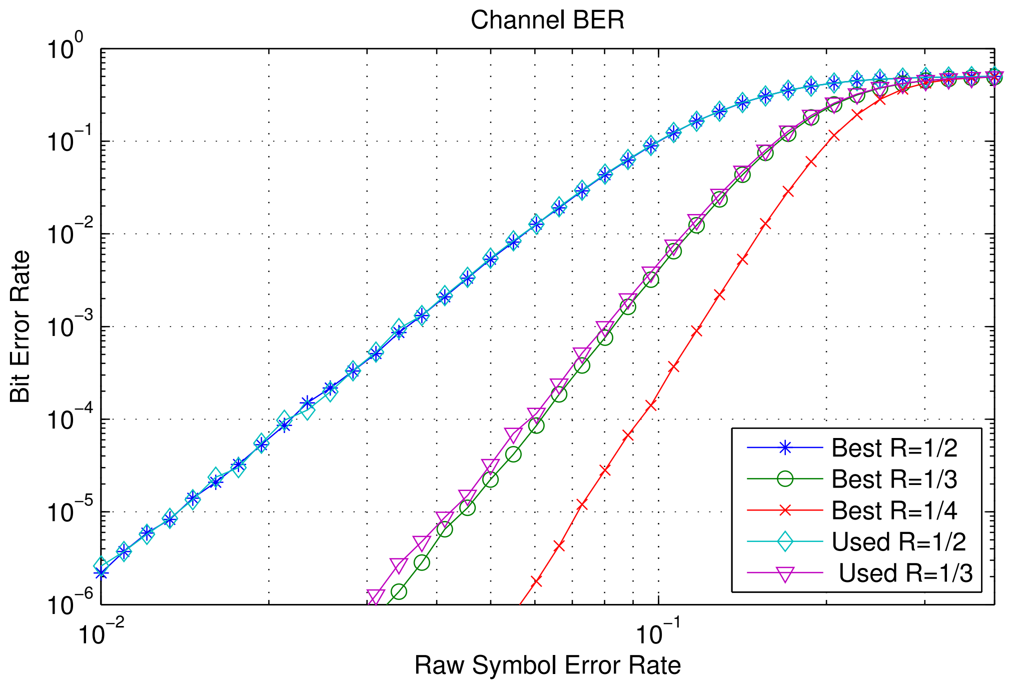


Figure 5.6 – Code comparison with best known codes, window=32 bits.

The simulated performance of the entire family of codes is shown in Figure 5.7. These are all decoded with a 32 bit window using the same algorithm implemented in the Viterbi decoding hardware.

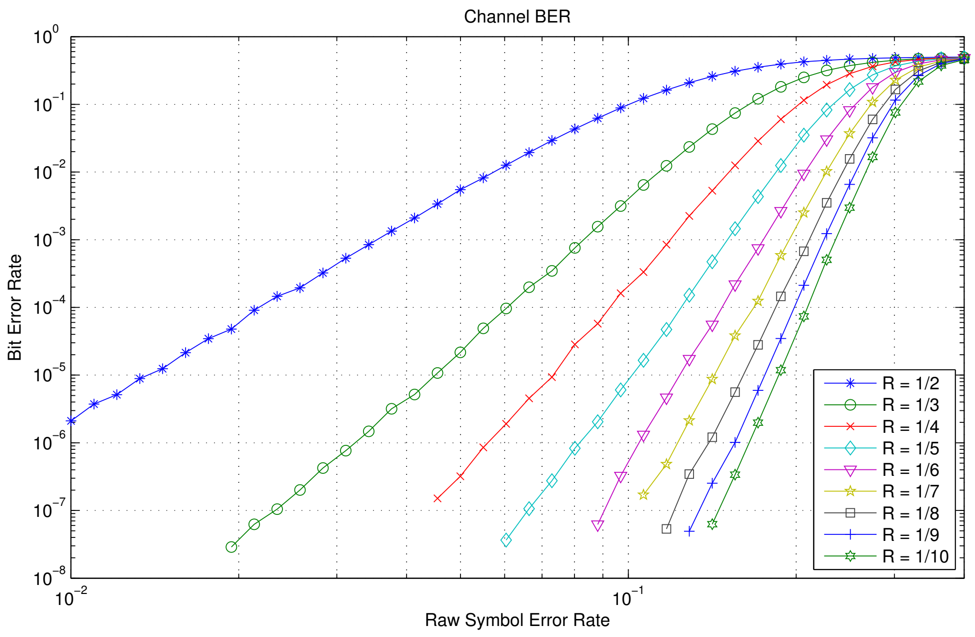


Figure 5.7 – Code performance, window=32 bits

Figure 5.8 shows the average contiguous error-free block length of data bits. This graph is useful for the transmitter in choosing the appropriate code given a packet size and symbol error rate. This graph takes into account that the errors in the decoded message generally appear in groups. In a network, if any group of errors is encountered in the packet, the entire packet will need to be re-sent unless there is an outer code present. Therefore, if the transmitted packet size is much less than the average contiguous error-free block length, then the packet has a good probability of correct reception.

5.2.4 Implementation

Both the encoder and decoder are implemented in hardware. Both are described in VHDL and synthesized. To test the encoder and decoder, the two modules were separated from the rest of the chip and synthesized separately. Two VHDL testbenches were developed to test the encoder and decoder. The testbenches read data files produced by the simulation program which includes the original message, the encoded message, the encoded message

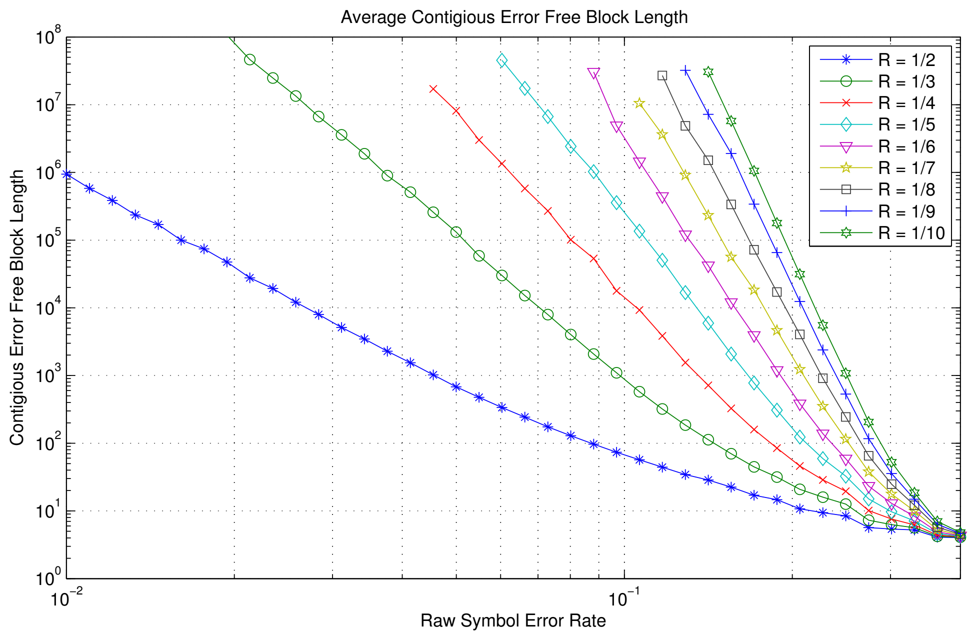


Figure 5.8 – Error free data bit block length, window=32 bits

with noise, and the decoded message. With these datasets, the testbenches verified the correct operation of the encoder and decoder by checking that the simulations agreed with the hardware under various channel error rate conditions. After verification, the encoder and decoder blocks were re-combined with the rest of the VHDL for the chip and synthesized again along with the rest of the chip.

For a code of constraint length υ, the code must be terminated with υ zeros so the last few data bits of the packet have the same error protection as the others. For this code the five padding bits are added automatically at the transmitter at the end of the bit stream. The receiver can then assume that the Viterbi decoder ends in state 0.

5.2.4.1 Encoder

The encoder is a simple circuit and is shown in Figure 5.9. Each of the polynomials in Table 5.1 have an XORing output stage. The output data selector may only select some of the outputs depending upon the selected rate.

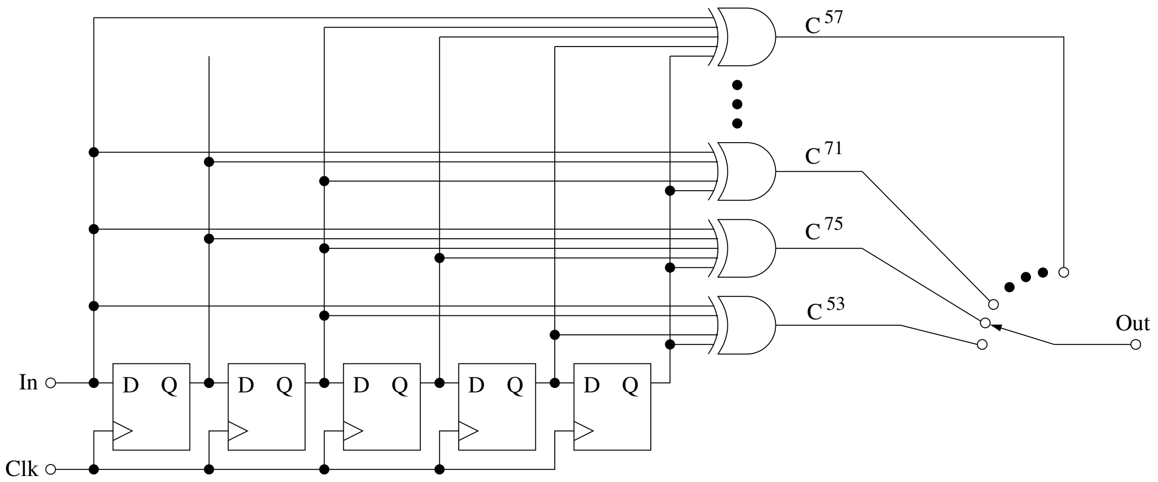


Figure 5.9 – Convolutional encoder schematic.

5.2.4.2 Decoder

The Viterbi decoder consists of 32 add-compare-select units—one for each of the 2^{υ} states. This highly parallel implementation can handle data at much higher rates than required for this application (up to Gbps). Each state requires 39 flip-flops—7 for the path cost and 32 for the window. These, combined with the combinational logic, make the entire Viterbi decoder require much silicon area. The schematic of a single add-compare-select unit is shown in Figure 5.10.

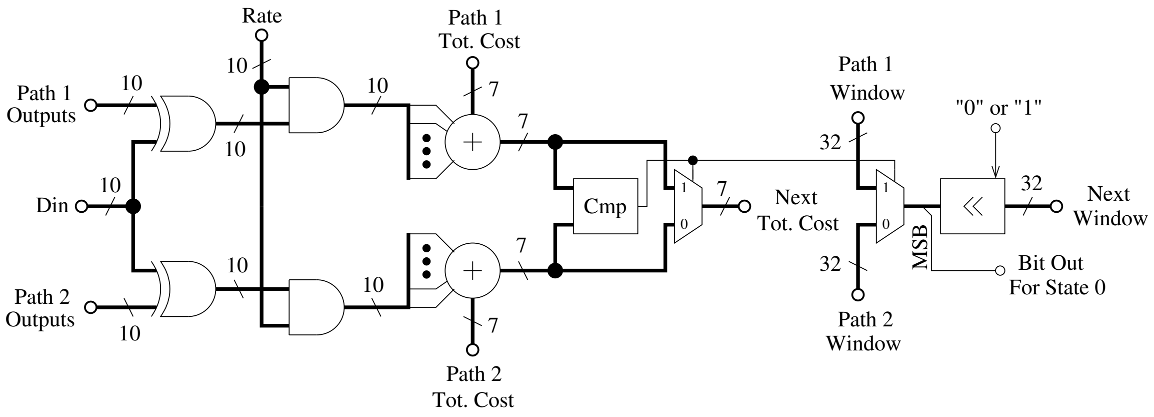


Figure 5.10 – Add-compare-select unit for the Viterbi decoder.

An input word is applied in parallel, and the first XOR gates and following AND gates compute the Hamming distance between the received data and the expected outputs of each of the two possible paths into this state. In actual hardware, the “Paths Outputs” are a constant (though different for each state), so the first XOR simplifies to either a wire or an inverter. The “Rate" input selects the valid bits. The “Rate” input is thermometer encoded meaning that rate 1/2 is encoded as “0000000011”, rate 1/3 as “0000000111”, rate 1/4 as “0000001111” and so forth. The following adder circuit adds the Hamming distance of each bit together with the previous path cost from the state which is before this in the trellis. The comparator selects which is the minimum path cost, and the two multiplexers select the correct path cost and window to latch into the flip-flops. The window is also left shifted and either a zero or one is added to the LSB, depending on which state this is. For state zero, the MSB of the window is the decoded output bit.

Not shown in Figure 5.10 another circuit is responsible for assuring that the path cost variables do not overflow. This circuit works by sensing when the MSB of all the path costs are one and resetting all the MSBs back to zero. This has the effect of subtracting 64 from the path cost of each state, but does not affect decoding. When this occurs, a counter is incremented so that at the end of decoding, the total number of errors which occurred in the input stream can be determined and reported to the microcontroller. This then assists the microcontroller in determining the proper code for the next transmission.

The width of the path cost adder is determined via simulation to be large enough so it will never overflow.

There is a 32-bit delay through the Viterbi decoder, which the surrounding digital logic must accept. When data is first applied to the input, valid data is produced 32 cycles later. At the end of receiving the bit stream plus the five padding bits, the window of state zero has 32-5 or 27 remaining bits of valid data present which must be clocked out.

5.3 Interleaver

UWB channels generally do not suffer from the multi-path fading present in narrowband channels, and, by itself, the channel is relatively time-invariant compared to narrowband channels. This means that there is minimal need for an interleaver. However, due to the architecture of the de-modulator. The sudden presence of a strong narrowband interferer will cause a short burst of errors with a length of up to 10 symbols. The interleaver will distribute those errors so the convolutional code can correct them. The alternative to a random interleaver is a simple block interleaver. This has advantages in some circumstances, for example it spreads out a block of errors in the most correctable fashion. However, the disadvantage is that a switching power supply at a particular frequency could make communication very difficult.

A switching power supply of a certain frequency will introduce errors which are very systematic, but not grouped. For an 8x8 block interleaver, a switching supply which interferes with every eighth transmitting symbol would be the worst case, because the inverse interleave function would map the eight symbols with high probability of error to adjacent symbols. For this reason a random interleaver was chosen.

The forward and reverse interleaver functions are shown in Table 5.2. The interleaver

Table 5.2 – Forward and reverse interleaver functions

(a) Foward (Transmitter) interleaver

0, 2	3, 2	7, 4	6, 5	2, 2	5, 3	1, 6	0, 3
0, 0	6, 6	0, 6	5, 6	7, 7	6, 1	5, 5	2, 7
5, 7	2, 0	1, 2	3, 5	0, 1	4, 4	6, 2	5, 4
7, 1	4, 2	5, 1	4, 1	3, 1	2, 6	1, 5	2, 1
1, 7	1, 3	5, 0	6, 4	3, 4	7, 2	0, 4	4, 3
5, 2	4, 6	2, 3	1, 4	1, 1	7, 6	7, 0	2, 4
3, 3	0, 5	3, 6	7, 3	6, 0	4, 0	7, 5	1, 0
4, 5	2, 5	0, 7	3, 7	4, 7	6, 7	3, 0	6, 3

(b) Reverse (Receiver) interleaver

1, 0	2, 4	0, 0	0, 7	4, 6	6, 1	1, 2	7, 2
6, 7	5, 4	2, 2	4, 1	5, 3	3, 6	0, 6	4, 0
2, 1	3, 7	0, 4	5, 2	5, 7	7, 1	3, 5	1, 7
7, 6	3, 4	0, 1	6, 0	4, 4	2, 3	6, 2	7, 3
6, 5	3, 3	3, 1	4, 7	2, 5	7, 0	5, 1	7, 4
4, 2	3, 2	5, 0	0, 5	2, 7	1, 6	1, 3	2, 0
6, 4	1, 5	2, 6	7, 7	4, 3	0, 3	1, 1	7, 5
5, 6	3, 0	4, 5	6, 3	0, 2	6, 6	5, 5	1, 4

works by reading in one square of data sequentially, starting at the top line, and working to the bottom line. Then it does the forward or inverse transform as indicated in Table 5.2 before reading out the data sequentially again. The transform in Table 5.2 is described so the two values in each cell are the row,column numbers from which the data in that cell is copied.

5.4 CRC

The UWB transmitter and receiver as well as the narrowband transmitter and receiver all integrate an 8-bit hardware CRC which is used to generate or check the CRC for the associated data packet. The transmitter modules generate the CRC and append it to the end of the packet when transmitting, while the receiver modules compute the CRC over the entire packet, including the CRC byte, and then check that the data in the CRC generator is all zeros which indicates the CRC checked. Before transmission or reception, the CRC is cleared. The schematic of the CRC module is shown in Figure 5.11. The CRC is coded with the same convolutional code as the data packet.

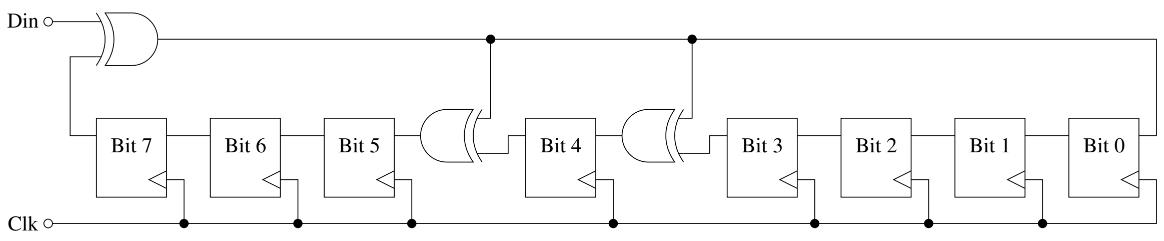


Figure 5.11 – 8-Bit CRC schematic

5.5 Transmitter

The UWB transmitter module contains the seralizer, a hardware CRC generator, a variable-rate convolutional code encoder, interleaver, the LFSRs for the two packet preambles, and the UWB modulator. The complete system is shown in Figure 5.12.

5.5.1 Digital Architecture

The processor first writes a packet into the packet buffer. The packet buffer is a dual-port register file which has one read port and a second write port. The two ports are independent. In this case, the write port is connected to the processor and the read port is connected to the transmitter. This means the processor cannot read the data it has written to the write buffer, only the transmitter can. This buffer has space for two packets, each up to 256 bytes

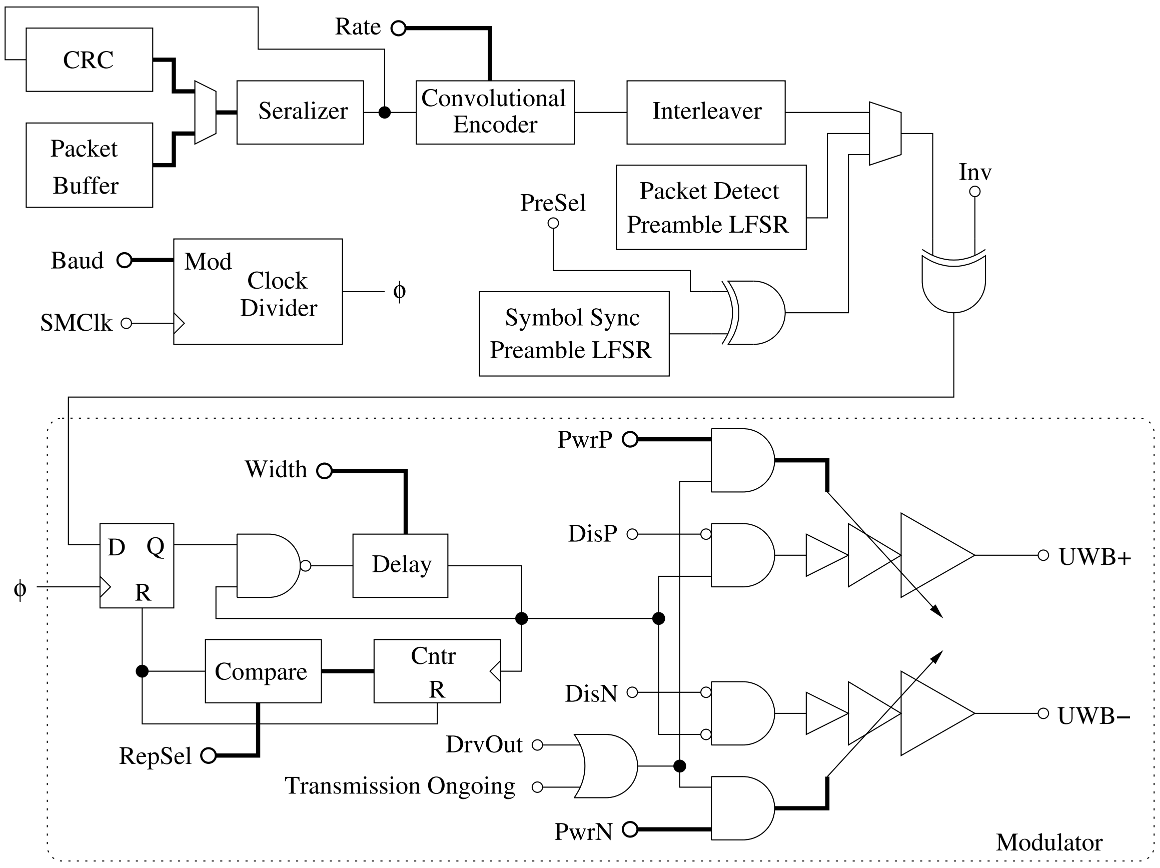


Figure 5.12 – UWB transmitter architecture

in length (254 bytes of data payload). The two packet buffers are alternately transmitted, so the processor can be filling one while the other is transmitting. The first byte written to the packet buffer is the length of the packet, including the length byte, but excluding the CRC or the address byte. The address byte sent with the packet is not written in the buffer, instead it is written to the TxAddress field in the UWBTX_CFG register.

After writing the packet to the correct buffer, the microcontroller sets the Start bit in the UWBTX_CR register to begin the transmission. As shown in Figure 5.12, the two LFSRs generate the two preambles which are needed by the receiver. The length of both of the preambles is variable length and must be set to be compatible with what the receiver is expecting. The first packet-detect preamble consists of a single seven-bit repeating sequence. The sequence is “1001011” which is generated by a 3-bit maximal-length LFSR whose schematic is shown in Figure 5.13.

Figure 5.14 shows the 8-bit LFSR used to generate the symbol synchronization pream-

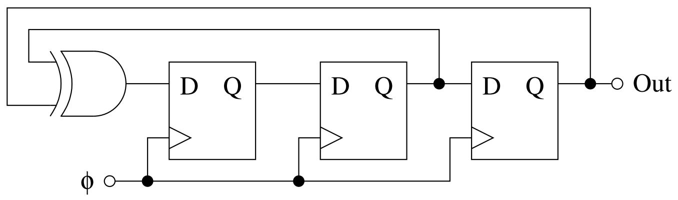


Figure 5.13 – Packet detection LFSR schematic.

ble. The size of the preamble can be adjusted to be one of four values: 32, 64, 96, or 128. For the 128 bit preamble, the complete sequence shown in Figure 5.14 is sent. For smaller sizes, only the last part of the sequence is transmitted. Because the last part of the sequence needs to be transmitted, the initial condition for the LFSR is different for each transmitted preamble size. Table 5.3 shows the initial states for the LFSR for each of the transmitted preamble size.

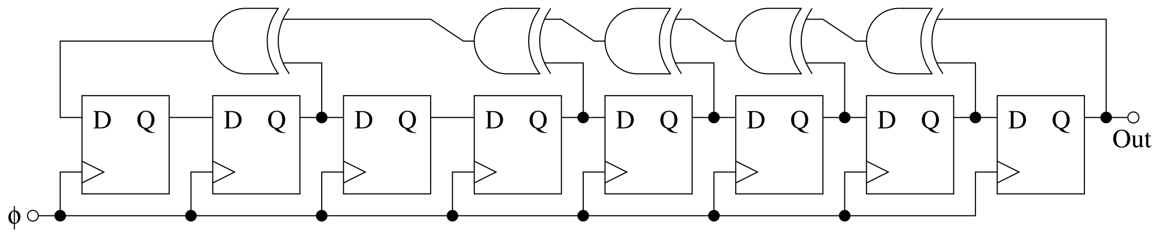


Figure 5.14 – Symbol synchronization preamble LFSR generating the following 128 bit sequence:

0000101001010101000101110111010111001001110110000101100011111101
1010110011011011100001110001101010011111000100001100101000000111

Table 5.3 – LFSR Initial Conditions

Preamble Size	Inital Condition
32	11111001
64	00110101
96	10010011
128	01010000

The polarity of the transmitted symbol synchronization preamble can be selected at the transmitter. The receiver will then use this information to determine which one of the two previously agreed upon rates to use. Before the data goes to the modulator, the polarity of the bits may be flipped. This effectively reverses the one and zero position in the transmitted UWB symbols.

Two signals, a UWB clock and data are output from the digital section. These can be viewed or overridden externally. There are two clock pulses per UWB symbol. In the location where the UWB symbol is to be present the data is one, otherwise the data is zero. So the modulator does not worry about PPM modulation, it only produces a pulse or does not produce a pulse for each input clock cycle.

5.5.2 Modulator Output Stage

The schematic of the modulator was shown in the lower part of Figure 5.12. If the input is high on the rising edge of the clock it can produce one or more pulses of adjustable width. The adjustable delay is implemented as shown in Figure 5.15. It consists of four stages, each of which can select either a longer or shorter path to affect the delay. The NAND gate in each path disables any digital transitions in that path when it is not being used and therefore saves power. Of course, there is a fixed minimum delay though the multiplexers.

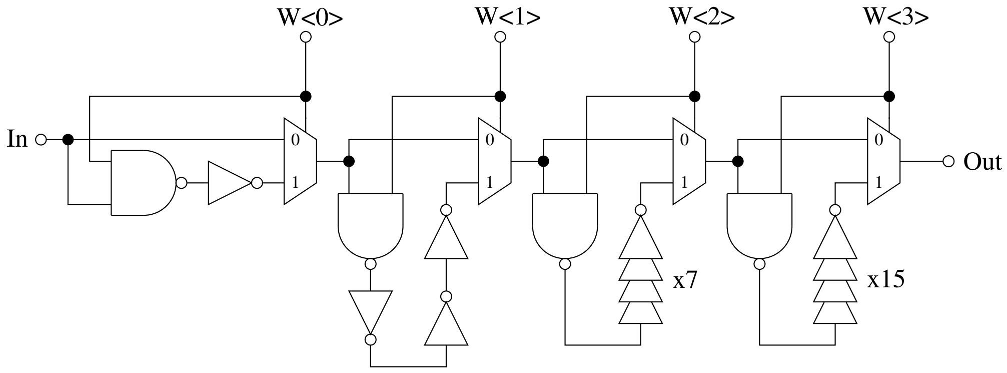


Figure 5.15 – UWB transmitter adjustable delay

The delay is fed-back on itself to create a ring oscillator. The number of cycles before the ring oscillator is disabled is controlled by the repeat select counter and equality comparator. So, overall the modulator produces pulses which range between 500 ps and 4 ns and each pulse can be repeated one to four times. This signal is then driven through a special level shifter which is designed to handle fast pulses which translates the signal from the analog 1.8 V supply to the I/O 3.3 V supply. The two variable-drive drivers then buffer the signal to the pad. The variable-drive drivers themselves consist of four binary-weighted three-state drivers, each of which can be enabled or disabled independently. The schematic of a single driver is shown in Figure 5.16. Four matched versions of Figure 5.16 are used for each driver. Each individual three state driver must be matched with the others so that

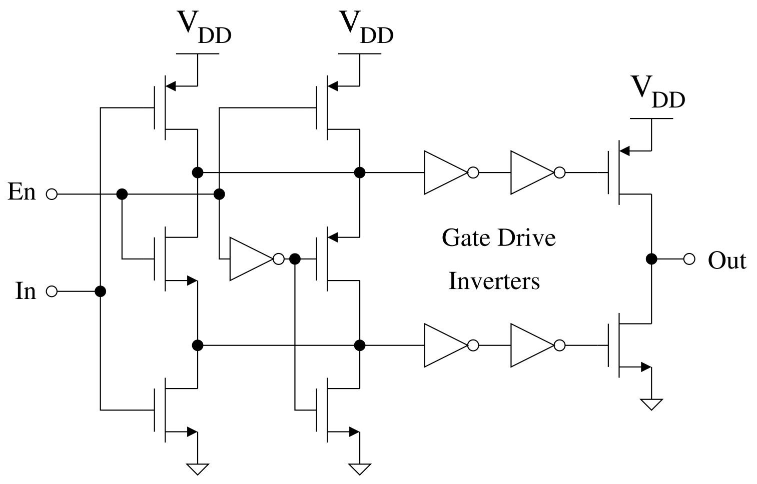


Figure 5.16 – Single UWB output stage.

when multiple drivers are enabled they do not fight each other. Also, the gate drive inverters are carefully sized to introduce a very small dead time when switching. The driver shown in Figure 5.16 is used in the output section of digital pads used in the third and fourth chips.

The UWB transmitter has changed much from chip to chip with the exception of the UWB driver (coincidentally, the UWB driver is the only layout used on the third and fourth chips which is identical to the layout used on the second chip, all other layouts on the entire chip were changed when going from the second to the third chip.) An initial UWB transmitter was designed for the first AMS chip, but it did not work for some unknown reason. It was designed by a third party and was therefore difficult to debug.

In the second chip, the exact modulation for the UWB signal was not known, as this was going to be determined by the UWB receiver which was not yet designed. Therefore, the transmitter could produce many different modulation waveforms, with or without a reference pulse. However, as it turned out, it could not easily produce the one used in the actual UWB receiver. Therefore, it was re-designed in the third chip to what has been presented. However, the UWB driver was kept from the second chip. The third and fourth chip have identical UWB modulator stages, however the digital architectures were changed. In particular, in the fourth chip the convolutional encoder and the interleaver were added.

5.6 Receiver RF Front End

The UWB receiver front end architecture is shown in Figure 5.17. It is responsible for amplifying the received signal, finding the power of the signal, and controlling the gain to keep the electronics from saturating.

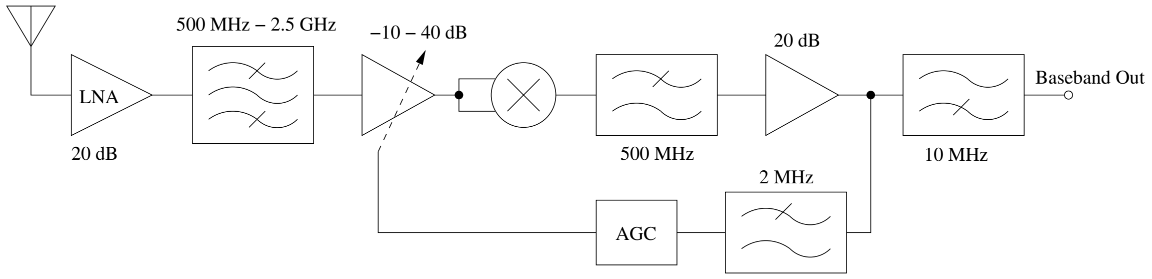


Figure 5.17 – UWB receiver front end.

Most of the power consumed by the entire UWB receiver is taken in the RF front end. In particular, the LNA and subsequent RF gain stages require about 13 mW of power, out of the total UWB receiver power consumption of 20 mW (simulated). Due to this fact, the RF section of the signal chain is single-ended to conserve power. After the squarer circuit however, the signal chain is differential and stays that way throughout the baseband section.

5.6.1 LNA

The LNA of the receiver is a very important block. It generally sets the receiver noise figure as well as the input matching. It is also critical in terms of power consumption. Because this is a UWB receiver, it must have a very wide bandwidth which makes the circuit particularly demanding. The following sections will discuss the LNA design principles and the implemented design.

5.6.1.1 Preliminaries

Standard narrowband LNAs are a well-known and researched circuit. UWB LNAs are much more difficult. One common approach is to take a standard narrowband inductively source-degenerated LNA and decrease the Q or add multiple resonators to widen the bandwidth to make it wideband. For example in [73–76] a standard narrowband LNA is modified with multiple resonators on the input to provide input matching over a range of frequencies. This can achieve good wideband input matching and has good noise figure at frequencies around where the standard narrowband LNA is matched, but possibly quite poor performance at other frequencies.

A common-gate stage can be used to match the input, [77–80] which provides a good wideband match, at the expense of somewhat poor noise figure at all frequencies. Nonetheless, the maximum noise figure can be better than the previous architecture. One can also build circuits which attempt to cancel the noise in the common gate stage, but these also add noise with the added circuitry [77].

A third matching method is with a feedback resistor from the drain to gate [81]. This provides good wideband matching, though the resistor adds noise. While in [82] a shunt-series feedback is used which is where the feedback resistor is used but in conjunction with a source-degeneration resistor. These feedback topologies also have problems with stability due to their negative feedback.

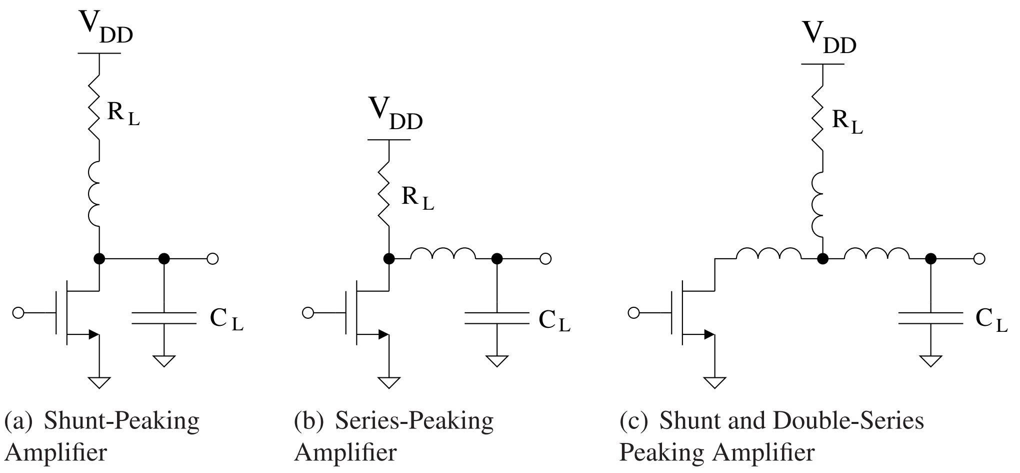
The common gate LNA is the approach used on this project. It provides good matching over a wide bandwidth, requires few inductors, and can be stacked to minimize power. However, the $1/g_m$ input matching sets the bias conditions for the input transistors which are not necessarily optimal for best noise performance. For a complete analysis of the matching and design see [78].

Multiple amplifiers can be stacked to reuse the bias current. This increases gain at the expense of headroom and linearity [74,75]. This approach was used in this design as well.

Most of the UWB LNAs make some use of various techniques for bandwidth extension which were pioneered and used in the 1930s for television applications. Normally resonant structures are used as the loads of narrowband LNAs, however for wideband LNAs these must be altered and use different techniques such as shunt-peaking, series-peaking, and shunt-series peaking.

In a standard amplifier with a resistive load and no inductor, the first pole in the frequency response is given by $\omega = 1/C_L R_L$. However, it is possible to introduce a zero close to the pole to increase the bandwidth. Various schemes to do that are shown in Figure 5.18.

In the shunt-peaking amplifier shown in Figure 5.18(a), the inductor creates a zero (as well as more, possibly complex, poles). After much derivation [83], this can be shown to extend the bandwidth by a factor of up to 1.8 times without any increase in power. Intuitively, this circuit works by increasing the impedance through the R_L circuit branch at high frequencies, thereby boosting the gain just as the gain would naturally start to roll-off. Or, said another way, it keeps the high-frequency signal current from flowing through the



V_DD
R_L
C_L
(a) Shunt-Peaking Amplifier
V_DD
R_L
C_L
(b) Series-Peaking Amplifier
V_DD
R_L
C_L
(c) Shunt and Double-Series Peaking Amplifier

Figure 5.18 – Bandwidth enhancement techniques

R_L branch of the circuit and instead it all flows through C_L, increasing the gain. Of course, parasitic MOSFET capacitors are also modeled in C_L.

In the series-peaking amplifier shown in Figure 5.18(b), the charging and discharging of the load capacitance is split from the charging and discharging of the parasitic MOSFET capacitance. For this type of circuit there is no simple maximum bandwidth expansion factor, because it depends on the size of C_L and the size of the parasitic MOSFET capacitance. This type of circuit is good where the load capacitance is large. The added inductor allows the circuit to sequentially charge the parasitic capacitance and the load capacitance which increases the speed instead of charging or discharging them together. The shunt and series peaking amplifiers can be combined to achieve more bandwidth extension.

Adding even more inductors, one arrives at the shunt and double-series peaking amplifier shown in Figure 5.18(c). You should be noticing a trend toward something that looks like a distributed load! In this scheme, a bandwidth improvement of up to 2.8 times can be achieved. It combines the advantages of both types of amplifier. The drawback is of course the three inductors present. However, the three inductors can conveniently be implemented as a pair of coupled inductors (i.e. a transformer). On the die two inductors can be drawn overlapping to produce a transformer. The coupling between the inductors is important, and sets the amount of frequency-response peaking, and to some degree, the amount of bandwidth expansion. The coupling between the two coils can be set by the amount of overlap between the two inductor coils.

Using a transformer is great in theory, however actually implementing one is difficult. Most importantly, the process we are using does not support dual thick metals. Therefore, one inductor would be a good one using the top thick metal, and the other would be a poor one using lower thin metal layers. This thin-metal inductor would be very inferior to the already-poor thick-metal inductor, negating many of the gains this circuit promises. Furthermore, the standard design flow does not support the estimation and modeling of the coupling between two inductors, so finding the proper overlap would require a third-party inductor simulator, a method fraught with difficulty.

So instead, a standard fixed-coupling center-tapped symmetric inductor was used as the transformer. This component is available as a standard p-cell in the IBM kit complete with spice models which properly model the coupling. It is designed to be a symmetric inductor for use with symmetric differential circuits such as VCOs. With the optional center-tap, it becomes an autotransformer with a high fixed coupling coefficient, and a 1:1 turns ratio. This high fixed coupling coefficient then sets the frequency peaking of the circuit to be substantial, however, the overall response can be flattened through the creative use of multiple stages. The transformer configuration is called a T-coil.

This use of a symmetric inductor as a T-coil load is unique among other wideband LNA and amplifier designs. Other designs generally simply use series peaking and occasionally shunt peaking.

The input matching of the LNA is also another challenge. Conventional narrowband LNAs use resonant structures to match the input at a given frequency and achieve good $S_{1,1}$. However, broadband matching is much more difficult. It is generally done either with feedback resistors or by using a common-gate input stage. The input impedance of a common gate input stage is $1/g_m$, so if the input transistor has a g_m of 20 mS, then 50 Ω matching then results. This is the method chosen for this design.

5.6.1.2 Design

The schematic of the LNA is shown in Figure 5.19. It actually consists of two gain stages stacked on top of each other to share the large bias current to get more gain for the same power consumption. The first stage consists of L1, M1, M2, L2, and R2. This is a common-gate stage for input impedance matching. L1 presents a low-impedance to bias currents but high-impedance to signal currents. L2 and R2 are the T-coil load which allows for much bandwidth expansion as described in the previous section. The g_m of the input transistor must be 20 mS for correct matching, so the transistors are large with much current through them.

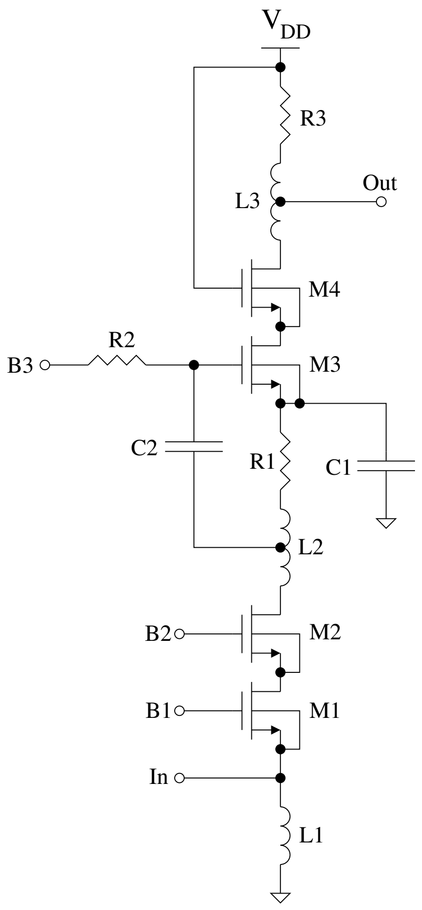


Figure 5.19 – Stacked LNA schematic.

The second gain stage which is stacked on top of the first gain stage consists of everything from the source of M3 upwards. Capacitor C1 is very large and effectively produces an AC ground at the source of M3 upon which the upper gain stage is referenced to. If it is not large enough, stability problems may result due to coupling between the stages. The upper stage is biased from B3 through R2 and the signal is coupled in through C2. This stage is a basic common source stage with a T-coil load as previously discussed. Each stage in the LNA provides about 10 dB of gain for a total LNA gain of 20 dB. The bias current is 1.9 mA for a power consumption of 3.4 mW.

Figure 5.20 shows the simulated noise figure of the LNA and following gain stages. It has a minimum noise figure of 3.4 dB and a maximum noise figure of 8.2 dB at 500 MHz.

The simulated input matching is shown in Figure 5.21. This shows a matching of less than -10 dB from 580 MHz to more than 10 GHz.

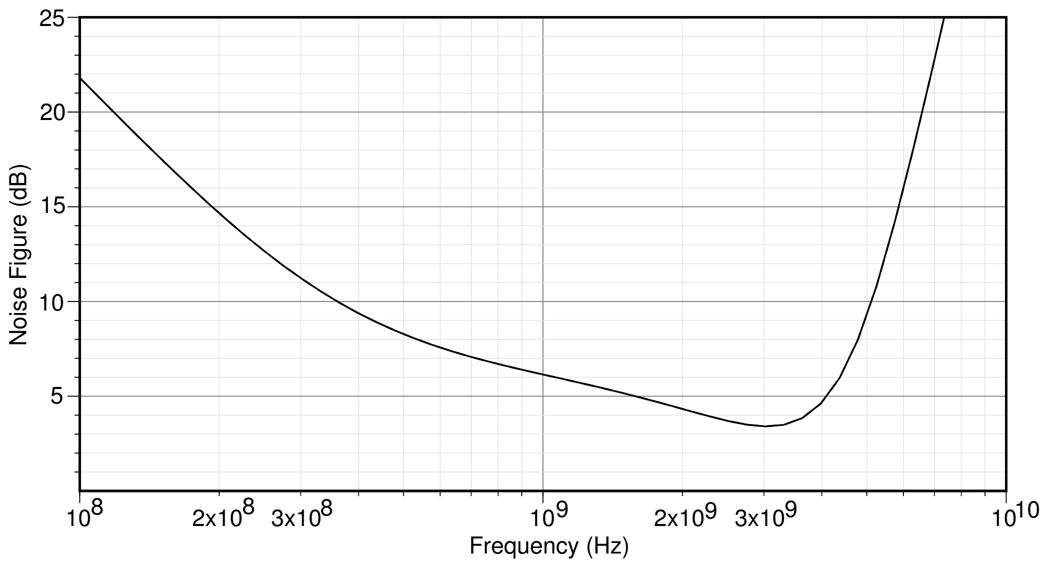


Figure 5.20 – UWB receiver noise figure.

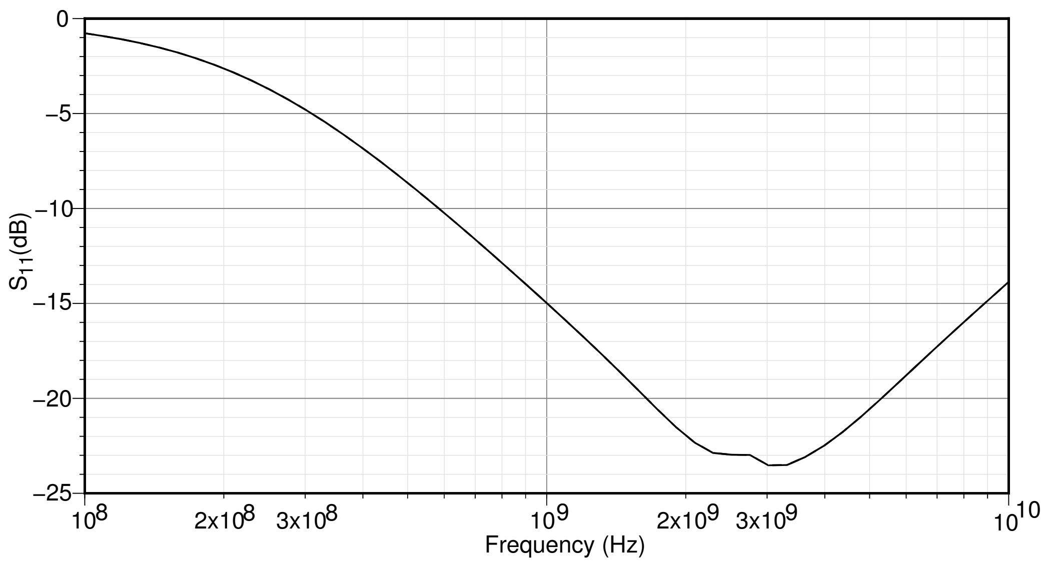


Figure 5.21 – UWB receiver input matching.

5.6.2 Gain Stages

After the LNA, an adjustable amount of gain of up to 50 dB is applied. This is done with 5 identical cascaded gain stages. The schematic of one gain stage is shown in Figure 5.22. It is a simple common cascode stage with a T-coil load. It has an integrated high pass filter formed by R1 and C1 which sets the low-end bandwidth cutoff. The gain of the stage is adjustable by adjusting the BiasN voltage. The bias current through each stage is 1 mA for

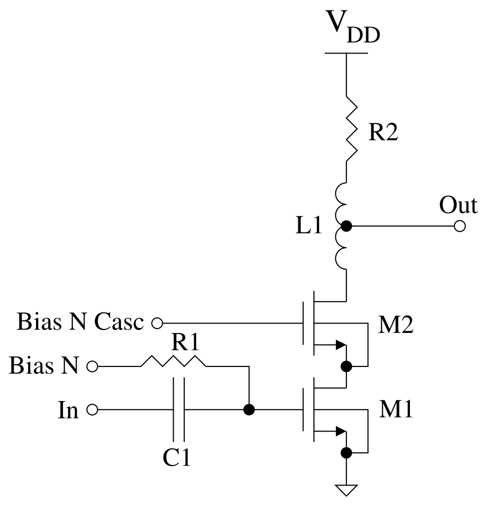


Figure 5.22 – UWB gain stage.

a total combined power dissipation of all gain stages of 9 mW. This is the maximum but it decreases dramatically with a lower gain setting.

5.6.2.1 Multi-Peak Response Flattening

As previously discussed, the transformer load, or T-coil, which is used as a load for the LNA and gain stages has a fixed, relatively high, coupling coefficient. The high coupling coefficient makes the maximum possible bandwidth extension large, at the expense of a large peak in the frequency response. The only way to decrease the size of the peak is to change the coupling, but that is not possible with this process. Therefore, to get a flatter frequency response, the peaks of multiple stages can be staggered so when they are cascaded they produce a more even response. The location of the peak and also its size is largely affected by the value of the series resistor.

In this design there are 7 effective stages, two in the LNA and 5 gain stages following. It simplifies the design to have the 5 gain stages be identical. They were each chosen to have the maximum frequency extension and a peak at 2.3 GHz. The first stage of the LNA has a peak at a high frequency of 2.8 GHz, while the second LNA stage as a peak at 1.3 GHz. This low-frequency peak is responsible for offsetting the dip which occurs at this frequency in the frequency response of all other stages. Summed together, they produce a frequency response which is within 2 dB with the maximum gain setting. The gain flatness gets worse with lower gain settings because the gain of the LNA does not change while the gain of the other stages do. Figure 5.23 shows a simulation of the frequency response after each stage as it progresses through the amplifier chain.

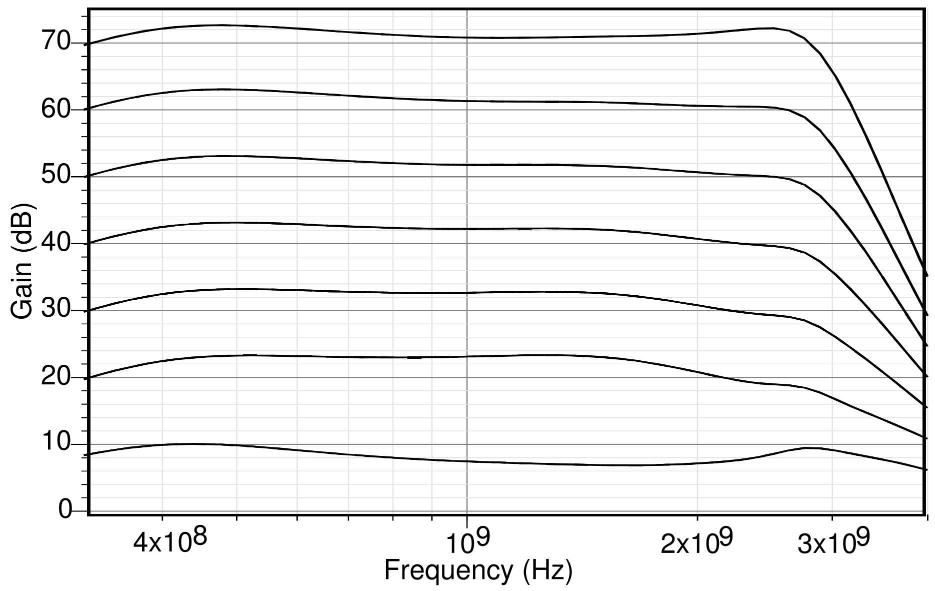


Figure 5.23 – Outputs of each stage

5.6.3 Self-Mixer

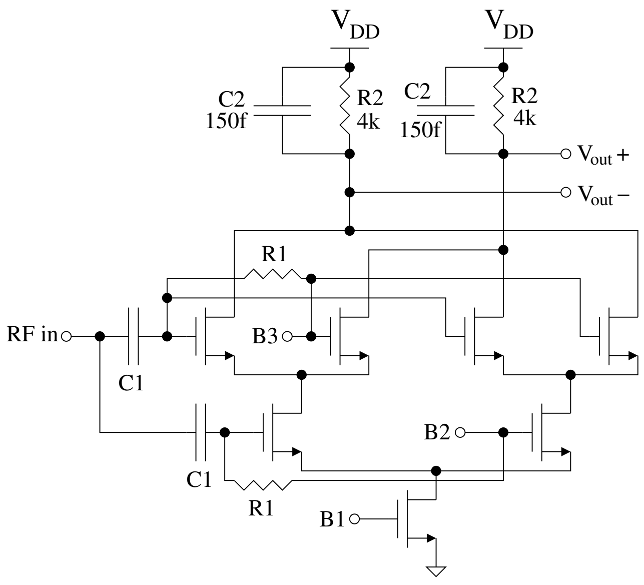


Figure 5.24 – UWB self-mixer schematic

The self-mixer squares the signal to produce the energy. It is a straightforward four-quadrant Gilbert multiplier, the schematic of which is shown in Figure 5.24. The one RF input goes to both inputs of the Gilbert multiplier so a squaring operation takes place. The only difficultly with the circuit is generating the three bias voltages required. The one at the bottom, B1, sets the current through the mixer to be 365 μA. R2 and C2 produce a low-pass filter to filter out the double-frequency terms from the squaring operation. A second stage of the low-pass filter is present in the gain stage after the mixer.

The signal levels in the mixer are such that there is generally some attenuation through the mixer, so the inputs are around 20 mV and the output is around 10 mV. The signals are kept at these levels via the AGC loop. Before the mixer the signals were always single-ended to save power; after the mixer the signal chain is differential. After the mixer the bandwidth is much lower so the power consumption of these stages is substantially reduced. Therefore keeping the signals differential has advantages for noise immunity and biasing.

After the mixer, another 20 dB of gain amplifies the signal and provides a very strong driver which will drive all the baseband processing circuits. There is substantial capacitance associated with the baseband processing circuitry, so a very robust driver is needed. The driver consists of a current-mirror gain stage followed by an output stage. The schematic of the gain and driver is shown in Figure 5.25. It is an open-loop circuit, as the bandwidth

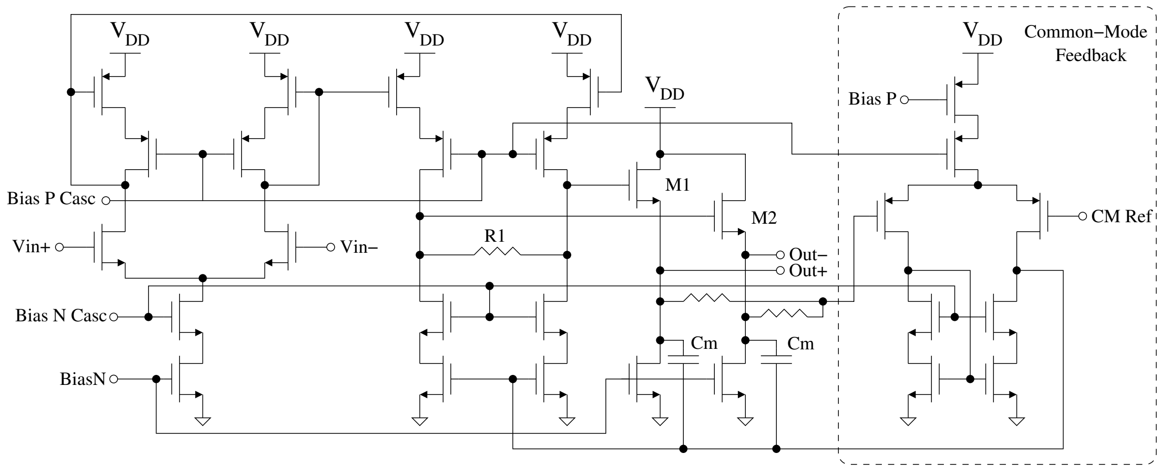


Figure 5.25 – UWB baseband gain

is still too great to use a closed-loop architecture. The gain is set by

$$A = g_m R1 \tag{5.2}$$

where g_m is the transconductance of the input differential pair.

After the current mirror gain stage, a source follower drives the output. Transistors M1 and M2 are zero-V_t transistors to minimize the signal swing range degradation. The common-mode feedback circuit uses Miller compensation with the two Miller capacitors Cm. In the third chip, a simple lead compensation scheme was used. However, this was a contributing factor to the oscillation of the AGC control loop in that chip. This is because the common-mode feedback was not fast enough, and large signals would temporarily disturb the common-mode level in such a way it would interfere with the AGC circuit downstream and set up an oscillation. The Miller feedback capacitance fixes this problem as now there is an extremely fast path between the output and the common-mode level which will keep the common-mode level correct continuously. The Miller capacitors are also the compensation for the common-mode feedback. The gain of the amplifier circuit rolls off at about 500 MHz and works to further filter the double-frequency terms from the mixer.

5.6.4 AGC

The AGC control loop is responsible for adjusting the gain of the front end so the downstream circuits do not saturate. It is mostly concerned with keeping noise and narrowband interferers from saturating the front end—if the UWB signal saturates the front end, it is not too much of a problem. Therefore, the AGC loop works by keeping the DC component of the energy signal at a small fixed amplitude. This is achieved by first subtracting the baseband from a fixed value. After it is low-pass filtered, it goes to a circuit which generates the bias to drive the front-end gain stages.

The first step is to subtract it from a fixed value, the target value. The target value is set with a very simple low-power differential DAC. This DAC is shown in Figure 5.26. It simply consists of two adjustable resistors with a fixed current flowing through them. The voltage is held centered around the common mode level via a common-mode feedback circuit. The current reference Iref is derived from the main system current reference.

The DAC's output is subtracted off the baseband signal with a differential difference amplifier and the result is integrated with the circuit shown in Figure 5.27. It is uses source-degeneration to linearize the transconductance. The input stage used assumes both input signals have the same common-mode level, which is the case. The integrator is a g_m-C integrator with a digitally adjustable capacitor. There is also the option of using

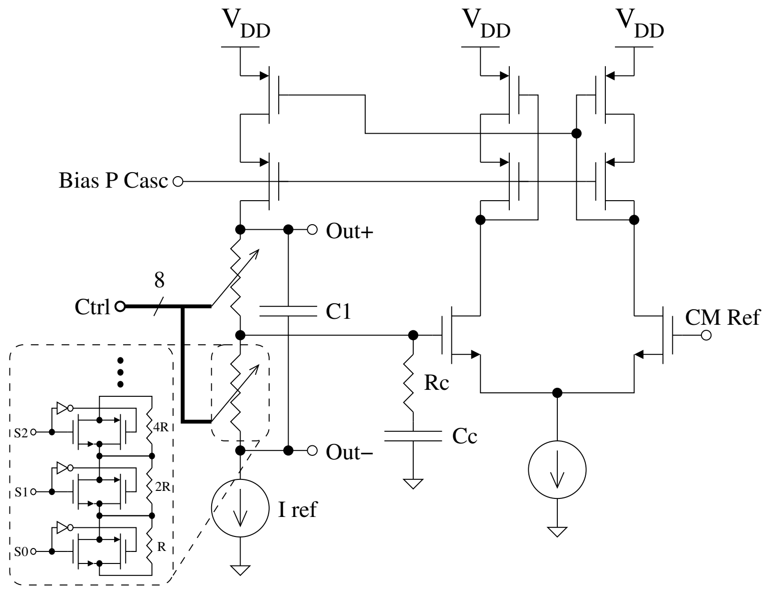


Figure 5.26 – Differential DAC

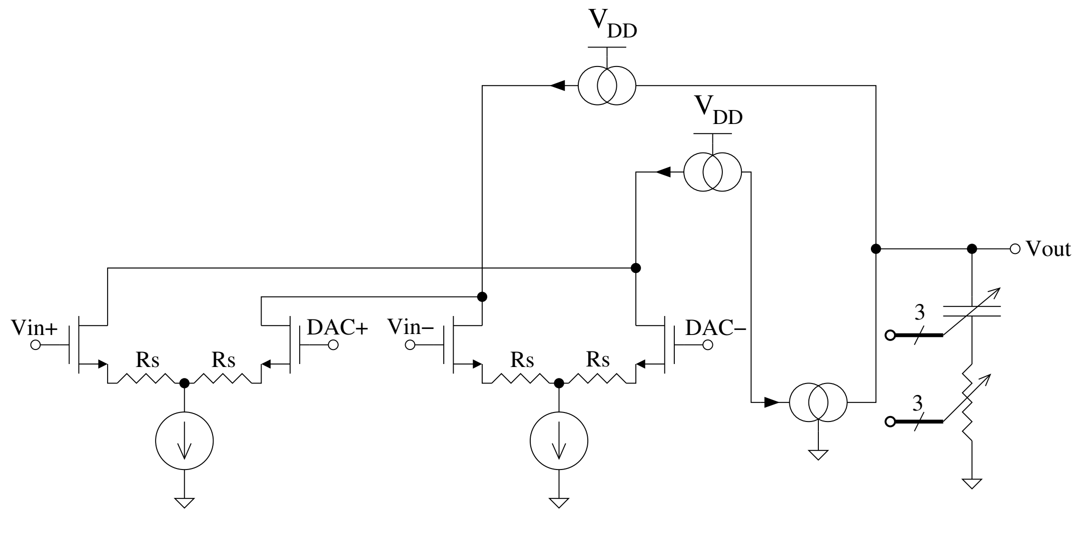


Figure 5.27 – UWB AGC amplifier and loop compensation

lead-compensation to help stabilize the entire feedback loop which is implemented with a digitally adjustable resistor.

The output of this integrator now goes to the bias generator which produces the biases required for the initial gain stages. The goal is to have a linear relationship between the integrator's output voltage and the g_m of the gain stages which in turn sets the circuit's gain. For this purpose, a standard constant-g_m bias circuit was modified to use a triode transistor in place of the bias resistor. This is shown in Figure 5.28

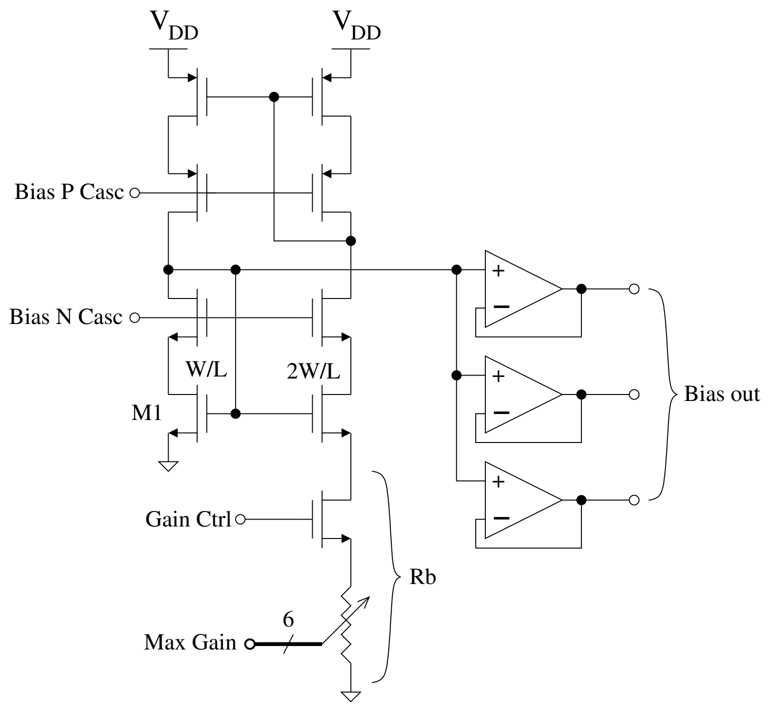


Figure 5.28 – UWB gain stage bias control

The g_m of transistor M1, which is proportional to the g_m of the following UWB gain stage, is given by [64]

$$g_{m1} = \frac{2-\sqrt{2}}{R_b} \tag{5.3}$$

where R_b is labeled in Figure 5.28 and is composed of a digitally adjustable component which sets the maximum gain and another component set by a triode transistor. The triode resistance is given by

$$r_{ds} = \frac{1}{\mu_n C_{ox}(W/L)(V_{gs} - Vt)}. \tag{5.4}$$

Combining Equation 5.3 and Equation 5.4 we get

$$g_{m1} = k'_p(V_{gs} - Vt)(2-\sqrt{2}) \tag{5.5}$$

where $k'_p = \mu_n C_{ox}(W/L)$ and we have assumed the value of the digital resistor is zero. Increasing the value of the digital resistor lowers the maximum g_m value and thus the gain. As can be seen, the g_m is now linear with the applied voltage, V_{gs}, as long as the transistor stays in the triode region.

After this bias loop, the bias voltage is buffered by three op-amps. The main purpose of these op-amps is reverse isolation. Without this isolation, the signal from a later stage will couple into the bias line, and in turn get injected in the loop at an early stage and cause the loop to oscillate. This bias voltage actually drives 5 stages; two of the op-amps drive two adjacent gain stages. Adjacent stages can share the same bias line and will not oscillate because the gain through the loop is not large enough to cause oscillation. Extreme care must also be taken during layout to assure that no signals from later stages are coupled into the bias voltages going to earlier stages. This is achieved by using a complete ground shield around all bias lines going to the LNA and early gain stages.

The bias coupling problem is not unique to this bias, and is a problem when supplying all of the 16 bias voltages needed for the RF front end. The solution consists of using op-amp buffers to isolate the later stages from the bias of the earlier stages, keeping in mind that adjacent stages can share a bias.

5.6.5 Front End Oscillation

On the third chip, which was the first to include the UWB receiver, the front end oscillated. Though all observed aspects of the oscillation could never be replicated in simulation, it was apparent that the oscillation was caused by a number of factors including:

- The sharing of the supply of the RF front end with that of the baseband circuitry
- The the common-mode feedback circuitry on the baseband gain not being fast enough causing large perturbations in the common-mode level if the input saturated
- The isolation between later stages and earlier stages, and the isolation between the baseband and the front-end section not being good enough.

First, the baseband circuitry caused problems with the front end; this circuitry has some digital in it and has a large number of analog circuits which process (comparatively) large signals which are still high enough frequency they can couple readily to the power supply and to other circuits in the vicinity. Furthermore, the complete UWB receiver with the front end and the baseband is an extremely large circuit, and it takes days to simulate even a single UWB symbol. Therefore, the complete circuit was not extensively simulated. Instead, the circuit was split and the two parts simulated separately so complex interactions were not explored in detail in the simulator.

The sharing of the supply with the RF front end and the baseband circuitry caused the baseband signal to couple to the supply, which in turn coupled back to the front end causing oscillations. Also, though not causing oscillations, the baseband circuitry also coupled digital hash onto the supply which was also amplified by the front end and raises the noise floor. For this reason, in the fourth chip the two supplies are kept separate until outside the chip. This is not a cure-all as any signals crossing the boundary are particularly problematic and cause current to flow in very large loops, therefore one must be careful in determining where to split the supplies. Of particular concern are digital signals which cross supply boundaries.

As stated before, the common-mode feedback circuit in the baseband gain circuit was very slow, and therefore could allow the common mode level to drift very far if the input was recovering from a saturated condition. The saturation could come from, for example, the digital buck power supply switching. This would cause the inputs to the next stage, the differential difference amplifier, to go out of their normal range and cause that amplifier to malfunction. If the common-mode level of the Vin terminals in Figure 5.27 go too low, the output will go high which will in turn tend to force the front end into saturation again, repeating the process. This by itself does not explain all the oscillations, but is a contributing factor in the size of oscillations.

Finally, the isolation between the later stages and the earlier stages as well as the isolation between the digital part of the baseband section and the analog front end is the root problem with the oscillation. Simply put, any feedback from the later stages to the bias inputs of the earlier stages, either via capacitive or direct connection can cause an oscillation.

For the final chip all aspects were addressed with a particular focus on isolation. The existing barrier between the analog front end and the baseband circuitry was greatly strengthened by using a wall of metal and staggered via layers between the two circuits. The isolation within the front end between earlier and later gain stages was also strengthened. Furthermore, attenuating capacitors were placed on the bias lines of the gain stages to filter out the highest-frequency signals. These methods were sufficient to stop the oscillation and there was no measurable oscillation in the fourth chip.

5.7 Packet Detection

A block diagram of the packet detection circuitry is shown in Figure 5.29. This circuit is responsible for finding the repeating 7-bit PN code present in the first preamble. It consists of 16 interleaved simple receivers, each designed to detect the start of the packet. The receivers are each looking for the preamble in a different place, but between them all,

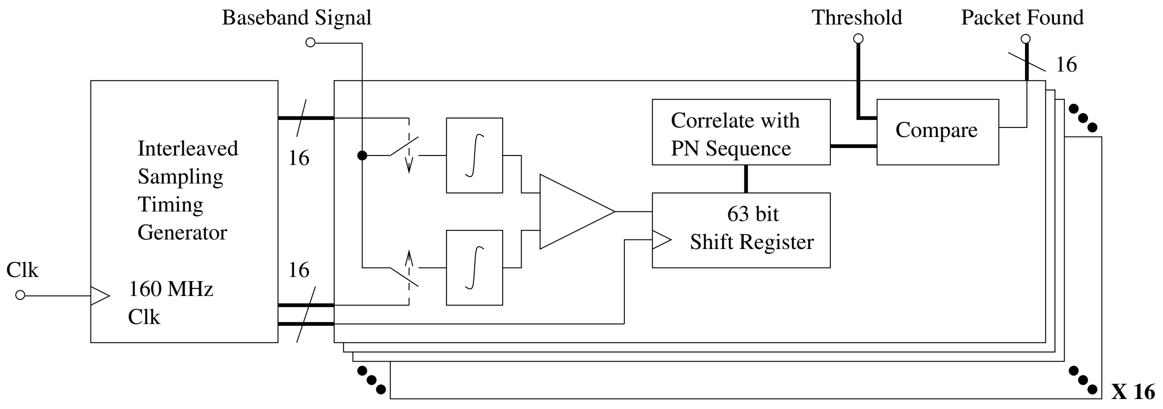


Figure 5.29 – Packet detection circuitry

they cover the entire range of possible locations for the packet to start. The receivers can be configured to overlap with each other to maximize probability of detecting the packet. Each receiver consists of a simple energy detector circuit, which chooses which position has more energy, and a correlation operation with the expected PN code repeated a number of times. The number of bits which must be correct for the preamble to be recognized as valid is adjustable as is the length of the preamble. Unlike the correlator which looks for the second preamble, this one is only sensitive to positive correlation peaks and ignores negative correlation peaks.

There are 16 clock cycles of the 160 MHz clock within each UWB symbol period. Each of the 16 correlators start on a different clock edge within the symbol time. The integration time of all the integrators can be adjusted to be from one to four clock cycles. This allows flexibility to adjust the integration time to suit different channel environments. It also allows the use of a slower clock speed and slower communication rate while still maintaining the same integration time. When using integration times longer than one clock cycle, two or more integrators will overlap and both be integrating the same signal for a portion of the time. This increases the chance that one of them will successfully detect an incoming packet. The integrator waveforms of four of the packet detection receivers is shown in Figure 5.30.

The symbol synchronization circuitry is normally off and held in the reset position until a packet is found by one of packet detect correlators. At that point the symbol synchronization circuitry and bit detection circuitry are enabled and the packet detection circuitry is disabled.

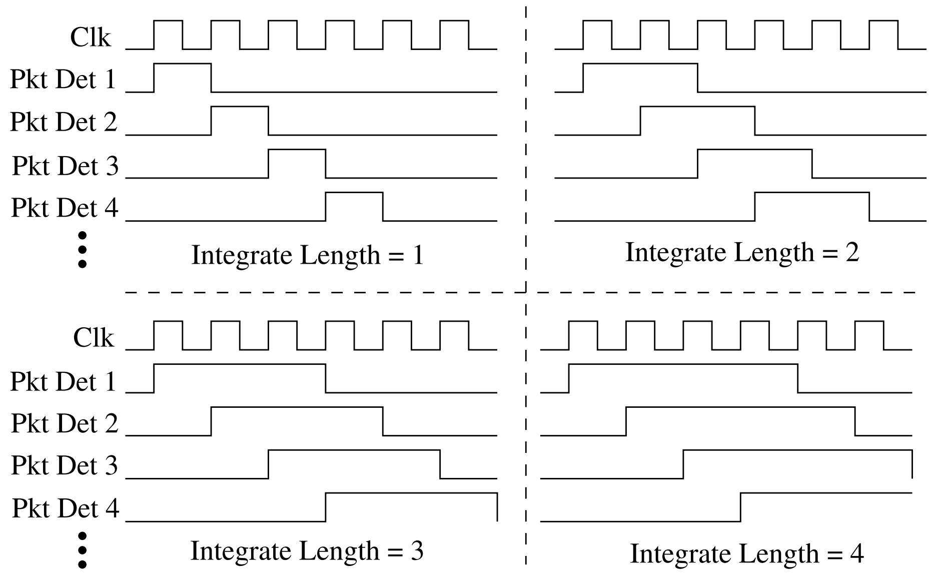


Figure 5.30 – Packet detect integrator waveforms with varying integration length.

5.7.1 Packet Detection Performance

The performance of the packet detection circuitry is critical to the performance of the entire UWB receiver. The packet detection circuitry affects the overall performance by improving the probability that the start of an incoming packet will be recognized as such. If the start of a packet is not recognized, the rest of the packet will not be received. There are two ways in which the packet detection circuitry can decrease the packet reception rate—through false positives as well as false negatives (misses). False negatives occur when a packet is being transmitted but due to symbol errors the packet is not recognized. False positives occur when the packet detection circuitry falsely detects the presence of a packet. A small number of false positives are quite benign as the second packet preamble, the synchronization preamble, will not be found and after a time the UWB receiver will decide no packet is present and revert to searching for a packet. However, during this time the receiver is searching for the second synchronization preamble, if an actual packet is transmitted, it will not be received by the UWB receiver. Therefore large numbers of false positives will increase the probability the UWB receiver will miss packets—even very strong packets.

The probability of false positives and false negatives is determined by the length of the correlator in Figure 5.29. This is run-time selectable to be from 16-63 bits in length. Longer correlator lengths give better packet detection performance at the expense of requiring larger preambles, and therefore overhead, for each packet. A maximum 63 bit correlator length was chosen as as a tradeoff between performance and hardware complexity. The hardware requirements for those 16 correlators are substantial, so using longer correlators comes at much hardware cost and power. The correlation function is defined as

$$R = \sum_{i=0}^{N-1} \overline{M(i) \oplus PN(i)} \tag{5.6}$$

where N is the number of bits in the correlator, M is the received symbol sequence, and PN is the expected PN code.

To simplify the analysis, we will assume that if there is no packet present either symbol is equally likely. In actuality, this is not the case because of offsets within each packet detection receiver which cause each one to favor one symbol state over the other, but this effect is quite small and will be neglected. A further assumption is that each receiver is independent of the others. This is also not the case, but has little effect on the actual performance. Both of these assumptions are shown to be correct in subsection 10.2.2 on page 256 where it is seen that the measured false alarm rate matches the predicted false alarm rate very well. The probability of a false positive is then the probability that the correlation value will exceed a user-defined threshold given random input bits. This can be calculated by

$$P_{fp} = \frac{\sum_{i=0}^{N-k} \binom{N}{i}}{2^N} \tag{5.7}$$

where k is the correlation threshold which must be exceeded to be considered a correct packet, or stated another way, N-k is the number of bits which may be in error and the packet will still be received. If we assume $N \gg (N-k)$, i.e. k is “close” to N, then the sum in the numerator can be approximated as being only the largest term, so the formula then becomes

$$P_{fp} \approx \frac{\binom{N}{N-k}}{2^N}. \tag{5.8}$$

The false alarm rate, as measured in false alarms per second, is approximately the probability of a false alarm times the clock frequency which is 160 MHz in this case. This is given by

$$R_{fa} \approx \frac{F_{clk}\binom{N}{N-k}}{2^N}. \tag{5.9}$$

The above equation is an approximation which only holds if the false alarm rate is small.

Finally, after each false alarm the UWB receiver attempts to find the second synchronization preamble for a period of time before giving up and returning to packet searching. This duration of time is software selectable. Setting it to a value too small will catastrophically affect the packet reception rate because the UWB receiver will time out before receiving the synchronization preamble on each packet. However setting it to a value too large negatively affects the false alarm recovery time. The percentage of the time that the UWB receiver is recovering from false alarms and therefore not able to receive packets is

$$P_{loss} \approx \frac{T_{far}\binom{N}{N-k}}{2^N} \tag{5.10}$$

where T_{far} is the number of cycles the receiver searches for the synchronization preamble.

This P_{loss} is a percentage of packets which will be lost, irrespective of their incoming signal strength. This is a fixed loss. This loss can be readily decreased by increasing the correlation threshold. Figure 5.31 shows the packet loss rate as a function of the threshold.

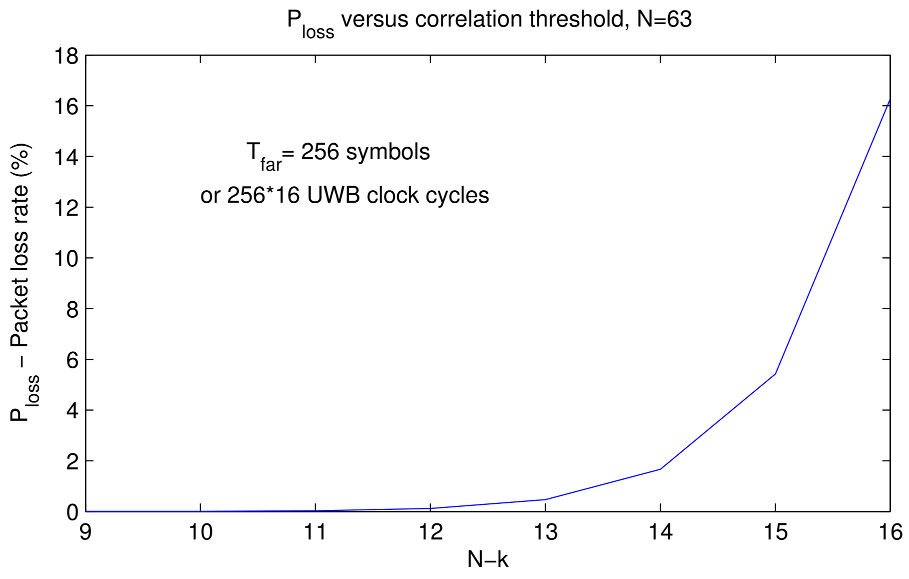


Figure 5.31 – P_{loss} versus N-k with N=63

There is also some probability that the received packet detect preamble will contain too many symbol errors and will not be detected. To simplify the analysis, two assumptions will be made. One is that the receivers are configured so the integration times do not overlap and the incoming pulses are short enough that only one of the packet detection receivers is in a position to receive the packet. The second assumption is that the length of the packet detection preamble sent by the transmitter is the same length preamble which the receiver is expecting. In actuality, the transmitter may transmit a longer preamble than the receiver is expecting and increase the packet detection performance.

With these two assumptions, the following derivation is a lower bound on the performance of the of the packet detection receiver, with the actual performance being possibly much better.

For a given UWB symbol error rate P_s, the probability that packet will be correctly found is

$$P_c = \sum_{i=0}^{N-k} \binom{N}{i} P_s^i (1-P_s)^{N-i} \tag{5.11}$$

where as before k is the correlation threshold which must be exceeded to start a packet.

Figure 5.32 shows the packet recognition rate for varying values of $N-k$. This figure incorporates both P_{loss} and P_c.

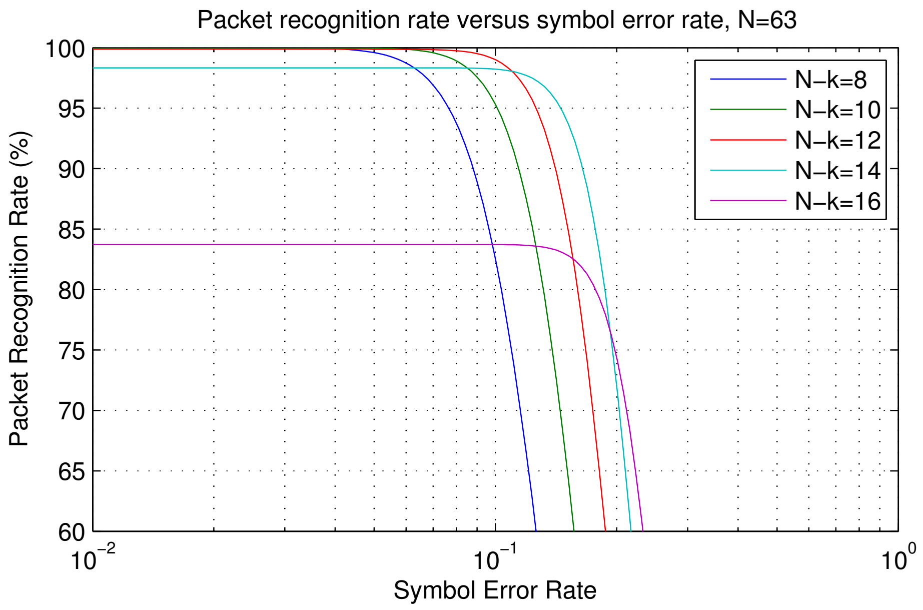


Figure 5.32 – Packet recognition rate versus symbol error rate for varying values of $N-k$.

From Figures 5.31 and 5.32 it is clear that a N-k value of 14 or 15 is a good value to choose when $N = 63$. When $N = 14$, about 2% of the packets will always be lost, irrespective of the incoming signal strength, and the worst-case packet recognition rate when the symbol error rate is 0.2 is 70%. In actuality the recognition rate will be higher because the simplifying assumptions stated before do not hold, so this is a lower bound.

Figure 5.33 shows a simulation of the packet recognition performance versus the symbol error rate and threshold value $N-k$. This is much better than the calculated lower bound in Figure 5.32 for a number of reasons. First, for the simulation the transmitter is transmitting 128 symbols but the receiver only needs 63, so the receiver has multiple chances to correctly recognize the packet. As can be seen, a value of $N-k = 12-13$ would be quite sufficient for detecting over 95% of the packets over the entire range of symbol error rates below 0.2.

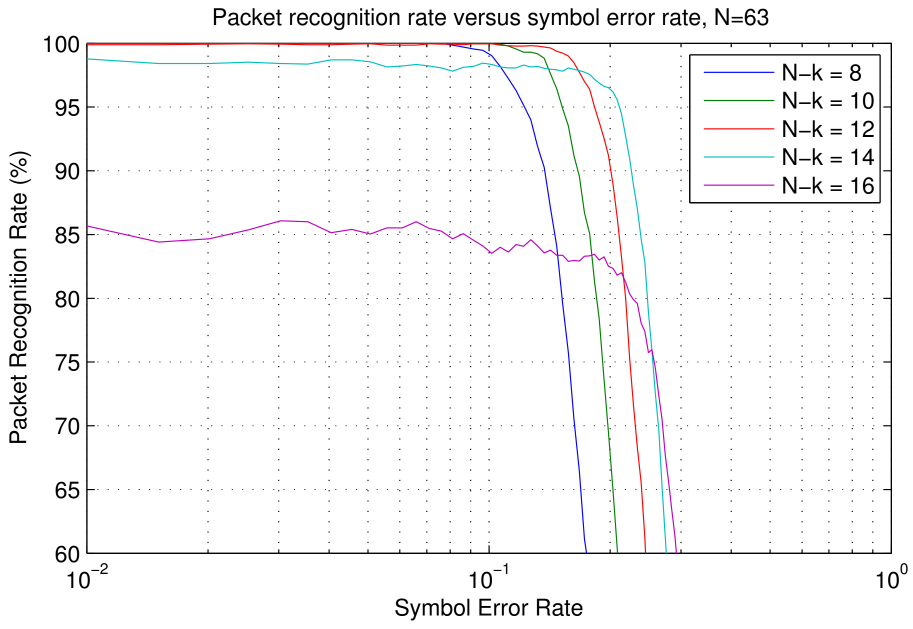


Figure 5.33 – Simulation of packet recognition rate versus symbol error rate for various values of $N-k$.

5.8 Bit Detection

A block diagram of the bit detection circuitry is shown in Figure 5.34. It is responsible for detecting the second preamble and recovering the data from the baseband signal. The first part is a standard energy detector which finds the bit position with the larger integrated energy content. A correlator correlates the output with the PN code in the second preamble. The correlator looks for both positive and negative correlation peaks to determine which polarity of the PN code the transmitter sent.

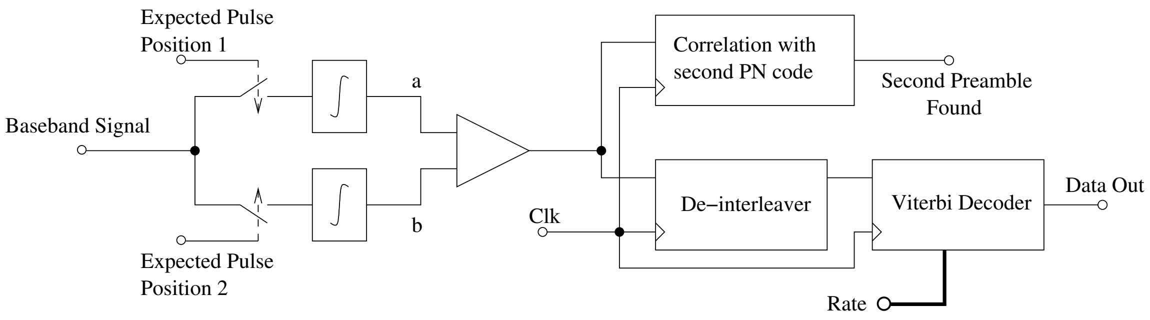


Figure 5.34 – Bit detection block diagram

The bit detection makes a hard decision after every bit, and does not use any pulse repetition to increase the energy per bit, but instead relies on the convolutional code to correct for erroneous bits. Using soft decision decoding is better on the AWGN channel. Also, if pulse repetition was employed as is the case in other conventional receivers, a better decoding on the AWGN channel would be to integrate the soft decisions over all the bits before making a decision. However, the AWGN assumption does not always hold for the UWB channel, particularly in the presence of other UWB transmitters, switching power supplies, or some narrowband interferers. In these cases, the probability density function of the noise in the channel has a good chance to be high variance and not Gaussian. Therefore it is better to use a bit detection architecture which has some type of limiting so large noise values do not propagate to other symbols [59, 84, 85].

The message is recovered by first de-interleaving the data which was interleaved originally at the transmitter. The interleaver is a 64 bit random interleaver, and is present to decrease the effects of short-term saturation of the front end caused by the sudden appeareance of a narrowband interferer. After the interleaver, a variable-rate hardware Viterbi decoder is used recover the message. The rate varies from 1/2 to 1/10. A different rate can be used to encode the first two bytes of the message (the address and packet size) than the rest of the message if desired. The Viterbi decoder can report to the microcontroller the number of symbol errors found in the packet after successful packet reception.

The bit detection circuitry attempts to measure the average value of the integrated signal and noise over the reception of the packet so the microcontroller can access the information for debugging or for determining the signal strength. This is computed with the circuit shown in Figure 5.35. Figure 5.35 corresponds to the actual implementation of Figure 5.34, and the correlation points “a" and “b" are labeled in both. The comparator is also shown in both figures.

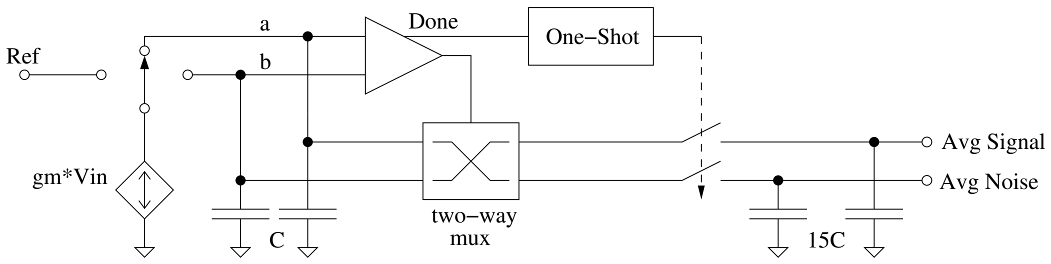


Figure 5.35 – Average signal and noise computation

The switch selects one of the capacitors to integrate the incoming baseband signal onto, or sinks the current into the reference. The switch selects the integration time periods for the two UWB pulse locations. The comparator is a clocked comparator, and it is clocked after both pulse locations have been integrated. After the comparator has resolved the output, the “Done" line goes high, and the one-shot creates a short pulse. This short pulse shares the charge on the integration capacitors with the much larger averaging capacitors. After many UWB symbols, the averaging capacitors contain an average of the signal and noise values. The size of the averaging capacitors are digitally adjustable.

There are a couple methods in which offsets can affect this architecture. First, the offset in the transconductor itself has no effect, as it is integrated on both capacitors equally. This eliminates the largest source of inaccuracy in other circuits which use two transconductors. However, mismatch in the capacitors causes problems as does offset in the comparator. The capacitors can be well matched, while the comparator offset effect can be minimized by making the swing on the capacitors large (i.e. using small capacitors). In the implementation, the size of both the integrating and average capacitors are digitally adjustable. In the implementation, offsets within these receivers did not pose a problem.

The timing of the integrated positions as well as the comparator control signals are programmable and are shown in Figure 5.38. See subsection 5.9.1 on page 154 for more information about the programmable waveform generation.

5.9 Symbol Synchronization

A block diagram of the symbol synchronization circuitry is shown in Figure 5.36. This circuit is responsible for tracking the input data stream and changing the clock to correct for timing differences. The timing correction can be done in one of two ways. One way is by

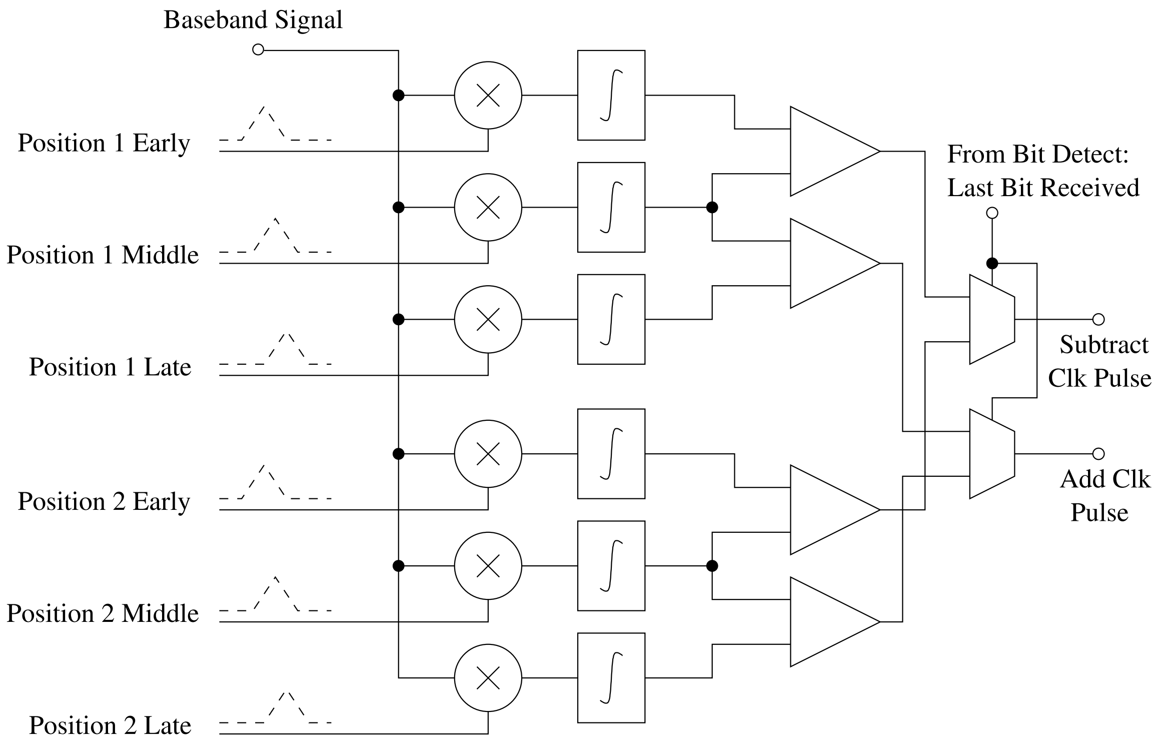


Figure 5.36 – Symbol synchronization block diagram

adding or subtracting clock cycles from the main clock to stay synchronized. The other way only works if the clock is being generated by the PLL, where the symbol synchronization can control the charge pump to increase or decrease the frequency of the VCO to track the incoming data.

The estimated timing error is achieved by using an early-middle-late sampling scheme. The input is correlated with triangle waves corresponding to the optimal position and positions +/- 6.25 ns away. There are two symbol synchronizers, one for each of the possible pulse positions. When the actual pulse position is determined by the bit detection circuit, the valid symbol synchronizer circuit is selected and may adjust the clock.

An extensive digital control section must be implemented in association with the symbol synchronization circuitry to generate the waveforms to control the triangle wave generators, correlators, and clocked comparators. To provide further flexibility, in this implementation these waveforms are adjustable to accommodate varying channel environments and differing UWB symbol rates. Furthermore, these adjustable waveforms can be used to compensate for offsets within the UWB receiver.

To correct the clock by adding and subtracting clock pulses, a digital filtering operation is performed to filter out noise. After the digital filter, a circuit either adds or subtracts pulses from the main 160 MHz clock to keep the synchronization correct. This method of synchronization works when the clock is from a fixed frequency source which can't be changed. However, if there is a frequency error, the symbol synchronization will be forced to continually make corrections to stay in synchronization.

The other method is to adjust the frequency of the VCO within the PLL. When this is selected, the PLL operates as a normal PLL, using the high frequency crystal oscillator as a reference, until a packet is detected. At this point, the charge pump control signals from the normal phase-frequency detector are ignored and instead the up/down control signals are controlled by the symbol synchronization logic. Each time the symbol synchronization logic detects a timing error, a fixed length pulse is applied to either the up or down control signal of the charge pump, therefore changing the VCO frequency. The width of the pulse, and therefore the amount of corrective action, is controlled in software by the Width field in the UWBRX_CEA register. If this is set incorrectly, the system may oscillate, or respond too slowly. The PLLAdj bit in the UWBRX_CEA controls which mode is used. For more details on the PLL and the implementation of this scheme see subsection 7.1.2 on page 181.

5.9.1 Symbol Synchronization Waveforms

The waveforms for the symbol synchronizer and the bit detection logic are adjustable in software. This was done to allow flexibility in testing the receiver to handle various eventualities. The basic architecture to generate the UWB waveforms is shown in Figure 5.37. It consists of a counter and a number of digital comparisons to set and clear a digital value at arbitrary times. The counter and comparators are all five bit, but for high-rate UWB transmissions, the counter can be configured as a 4-bit counter, so the incoming clock frequency only needs to be 16 times the UWB symbol rate as opposed to 32 times. The 5-bit counter mode can be used if the UWB receiver is receiving data at a slow symbol rate (for example if using rate-division multiple access).

Figure 5.37 shows the two basic types of waveform generation circuitry, either a single comparator is used and a single one-clock-cycle pulse is always produced on the output, useful for resetting a comparator or integrator, or two comparators are used which gives independent control over the setting and clearing of the digital waveform. A similar scheme is used with three comparators and a triangle wave generator to produce the three triangle waves for the symbol synchronization circuitry.

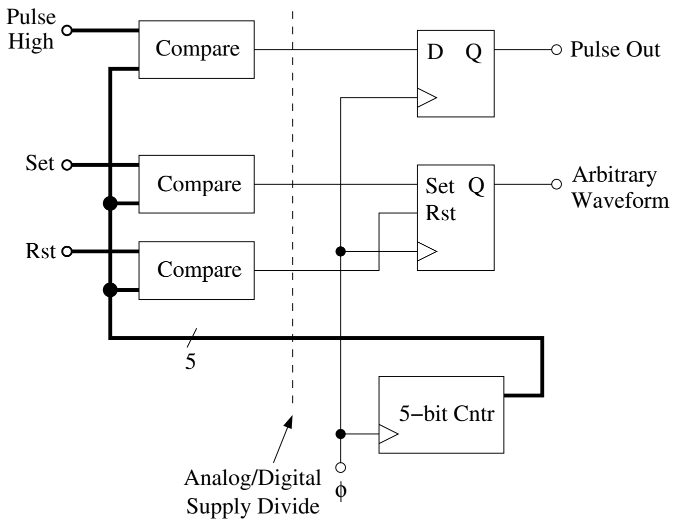


Figure 5.37 – Symbol synchronization waveform generation circuitry

The analog-digital supply divide cuts down the middle of this waveform generation circuitry. This is important because there is a delay as a signal crosses the level-shifter across the supply divide, and furthermore the high-to-low and low-to-high propagation times are highly unequal. Therefore to maintain accurate waveforms for the UWB receiver, all the clocked components (which set the edges for the signals) must stay on the analog side of the divide. The comparators were placed on the digital side of the divide in order to decrease the number of signals which must cross the divide, as this is a limited resource in the receiver because there is only a finite length of analog-digital dividing line in the receiver, and all the level shifters must straddle it.

Figure 5.38 shows the intended waveforms of the UWB receiver. In actuality, the waveforms may be adjusted on a per-chip basis to compensate for offsets within the symbol synchronization circuits. In particular, some mismatch between the early, middle, and late correlators causes problems. Some of this can be compensated for by making the triangle waveforms either bigger or smaller to correct for the correlator offset. This only works to a small extent, and offsets within the symbol synchronization circuitry is the most common reason for the UWB receiver not to work in a particular chip. This is the biggest bug with the receivers and does inhibit the full testing of the receivers as we will see in chapter 10.

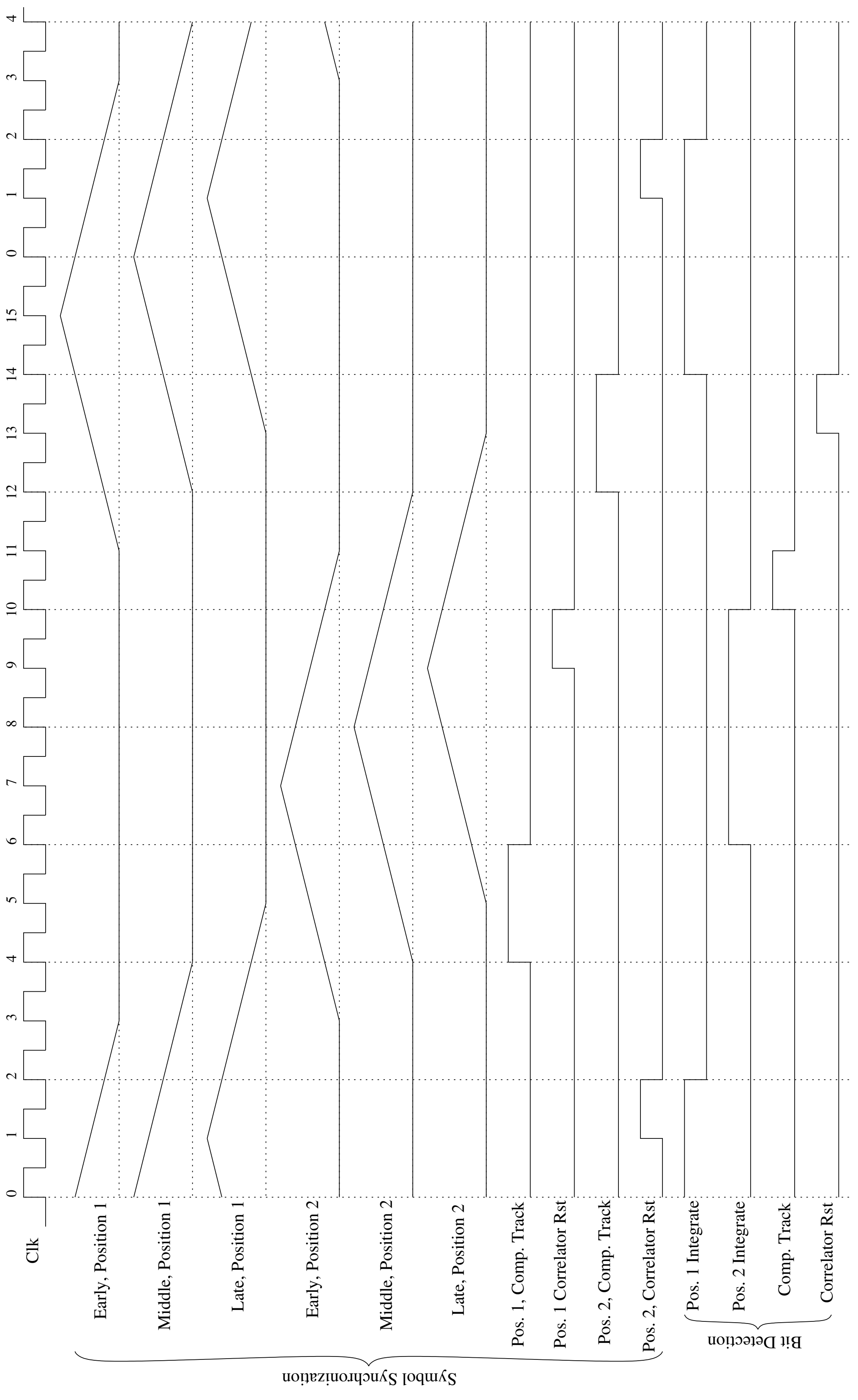


Figure 5.38 – Symbol synchronization and bit detection waveforms

5.10 UWB Antenna

The UWB antenna was modeled after well-known circular disc monopole designs in the literature [86]. The dimensions were up-sized, to work with a lower frequency and the feedline was changed from a more conventional microstrip to a coplanar waveguide structure because that works better with thicker boards. The board is a standard 0.64 inch thick FR-4 material. The dimensions of the antenna is shown in Figure 5.39.

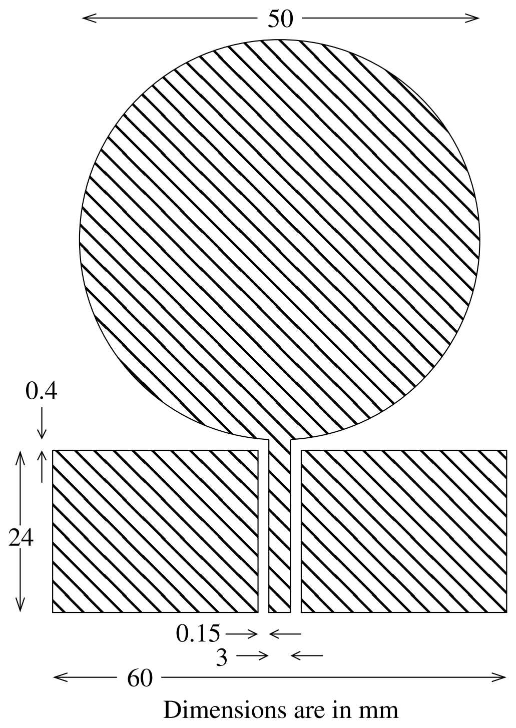


Figure 5.39 – UWB antenna dimensions (not to scale)

The design was simulated in Ansoft's High Frequency Structure Simulator (HFSS) to characterize the radiation pattern and input matching. However, due to licensing problems, the graphs are not available. As seen in the literature, this type of antenna is quite omnidirectional. The input matching, $S_{1,1}$, is below 10 dB from 1 GHz to 6 GHz. The matching on the low-frequency range was sacrificed in order to minimize the size of the antenna.

These antennas were fabricated as standalone antennas as well as versions integrated with the active circuitry on the same board.

5.11 Conclusion

This chapter presents the design of a non-coherent UWB transceiver. The UWB receiver has many novel features such as the heavy reliance on analog domain processing for the baseband to decrease power and the presence of a variable-rate convolutional code and interleaver to make it resistant to poor channel conditions. The RF front end includes novel use of a symmetric inductor as a T-coil load to reap better bandwidth at a lower power consumption while only using one inductor. The receiver has a simulated total power consumption of 20 mW and a silicon area of 0.905 mm^2 for the analog section. The all-digital UWB transmitter is simple and low-power yet effective and configurable.

Chapter 6

Narrowband Transceiver Design

This chapter presents the design of a very simple and low-power narrowband transceiver. The transceiver was designed to have a very low-power receiver to operate on the node-side. The transceiver uses a variant of on-off keying for ease of decoding and transmitting. Figure 6.1 shows how the narrowband transceiver fits with the overall node architecture.

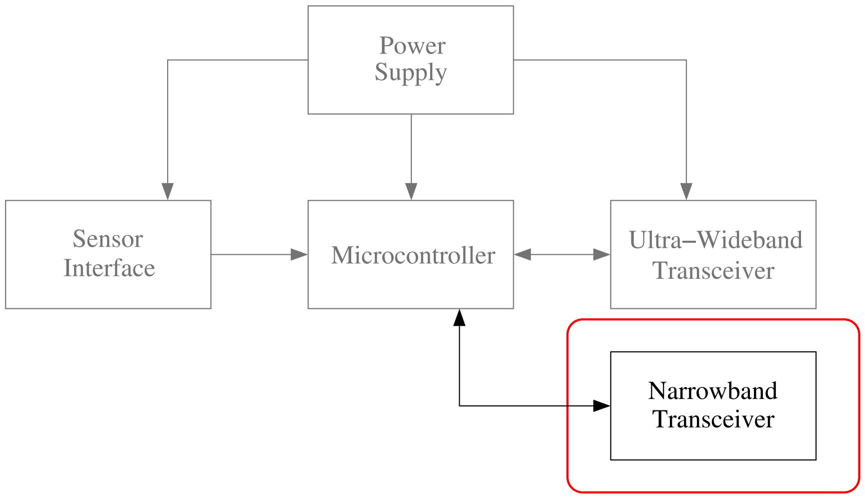


Figure 6.1 – Narrowband transceiver section enclosed in red box.

The rest of this chapter is organized as follows: section 6.1 presents the design motivations and architecture for the transceiver, section 6.2 presents the transmitter design and implementation while section 6.3 presents the receiver design and implementation. Section 6.4 concludes the chapter.

6.1 Architecture

The architecture of the narrowband receiver is much simpler than the UWB transceiver, however the RF front end shares a similar architecture. The design tradeoffs for the narrowband transceiver were focused on simplicity and low-power. It was originally assumed that the wireless sensor network would use a hybrid architecture where UWB was used in one direction with narrowband used in the opposite direction. In this configuration, the narrowband only has to surpass the range of the UWB transceiver. Any further range would not be used as the network loses bidirectional communication capability after the UWB range is exceeded. As the UWB is quite short range, the range of the narrowband transceiver does not need to be exceptional, so a very low-power receiver can be used at the expense of range.

The narrowband transceiver operates at 912 MHz which is in the ISM band. The transmitter is compatible with the integrated narrowband receiver, however does not conform to any standards. It uses a simple manchester-encoded on-off-keying (OOK) data transmission format as shown in Figure 6.2. There is no error control coding on the link, only an 8-bit CRC to check for correct reception. The designed symbol rate is 500 kbps. No shaping of the pulses is performed except for that which is achieved via the high-Q filter on the output of the power amplifier.

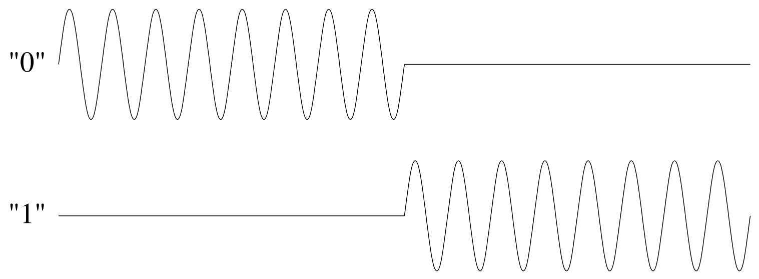


Figure 6.2 – Narrowband Manchester-encoding scheme

For the first version of the narrowband receiver on the second chip, a simple on-off keying receiver was used without manchester encoding. However, this receiver architecture was changed in the third and fourth chips and the new architecture needs a 50% duty cycle. Furthermore this encoding scheme guarantees at least one transition in every symbol period which is useful for symbol synchronization.

The packet format is shown in Figure 6.3. The packet begins with a synchronization preamble consisting of a number of consecutive zeros transmitted (which manchester encoding converts to a series of alternating on and off times), after which a single one is transmitted before the packet begins. The first synchronization sequence needs to be at least 4-6 bits, though more bits can be used by the transmitter. The following “1” indicates the end of the synchronization header. The address and length are followed by the data payload and an 8-bit CRC which is generated and checked in hardware.

Synchronization (4–6 bits, all zeros)	"1"	Address 8 bits	Length 8 bits	Data Payload up to 254 bytes	CRC 8 bits

Figure 6.3 – Narrowband packet format

6.2 Transmitter

The narrowband transmitter architecture is shown in Figure 6.4. It consists of the PLL which it uses as its frequency source, the digital baseband, and the power amplifier. The digital baseband part of the transmitter was present starting with the second chip, however for the second and third chips the RF part had to be implemented on the PCB. For the fourth chip the PLL and power amplifier were also integrated for a fully integrated solution.

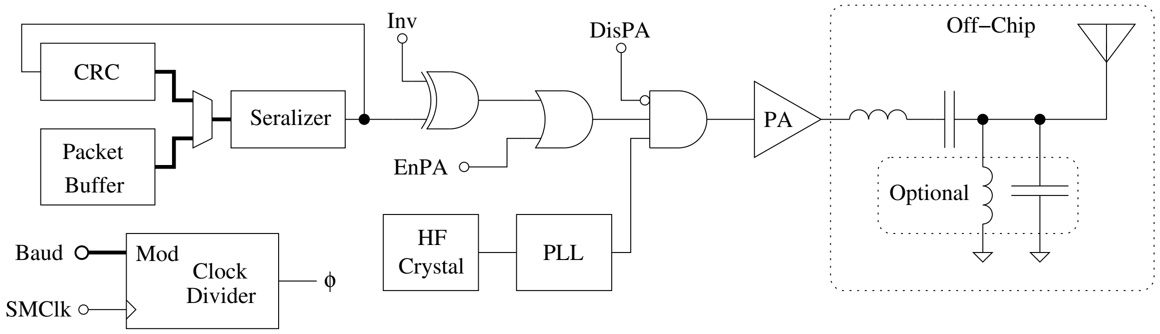


Figure 6.4 – Narrowband transmitter architecture

The following sections will discuss the individual parts of the transmitter, however the frequency synthesizer is discussed in the clock generator section, subsection 7.1.2.

6.2.1 Digital Section

The buffer in the second chip is a dual-port RAM, which has two read and two write ports which allows the processor to read and write the buffer while the UWB receiver reads the data. For the third and fourth chips this was changed to a dual-port register file which has one independent write port and one read port. The write port is associated with the processor and the read port is associated with the UWB transmitter. This means that the processor cannot read data after it has written it to the buffer. This switch was made to save area, but mostly to save power as the dual port register file consumed much less static power (the savings was idiopathic, both were hard IP blocks from ARM).

The serializer digital section is one of the simplest modules in the entire digital portion of the chip. It is responsible for putting the synchronization header on the packet, reading from the data buffer and sending the packet. It also calculates and appends the CRC to the packet. As with all processor peripherals, it can run on a clock which is asynchronous to the processor clock. It has an independent divider to select the symbol rate. Practically, it works best to operate it directly on the 16 MHz crystal clock and divide the clock by 16 to get the 1/2 symbol time base (1 μs) that the transmitter requires.

As with the other transceiver blocks in the chip, the transmitter is double-buffered, meaning there are two buffers and the processor can be filling one while the transmitter is transmitting the other. The transmitter alternates between transmitting the two buffers. This maximizes the bandwidth and flexibility when transmitting. The two buffers are actually implemented as a single 512 byte dual-port register file.

The CRC is the same as the one used on the UWB link and is discussed in section 5.4

6.2.2 Power Amplifier

The PA is just a chain of inverters. This makes a high-efficiency (simulated at 50%) constant-envelope amplifier; however, the output power can be changed by changing the power supply voltage which is the digital core supply. Therefore, the RF output power is set by the DVS voltage setting during transmission. The output power of the PA is ideally set by the power of the fundamental component of the square wave which is:

$$P_{out} = \frac{8 \times \mathrm{VDD}_{\mathrm{core}}^2}{\pi^2 \times R_{\mathrm{out}}} \tag{6.1}$$

where R_{out} is the impedance of the load. This is the ideal case; the actual output power will be much less. With a 1.8 V supply and a 50 Ω load, the maximum ideal instantaneous power is about 50 mW. However, in practice the maximum instantaneous power is closer to 10 mW due to extensive losses in the driving MOSFETS. Furthermore, the average power is half that because of the 50% duty cycle, so an average power of 5 mW can be expected. The output power can be adjusted at run-time by adjusting the VDD_{core}, or can be changed at design time by using a lower load impedance. This could be accomplished with a matching network between the power amplifier and the load.

The two LC tank circuits present in the output are to filter the high frequency harmonics present in the square wave output. The first series LC tank is required to maintain high efficiency in the PA. It presents a low impedance at the center frequency, but blocks the harmonics. The second parallel LC tank is optional, and can be included to further filter the harmonics if necessary. High-Q filters should be used to minimize the bandwidth. The parasitic capacitance to ground at the output of the PA, before the series inductor should be kept to an absolute minimum. Any capacitance here dramatically decreases PA efficiency. Therefore, the series LC tank should be as close as possible to the IC pin to minimize parasitic capacitance. After the first LC tank, the board parasitic capacitance can be absorbed into the C of the second parallel LC tank if the dimensions allow a lumped-element model to be used.

The efficiency is set by losses in the two switches. These take the form of gate drive losses, I^2R losses, and capacitive losses caused by charging and discharging the parasitic capacitance at the output. These three parameters are all affected by the size of the driver transistors, and must be adjusted for optimal efficiency. The efficiency in simulation was 55% with realistic pad capacitance at the output node.

The PA can be disabled or always enabled by setting the appropriate bits in the control register, however this is mainly for testing.

6.3 Receiver

The narrowband receiver shares a similar architecture with the UWB receiver. Its architecture is shown in Figure 6.5.

First is an external surface acoustic wave filter (SAW filter) which is an extremely high Q filter. It has a bandwidth of about 2 MHz and has a very good drop off on either side. It has up to 3 dB of insertion loss, so it very negatively affects the noise figure. Nonetheless, this type of architecture is simple and low-power to implement. The SAW filter does all the channel selection of the entire receiver.

Placing a SAW filter before the LNA to do frequency selection has many advantages and disadvantages. The two large disadvantages are the reduced noise figure and the poor channel selectivity. Interferers which are close in frequency will easily pass through the

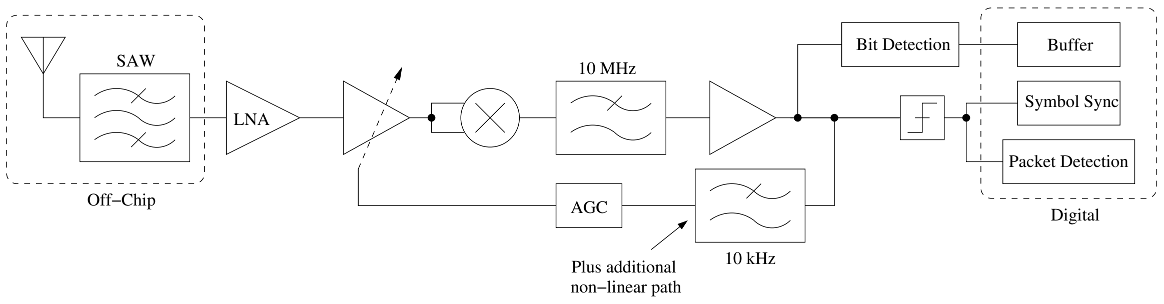


Figure 6.5 – Narrowband receiver architecture

SAW filter and cause the receiver to malfunction. However, using this architecture also makes for a simple and low-power receiver architecture. First there is no need for a frequency synthesizer or an intermediate frequency and the associated power consumption. Secondly, the RF circuits such as the LNA and later gain stages have extremely relaxed linearity requirements because intermodulation is not of concern.

6.3.1 RF Front End

The RF front end architecture is very similar to that of the UWB receiver. It is an energy-detection receiver with an AGC loop to keep the signal from saturating the receiver. It consists of an LNA, 8 gain stages, a self-mixer and the AGC control. Each part is discussed in the following sections. The front end requires the most power of the receiver, consuming a total of 830 μA or 1.5 mW.

6.3.1.1 LNA

The LNA is a stacked narrowband LNA, the schematic of which is shown in Figure 6.6. This is two gain stages stacked on each other so they share the same bias current. The first stage which consists of the devices below the source of M3 is a standard inductively degenerated common-source LNA with narrowband input matching. L1, L2, and C1 are responsible for the input matching with C2 being a DC block. L3 and C4 are a resonant load. C3 is very large and enforces an AC ground at the source of M3 for the upper stage to use. C5 couples the signal from the output of the first stage to the second stage. The second stage also has a resonant load composed of L4 and C6. Altogether, the LNA produces 39 dB of gain with a noise figure of 1.6 dB and a bias current of 380 μA. The impressive noise figure, gain, and power consumption was obtained at the expense of using an external

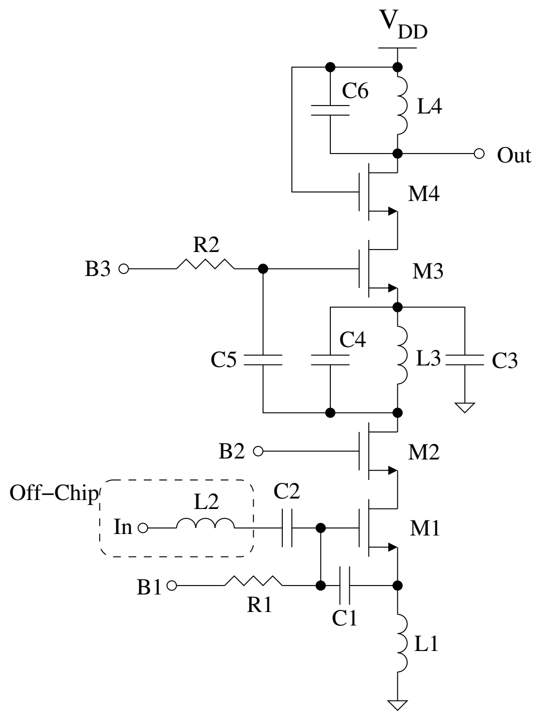


Figure 6.6 – Narrowband LNA schematic

inductor and at the expense of very poor linearity and IIP^3. However linearity and intermodulation is not a concern with this type of receiver. The input matching $S_{1,1}$ was less than -10 dB, however due to the external inductor and poorly modeled PCB and package parasitics, etc., this will shift dramatically once fabricated. Furthermore, it is difficult to measure input matching once fabricated because of the intervening SAW filter.

6.3.1.2 Gain Stages

After the LNA, 8 narrowband gain stages produce up to 6.5 dB of gain per stage for a total of 53 dB. Each gain stage is un-tuned and does not require an inductor due to silicon area constraints. The schematic of each gain stage is shown in Figure 6.7.

The amplifier consists of a common-source cascode stage with a modified active load. The gain can be adjusted by varying the “Bias N” voltage. C1 and R1 form a low-pass filter. The load composed of M3, R2, and C2 form a inductive simulant which increases the bandwidth without the need for a large inductor.

The load used looks like a standard diode-connected transistor at low frequencies. In that case, the capacitor is effectively out of the circuit and the impedance looking into the

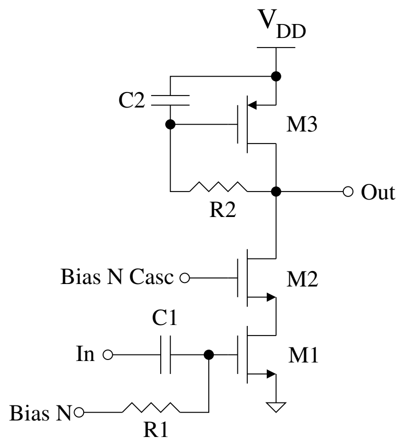


Figure 6.7 – Un-tuned narrowband gain stage

load is about $1/g_m$. At high frequencies above the cutoff frequency formed by R2 and C2, the impedance looking into the circuit is

$$R_o = R2 \,||\, r_{ds} \tag{6.2}$$

where r_{ds} is the output impedance of transistor M3. This impedance is much larger than $1/g_m$ so the impedance of the circuit increases at higher frequencies and thus behaves like an inductor.

This rising impedance at high frequencies introduces a zero in the gain which can greatly extend the bandwidth. For this high-frequency application, the capacitor size is mostly just the parasitic C_{gs} of M3, with an extremely small amount of added capacitance in parallel. This was actually implemented as a parasitic capacitor, not an actual MIM capacitor.

The bias current through each stage is 44 μA for a total power dissipation of all the stages of 634 μW. The total maximum gain from the input of the LNA to the end of the RF stages is 93 dB. This value of gain is much more than needed, as this gain will amplify the noise to a saturating level, however the circuit was designed with extra gain to compensate for any potential gain loss post-fabrication. In the second chip, after fabrication there was less gain than expected and this limited the receiver performance. Therefore, extra gain was added in later chips. The AGC loop will decrease the gain to an acceptable value automatically, and the power consumption of the circuit will naturally decrease, so there are minimal problems with this intentional over-design.

6.3.1.3 Self-Mixer

The self-mixer is a Gilbert multiplier, whose schematic is identical to the one used for the UWB receiver as shown in Figure 5.24 except the component values have changed. It also has exactly the same function and will not be discussed further here. The bias current is 96 μA.

6.3.1.4 AGC

After the self-mixer, another 20 dB of baseband gain is added with a baseband amplifier. This amplifier is very similar to the one for the UWB receiver except the source-follower output buffer stage present for the UWB receiver was not needed. Other than that, the circuit is similar.

The AGC amplifier and loop compensation is similar to that of the UWB receiver with one exception. A pull-down accelerator was added as shown in Figure 6.8. In Figure 6.8, the OTA is similar to the one shown in Figure 5.27 for the UWB receiver, however, this circuit has the addition of a differential comparator which enables a current source to quickly decrease the gain of the receiver.

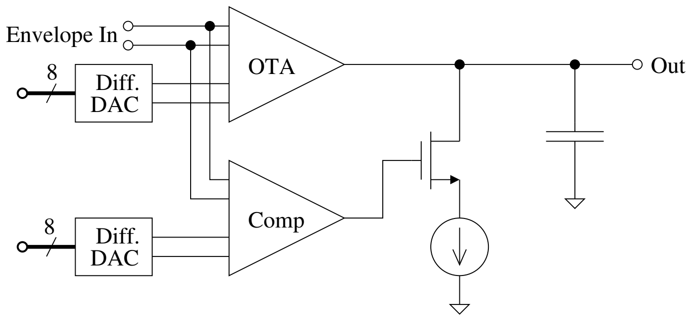


Figure 6.8 – Narrowband AGC amplifier with accelerator

This additional comparator allows the gain of the receiver to quickly decrease during the initial few synchronization bits of the packet. The time constant of the OTA-C integrator must be very slow compared to the bit time otherwise the AGC loop will respond to each bit and cause problems. However, this causes a problem when the receiver begins receiving a strong packet. The strong packet saturates the front end and the AGC loop takes a long time to respond at which point it may be too late to detect the synchronization preamble. Therefore, the comparator provides a very quick way of decreasing the gain in the case of a very large input signal.

The two DACs are adjustable via system software and their setting plays an important role in the correct functioning of the narrowband receiver. The DACs are of the same architecture as those in the UWB receiver shown in Figure 5.26.

The AGC control voltage then goes to a bias generation circuit which produces the bias voltages necessary for the un-tuned gain stages. This circuit is identical in topology to the one used in the UWB receiver shown in Figure 5.28 except four op-amp buffers are used to drive the 8 gain stages.

6.3.2 Analog Baseband

The analog baseband section of the receiver is not as complicated as that for the UWB receiver. The bit detection is done in the analog domain with an integrator and comparator, while the symbol synchronization and packet detection are done in the digital domain with just a comparator needed in the analog domain.

A schematic of the analog baseband is shown in Figure 6.9. The bit detection circuitry consists of the template generator and the mixer which correlate the envelope signal to the expected bit waveform. The output of the multiplier is left as a current which then integrates on the capacitors labeled “C” and a comparator determines the received bit. After the comparator settles, the charge on the integrating capacitors is shared with the much large averaging capacitors labeled 15C to get a measure of the signal strength.

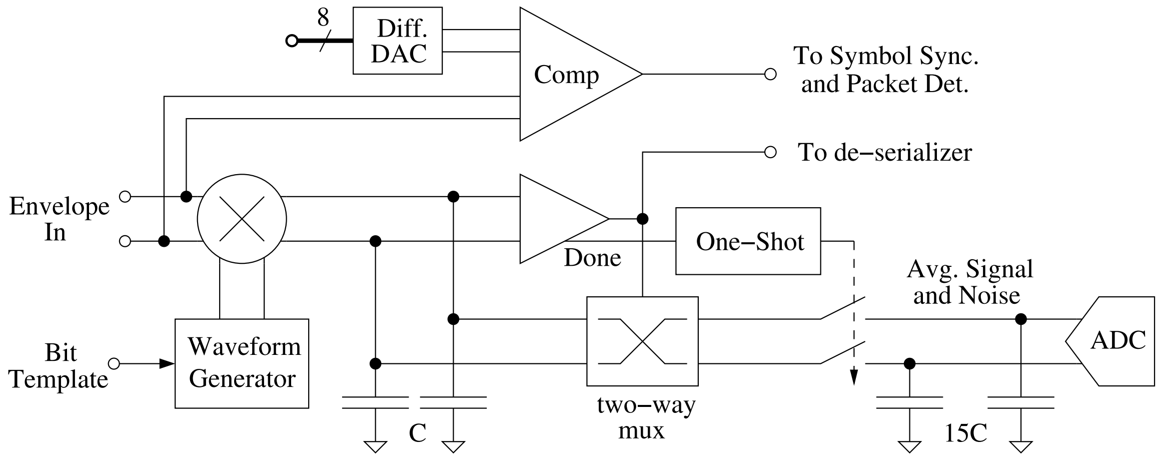


Figure 6.9 – Narrowband baseband circuits.

The differential difference comparator sets a threshold for the envelope and the information of when the envelope surpasses the reference is used in the digital domain for packet detection and symbol synchronization.

The waveform generator accepts a digital signal from the digital section and produces a small (200 mV) differential waveform to feed into the multiplier. The multiplier is a standard Gilbert multiplier, with source degeneration on the envelope inputs. The correlation inputs are square waves and are designed to switch the current, so it is not used as a conventional multiplier but instead driven to non-linearity on the correlation waveform inputs. The output currents are mirrored around to differential outputs which integrate on the output capacitors. The comparator used to determine the bit is the exact same comparator used for the SAR-ADC which was discussed in subsection 3.3.1.

The ADC which is used to sample the average signal and noise values is a copy of the 8-bit SAR-ADC used in the front end.

6.3.3 Digital Baseband

The digital baseband is responsible for finding the preamble synchronization sequence, performing symbol synchronization, and general control. The baseband is clocked at 32 times the baud rate. This allows the symbol synchronization to be performed by making a symbol period with either 31 or 33 clock cycles in it to correct for drift.

The diagram of the digital section is shown in Figure 6.10. There are two distinct parts, one part is simply responsible for de-serialization of the data from the bit detection circuitry and computing the CRC, and the other is responsible for symbol synchronization and packet detection.

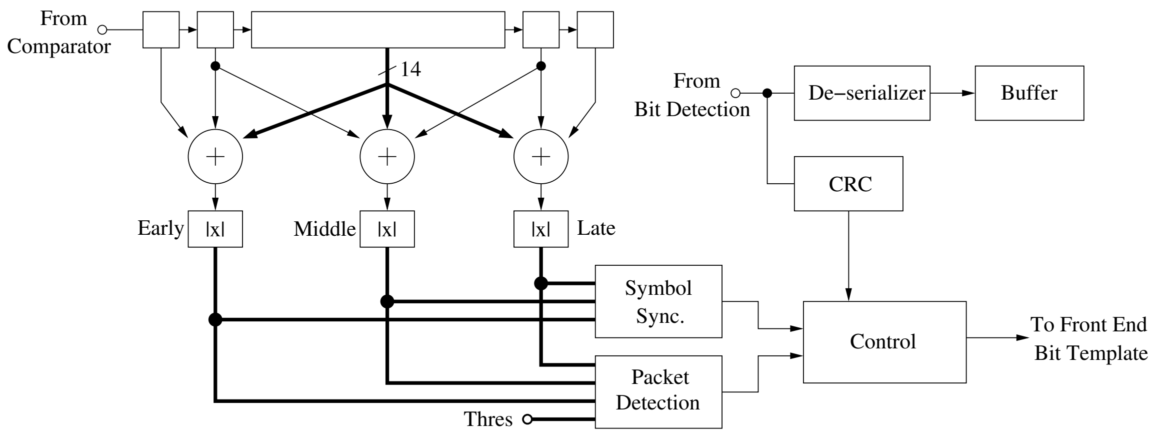


Figure 6.10 – Narrowband digital architecture.

The symbol synchronization and packet detection circuits share a correlator type architecture and early-middle-late sampling stages. For packet detection, the logic waits until it sees a peak in the correlation function as determined by when the middle value is greater than the early value and greater than or equal to the late value, and when that peak exceeds a software settable threshold. After one such event is detected, it next determines that the next n bits also have a correlation value over the threshold so as to filter out noise. The n value is also software selectable. After the detection circuitry has determined a strong packet preamble exists, it then waits for a '1' bit to initiate packet reception.

The symbol synchronization part of the circuit checks the relative magnitudes of the early, middle, and late values at the end of each bit to determine if the next symbol time should be 31, 32, or 33 cycles in length.

For an unknown reason, the hardware CRC does not work in the final chip. The packet is received correctly, but the CRC does not check most of the time. The fix is to make the receiver not send or check the CRC with hardware and instead use the 16-bit general-purpose CRC module to generate the CRC and check it. This requires some extra software, though only a minimal amount of power consumption as the CRC is still generated in hardware, just not in the transceiver itself.

6.4 Conclusion

This chapter presents the design of a very simple narrowband transceiver. The transceiver operates in the 912 MHz ISM band and the receiver only requires 2 mW of power with a receiver silicon area of 0.345 mm^2 for the analog section.

Chapter 7

Supporting Circuitry

This chapter presents a number of parts of the design which play a supporting role in the wireless sensor network node. An important part of the design is the clock generator which is discussed in section 7.1. The clock generator contains a multitude of different oscillators and frequency synthesizers which are used by different portions of the chip. A flexible clocking system is critical to the low-power consumption of the complete system. Section 7.2 discusses the design and implementation of the high-efficiency buck power supplies which are critical to low power operation, particularly of the digital section. The later sections discuss some other aspects of the design such as a $\Delta\Sigma$ ADC in section 7.3, a set of programmable electronic fuses in section 7.4, and the biases used by the rest of the chip in section 7.5. Finally, section 7.6 concludes the chapter.

7.1 Clock Generator

The clock generator circuitry is responsible for generating clocks and distributing them to the modules which require them. The individual properties of the clocks themselves are critical to the proper operation of the rest of the chip. A total of 6 independent oscillators can produce clocks with varying properties, from the high-frequency low-jitter RF clock, to the extremely low-power 32 kHz crystal oscillator used for timekeeping functions.

The clock generator supplies three main clocks for use by the internal modules. They are the main processor clock (MCLK), the sub-system main clock (SMCLK), and the UWB receiver clock. The SMCLK can be sourced from one of 6 internal clock sources, be supplied externally, or be disabled. The MCLK can be sourced from one of 4 clock sources, including being sourced from the SMCLK, or disabled. The processor uses the MCLK divided by an optional prescale, and most peripherals use the SMCLK. Each peripheral has an independent clock prescaler to use a slower clock if desired.

A few peripherals do not use the SMCLK. They are the UWB receiver and the timer modules. The UWB receiver requires a 160 MHz clock which can be supplied from one of two sources or an external clock. The clock can be disabled to save power when not in use. This clock may glitch when changing from one clock source to another. Each timer can independently use either the MCLK, SMCLK, or either of the two crystal oscillators. These can also glitch when changing clock sources.

A diagram of the clock system is shown in Figure 7.1. It shows the numerous clock sources available within the chip. This includes the PLL, two crystal oscillators, and three internal free-running oscillators.

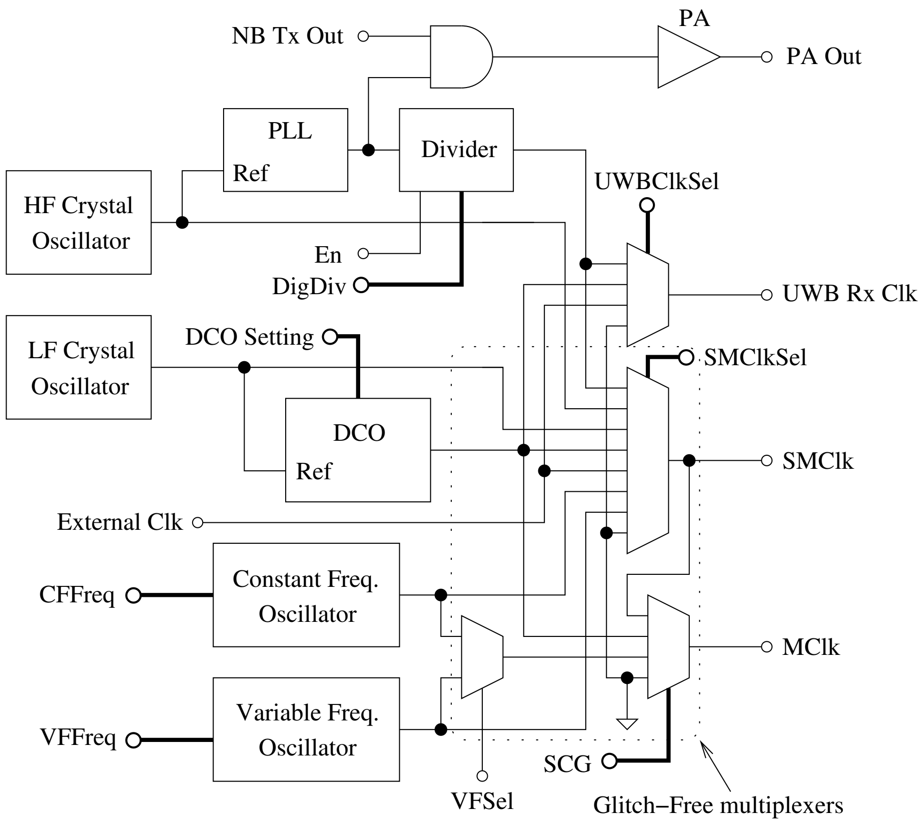


Figure 7.1 – Overall clock system

The PLL operates with the high-frequency crystal as the reference clock. The output of the PLL can be modulated by the output of the narrowband transmitter module and driven to the power amplifier. It can also be divided by a small value to be used as a general purpose, low-jitter clock, mainly for RF transmission and reception. The low-frequency crystal oscillator is the reference for the digitally stabilized digitally controlled oscillator (DCO). The DCO starts quickly and can be used as a low-power clock when a fixed-frequency clock is necessary. However, the jitter makes it unacceptable for RF transmission and reception. The other two oscillators, the constant frequency oscillator and variable frequency oscillator, are unique in that they operate on the digital supply, while all the other oscillators operate on the analog supply. This means these two clocks can be used before the analog supply is started.

The clock multiplexers inside the dotted box are the only multiplexers in the entire chip which are guaranteed glitch-free. All other clock multiplexers in the design, including ones not shown in the figure, such as the ones in the timer modules, may have glitches on the output when the clocks are changed. The glitch-free multiplexers require much circuitry and cannot easily be synthesized within the digital section. This is because they interfere with clock tree synthesis. Therefore, they were not synthesized but were drawn manually. When a clock change is required, the multiplexer first waits for the presently selected clock to go low. After disabling the present clock, it waits for a full cycle of the new clock. Then it switches over to the new clock when the new clock line is also low. This procedure guarantees there can be no glitches on the clock when switching between clocks.

7.1.1 Crystal Oscillators

There are two crystal oscillators in the design. One is a low-frequency low-power one for timekeeping functions, and the other is a high-frequency low-jitter one for RF transmission. They both operate on the same principle, though due to their vastly different frequencies, their implementation is different.

7.1.1.1 Theory

A crystal can be modeled as shown in Figure 7.2. It consists of a capacitor C_{12} which is the physical capacitance between the two terminals, and a number of series RLC resonant circuits. Generally crystals are designed to operate at either the fundamental, or for some high frequency crystals, the third overtone. Generally only one resonant mode is desired, and external circuits may be necessary to force the crystal to oscillate in the desired mode. Each RLC circuit has an extremely large Q (on the order of 10,000) and a large L/C ratio. The resistance models the mechanical losses.

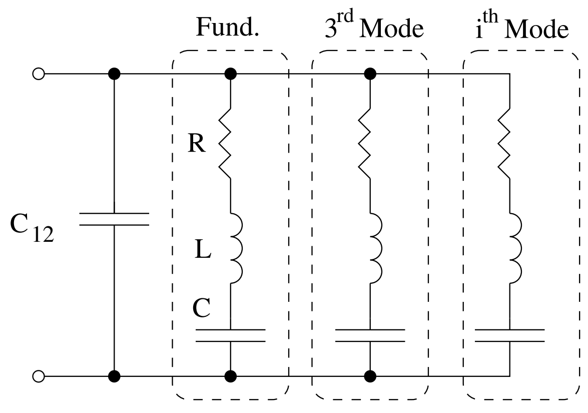


Figure 7.2 – Crystal model

From [87], a crystal oscillator circuit can be analyzed as a three point oscillator. A basic three point crystal oscillator is shown in Figure 7.3 while a generalized one is shown in Figure 7.4. In Figure 7.4 the crystal has been modeled as a series RLC circuit. This assumes the other crystal modes are not going to play a role in the circuit which may or may not be the case.

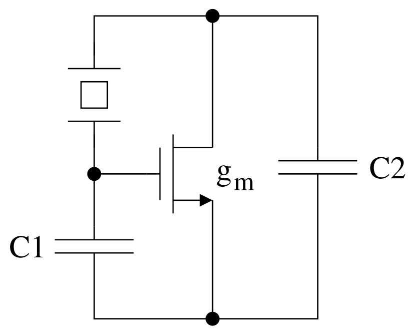


Figure 7.3 – Basic three point oscillator

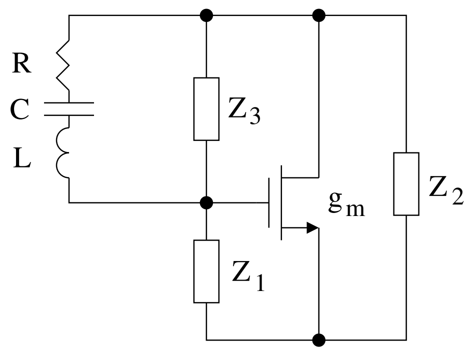


Figure 7.4 – General three point oscillator

From analysis of this circuit [87], it can be shown that the there is a minimum g_m required for the circuit to sustain oscillations given by

$$g_{m\text{crit}} = \frac{\omega}{QC}\frac{(C_1C_2+C_2C_3+C_3C_1)^2}{C_1C_2} \tag{7.1}$$

where Q is the Q of the RLC tank and C_3 is the capacitance intrinsic across the terminals of the crystal (Z_3 in Figure 7.4). At g_m values above the $g_{m\text{crit}}$, the amplitude of oscillations increase exponentially. This is required during start-up. However, at some large

g_m, the oscillator will not oscillate. An optimal g_m therefore exists where the amplitude of oscillations increases most quickly and is desired during startup. This g_m is given by

$$g_{m\text{opt}} = \omega \left(C_1 + C_2 + \frac{C_1 C_2}{C_3} \right). \tag{7.2}$$

However, the rate of amplitude increase does not change very much for small variations around $g_{m\text{opt}}$, so it is only important to get close to this value.

The crystal has a mechanical resonant frequency ω_m, however it is operated at frequency ω which is close to ω_m but not the same. The frequency pulling is defined as

$$p = \frac{\omega - \omega_m}{\omega_m} \ll 1 \tag{7.3}$$

where p is the relative amount of frequency pulling. This can also be defined in terms of the circuit parameters in Figure 7.4 by

$$p = \frac{C}{2\left(C_3 + \frac{C_1 C_2}{C_1 + C_2} \right)}. \tag{7.4}$$

When the g_m value is above $g_{m\text{crit}}$, the amplitude of oscillations increase until non-linearities generate harmonics in the output current. At this point the g_m must be replaced with the g_m for the fundamental frequency which decreases as the non-linearities increase, eventually going to $g_{m\text{crit}}$ and setting the oscillation at a constant value. However, these large harmonic components are not desirable for frequency stability. Harmonics in i_D create harmonics in the voltage across Z_2. The harmonics current flows through Z_3 which distorts the driving voltage at the gate. This distorted gate drive is intermodulated with device non-linearities to create an output current component at a different phase, pulling the frequency. Therefore, we would like to minimize the harmonic distortion. Also, it is advantageous in terms of power to use a smaller g_m when possible.

7.1.1.2 Overview

The chip integrates two crystal oscillators. One, a low-frequency low-power 32 kHz crystal oscillator and the other a low jitter high-frequency crystal oscillator. The main difference between them is the current each requires. The 32 kHz crystal is designed as an always-on timebase for timekeeping operations, while the high frequency oscillator is designed to be enabled when required and disabled at the other times. It is required as the reference frequency to the high-frequency PLL used for the RF transceivers.

Both crystal oscillator circuits are designed to operate well with a wide variety of external crystals, with no external components other than the crystal itself and possibly external load capacitors. They are designed to use a g_m close to $g_{m\text{opt}}$ during startup to minimize startup time, but once when the oscillations are started they decrease the g_m to $g_{m\text{crit}}$ to keep the amplitude of the oscillations small and harmonics to a minimum. The gain limiting has a number of desirable effects. It lowers the power consumption of the circuit, so the oscillator only requires the minimum amount of current needed to keep the g_m of the amplifier large enough to sustain oscillations. This is significant, as the oscillators will generally take 10 times the current when starting up as is needed to maintain oscillations. It also keeps all the circuits operating in a linear regime, which limits the third-order harmonics. This in turn stabilizes the crystal oscillator frequency by reducing frequency pulling which can occur with large amplitude oscillations. A block diagram of the crystal oscillators is shown in Figure 7.5.

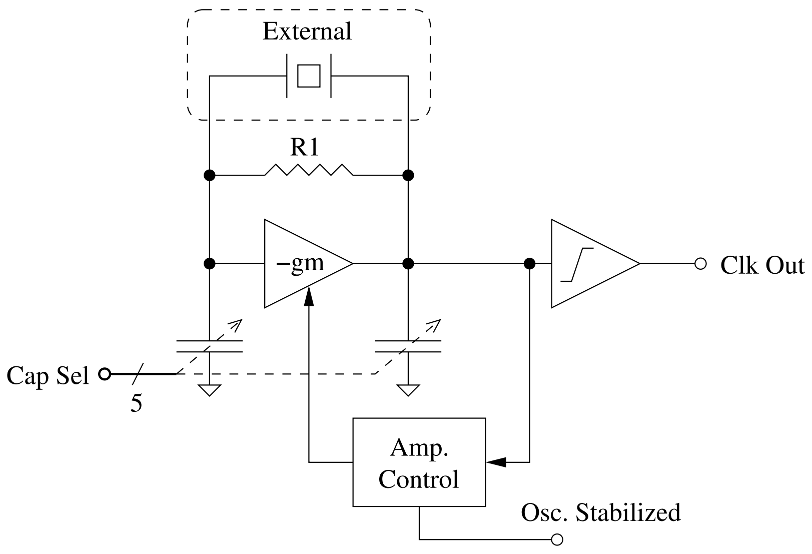


Figure 7.5 – Crystal oscillator

The figure shows the gain stage which is controlled by the oscillation amplitude. A set of digitally adjustable load capacitors are integrated on chip. The resistor R1 is a biasing resistor.

The crystal oscillators require time to start—on the order of 1000s of cycles. The oscillator architecture shown in Figure 7.5 starts more quickly than others for two reasons.

First, the increased g_m at the start improves the startup time, and second, the limiting amplifier has very high gain and only requires oscillations of 10 mV to produce reliable digital output waveforms.

Both crystal oscillators feature on-chip adjustable load capacitors can vary between 0.625 pF and 20 pF. These are present mainly to allow the processor to slightly pull the frequency of the crystal to attempt to correct for clock mismatches between this node and its neighbors. For example, the UWB receiver will report how bad and in which direction the clock synchronization error is between the transmitter and receiver. With this information, the processor could adjust the load capacitors, and as shown in Equation 7.4, the frequency will shift slightly. Therefore the receiver will be better matched to the transmitter for the next reception.

The analog waveform associated with the Xin side of the crystal can be viewed via the low frequency test port. This is better than directly connecting an oscilloscope to the external pins because the capacitance associated with the probe will severely affect the oscillations. Some other internal signals are also available on the low frequency test port. This includes a analog value which starts out high and drops as the g_m is decreased.

7.1.1.3 Low-Frequency Oscillator Implementation

For the low-frequency oscillator, the resistor R1 in Figure 7.5 is not implemented as an actual resistor as the value would need to be in the 10-100 GΩ range. instead it is implemented as a simple op-amp circuit with very low bias currents. The oscillator core is implemented as a single n-channel transistor. The schematic of the circuit is shown in Figure 7.6. The core of the oscillator which provides the g_m is transistor M1.

An average 32 kHz crystal will use C_1 and C_2 values of 20 pF. The parameters of the crystal itself from Figure 7.2 on page 174 can be $C_{12} = 1.6\,\text{pF}$, $R = 40\,\text{k}\Omega$, $L = 7.4\,\text{kH}$, and $C = 3.3\,\text{fF}$. This gives a Q of 40,000. From Equation 7.1, the required $g_{m\text{crit}}$ is only 840 nS. The optimum g_m during startup is 60 μS. This g_m value is easily achievable with low currents. During startup, the presented circuit has a g_m of 61 μS and requires a bias current of 2.6 μA. The steady-state g_m required can be achieved with only 35 nA of bias current through M1. The op-amps OP1 and OP3 are the aforementioned op-amps with very small bias currents, on the order of 5 nA. These act as extremely large resistors used for DC biasing M1 and M3. OP2 is the output amplifier which produces the digital output waveform from the small oscillations.

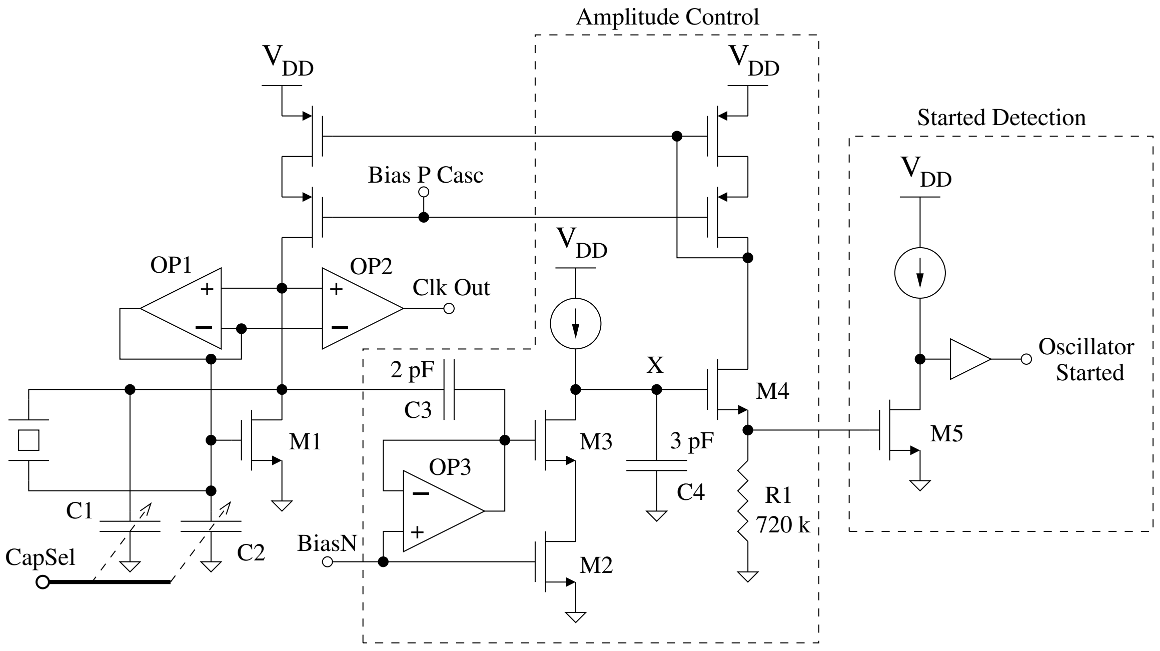


Figure 7.6 – Low-frequency crystal oscillator schematic

The amplitude limitation circuit works as follows. When there are no oscillations, the voltage at point 'X' goes to V_{dd}. This forces the current through R1 to be

$$I_{R1} = \frac{V_x - V_{gs_4}}{R1}. \tag{7.5}$$

Using an extreme oversimplification and modeling V_{gs_4} as the transistors V_t which is a constant (also a simplification because we are ignoring the body effect), the bias current I_{R1} is proportional to the voltage at V_x. This current is multiplied by a mirroring gain before being applied to the input transistor M1. With V_x at V_{dd}, about 3 μA of current flows through M1. As the oscillations build up, the voltage at the gate of M3 begins to turn on M3 for some portion of each cycle. This pulls the voltage at node X down, which in turn decreases the current through the input transistor until the amplitude of oscillations stabilize at about 300 mV P-to-P.

Transistor M5 uses the voltage at the top of the resistor to sense when the amplitude is being limited. At startup, the voltage on the gate of M5 is large, turning on M5 and pulling the output low. When the current decreases, M5 is shut off and the output goes high. This signal can be used by the microcontroller to determine when the crystal oscillator has started.

A cascode current source was chosen to drive the input transistor because the high output impedance decreases losses in the oscillator circuit and helps frequency stability.

A feedback loop exists in the amplitude control circuit which may become unstable if not compensated properly. The compensation is achieved by capacitor C4 interacting with the output impedance of transistors M2 and M3. It is not possible to do AC analysis on this feedback loop in an open-loop configuration and determine the exact properties of the loop because the variable being controlled is not a simple voltage but the amplitude of oscillations. Furthermore, the driving transistors M2 and M3 produce a non-linear transfer function. Therefore, compensation was largely done by trial and error to find a configuration where the loop is far from being unstable. Transistors M2 and M3 are very long channel devices.

There is another high-frequency positive-feedback path which can cause the loop to become unstable within one crystal oscillator cycle. To illustrate this, imagine the gate of M3 is pulled high and begins conduct. This will pull the voltage at node ‘X’ down, which will decrease the bias current. This decrease in bias current will raise the voltage at the drain of M1, which thereby further increases the voltage at the gate of M3. This can cause a runaway positive feedback loop. To fix this, transistor M2 is added to limit the speed with which M3 can decrease the voltage at node ‘X.’ An alternative would have been to connect the amplitude feedback point to the gate of M1 instead of the drain, however connecting it to the drain produces a much desired loop stabilizing zero without which is it very difficult to maintain stability.

The low-frequency crystal oscillator is not present in the first or second chips, and in the third the transistors M2 and M3 were replaced by a single high-V_t transistor. This worked in simulation, and sometimes worked in the fabricated chip, however the main problem was the BiasN voltage which was generated referenced to a regular V_t transistor did not track with the high-V_t transistor causing problems in many chips. For this reason, this oscillator only works in about 50% of the fabricated third chips. The two-transistor regular V_t solution was added for the fourth chip and is much more robust to PVT variations. No PVT problems were found with this oscillator on the fourth chips. Omitted for clarity, extra circuitry is also present to disable the oscillator when not in use.

For the low-frequency oscillator, about 20-50 ms after oscillator is enabled, the oscillations will be large enough for digital waveforms to be produced. After 8 cycles, a counter can alert the microcontroller of this fact. At this point the oscillator amplitude has not stabilized, so the frequency has also not stabilized. Later, after about 60-100 ms, the amplitude will have grown enough that the amplitude control will begin decreasing g_m. When it does so, it also may alert the microcontroller of this fact via the oscillator stabilized digital output. After the oscillator stabilized digital output is asserted, it is still about another 10-20 ms before the steady state is actually achieved.

7.1.1.4 High-Frequency Oscillator Implementation

For the high-frequency crystal oscillator, different design challenges produce a different circuit. The most important change is the much larger g_m required for oscillations. To achieve this with low power consumption, both n-type and p-type MOSFETs were used in parallel so that the g_m of each would add. Also, because of the much higher frequency, the RC time constant associated with R1 in Figure 7.5 does not need to be as large, so a standard on-chip resistor can be used with on-chip capacitors instead of the op-amp used for the low-frequency case. Figure 7.7 shows a schematic of the high-frequency crystal oscillator.

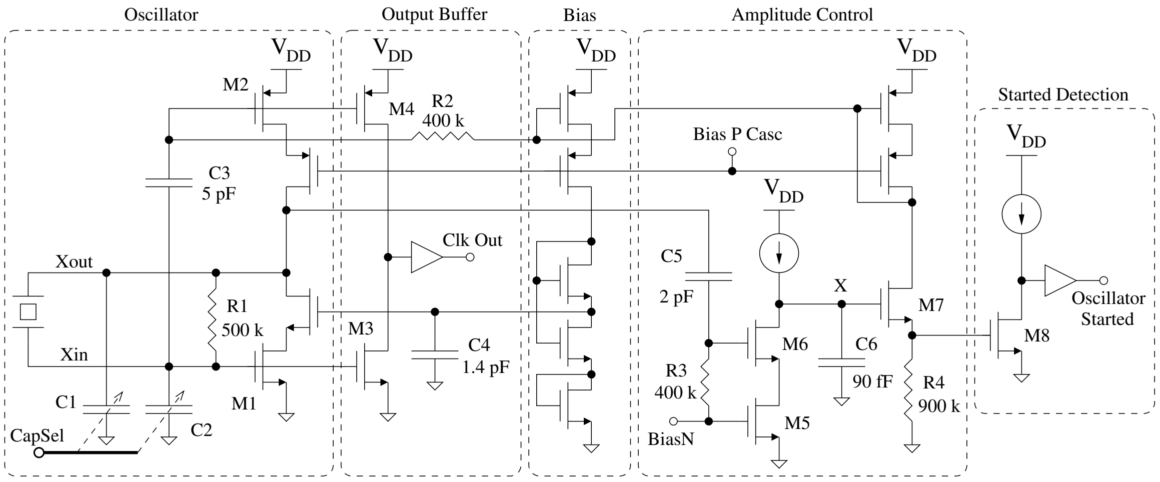


Figure 7.7 – High-frequency crystal oscillator schematic

An average 16 MHz crystal may use values of 40 pF for C_1 and C_2. The parameters of the crystal itself from Figure 7.2 on page 174 can be $C_{12} = 7\,\text{pF}$, $R = 40\,\Omega$, $L = 24\,\text{mH}$, and $C = 4.14\,\text{fF}$. This gives a Q of 60,000. From Equation 7.1, the required $g_{m\text{crit}}$ is 1 mS. The optimum g_m during startup is 31 mS. The challenge is achieving the desired g_m with the minimum power consumption, therefore the parallel n-channel and p-channel MOSFETs improves this. The optimum g_m is too large to easily achieve, and the presented circuit only achieves a maximum g_m of 6 mS requiring a current of 190 μA at startup. At the steady state value of $g_m = 1\,\text{mS}$, the circuit requires 23 μA. The power consumption and start-up time could be improved by using smaller C_1 and C_2. This has the effect of shifting the frequency somewhat (though if all nodes do it, this has no effect on relative timing) and decreasing frequency stability.

Transistors M1 and M2 provide the g_m for the circuit. They are both cascoded to increase the output impedance because higher output impedance reduces losses in the circuit. The bias circuit generates the cascode bias voltage for the n-channel FET. The output buffer utilizes the fact that the p-gate and n-gate waveforms are available to produce a digital output waveform. The amplitude control circuit and started detection circuit works almost identically to the low-frequency version.

The high-frequency oscillator is not present in any of the first three chips and was added for the final fourth chip. Omitted for clarity, extra circuitry is also present to disable the oscillator when not in use. This crystal oscillator also has the ability to connect the amplitude control node 'X' to an external signal. This can be used to trade-off power consumption for jitter performance. Setting this permanently to a higher value increases power consumption but decreases jitter and vice-versa.

7.1.2 PLL

A phase-locked-loop (PLL) is present to act as a frequency source for the narrowband transmitter and as a low-jitter clock source for the UWB transceiver. The PLL uses the high-frequency crystal as its reference.

7.1.2.1 Theory

A PLL is highly non-linear, however for convenience of analysis it can be approximated as a linear system for small phase disturbances around the locking point. Large signal behavior, for example when locking, is difficult to describe analytically. The linearized PLL is shown in Figure 7.8 and consists of a VCO which is modeled as an integrator, a summing junction to find the error signal, and a transfer function $H(s)$.

If the transfer function is a simple scalar gain, it is said to be a first order loop due to the single pole created by the VCO/integrator. It can be shown [83] that this configuration has two disadvantages. First, the bandwidth of the loop and the gain are linked, and there is a steady-state phase error. For these two reasons, a higher-order loop filter is generally used.

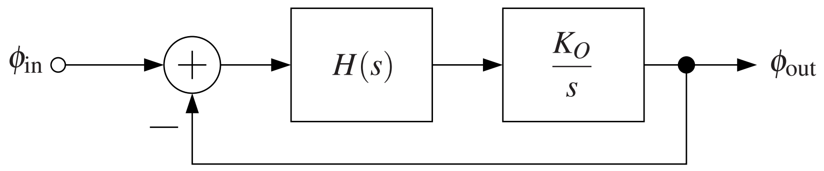


Figure 7.8 – Linearized PLL model.

Figure 7.9 shows the model of a second-order PLL. In this case the transfer function has an integrator, which forces the steady-state phase error to zero, and a loop-stabilizing zero at $1/\tau_z$. K_D is an arbitrary loop gain, and K_O is the VCO constant. Also shown is a model of the noise present on the VCO control line.

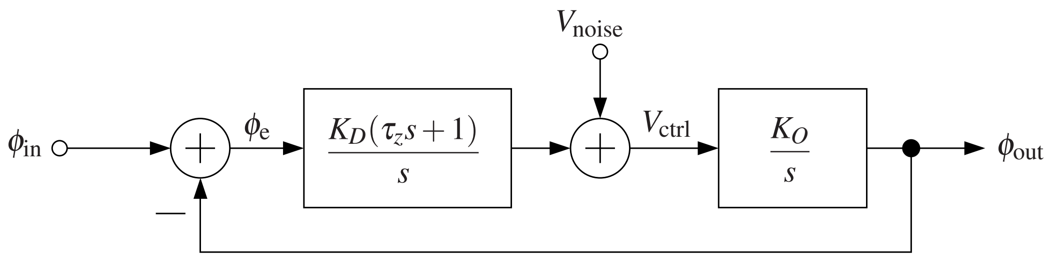


Figure 7.9 – Second-order linearized PLL model with noise input.

From [83], the closed loop phase transfer function is

$$\frac{\phi_{\text{out}}}{\phi_{\text{in}}} = \frac{\tau_z s + 1}{\frac{s^2}{K_D K_O} + \tau_z s + 1}, \tag{7.6}$$

from which we define the natural frequency as

$$\omega_n = \sqrt{K_D K_O} \tag{7.7}$$

and the damping ratio

$$\zeta = \frac{\omega_n \tau_z}{2}. \tag{7.8}$$

If the crossover frequency is well above the zero frequency as is usual, the crossover frequency is approximately

$$\omega_c \approx \frac{\omega_n^2}{\omega_z}. \tag{7.9}$$

PLLs are very sensitive to noise coupled to them by other circuits. The noise coupled into the PLL is modeled as additive noise on the control voltage line. It can be shown that the noise to phase error transfer function for the system is

$$-\frac{\phi_e}{V_{\text{noise}}} = \frac{sK_O}{s^2 + s\tau_z K_D K_O + K_D K_O}. \tag{7.10}$$

If we assume a step-function noise input, it is possible to calculate exactly the maximum phase error and time at which it occurs [83], however the rather large equations provide little design insight. However, in the limit of high damping, they simplify to:

$$-\phi_{e,\max} \approx \frac{\Delta\omega_i}{\omega_c} \tag{7.11}$$

$$t_{\max} \approx \frac{2\ln 2\zeta}{\omega_c} \tag{7.12}$$

where $\Delta\omega_i$ is the initial frequency error due to the step function input on V_{noise}, or

$$\Delta\omega_i = K_O \Delta V_{\text{noise}}. \tag{7.13}$$

So, to decrease the phase noise, also known as jitter, we need to decrease $\Delta\omega_i$ and increase ω_c. Because this is a sampled data system, i.e. the phase error measurements are only available once in each cycle of the reference clock, the ω_c must be kept to a small fraction of the reference clock frequency, typically $f_{\text{ref}}/10$. If ω_c becomes to large relative to the reference clock, the delay causes a phase shift, de-stabilizing the loop. Therefore, ω_c is fixed and the designer has little control over it.

Another way to decrease noise is to decrease the VCO constant K_O which decreases the initial noise injected. K_O may appear to be out of the designer's control at first because VCO has some frequency range it needs to operate over, and the control voltage has some limited voltage range which then sets K_O. However, decreasing K_O can be accomplished by splitting the full range of the VCO into many smaller ranges and digitally selecting the proper range for the VCO at any time. This makes the K_O term, in units of MHz/V, become much smaller. In this design, the operating frequency range of 800 MHz to 920 MHz was split into 16 ranges to help the noise properties. The tradeoff is that the digital control (the processor in this case) must have some method of finding the correct range.

To decrease the noise injection on the VCO control line, it was decided to use a fully integrated loop filter. On many general-purpose frequency synthesizer designs, the loop filter is external to the chip which gives flexibility in the design of the loop filter for different bandwidths and applications. However, any ground bounce between the on-chip grounds and off-chip grounds appears as noise to the VCO, causing phase noise. For this reason, the VCO is generally operated on its own supply to decrease the noise coupled in from other circuits. However, on this chip an integrated loop filter was used which does not have problems with ground bounce. Another benefit is the much smaller capacitors used, which allows a smaller charge pump current and lower power consumption.

7.1.2.2 PLL Implementation

The PLL has a number of functions dealing with RF transmission and reception. It supplies a low-jitter clock which should be used when transmitting or receiving any data wirelessly. Furthermore, it supplies the RF frequency for the narrowband transmitter. Finally, it is integrated with the data synchronization scheme for the UWB receiver to allow more effective synchronization when receiving UWB data.

The PLL is an integer-N synthesizer, using the high-frequency crystal oscillator as its reference. Figure 7.10 shows the complete PLL along with some supporting circuitry. The figure includes the block which, in conjunction with the UWB receiver, creates the pulses to adjust the clock frequency when receiving packets; the charge pump; loop filter; and an analog circuit to detect when the PLL is locked.

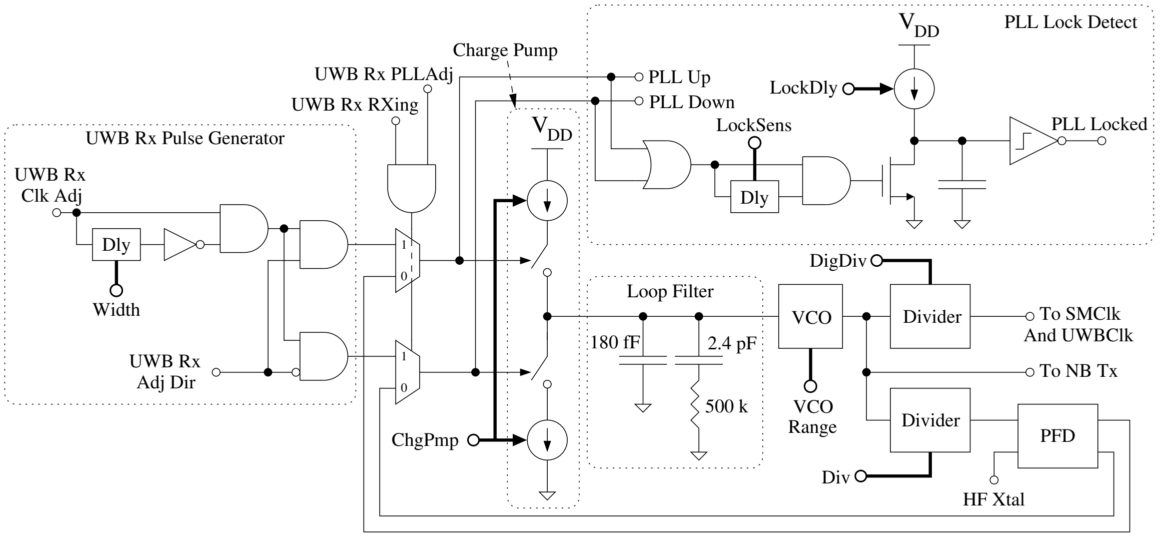


Figure 7.10 – PLL and supporting circuitry

The output of the VCO can be used directly by the narrowband transmitter as the RF source. It is also divided by two separate dividers. One divides the output in a way to maintain 50% duty cycle so that clock can be used as the UWB receiver clock or as the

SMClk. A second divider is in the PLL feedback loop and sets the VCO frequency. The VCO frequency is set by $F_{\text{out}} = F_{\text{ref}} \times \text{Div}$. This divider value can be up to 63.

After the divider a phase-frequency detector (PFD) compares the reference clock to the divided VCO clock to produce the pump-up and pump-down control signals. The PFD has an anti-backlash pulse width of about 1 ns, so if the reference and the divided VCO clock are exactly in phase, both the pump-up and the pump-down lines will pulse high for 1 ns. This should not affect the control voltage, but instead linearizes the transfer function around zero phase error. If the UWB receiver clock synchronization is not enabled, these signals control the charge pump to adjust the VCO control voltage. The charge pump current is adjustable which controls the loop dynamics. A lock detect circuit senses when the PLL is locked.

The PLL up and PLL down control signals are available to be driven to an external pad if desired.

The PLL is only present in the fourth chip. It is not present in any of the previous chips.

7.1.2.3 VCO

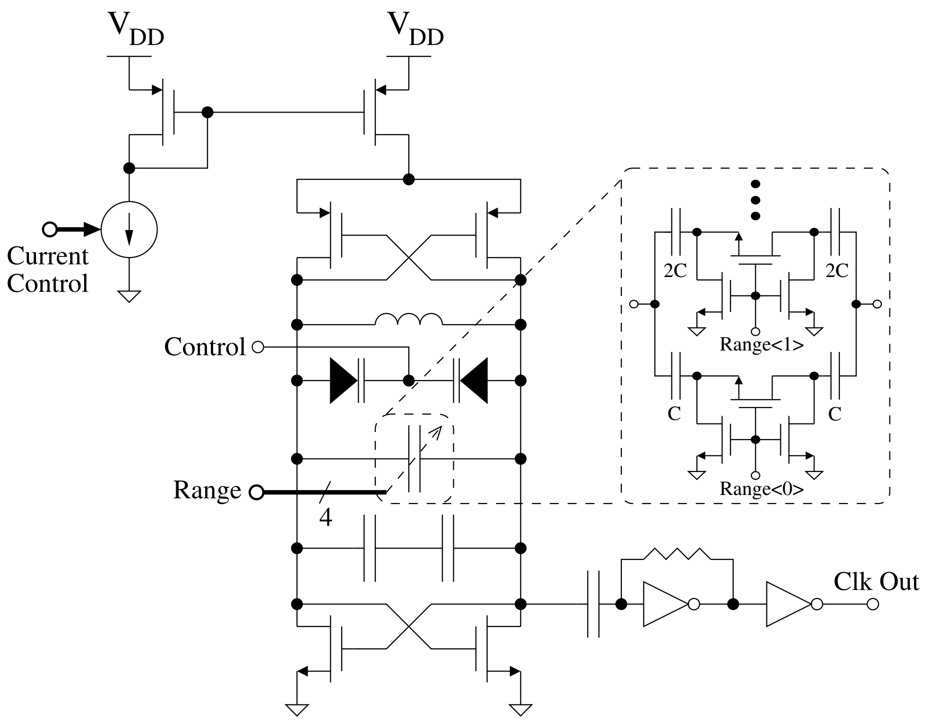


Figure 7.11 – Differential LC VCO

The VCO is a differential LC VCO whose schematic is shown in Figure 7.11. It uses both n-channel and p-channel MOSFETs so the g_ms add allowing a lower bias current. The structure is almost fully symmetric, with the output buffer the only asymmetric component. However, the size of the inverter acting as the output buffer is very small, so minimal capacitance is added. Also, for this application, a completely symmetric output is not needed. The inductor is a special symmetric inductor which is a p-cell in the IBM kit. Its windings are interwoven in such a way to maintain fully symmetric operation. Symmetric capacitors are made with two capacitors in series or parallel. Because of the small capacitor size needed, series capacitors were used.

The variable capacitor is implemented as an array of binary weighted capacitors. The switches which enable and disable the capacitors also have binary weighted sizes. The varactor diodes are implemented using the hyper-abrupt junction varactor diode option available in the process. The control voltage can range from 600 mV to 1.8 V. Below 600 mV, the varactor diodes begin to conduct. The current which flows through the VCO core is adjustable and varies between 150 μA and 745 μA. Using a lower current will increase the jitter but save power.

The range setting bits enable or disable different capacitor combinations to achieve the different frequency ranges. A varactor diode is in parallel with the capacitors which makes the frequency adjustable. Using small ranges with a low VCO constant gives better jitter performance, however it is more difficult to find the correct range for operation. The best way to tell if the VCO band is correct is to measure the VCO control voltage. This voltage should be between 600 mV and 1.2 V. The processor can do this automatically by driving out the VCtrl voltage on the analog test port, then using either the ADC in CSA3 to sample the voltage, or use the sigma-delta ADC to sample the voltage. A more crude way of finding a working range is cycling through the most probable ranges for a short time and see if the PLL locks. If it locks, the range is workable, however, it may not be optimal as far as noise and jitter are concerned.

Figure 7.12 shows a post-layout simulation of the frequency ranges of the VCO. In each range the control voltage is swept from 600 mV to 1.2 V. As desired, there is some overlap between ranges. The VCO constant K_O is not constant across all ranges and varies between 16.7 MHz/V and 27 MHz/V. This makes design of the loop filter more difficult.

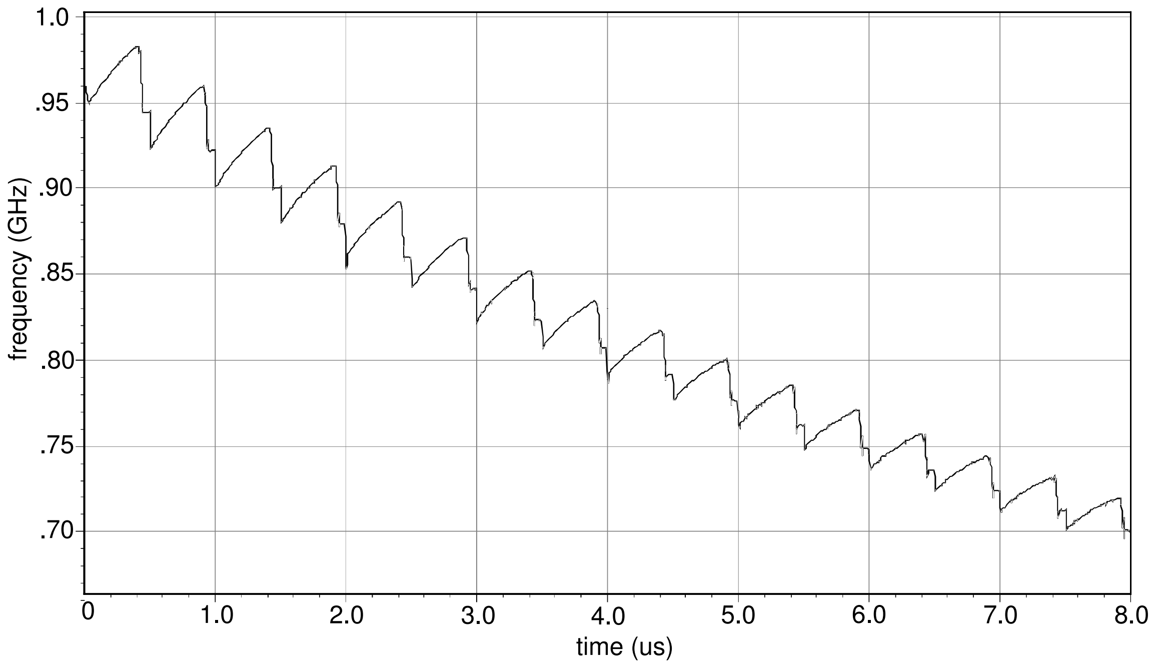


Figure 7.12 – VCO frequency ranges

7.1.2.4 Phase-Frequency Detector

The schematic of the phase-frequency detector (PFD) is shown in Figure 7.13. It is a standard extended-range PFD which has a linear response from -2π to 2π. Outside of that range, it will produce an output which attempts to push the VCO in the correct direction to lock the PLL, however its output is not easily characterized or linear.

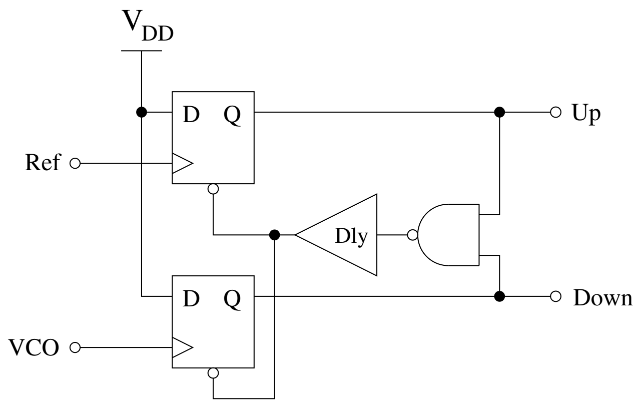


Figure 7.13 – Phase-frequency detector

The length of the delay sets the width of the anti-backlash pulse width. The anti-backlash pulse width is the width which both the up and down control signals will go high if the inputs are perfectly in phase. Some small anti-backlash pulse is required to linearize the PFD transfer function around the zero phase error point. If not present, PVT variations may cause a discontinuity in the transfer function around zero phase error having disastrous consequences for loop stability and phase noise.

If this PFD is connected to a charge pump with current I_{pump}, the loop gain K_D is given by

$$K_D = \frac{I_{\text{pump}}}{2\pi}. \tag{7.14}$$

7.1.2.5 Loop Filter

The loop filter consists of a charge pump and a C-RC filter combination as shown in Figure 7.14. The core elements are C_O and R_O. C_O combined with the charge pump creates an integrator, and R_O creates a loop-stabilizing zero at $f = 1/(2\pi R_O C_O)$. Because of the integrator formed by C_O and the integrator formed by the VCO, this is a second-order loop filter. The capacitor C_A helps filter out the high-frequency “teeth” on the control voltage caused by the discrete nature of the system. The pole created by this capacitor should be well above the crossover frequency, otherwise it will be unstable.

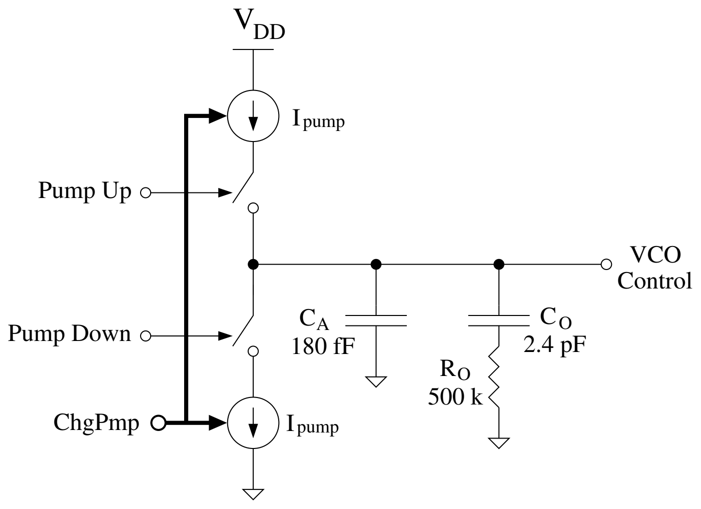


Figure 7.14 – Charge pump and loop filter

It is common to use a third-order loop filter by adding another R-C pole after the existing one. This helps further suppress the out of band components on the control line and improves jitter performance. This was not done because the added series resistor would add much parasitic capacitance to the control voltage line, overwhelming the C_A, and requiring the charge pump currents to be increased. The application does not require extremely good jitter performance, so this was deemed unnecessary.

The PLL will be operating with a reference frequency of 16 MHz, or a UWB symbol rate of 10 MHz. The loop filter was designed for a crossover frequency of 500 kHz, and the adjustable charge pump current allows this to be adjusted between 200 kHz and 1 MHz. The zero is located at 133 kHz. The non-dominant pole created by C_A is at 1.8 MHz. The simulated loop-filter response is shown in Figure 7.15. Three curves are shown for different values of I_{pump}.

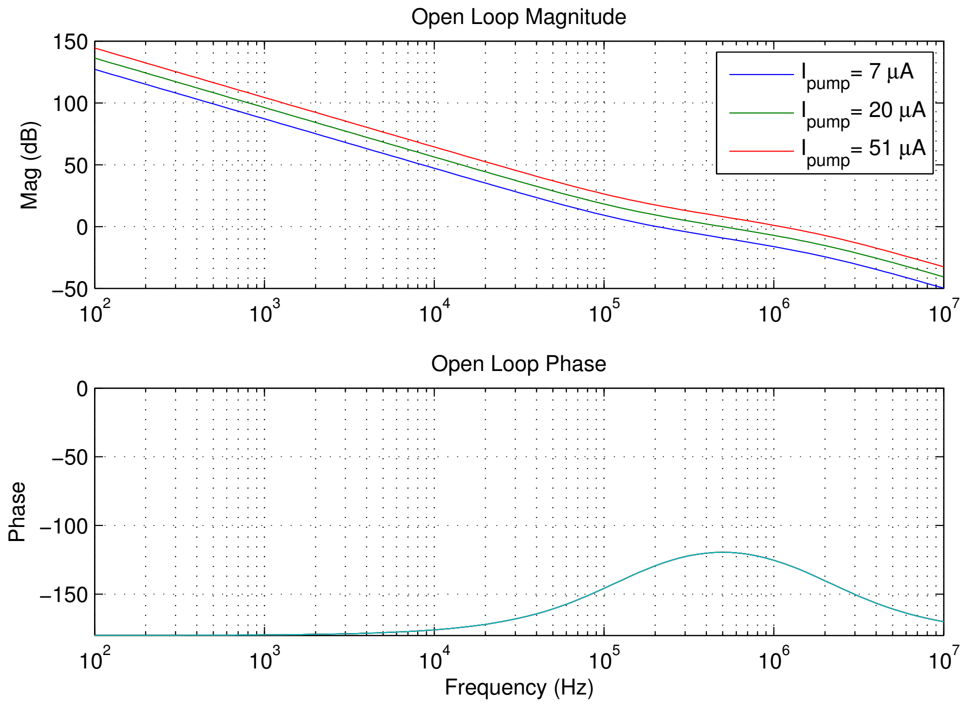


Figure 7.15 – Loop filter open-loop response

Table 7.1 gives the simulated loop bandwidth and phase margin for a number of different charge pump current settings.

7.1.2.6 UWB Receiver Pulse Generator

The PLL can also use information from the UWB receiver's symbol synchronization circuitry to change its clock frequency to stay synchronized with the incoming data stream. The UWB receiver produces a clock pulse and direction signal when a symbol is received

Table 7.1 – Loop-filter response

I_{pump}	ω_c (kHz)	Phase Margin
7 μA	204	51
12 μA	316	58
20 μA	494	60
34 μA	786	58
51 μA	1098	53

which is out of sync. The pulse generator produces a short pulse on either the up or down control line to shift the VCO control voltage to correct the synchronization error. This circuit is shown Figure 7.10 on page 184.

The circuit works by using an adjustable delay to produce a short pulse. The charge pump either uses the signal from the PFD or the signal from the UWB receiver pulse generator depending on the status of the UWB receiver. If the UWB receiver is receiving a packet and this type of synchronization correction is enabled, these generated pulses are used. The width of the pulses affect the frequency change for each received symbol error.

The loop filter is designed to operate with a reference frequency of 16 MHz; however, the 10 MHz frequency which the UWB receiver receives symbols, and therefore may apply error signals to the PLL, is comparable so the reference frequency, so stable operation should result if the other settings are set properly.

After a UWB packet reception, the control of the PLL will once again revert to being controlled by the reference frequency. It will take a short time to re-lock to the reference frequency. During packet reception the PLL lock indicator signal as described in the next section is not valid.

7.1.2.7 PLL Lock Detect

The purpose of the lock detector is to sense when the width of the control pulses have become sufficiently small for a sufficient period of time which indicates that the phase error is small. This circuit is shown Figure 7.10 on page 184. The lock detect circuit has two parts. In the first, the two control signals are combined, and pulses less than a certain width are filtered out. The width threshold sets the maximum phase error which the PLL must be below before a lock can be declared. The second stage is a one-shot delay which is reset by any pulse which is not filtered out. This delay time sets the length of time the PLL must be within the specified phase error before a lock is declared. Both of these two times are adjustable via the PLL_CR register.

The first stage has an adjustable delay which sets the maximum width of the pulse. However, due to the PFD producing anti-backlash pulses around zero phase error, the rejection pulse width does not directly correspond to phase error, instead the rejection pulse width corresponds to the phase error plus the anti-backlash pulse width (of about 1 ns). The second stage implements the delay by charging a capacitor to a threshold. The charging current is adjustable to produce an adjustable delay. Any pulse which is above the threshold width will reset the capacitor and re-start the delay.

7.1.3 Digitally-Controlled Oscillator

The DCO is a relaxation oscillator whose frequency is controlled by a 14 bit digital word. It is a low-power but highly adjustable oscillator which operates from a few kHz to 200 MHz. It can start very quickly. Its jitter performance is not good enough to allow it to be used as the reference clock for RF transmission and reception however. The core of the oscillator is shown in Figure 7.16. It works by charging a capacitor with a constant current until a threshold voltage is reached, at which point the other capacitor begins charging and the previous one is discharged. The 14-bit setting word affects the capacitor charging current, and adjustable capacitors set which frequency range is used.

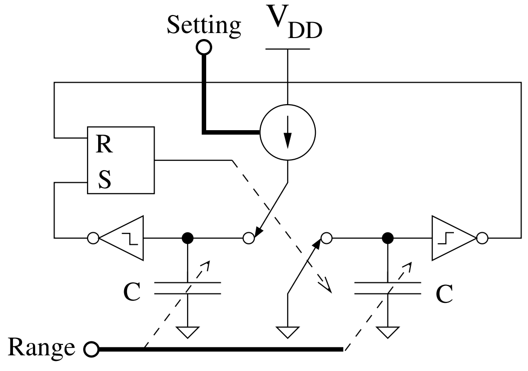


Figure 7.16 – Oscillator architecture

The frequency of oscillation is given by

$$F_{\mathrm{DCO}} = \frac{I}{2CV_{\mathrm{thres}}} \tag{7.15}$$

where V_{thres} is the threshold voltage which the capacitor is charged to. The core frequency has a nice linear dependence on current.

The actual implementation of the core is shown in Figure 7.17. Not shown, the current I_B and the capacitor are adjustable. M1 and M2 form the charging and discharge path for the C, and M4, M5, and M6 form the threshold comparator. This transistor configuration is a modified inverter, where a diode-connected transistor M4 has been added in the NMOS path to increase the threshold voltage. M6 is a thick-oxide high-V_t device to minimize shoot-through currents when the inverter is switching. This is desirable to minimize power consumption and to decrease the circuit threshold voltage dependence on the power supply voltage. The circuit has a V_{thres} of about $2V_t$ and has almost zero shoot-through current. M1 and M3 reset the circuit when the other half of the circuit is charging its capacitor.

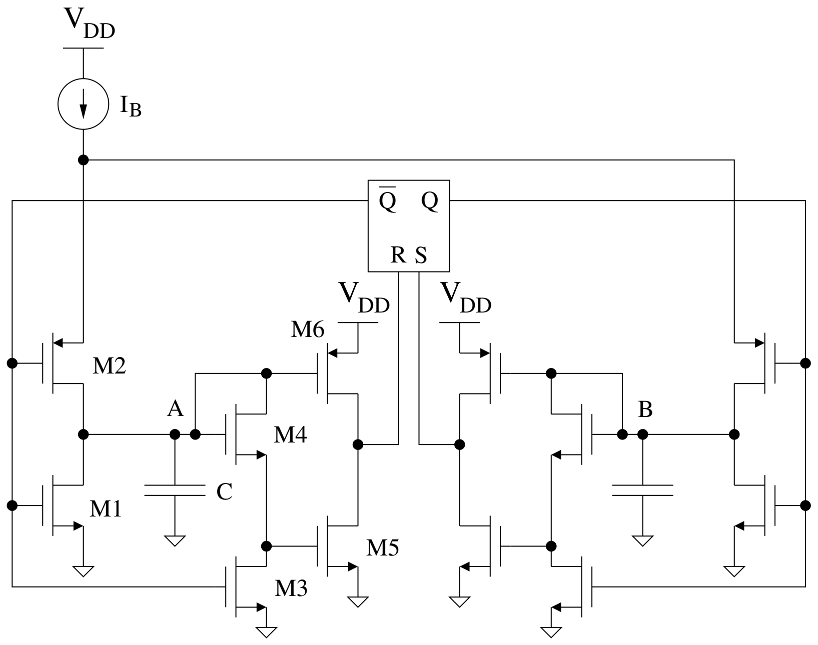


Figure 7.17 – DCO core schematic

In Figure 7.17 the capacitor is adjustable to select the range of the oscillator and I_B is adjustable with a 14-bit DAC to select the frequency. The DAC used is a standard R-2R resistor ladder DAC which is shown in Figure 7.18. The DAC is a standard architecture and has the advantage of not requiring the matching of very large and small resistors.

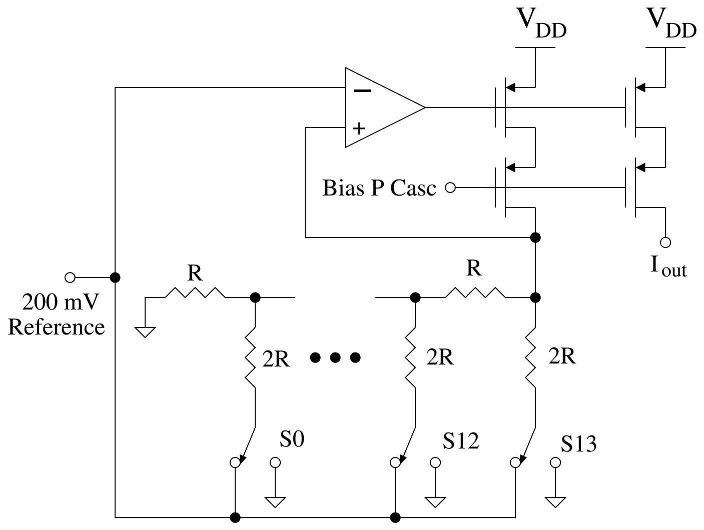


Figure 7.18 – DCO current DAC

A standard R-2R resistor ladder cannot achieve 14-bit accuracy without complex laser trimming or calibration of some sort. However, because in this case there will be feedback around the DAC, this DAC does not need to have good performance, it only needs to have many possible settings so the feedback loop can eventually find one which produces the desired output frequency. Therefore no calibration is used.

The low 200 mV reference was chosen to reduce power consumption and reduce the required resistor values. The R value is 160 kΩ.

7.1.3.1 DCO Stabilization

The frequency of the DCO is not accurate or repeatable from chip to chip due to PVT variations. It is mostly affected by process, such as V_t shifts which shift the V_{thres} of the comparators, or resistance mismatch in the DAC. For this reason, a mechanism is provided where an external crystal oscillator can stabilize the frequency of the internal DCO to a known value. The DCO stabilization utilizes the low-frequency crystal oscillator as the reference clock.

Accurate stabilization can be achieved quickly: within 4 - 10 crystal oscillator clock cycles. The circuit works by counting the number of DCO clock cycles present in one crystal oscillator clock cycle, and adjusting the DCO to make it match a desired number.

After the frequency is close, it switches modes and clocks the number of DCO cycles in four crystal oscillator clock cycles. This gives a more accurate setting, with an expense associated with extra delay. A simplified functional diagram of the clock stabilizer is shown in Figure 7.19

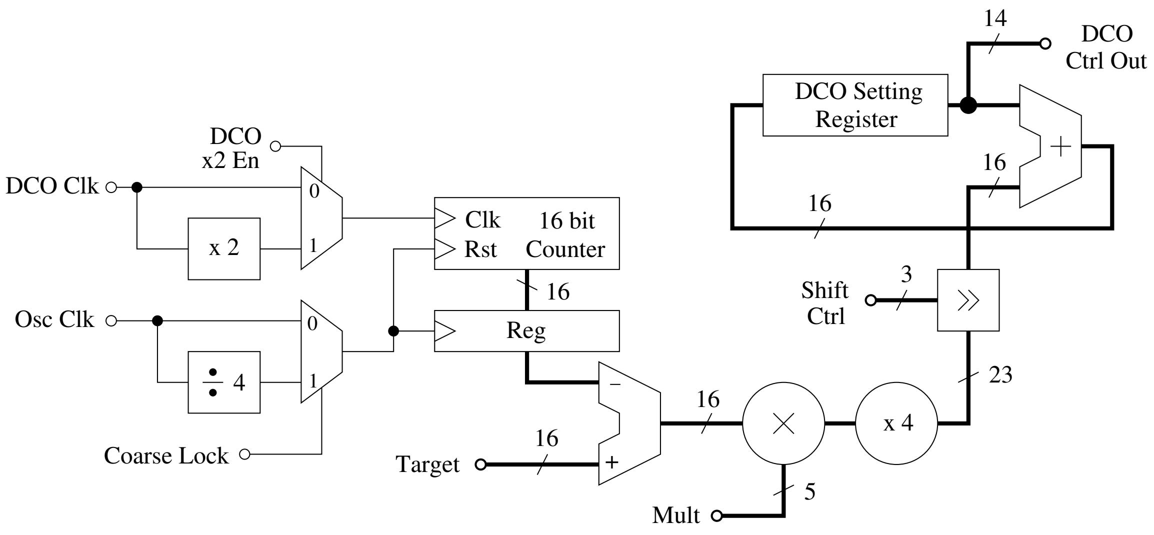


Figure 7.19 – DCO stabilization functional diagram

The stabilization circuitry uses a 16 bit counter to count the number of DCO clock cycles within each oscillator clock cycle. Not shown in the figure, this value may be shifted by two bits to compensate for the option to divide the oscillator clock by four. This measured value is subtracted from the target value stored in the CG_Target register. The error signal is then multiplied by a user-selectable amount, and right-shifted by a user selectable amount. Finally the error signal is added to the current DCO setting.

The multiplier value is set via the scale registers. the scale registers define four separate multiplier values. Each value is associated with a different DCO range setting. The multiplier accepts a 5 bit unsigned value and a 16 bit signed value and produces a 22 bit signed result. The shift value is also set via the scale registers. There are also four different shift values, and one is selected depending on the DCO range setting. Not shown in the figure, the shift control value is incremented by one if the DCO x2 enable is set. The values of the multiplier and shifter can be set once during calibration and never changed.

At first, the circuit operates with the oscillator clock not divided by 4. Once when the error signal is within 8 cycles of the desired target, the coarse lock signal is set. At this point the circuit switches to using the oscillator clock divided by four. This allows more accuracy to be obtained. Once when the DCO is within one clock cycle of the target, the fine lock signal is set. Both of these signals are observable via the status register, and interrupts can be triggered on either signal. Once when the fine lock is set, the circuit disables all the computational part and only looks at the sign of the error. If the error is positive it increments the DCO setting; if the error is negative it de-increments the DCO setting. It continues to monitor the error and will disable the fine lock and even the coarse lock if the error is large (i.e. the target frequency is changed).

In actuality however, the jitter of the DCO and the jitter of the low-frequency crystal oscillator only practically allow the coarse lock to be set, and the fine lock is only set occasionally for a single cycle before unlocking the next. Therefore, the coarse lock is the only valid lock signal.

7.1.4 Digital-Supply Oscillators

Two more oscillators operate on the digital supply and can provide a clock before the analog section is enabled. They are also the lowest-power oscillators. They both suffer from process variations, and cannot be stabilized by an external crystal except via creative software.

The variable frequency oscillator is a ring oscillator whose frequency will change as the supply voltage is scaled. This makes it useful as a set-and-forget clock when using DVS. When using DVS, the frequency of this clock will track the supply voltage as necessary to keep the processor operating within its recommended operating conditions. There is a control which allows one to adjust the frequency. The frequency will be sensitive to PVT variations (but that is the point of the design), so it cannot be used as reference clock for example when communicating via the UART. The variable frequency oscillator can operate over the entire core supply voltage range.

The constant frequency oscillator also operates on the digital supply, but is designed to be unaffected by supply voltage variations so supply voltage ripple will not cause jitter. It can be used as a reference clock to communicate over the UART. However, it will be affected by process variations and has a linear temperature dependence. Internally, the oscillator bias circuit applies a current through a diode, takes the resulting voltage across the diode and produces a reference current which is V_D/R. The reference current is then used in an oscillator which produces a frequency linearly dependent on the current. Therefore, the temperature dependence of the diode voltage is directly converted into temperature dependent frequency. The frequency of this oscillator could be measured against a reference frequency to measure the die temperature. This architecture was chosen because of the simple implementation and low power consumption. This analog circuit is self-biased and can operate without the analog supply. It uses the exact same core as the DCO as shown in Figure 7.16 and Figure 7.17 except the capacitor is not adjustable, and the bias current is selectable from one of 16 values.

Though it is designed so the frequency is independent of the supply voltage, some dependence still exists, and the frequency of this oscillator will increase slightly with decreasing supply voltage until the supply falls below 1.2 V. This is because the V_{thres} in Equation 7.15 decreases slightly at lower supply voltages. At supply voltages lower than 1.2 V, the analog biasing circuits fail and the frequency drops very rapidly. This oscillator also has a control which selects the frequency of operation, however the frequencies will not be consistent from chip to chip.

7.2 DVS-Capable Power Supply

This section presents a power supply for the digital and analog sections of the chip. The digital supply supports DVS which is critical to lower the power consumption of the digital core. The power supply is the most significant portion of the power management hardware which is shown in Figure 7.20.

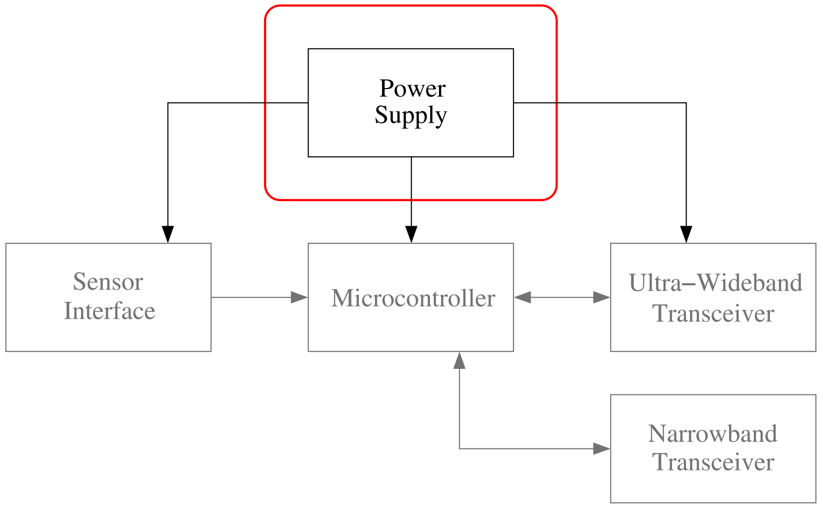


Figure 7.20 – Power section enclosed in red box.

7.2.1 Introduction

The power supply is a critical part of the system design to assure low-power operation. multiple tradeoffs exist between power consumption and performance. The simplest supply is a linear regulator. Two types exist, ones that use a n-type pass element and ones that use a p-type pass element. The regulators which use a p-type pass element have a lower dropout voltage and are the best option for a WSN, however they are more difficult to compensate. A low-dropout voltage regulator (LDO) is shown in Figure 7.21.

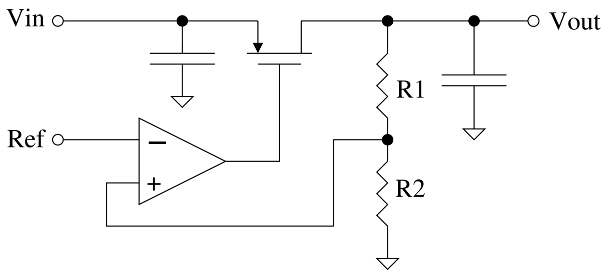


Figure 7.21 – Architecture of a low-dropout regulator.

The output voltage of the regulator is given by

$$V_o = \frac{V_{ref}(R_1 + R_2)}{R_2} \tag{7.16}$$

where V_{ref} is the reference voltage.

The linear regulator is a simple regulator with good power supply quality. The transient response of the regulator is the best of any type of power supply, so it is desired for RF or other high-performance analog applications where a good quality supply is desired. It also has no output ripple. The input and output capacitors may or may not be required for stability, however they generally improve noise and transient response. The main drawback of this regulator is efficiency. The efficiency of the regulator is given by

$$\eta_{LDO} = \frac{P_{out}}{P_{in}} = \frac{I_o V_o}{V_{in}(I_o + I_{amp})} \tag{7.17}$$

where I_{amp} is the current draw of the error amplifier. In the limit when $I_o \gg I_{amp}$ the efficiency is simply V_o/V_{in}. This sets an upper bound on the efficiency which, if the input and output voltages are much different, is very poor.

A supply with improved efficiency is a buck regulator as shown in Figure 7.22. It

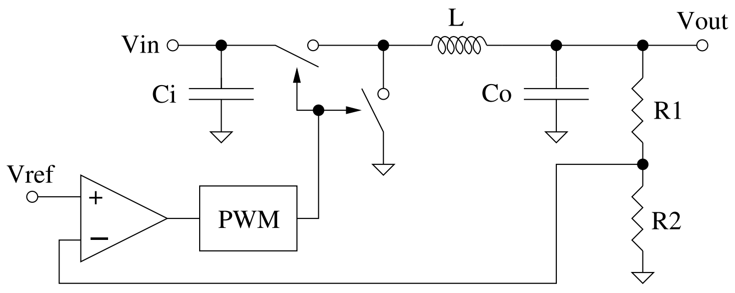


Figure 7.22 – Architecture of a buck regulator.

consists of a pair of complementary switches which produce a square wave output signal. The square wave is then filtered by an LC filter to produce the DC component which is applied to the load. The output voltage is the same as the linear regulator and is given in Equation 7.16. The efficiency can ideally be 100% as there are no intrinsically lossy elements in the design, i.e. an ideal switch is lossless.

Because of the switching operation, there exists some ripple in the output voltage which can cause problems with analog circuits. The switching action also generates EMI which can interfere with nearby circuits, especially RF circuits. The transient response of the regulator is not as good as a linear regulator because the bandwidth of the control loop is generally smaller because of the sampled-data nature of the system and the fact that the current through an inductor cannot change quickly.

7.2.2 DVS Theory

Dynamic voltage scaling is the practice of scaling the supply voltage of a digital circuit when less performance is required to save power. In a digital circuit, the switching energy consumed for a given operation is

$$E = n\alpha C V_{dd}^2 \tag{7.18}$$

where n is the number of cycles required for an operation, C is the capacitance of the digital circuit, and α is the activity factor. Decreasing power in a circuit generally focuses on either decreasing α and C by various architectural and circuit choices, or decreasing V_{dd}. Decreasing V_{dd} is very effective at reducing energy consumption because of the quadratic dependence of energy on voltage.

The gate delay, and therefore maximum processor frequency, is also dependent on supply voltage. The gate delay is given by

$$T_g = \frac{V_{dd}}{K(V_{dd} - V_{th})^a} \tag{7.19}$$

where K and a are processor dependent and a is generally around 2. If V_{dd} is much larger than V_{th}, the gate delay goes as $1/V_{dd}$ or the maximum frequency is about linear with voltage. Therefore, scaling the voltage decreases the energy quadratically, the clock frequency linearly, and power cubically.

However, the previous discussion assumes the power supply is 100% efficient. If using a linear regulator to supply the power to the processor, half of the power savings (in dB) which DVS confers are dissipated in the linear regulator. This is because from Equation 7.17, as the output voltage is scaled, the regulator efficiency decreases linearly. If the linear regulator is used as the supply in a DVS system, the total energy savings only decreases linearly with voltage, not quadratically. However, a buck regulator can supply the power at a relatively constant, high efficiency to the load regardless of the output voltage allowing the full quadratic energy savings to be realized.

Presently, there are no small embedded systems targeted for low-power wireless sensor networks which integrate a buck regulator. Few support DVS at all, and the ones which do all integrate a linear regulator. Using an external buck regulator is difficult due to the tight integration required with the processor and the extremely stringent power requirements. Generally, a buck regulator has very poor efficiency at light loads due to the more complicated circuitry which must operate continuously which consumes power. This makes it difficult to use a buck regulator in a sensor network application where long-duration sleep periods with very little power consumption is common. In this dissertation, buck regulator has been designed to supply power efficiently even at light loads.

7.2.3 Buck Regulator

The standard buck regulator is shown in Figure 7.22. Its efficiency is determined by a number of losses in the circuit originating from various sources. These are the I^2R losses, the inductor core losses, the switching losses, the gate drive losses, and the control circuit losses.

First, there are the I^2R losses in the inductor and MOSFETs. These can be calculated by finding the RMS current through each element and using the resistance of the inductor and on-resistance of the two switches to find the I^2R losses. These losses are made worse if there is a large ripple current through the inductor because the ripple current component adds to the RMS current but does not add to the output current. Of course these can be improved by selecting high-rating switches and inductor.

The inductor core losses originate from the hysteresis within the ferrite inductor core. During each switching cycle, current increases and decreases in the inductor which causes the internal B-field to trace out a hysteresis loop. The area within the hysteresis loop is the energy lost during each cycle. This type of loss can be reduced by decreasing the ripple current which decreases the size of the hysteresis loop. It is also proportional to switching frequency.

The switching losses is the power absorbed by the switches during their turn-on or turn-off events. The idealized switching waveforms are shown in Figure 7.23. When the switch

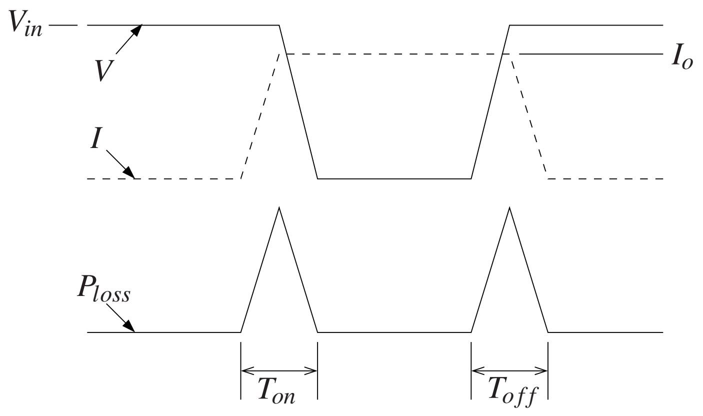


Figure 7.23 – Switching losses in a buck regulator.

turns on, the current increases from zero to its maximum value before the voltage decreases from its maximum value to a small value. The opposite occurs during turn-off. During these times, a large current and voltage are present simultaneously across the switch which produces a large peak power dissipation in the switch. This type of loss is proportional to the switch turn-on/off time (which is proportional to the capabilities of the gate driving circuitry) and the switching frequency. It can be given approximately as

$$P_{Lsw} = \frac{T_{on} + T_{off}}{2} V_{in} I_o f_{sw} \tag{7.20}$$

where f_{sw} is the switching frequency. The turn-on time and turn-off time can be decreased by increasing the driving capabilities of the driver which is driving the gates of the power transistors. For a solution where external power MOSFETs are used, this has the disadvantage of creating ringing and EMI problems, but these are less severe in an integrated solution. The switching loss is proportional to the frequency of operation.

The gate drive loss is the power lost due to the charging and discharging of the gate capacitance of the power MOSFETs. this loss is approximately

$$P_{Lgate} = (Q_{g1} + Q_{g2}) V_g f_{sw} \tag{7.21}$$

where Q_{g1} and Q_{g2} is the total gate charge of the two power MOSFETs and V_g is the gate drive voltage which in this case is V_{in}. This loss is also proportional to switching frequency.

The final loss is the losses due to the power consumed by the control circuit. The error amplifier and other control circuits consume some power which directly decreases the efficiency of the regulator.

The standard buck regulator uses a PWM control scheme. In this scheme, the on-time of a constant-frequency signal is adjusted to control the output voltage. The switches operate at a constant frequency which makes it easier to design the output filter. The transient response of the PWM-controlled regulator is also very good. Finally, the power-handling capability of the regulator is the best possible given the ratings of the switches and inductor. For high loads, the PWM controlled converter is the best in virtually all respects; however, during low loads, the efficiency is very poor. This is because many losses in the regulator are constant, or have a high minimum value, which do not scale as the output power scales, therefore causing poor efficiency at low loads. For example, the gate drive losses are constant and the ripple current in the inductor is always present whatever the load. The ripple current causes I^2R losses, switching losses, and inductor core losses even under no-load conditions. All these losses are small relative to the rated output power so only slightly decrease peak efficiency, however at light loads they dominate and produce poor efficiency.

Another regulation scheme which can have very good light load efficiency is pulse-frequency modulation (PFM) or discontinuous conduction. Under this mode of operation, at light loads the frequency of switching drops dramatically. In between switching events, the current in the inductor goes to zero (hence the name discontinuous conduction) and all activity in the circuit stops until another switching event is required. In this way the switching losses, gate drive losses, I^2R losses, and inductor core losses can all scale with output power, leaving only the control overhead current as the only fixed loss.

A purely PFM scheme generally has much higher losses at rated load and has a lower maximum rated output power because it has a higher ripple current. Therefore a hybrid scheme is popular where a PFM control is used at light loads and at higher loads the regulator switches and uses PWM mode. The most popular type of PFM control in this scheme works by having a comparator decide when the output is below a threshold. When it is, the PWM control is engaged and a number of switching events occur to charge up the output capacitor to another high threshold at which point all switching is stopped until the output decays to the lower threshold again. This type of control scheme has a larger output ripple, but is relatively simple to implement if the PWM control scheme is already present.

In this design, the PWM control scheme is eliminated completely and a very simple PFM scheme is used to control the output voltage. Due to the very small power levels involved, the high-power optimized PWM scheme is not needed and only increases circuit complexity which in turn increases the fixed control circuit power loss. Because of the very simple control scheme, the low-load efficiency of the regulator is very good.

7.2.4 Buck Regulator Design

The architecture of the design is shown in Figure 7.24. It consists of a buck regulator utilizing a single comparator to keep the output in regulation. The comparator turns on M1 when the output voltage is too low and turns it off when the output goes high again. M2 is turned on when M1 turns off and stays on until the second comparator senses the current in this MOSFET is zero. The second comparator is responsible for shutting off M2. In this way synchronous rectification is achieved for higher efficiency.

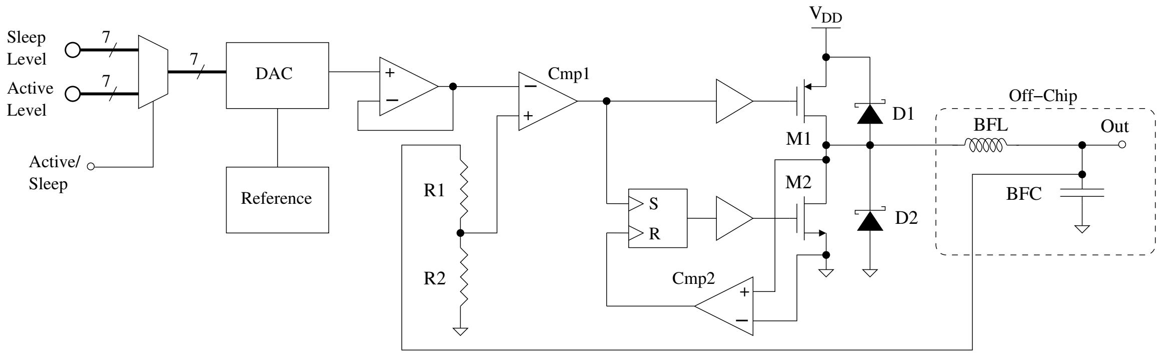


Figure 7.24 – Architecture of the buck regulator.

The advantage of this control scheme is the simplicity of implementation and high efficiency at low power levels. The main disadvantage is that the controller always operates in discontinuous conduction mode so the ripple current through the inductor is very large.

The large ripple current leads to higher I^2R losses in the MOSFETs and inductor as well as larger core loss in the inductor. However, because the design is targeted at very low power applications, the extra cost of the inductor and size of the power switches needed to compensate for this effect is negligible.

Because an ultra-wideband receiver is present in the design, it is desirable to decrease the switching frequency and increase the ripple voltage for the digital regulator where the ripple is not important. This can be accomplished by replacing R2 in Figure 7.24 with the schematic shown in Figure 7.25. The MOSFET in the schematic turns on when M1 turns on and off otherwise. R3 is much smaller than R4. With this circuit the ripple can be adjusted to any arbitrarily large value by adjusting the ratio of R4 to R3. Without this circuit, the presented regulator produces about 2 mV of ripple and therefore switches much more often than it would if allowed more output voltage ripple.

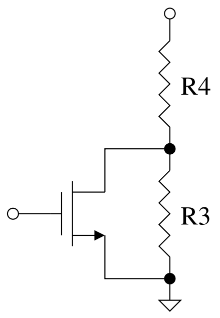


Figure 7.25 – Ripple voltage adjustment.

The dynamic voltage scaling is implemented with a very low power DAC controlled by the microcontroller. The DAC consumes about 140 nA. The dynamic voltage scaling can be used by the microcontroller to save power in two ways. The first is when the microcontroller is active where the voltage is adjusted in response to varying processing requirements on the microcontroller. The second is when the microcontroller is in sleep mode. In this state, the voltage can be scaled dramatically, down to the minimum required for RAM retention, in order to reduce leakage power consumption in the microcontroller core and related circuits. Hence, a multiplexer is used to select which voltage level should be used depending on the operating state of the microcontroller. The microcontroller requires a power-good signal from the regulator to sequence the power-up and to control when the microcontroller can come out of sleep. When transitioning out of sleep, the microcontroller waits for the power good signal before proceeding to execute instructions.

The op-amp which buffers the DAC voltage serves the function of providing soft start-up and controlled transitions from a lower voltage to a higher voltage. The op-amp also produces the power-good signal. The op-amp schematic is shown in Figure 7.26. The

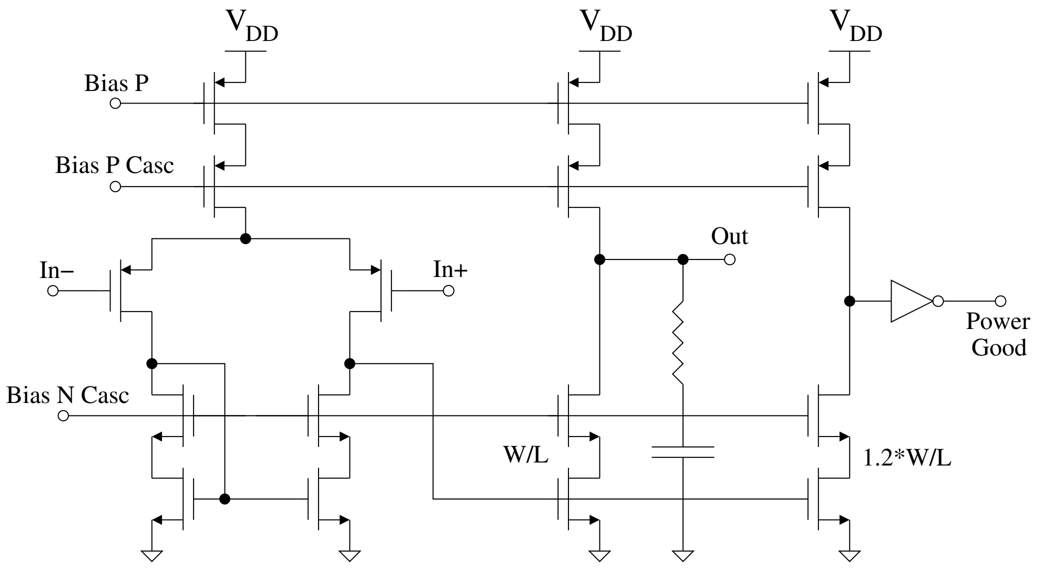


Figure 7.26 – Schematic of the reference/DAC buffer.

op-amp is designed using very small bias currents of 5 nA. These small bias currents, combined with a large output capacitor, create a very slow positive-going slew rate. This positive-going slew rate sets the start-up time and transition time of the converter. The slew-rate of the implemented design is set at 1 V/ms. The time for the output voltage to make an upward transition is given by

$$T = \frac{\Delta V}{SR}\left(\frac{R_2}{R_1 + R_2}\right) \tag{7.22}$$

where *SR* is the slew rate of the reference buffer and ΔV is the change in output voltage.

A second output stage on the DAC buffer produces a digital signal which is low when the buffer is slewing and high when it is in regulation. This signal is used as the power-good signal to indicate when the converter has a stable output voltage. The W/L of the n-channel transistors of the second output stage are 1.2 times the W/L of the actual output stage. This guarantees the output of the second stage is a digital signal which is high when the op-amp is not slewing.

The SR flip-flop, which controls M2 in Figure 7.24, is implemented as a regular flip-flop and a glitch-generating circuit as shown in Figure 7.27. It must be edge-triggered for both the reset and set inputs. The set input is generated by a comparator which detects when the current is zero through M2.

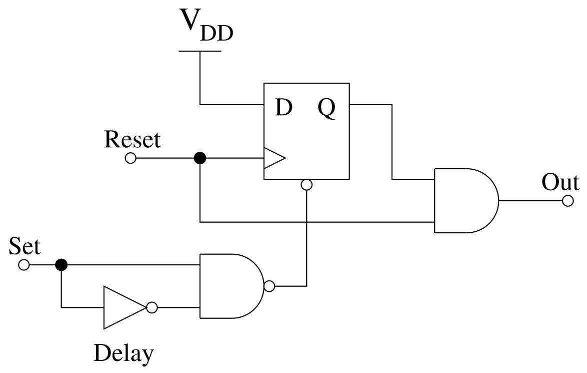


Figure 7.27 – Schematic of the SR flip-flop.

Figure 7.28 shows the comparator which detects the zero-current condition through M2. It is a basic comparator with a couple of modifications. The comparator is pre-charged, and

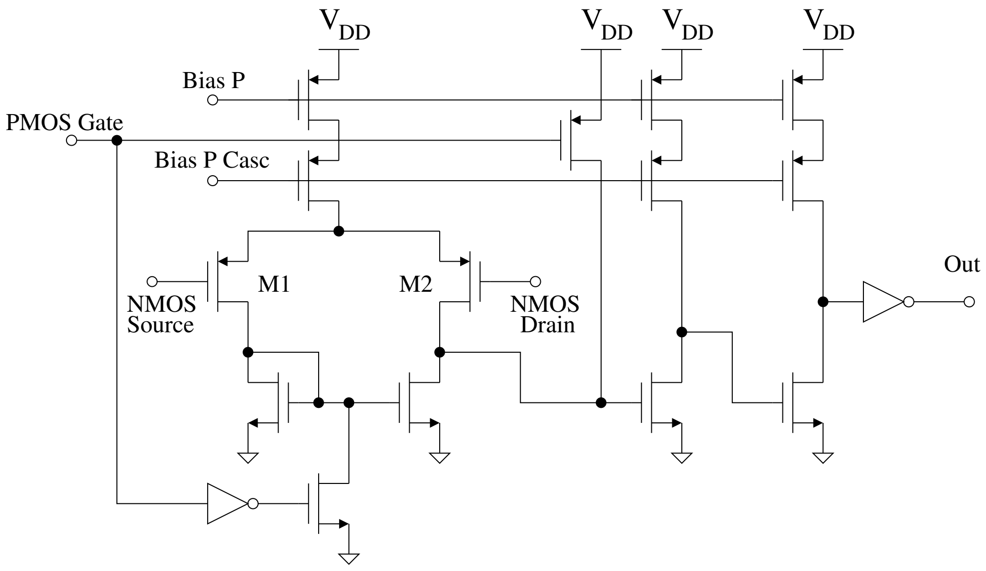


Figure 7.28 – Schematic of zero-current comparator.

the output is forced to zero, when M1 in Figure 7.24 is turned on. Without this pre-charging operation, if M1 is turned on for a very short time which would only allow a small forward current, the output of this comparator would not cycle from low to high, and therefore it would not shut off M2. The current would then reverse through the inductor and cause a glitch in the output voltage. With this pre-charge operation, the output is always forced low when M1 is on and it will always go high as the current through M2 transitions from positive to negative, thereby shutting off M2.

The comparator input differential pair is not matched. In Figure 7.28, M1 is slightly larger than M2. This makes a systematic input offset voltage which helps compensate for the finite propagation time of the comparator. If M1 and M2 were matched, the current would always reverse through the inductor with a peak reverse current given by

$$I_{rev} = \frac{V_{out}\, t_{prop}}{L_{BFL}} \tag{7.23}$$

where t_{prop} is the propagation time through the comparator and gate drive circuitry. To compensate for this, a systematic input offset voltage can be introduced. The ideal input offset voltage can be calculated by $I_{rev}R_{on}$, where R_{on} is the on resistance of M2 from Figure 7.24. This ideal offset voltage depends on both the input and the output voltage, so in a system with DVS, the input offset voltage will only be ideal at a particular output voltage which was set at 1.1 V in this design. When running at other output voltages the switching time will be non-ideal which will degrade efficiency slightly—by about 1-2%.

The gate drivers are designed to enforce a dead time between the upper MOSFET turning off and the lower MOSFET turning on to prevent cross conduction.

This buck power supply was first included in the third chip. In that chip it works, however the tolerance of the feedback resistors is unacceptable. This is not surprising as they are extremely large resistors (10 MΩ) and must have very small width to fit in a reasonable area. However the poor tolerance of these resistors made the actual output voltage highly variable from chip to chip. To fix this, in the fourth chip R2 in Figure 7.24 was made digitally adjustable with 6 bits for calibrating. There is an independent calibration setting for both the analog and digital regulators.

Furthermore, there is also digital calibration for the on-chip bandgap voltage reference. This was digitally adjustable on the third chip, however not over a wide enough range to fully correct for the process variations. Henceforth, on the fourth chip the calibration range was extended.

A bias network, low-power reference, power-up reset circuit, and startup circuit is also integrated in the design. These are shown in Figure 7.29. The power-up reset circuit (POR) stops the regulator from starting until after the supply voltage reaches a threshold of about 1.4 V. After the threshold is reached, the bias network and reference are enabled and the startup circuit creates a delay of 500 μs after which time the rest of the regulator

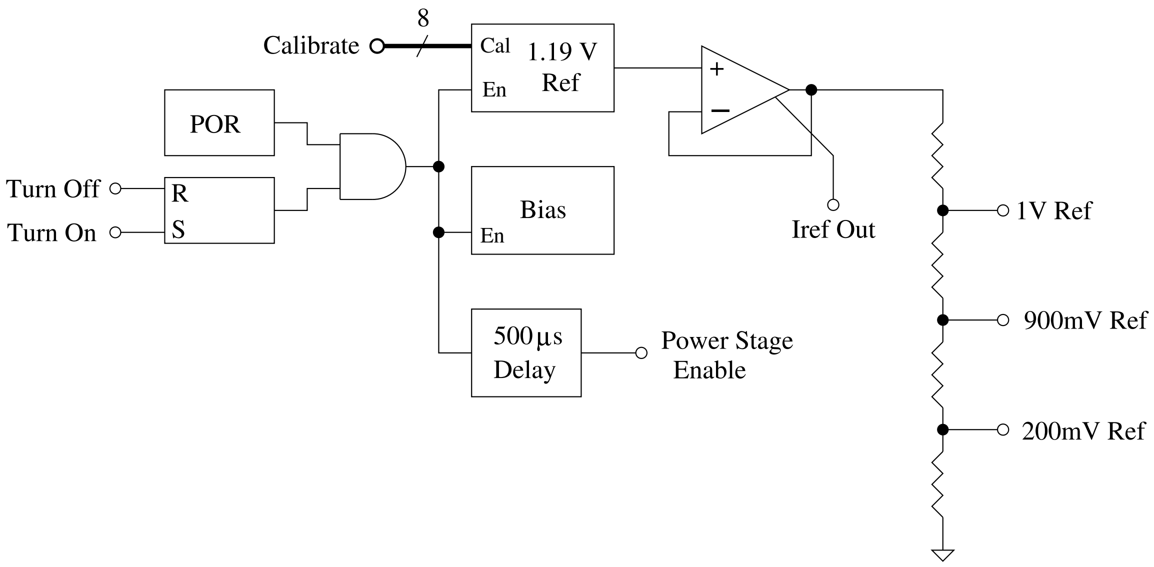


Figure 7.29 – Reference and supporting circuitry

is enabled. This allows time for the bias and reference to settle before enabling the power section. During this 500 μs of delay, the output capacitor of the reference buffer is shorted to ground. After the delay, the output voltage begins to rise and reaches the specified output voltage in a time given by Equation 7.22.

The bandgap voltage reference operates at 1.19 V as opposed to the standard 1.24 V because of the temperature coefficient of the integrated resistors. An op-amp buffers this reference which drives a resistor string to produce the three reference voltages used by the rest of the system. Also, a second output stage of the op-amp produces a constant reference current which is also used by the rest of the chip.

7.2.4.1 Simulations

The IBM process supports Schottky barrier diodes, so they were used to implement the protection diodes D1 and D2 on chip. The feedback resistors were also integrated on chip. With pads and associated ESD protection, the design occupies 0.156 mm^2. The no-load input current is 960 nA. This low current assures very high efficiency at light loads. The layout is shown in Figure 7.30. Some relevant circuit specifications are given in Table 7.2.

The layout of the design was extracted and simulated in Spectre. A simulation of the output responding to load current changes and demonstrating the dynamic voltage scaling is shown in Figure 7.31. The top graph shows the output voltage, the middle shows the

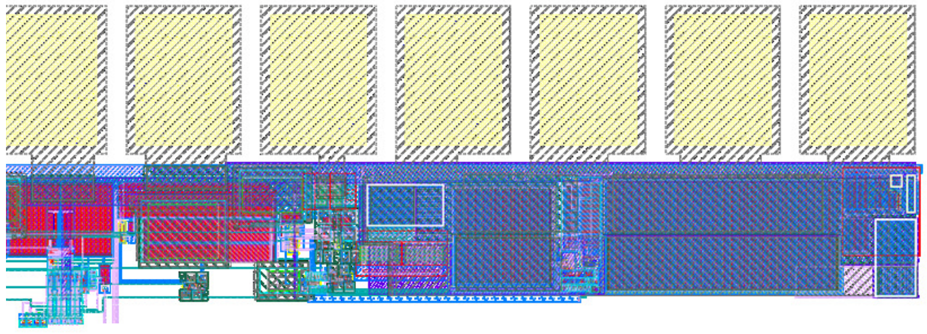


Figure 7.30 – Layout

Table 7.2 – Circuit Specifications.

Process:	0.18 μm
Area:	0.156 mm^2
No-load Input Current:	960 nA
Max Output Current:	50 mA
Output Voltage:	400 mV - 1.8 V
Input Voltage:	1.6 V - 3.6 V
Ripple Voltage:	2-20 mV*
Minimum V_{in} - V_{out}:	100 mV
NFET R_{on}	357 mΩ
PFET R_{on}	500 mΩ

*2 mV is minimum ripple at full output current, 20mV is ripple of implemented system with circuit shown in Figure 7.25.

load current, and the bottom shows the power good digital output. The simulation shows the converter starting up and then a load being applied. The output voltage is then decreased from 1.8V to 900mV and eventually raised back to 1.8V. As can be seen, the power good signal goes low during power-up and also during the transition from low to high voltage. The power-good signal is referenced to the output voltage, so when it is high it tracks the output voltage.

Figure 7.32 shows the simulated efficiency as a function of load current. As can be seen, the regulator retains a high efficiency even at very low loads. The peak efficiency is over 95% and it stays over 90% efficient from 40 μW to full load.

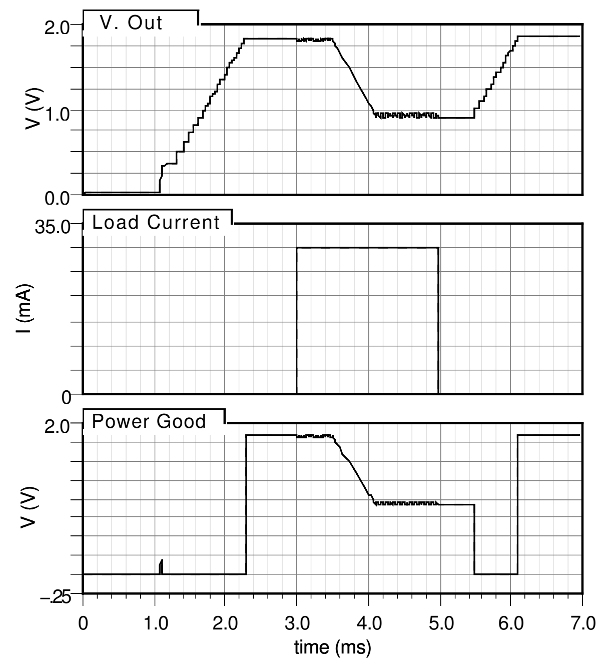


Figure 7.31 – Simulation with changing load current and dynamic voltage scaling
BFC = 20 μF BFL = 100 μH Vin = 3.3 V

7.3 Delta-Sigma ADC

The third and fourth chips integrate a delta-sigma analog to digital converter. The converter has no particular use when for the radiation detection application, but was included to allow a future sensor to be interfaced without the need for an external high-precision low-speed ADC. The ADC has a designed SNR of over 100 dB which is approaching 16 ENOB. It has a signal bandwidth of 1 kHz. The design uses a second-order modulator with a very high oversampling rate to achieve high resolution. It operates with single-bit quantization which simplifies implementation. The modulator operates at a 1 MHz clock. A digital sinc^3 filter produces the final digital output words at 8 kHz. The overall architecture of the modulator is shown in Figure 7.33.

For details of the design of the transfer function and modulator, see [88]. The transfer function of the modulator is given by

$$H(z) = \frac{(1-z)^2}{(z-0.45)(z-0.156)}. \tag{7.24}$$

A Matlab simulation of the modulator with a -3 dbFS input is shown in Figure 7.34. The

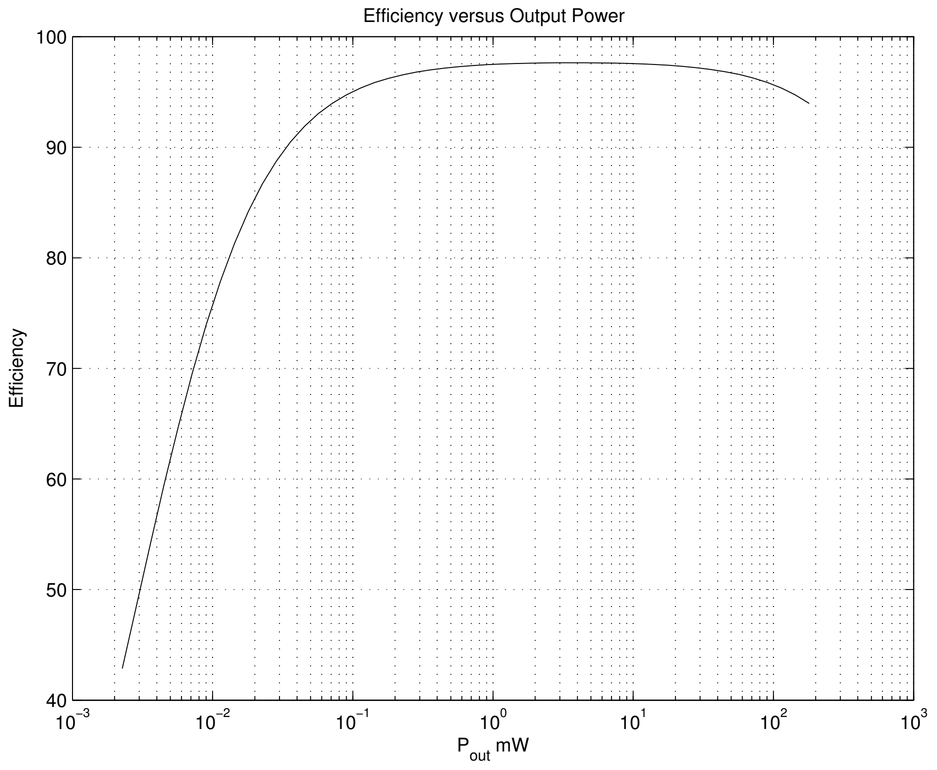


Figure 7.32 – Simulation showing efficiency as a function of output power
BFC = 20 μF BFL = 100 μH Vin = 3.3V Vout = 1.8V

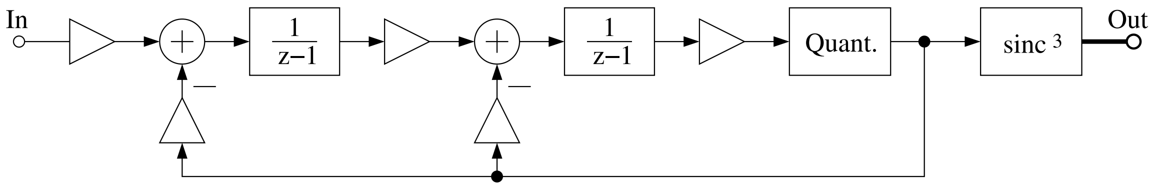


Figure 7.33 – Delta-Sigma ADC converter architecture showing the second-order modulator and the digital sinc^3 filter.

simulated SNR is over 100 dB. For the simulation a Hann window was used for spectral estimation.

The analog modulator is based on a design presented in [88]. It uses a switched-capacitor circuit as they generally have higher performance with similar power consumption when compared to their continuous-time counterparts. A fully differential signal path is used for precision and noise rejection, however the input is single-ended. The schematic is shown in Figure 7.35.

The ratio of capacitor values sets the gains and the value of the transfer function. More

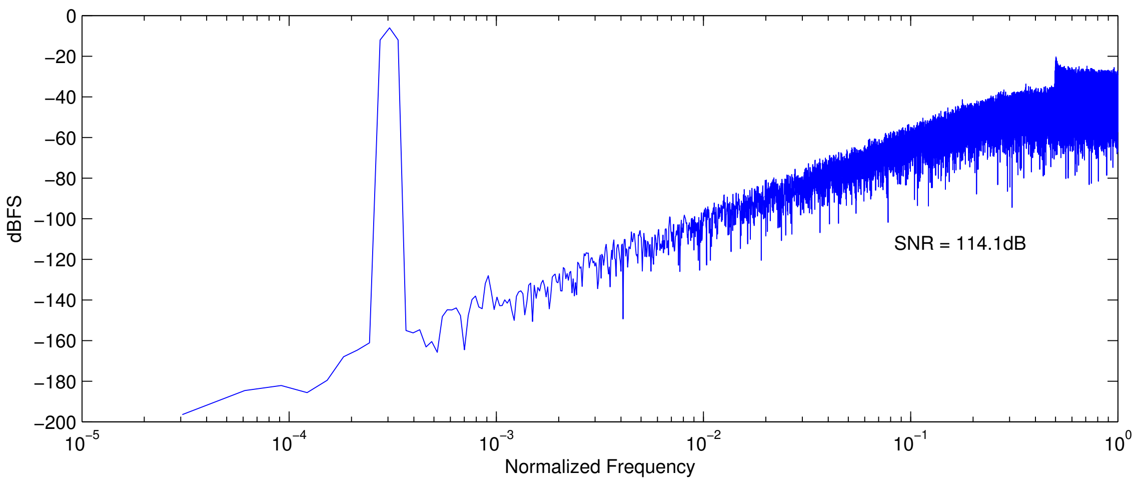


Figure 7.34 – Delta-Sigma modulator simulation.

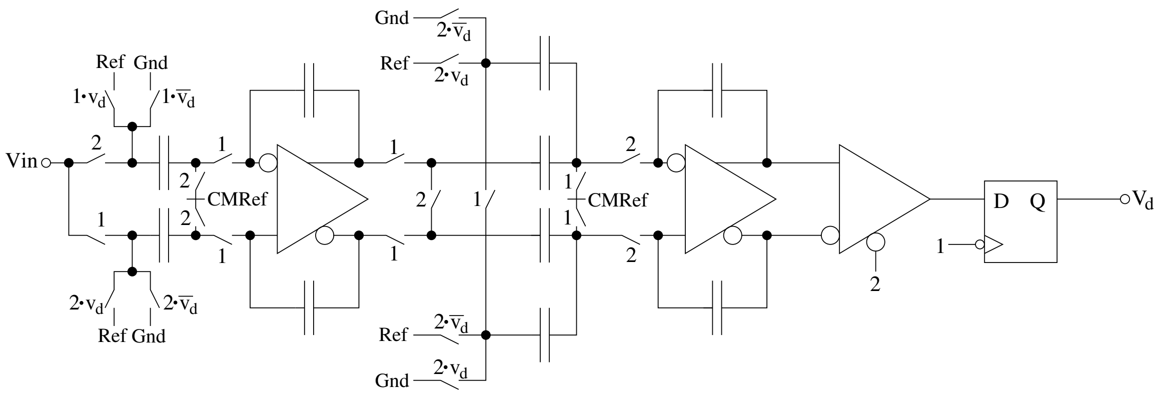


Figure 7.35 – Delta-Sigma switched-capacitor modulator implementation

information can be found in [88]. The switched capacitor operates with two complementary non-overlapping clocks labeled '1' and '2' in the figure. Furthermore, some switches are conditionally controlled by the output V_d.

The single-bit output goes to a sinc^3 filter. A single-order sinc filter has an impulse response which is a constant value out to some value N and then zero. It is basically a running-average whose output is given by

$$w(n) = \frac{1}{N}\sum_{i=0}^{N-1} v(n-i) \tag{7.25}$$

where v is the input and N is in the length of the impulse response. The transfer function is

$$H_1(z) = \frac{1}{N}\frac{1-z^{-N}}{1-z^{-1}}. \tag{7.26}$$

The frequency response of a sinc filter is given by

$$H_1(e^{j2\pi f}) = \frac{\mathrm{sinc}(Nf)}{\mathrm{sinc}(f)} \tag{7.27}$$

which has nulls around f_s/N, $2f_s/N$, $3f_s/N$ etc. If we choose N to be the oversampling ratio (OSR) then the nulls will correspond to the frequencies which get folded back to DC after downsampling. This is very good as it decreases the noise around the low-frequency signals of interest.

A standard FIR filter is a much better filter, however the sinc filter can be realized very easily. A sinc filter followed by a downsample by N is simply an accumulator which counts the number of ones in the input stream for N cycles. After N cycles, the value in the accumulator is the desired output sample and the accumulator is reset to begin counting anew. This compares to an FIR filter implementation with many multiply-accumulate operations required.

Generally a single order sinc filter is not good enough for the filter. Instead a sinc^{m+1} filter is used where m is the order of the modulator. In this case, this means a third order filter was used. The transfer function of a third order filter is simply Equation 7.26 cubed.

Higher order sinc filters can be created by cascading standard sinc filters together, however a simple accumulator structure cannot be used. Instead a simple architecture is shown in Figure 7.36 where the numerator of the transfer function is realized by three accumula-

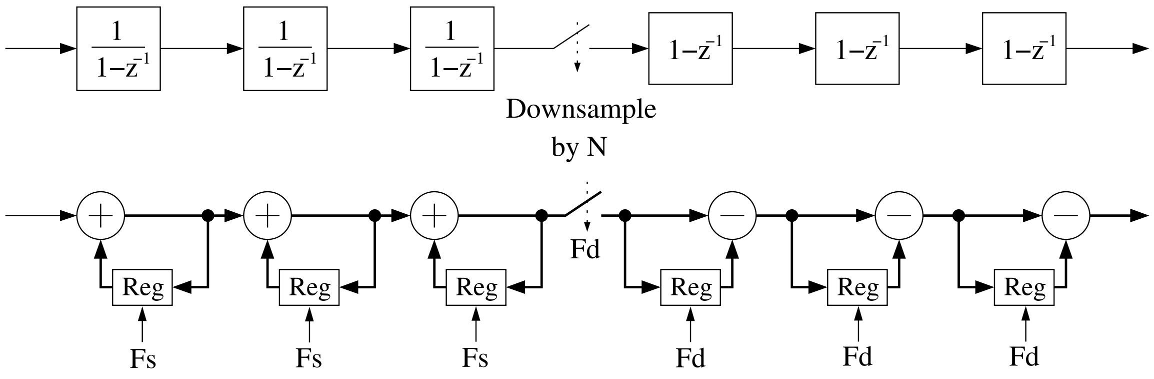


Figure 7.36 – Sinc^3 filter architecture.

tors followed by the downsample and then the denominator is implemented by three differentiators [89]. Figure 7.36 shows the concept first followed by a naïve implementation.

A more hardware efficient implementation can be achieved by combining all the three adders and three subtractors in the circuit to one adder/subtractor which is multiplexed. The datapath of the implemented sinc^3 filter is shown in Figure 7.37.

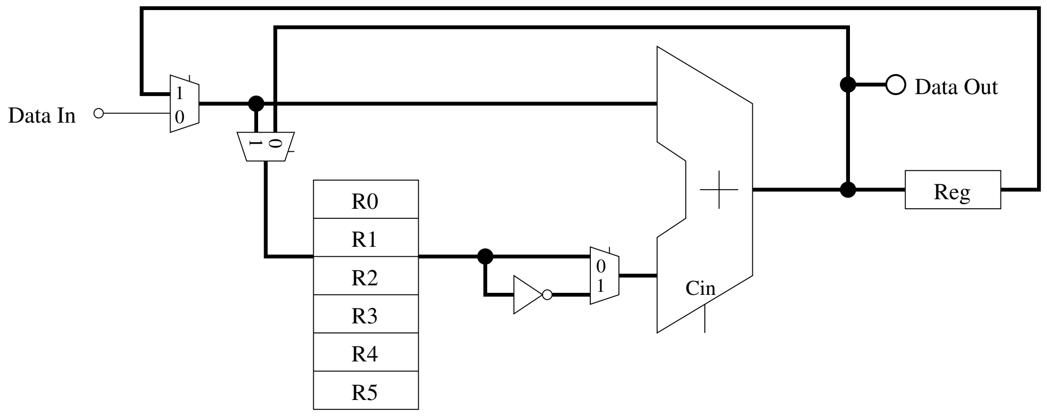


Figure 7.37 – Sinc^3 filter implemented datapath.

In this implementation the six registers are stored in a register file and are manipulated with the single adder/subtractor. The digital clock frequency must be 6 times faster than the incoming data rate, or 6 MHz, to process the data fast enough.

On most delta-sigma converters, another filter, usually an FIR filter, is required after the sinc filter to further filter out the high-frequency noise. In this case, it would take the 8 kHz words produced by the sinc^3 filter and filter and downsample them to 2 kHz. This post-processing step may be implemented in software if desired, however the performance directly from the sinc^3 filter is acceptable in most cases because of the very high OSR.

7.4 EFuse

The chip implements 64 bits of programmable non-volatile fuse-based memory. This memory consists of small polysilicon wires which can be selectively blown by inducing large currents in them. They are read by attempting to force a current through them and sensing the voltage. Once blown, it cannot be reversed. The 64-bits can be useful for programming unique identifiers to each silicon die for encryption, authentication, or other reasons. One bit, bit 63, is a lock bit. Once that bit is blown, the hardware does not allow further programming of the remaining EFUSE bits, thereby irreversibly locking the data. The EFUSE operates on the 3.3 V I/O supply. When programming the EFUSE, a supply voltage of at least 3.3 V is required. IBM recommends a supply voltage of up to 5V. This will stress the I/O transistors, so prolonged operation at this supply voltage is not recommended. Blowing a single EFUSE can require up to 15 mA of current, so if programming many fuses, it may be best to do them sequentially to minimize the strain on the power supply. Reading the EFUSE requires up to 400 μA per fuse. This power is consumed whenever the read enable bits in the `EFUSE_CR` are active.

Any read or programming of the EFUSE registers first requires the mode be setup in the `EFUSE_CR` register. To program the EFUSE, the following sequence should be followed:

1. Enable writing in the `EFUSE_CR` for the associated EFUSE register.
2. Write ones to the desired bits to program in the `EFUSEx` register.
3. Wait 1 ms to allow time for the fuse to blow.
4. Write all zeros back to the `EFUSEx` register.
5. Repeat steps 2-4 for any further bits to program in this EFUSE register.
6. Clear the write enable bits in the `EFUSE_CR` register.

When enabling writing in the `EFUSE_CR` register, the hardware first reads the EFUSE bit 63 to see if it is programmed. If it is, the write enable bit in the `EFUSE_CR` register will not be set, and writes to the EFUSE registers will have no effect. The status of this write enable bit can be read to check if it is possible to write the EFUSE registers.

7.5 Biases and References

The chip includes an on-chip bandgap voltage reference and constant-g_m bias voltage generator. The bandgap reference operates on the 3.3 V buck control supply. The internal bandgap reference operates at 1.193 V. From this the 1 V reference, 900 mV reference, 200 mV reference, current reference, and a current which is proportional to absolute temperature are generated. The current proportional to absolute temperature is used in the on-chip temperature sensor. The current reference is used by many DACs throughout the chip. The 200 mV reference is used by the DCO, while the 900 mV reference is used as the common mode level for the differential circuits in the RF transceivers. Finally, the 1 V reference is used by the buck regulators and the ADCs. The reference voltages and currents are all accessible from outside the chip for debugging purposes.

The internal reference and bias generators are digitally adjustable to compensate for process variations. The adjustments to these circuits can be made via the reference calibrate register. Increasing the bias generator value will generally speed up the internal analog circuits and make them consume more power. Changing the bandgap voltage reference will change all the other voltage and current references. The current reference can also be calibrated independently of the voltage references.

The bias generator produces four bias signals which are used in almost all low-power analog sections of the chip. These bias voltages are buffered with unity-gain op-amps, and driven onto external pins. The op-amps are designed to drive large capacitive loads, and require at least 2.2 nF of external capacitance to be stable. 10 nF or more is recommended. The capacitance can be increased without limit to decrease the noise bandwidth of the system. Disabling the bias generator via the global bias enable bit in the system control register will three-state these lines. External biases can then be applied.

The voltage references are similar. The 1 V and 900 mV references require at least 10 nF external capacitors to be stable, with 100 nF recommended. The 200 mV reference on the other hand will become unstable with too much external capacitance. The external current reference pin can either drive the internal reference to an external load, or accept an external current reference to use for the chip. All these settings are controlled via the respective bits in the system control register.

7.6 Conclusion

This chapter presented some of the supporting circuitry which completes the wireless sensor network node. An extensive clock system was presented which supports a variety of clocks from low-power high-jitter clocks to the high-power but low-jitter PLL. These clocks are critical to allowing the minimal possible system power consumption. This chapter also presented a buck power supply with very high efficiency at light loads and supporting dynamic voltage scaling. This is critical to achieving low power consumption in the digital section. Finally a number of other systems were presented including a $\Delta\Sigma$ ADC for high-precision voltage measurements, the bias and reference generation circuits, and the experimental EFUSE module.

Part III

Results

Chapter 8

Test Setup

A number of test boards were designed to test the various versions of the chips. Generally, the test boards became smaller and simpler since the chip generations progressed as more functionality was integrated into the chip with each generation. This chapter presents the individual chip revisions in more detail and discusses the test setup for each chip. Table 8.1 shows a short summary of the four generations of chip designs.

Functional tests were performed on the important parts of each design. The designs were not rigorously tested, but instead the parts which were required for the desired functionality were tested and characterized, while other parts of the design may have remained somewhat untested. This is particularly true in later designs when more features were added, many of which were not required for the desired functionality. These extra features only received minimal testing.

Section 8.1 presents the first 0.35 μm chip while sections 8.2-8.4 present the second through fourth chips. Finally, section 8.5 presents the software which was used on the test boards. This is a *Forth* system which is an interpretative programming language in which functions can be compiled by the system at run-time. It is exceedingly useful when debugging hardware as it provides an easy interface to read and write hardware registers and immediately see the results without having to re-compile the code and re-program the device. Though it is not an operating system, it supplies a type of shell where commands can be entered and executed.

Table 8.1 – Chip Comparison

Description:	First:	Second:	Third:	Fourth:
Year:	2007	2007	2008	2009
Process:	0.35 μm	0.18 μm	0.18 μm	0.18 μm
Area: (mm^2)	10	9	9	13.5
Processor:	MSP430	MSP430	MSP430	MSP430X
Proc. Freq.: (MHz)	20	40	60	55
RAM: (kB)	7.5	48	48	96
ROM: (kB)	0.5	0.5	2	4
CSA:	×		×	×
Diode Bias:			×	×
Narrowband Rx:		×	×	×
Narrowband Tx:		part	part	full
UWB Rx:			×	×
UWB Tx:	×	×	×	×
DVS:				×
Clock System:			×	×
PLL:				×

8.1 First Chip

The first Austriamicrosystems 0.35 μm chip contained the processor, serial peripherals, timers, manually designed SRAM and ROM, and the low-power front end. The architecture of the chip and the testbed used to test it are shown in Figure 8.1. The testbed includes the interface to the detection diodes, the external serial flash, the computer interface via a USB to serial converter, and an external clock generator. It is not until the third chip that the clock generator is integrated.

The die photo of the first chip with markups showing the individual functional parts is shown in Figure 8.2(a). The final die photo is shown in Figure 8.2(b).

The ARM microcontroller and the external SRAM shown in the testbench architecture in Figure 8.1 are used to fix a bug related to the internal SRAM not working. Due to a timing race condition in the SRAM, the internal SRAM did not always work properly. This

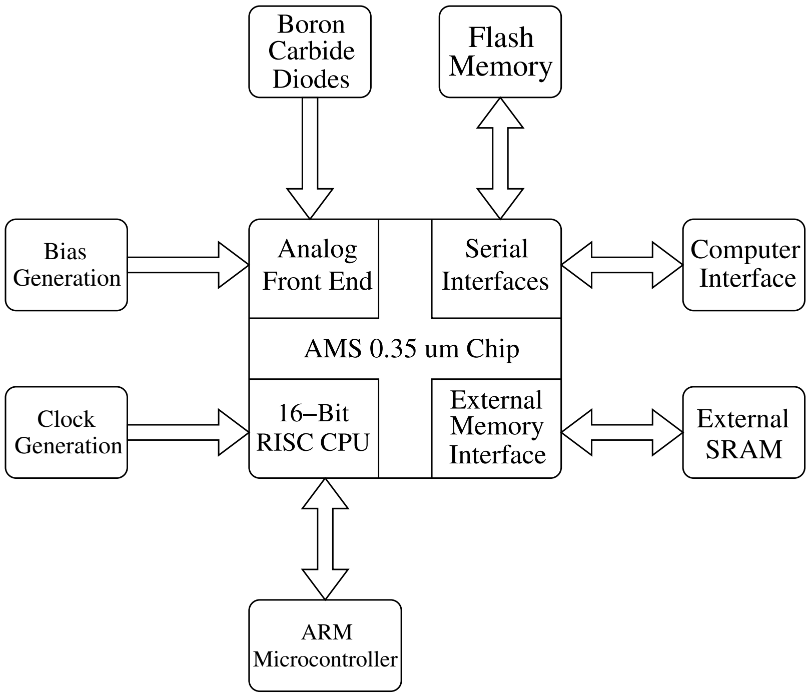
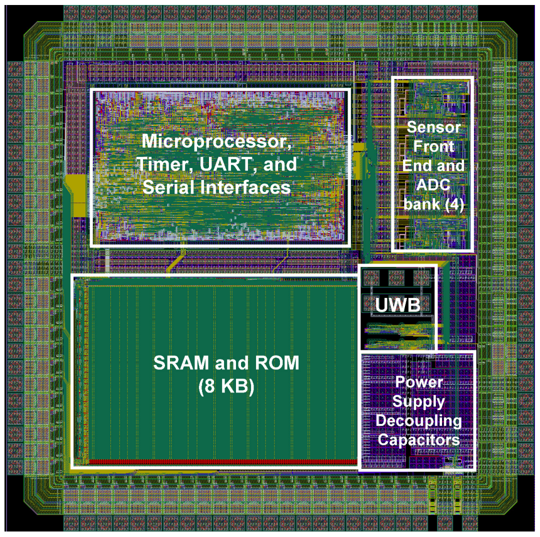
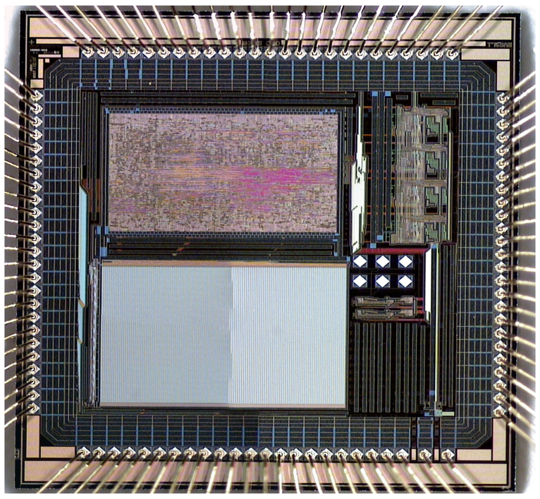


Figure 8.1 – First AMS 0.35 μm chip and testbed architecture


Microprocessor, Timer, UART, and Serial Interfaces
Sensor Front End and ADC bank (4)
UWB
SRAM and ROM (8 KB)
Power Supply Decoupling Capacitors

(a) First chip layout with markups



(b) First chip die photo

Figure 8.2 – First chip layout and die photo

type of problem was anticipated at the design stage, and an external memory interface was added to the chip so that external memory could be used in place of the internal one should it fail. However, using external SRAM also overrode the internal boot ROM, and without that the processor could not copy the data from the external serial flash to the RAM for execution. Therefore an ARM processor was added with the sole purpose of loading the program from the flash memory to the SRAM when the processor was reset. After this, the ARM processor did nothing more.

The 0.35 μm process used means a supply voltage of 3.3 V was used. No dynamic voltage scaling was supported with this chip. The chip is packaged in 100-pin leadless package (QFN-100) with a large solder pad on the back. The lead spacing is 0.4 mm. The PCB is a four-layer PCB with the internal layers used for power planes. A photo of the first chip testbed is shown in Figure 8.3. The chip is clearly visible at the center left.

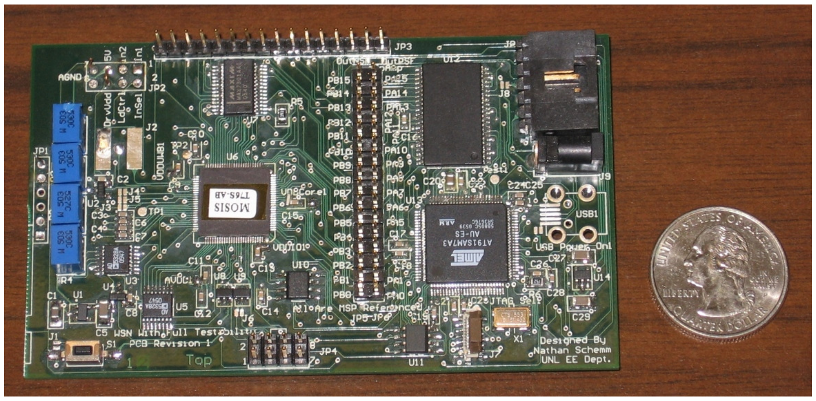


Figure 8.3 – First AMS 0.35 μm chip testbed

The chip also includes 16 12-bit DACs which can be used to supplement the internal bias generators. They can be used to override internal bias voltages if necessary. They are also used to supply the feedback voltage which adjusts the resistor in the CSA front end, as well as the noise threshold voltage. The internal DACs to supply these voltages were later added in the third chip.

This version used an external serial flash programmer to program the serial flash memory. It was not until the third chip that in internal one was added in the bootcode.

Besides the SRAM bug, there was also a bug relating to the UART. This was mainly caused by lack of experience in digital design and not enough attention was paid to races and glitches in this design. Therefore the UART had a glitch which caused it to malfunction. In this chip the UART was emulated in software as a workaround. The CSA generally worked well, with only the previously mentioned problem of poor DNL performance caused by a faulty comparator design in the ADC.

Only one board was fully populated with the ARM processor, though two other limited systems were built at first which did not have the ARM processor. Therefore only one fully-functioning test board exists.

8.2 Second Chip

The second chip was somewhat rushed because it was a free fabrication run with a hard deadline attached to it. Therefore not all functionality from the first chip was included in this fabrication. Namely, the CSA and front end were not included. However the UWB transmitter and narrowband receiver were included. Furthermore the digital design experience from designing the first chip allowed a much better digital section to be included. The CPU was re-designed for this version to take advantage of the synchronous memory generators instead of using the asynchronous memory in the previous chip. The maximum clock frequency was increased from about 20 MHz in the previous chip to about 40 MHz in this chip through a combination of design improvements, digital synthesis flow improvements, and technology scaling.

Figure 8.4 shows the chip and testbed architecture. The most complicated part of the testbench is the RF section of the narrowband transmitter which consists of a frequency

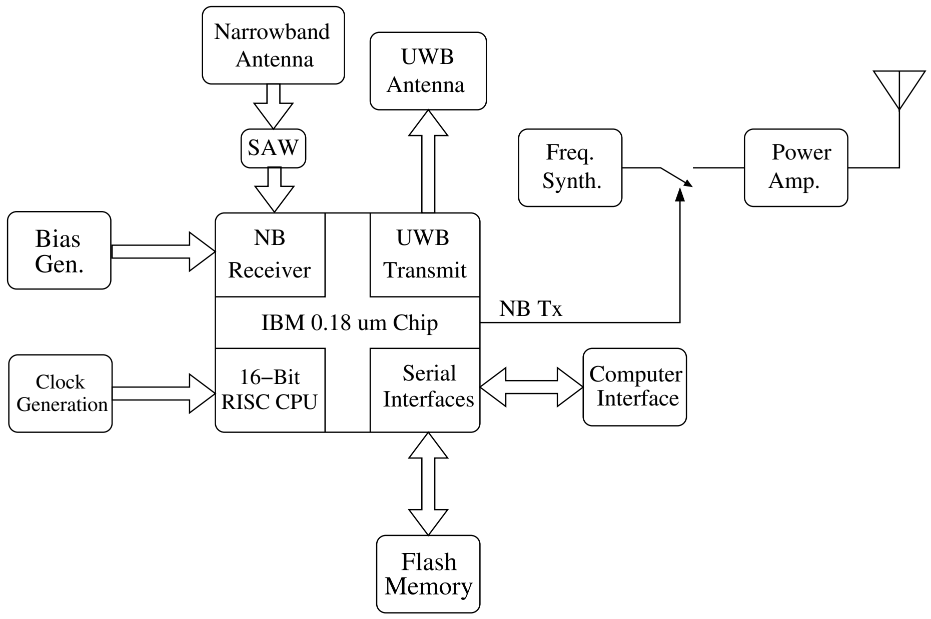


Figure 8.4 – Second IBM 0.18 μm chip and testbed architecture

synthesizer, modulator (an RF switch), and a power amplifier. This does not include external RAM, as memory generators are used for internal RAM and it is assumed these are good.

The die photo of this chip is shown in Figure 8.5. The digital section is the large rectangle, with the narrowband receiver to the right, and the UWB transmitter on top. This chip has much unused space on it.

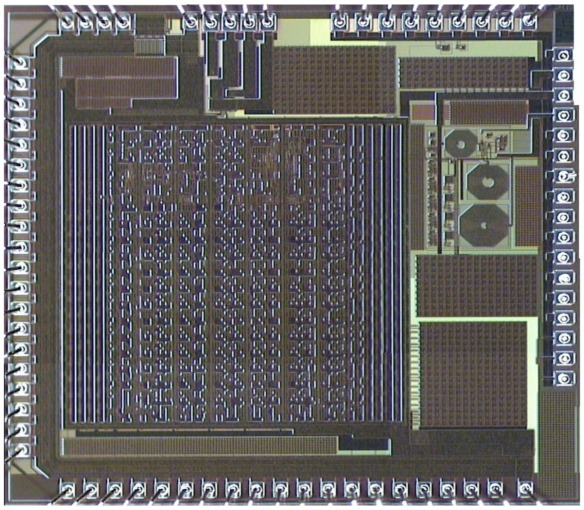


Figure 8.5 – Second IBM 0.18 μm die photo

As before, the test board contains an external clock generator as well as an external bias generator which was used for a couple control voltages inside the narrowband receiver. A custom four-layer board was designed and populated. The package for this chip is an 80 pin 0.5 mm pitch leadless package which looks similar to the previous one. The test board is shown in Figure 8.6. Note the footprint for a ARM processor which could have been used

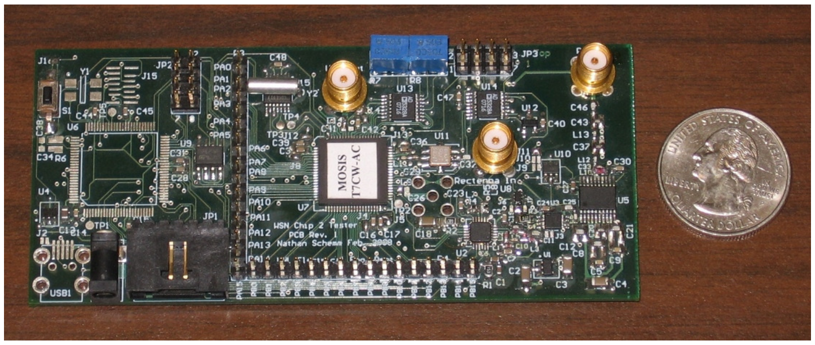


Figure 8.6 – Second chip test board

to debug the processor core if it had not worked. Happily, the processor core worked and the ARM processor was not needed so that section of the board was not populated. This unpopulated section is visible in the left part of Figure 8.6.

Two test boards were populated and were able to achieve bidirectional wireless narrowband communication between the boards.

The test boards do not natively support dynamic voltage scaling, however some tests were done in the lab using manual control of the supply voltage to determine if the ARM memory IP would operate at lower voltages, and to determine the potential power savings. After it was determined this could save very large amounts of power, the DVS-based power supply was pursued for the next chip. Up until this point it was not known whether the ARM memory would operate at lower voltages because it was only specified to operate from 1.65-1.9 V. These tests determined it would readily operate down to 900 mV.

The second chip had a nearly-catastrophic error in the layout of the pad frame. The pads which were used were from the ARM IP library, and no layout level view was available for them. Instead the only available view was a dummy blockage view which consists of large rectangles of dummy metal layers which indicate that there is metal in that region in the actual layout. Therefore, the pads as well as the pad spacers look like large featureless rectangles when laying them out, with no visible sign as to their rotation. One must manually check the properties of the instance to determine their rotation. At the last minute two pads were deleted and spacers were inserted in their stead, however those spacers erroneously were flipped by 180 degrees. This was not visible from the layout, and no automatic checker could catch the problem because the inserted spacers do not have any pins which connect to any nets (other than power and ground, but these were not explicitly available as pins in the layout view).

This error was not found until after the design was mid-fab. However, due to an extraordinary stroke of luck, this error did not cause the power and ground wires—which run close together in the pad frame—to short, and did not affect the operation of the chip at all. The design was tested with no further consequences of this error. This error is visible on the layout in Figure 8.5 in the top left corner. It is seen between the fourth and fifth pad connections from the left on the top row of pads.

This was one of the reasons why for future chips the pads were manually designed instead of using the ARM library. Another important reason was to make it possible to use sufficient isolation between the pad I/O supply, digital supply, analog supply, and RF supply. The existing ARM library pads did not have the required isolation structures to support robust isolation between the supplies, so pads were designed which did support this. (Actually, the ARM library pads may have had the isolation structures, but without the layout views, which ARM does not share with universities, it is impossible to know how to interface with them.)

8.3 Third Chip

The third chip was a vast increase in complexity and functionality over any previous chips. At this point the chip transformed from being part of a system but still requiring much support circuitry to begin the complete system with only minimal, mostly passive, support components necessary.

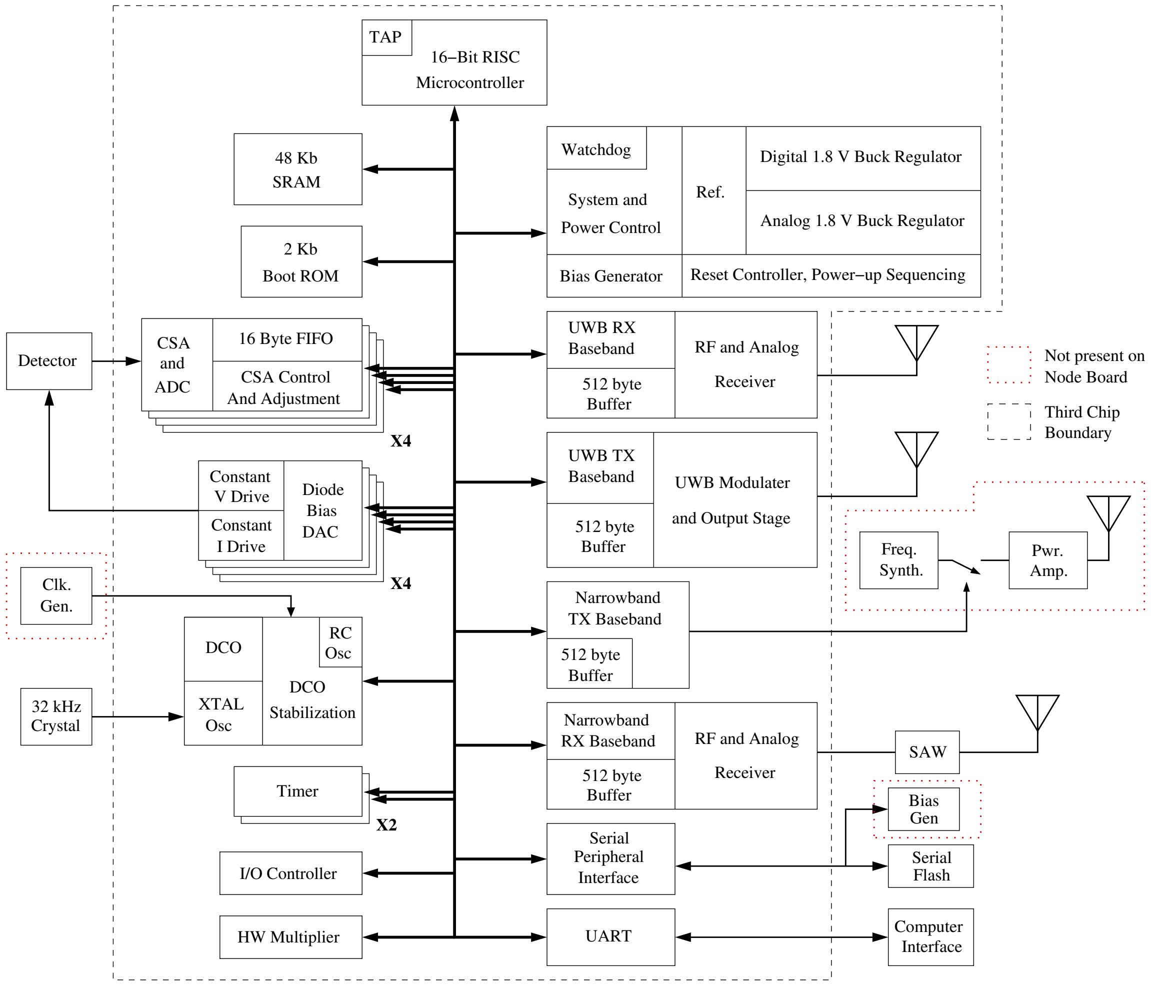


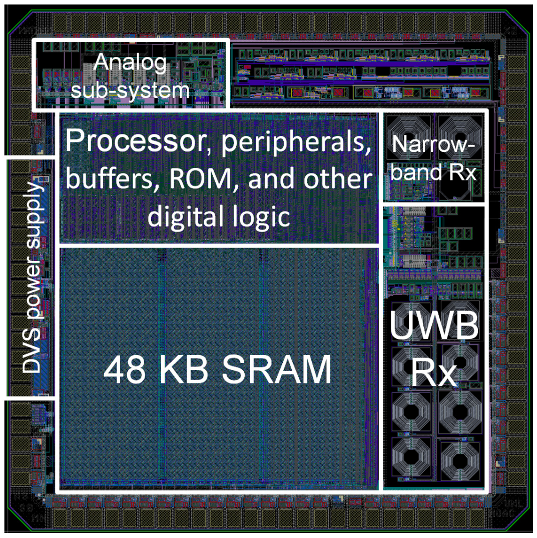
Figure 8.7 – Third IBM 0.18 μm chip and testbench architecture

For this chip some incremental optimizations were performed on the CPU to increase the processor clock speed to 60 MHz, and more peripherals were added. The low-power front ends were included, but with much lower power consumption than on the AMS 0.35 μm chip. A clock generator with a crystal oscillator and digitally-controlled oscillator was added to make the microcontroller a complete system. The narrowband receiver was re-designed to its final form, and the UWB receiver was implemented. Finally the DVS enabled buck power supplies were added.

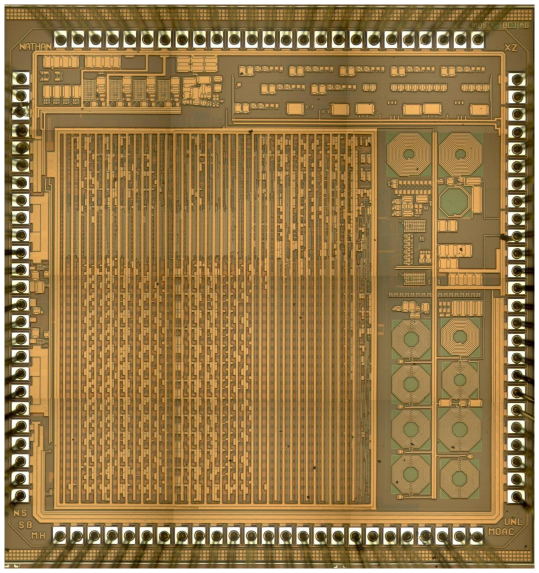
The architecture of the third chip combined with the testbenches is shown in Figure 8.7. The figure shows the internal architecture of the third chip as well as external components on the board. Two test boards were designed for the third chip. One, the full testing board has everything shown in Figure 8.7. The other one is a much smaller sensor node board which includes only the parts which are required for a sensor node. That board excludes some circuitry which is shown in Figure 8.7 inside a dotted red box.

For the complete board, all the antenna connections were brought out on SMA connectors, while for the node version an integrated PCB antenna was used as a combination UWB and narrowband antenna.

Figure 8.8(a) shows the layout of the chip with markups showing the important functional blocks. Figure 8.8(b) shows the final die photo of the chip.



(a) Third chip layout with markups



(b) Third chip die photo

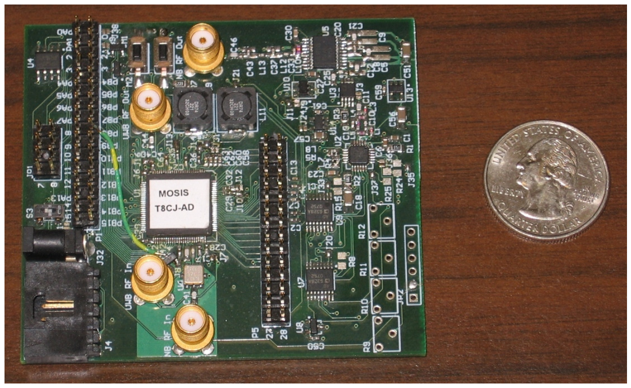
Figure 8.8 – Third chip layout and die photo

This chip includes a internal clock generator with a 32 kHz oscillator as well as a high-frequency digitally-controlled oscillator (DCO). The DCO's frequency can be stabilized using the crystal as a reference, however an external clock generator was also included on the full board in case the DCO had too much jitter for RF communication. This turned out to be the case, and this external oscillator was required to perform RF communication. This jitter is unavoidable with the low power consumption of the DCO. Therefore in the fourth chip the DCO was supplemented with a high-frequency crystal oscillator and low-jitter PLL which can be used for RF communication.

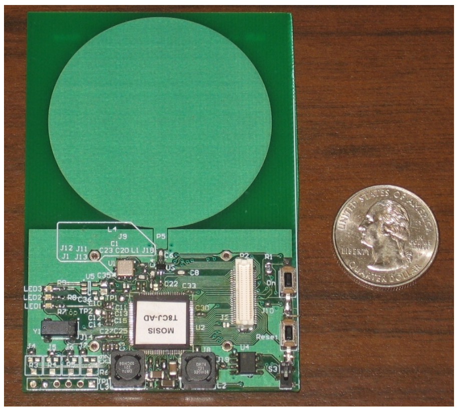
The full board also includes a bias generator, which was fully optional with this chip. Unlike the previous chips, all required bias and reference voltages could be generated internally through internal DACs and references, so this is not present on the node board. It is included on the full board for maximum flexibility during testing.

As on the previous board, the RF section of the narrowband transmitter is present on the board. This is a complicated part of the board and was not included for the node board.

Figure 8.9(a) shows the full testing board with all the circuitry shown in Figure 8.7. Two of these boards were populated and this is the main testing platform for this chip. Figure 8.9(b) is the node sensor board which has a smaller amount of circuitry on it. It does include a fine-pitch 50 pin connector which will be included on all future small testing boards. This connector has access to all the digital I/O pins of the processor as well as some of the important analog lines. This allows other expansion boards to be connected to add other functionality or this may be used to probe the wires.



(a) Third chip full testing board



(b) Third chip node testing board

Figure 8.9 – Two testing boards for the third chip

The small node board was not used much because without the clock generator it could to transmit and receive anything. Therefore most testing was carried out on the larger board. The large board is a four-layer board as before, while the small board is a two-layer board. The chip was packaged in a 100-pin 0.4 mm QFN package like the original AMS chip, which is the same footprint used for the fourth chip.

The third chip is the first chip to include a bootcode which can also function as a serial flash programmer. With this improvement, there is no requirement for an external dedicated serial flash programmer to be used to program the serial flash chip, instead the bootcode can perform this function.

8.4 Fourth Chip

The fourth and final chip is the culmination of all previous designs. In this chip the digital processor was upgraded to the MSP430X architecture to support a larger memory of 96 kB, however this decreased the maximum clock frequency slightly to 55 MHz. Many bugs in the previous chip were fixed, while some new ones were added! The integration of the RF PLL, RF modulator, and power amplifier allowed the narrowband transmitter to be fully integrated. With this improvement, there was little need for a large full-featured test board, and instead only small node-sized boards were used for testing.

The architecture of the fourth chip as well as the test board architecture is shown in Figure 8.10. A number of different boards were made with different varieties of the external circuits shown in the figure, but Figure 8.10 shows the final complete version.

Most parts of the system are integrated on the chip. There are a few power-supply related parts which are not shown. There is a 5 V to 3.3 V regulator on the board which is only powered when operating from the computer's USB port or other 5 V source. When operating from a battery, this is disconnected as it is not necessary. There is also a second 1.8 V linear regulator which can be used to supply the analog and RF 1.8 V supply in case the built-in supply is not clean enough. Finally, due to a bug in the bootcode of the chip, an external three-inverter ring oscillator is on the board to supply a clock to the processor at start-up. After start-up the ring oscillator is disabled as the internal clocks may then be used.

Figure 8.11(a) shows the layout of the chip with the main functional areas of the chip labeled. Figure 8.11(b) shows a die photo of the chip.

A large number of different test boards were designed and populated for this chip,

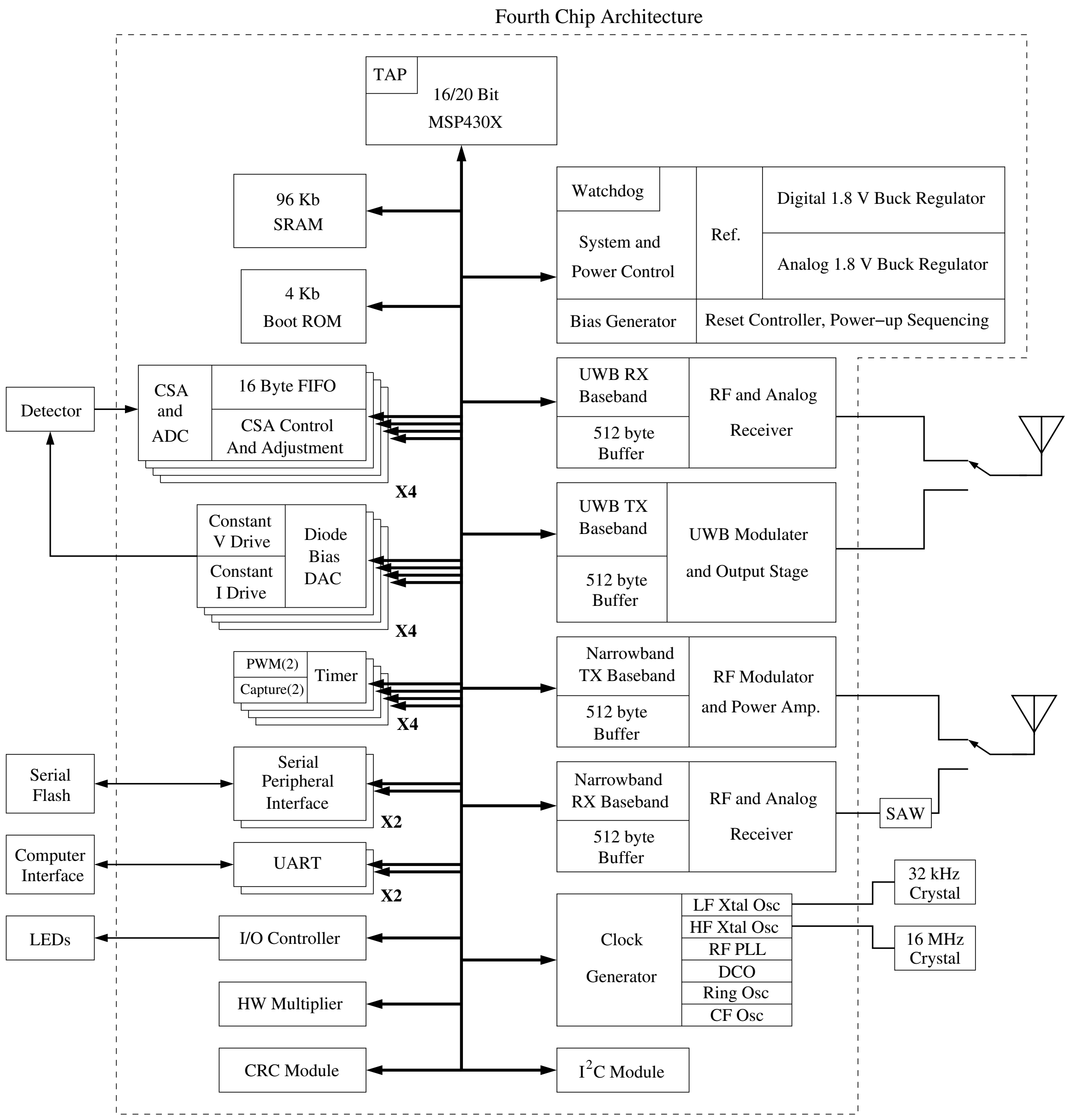
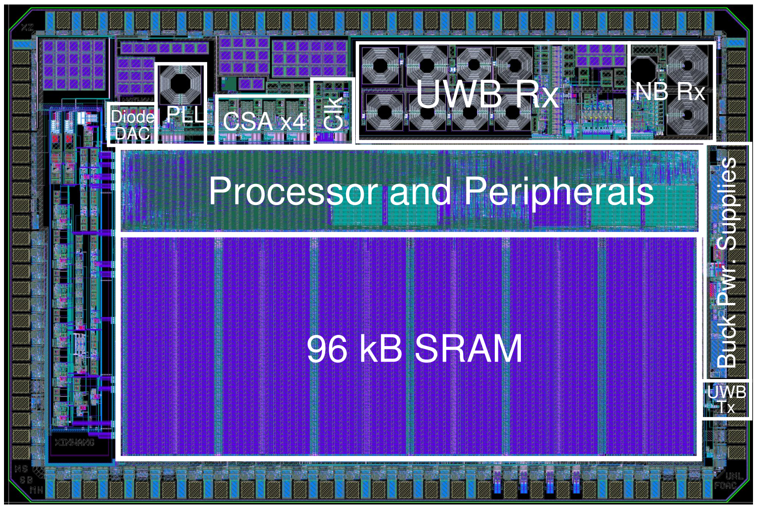


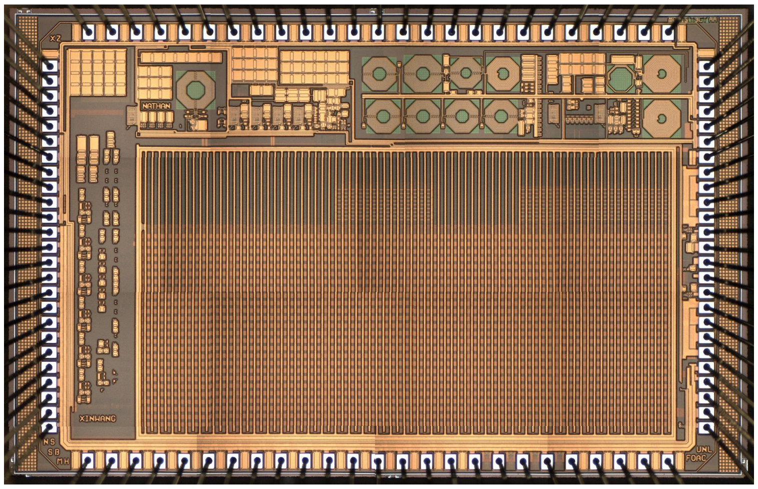
Figure 8.10 – Fourth IBM 0.18 μm chip and testbench architecture

however they all fall into one of four representative types. Those basic types are shown in Figure 8.12.

In Figure 8.12(a) the main full test board is shown. There are three versions of this board, with the first two only having slight differences from the final one. The one shown is the last version. A large number of this type of board was made. Figure 8.12(b) shows a version where the antenna is integrated with the PCB. There are two versions of this, the



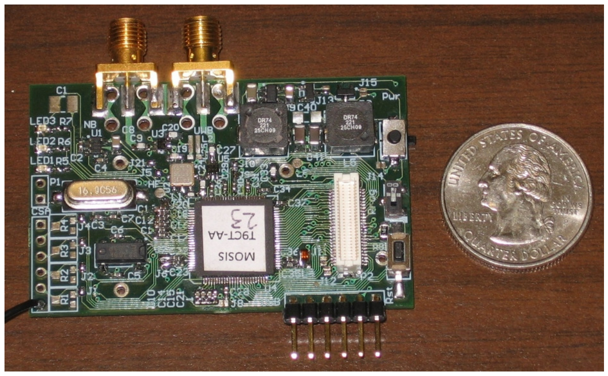
(a) Fourth chip layout with markups

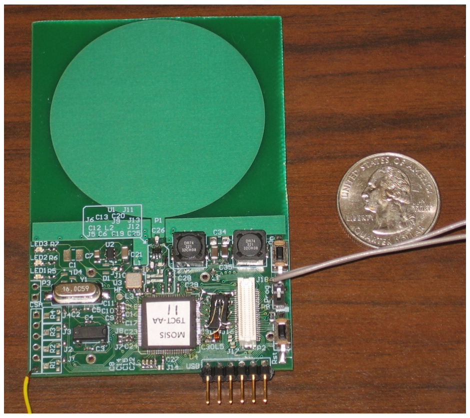


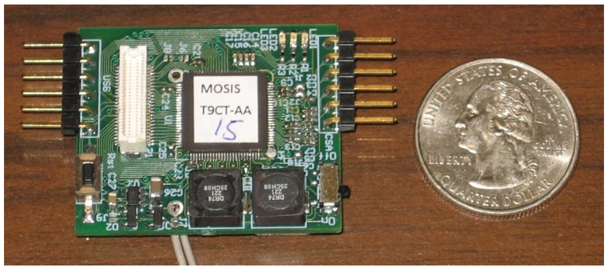
(b) Fourth chip die photo

Figure 8.11 – Fourth chip layout and die photo

one shown in the figure is the first one. The second version was never assembled, the PCB was made, however it was later decided to use the node style with the separate antennas, so these second-revision boards were not populated. Figure 8.12(c) shows a version of the board with no RF components present. This is a convenient board for testing the front end and doing data collection and processing, without the extra bulk associated with the RF components. Only one of this type of board was populated, however it was used for most of the standalone CSA testing as well as radiation detection. The final board style shown in Figure 8.12(d) was used to test numerous chips to determine which chips where functional


(a) Final test bench


(b) Test bench with integratedUWB antenna



(c) Board with no RF parts

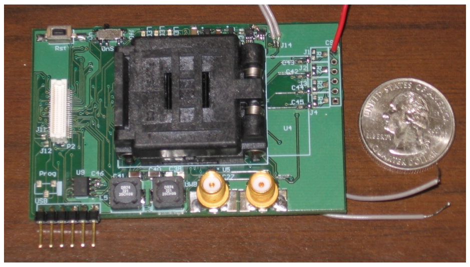

(d) Board with socketed chip

Figure 8.12 – A number of fourth chip test boards.

and which ones had the best performance and use those to populate the other boards. This board has schematics identical to Figure 8.12(a) but using an expensive socket for the chip which allows chips to be swapped at will. Only one of these boards was populated.

Each of the test boards in Figure 8.12 has the same 50-pin fine pitch connector. This connector allows a number of other boards to be connected to each board. One board is an expansion board which simply allows access to all the digital and analog lines in the 50-pin connector. This is invaluable for viewing the waveforms on the oscilloscope. Another possible expansion includes a board with a CC2420 commercial radio transceiver from TI installed. This allows the boards to communicate via the IEEE 802.15.4 standard. Finally, another board was made for a class project which included a CC2420 radio as well as three microphone channels. The expansion board and the CC2420 board are shown in the top part of Figure 8.13.

Figure 8.13 also shows the UWB antenna which can connect to the boards to allow UWB communication.

An array of test equipment was used to measure the performance of the chips. This

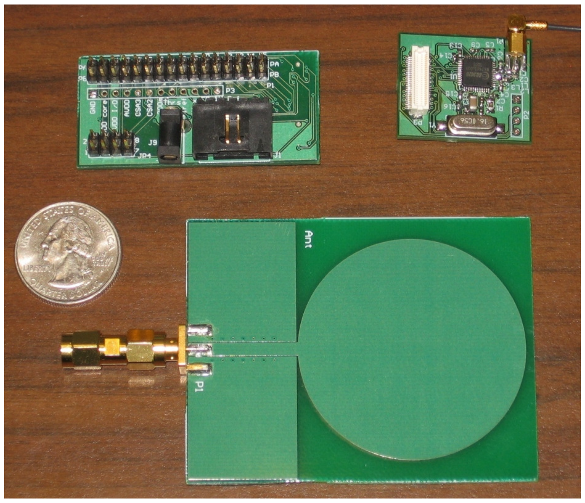


Figure 8.13 – Fourth chip test boards accessories

included a fast 24 GHz oscilloscope, 6 GHz spectrum analyzer, RF signal generator and many others. A picture of some of the equipment used is shown in Figure 8.14.

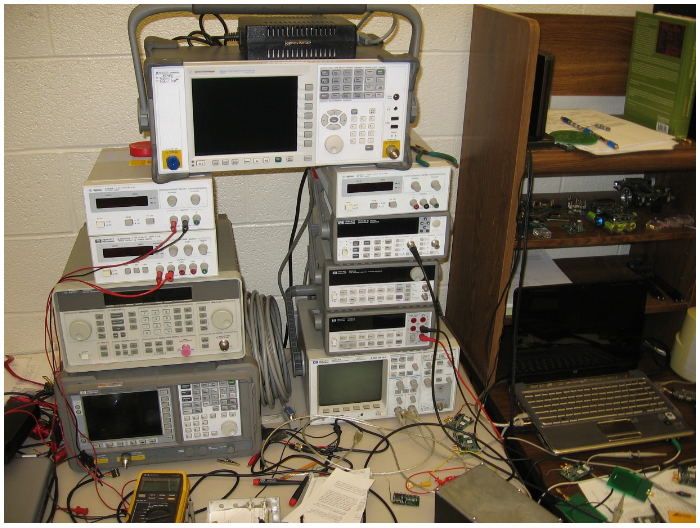


Figure 8.14 – Test equipment used in testing the chips.

8.5 *Forth* System

The software used on the boards is a small *Forth* system. *Forth* is a simple programming language which can operate in an interpretative mode, but can also act as a compiler to produce new functions at run time. Having a full programming language available at run-time to dynamically write functions or to simply change registers simplifies debugging as it reduces the need to constantly change and re-compile the firmware. The most common operation is to write a single or combination of registers at run-time and the results observed before possibly changing the registers again. Simple functions can also be written at run time to do more complex tasks. For example, the function to capture the INL and DNL data for the ADC was written and programmed at run time.

The *Forth* system grew from an extremely basic version present on the first chip test board to a much larger version with many debugging related commands for the final board. For the first chip, due to memory constrains, the complete *Forth* system fit in 7.5 kB of shared program and data RAM. This version had 59 built-in commands. The final *Forth* program occupied over 16 kB of code space and had over 100 commands. The added commands were all related to testing the chip. The *Forth* version implemented is only a subset of the widely used *Forth* programming language.

8.6 Conclusion

This chapter presented a brief overview of each of the chip revisions and the test boards which were made to test each revision. The boards were usually custom four-layer boards which were manually assembled. As the chips get more complicated, the testing boards get simpler as more functionality is implemented on the chip and less is required on the board. The final testing board for the fourth chip includes all the hardware necessary for a wireless sensor network mote in a small form-factor of 37.4×59 mm^2. Multiple boards were made of the final chip with differing capabilities. Each board was designed with testing in mind, so features like power supply jumpers for simple current measurement were included on all boards.

Chapter 9

CSA Results

Two distinct versions of the CSA and front end were tested, the one in the AMS 0.35 μm process and the one in the IBM 0.18 μm process. There are only minimal differences between the third and fourth chips with respect to the CSA front end.

This chapter is organized a follows: section 9.1 presents the testing results performed in the lab on the front end. This includes basic characterization of the front end. Section 9.2 presents tests performed with a GdHf neutron detection diode. Section 9.3 presents results when using a NaI scintillator based detector, while section 9.4 concludes the chapter.

9.1 Lab Test Results

To introduce a known charge pulse to the front end in the lab, a sawtooth wave was differentiated with a capacitor. The amplitude of the charge pulse is set by the sawtooth generator waveform size and the capacitor size. The charge pulse duration is set by the time duration of the negative-going edge of the sawtooth generator. This was 100 ns for the generator used. The time constant of the circuit was adjusted to match this input by adjusting the feedback voltage.

Figure 9.1 shows an oscilloscope capture of the CSA output with an 8.3 pC introduced charge and with a feedback capacitor setting of 15.2 pF. It also shows the output of the comparator that triggers the ADC on the rising edge. This scope capture is for the 0.18 μm chip, however it looks similar for the 0.35 μm chip.

Figure 9.2 shows the measured value of the CSA peak versus the input charge content. Curves are given for four different gain settings. As can be seen, the CSA stays linear for over 1 V of output swing range. The measured gain is very close to the ideal gain. If

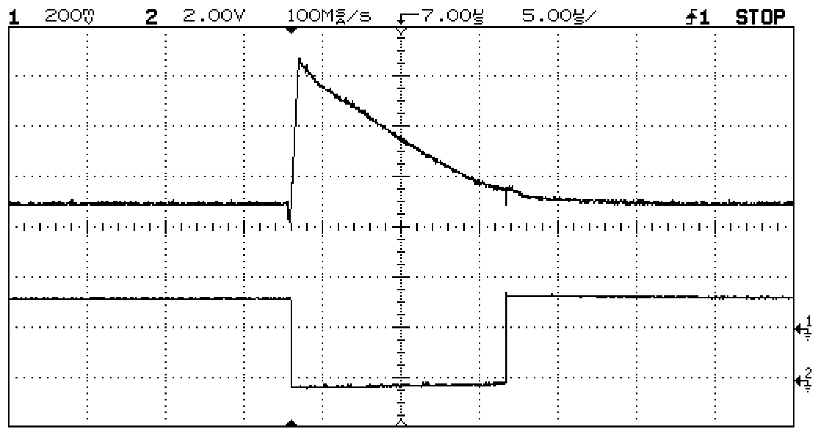


Figure 9.1 – Measured CSA output and the ADC trigger line.

a linear curve is fitted to the data in Figure 9.2, the reciprocal of the slope should be the feedback capacitance. This is given in Table 9.1.

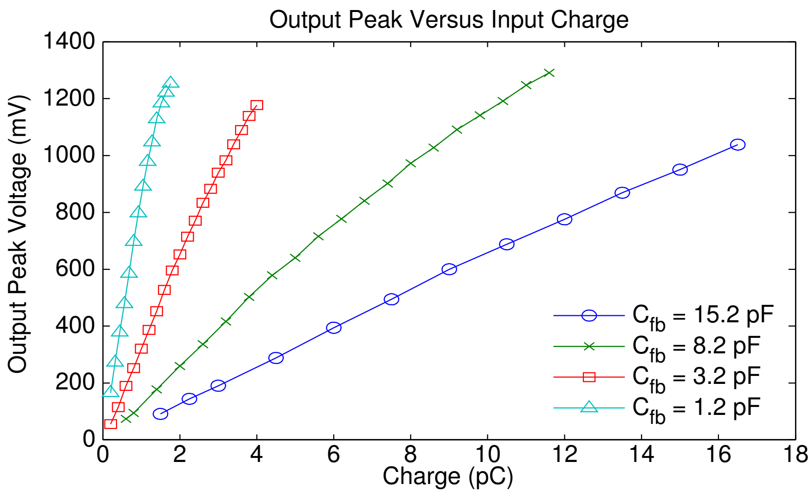


Figure 9.2 – CSA output peak versus charge input.

Table 9.1 – Measured Gain

C_{fb} (pF)	1.2	3.2	8.2	15.2
Meas. 1/Gain (pF)	1.26	3.23	8.44	15.57

The DNL and INL measurements of the integrated low-power 8-bit, successive approximation, charge redistribution type ADC are shown in Figure 9.3. There are no missing codes, with the worst DNL value of 0.36 LSB. The INL is within 0.5 LSB over the entire range. This is for third and fourth chips without the defective comparator.

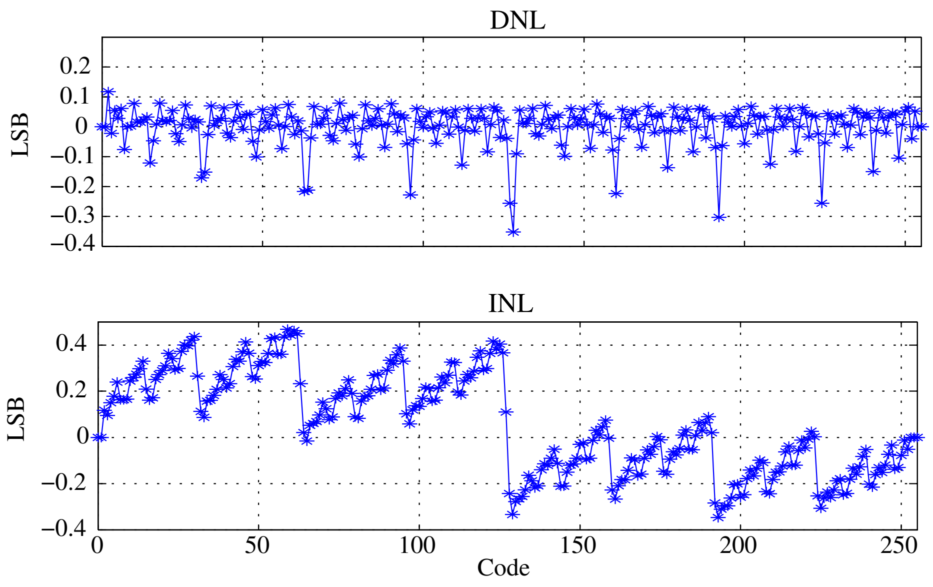


Figure 9.3 – ADC DNL and INL measurements.

The pulse height spectra gathering application has been developed within the *Forth* application on the chip. With an 8-bit ADC resolution, the application uses 256 separate counters to keep track of how many events occurred with each pulse height. A plot of those counters produces a histogram of the pulse heights. Using an arbitrary waveform generator, a sequence of pulses with three distinct amplitudes with different arrival rates were applied to the analog front end of the computational sensor chip. Figure 9.4 displays a sample pulse height spectra with 256 bins generated by the *Forth* system running on the chip.

From the measured results, the CSA, peak detector, comparator, on-chip bias circuits and configuration DACs consume a total of 4 μW in the 0.18 μm chips and 15.3 μW in the 0.35 μW chip. This is the minimum power consumption when no events are occurring. The ADC has negligible power consumption when there are no detection events. It consumes 75 pJ per conversion for the 0.18 μm chip and 690 pJ in the 0.35 μm chip. Other pertinent specifications of the design are shown in Table 9.2.

A more detailed breakdown of the measured power consumption of the CSA and diode DACs is shown in Table 9.3. Unless otherwise stated, the diode driver supply is 3 V, the analog supply is 1.8 V, and the digital supply is 1.2 V.

Table 9.4 displays a comparison of this work (using the 0.18 μm chip) with related recent work in the field. The power consumption figures listed in the table vary from including just the CSA, to some including a more complete front end. However, the power consumption for this work consists of the complete front end. The sensitivity of the front-

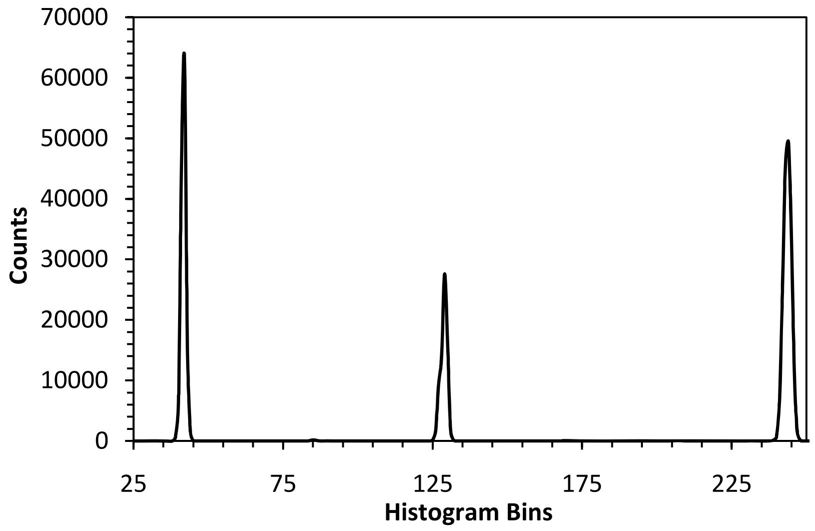


Figure 9.4 – A sample histogram acquired with a function generator producing three distinct charge pulse sizes

Table 9.2 – CSA Specifications

Description:	**0.18 μm Chips**	**0.35 μm Chip**
CSA Power Consumption:	$4+0.075R\,\mu$W*	$15.3+0.69R\,\mu$W*
Diode Bias Power Consumption:	1.3 μW	N/A
Area:	0.045 mm^2	0.076 mm^2
CSA Feedback Capacitance:	0.2 - 15.2 pF in 1 pF increments	
Supply Voltage:	1.8 V	3.3 V
Equivalent noise charge, detector disconnected, C_{fb}=0.2 pF:	525 e$^-$	Not Meas.
ADC Bits:	8 Bits	
Max Particle Rate:	4×10^5	3×10^5

*R is particle rate in thousands particles/sec

end is set by the input referred noise, as measured by the equivalent noise charge (ENC). All ENC numbers are listed with the detector disconnected. The non-linearity is computed by fitting a straight line to the data in Figure 9.2 and dividing the largest residual by the maximum signal range. The main goal of this design effort, to minimize power consumption, has been achieved when compared to other work. It is difficult to draw conclusions from such a table however, as each CSA is generally tailored to specific applications, and the design tradeoffs are different for each application.

Table 9.3 – Front End Detailed Power Measurements

CSA Quiescent Power Consumption:	4 μW
CSA Counts	0.075-0.4* $\frac{\mu J}{K_{counts}}$
Histogram Event Processing:	7.2 $\frac{\mu J}{K_{counts}}$
Diode Bias Power Consumption, Constant I drive:	1.3 μW
Diode Bias Power Consumption, Constant V drive:	3.6 μW

*Power varies depending on the size of input pulse and integrating capacitor size.

Table 9.4 – Comparison with Related Work

Ref:	Process (μm)	Power Supply (V)	ENC (e^-)	Power (mW)	Non-linearity (%)
[33]	0.35	3.3	254	0.165	<0.57
[34]	0.8	4	87	4	
[35]	0.5	3.3	33	5.4	
[36]	0.25	2	<1500	1.5	
[37]	0.5	3.3	400	12.4	
[38]	0.35	3.3	370	2	<1
[39]	0.5		800	2.6	0.8
This Work	0.18	1.8	525	0.004	1.7

9.2 GdHf Diode Results

Neutron field tests were conducted with a paraffin moderated 5.2 curie plutonium-beryllium neutron source which provided 2.2 $\times$ 10^4 thermal to epithermal neutrons/cm^2-second, as calibrated by foil activation methods. Figure 9.5 displays the actual test setup used in the testing of the computational sensor.

As the front end CSA design is well suited for semiconductor neutron detectors [62], heterojunction diodes composed of mixed gadolinium and hafnium oxide ($GdHfO_2$) on silicon have been utilized as the neutron sensing devices in this work [90]. In particular, the diodes that were utilized in the field tests are constructed with slightly less than 100 nm oxides of 15% gadolinium and 75% hafnium oxide films that were deposited on n-type

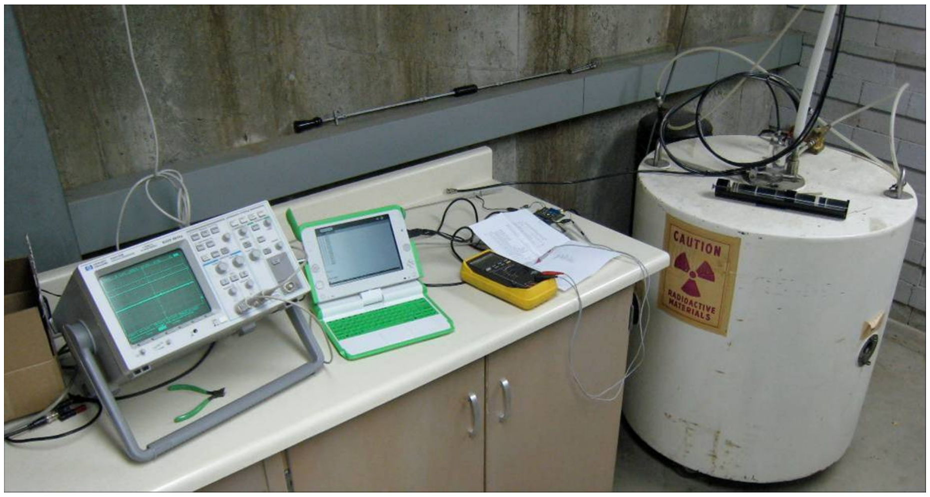


Figure 9.5 – Field test setup

single crystal Si(100) substrates using pulsed laser deposition [90, 91].

The absorption of the neutron leaves the ^{158}Gd in an excited state that releases energy through emission of high energy gamma rays, low-energy gamma rays, X-rays, internal conversion (IC) and Auger Coster-Kronig (ACK) conversion electrons as:

$$^{157}\mathrm{Gd}(\mathrm{n},\gamma) \;\rightarrow\; ^{158}\mathrm{Gd}+\gamma+\mathrm{X{-}rays}+\mathrm{IC\ e^-}+\mathrm{ACK\ e^-} \tag{9.1}$$

During a neutron capture event, the conversion electrons generate charge pulses in the heterojunction diodes which are converted to voltage by the front end section of the computational sensor node.

Typical charge sensitive amplifier output waveforms captured over a period of 1 msec by the sensor system are shown in Figure 9.6. The waveforms show several large spikes resulting from neutron detection at a charge-to-voltage gain setting of 833 μV/fC, corresponding to 1.2 pF of integration capacitance with an amplifier time constant setting of 20 μs. A diode reverse bias level of 2 V has been used with a bias resistor value of 20 kΩ during the experiments.

For the pulse height spectra gathering, 256 histogram bins were created using an incremental bin resolution of 5.64 fC/bin, which corresponds to 4.7 mV after charge-to-voltage conversion. The experimental pulse height spectrum is shown in Figure 9.7. Following electronic suppression of pulses below 170 mV, which dominate the pulse height collection without electronic suppression, the main contributions to the detection signal come from the conversion electron and the Gd k-shell Auger electron resonances. For ^{158}Gd, the

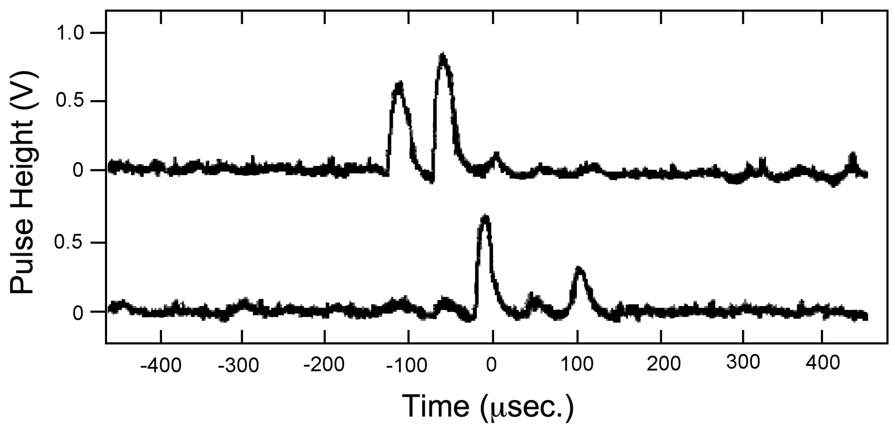


Figure 9.6 – Charge sensitive amplifier output waveforms.

current pulses generated by 79.5 keV (and other) conversion electrons can be efficiently measured and reliably identified. The K-shell binding energy is 50.2 keV resulting in a variety of Auger electron resonances of decreasing kinetic energy, ending in the 29.3 keV conversion electron centered pulse [92]. The K-shell electron excitation is accompanied by a 44 keV X-ray [92–94] which is not identified in the spectra in Figure 9.7. The M shell binding energy is 1.8 keV, resulting in only a small reduction of the 79.5 conversion electron energy to about 77 keV. Overall the pulse height spectra results in the creation of 40 to 80 keV pulses largely for ^{157}Gd(n,γ) that are in good agreement with the model calculations. The features in the experimental spectra can be assigned to specific expected K-shell Auger resonances, but with a non-uniform distribution over this range [92].

To provide a comparison with theory, the devices have been simulated using the Monte Carlo N-Particle Transport Code (MCNP5.0) [95], where a planar source of 10^{11} neutrons was assumed and a model neutron spectrum was calculated from 30 eV to 14 MeV. To better compare experiment with the model simulations, there are corrections at the low energy end of the model pulse height spectra to account for the large number of counts rejected by electronic noise and signal suppression below a 170 mV pulse height in the experiment. By incorporating these corrections to the model, there is considerable agreement between experimental and theoretical pulse height spectra displayed in the bottom portion of Figure 9.7. The deviations observed between simulation and experiment at larger pulse heights can be attributed to incomplete charge collection due the fact that the 15% Gd doped HfO_2 film is rather thin.

It should be noted that the presented prototype system is not radiation hardened, thus it will be affected by radiation striking the electronics. Fast neutrons may cause crystal de-

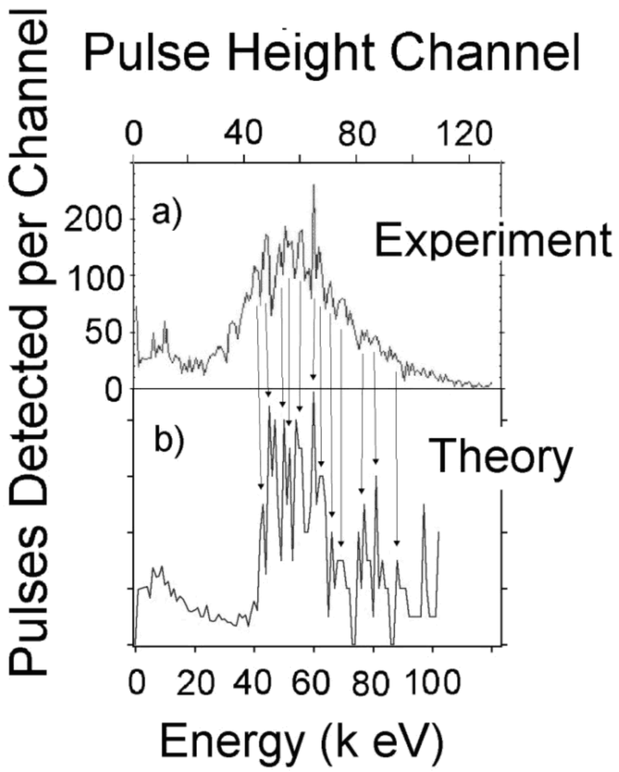


Figure 9.7 – The pulse height spectra of 15% Gd doped HfO_2 on n-type Si(100) with thermalized neutrons from a PuBE source.

fects which accumulate slowly over time to eventually cause permanent circuit failure and thermal neutrons may cause single-event upset errors. The software includes a watchdog timer to facilitate recovery from such software errors.

9.3 Gamma Detection Results

The front end was connected to a PMT-based gamma detector. The resulting histogram can be processed to determine the isotopes present in the environment. This is an application which has previously been accomplished using much more powerful computers that require much more power.

The following sections describe the operation of the detector, the algorithm used, and the results.

9.3.1 Detector Operation

The basic operation of scintillation based gamma detectors is shown in Figure 9.8. An incident gamma ray is absorbed by the scintillator which then emits a large number of photons which are converted into photoelectrons by the photocathode of the photomultiplier tube (PMT). The photoelectrons are amplified by each dynode stage before being collected at the anode to produce the desired signal. The number of photons and therefore the size of the anode current pulse are proportional to the incident gamma ray energy.

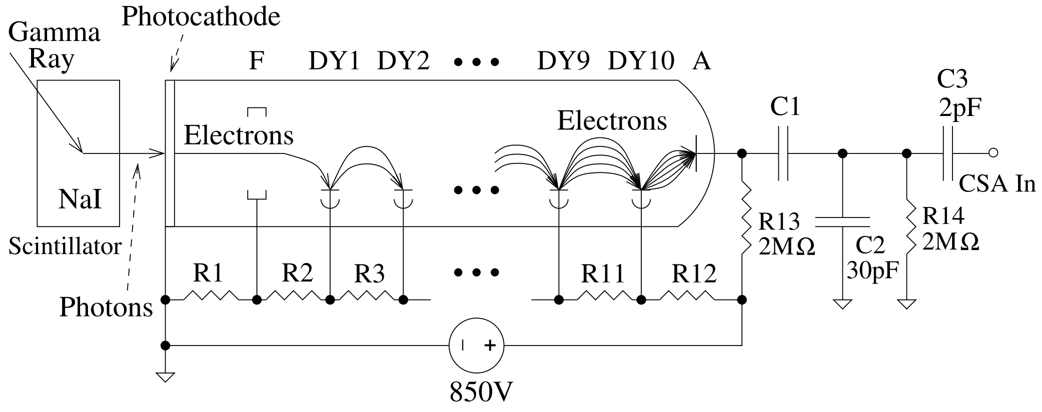


Figure 9.8 – Gamma detector schematic.

The detector used is a simple NaI based scintillation detector. It consists of a 1×1 inch cylindrical scintillator crystal optically coupled to a 2 inch photomultiplier tube. The 10-stage PMT has a quantum efficiency of 25% and produces a nominal 10^6 current gain with the maximum high voltage bias of 1.1 kV. The assembled crystal and PMT are enclosed in a light-tight housing and μ-metal shield to decrease the effects of environmental magnetic fields on the operation of the PMT.

The detector interface circuit is shown in Figure 9.8. It consists of the biasing scheme of the PMT using a grounded cathode configuration, and a charge divider to decrease the charge amplitude before going to the charge sensitive amplifier. Though the conventional method of biasing a PMT is by grounding the anode, a grounded cathode configuration must be used in scintillator detectors so that the scintillator crystal and μ-metal shield can be connected to the cathode potential. If the shield is instead connected to the anode potential, glass scintillation may result in which the electrons inside the PMT strike the inner bulb wall instead of the dynodes, resulting in increased noise [96].

The resistors R1-R12 form a voltage divider to bias the 10 individual dynodes and focusing electrode, while C1 is a high voltage DC block capacitor. C2 and C3 form a capacitive charge divider to attenuate the signal so it does not saturate the front end. If a PMT with fewer stages was used, thereby reducing the complexity and high voltage bias supply, the charge attenuation could be reduced or eliminated.

Presently, a high voltage supply of 850 V is used to bias the PMT. This is much less than the maximum 1.1 kV. The lower voltage produces a lower gain which decreases the necessary charge attenuation. A CsI(Tl) type scintillator could be used and combined with a photodiode to greatly reduce the required power consumption of the detector and overall sensor system. The CsI(Tl) scintillator, with its larger photon output, can be directly interfaced with a photodiode, thereby eliminating the need for a PMT and the high voltage supply. However, the CsI(Tl) scintillator has a poorer energy resolution, so the nuclide identification performance may suffer. A survey of potential detector technologies can be found in [97].

9.3.2 Gamma Isotope Identification

The embedded code to perform gamma isotope identification consists of three main parts. First, a simple real-time program collects data from the front end and processes the data into a histogram. This effectively implements a multi-channel analyzer. Next a peak search algorithm identifies the locations of and areas under the located peaks. Finally, a matching algorithm uses the peak locations and area data from a calibration dataset and the positions and area from the detected dataset to determine which isotopes are present.

Because the processor does not have floating-point resources in hardware, fixed point arithmetic is used throughout. Generally 16-bits are used for the fixed-point representation. However, 32-bits are occasionally used as intermediate results of calculations to maintain precision. As is common with many low-end embedded systems, no hardware divide resources exist, so division operations are performed in software, making them computationally expensive. For this reason, the number of divisions in the code was minimized.

The histogram generation code is executed by the interrupt triggered from the front end. This interrupt is triggered whenever more than 12 events are waiting in one of the front end FIFOs. The raw histogram is shown in Figure 9.9(a) for the ^{133}Ba, ^{65}Zn, and ^{137}Cs combined spectrum. This data is normalized to the peak value in the histogram before further processing.

The peak search algorithm is based on well-known methods described in [98]. The algorithm finds the negative going zero crossings of the first derivative to identify the peaks.

However, because the derivative amplifies noise, a triangular smoothing algorithm is first performed on the data. The formula for the smoothed data is given by:

$$S_j = \frac{Y_{j-2} + 2Y_{j-1} + 3Y_j + 2Y_{j+1} + Y_{j+2}}{9} \qquad j = 3 \ldots N-2 \tag{9.2}$$

where N is the number of channels. This is equivalent to two passes of a 3-tap rectangular smoothing filter. The smoothed data is shown in Figure 9.9(b).

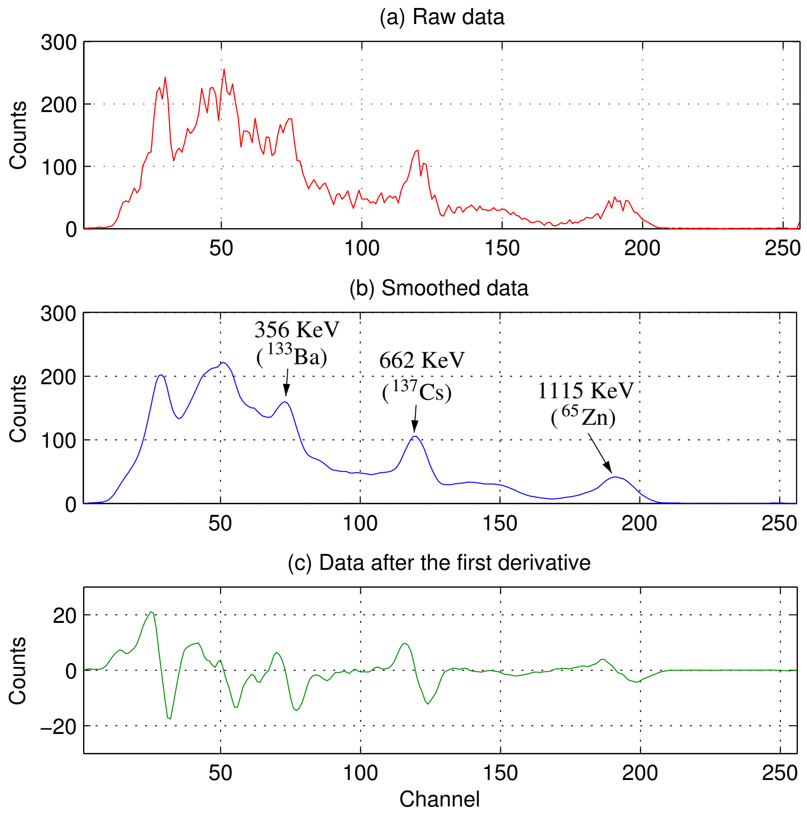


Figure 9.9 – ^{133}Ba, ^{65}Zn, and ^{137}Cs combined gamma energy spectrum.

Next the first derivative is taken which is shown in Figure 9.9(c). The negative-going zero crossings are found as potential peaks. To determine the significant peaks from the noise peaks, a measure of the peak area is necessary. To determine the area, the peak width and the background level must be estimated. The peak width is determined by the detector and detector electronics, and is a function of the gamma energy. This function is determined a priori, and is approximated by an increasing linear function. The background is simply estimated as a step function with the two endpoints selected by the width of the peak. This is illustrated in Figure 9.10. The area of each peak is then compared to a threshold to filter out the noise peaks.

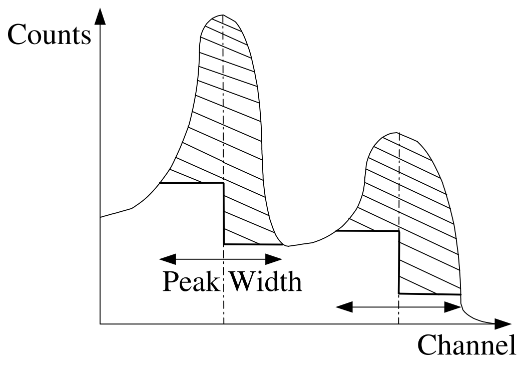


Figure 9.10 – Area calculation example

The peak search algorithm is executed on a known calibration data set for each isotope. The peak locations and areas are stored in a library for later comparison with the detected data. The same peak search algorithm is executed on the detected data to produce a set of peak locations and areas. For each isotope in the library, the matching algorithm first looks for peaks in the detected data to match with the calibration peaks. To match calibration and detection peaks, the location of the peaks must be in within 3-5 channels of each other. Once the peaks of interest are determined in the measured spectrum, the sum of the areas of these peaks is normalized to the sum of the areas of the calibration peaks. This allows a direct comparison between the area of each detected peak to the area of the associated calibration peak. If the locations of the peaks are similar and the areas are similar, then a match is declared.

9.3.3 Performance

Three gamma radiation sources were used for testing: a ^{133}Ba source, a ^{60}Co source, and a combination ^{65}Zn and ^{137}Cs source. The ^{133}Ba source emits a low energy gamma ray peak at 356 KeV; the ^{60}Co source emits radiation at 1.17 MeV and 1.33 MeV; and the combination ^{65}Zn and ^{137}Cs source emits a 662 KeV peak from the ^{137}Cs and a 1.12 MeV peak from the ^{65}Zn. A spectrum showing the combined ^{133}Ba, ^{137}Cs, and ^{65}Zn was shown in Figure 9.9 and the smoothed spectrum of ^{60}Co is shown in Figure 9.11, for completeness.

Table 9.5 gives pertinent specifications of the implemented design. The power consumption of the gamma isotope identification depends on the number of times the gamma

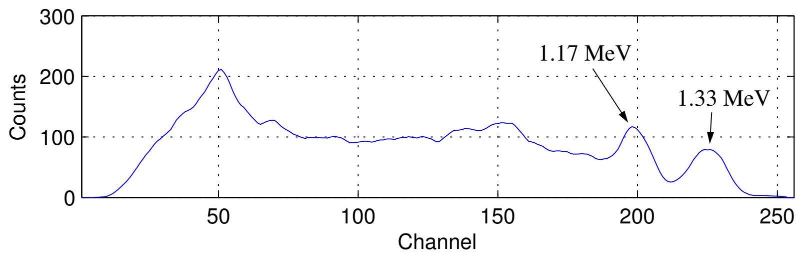


Figure 9.11 – ^{60}Co smoothed spectrum

isotope identification is performed in a given time interval. For example, if a certain application requires the gamma isotope identification to be run every 100 seconds, then the incremental power consumption would be energy/time or 4-12 μW; the trade being execution time versus power consumption. The gamma isotope identification software requires 11 kB of code space and can execute in less than 100 ms.

Table 9.5 – Gamma Detection Code Specifications

Debugging System Code Size:	5 kB
Gamma ID Code Size:	11 kB
RAM usage:	8 kB
Gamma ID Execution Time:	68.8 ms[a]
Gamma ID Execution Energy Consumption:	0.41-1.24 mJ[b]

[a] With the processor clock at the maximum of 55 MHz.
[b] Energy/Power varies depending on DVS voltage setting.

The system was first calibrated with data sets from the sources independently. Next, 100 samples were collected from each of the sources alone as well as in combinations. Every collected gamma ray spectrum was analyzed independently for the presence of each of the isotopes in the library. The results are shown in Table 9.6, which shows reliable detection of single and multiple isotopes. The imperfect detection performance is generally caused by the peak locations shifting due to environmental factors and noise.

The gamma detection results show the designed system is versatile enough to work with different sensors and has sufficient processing power to handle numerous computational tasks relating to the sensors. The processor has sufficient processing power such that a more complex algorithm could be added for gamma detection to improve nuclide identification performance.

Table 9.6 – Gamma Classification Results

Isotope Present	**Isotope Detected**		
	^{133}Ba	^{65}Zn & ^{137}Cs	^{60}Co
^{133}Ba	97%	0%	0%
^{65}Zn & ^{137}Cs	2%	97%	0%
^{60}Co	1%	3%	99%
^{133}Ba, ^{65}Zn & ^{137}Cs	99%	94%	0%

9.4 Conclusion

This chapter presented some basic lab tests of the front end followed by some application specific tests. The included front end is versatile enough to work with numerous sensors, and the processor can run different algorithms depending on the sensor. The front end has sufficient performance for most applications, and in the applications tested, the sensor itself was the limiting part of the system, not the front end electronics.

The low power consumption allows for long-duration monitoring with a small battery. This is a major improvement over other systems which use much more complex and power consuming circuits to accomplish the same tasks.

Chapter 10

RF Transceiver Results

This chapter presents the measured results of the UWB transceiver and the narrowband transceiver. Both of them have bugs which complicate testing, but we can still perform some tests to acquire useful data. Section 10.1 presents the results for the UWB transmitter while section 10.2 presents the results for the UWB receiver. Section 10.3 presents the results for the narrowband transmitter and section 10.4 presents the results for the narrowband receiver.

10.1 UWB Transmitter Results

The UWB transmitter consists of the digital packet buffer, the digital serializer, the modulator, and the output drivers. Everything except the modulator and output drivers are synthesized within the digital section. The area for the UWB transmitter is given in Table 10.1.

Table 10.1 – UWB Transmitter Circuit Area

Description:	**Area: (mm^2)**
Modulator:	0.002
Drivers and Pads:	0.043
Digital Section:	0.11
Total:	**0.155**

Obviously, the transmitter area is very small, and that is mostly occupied by the packet buffer and the pads.

The measured power consumption of the UWB transmitter is shown in Table 10.2. The first measurement is the power consumption of the modulator which produces the small pulses. This is very small. The digital section consumes a moderate amount of power, but the primary consumers are the antenna drivers. The power for a number of widths and repetitions is given. It is obvious that beyond a width of 1.4 ns it does not consume more energy. Of course, the repetition value also increases the power consumption.

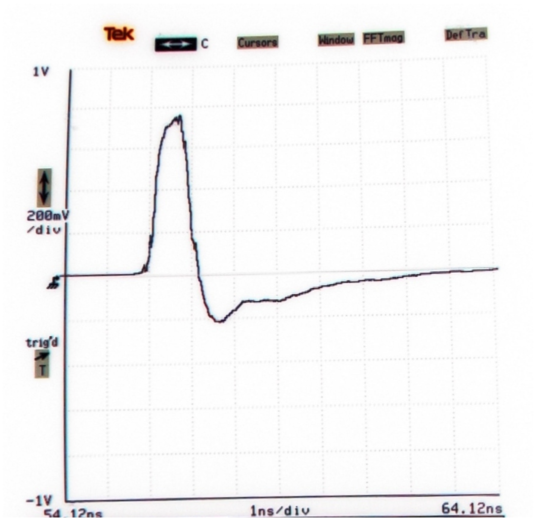
Table 10.2 – UWB Transmitter Power Consumption, $F_{\text{symbol}} = 10$ Meg, $\text{V}_{\text{dig}} = 1.4$

	Description:	**Power:**
	Modulator, no Digital or Power Drivers:	11.25 μW
	Tx Digital Section	451 μW
Selectable	Tx Driver, Rep = 1, Width = 600 ps	1.56 mW
	Tx Driver, Rep = 1, Width = 800 ps	1.84 mW
	Tx Driver, Rep = 1, Width = 1 ns	2.29 mW
	Tx Driver, Rep = 1, Width = 1.4 ns	3.42 mW
	Tx Driver, Rep = 1, Width = 1.8 ns	3.42 mW
	Tx Driver, Rep = 2, Width = 1 ns	3.96 mW
	Tx Driver, Rep = 3, Width = 1 ns	5.4 mW
	Total UWB Transmitter	**2-7.2 mW**
	Energy/coded bit (Transmitter):	**0.46-2.3 nJ**

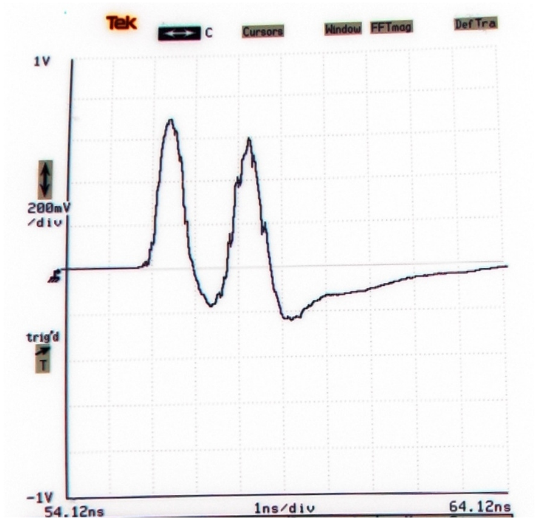
The UWB transmitter is an all-digital implementation whereby a digital circuit generates one or more digital pulses. These can be filtered if desired to meet a spectrum mask requirement. The transmitter is adjustable, so the width of the pulses can be adjusted from 600 ps to 4 ns. Furthermore, the pulses can be repeated up to three times to form a wave packet which affects the transmitted spectrum. Figure 10.1 shows some of the transmit waveforms which can be generated.

For later characterization of the receiver, the first waveform as shown in Figure 10.1(a) is used unless otherwise specified.

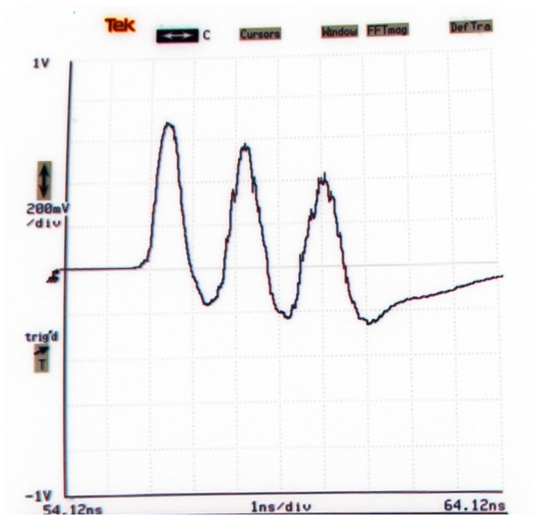
The spectra of the transmitted waveforms vary. Figure 10.2 shows the spectrum of the waveform in Figure 10.1(a). This is a very wide bandwidth waveform.



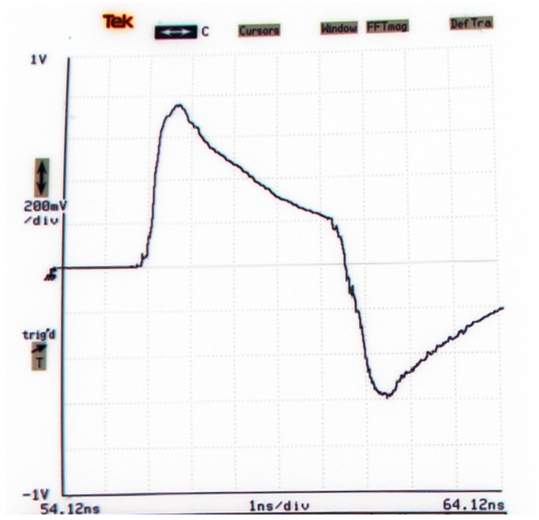
(a) Single transmit pulse



(b) Two transmit pulses



(c) Three transmit pulses



(d) Maximum width UWB transmit pulse

Figure 10.1 – Various transmitter waveforms.

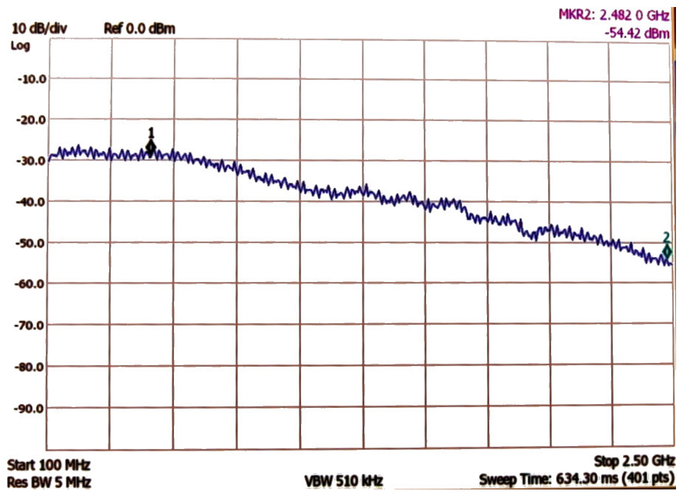


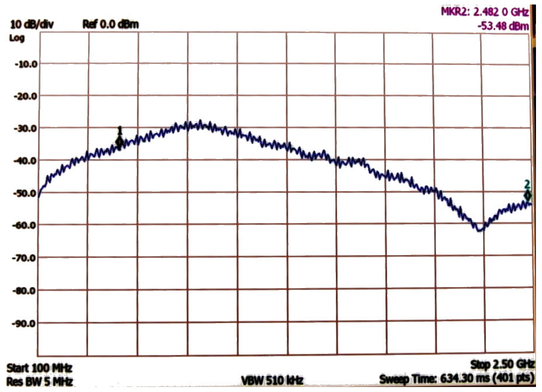
Figure 10.2 – Spectrum of unfiltered UWB pulses.

A filter can be added on the output to decrease the bandwidth. The filter added consists of a series LC filter with a center frequency around 1.7 GHz. This narrows the spectrum. Figure 10.3 shows the spectra of four different transmitted waveforms with a varying number of pulses transmitted. In the figure, two markers mark the 500 MHz and 2.5 GHz frequencies which are the edges of the receiver bandwidth.

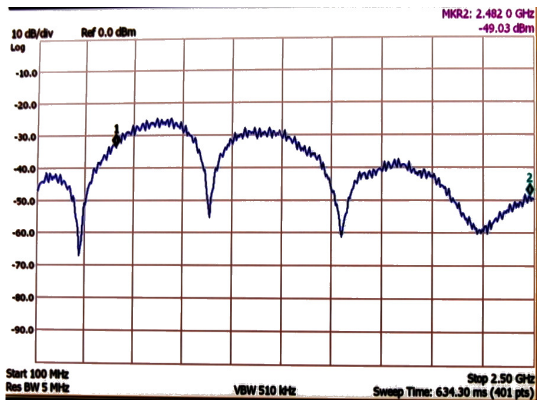
Transmitting multiple pulses changes the spectrum and the transmitted power. Using more pulses increases the transmitted power, and also increases the peaks of the frequency spectrum which may makes it harder to fulfill FCC spectrum masks for a given transmit power. Table 10.3 shows the transmitted power and the peak power spectral density of each of the four waveforms shown in Figure 10.3. As can be seen, the transmitted power increases, but the peak increases faster, therefore making any spectral masks harder to meet with more pulses. For further characterization of the UWB receiver, one pulse was always used.

Table 10.3 – Transmitted Power

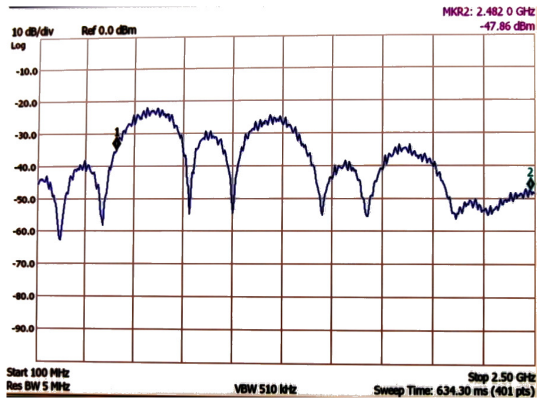
Pulses	**Tx Power (dBm)**	**Peak PSD (dBm)**
1	-9.9	-30
2	-7.14	-27
3	-5.5	-23
4	-4.25	-21



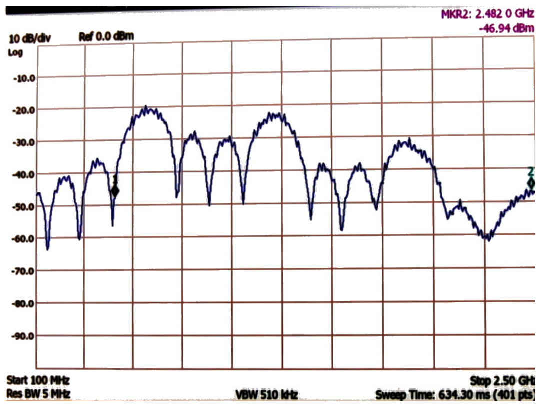
(a) Single transmit pulse



(b) Two transmit pulses



(c) Three transmit pulses



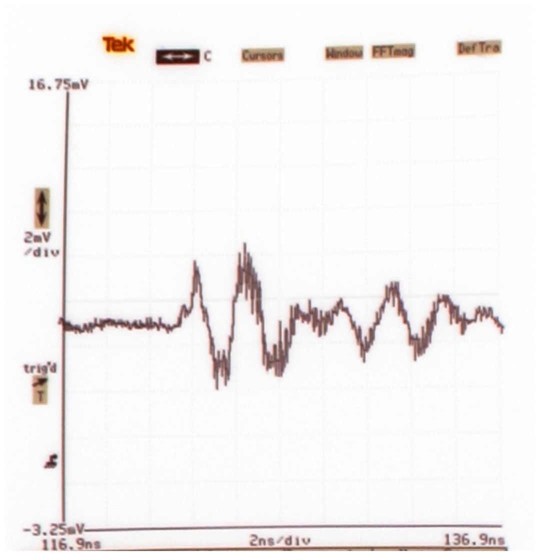
(d) Four transmit pulses

Figure 10.3 – Spectra for transmitted filtered signal.

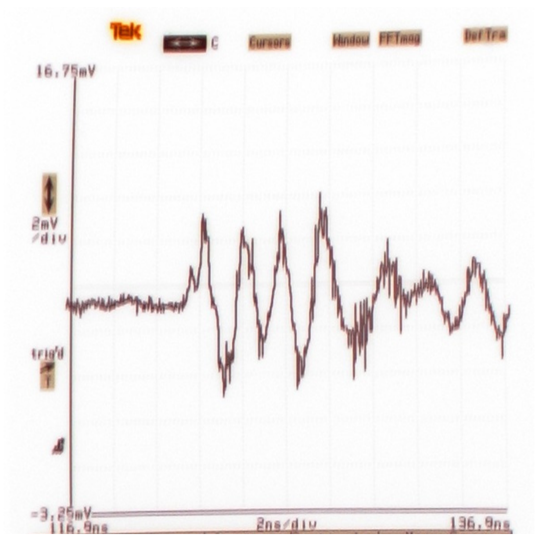
To characterize the transmit and receive antennas, Figure 10.4 shows the waveforms at the receiving antenna when both are placed in an anechoic chamber. Two things are noticeable in the figures. First, there is some reflection which shows up after the first received pulse, which may be caused by a reflection off a section of the anechoic chamber wall where the foam has fallen off! This does not affect the characteristics of the first received waveform though.

The second thing to notice about the received waveforms is how an extra pulse is added from that which is transmitted. This is due to the transfer function of the antennas. The figure was acquired with the transmitting antenna about 1 meter from the receiving antenna.

Figure 10.5 shows the effects a multi-path environment has on the received signal. These measurements were acquired in the lab, with cinder-block walls and other office furniture present and generally a large number of hard reflecting surfaces. Once again, these were taken with the transmit antenna about 1 meter from the receiving antenna.

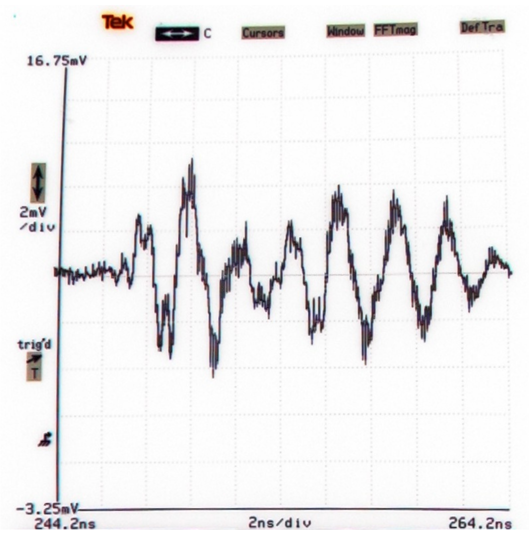


(a) One transmit pulse

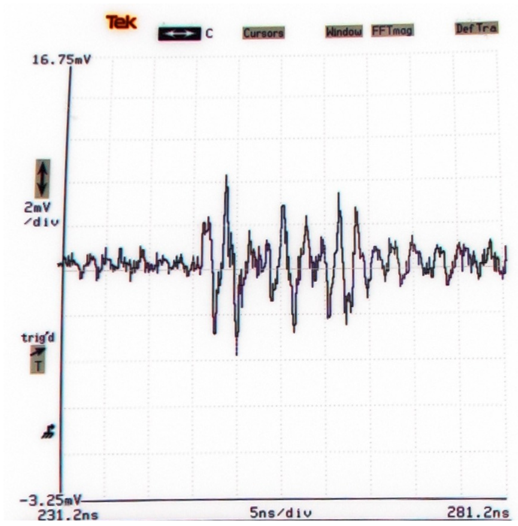


(b) Three transmit pulses

Figure 10.4 – Received waveforms in the anechoic chamber



(a) 2 ns/div



(b) 5 ns/div

Figure 10.5 – Received waveforms in the lab, one pulse transmitted.

The figures show many reflections which extend the signal energy over a much longer time period. Furthermore, if the transmitting and receiving antennas were further apart, the time spread and the number of resolvable reflections would increase. However, unlike in a narrowband communication system, the multi path environment is actually good for a non-coherent UWB receiver which will have better performance in a multi-path environment because all the separate resolvable reflections all get added in the receiver and increase the SNR. In the received waveform, the signal is spread out over 20 ns. The integration window of the receiver is set to 25 ns (but that may be changed in software), and the time between positions in the PPM modulation is 50 ns.

10.2 UWB Receiver

The UWB receiver consists of two main portions, the analog and digital parts. The digital portion consists of the packet buffer, the Viterbi decoder, the packet detection correlators, and the rest of the control logic; while the analog portion consists of the analog front end and the baseband. The area breakdown of all the parts of the UWB receiver is given in Table 10.4. Though the receiver is much larger than the transmitter, it is still quite small compared to other receivers presented in the literature.

Table 10.4 – UWB Receiver Circuit Area

Description:	**Area: (mm^2)**
RF Front End:	0.63
Packet Detection:	0.048
Bit Detection and Symbol Sync:	0.095
Total UWB Receiver Analog:	**0.9**
Digital Packet Buffer:	0.09
Packet Detection Correlators:	0.197
Viterbi Decoder:	0.179
Total Receiver Digital:	**0.59**
Total UWB Receiver:	**1.49**

Table 10.5 shows the measured receiver power consumption. The receiver takes slightly less power when receiving a packet than when searching for a packet. Much power is consumed in the RF front end, of which most is used in the LNA. The analog baseband, even with all the analog correlators, does not consume much power. The PLL which is operating at 800 MHz consumes much power. The PLL is then divided by 5 to get the 160 MHz clock the UWB receiver requires. This is also fast enough that the digital logic operating on it consumes a significant amount of power.

Table 10.5 – UWB Receiver Power Consumption, $F_{\text{symbol}} = 10$ Meg, $V_{\text{dig}} = 1.4$

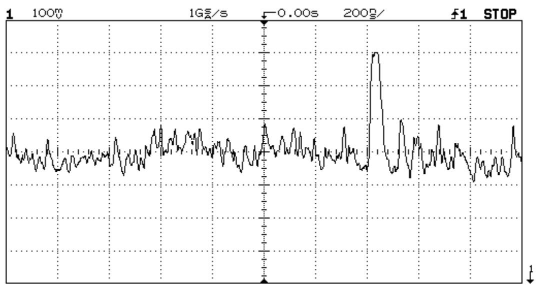
Description:	**Power:**
Receiver RF Front End:	10.1 mW
Receiver Analog Baseband, Searching for Packet:	3.24 mW
Digital Baseband, Packet Searching:	1.77 mW
Digital Baseband, Receiving Packets:	1.67 mW
Crystal, PLL, and Clock Generation & Distribution:	2.48 mW
Total UWB Receiver:	**17.6 mW**
Energy/information bit:	**3.52-17.6 nJ**

10.2.1 Receiver RF Front End

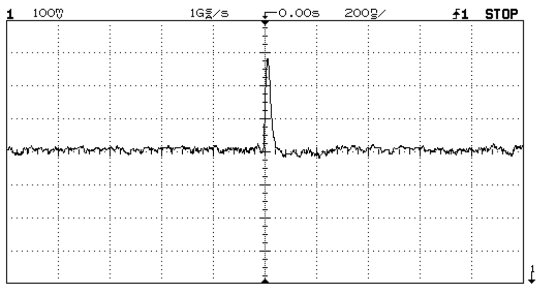
The RF Front end comprises the LNA, variable gain amplifier, self-mixer, and AGC loop. The resulting output signal is an amplified and squared version of the input which has a pulse where each wave packet exists. The resulting output signal under various conditions is shown in Figure 10.6.

Figure 10.6(a) shows the output signal when there is no input. In this state the output of the front end is the amplified noise floor, as can be seen in the scope capture. The spike at the right is the switching power supply noise.

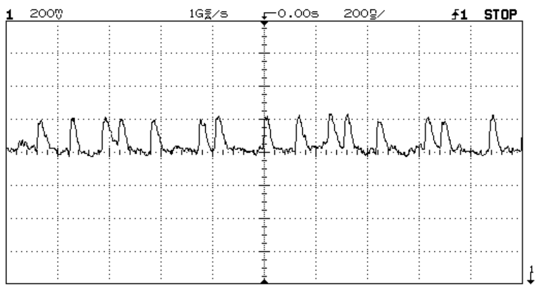
When the switching power supply switches, this noise is amplified in the receiver and always shows up as a large spike in the waveform, regardless of what is being processed by the front end. Thankfully, the power supply switches are few, so these spikes only rarely affect the receiver, and their effects are corrected for by the code. However, for the sake of interest, most of the waveforms in Figure 10.6 show this interference. This was expected and planned for during the design.



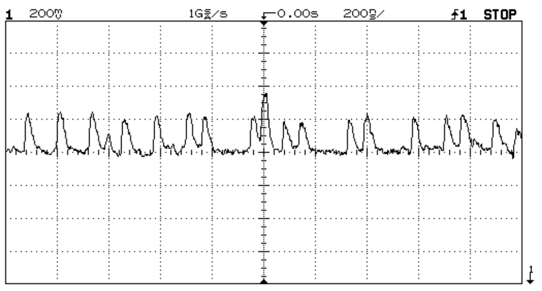
(a) No signal, only power supply interference



(b) No signal with a -50 dBm narrowband interferer and power supply interference



(c) Receiving a packet



(d) Receiving a packet with power supply interference

Figure 10.6 – UWB baseband signal with different conditions.

Figure 10.6(b) shows the output in the presense of a narrowband interferer. In this case the -50 dBm narrowband single-tone applied causes the RF front end to decrease its gain to stop it from saturating. This has the effect of making the noise much smaller as well. The power supply interference is still very visible (even more so than in Figure 10.6(a) because in that case the front end saturated on the spike).

Figure 10.6(c) and Figure 10.6(d) show the receiver receiving a packet; in this case at close range with good signal power. The UWB pulses are easily visible, and the power supply interference pulse is also seen in Figure 10.6(d).

The noise of the RF front end is difficult to measure exactly because the RF signal is not accessible anywhere along the signal chain, instead the squared RF signal is available (the baseband signal) as well as the AGC control loop voltage. A good approximation of the total input referred noise can be achieved by measuring the power of a narrowband tone required to produce a noticeable change in the AGC control loop voltage. At this point, the power of the narrowband tone is on the same order of magnitude of the power of the noise. At power levels above this, the narrowband tone dominates the baseband signal and the noise disappears, and at signal levels below, the noise covers the narrowband signal.

Table 10.6 gives the applied tone power, simulated input noise, and deduced input noise using this method.

Table 10.6 – RF Front End Noise

LNA Noise Figure:	8 dB
Simulated Total Input Referred Noise:	70 μV
Measured Narrowband Tone Power:	-63 dBm
Measured Noise Input Referred Noise:	158 μV

This is not a particularly accurate method of noise measurement, so the preceeding result can be taken to mean that the noise is somewhere close to what is expected via simulation.

10.2.2 Packet Detection

The sixteen packet detection receivers do not have any offset-cancellation circuitry, so their offsets in the comparators cause each receiver to either favor a zero or one. The integrators do not have a systematic bias because there is only one physical integrator and it is used for integrating over both time periods on the two different capacitors, so any non-ideality in the integrator is canceled. Nevertheless, each receiver exhibits a possibly large systematic bias due to the comparator offsets. This is different for each chip a sample of which is shown in Table 10.7. In this test no signal was applied, so the baseband signal is amplified noise. Under this condition, the individual packet detection correlators should produce truly random outputs. However, the measured correlator statistics are not random as seen in Table 10.7.

This systematic bias has some effect on the performance of the packet detection, however it is not too great. It has a similarly small effect on the false alarm rate.

The false alarm rate versus the correlation threshold is shown in Figure 10.7. The predicted value is the value obtained from Equation 5.9. The deviation at the low threshold is due to the inaccuracy in the low false-alarm rate measurement technique. The deviation from the predicted curve at high threshold values is because the predicted equation assumes low false-alarm rates, while this assumption is broken at high threshold values.

The measured values in Figure 10.7 use the packet detection bias values shown in Table 10.7. As can be seen by the close agreement between the measured and predicted values, the correlator bias has little affect on the false alarm rate.

Table 10.7 – Packet Detection Correlator Systematic Bias

Correlator Num.	**Prob. of ‘1’**
0	0.36
1	0.38
2	0.46
3	0.53
4	0.37
5	0.52
6	0.63
7	0.4
8	0.45
9	0.4
10	0.39
11	0.44
12	0.46
13	0.34
14	0.43
15	0.29

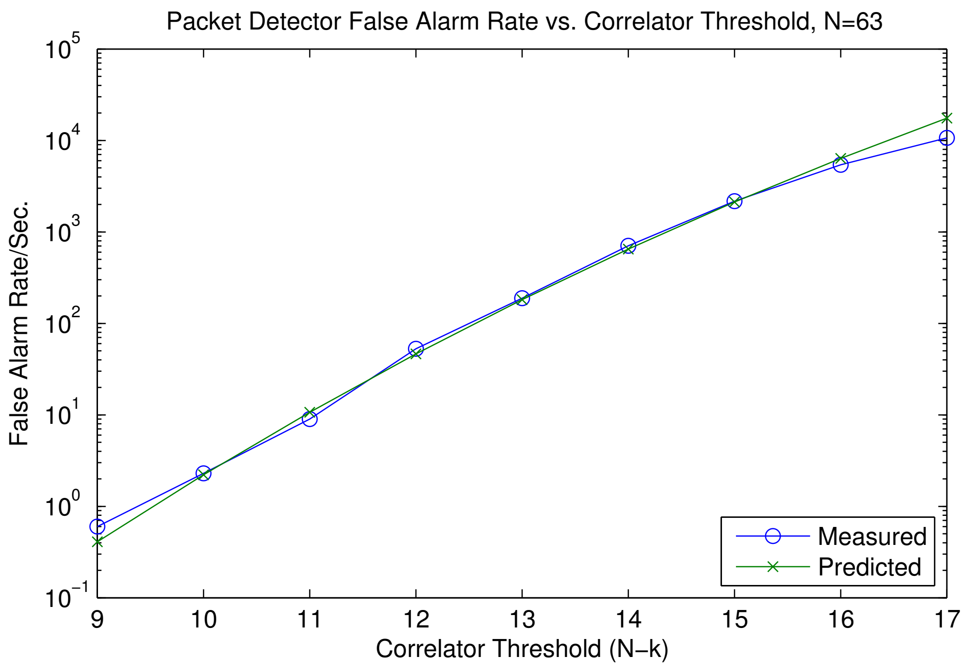


Figure 10.7 – False alarm rate versus correlator threshold

A digital bug exists in the digital packet detection correlators. The putative cause of this bug is crosstalk between two unknown wires; pinpointing the exact pair of wires is almost impossible. The problem manifests itself as causing large numbers of false alarms. In fact, when the bug is active, the receiver will not enter into the packet detection mode for more than a couple cycles before it “finds” a packet. This is invariant of the threshold value. In some chips this bug is always active (which therefore makes the complete receiver non-operational). However, in other chips it is only active sometimes, but will disappear at other times. The temperature of the chip seems to affect weather the bug is present or not; heating the chip slightly will sometimes make the bug disappear. This heating changes the details of the timing slightly which fixes the problem sometimes. This is one of the major flaws in the receiver and stops many, if not most, chips from working. Virtually all the chips will have this bug when first powered up, but on many chips it will disappear after operating the UWB receiver for 10 minutes (probably due to the localized heating mentioned before).

A telling result giving an indication of the range and sensitivity of the receiver is the number of packets detected as a function of received power. For this test, the packet detection circuitry used N=63 and N-k=13. This setting is the maximum value for N, so it provides the best detection performance versus a smaller N value. For this test, a transmitter was enclosed in a Faraday cage, the transmitted signal was brought out on coax where it was passed through a variable attenuator and fed into the UWB receiver which was in a separate Faraday cage. By knowing the UWB transmitted power and the setting of the attenuator, the power of the signal can be determined. Figure 10.8 shows the percentage of packets detected as a function of received power.

From this figure, we can see that the packet detection starts to fail around -85 dBm. This also means that this is where the symbol error rate goes above 0.2 and where the rest of the receiver would stop operating reliably.

A similar test can be performed except using antennas and plotting the packet detection performance versus the range which is shown in Figure 10.9. This data was collected in a hallway, so there were strong multi-path components present. The data shows the packets can be found reliably out to 8-9 meters.

10.2.3 Bit Detection and Symbol Synchronization

There exists a bug in the symbol synchronization circuitry. In particular, the mismatch between the early-middle-late sampling correlators is too great which stops them from work-

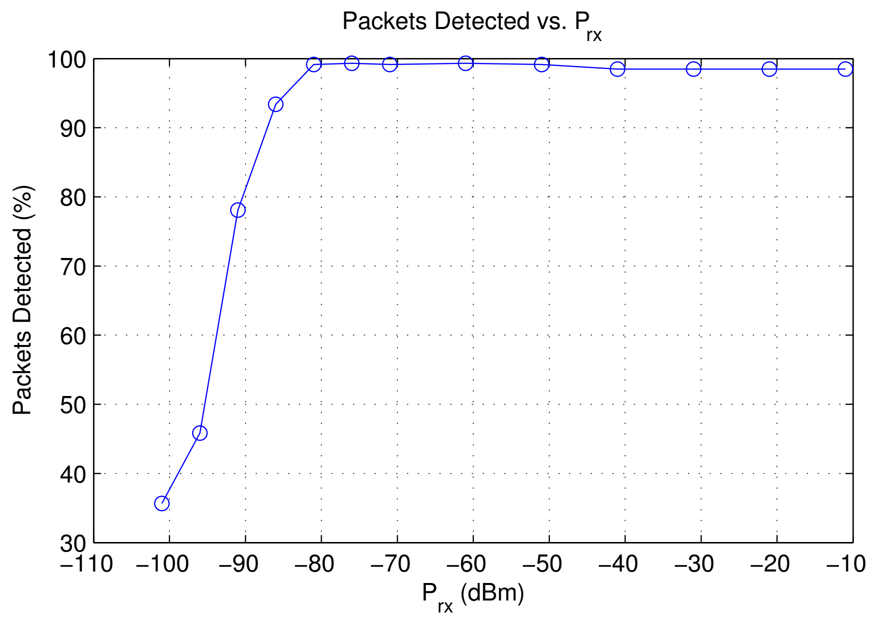


Figure 10.8 – Packet detection results versus received power.

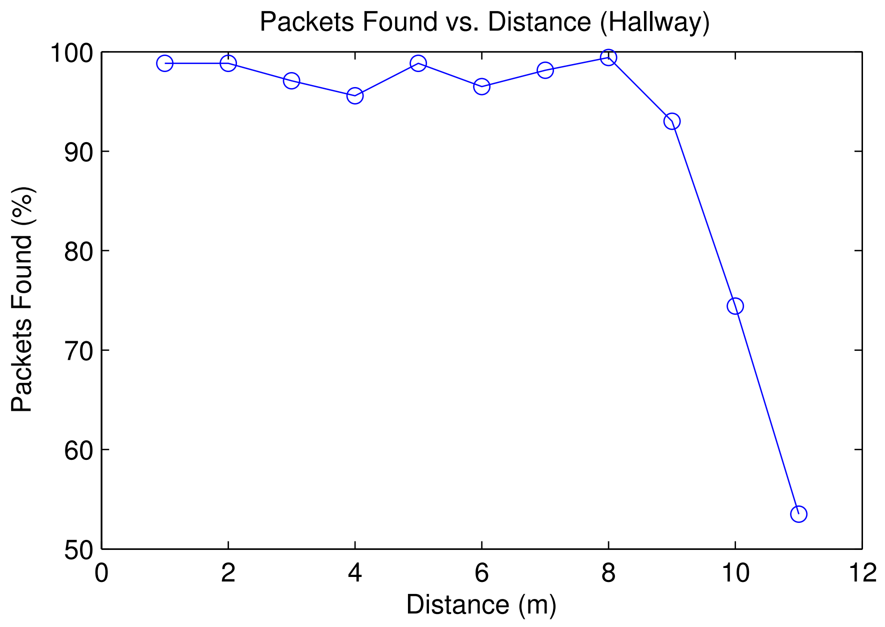


Figure 10.9 – Packet detection results versus distance.

ing well in low SNR conditions. Some adjustment can be done on the triangular waveforms to compensate for the mismatch, but this cannot eliminate it entirely. Some receivers work well at short range (usually 3 m or less) but at longer range the symbol synchronization fails and virtually no packets are received correctly. At short ranges, the receiver will receive virtually all (>97%) of the transmitted packets without error. This bug limits the performance of the receiver considerably as seen in Figure 10.10. The figure shows the number of correctly received packets as a function of received signal strength. Compared to the power at which the packets can be reliably detected (-85 dBm from Figure 10.8), the performance is very poor.

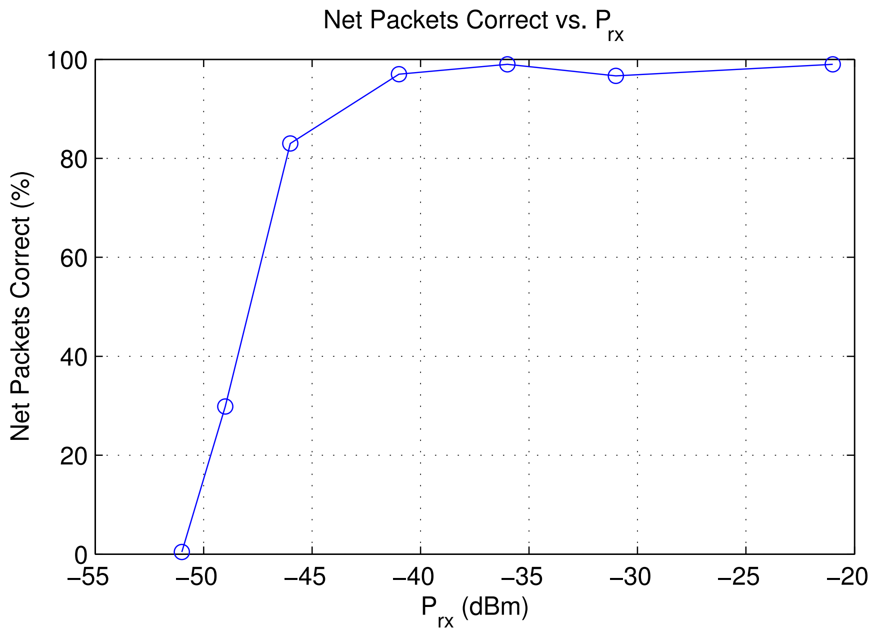


Figure 10.10 – Correct packet reception versus received signal power with symbol synchronization enabled.

This unfortunate bug does not stop all testing in the long-range regimes, instead the symbol synchronization can be disabled completely, and rely on the correctness of the initial synchronization provided by the packet detection circuitry and the fact that the transmitter and receiver will not drift too far if the packet is short. The probability of a packet being received in this method varies greatly depending on the number of symbols in the packet.

Figure 10.11 shows the number of packets received correctly versus distance with different coding rates. The packet length is 10 data bytes, or 13 bytes with address, length, and CRC fields. The percent correct on the y-axis has been scaled for each curve so that the maximum is 100%. This is because there are a certain percentage of packets which will be lost, regardless of the BER of the channel, due to the symbol synchronizer not being enabled. The percentage of packets where symbol synchronization is lost varies depending on the coding rate (and therefore the number of symbols in the packet) from 14% lost for the rate 1/2 code to 44% lost for the rate 1/10 code. Each curve has been scaled by the maximum percentage correct to attempt to compensate for this loss. The data for Figure 10.11 was collected in a hallway with with strong multi-path signals present.

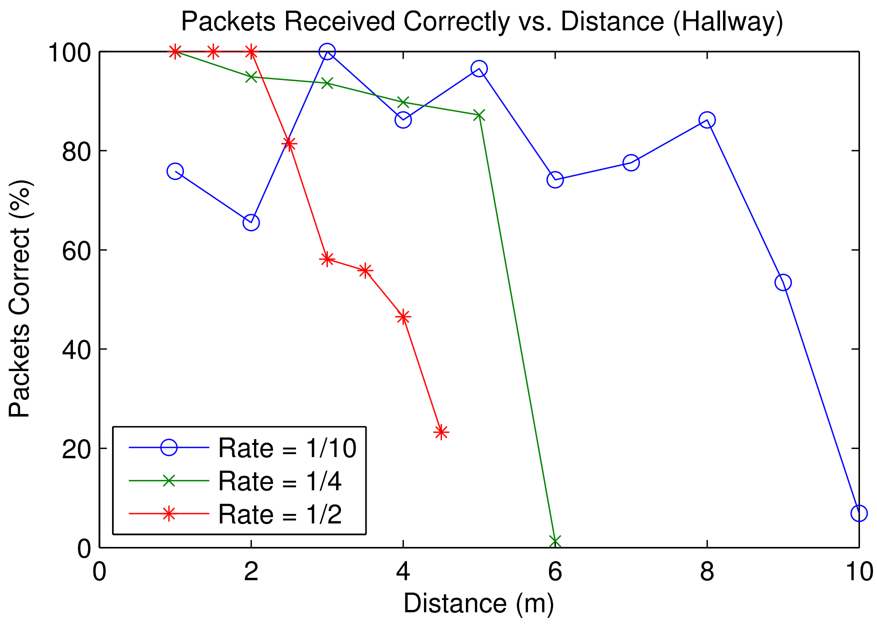


Figure 10.11 – Percent of packets received correctly vs. distance with no synchronization enabled.

The packet loss in Figure 10.11 is not strictly caused by the symbol error rate overwhelming the code. It can also be caused by systemic errors in the accuracy of the initial synchronization fix provided by the packet detection logic. For example, there is a loss of packets around 1-2 m for the rate 1/10 code which was investigated further. In this range, the signal is very strong and the error is caused by a loss of synchronization. This dip was consistent even when moving the receiver antenna, showing that it was not just caused by fading. It could be caused by, for example, the packet detection logic consistently giving a symbol sync value which is too early causing more packets to lose sync later in the packet.

The variability in Figure 10.11 is also partially caused by the random fading of the channel. Empirically, this was worse at the extreme of the range—as would be expected. Even though the data in the figure is not perfect, it gives a good approximation of the what the range would be if symbol synchronization worked well. It also agrees with the range estimate given in Figure 10.9. As expected, the range drops off quickly for higher rate codes.

A similar test can be run but instead sweeping the received signal power. This is shown in Figure 10.12. This is only for the rate 1/10 code. Once again, in this figure the data

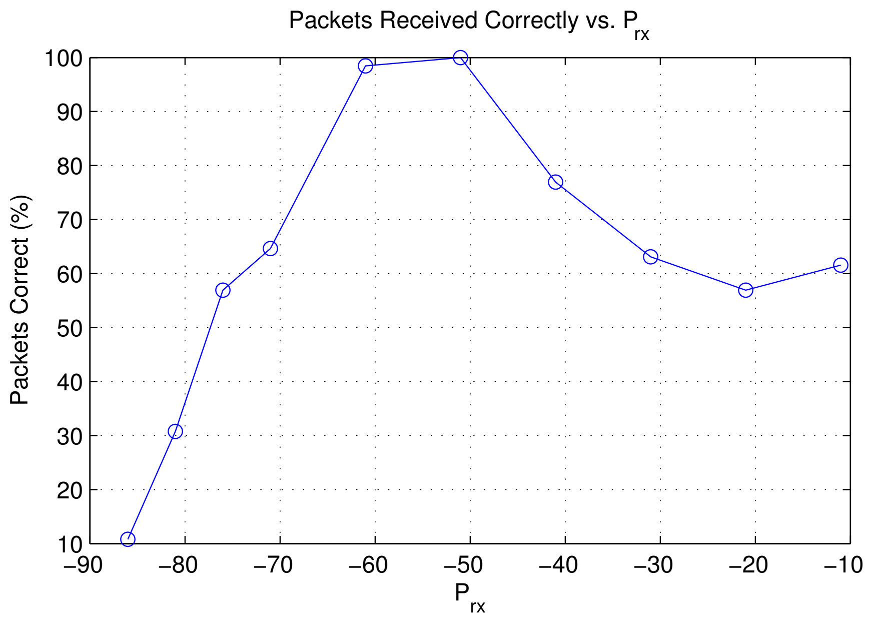


Figure 10.12 – Percent of packets received correctly vs. received signal power with no synchronization enabled.

is normalized to the maximum. There is a very noticable drop with very strong signals. This is due to some effect in the receiver whereby strong signals cause the initial packet detection synchronization to be off and causes the loss in packets. This drop is not noticed if the symbol synchronization circuitry is enabled which works in this range because of the high signal strength.

Figure 10.13 shows an approximation of the symbol error rate versus received signal power. This was taken by selecting the packets which were decoded correctly with the rate 1/10 code and finding the average symbol error rate as reported by the Viterbi decoder. This corrupts the data in two ways. First, a self-selection bias is present whereby only the packets with low enough symbol error rate to be decoded correctly get counted. This has a tendency to under report the symbol error rate. This effect is largest at high symbol error rates and negligible at low symbol error rates. A second error is from a few packets which lose synchronization late in the packet but where the packet can still be decoded correctly due to the interleaver spreading the errors. In this case, the Viterbi decoder will find many errors, but they are mostly caused by the receiver losing synchronization, not from noise. This effect causes the data to overestimate the symbol error rate, particularly at low symbol error rates. The on-board switching buck converters also cause errors to occur—even in very strong signal conditions, so the symbol error rate is never very low.

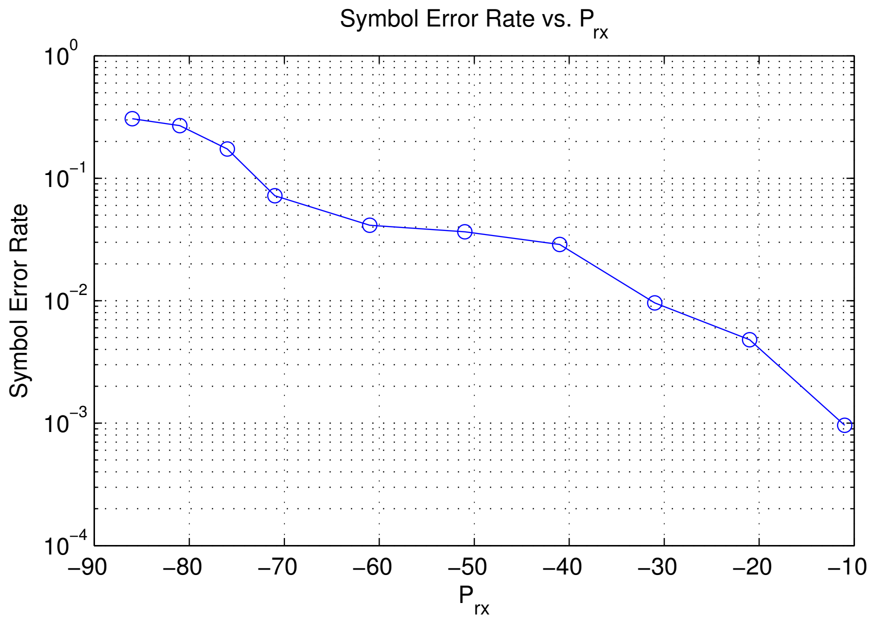


Figure 10.13 – Symbol error rate vs. received signal power.

Figure 10.9 can be used to estimate the range and BER of the receiver if the symbol synchronization bug was not present. From Figure 10.9, the packet detection begins to fail around 8-9 m. We know from the packet detection simulation in Figure 5.33 on page 150 that the packet detection begins to fail at around 0.2 symbol error rate. Then from our convolutional code simulation (Figure 5.7 on page 121) we know that this gives a bit error rate of less than 10^{-4} at 8 m.

Table 10.8 gives a comparison of this receiver with some related work in the field. Each receiver uses many different approaches which are hard to summarize in a table, so a short description of the individual works follows. In the table, the sensitivity measurement must also be paired with the symbol rate at which the measurement was taken. This is because the receiver has a limit of the minimum received energy/pulse, but the sensitivity is the received energy/unit time. Therefore, the pulse rate is required to compare two receivers with each other. For example, when comparing this work (-85 dBm 10 Meg) to [58] (-

99 dBm 0.1 Meg), this work is actually more sensitive in terms of energy/pulse because of the lower data rate of [58]. If the presented transceiver operated at 0.1 Meg symbol rate, the sensitivity would be -105 dBm.

Table 10.8 – UWB Receiver Comparison

Description	**This Work**	**[58]**	**[61]**	**[99]**	**[100]**
Year	2010	2007	2010	2006	2009
Process	0.18 μm	90 nm	0.13 μm	0.18 μm	0.13 μm
Area (mm^2)	1.49	1	4.52	0.38	6
Supply (V)	1.8	0.5 & 0.65		1.8	1.2
Frequency (GHz)	0.5-2.5	3-5	0-0.96	0-0.96	4-5
Modulation	PPM	PPM	BPSK	OOK	DBPSK
Symbol Rate (Meg/sec)	10	16.7	39 & 19.5	1	31
Energy/Symbol (nJ)	1.25	2.5	0.1	0.3	1.1
Bit Rate (Mbps)	1-5	16.7	0.3-39	1	0.347
Energy/Bit (nJ)	3.5-17.6	2.5	1.5	0.3	130
Aquisition Implemented	yes	no	yes	see text	yes
Sync Symbols	64-192		960		
Tracking Implemented	yes	no	yes	see text	yes
Sensitivity (dBm)	-85	-99	-69	-30	-78
Sens. symbol rate (Meg)	10	.1	19.5	1	unknown
Range (m)	8		10	1	22

In [61] a 0-960 MHz receiver is presented in 0.13 μm CMOS. The receiver achieves low energy/symbol by using a low-frequency RF band. It maps each bit to a PN code of 1-15 bits to increase the SNR. It uses dual analog correlators, so it takes a long time to acquire the signal and cannot do so in poor SNR conditions.

In [99] a sub-1 GHz receiver is implemented in 0.18 μm CMOS. It is a simple architecture with one template generator which is correlated with the received signal. Its acquisition scheme consists of searching for the packet by sequentially trying all locations which is very slow. Its tracking scheme consists of waiting until synchronization is lost and then searching the local neighborhood to re-gain the signal. Of course, this requires an external ECC code to correct the errors which will occur when it loses synchronization which was not implemented. This tracking and acquisition scheme requires a very strong signal with low noise, hence the low sensitivity. It is small in area due to the lack of inductors.

In [100] a complete IEEE 802.15.4a compliant UWB transceiver is implemented. This is done using energy detection and a fast direct digitization of the energy signal with baseband all in the digital domain. They use a unique architecture which allows good performance from a 1-big digitizer with moderate RF front end power consumption (17 mW), however the digital baseband requires 8 mW when receiving the packet and 27 mW when searching for a packet. However, the all-digital baseband allows good baseband performance and their unique RF architecture allows good sensitivity. The energy/symbol figure only accounts for the analog section power consumption, not the digital baseband while the energy/bit accounts for the baseband power.

This sampling of the published papers shows a wide range of different receiver architectures and performance. The presented work boasts a tolerance for low SNR that is unmatched by any receiver. The signal acquisition is very robust compared to other presented receivers. Some other receivers have a lower-noise RF front end usually by decreasing the bandwidth which helps their performance.

10.2.4 Conclusion

In conclusion, the receiver presented has a few major strong points. First, it is very resistant to poor SNR and channel conditions and can work with raw symbol error rates up to 0.2. Secondly, even under these poor conditions, it can acquire packets reliably while only requiring a minimal number of synchronization symbols appended to the packet. Though many RF front-end interfaces in the literature are presented with similar performance, the baseband processing allows the receiver to work at lower SNR values for longer range. The baseband processing hybrid of many analog correlators is a very low power architecture with power consumption as low as any presented in the literature.

10.3 Narrowband Transmitter

The chip implements a very simple narrowband transmitter. The transmitter is fully integrated with the exception of a single LC tank. The transmitter is quite simple in concept—modulating the output of a frequency synthesizer with a digital square wave generated by the digital section to produce on-off keying. This signal then goes to a series of inverters which drive the antenna. The data rate is 500 kbps.

This signal chain basically works in the chip, but one problem did arise. Within the digital section, due to crosstalk or other unknown reasons which are not detectable in the simulator, the 16 MHz crystal oscillator frequency is fully or partly driven out to the signal which modulates the RF carrier. This then causes the RF carrier to be modulated by the crystal frequency. This problem varies in intensity from chip to chip, and is a strong function of the supply voltage as will be explored later. It is present to some degree in all chips tested, however, for many chips the effect is small enough it has negligible impact on the entire operation of the transmitter, instead just making the receiver have strong 16 MHz spurs.

Table 10.9 shows the die area used by different parts of the narrowband transmitter, as well as the power consumption. More detailed results on the power amplifier power consumption are given later in the chapter.

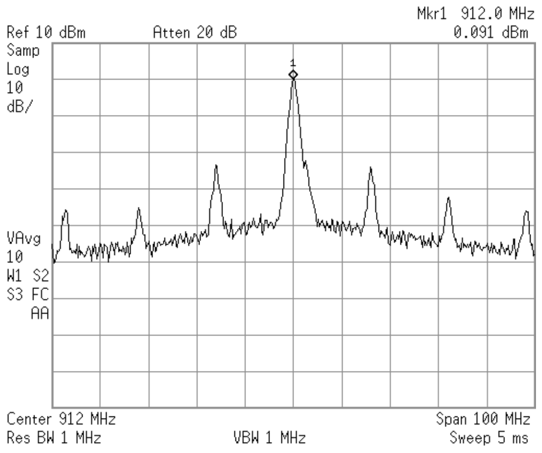
Table 10.9 – Narrowband Transmitter Specifications

Area	
PLL Area:	0.116 mm^2
Digital Section Area:	0.102 mm^2
PA and Pad Area:	0.018 mm^2
Total Area:	**0.236 mm^2**
Power	
PLL Power Consumption	1.8 mW
Digital Section Power	0.72 mW
Power Amplifier Power, $P_{tx} = 0$ dBm	6.46 mW
Total Transmitter Power:	**9 mW**
Energy/bit:	**18 nJ**

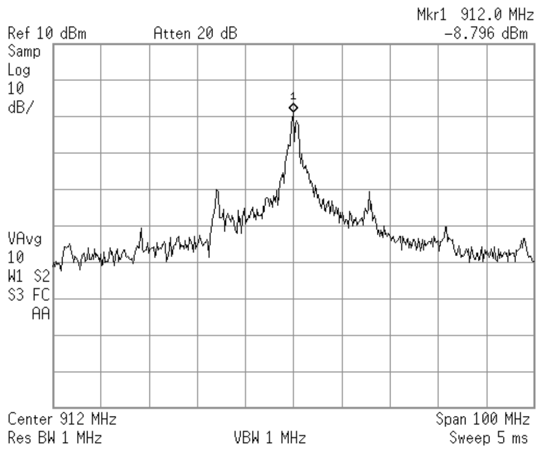
Figure 10.14 shows the spectrum of the narrowband transmitter. Figure 10.14(a) shows the spectrum of just the carrier, without the data packets modulated on it. However, the 16 MHz spurs are obvious, these are caused by the unintended modulation with the 16 MHz crystal frequency. On some chips, operating at high supply voltages, the spectrum has much smaller 16 MHz spurs, while on others the spurs are much larger.

Figure 10.14(b) shows the modulated spectrum. Because there is no pulse shaping, the spectrum spreads out over a wide bandwidth. The spurs are still present, but not as visible. The data is Manchester-encoded, so the 500 kbps data rate translates into a maximum modulation frequency of 1 MHz.

Figure 10.15 shows a time-domain view of the carrier and the unwanted modulation. Figure 10.15(a) shows a close-up of the time domain carrier. The optional L-C tank shown



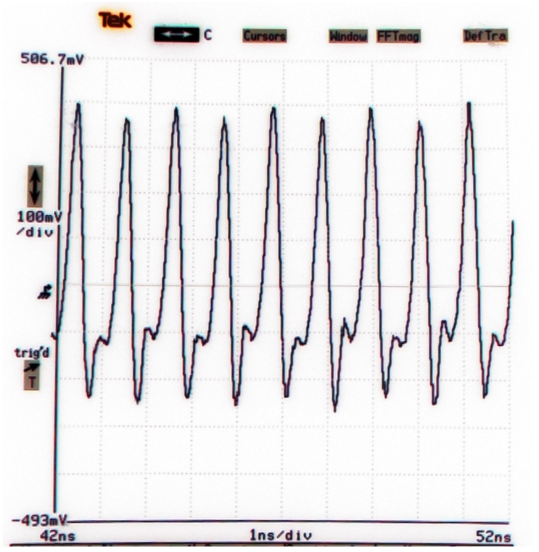
(a) Carrier only



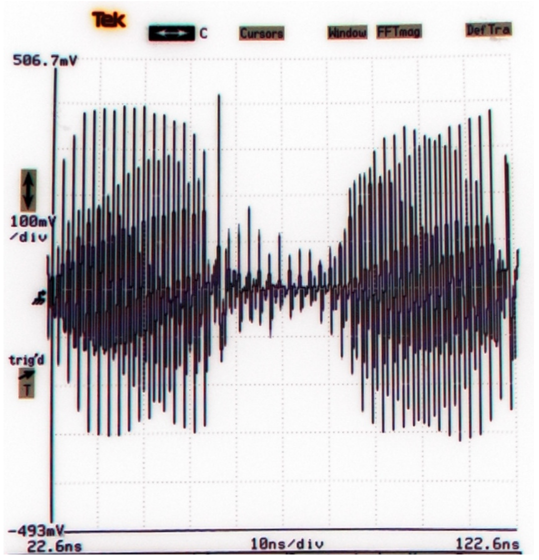
(b) Modulated spectrum

Figure 10.14 – Narrowband transmitter spectrum.

in Figure 6.4 on page 161 is not included in the circuit that generated the data shown in the figure. If it were, the carrier would be more sinusoidal. Some of the rest of the non-sinusoidal carrier is caused by the duty cycle of the output not being exactly 50%. Though it was designed to be 50%, process variations make it not, which causes odd harmonics.



(a) Time-domain carrier signal



(b) Time-domain carrier showing 16 MHz modulation

Figure 10.15 – Narrowband transmitter time-domain signal.

Figure 10.15(b) shows how the unwanted 16 MHz modulation appears in the time domain. It appears as a 16 MHz square wave modulated on the RF carrier. This is a bad case of the modulation. In many of the chips, and particularly at higher supply voltages, there would be little visible signs of the modulation at this time scale. However, in some chips, the modulation is even worse. Looking at the time-domain waveform, it is apparent that the 16 MHz digital clock is modulating the carrier in the same way that the desired data signal would. This then leads to the conclusion that this is where the modulation is getting injected. The digital module which produces the modulation signal is clocked by the crystal oscillator clock, so therefore the signal would have plenty of opportunity to contaminate the output.

Figure 10.16 investigates the link between the power supply voltage and the unwanted modulation which was alluded to previously. The figure shows the power of the unwanted spur (relative to the carrier frequency) as a function of the supply voltage. It is obvious that this modulation is greatly affected by the supply voltage, up to a point. Nevertheless, -40 dBc is good enough performance that it does not affect the testing of the rest of the transceiver, it would just cause FCC licensing problems! Therefore the rest of the transceiver tests can be carried out by running the transmitter at or above 1.8 V supply.

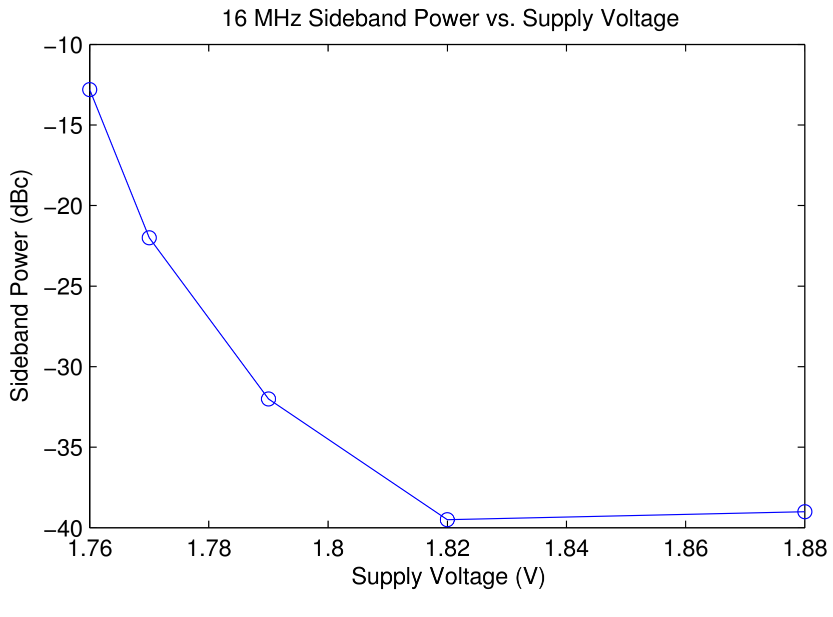


Figure 10.16 – Narrowband 16 MHz sideband power versus supply voltage.

Figure 10.17 shows the output power of the PA. The output power shown in the figure is while transmitting a packet, so the power for only transmitting a carrier is twice that because during a packet transmission it is off half the time. The maximum measured power is 2.2 mW which is less than the simulated 5 mW. Unaccounted for losses include the loss in the external LC tank and the added capacitance of this. The simulation also does not account for the unwanted modulation. In Figure 10.17, the unwanted modulation starts taking effect below 1.8 V supply and results in a steep drop off of transmitted power. Above 1.8 V, the curve behaves as expected, with a slight increase in transmitted power with increasing supply voltage.

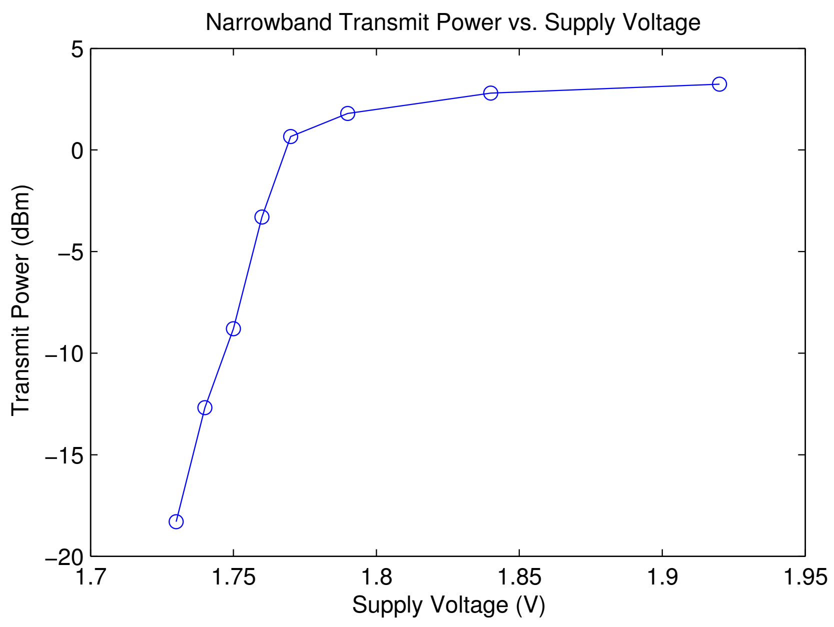
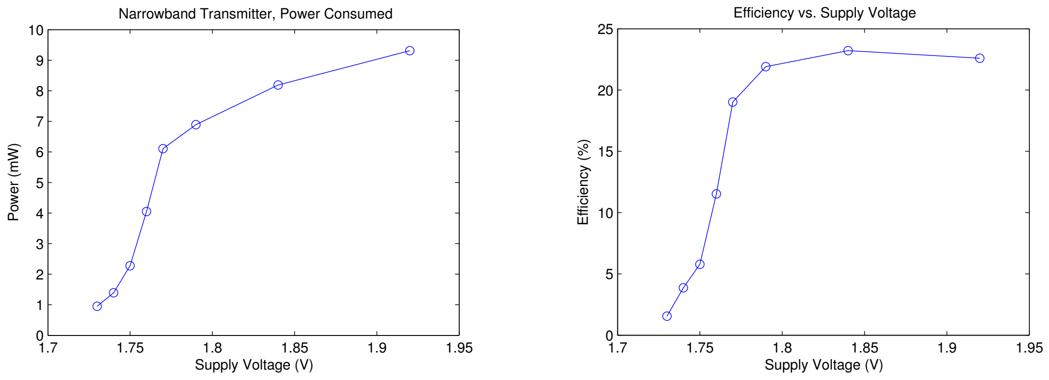


Figure 10.17 – Narrowband transmitter power output versus supply voltage.



(a) Power consumed versus supply voltage (b) Power amplifier efficiency versus supply voltage.

Figure 10.18 – Narrowband transmitter power consumed and efficiency.

Figure 10.18 shows in power consumed and the power amplifier efficiency versus supply voltage. Both curves show the effect of the unwanted modulation below 1.8 V. The efficiency shown in Figure 10.18(b) is somewhat lower than the maximum of 50% in simulation. This is most likely due to more board capacitance than expected, as well as the non-ideal off-chip inductors and capacitors. In particular, the parasitic capacitance of the inductor would hurt efficiency.

10.4 Narrowband Receiver

The chip implements a very simple narrowband receiver. The receiver was designed to have a range somewhat longer than the UWB transceiver. However, as the grant proposal promised narrowband communication in one direction coupled with UWB in the other, making the narrowband receiver long range was not the goal, as the system would be limited by the UWB range anyway.

The narrowband receiver works on most chips. It works at 500 kbps. Some specifications and measured power consumption figures are given in Table 10.10.

Table 10.10 – Narrowband Receiver Specifications

Area	
RF Front End Area:	0.213 mm^2
Analog Baseband Area:	0.091 mm^2
Digital Area:	0.129 mm^2
Total Area:	**0.433 mm^2**
Power	
RF Front End and Baseband Power:	2.79 mW
Digital Section Power (receiving packets):	0.88 mW
Total Receiver Power:	**3.67 mW**
Energy/bit:	**7.34 nJ**

The receiver power consumption is very good compared to others in the literature. However, as we will see, the performance is not. This tradeoff is due to the receiver architecture.

The first stage in the narrowband receiver is the SAW filter which performs the band and channel selection. The response of this filter sets the selectivity of the entire receiver. Figure 10.19 shows the response of the filter used.

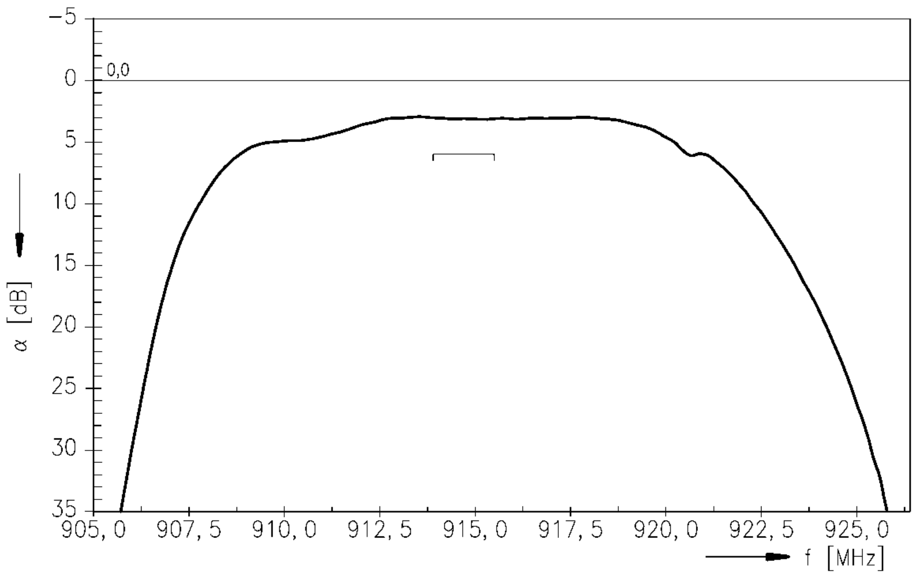
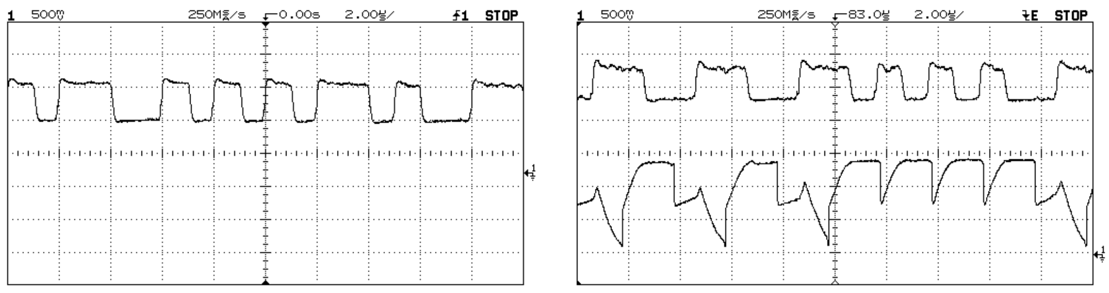


Figure 10.19 – Narrowband receiver SAW filter response.

The 3 dB passband loss adds directly to the 3 dB noise figure of the receiver, so the specifications of the SAW filter contribute to the performance of the complete system.

After the SAW filter, the signal goes through the LNA and gain stages before being squared, thus becoming a baseband signal. A scope capture of the baseband signal is shown in Figure 10.20(a) (for a case with good SNR). The waveform is very clean with the Manchester-encoded data plainly visible.



(a) Baseband signal.

(b) Baseband signal (top) with correlator signal (bottom).

Figure 10.20 – Narrowband receiver baseband and correlator signals.

Figure 10.20(b) shows the baseband signal and then the signal after the bit-detection correlators. (The differential signal has been converted to a single-ended version on-chip). In the correlator's signal, one can see the moment where the integrator is reset after which point it begins integrating. If the baseband signal is high, it integrates up, otherwise it ideally stays flat. In the second phase, if the baseband signal is high it integrates down, otherwise it stays flat. At the end of the symbol a comparator decides weather it is higher or lower than its starting point to determine which bit to select. The comparator has a positive bias, and the baseband signal is too large which causes it to saturate in the positive direction. The amplitude of the baseband signal can't be decreased due to an unrelated bug, so this saturating of the correlator must be tolerated. However, in a lower SNR case, the correlator would have much smaller signals and would not saturate. Furthermore, the integrator bias does not affect the decision, because it affects both the plus and minus signals of the differential signal chain together.

The slight timing inaccuracy in Figure 10.20(b) is not in the circuit itself, but due to measurement constraints. It is not possible to actually view both of those signals at the same time because they share a common readout channel. Therefore to produce Figure 10.20(b), one waveform was stored and reproduced on top of the other one which was gathered at a different time. The timing mismatch between the traces is due to the fact that they were not acquired together, but were instead acquired separately, with a slight timing error.

There is one main bug with the narrowband receiver. This is caused by 1/f noise causing problems in the AGC control loop. In particular, there are a couple stages in the AGC control loop which have very relaxed bandwidth requirements. Therefore, to save power, very low bias currents were used when designing these stages, however the noise contribution of these stages was never fully analyzed. After getting the chip back, it became apparent that something was causing the effective gain of the front end to shift, which was traced back to the 1/f noise in the AGC control loop. This noise is particularly bad in small devices, such as the 0.18 μm process used. A manifestation of the noise is shown in Figure 10.21.

The figure shows the AGC control line and the baseband signal. As can be seen, suddenly, for no apparent reason, the gain of the RF front end shifts lower, causing the amplitude of the envelope signal to get much smaller. The AGC control loop eventually detects this and increases the gain to restore the envelope. Then over time the AGC control moves to its original value. These gain fluctuations are constantly occurring and are due to 1/f noise in the AGC control loop circuits.

The best way to combat this is to have the AGC control loop attempt to keep the enve-

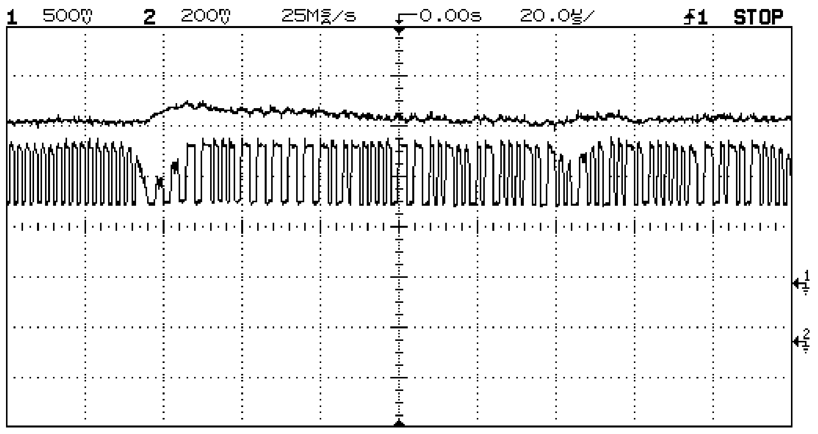
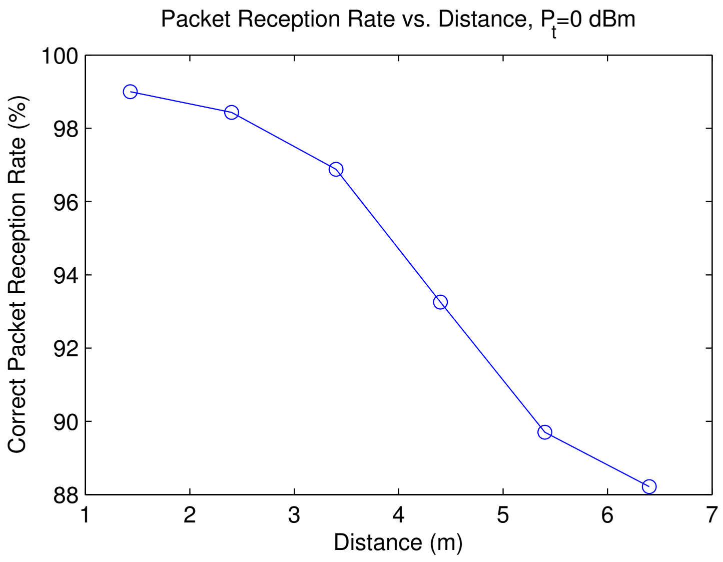


Figure 10.21 – Baseband signal (bottom) and AGC control voltage (top) showing sudden gain fluctuation.

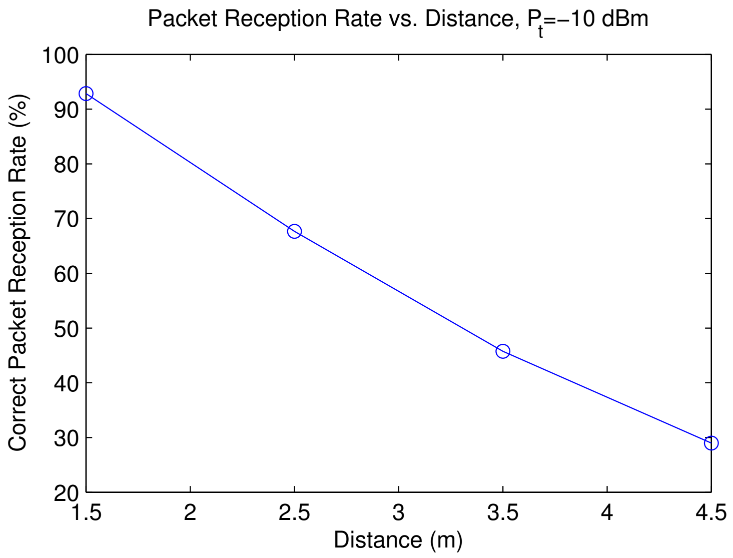
lope at a large value, even saturating. This allows the gain to drop unexpectedly and still have enough signal for proper receiver functioning. Occasionally the gain will drop so far that it causes an error, so this hurts the receiver performance. For example, the receiver should have no problem handling the drop in gain seen in Figure 10.21.

Testing the range of the transceiver is difficult in a fading channel, because of the log-normal fading. Therefore, to decrease variability, some tests were performed in an anechoic chamber. Though this is not indicative of a real-world environment, it is reproducible and deterministic. Figure 10.22(a) shows the initial testing in the anechoic chamber, but the length of the chamber soon became the limiting factor, so a 10 dB attenuator was installed on the transmitting antenna and the results are shown Figure 10.22(b). This promptly shows a sharp drop off in the correct packet reception rate. For the test, the packets were 12 bytes in length (10 data payload bytes), and no coding was performed on the link.

The measured sensitivity of the receiver is only -78 dBm, which is quite poor compared to other narrowband receiver designs, though the power consumption is good. The problem lies in the architecture itself. The receiver suffers from much noise. First, the SAW filter (the first in the signal chain) introduces 3 dB of noise figure. Second, and most importantly, the noise bandwidth of the receiver is large, much larger than the received spectrum. This is because all frequency selection is performed before the LNA, and the filtering later in the signal chain is not a high-Q filter like the SAW filter. Therefore, the noise introduced by the LNA and later stages does not get removed by later filtering stage. This is true for noise even far outside the bandwidth of interest. Instead it all gets squared and adds much noise to the signal. This is the biggest drawback of this receiver architecture.



(a) $P_t = 0$ dBm



(b) $P_t = -10$ dBm

Figure 10.22 – Correct packet reception versus distance–anechoic chamber.

Chapter 11

Overall System Results

This chapter presents results relating to other circuits not presented already such as the performance of the processor, power supplies, and clock generator. A comparison between similar commercially available motes is also presented.

Section 11.1 presents the measured results of the processor core and digital circuitry. Most important is the power consumption. Section 11.2 presents the results of the supporting circuitry such as the power supplies and the clock system. Finally section 11.3 presents a comparison of the complete system with other similar systems.

11.1 Digital Core

The digital core has changed much over the chip revisions. In the latest chip, it is a MSP430X clone with an extensive set of custom peripherals. The power consumption of the core, both in the static and dynamic case are of critical importance to the power consumption of a complete sensor node. Both the static and dynamic power of the core can be minimized with the use of DVS to save power.

The area breakdown of the entire digital section is given in Table 11.1. The table shows the area of individual modules, and—in cases where there are multiple copies of the module—the total area. The majority of the area is in the hard memory IP blocks.

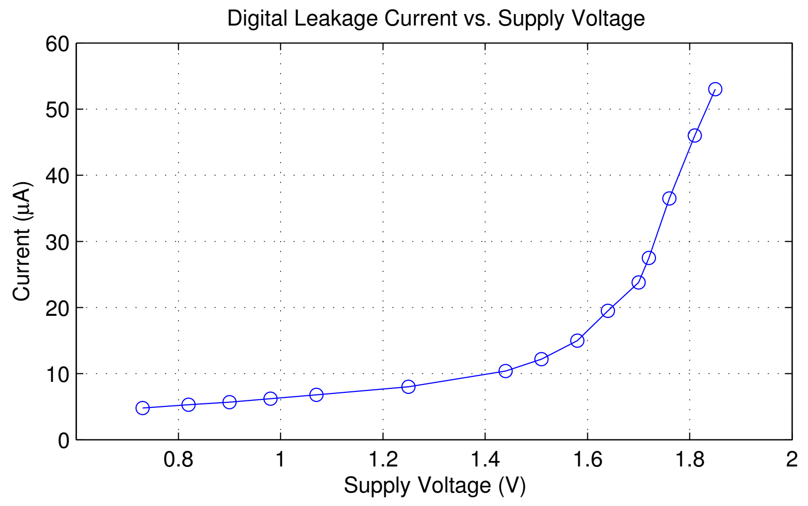
The entry "System Module" contains a number of functions including the CSA interface and FIFOs, the reset controller, watchdog, and a large number of different miscellaneous functions which all got grouped in that module. The entry "Overhead" is the logic required for the bus interface to interconnect all the modules as well as some testing logic.

Table 11.1 – Digital Section Area

Description:	**Area (mm^2)**		
	One	**#**	**Total**
Memory IP			
16 kB SRAM:	0.81	6	4.83
512 byte dual-port reg. file:	0.09	4	0.36
4 kB ROM:	0.084		
Sub-total:	5.28		
Synthesized			
UWB Receiver:	0.503		
System Module:	0.14		
MSP430X:	0.128		
Hardware Multiplier:	0.062		
Clock Generator:	0.053		
UWB Transmitter:	0.045		
Programmable I/O Controller:	0.041		
Narrowband Receiver:	0.038		
I^{2}C Module:	0.035		
$\Delta\Sigma$ ADC:	0.031		
Timer	0.024	4	0.096
Overhead:	0.022		
UART:	0.0165	2	0.033
Narrowband Transmitter:	0.016		
SPI:	0.013	2	0.026
Sub-total:	1.27		
Grand Total:	**6.55**		

The total area of the digital section is 6.55 mm^2 with 80% of that area consumed in the memories. The CPU core itself requires only 2% of the area.

Figure 11.1 shows the leakage (static) current and power consumed by the entire digital section of the fourth chip. This includes the processor, memory and all peripherals. What is immediately obvious is that one pays a high leakage power penalty to operate the device above 1.5 V. In this regime, the E-field in the channel is large enough for effects such as drain-induced barrier lowering (DIBL) to affect leakage, while below that much more moderate leakage processes are seen. The leakage can easily be less than 10 μW which is good for battery life.



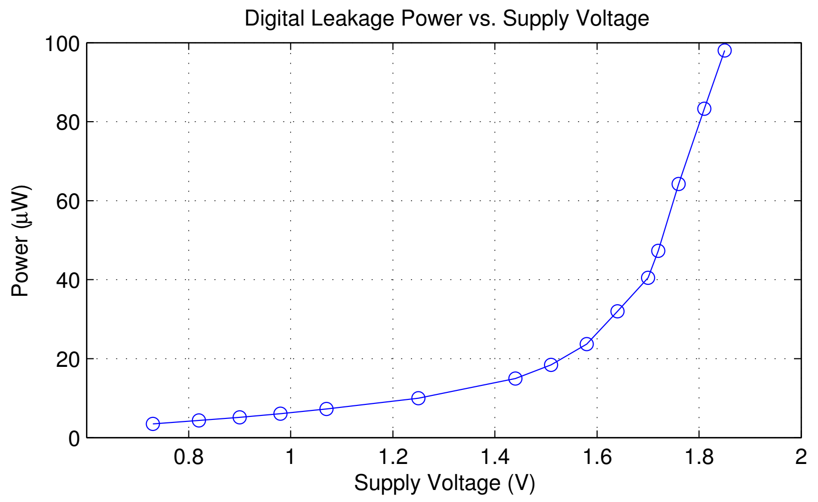


Figure 11.1 – Measured leakage current and power consumption versus supply voltage for the fourth chip.

Previous chips had more and less leakage. The measurements on the second chip indicated more leakage, and it was discovered this was mainly due to some circuits in the dual-port RAM used (that is IP, so it is not possibly to know what in them was consuming the power). For the third chip a dual-port register file was used with lower leakage. This chip then had the lowest leakage. The last chip doubled the RAM, which is a major source of leakage, so the leakage increased, though not significantly.

Figure 11.2 shows the dynamic current and power consumption of the MSP430X processor core as a function of supply voltage (included is the power consumed by the buses and memories). As predicted, the current versus voltage shows a linear relationship while the power versus voltage is quadratic due to the use of the buck regulator.

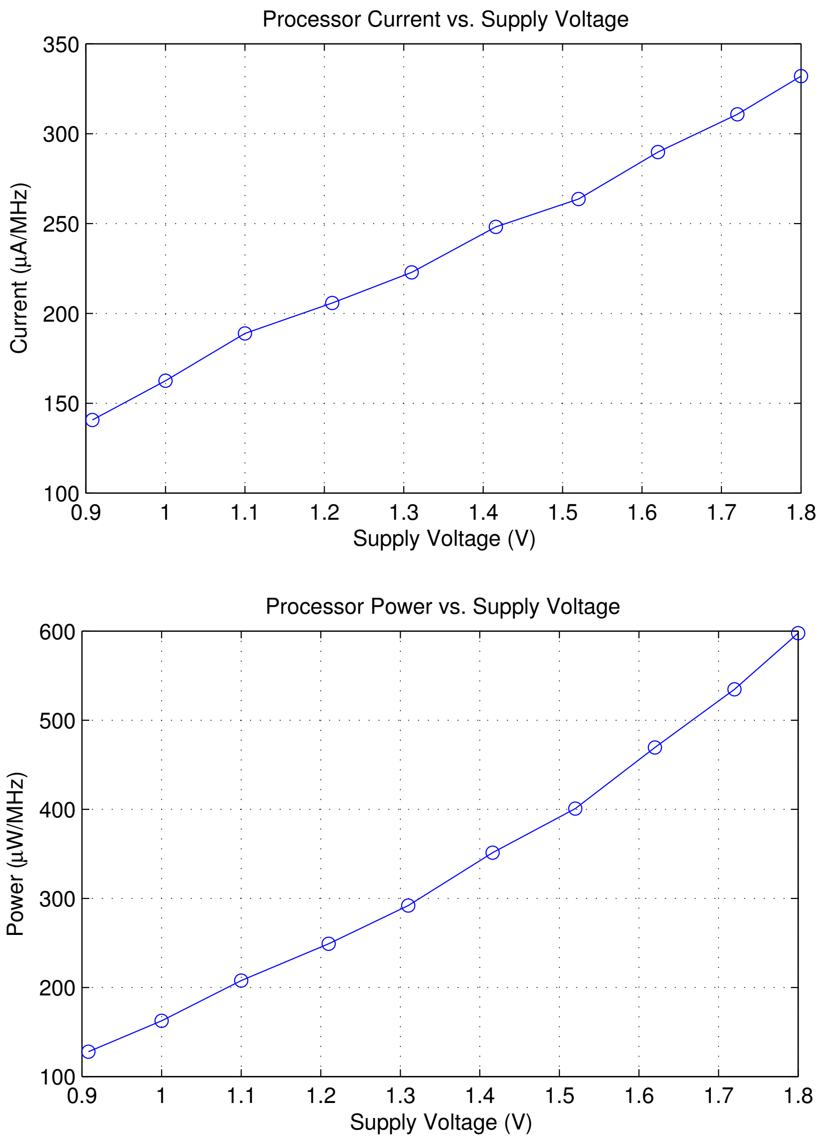


Figure 11.2 – Measured processor dynamic current and power consumption versus supply voltage for the fourth chip.

As the voltage is decreased, the maximum frequency also decreases. This is shown in Figure 11.3. The maximum frequency when operating at 1.8 V is 55 MHz, but drops to 12 MHz at 0.9 V. Therefore, a tradeoff is involved between processing speed and energy consumption. Nonetheless, a minimum processing speed of 12 MHz is still as fast or faster than many of the commercial wireless sensor network motes available.

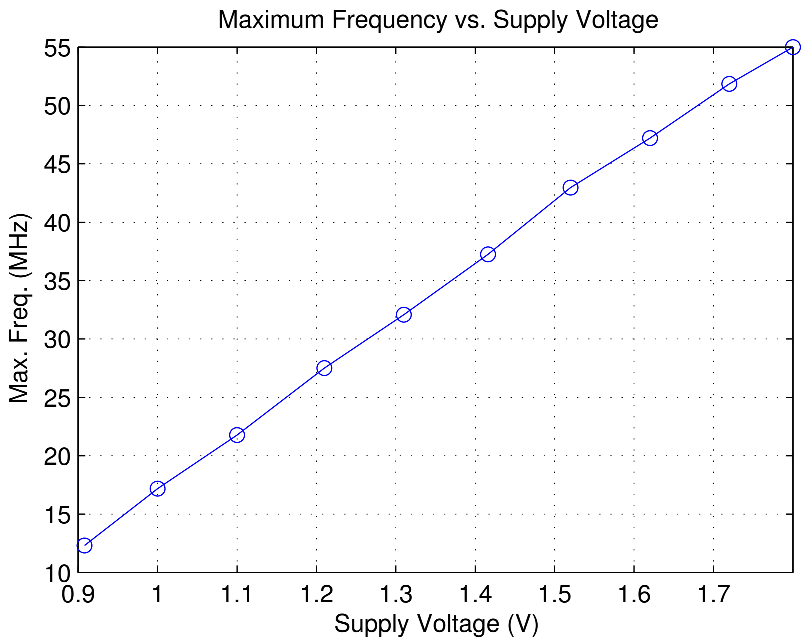


Figure 11.3 – Maximum frequency versus supply voltage for the fourth chip.

The previous power graphs assume a 100% efficient power supply. This is not the case, though for a buck regulator a high, approximately constant efficiency, the results are similar. One difference is the addition of a fixed loss term, which in this case is about 2.8 μW due to the buck regulator quiescent current. However this fixed loss term is not compatible with the practice of measuring the dynamic current of the digital section in μW/MHz, so it was ignored in the previous graphs. In practice, the fixed loss term would be added on to the resulting power consumption, however it is small enough that for the dynamic power it is negligible.

The finite efficiency of the regulator causes the current to increase. So if the efficiency is 95% efficient, the total power would be divided by 95% to find the actual power.

The use of the buck converter makes the processor much more efficient than the use of a linear regulator such as an LDO. To illustrate this, Figure 11.4 shows what would happen to the power consumption if an LDO was used versus the buck. In the figure, the quiescent current of the LDO and the buck converter were ignored so the units on the Y axis could remain in μA/MHz. The quiescent current of both types of converters is similar, and this current manifests itself as a constant power loss term and does not really affect the result.

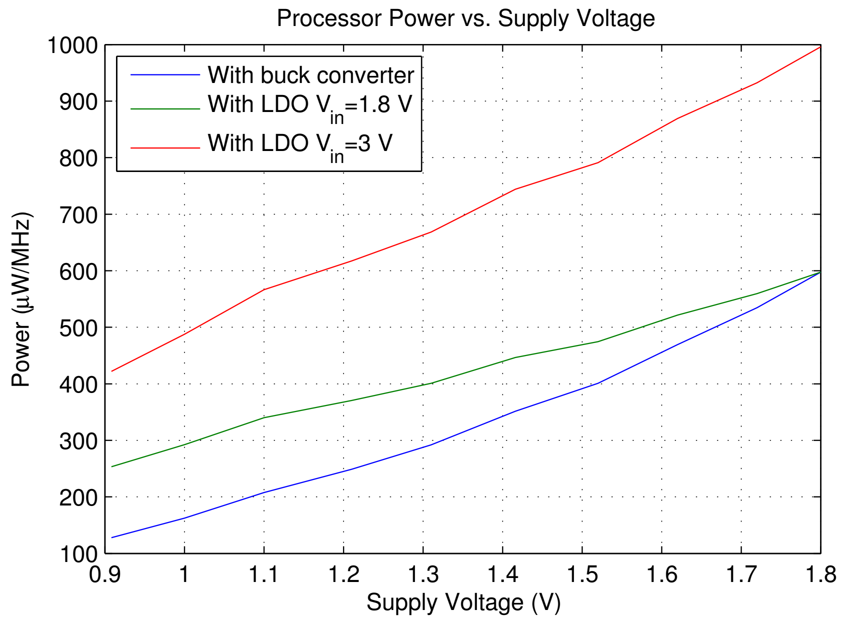


Figure 11.4 – Comparison between using an LDO and buck regulator to power the digital core.

In Figure 11.4, two LDO curves are shown. One, the $V_{in} = 1.8V$ curve, is the best case with an LDO where the output is the same as the input when the output is maximum (in actuality, the input must be slightly higher by the amount of the dropout voltage, but we will ignore that here). In this case, the power consumed using a buck or LDO is the same at the maximum output voltage, but the buck quickly out-performs the LDO at lower output voltages. This case is not typical, because it requires the input to be a fixed voltage of the precise right voltage. Usually the input is a battery, so this is not the case. The second case, which is the typical case, is where the input is a 3 V battery, a very common value in sensor network nodes. In this case the efficiency of the LDO, even when supplying 1.8 V is only 60% and it drops from there. Here the buck converter works even better. The buck converter performs the same whether the input is at 1.8 V or at 3 V. Once again, as with the LDO, the dropout voltage of the buck regulator must be met so the input voltage must be slightly larger than the output voltage.

11.2 Support Circuitry

Other major circuits which are required to complete the system include the power supplies and the numerous clocks and oscillators. A high efficiency power supply is critical to attain low power consumption of the digital section as was seen the previous section.

11.2.1 Power Supply

Two buck power supplies were integrated on the chip, one for the digital supply, the other for the analog supply. They are included in the pad frame, so they take little space out of the core area of the chip.

The process supports Schottky barrier diodes, so they were used to implement the protection diodes on chip. The feedback resistors were also integrated on chip, and for the last chip they were made digitally adjustable to be able to trim out the process variations which caused much supply voltage error in the earlier third chip. With this addition, it is generally always possible to make the regulators operate at their desired output voltage. With pads and associated ESD protection, the design occupies 0.156 mm^2. The no-load input current is 960 nA. This low current assures very high efficiency at light loads.

Figure 11.5 shows the measured efficiency as a function of load current. As can be seen, the regulator retains a high efficiency even at very low loads. The peak efficiency is over 91% and it stays over 80% efficient from 20 μW to 40 mW.

The measured efficiency is somewhat lower than the simulated efficiency in Figure 7.32 on page 210, particularly when under higher load. This high-load drop off would indicate higher I^2R losses than anticipated. This may be due to higher DCR in the inductor than in the simulation. Another factor would be the core losses in the inductor which were not well modeled in the simulation. The core losses in the inductor would be most prominent at high ripple currents, which are present under high loads.

The efficiency also varied from chip to chip. This was due to the effects of process variations on the zero current comparator (Cmp2 in Figure 7.24 on page 202). Process variations on this comparator affect when the lower MOSFET turns off. In some chips it turns off too quickly, and the full benefit of synchronous rectification is not realized which decreases efficiency, and in others it turns off too late allowing the current in the inductor to reverse which also decreases efficiency. Figure 11.6 shows a scope capture of a switching event when the comparator turns off too early. The figure shows the voltage at the switching node which is the node connected to the two power MOSFETs and the external inductor.

In the figure, the first high pulse is the upper MOSFET turning on which ramps up the current through the inductor. This MOSFET turning on pulls the voltage to the 3.3 V supply. Next this MOSFET turns off and the lower MOSFET turns on which pulls the voltage to ground. The current is flowing out of the drivers, so the voltage actually goes

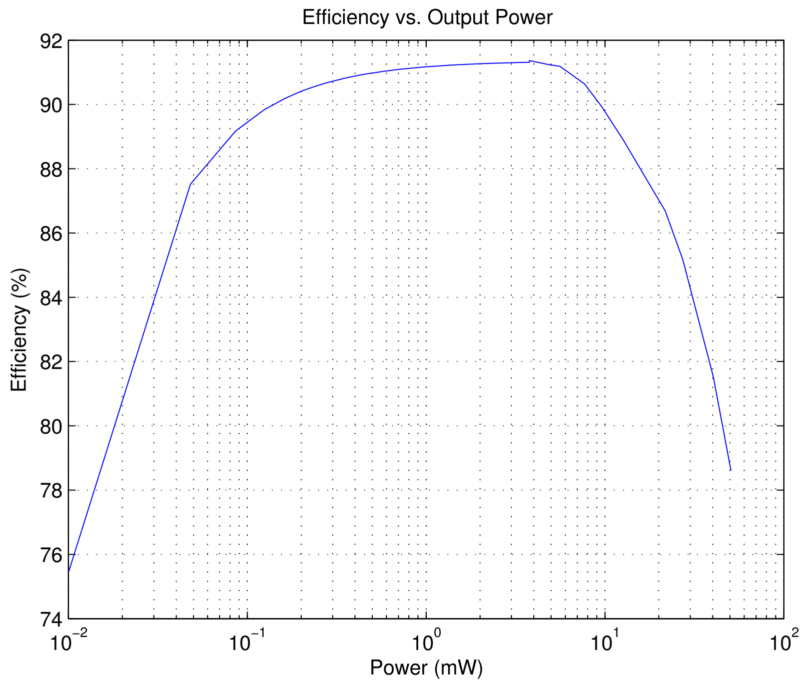


Figure 11.5 – Measured efficiency as a function of output power.
BFC = 20 μF BFL = 330 μH Vin = 3.3V Vout = 1.8V

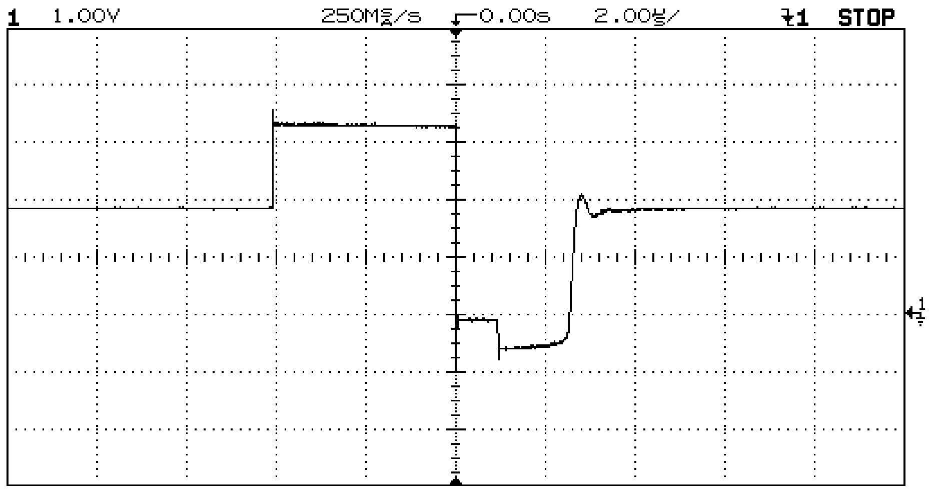


Figure 11.6 – Switching event when the lower MOSFET turns off too quickly.

below ground. Then, midway through this pulse, the zero current comparator triggers and shuts off the lower MOSFET, but in this case it is too early because there is still current flowing through the MOSFET. At this point the remaining current can not flow through the MOSFET, so it instead flows through the lower protection diode. However, this diode has a higher forward voltage drop across it than the on MOSFET, so the voltage becomes more negative for a time before the current finally dies down and the voltage settles to the output voltage to await the next switch. The extra loss is caused by the extra voltage drop that the diode imposes which could have been mostly avoided if instead the MOSFET was on during that time.

Figure 11.7 shows a scope capture of the waveform when the zero current comparator turns off too slowly. In this case, the lower MOSFET turns on correctly, but stays on even after the current goes to zero through it. Then the current starts to reverse and flow back from the output node to ground. Eventually the MOSFET turns off. Then the reverse current has no place to go but through the high-side protection diode back to the source. This forces the voltage to increase a diode drop above the source voltage of 3.3 V for a short time until the current becomes zero, at which point the voltage then slowly floats back to 1.8 V for the next switching cycle.

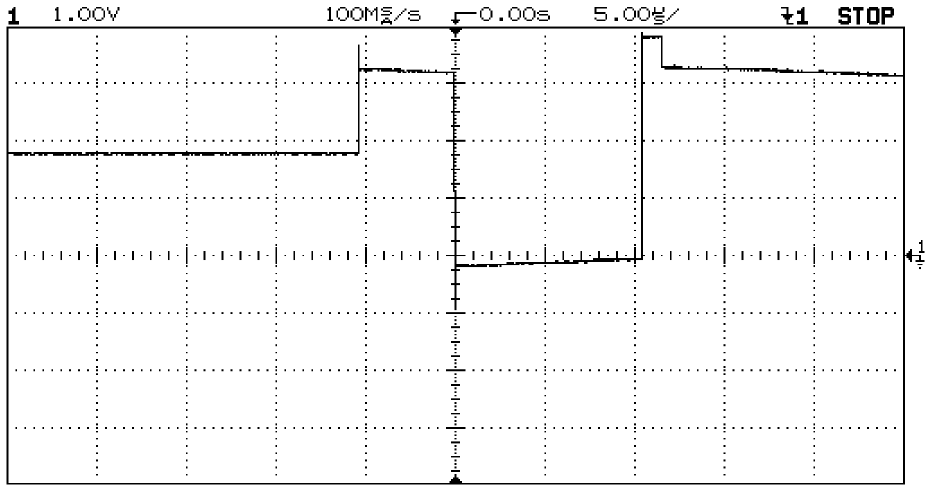


Figure 11.7 – Switching event when the lower MOSFET turns off too slowly.

During design, it was assumed that the comparator would have some offset and therefore on virtually every chip the comparator would not switch at the precise right time. This does not affect efficiency much, because as long as the comparator switches when the current through the inductor is low, the losses are minimized. In both of the two cases, the switching power supply is operational. In fact, though the comparator offset did greatly affect each chip, and in some cases affected efficiency, the power supplies were one of the most reliable sections on the chip—they both worked on every chip tested.

The switching frequency of the converter is variable and depends on the load conditions. Even with a fixed load, however, the switching frequency is not constant from cycle to cycle due to noise (1/f) in the voltage regulation comparator. Figure 11.8 shows a longer time scale of the switching events as well as the output voltage. The output voltage has some ripple as expected due to the switching events, and as can be seen the cycle-to-cycle jitter is quite large. This does not really affect any parameters of interest, with the possible exception of ripple voltage, and this only slightly.

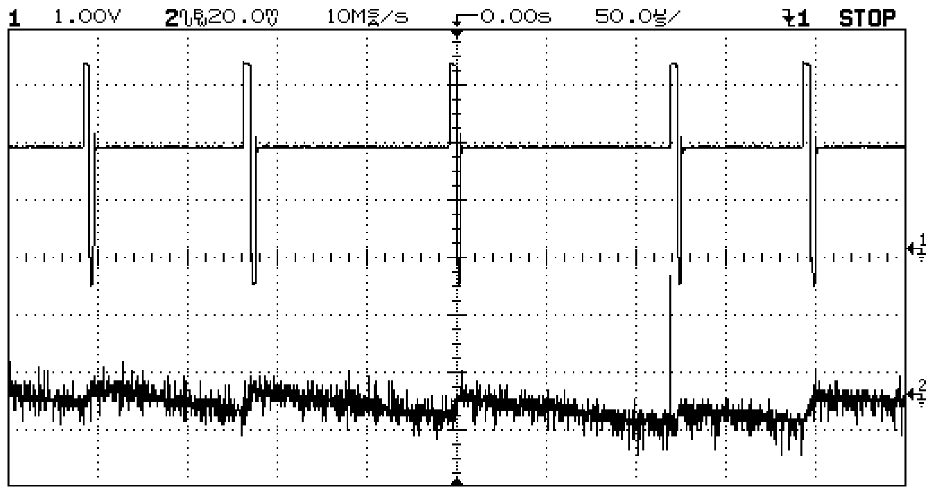


Figure 11.8 – Switching events (top) and output voltage ripple (bottom).

The digital buck converter was able to supply any voltage which the digital section desired. It works with the processor to ramp up and down the supply as needed when the processor goes into and comes out of sleep.

11.2.2 Clock Generator

The clock generator section includes many different oscillators with varying performance and frequency characteristics. Some are designed to be low-power fast-starting oscillators suitable for clocking the processor or possibly UART communication, while others are designed for low jitter and RF communication use. The low frequency crystal oscillator is designed as a low-power time base for the mote, but is too slow and has too much jitter to be used as the reference for the frequency synthesizer. Therefore, a high frequency crystal is used for this purpose.

The digitally-controlled oscillator is an oscillator with fine digital control and with low-power and fast-startup performance. It can be set to nearly any frequency, and can be stabilized with the low frequency crystal. This is a good general purpose clock for lower performance applications.

The constant frequency oscillator runs on the digital supply, but is supposed to be invariant of the supply voltage. It is also a good basic clock, but does not have as much frequency adjustability as the DCO. The variable frequency oscillator is a ring oscillator running off the digital supply whose frequency scales with voltage. It is convenient to run the processor off this if using DVS so the software does not have to worry about changing the processor clock frequency when changing the supply voltage. The peak-to-peak jitter measurements are shown in Table 11.2. Though other measures of jitter may be more important, this is the only jitter measure which could be performed due to the limited availability of test equipment. The high-frequency low-jitter measurements may be in error due to difficulties of accurately measuring small jitter on an oscilloscope which is not quite capable enough. This is the case for the VCO jitter measurement where an accurate measurement could not be determined.

Table 11.2 – Clock Peak-to-Peak Jitter

Clock:	**Measured Freq.**	**Jitter (ns)**	**Jitter (%)**
LF Crystal:	32.768 kHz	300	1
HF Crystal:	16 MHz	0.3	0.48
DCO, Range 0:	43.3 MHz	1.2	5.2
DCO, Range 1:	12.5 MHz	6	7.5
DCO, Range 2:	5.4 MHz	10	5.4
DCO, Range 3:	2.66 MHz	23	6.1
Constant Freq. Osc:	34 MHz	1.3	4.4
Variable Freq. Osc:	34 MHz	1	3.4
VCO Divided:	57 MHz	<0.1	

Table 11.3 gives the current consumption measurements of the various oscillators.

The constant frequency and variable frequency oscillators are difficult to separate from the power consumption of the processor. For this reason, their power consumption was not

Table 11.3 – Clock Current Consumption

Clock:	**Current:**
LF Crystal, $C_l = 10pF$	$0.22\ \mu$A
LF Crystal, $C_l = 20pF$	$0.3\ \mu$A
LF Crystal, $C_l = 30pF$	$0.45\ \mu$A
HF Crystal, V_{amp} normal	$48\ \mu$A
HF Crystal, $V_{amp} = 900mV$	$453\ \mu$A
DCO, Range=0	$2.2 + 0.33F_{DCO}\ \mu$A[a]
DCO, Range=3	$2.5 + 1.2F_{DCO}\ \mu$A[a]
PLL—no digital clock divider, i.e. PLLClkEn =0, $F_{VCO} = 912$ MHz, $I_{chg\ pump} = 31\ \mu A$, $I_{VCO} = 666\ \mu A$	$955\ \mu$A[b]
PLL digital clock divider, i.e. additional current when PLLClkEn =1, $F_{VCO} = 912$ MHz	$320\ \mu$A

[a] F_{DCO} in MHz.

[n] Does not include crystal oscillator current.

measured individually. They are lower power than the DCO. Their power consumption was included when making Figure 11.2.

The power consumption of the low-frequency crystal oscillator changes with load capacitance because the g_m required to maintain oscillations increases with increasing load capacitance thereby requiring a larger bias current.

The high frequency crystal oscillator had a bug in many chips relating to the amplitude control circuit. In many chips, the amplitude control circuit would decrease the current too aggressively and either result in very small oscillations which cause poor jitter performance, or it would completely quench the oscillation. A simple fix for this is to externally override the amplitude control voltage externally with some voltage. In this case the 900 mV reference was a convenient voltage. This disables the amplitude control circuit and increases the power consumption greatly. So the two power consumption numbers are when not using the amplitude control override and when using it. If the oscillator did not have the bug, the power would be a little larger than the "normal" case.

Figure 11.9 shows the low frequency crystal oscillator starting. The top trace is the amplitude control voltage which affects the current going through the amplifier and therefore the g_m and power consumption. It starts high, using much current (5-10 μA), and goes low after the oscillator oscillations build up in magnitude. The steady-state current is only 300 nA.

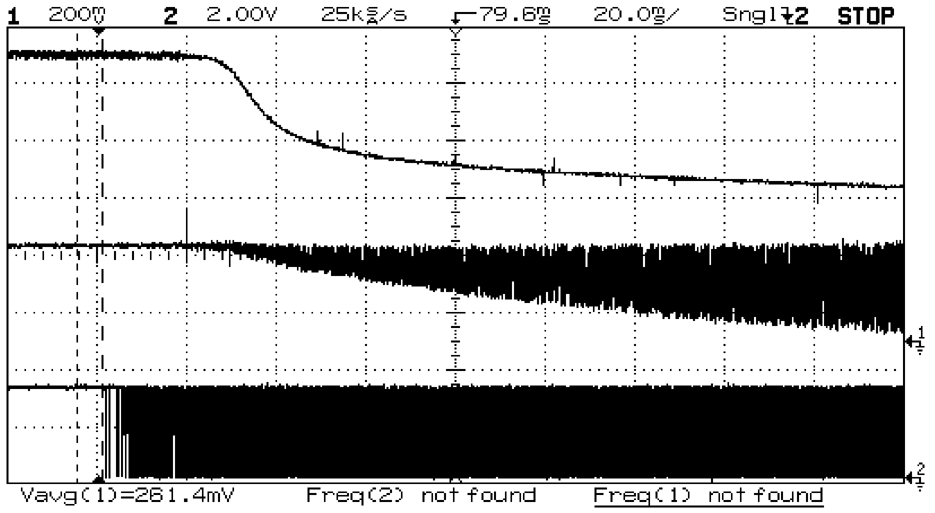


Figure 11.9 – Low frequency crystal oscillator starting, top (500 mV/div) is the amplitude control voltage, middle (200 mV/div) is the amplitude of the oscillator, bottom (2 V/div) is the digital output.

The amplifier which produces the digital waveform starts producing the clock well before the amplitude of the oscillations have stabilized. These clock cycles can be used, but they are not as accurate as after the oscillator has stabilized.

11.3 Overall Sensor Mote Comparisons

The previous chapters and sections have discussed each and every part in the system and have presented their power consumption and performance specifications. Each one is low power and can individually lower the system power consumption; however, when the savings combined a very large power savings can be accrued. Table 11.4 shows a comparison among other sensor motes showing the most important specifications of a sensor mote. Most of these have to do with power consumption of the system. A quick glance through the table shows the presented work provides significant power savings in virtually every category.

Table 11.4 – Mote Hardware Comparison

		This Work	**MicaZ**	**TelosB**	**Iris**	**CC430F5137**	**SunSPOT**
Processor	**Processor Name**	MSP430X	ATmega128L	MSP430	ATmega1281	MSP430X	ARM9
	Bits	16/20	8	16	8	16/20	32
	Freq. (MHz)	12-55	8-12	8	8	8-16	180
	Dynamic I (μA/MHz)	140-330	1000	500	800	170-220	483
	Dynamic P (μW/MHz)	130-600	3000	1500	2400	500-660	1740
Mem.	**Program (kB)**	96 Shared	128	48	128	32	4 MB
	Data (kB)	96 Shared	4	10	8	4	512
	Logging (kB)	128	512	1024	512	512	
Radio	**Data Rate (kbps)**	1000-5000	250	250	250	250	250
	Tx Pwr (mW)	2	52	52	51	54	52
	Tx Energy/Bit (nJ)	0.46-2.3	209	209	204	216	209
	Rx Pwr (mW)	17.6	56	56	48	45	56
	Rx Energy/Bit (nJ)	3.5-17.6	226	226	187	180	226
	Range (m)	8	200	100	300	200	30
Sleep	**All Off (μW)**	6.8	2	2	6	6	
	Timekeeping (μW)	14	30	15	24	8	115

The row “All Off” is the absolute minimum power consumption the mote can take if everything is powered off, including the slow crystal oscillator. This is not too useful as it is only possible for an external event to wake the processor, not an internal timer. This is usually the leakage, however for this work this also includes the current for one buck converter.

The MicaZ, TelosB, and Iris motes are the most commonly used motes which are targeted for similar applications as this mote. For their power consumption figures, it is assumed they operate directly off a 3 V battery. None use dynamic voltage scaling, so they suffer from poor processor power consumption. Many of these motes use the same radio transceiver chip from TI and therefore have the same radio power consumption figures.

The entry labeled “CC430F5137” is not a commercially available mote, however, this chip is a new RF SoC available from TI which includes a processor and low-power transceiver on a single chip and would be particularly attractive to design a sensor node around, so it is included here as if a sensor node has been designed around it. It includes an on-chip LDO which allows primitive dynamic voltage scaling, so it operates with much less power than the other motes. However, because it uses an LDO and the presented work uses a buck converter, it cannot compare to the presented power consumption. Nonetheless, it is a very attractive option for a wireless sensor network, particularly if it is later offered with more program flash memory.

The SunSPOT is a much more powerful mote which was included to compare the processing power consumption of a powerful ARM processor. Though with a 180 MHz 32-bit processor it is not in the same processor class as the presented work. However, the presented work can still beat the power handsomely in power per MHz. The SunSPOT uses dual buck regulators to supply the processor, though it does not use DVS. They are able to get reasonable sleep power consumption by disabling the buck regulators entirely when in sleep mode and using an LDO to supply the SRAM and powering off everything else.

The presented data shows the much lower power consumption attainable with the presented mote hardware over other commercially available hardware. The improvement in virtually every category shows the innovative solutions pursued had the intended effect of dramatically reducing power consumption.

11.4 Conclusion

This chapter presented the measured results of the processor, power supplies, and clock generator. The measured processor power consumption was seen to be very good when using the on-chip buck regulators. The on-chip buck power supplies were functional with good efficiency, and the clock generator included a variety of clocks with different energy consumption and jitter performance.

A summary and comparison with other sensor networks shows the substantial performance and power gains which can be achieved using this chip in a wireless sensor network. In most categories the presented work shows over an order of magnitude improvement in power consumption.

Chapter 12

Conclusions And Future Work

This chapter presents some future work which could be pursued relating to the wireless sensor network as well as summarizing and drawing some conclusions from the work.

12.1 Future Work

Future work could focus broadly on hardware improvements or software improvements. There are a number of ways the hardware could be improved, all of which require designing a new chip. The software was not the focus of this dissertation, so no work was done in that area. Nonetheless, much work could be done with the existing hardware to accomodate it in a wireless sensor network.

12.1.1 Hardware

Improvements could be made to the integrated buck power supplies. These improvements could improve the startup-time, therefore decreasing the interrupt latency when bringing the supply from a lower to higher supply to exit sleep, or these improvements could focus on a better control scheme.

Future work on the digital core could focus on the energy improvements gained by using a pipelined processor. Using a pipelined processor has not been done in previous wireless sensor network hardware mainly because there was no efficient way to trade off maximum frequency for energy consumption when using conventional LDOs. Without this option, processor pipelining only increases energy consumption (of course accompanied by the large increase in maximum processor frequency). However, with the efficient DVS implemented, pipelining can save energy because the high processor frequency can be traded off efficiently for energy savings.

Other digital power saving techniques could also be pursued. For example, using substrate biasing, or sub-threshold dynamic voltage scaling are promising techniques of reducing the sleep and dynamic power, respectively. Both of these require some non-standard steps in the digital synthesis flow.

Numerous UWB implementation improvements could be pursued. Basic improvements to the UWB receiver architecture are continually being researched and will appear in the research in the future. Another major improvement which is possible with UWB but was not implemented in this receiver is the possibility to do time-of-flight ranging because the short time domain pulses allow extremely good timing information to be gleaned. Accuracy down to the centimeter range has been demonstrated [61,99,100]. Another method to save power would be to switch bias currents on and off at the symbol rate. Because UWB signals are only present for a short period, it is also possible to switch the bias currents in the RF front end off when not in use. This requires fast starting circuits in the RF section, but can be done to conserve power [99].

Other novel UWB hardware which could be implemented includes a full-duplex UWB link. This could be achieved by interleaving the pulses sent from each node. For example, node 1 would send the first pulse to node 2, while node 2 would send the second pulse to node 1. This full duplex link may not have too many uses as such, as most of the time data sent in a WSN is unidirectional, but it could have some interesting applications. For example, it could be used as a real-time hardware acknowledge in such a way that an optimal channel coding is always performed. In this scheme, the data would only be going one direction, however the reverse channel could be used to tell the transmitter when the receiver has enough information to fully decode the packet. The communication could proceed as follows: The sending node first sends the transmitted packet, then begins sending coded check bits, but after each N bits, it checks to see if the receiver can correctly decode the packet. This reverse information is provided by interleaving with the transmitted pulse. The transmitter continues to send more redundant check bits until the packet can be correctly decoded at the receiver. In this way, all packets will get a strong enough code, but will not waste power sending more check bits than necessary.

12.1.2 Software

This dissertation did not focus on the software required to take full advantage of the low-power features of the designed wireless sensor network node. Due to its unique features, some different software and algorithms should be developed to take full advantage of the hardware.

For example, TinyOS is used in many COTS solutions. This could be ported to work with this chip, however new interfaces and algorithms would need to be developed to support and make optimal use of DVS. This is a well studied problem in the context of complex multi-tasking operating systems, but simple versions of these algorithms may need to be developed to work with the TinyOS environment. Furthermore, the possibility of using a lower sleep mode voltage than operating mode voltage can be investigated to lower power. This is not a well studied problem because other systems do not do this. Lowering the sleep mode voltage decreases sleep power at the expense of interrupt latency. An intelligent algorithm to determine when this is possible without missing a deadline would not be trivial.

Substantial changes would be required to accommodate the peculiarities of the UWB transceiver in the medium-access control layer of the protocol stack. The most significant changes are required because the UWB receiver has no carrier sense. It is possible that an unreliable one could be added in the hardware in a future version, though it would always suffer from some unreliability due to the very small signal amplitude of the UWB signals. Furthermore, the implementation of it is difficult. Presently, virtually all MAC schemes rely on carrier sense to determine when the channel is free and avoid collisions. For the UWB transceiver, a more synchronized approach would have to be taken to lower the collision probability.

A further modification of the MAC which could take place is to decrease the reliance on the receiver listening for a packet over a length of time, but instead have the transmitter transmit the packet repeatedly over that length of time and the receiver wake up for a much shorter period. This decreases power because the transmitter has much less power consumption than the receiver does. However, having the transmitter on for a longer duration increases the probability of a collision, thus revealing a tradeoff.

12.2 Conclusion

The wireless sensor network node presented in this dissertation is a great improvement over other COTS solutions in virtually all areas. Not only is the processing power consumption and communication power consumption down by at least an order of magnitude, the maximum processing frequency is greater allowing more complex algorithms to be implemented in software while still only requiring minimal power. The features of the presented work are as follows:

- The use of low-power dynamic voltage scaling is the main driving factor for lower digital processing power consumption, and the buck regulator's high efficiency at light loads maintains the low sleep power. The dynamic power consumption is down over an order of magnitude from similar commercially available motes.
- The charge sensitive amplifier and event-driven front end allow a radiation sensor to be interfaced to the detector. Unlike many other sensors used in WSNs, this sensor must be continually monitoring, so the power of this is critical to the power of the complete system. The low power CSA and event driven back end allows for much lower power consumption than competing systems while still maintaining a high level of functionality as a multi-channel analyzer.
- The UWB transceiver is a revolutionary technology in WSNs which promises much lower power consumption and higher data rates, albeit at a shorter range. The transceiver offers an order of magnitude improvement in data rate, an order of magnitude decrease in receiver energy, and two orders of magnitude decrease in transmit energy. This is facilitated by a novel baseband architecture which can reliably acquire the signal and receive packets in a much more noisy environments than previously published architectures.
- A narrowband transceiver was implemented with a very low power receiver. Though the architecture has poor sensitivity, the low energy consumption when compared to competing systems and the relative simplicity make it attractive for short-range applications.
- The fully integrated solution demonstrates a much lower power consumption than competing multi-chip solutions along with a small form factor. The test board de-

signed has node-sized dimensions which demonstrate feasibility of making a wireless sensor node from the chip.

The front end radiation detector was shown to work in a number of different field tests with different detectors, and all other major functionality of the device was verified. The measured results show the validity of the design and the approaches taken.

Though improvements could be made to a number of the parts of the mote, overall the performance of the mote far exceeds the present state of the art. The use of such motes in future wireless sensor networks will allow them to be used in countless new applications. The presented hardware should extend the battery lifetime of present wireless sensor network applications from weeks and months to years.

Bibliography

[1] D. Puccinelli and M. Haenggi, “Wireless sensor networks: applications and challenges of ubiquitous sensing,” *Circuits and Systems Magazine, IEEE*, vol. 5, no. 3, pp. 19–31, Sept. 2005.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[3] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in *WSNA ’02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 88–97.

[4] E. Biagioni and K. Bridges, “The application of remote sensor technology to assist the recovery of rare and endangered species,” *International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 315–324, Aug. 2002.

[5] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, “Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet,” in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*. New York, NY, USA: ACM, 2002, pp. 96–107.

[6] D. Anthony, P. Bennett, M. Vuran, M. Dwyer, S. Elbaum, and F. Chavez-Ramirez, “Simulation and testing mobile wireless sensor networks,” in *ACM Int. Conf. on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (ACM MSWiM ’10)*, Bodrum, Turkey, Oct. 2010.

[7] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, “Monitoring volcanic eruptions with a wireless sensor network,” in *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005*, Jan 2005, pp. 108–120.

[8] J. Burrell, T. Brooke, and R. Beckwith, “Vineyard computing: Sensor networks in agricultural production,” *IEEE Pervasive Computing*, vol. 3, pp. 38–45, 2004.

[9] I. F. Akyildiz and E. P. Stuntebeck, “Wireless underground sensor networks: Research challenges,” *Ad Hoc Networks*, vol. 4, no. 6, pp. 669–686, 2006.

[10] I. F. Akyildiz, Z. Sun, and M. C. Vuran, “Signal propagation techniques for wireless underground communication networks,” *Physical Communication Journal (Elsevier)*, vol. 2, no. 3, pp. 167–183, Sept. 2009.

[11] A. Silva and M. Vuran, “Empirical evaluation of wireless underground-to-underground communication in wireless underground sensor networks,” *Distributed Computing in Sensor Systems*, pp. 231–244, 2009.

[12] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, “Codeblue: An ad hoc sensor network infrastructure for emergency medical care,” *Mobisys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004)*, June 2004.

[13] M. Maroti, G. Simon, A. Ledeczi, and J. Sztipanovits, “Shooter localization in urban terrain,” *Computer*, vol. 37, no. 8, pp. 60–61, Aug. 2004.

[14] A. Chandrakasan, R. Min, M. Bhardwaj, S. Cho, and A. Wang, “Power aware wireless microsensor systems,” in *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European*, Sept. 2002, pp. 47–54.

[15] A. Goldsmith and S. Wicker, “Design challenges for energy-constrained ad hoc wireless networks,” *Wireless Communications, IEEE*, vol. 9, no. 4, pp. 8–27, Aug. 2002.

[16] C. Schurgers, O. Aberthorne, and M. Srivastava, “Modulation scaling for energy aware communication systems,” in *Low Power Electronics and Design, International Symposium on, 2001.*, 2001, pp. 96–99.

[17] L. Zhaohua and G. Mingjun, “Survey on network lifetime research for wireless sensor networks,” in *Broadband Network Multimedia Technology, 2009. IC-BNMT ’09. 2nd IEEE International Conference on*, Oct. 2009, pp. 899–902.

[18] D. Tian and N. Georganas, “A node scheduling scheme for energy conservation in large wireless sensor networks,” *Wireless Communications and Mobile Computing*, vol. 3, no. 2, pp. 271–290, 2003.

[19] X. Tang and J. Xu, “Extending network lifetime for precision-constrained data aggregation in wireless sensor networks,” in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, April 2006, pp. 1–12.

[20] M. Haenggi, “Energy-balancing strategies for wireless sensor networks,” in *Circuits and Systems, 2003. ISCAS ’03. Proceedings of the 2003 International Symposium on*, vol. 4, 25-28 2003, pp. IV–828–IV–831.

[21] W. Seah, Z. A. Eu, and H.-P. Tan, "Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP) - survey and challenges," in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, May 2009, pp. 1–5.

[22] A. Hande, T. Polk, W. Walker, and D. Bhatia, "Indoor solar energy harvesting for sensor network router nodes," *Microprocessors and Microsystems*, vol. 31, no. 6, pp. 420–432, 2007, special Issue on Sensor Systems.

[23] C. Alippi and C. Galperti, "An adaptive system for optimal solar energy harvesting in wireless sensor network nodes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 55, no. 6, pp. 1742–1750, July 2008.

[24] Y. Ammar, A. Buhrig, M. Marzencki, B. Charlot, S. Basrour, K. Matou, and M. Renaudin, "Wireless sensor network node with asynchronous architecture and vibration harvesting micro power generator," in *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*. New York, NY, USA: ACM, 2005, pp. 287–292.

[25] J. A. Paradiso, "Systems for human-powered mobile computing," in *DAC '06: Proceedings of the 43rd annual Design Automation Conference*. New York, NY, USA: ACM, 2006, pp. 645–650.

[26] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *1994 IEEE Symposium on Low Power Electronics*, Oct. 1994, pp. 8–11.

[27] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[28] K. Roy, R. Roy, and T. Chou, "Design of low power digital systems," in *Designing Low Power Digital Systems, Emerging Technologies*, 1996, pp. 137–204.

[29] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[30] D. Reilly, N. Ensslin, H. S. Jr., and S. Kreiner, Eds., *Passive Nondestructive Assay of Nuclear Materials*. National Technical Information Service, March 1991.

[31] Y. Hu, J. Solere, D. Lachartre, and R. Tutchetta, "Design and performance of a low-noise, low-power consumption CMOS charge amplifier for capacitive detectors," *Nuc. Sci., IEEE Tran. on*, vol. 45, no. 1, pp. 119–123, Feb. 1998.

[32] S. Tedja, J. Van der Speigel, and H. Williams, "A CMOS low-noise and low-power charge sampling integrated circuit for capacitive detector/sensor interfaces," *Solid-State Circuits, IEEE J. of*, vol. 30, no. 2, pp. 110–119, Feb. 1995.

[33] T. Noulis, S. Siskos, and G. Sarrabayrouse, “Noise optimised charge-sensitive cmos amplifier for capacitive radiation detectors,” *Circuits, Devices & Systems, IET*, vol. 2, no. 3, pp. 324–334, June 2008.

[34] P. Grybos, A. Rodriguez, W. Dabrowski, M. Idzik, J. Gaitan, F. Prino, L. Ramello, K. Swientek, and P. Wiacek, “RX64DTH - a fully integrated 64-channel ASIC for digital x-ray imaging system with energy window selection,” in *Nuc. Sci. Symp. Conf. Rec., 2004 IEEE*, vol. 1, Oct. 2004, pp. 147–151.

[35] M. Pedrali-Noy, G. Gruber, B. Krieger, E. Mandelli, G. Meddeler, W. Moses, and V. Rosso, “PETRIC-a positron emission tomography readout integrated circuit,” *Nuc. Sci., IEEE Trans. on*, vol. 48, no. 3, pp. 479–484, Jun 2001.

[36] J. Kaplon and W. Dabrowski, “Fast CMOS binary front end for silicon strip detectors at LHC experiments,” *Nuc. Sci., IEEE Trans. on*, vol. 52, no. 6, pp. 2713–2720, Dec. 2005.

[37] M. Weng, E. Mandelli, W. Moses, and S. Derenzo, “A high-speed low-noise CMOS 16-channel charge-sensitive preamplifier ASIC for APD-based PET detectors,” *Nuc. Sci., IEEE Trans. on*, vol. 50, no. 4, pp. 898–902, Aug. 2003.

[38] J. Yeom, I. Defendi, H. Takahashi, K. Zeitelhack, M. Nakazawa, and H. Murayama, “A 12-channel CMOS preamplifier-shaper-discriminator ASIC for APD and Gas counters,” *Nuc. Sci., IEEE Trans. on*, vol. 53, no. 4, pp. 2204–2208, Aug. 2006.

[39] Z. Yacong, C. Zhongjian, L. Wengao, J. Lijiu, and Z. Baoying, “Design and test results of a front-end ASIC for radiation detectors,” in *Solid-State and Integrated-Circuit Technology, 2008. ICSICT 2008. 9th International Conference on*, Oct. 2008, pp. 1717–1720.

[40] Y. Hu, “High performance low noise charge preamplifier with DC coupling to particle silicon detectors in CMOS technology,” *Electronics Letters*, vol. 34, no. 13, pp. 1274–1275, 1998.

[41] K. Osberg, N. Schemm, S. Balkır, J. Brand, M. Hallbeck, P. Dowben, and M. Hoffman, “A handheld neutron-detection sensor system utilizing a new class of boron carbide diode,” *Sensors Journal, IEEE*, vol. 6, no. 6, pp. 1531–1538, Dec. 2006.

[42] Y. Zhao, Y. Dong, J. Gerrits, G. van Veenendaal, J. Long, and J. Farserotu, “A short range, low data rate, 7.2 GHz-7.7 GHz FM-UWB receiver front-end,” *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 7, pp. 1872–1882, July 2009.

[43] H. Sheng, P. Orlik, A. Haimovich, J. Cimini, L.J., and J. Zhang, “On the spectral and power requirements for ultra-wideband transmission,” in *Communications, 2003. ICC ’03. IEEE International Conference on*, vol. 1, May 2003, pp. 738–742.

[44] H. Kim, Y. Joo, and S. Jung, "A tunable CMOS UWB pulse generator," in *Ultra-Wideband, The 2006 IEEE 2006 International Conference on*, Sept. 2006, pp. 109–112.

[45] S. Bourdel, Y. Bachelet, J. Gaubert, R. Vauché, O. Fourquin, N. Dehaese, and H. Barthelemy, "A 9-pj/pulse 1.42-Vpp OOK CMOS UWB pulse generator for the 3.1–10.6-GHz FCC band," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 58, no. 1, pp. 65–73, Jan. 2010.

[46] B. Jung, Y.-H. Tseng, J. Harvey, and R. Harjani, "Pulse generator design for UWB IR communication systems," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, vol. 5, May 2005, pp. 4381–4384.

[47] Z. Bing, L. Baoxue, and F. Yuanchun, "An uwb receiver structure with improved performance in timing jitter sensitivity," in *Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, 2009 3rd IEEE International Symposium on*, Oct. 2009, pp. 1177–1180.

[48] Q. Li and W. S. Wong, "Performance analysis of the sample and compare receiver schemes for indoor high speed UWB system," in *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 5, Sept. 2004, pp. 3060–3064.

[49] M. Shen, T. Koivisto, T. Peltonen, L.-R. Zheng, E. Tjukanoff, and H. Tenhunen, "UWB transceiver circuits design for WPANs applications," in *Signals, Circuits and Systems, 2005. ISSCS 2005. International Symposium on*, vol. 1, July 2005, pp. 255–258.

[50] M. Oh, B. Jung, R. Harjani, and D.-J. Park, "A new noncoherent UWB impulse radio receiver," *Communications Letters, IEEE*, vol. 9, no. 2, pp. 151–153, Feb. 2005.

[51] C. Yang, Y. Lin, S. Lin, and T. Chiueh, "Design of a low-complexity receiver for impulse-radio ultra-wideband communication systems," in *Cir. and Sys., 2004. ISCAS '04. Proc. of the 2004 Int'l Symp. on*, vol. 4, May 2004, pp. IV–125–8.

[52] J. Ryckaert, M. Verhelst, M. Badaroglu, S. D'Amico, V. De Heyn, C. Desset, P. Nuzzo, B. Van Poucke, P. Wambacq, A. Baschirotto, W. Dehaene, and G. Van der Plas, "A CMOS ultra-wideband receiver for low data-rate communication," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 11, pp. 2515–2527, 2007.

[53] L. Stoica, S. Tiuraniemi, I. Oppermann, and H. Repo, "An ultra wideband low complexity circuit transceiver architecture for sensor networks," in *Cir. and Sys., 2004. ISCAS '04. Proc. of the 2004 Int'l Symp. on*, vol. 1, May 2005, pp. 364–367.

[54] R. Blazquez, P. Newaskar, F. Lee, and A. Chandrakasan, "A baseband processor for impulse ultra-wideband communications," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 9, pp. 1821–1828, Sept. 2005.

[55] S. Iida, K. Tanaka, H. Suzuki, N. Yoshikawa, N. Shoji, B. Griffiths, D. Mellor, F. Hayden, I. Butler, and J. Chatwin, "A 3.1 to 5 GHz CMOS DSSS UWB transceiver for WPANs," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, vol. 1, Feb. 2005, pp. 214–594.

[56] C. Chen, M. A. Do, K. S. Yeo, and C. C. Boon, "A low power UWB direct conversion receiver with pulse detectors," in *SoC Design Conference (ISOCC), 2009 International*, Nov. 2009, pp. 17–20.

[57] M. Crepaldi, M. Casu, M. Graziano, and M. Zamboni, "A low-power CMOS 2-PPM demodulator for energy detection IR-UWB receivers," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on*, Sept. 2007, pp. 461–466.

[58] F. Lee and A. Chandrakasan, "A 2.5 nJ/bit 0.65 v pulsed UWB receiver in 90 nm CMOS," *Solid-State Circuits, IEEE J. of*, vol. 42, no. 12, pp. 2851–2859, Dec. 2007.

[59] H. Shao and N. Beaulieu, "A novel zonal UWB receiver with superior performance," *Communications, IEEE Transactions on*, vol. 57, no. 4, pp. 1197–1206, April 2009.

[60] C. Ghosh and D. Agrawal, "Cross layer performance evaluation of a software defined radio UWB receiver using turbo decoding," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, June 2007, pp. 639–646.

[61] N. Van Helleputte, M. Verhelst, W. Dehaene, and G. Gielen, "A reconfigurable, 130 nm CMOS 108 pJ/pulse, fully integrated IR-UWB receiver for communication and precise ranging," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 1, pp. 69–83, Jan. 2010.

[62] A. Caruso, P. Dowben, S. Balkır, N. Schemm, K. Osberg, R. Fairchild, O. B. Flores, S. Balaz, A. Harken, B. Robertson, and J. Brand, "The all boron carbide diode neutron detector: Comparison with theory," *Materials Science and Engineering: B*, vol. 135, no. 2, pp. 129–133, 2006.

[63] McCreary, J.L., and P. Gray, "All-MOS charge redistribution analog-to-digital conversion techniques. I," *Solid-State Circuits, IEEE Journal of*, vol. 10, no. 6, pp. 371–379, Dec 1975.

[64] D. A. Johns and K. Martin, *Analog Integrated Circuit Design*. John Wiley & Sons, Inc., 1997.

[65] *MSP430x4xx Family User's Guide*, Texas Instruments, Dallas, TX, Lit. Num. SLAU056J.

[66] L. Lee, "New rate-compatible punctured convolutional codes for Viterbi decoding," *Comm., IEEE Trans. on*, vol. 42, no. 12, pp. 3073–3079, Dec 1994.

[67] F. Daneshgaran, M. Laddomada, and M. Mondin, "An extensive search for good punctured rate-k/(k+1) recursive convolutional codes for serially concatenated convolutional codes," *Information Theory, IEEE Trans. on*, vol. 50, no. 1, pp. 208–217, Jan. 2004.

[68] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *Communications, IEEE Trans. on*, vol. 36, no. 4, pp. 389–400, Apr 1988.

[69] J. Cain, G. Clark, and J. Geist, "Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding," *Information Theory, IEEE Trans. on*, vol. 25, no. 1, pp. 97–100, Jan 1979.

[70] P. Frenger, P. Orten, and T. Ottosson, "Code-spread CDMA using maximum free distance low-rate convolutional codes," *Communications, IEEE Trans. on*, vol. 48, no. 1, pp. 135–144, Jan 2000.

[71] K. Larsen, "Short convolutional codes with maximal free distance for rates 1/2, 1/3, and 1/4," *Information Theory, IEEE Trans. on*, vol. 19, no. 3, pp. 371–372, May 1973.

[72] T. K. Moon, *Error Correction Coding, Mathematical Methods and Algorithms*. Hoboken, New Jersey: John Wiley & Sons Inc., 2005, ch. 12, pp. 452–534.

[73] A. Bevilacqua and A. Niknejad, "An ultra-wideband CMOS LNA for 3.1 to 10.6 GHz wireless receivers," in *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, vol. 1, Feb. 2004, pp. 382–533.

[74] Z.-Y. Huang, C.-C. Huang, Y.-T. Hung, and M.-P. Chen, "A CMOS current reused low-noise amplifier for ultra-wideband wireless receiver," in *Microwave and Millimeter Wave Technology, 2008. ICMMT 2008. International Conference on*, vol. 3, Apr. 2008, pp. 1499–1502.

[75] H. Kao, A. Chin, K. Chang, and S. McAlister, "A low-power current-reuse lna for ultra-wideband wireless receivers from 3.1 to 10.6 ghz," in *Silicon Monolithic Integrated Circuits in RF Systems, 2007 Topical Meeting on*, jan. 2007, pp. 257 –260.

[76] M.-T. Hsu and K.-J. Li, "An ultrawideband CMOS low noise amplifier for 3.1-10.6 GHz wireless communication," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on*, Sept. 2007, pp. 457–460.

[77] B. Shi and M. Chia, "A CMOS ESD-protected RF front-end for UWB receiver," in *ESSCIRC, 2009. ESSCIRC '09. Proceedings of*, Sept. 2009, pp. 252–255.

[78] Y. Lu, K. S. Yeo, A. Cabuk, J. Ma, M. A. Do, and Z. Lu, "A novel CMOS low-noise amplifier design for 3.1- to 10.6-GHz ultra-wide-band wireless receivers," *Circuits*

and Systems I: Regular Papers, IEEE Transactions on, vol. 53, no. 8, pp. 1683–1692, Aug. 2006.

[79] W. Chunhua and D. Chao, “A CMOS 3.1-10.6 GHz merged UWB LNA and mixer,” in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM ’08. 4th International Conference on*, Oct. 2008, pp. 1–4.

[80] B. Park, S. Choi, and S. Hong, “A low-noise amplifier with tunable interference rejection for 3.1- to 10.6-GHz UWB systems,” *Microwave and Wireless Components Letters, IEEE*, vol. 20, no. 1, pp. 40–42, Jan. 2010.

[81] S. Limei, “A 1.2 V 3.1-10.6 GHz CMOS low noise amplifier with 24 dB gain and 2.4 dB noise figure,” in *Wireless Communications Signal Processing, 2009. WCSP 2009. International Conference on*, Nov. 2009, pp. 1–4.

[82] H. Xie, X. Wang, A. Wang, Z. Wang, C. Zhang, and B. Zhao, “A fully-integrated low-power 3.1-10.6 GHz UWB LNA in 0.18 μm CMOS,” in *Radio and Wireless Symposium, 2007 IEEE*, Jan. 2007, pp. 197–200.

[83] T. H. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits*, 2nd ed. New York, NY: Cambridge University Press, 2004.

[84] B. Hu and N. Beaulieu, “Accurate performance evaluation of time-hopping and direct-sequence UWB systems in multi-user interference,” *Communications, IEEE Transactions on*, vol. 53, no. 6, pp. 1053–1062, June 2005.

[85] M. Win and R. Scholtz, “Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications,” *Communications, IEEE Transactions on*, vol. 48, no. 4, pp. 679–689, Apr. 2000.

[86] M. Srifi, S. Podilchak, M. Essaaidi, and Y. Antar, “Planar circular disc monopole antennas using compact impedance matching networks for ultra-wideband (UWB) applications,” in *Microwave Conference, 2009. APMC 2009. Asia Pacific*, Dec. 2009, pp. 782–785.

[87] E. Vittoz, M. Degrauwe, and S. Bitz, “High-performance crystal oscillator circuits: theory and application,” *Solid-State Circuits, IEEE J. of*, vol. 23, no. 3, pp. 774–783, Jun 1988.

[88] R. Schreier and G. C. Temes, *Understanding Delta-Sigma Data Converters*. Piscataway, NJ, Hoboken, NJ: IEEE Press, Wiley, 2005.

[89] S. R. Norsworthy, R. Schreier, and G. C. Temes, Eds., *Delta-sigma data converters : theory, design, and simulation*. New York: IEEE Press, 1997.

[90] I. Ketsman, Y. Losovyj, A. Sokolov, J. Tang, Z. Wang, M. Natta, J. Brand, and P. Dowben, "Gd-doping of HfO_2," *Applied Surface Science*, vol. 254, no. 14, pp. 4308–4312, 2008, proc. 3rd Intn'l Workshop on Surf. Phy. "Nanostructures on Surface" - IWSP-2007.

[91] Y. Losovyj, I. Ketsman, A. Sokolov, K. Belashchenko, P. Dowben, J. Tang, and Z. Wang, "The electronic structure change with gd doping of HfO_2 on silicon," *Applied Physics Letters*, vol. 91, p. 132908, 2007.

[92] Y. Sakurai and T. Kobayashi, "Experimental verification of the nuclear data of gadolinium for neutron capture theory," *J. of Nuc. Sci. Tech. Sup. 2*, pp. 1294–1297, Aug. 2002.

[93] R. C. Greenwood, C. W. Reich, H. A. Baader, H. R. Koch, D. Breitig, O. W. B. Schult, B. Fogelberg, A. Bäcklin, W. Mampe, T. V. Eg1dy, and K. Schreckenbach, "Collective and two-quasiparticle states in ^{158}Gd observed through study of radiative neutron capture in ^{157}Gd," *Nuclear Physics A*, vol. 304, no. 2, pp. 327–428, 1978.

[94] P. Reeder, "Thin GSO scintillator for neutron detection," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 353, no. 1-3, pp. 134–136, 1994.

[95] X-5 Monte Carlo Team, "MCNP-A general monte carlo N-particle transport code, version 5 - volume I: Overview and theory," LA-UR-03-1987, Tech. Rep., Apr. 2003, revised Sept. 2003.

[96] H. Photonics, *Photomultiplier Tubes: Basics and Applications*, 3rd ed., Feb. 2006.

[97] M. Swoboda, R. Arlt, V. Gostilo, A. Lupilov, M. Majorov, M. Moszynski, and A. Syntfeld, "Spectral gamma detectors for hand-held radioisotope identification devices (RIDs) for nuclear security applications," *Nuclear Science, IEEE Transactions on*, vol. 52, no. 6, pp. 3111–3118, Dec. 2005.

[98] K. Debertin and R. G. Helmer, *Gamma- and X-Ray Spectrometry with Semiconductor Detectors*. Amsterdam, North-Holland: Elsevier Science Publishers B.V., 1988.

[99] T. Terada, S. Yoshizumi, M. Muqsith, Y. Sanada, and T. Kuroda, "A CMOS ultra-wideband impulse radio transceiver for 1-Mb/s data communications and ± 2.5-cm range finding," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 4, pp. 891–898, Apr. 2006.

[100] D. Lachartre, B. Denis, D. Morche, L. Ouvry, M. Pezzin, B. Piaget, J. Prouvee, and P. Vincent, "A 1.1nJ/b 802.15.4a-compliant fully integrated UWB transceiver in 0.13 μm CMOS," in *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, Feb. 2009, pp. 312–313.

Appendix A

Fourth Chip Pin-out

The chip is packaged in a 12mm x 12mm 100 pin quad flat no-lead (QFN) package with a large solder pad on the back. The large solder pad on the back is connected to the chip's high impedance substrate. This is common to all circuits on the chip. It is recommended to use this single pad as the star point for the common connection of all grounds

The chip has mostly untested ESD structures on most pins. The digital and analog pads were manually designed, and were not from a standard library. As such, the ESD structures may not be robust. The ESD structures were designed to withstand 4 kV using the human-body model. The RF pads have no dedicated ESD structures on them. The UWB receiver input has some built-in ESD protection due to the input circuit configuration, however the narrowband receiver input has no protection, parasitic or otherwise.

Table A.1 lists all the pins.

Table A.1 – Pinout

#	Name	Function	Comments
1	DDO(1)[1]	Diode Dac 1 Out	
2	DDC(4)[1]	Diode Dac 4 Compensation Point	
3	DDC(3)[1]	Diode Dac 3 Compensation Point	
4	DDC(2)[1]	Diode Dac 2 Compensation Point	
5	DDC(1)[1]	Diode Dac 1 Compensation Point	
6	NQR In	NQR matched input	50 Ω input
7	RFI In	NQR RFI matched input	50 Ω input
8	NQR-	High-Z NQR differential input	
9	NQR+	High-Z NQR differential input	
10	GND_{NQR}	Ground for NQR circuits	
11	VDD_{NQR}	V_{DD} for NQR circuits	
12	PA(0)	Serial Flash $\overline{\text{CS}}$	Implemented in boot code

Continued on next page

Continued from previous page

#	Name	Function	Comments
13	PA(1)	MISO0	
14	PA(2)	MOSI0	
15	PA(3)	SPI0 Clock	
16	PA(4)	UART0 TXD	
17	PA(5)	UART0 RXD	
18	PA(6)	IRQ In	
19	PA(7)	Prog. Mode	Pull low at startup to enter flash programming mode.
20	PA(8)	Narrowband TX	Pull low at startup to use the variable frequency oscillator.
21	GND_{IO}	I/O Ground	
22	VDD_{IO}	3.3V I/O Supply	
23	PA(9)	Flip Flash Bytes	Pull low at startup to flip the endianness of the bytes in the serial flash.
24	PA(10)	Clock Out	Selectable to be one of 8 internal clocks.
25	PA(11)	Unused	
26	PA(12)	$\overline{\text{Ext. Bus Ctrl}}$	Reset with peripheral function active
27	PA(13)	$\overline{\text{NMI}}$	Reset with peripheral function active
28	NBTxOut	Narrow-band transmitter power amplifier out	
29	GND_{core}	Core Digital Ground	
30	VDD_{core}	1.8V Core Digital V_{DD}	
31	PA(14)	$\overline{\text{Main Clock Override}}$	Reset with peripheral function active
32	PA(15)	Main Clock I/O	Can drive only MCLK
33	PB(0)	Per. 0: I^2C SDA	Per. 1: PLL Up[2]
34	PB(1)	Per. 0: I^2C SCL	Per. 1: PLL Down[2]
35	PB(2)	Per. 0: CSA1 ADC Trigger[2]	Per. 1: MISO1
36	PB(3)	Per. 0: NB Bit Det. In[2]	Per. 1: MOSI1
37	PB(4)	Per. 0: NB Bit Det. Clk[2]	Per. 1: SPI1 Clk
38	PB(5)	Per. 0: NB Seek In[2]	Per. 1: UART1 TXD
39	PB(6)	Per. 0: NB Seek Clk[2]	Per. 1: UART1 RXD
40	PB(7)	Per. 0: NB Bit Template[2]	Per. 1: Unused
41	PB(8)[3]	Per. 0: NB Integrator Rst.[2]	Per. 1: TIO0A

Continued on next page

Continued from previous page

#	Name	Function	Comments
42	PB(9)[3]	Per. 0: UWB Tx Clk[2]	Per. 1: TIO0B
43	PB(10)[3]	Per. 0: UWB Tx Data[2]	Per. 1: TIO1A
44	PB(11)[3]	Per. 0: UWB Rx Bit Det. In[2]	Per. 1: TIO1B
45	PB(12)[3]	Per. 0: UWB Rx Bit Det Clk[2]	Per. 1: TIO2A
46	PB(13)[3]	Per. 0: UWB Rx Adj Clk[2]	Per. 1: TIO2B
47	PB(14)[3]	Per. 0: UWB Rx Adj Clk Dir[2]	Per. 1: TIO3A
48	PB(15)[3]	Per. 0: Unused	Per. 1: TIO3B
49	VDD_{core}	1.8V Core Digital V_{DD}	
50	GND_{core}	Core Digital Ground	
51	$\overline{RST}$	Processor Reset	Input/Output, Open-drain
52	GND_{IO}	I/O Ground	
53	VDD_{IO}	3.3V I/O Supply	
54	UWB+	UWB+ Transmitter Output	
55	UWB-	UWB- Transmitter Output	
56	SW_{DVDD}	1.8V Digital Buck Regulator SW	
57	VDD_{pwr}	1.8V Digital Power Section V_{in}	
58	GND_{pwr}	1.8V Digital Power Section Ground	
59	SW_{DVDD}	1.8V Digital Buck Regulator SW	
60	FB_{DVDD}	1.8V Digital Buck Feedback Voltage	
61	$\overline{EN}$	Enable the 1.8V Digital Buck	
62	VDD_{Buck}	Buck Regulator Control Circuitry V_{DD}	
63	GND_{Buck}	Buck Regulator Control Circuitry Ground	
64	FB_{AVDD}	1.8V Analog Buck Feedback Voltage	
65	SW_{AVDD}	1.8V Analog Buck Regulator SW	
66	GND_{pwr}	1.8V Analog Buck Power Section Ground	
67	VDD_{pwr}	1.8V Analog Buck Power Section V_{in}	
68	SW_{AVDD}	1.8V Analog Buck Regulator SW	
69		External PMOS Power Switch	
70	GND_{BB}	RF Baseband Ground	
71	VDD_{BB}	1.8V RF Baseband V_{DD}	
72	VDD_{RF}	1.8V RF Front End V_{DD}	
73	GND_{RF}	RF Front End Ground	
74	NB In	NB Receiver RF In	No ESD Protection

Continued on next page

Continued from previous page

#	Name	Function	Comments
75	UWB In	UWB Receiver RF In	No ESD Protection
76	HF TP	High Frequency Analog Test Port	
77	V_{fix}[4]	CSA V_{fix}	Delta Sigma ADC Reference
78	V_{fb}[4]	CSA V_{fb} or Analog Override In	Delta Sigma ADC Input 1
79	V_{thres}[1,4]	CSA V_{thres}	Delta Sigma ADC Input 0
80	$\text{REF}_{900\text{mV}}$	900 mV Reference	
81	$\text{REF}_{1\text{V}}$	1 V Reference	
82	LF TP[4]	Low Frequency Analog Test Port	Delta Sigma ADC Input 3
83	BiasN	N-Channel MOSFET Bias	
84	BiasNCasc	N-Channel MOSFET Cascode Bias	
85	BiasPCasc	P-Channel MOSFET Cascode Bias	
86	BiasP	P-Channel MOSFET Bias	
87	HF Xin	High Frequency Crystal Oscillator In	
88	HF Xout	High Frequency Crystal Oscillator Out	
89	LF Xout[1]	Low Frequency Crystal Oscillator Out	
90	LF Xin[1]	Low Frequency Crystal Oscillator In	
91	CSA4 In[1]	CSA 4 In	ADC 4 In
92	CSA3 In[1]	CSA 3 In	ADC 3 In
93	CSA2 In[1]	CSA 2 In	ADC 2 In
94	CSA1 In[1]	CSA 1 In	ADC 1 In or Delta Sigma ADC Input 2
95	$\text{VDD}_{\text{Analog}}$	1.8V Analog Supply	
96	$\text{GND}_{\text{Analog}}$	Analog Ground	
97	$\text{VDD}_{\text{A3.3}}$	3.3 V Analog V_{DD}	
98	DDO(4)[1]	Diode Dac 4 Out	
99	DDO(3)[1]	Diode Dac 3 Out	
100	DDO(2)[1]	Diode Dac 2 Out	

Note [1]: These pins are multiplexed with NQR functionality.
Note [2]: Function used for debugging purposes.
Note [3]: Pin also used for external bus control.
Note [4]: These pins are heavily multiplexed with more functions than listed in the table. See detailed description of each.

Pin Function	Description
Diode Dac Compensation Point	This is an internal compensation point for the constant current diode DAC. It requires a 10 nF - 100 nF capacitor be connected between it and 3.3V AV_{DD}. Use a larger capacitor to improve stability of the diode driver with a large diode capacitance and/or high impedance diode.
Digital Core Power and Ground	These supply power and ground to the digital core circuitry. The two ground pins are internally connected to each other, but not to other chip grounds. They should be connected together externally, and connected to other analog grounds at a star point. These should be connected to the output of the digital buck regulator. Supply bypass capacitors should be placed close to the external pins. The supply voltage is adjustable via DVS. These pins are also the power supply to the narrowband power amplifier.
PMOS Power Switch	This is an output of the digital buck regulator. It goes low when the regulator is on, and is pulled to the input voltage of the regulator otherwise. It can be connected to the gate of an external PMOS load switch to turn on the 3.3V supplies to the rest of the chip when the digital 1.8 V buck regulator is enabled. The pad driver operates on the VDD_{buck} and GND_{buck} supply.
SW	These are the switching outputs of the buck regulators
VDD_{pwr} and GND_{pwr}	These are the supply lines for the power section of the buck regulators. A bypass capacitor between these lines should be placed as close as possible to the chip to contain the high di/dt values present on these lines.
FB	These are the feedback inputs to control the buck regulators. The output voltage of the buck regulator should be connected to this line. No external feedback network is required.
VDD_{buck} and GND_{buck}	This are the supply lines for the control and reference section of the buck regulators. A bypass capacitor between these lines should be placed as close as possible to the chip. The system bandgap reference operates on this supply, so this supply should be as clean as possible.
$\overline{EN}$	Pulling this line low enables the digital buck regulator. This line is latched internally, so once when pulled low, the regulator will stay on even if this line goes back high again. The regulator can only be shut off by the processor setting the TurnOff bit in the system control register. A switch can be connected between this terminal and ground to function as an on switch. The current status of this line can be read by the microcontroller, thereby reading the status of the switch. There is no integrated pullup on this line. The analog buck regulator is controlled exclusively by the microcontroller.

Pin Function	Description
UWB+ and UWB-	This is the differential output for the UWB transmitter. If a single-ended output is desired, one may be left disconnected. These may be independently disabled in software. An optional external filter may be connected between this and the antenna for pulse shaping.
VDD_{IO} and GND_{IO}	These are the power pins which supply power to the pad ring and the associated pad drivers. These also supply power for the UWB transmitter output buffers. They will operate from 2V to 3.6V. Bypass capacitors should be placed as close as possible to the pins to minimize ground bounce and related issues.
$\overline{RST}$	This is the main system reset input. When low, the microcontroller and related circuits are reset. This is a bidirectional pad, with an open collector output with an integrated 15 kΩ pullup resistor. During power-up, an internal power-on reset circuit pulls this line low until 500 μs after the digital buck regulator has reached regulation. If the microcontroller requests a manual hardware reset, this line will pulse low for 500 μs.
PAx and PBx	These pins can be either general purpose I/O or have dedicated functionality depending on the configuration. They can also have a 15 kΩ pullup resistor present under software control. During reset, the pins are all configured as inputs with pullups enabled. If the core supply voltage is less than 300 mV, they will be forced to high impedance with the pullups enabled.
VDD_{RF} and GND_{RF}	These are the power pins for the RF part of the internal RF transceivers. This should be nominally 1.8 V. Supply bypass capacitors should be placed as close as possible to the chip. These should be connected to the RF baseband power pins. They supply the RF amplifiers and mixers in the transceivers, but do not supply the baseband processing circuitry.
VDD_{BB} and GND_{BB}	These are the power pins for the analog baseband processing circuits for the RF transceivers. These are more noisy than the RF supplies, but must be at the same voltage as the RF supplies. The digital part of the baseband circuits operate on the core digital supply.
High Frequency Analog Test Port	This pin is an analog testing port designed to provide visibility to some high frequency test signals present inside the RF transceivers. The bandwidth of the signals is up to 500 MHz. All the signals are internally differential signals. The circuit for driving this pin is a differential to single-ended conversion circuit followed by a high frequency source follower. The output can drive a 10 pF load at 500 MHz. The actual signals driven out in this port is selected in the analog test control register. If the test port is disabled, this pin is high impedance

Pin Function	**Description**
UWB Receiver RF Input	This is a 50 Ω RF input for the UWB receiver. A DC blocking capacitor may be required to be in series with this input. No other matching components are necessary. The DC level of this pin is ground. This input as no dedicated ESD protection.
Narrowband Receiver RF Input	This is the 50 Ω RF input for the narrowband receiver. An external 47 nH inductor must be placed in series with this pin for matching purposes. An internal 10 pF DC blocking capacitor is integrated on this pin. This input as no dedicated ESD protection.
Narrowband Transmitter RF Output	This is the output of the RF power amplifier. The power amplifier is basically a large digital inverter, so this signal will be a square wave. It must be filtered by a series high-Q LC tank circuit to suppress the harmonics, and an optional parallel LC tank. The parallel tank should come after the series LC tank. A matching network can be used to increase the output power. The output is basically a voltage output, so varying load resistors produce varying output power. The output voltage is $\mathrm{VDD}_{\mathrm{core}} \times 4/\pi$, however, this is not taking into account the non-idealities of the output drivers. The power amplifier operates on the digital core supply.
NQR Matched Input	This is the input of the single-ended NQR signal. The NQR signal can be applied either to this input or to the differential input. This input is matched to 50 Ω. An external DC blocking capacitor may be required.
NQR Differential Inputs	These inputs are the other possible NQR input signal path. These inputs are high impedance (1 MΩ). They may need an external DC blocking capacitor.
NQR RFI Input	This input is for the RFI signal for the NQR receiver. It is a single-ended signal, matched to a 50 Ω line.
$\mathrm{VDD}_{\mathrm{Analog}}$ and $\mathrm{GND}_{\mathrm{Analog}}$	These are the power pins for the internal analog circuitry. This should be nominally 1.8V. Supply bypass capacitors should be placed as close as possible to the chip. This should be connected to the output of the analog buck regulator.

Pin Function	Description
CSA V_{fix}	This pin has a number of functions depending on the internal control register configuration. One function is to override the internal "V_{fix}" voltage going to the CSA with an externally supplied one. Generally the internally supplied one is sufficient, so this is not necessary. If the RefSel bit in the ADC_CR register is clear, this pin is used as the reference for the delta-sigma ADC, if the RefSel bit is set, the system 1 V reference is used. A third function is enabled if the DrvRef bit in the REF_CAL register is set, which connects this pin to the internal 200 mV reference used by the DCO. This pin can then either supply that reference if the internal 200 mV reference driver is disabled (bit 200mEn in the SYS_CR register), or view the voltage of the internal reference if the internal driver is enabled.
CSA V_{fb}	This pin also has a number of functions. One function is to override the internal feedback voltage applied to the CSA. Usually, the internal DAC will supply this voltage to the CSA, so this is unnecessary. A second function is one input to the delta-sigma ADC. Finally, this pin is used to override or view various control signals throughout the design. This pin can override • The current control for the high-frequency crystal oscillator (bit HFAmpOr in OSC_CR) • The VCO control voltage of the PLL (bit CtrlOr in PLL_CR) • The AGC control voltage of the UWB receiver (bit AGCOver in UWBRX_BDCR) • The AGC control voltage of the narrowband (bit AGCOver in NBRX_CR)
CSA V_{thres}	This pin also has a number of functions. One function is to override the internal threshold voltage applied to the CSA. Usually the internal DAC integrated with each front end will supply this voltage, so supplying it externally is unnecessary. A second function is as one input to the delta-sigma ADC. A third function is the current reference input/output. The bits IRefCtrl in the SYS_CR register affect if this pin is driven with the internal current reference, or if the current reference is supplied on this pin.

Pin Function	Description
1 V and 900 mV Reference	These are the output voltages for the internal 900 mV and 1 V references. They each require at least a 10 nF capacitor to ground for stability. There is no maximum limit on the capacitance which can be added. 100 nF of capacitance is recommended for better noise performance. If the references are disabled, these lines become high impedance and an external reference can be driven on these lines instead of the internal ones.
Low Frequency Analog Test Port	This output can drive many internal analog signals from throughout the chip to an external pin for debugging purposes. The output driver consists of a rail-to-rail op-amp in unity gain configuration. The op-amp is most stable with 6pF of external load, with a maximum external capacitance of 18pF. The bandwidth of signals is up to 20 MHz. The actual signal driven out on this line is controlled by the analog test control register. If the analog test port is disabled, this line is high impedance. This pin is also the input to the delta-sigma ADC as well as an auxiliary input for CSA ADC 3.
BiasN and BiasNCasc	These are the bias voltages required for the operation most analog circuits on the chip. They each require 10 nF or larger capacitors to ground. Disabling the global bias generator places these lines in high impedance. The internal drivers can source up to 10-50 μA of current, but can only sink a few tens of nA, so when measuring the voltage on these pins, always measure from ground to these terminals so the the multimeter current is sourced by the drivers. In actual circuit operation, the circuit only requires these to source current, not sink current.
BiasP and BiasPCasc	These are more of the bias voltages required for the operation of most analog circuits on the chip. They each require 10 nF or larger capacitors to $\text{VDD}_{\text{Analog}}$. Disabling the global bias generator places these lines in high impedance. The internal drivers can sink up to 10-50 μA of current, but can only source a few tens of nA, so when measuring the voltage on these pins, always measure from analog V_{DD} to these terminals so the the multimeter current is sunk by the drivers. In actual circuit operation, the circuit only requires these to sink current, not source current.
CSA Inputs	These are the inputs to the charge sensitive amplifiers. Externally, they require a DC blocking capacitor between this pin and the diode. They can also function is direct inputs to the ADCs associated with those channels. Finally, the CSA 1 input is also an input the delta-sigma ADC.
Low Frequency Xin and Xout	These are the inputs and outputs of the internal low frequency 32 kHz crystal oscillator. They have internal digitally adjustable load capacitors up to 20 pF.

Pin Function	**Description**
High Frequency Xin and Xout	These are the inputs and outputs of the internal high frequency crystal oscillator. They have internal digitally adjustable load capacitors up to 20 pF. The high frequency oscillator is designed to be connected to crystals between 10 and 20 MHz.
3.3 V Analog VDD	This pin supplies power to the diode drive circuits. These circuits can operate from 2V to 3.6 V.
Diode Dac Outputs	These pins are the outputs of the diode drive DACs. These supply either a constant voltage to bias the diode, or supply a voltage with a high impedance at signal frequencies depending on the configuration bits.