2011

Design of an FPGA-based Array Formatter for Casa Phase-Tilt Radar System

Akilesh Krishnamurthy
University of Massachusetts Amherst

DESIGN OF AN FPGA-BASED ARRAY FORMATTER FOR CASA PHASE-TILT RADAR SYSTEM

A Thesis Presented

by

AKILESH KRISHNAMURTHY

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2011

Electrical and Computer Engineering

DESIGN OF AN FPGA-BASED ARRAY FORMATTER FOR CASA PHASE-TILT RADAR SYSTEM

A Thesis Presented

by

AKILESH KRISHNAMURTHY

Approved as to style and content by:

Russell G. Tessier, Chair

Stephen Frasier, Member

Joseph Bardin, Member

C. V. Hollot, Department Chair
Electrical and Computer Engineering

ACKNOWLEDGMENTS

I would first like to extend my deepest sense of gratitude to my parents and brother for their encouragement and support to pursue graduate studies at UMass Amherst.

I would like to thank my advisor, Professor Russell Tessier and Eric Knapp for giving me the wonderful opportunity to work in their research group. My thesis work would not have been possible without their constant motivation and guidance. I will forever be grateful for all the encouragement and continued support I received from them. I also wish to thank Professor Stephen Frasier and Professor Joseph Bardin for their helpful comments and feedback.

I would like to acknowledge the help and support of my lab mates Deepak Unnikrishnan and Vishwas Vijayendra, with whom I have shared numerous technical discussions and brainstorming sessions. Thanks to Arunachalam Annamalai for his help during the final phase of the project. Finally, I would like to thank all my friends in Amherst, the members of the Reconfigurable Computing Group, and CASA, who have made my stay here truly memorable.

ABSTRACT

DESIGN OF AN FPGA-BASED ARRAY FORMATTER FOR CASA PHASE-TILT RADAR SYSTEM

SEPTEMBER 2011

AKILESH KRISHNAMURTHY

B.E., ANNA UNIVERSITY, INDIA

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Russell G. Tessier

Weather monitoring and forecasting systems have witnessed rapid advancement in recent years. However, one of the main challenges faced by these systems is poor coverage in lower atmospheric regions due to earth's curvature. The Engineering Research Center for the Collaborative Adaptive Sensing of the Atmosphere (CASA) has developed a dense network of small low-power radars to improve the coverage of weather sensing systems. Traditional, mechanically-scanned antennas used in these radars are now being replaced with high-performance electronically-scanned phased-arrays. Phased-Array radars, however, require large number of active microwave components to scan electronically in both the azimuth and elevation planes, thus significantly increasing the cost of the entire radar system. To address this issue, CASA has designed a "Phase-Tilt" radar, that scans electronically in azimuth and mechanically in elevation. One of the core components of this system is the Phased-Array controller or the Array Formatter. The Array Formatter is a Field Programmable

Gate Array (FPGA)-based master controller that translates user commands from a computer to control and timing signals for the radar system. The objective of this thesis is to design and test an FPGA-based Array Formatter for CASA's Phase-Tilt radar system.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

State-of-the-art weather prediction systems rely on the use of advanced radar systems for efficient and accurate information. Radar systems study the quantity of interest by transmitting an electromagnetic signal in space and detecting the echoed signal received from the target [17]. The comparison of transmit and receive signals reveals crucial target information, such as location, size, etc. The process is repeated several times to increase the accuracy of the target information collected. Current radar systems are digitally controlled using modern computers and sophisticated data acquisition systems. A modern radar system typically consists of a computer, an antenna panel for transmitting and receiving signals, and a data acquisition system for processing digital and analog signals as shown in Figure 1.1. The Data Acquisition Systems consists of Analog to Digital Converters, analog and digital circuitry, and hardware support to perform complex signal processing algorithms. They also require several high-speed interfaces to communicate with different radar components.

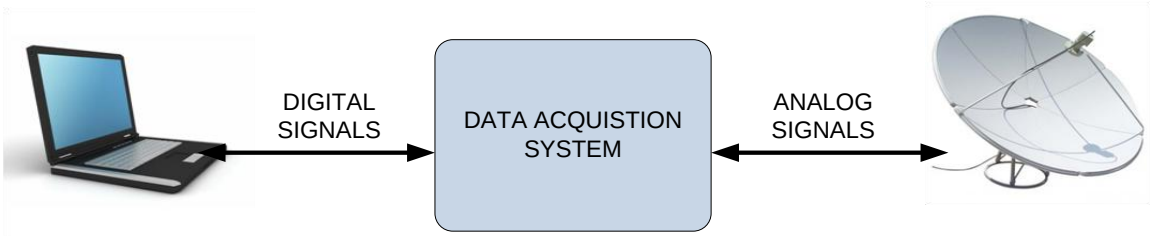

Figure 1.1. Modern radar system

Radars used for atmospheric sensing gather critical weather-related information, that can be used for weather monitoring and forecasting. The information collected by

weather radars can detect hazardous weather events such as tornadoes. Traditionally, these radars have been mechanically controlled, in which the antenna panel is tilted by a motor controller [5]. A typical scanning cycle of mechanically-tilted weather radars is as follows: The radar is first tilted in the elevation plane, and samples a small region of the atmosphere by transmitting a beam of energy, and then listening to the received energy for a certain period of time. The radar is then moved in the azimuth plane, and this process is continued across multiple azimuth beam positions, until the radar completes a 360 degree scan in the azimuth plane. The radar then changes the elevation angle to perform another azimuth scan. In this way, many azimuth scans are performed to scan the entire region of interest in the atmosphere. Such radars typically consume several minutes to perform a full cycle scan in the atmosphere [5] [7]. Alternative scanning techniques have been explored in the past to address the limitations of mechanically-scanned systems. Phased-Arrays are an emerging breed of radars, that have received significant attention from the radar community by virtue of their fast scanning abilities [8]. They have an array of antennas fed by signals, whose relative phases are varied to produce radiation patterns in desired directions and cancel patterns in other directions.

Phased-Array radars can send multiple beams at the same time without tilting or moving the radar mechanically as shown in Figure 1.2 [5]. Current state-of-the-art two-dimensional phased-array antenna systems are capable of electronic beam steering in both the azimuth and elevation planes. Radio Frequency (RF) circuitry such as phase-shifters and attenuators facilitate electronic beam steering in these radars. Typical scanning rates of phased-arrays are in the order of seconds, thus delivering better scanning performance when compared to their mechanical counterparts [5]. The ability to "dwell" or repeatedly sense the same quantity of interest increases the accuracy of the data sampled by these radars. Also, the failure rate of phased-

arrays is much lower than that of mechanically-tilted radars, because of the absence of mechanical components like motor controllers [5] [7].

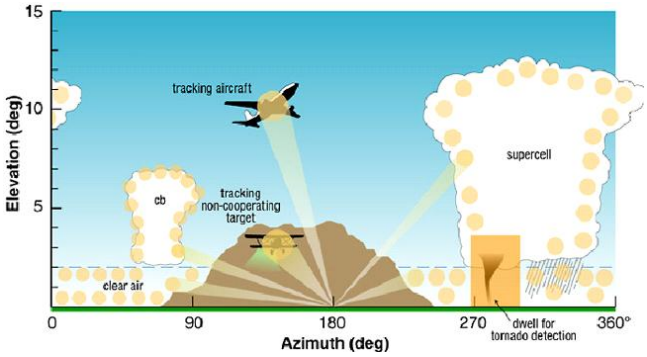

Figure 1.2. Scanning ability of electronically-scanned phased-array radars [5]

The electronically-scanned phased-array radars, however, suffer from two main limitations - (1) the high-cost of the active electronic circuitry behind the antenna array, and (2) the design of highly complex digital sub-systems to facilitate electronic beam steering. CASA has proposed to address these issues by designing a low-cost phased-array radar system called "Phase-Tilt" [8]. The Phase-Tilt radar scans electronically only in one dimension (azimuth), thereby, reducing the amount of active electronic circuitry behind the antenna array. Scanning in the elevation plane is performed mechanically using a tilt controller. The block diagram of the Phase-Tilt radar system is shown in Figure 1.3. The Phase-Tilt radar consists of the following sub-systems: Host Computer, Transceiver, Array Controller or Formatter, Tilt Controller, and the Phase-Tilt Antenna array [8]. The linear antenna array is a planar structure of 64×32 passive antenna elements [14] [15]. The array consists of 64 columns; each column has 32 antennas. The antenna columns are controlled by active electronic circuits called Transmit/Receive Modules (T/R Modules). There are 64 T/R Modules for 64 antenna columns; a T/R Module controls the 32 antenna elements in the column. The tilt controller is used for tilting the antenna array in the elevation plane. The Transceiver sub-system receives the RF signals from the

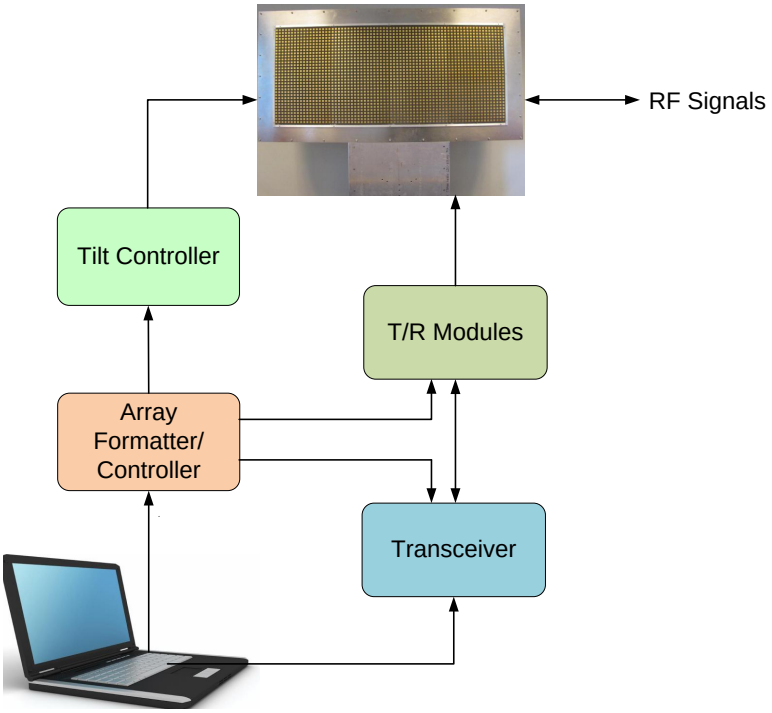

Figure 1.3. Phase-Tilt radar system block diagram

antenna array and performs all the relevant signal processing tasks needed for post processing of weather data.

The master controller in this system is the Array Controller or the Formatter which controls and synchronizes the different radar sub-systems with high-speed control, trigger, and switch signals as shown in Figure 1.4. The Formatter interfaces the computer to all the radar sub-systems. User commands from the computer are broken down, interpreted, and translated by the Formatter to data and trigger signals. The aim of this thesis is to design an Array Formatter for CASA's Phase-Tilt radar system.

The functions of the Array Formatter are: storing and transferring data files, updating beam positions, sending timing and switch signals to trigger events, and sending control signals to control the different radar sub-systems. The Formatter is the gateway for the Host Computer to communicate with all the other radar sub-systems. The details of the roles of the Formatter are described in Chapter 2 and

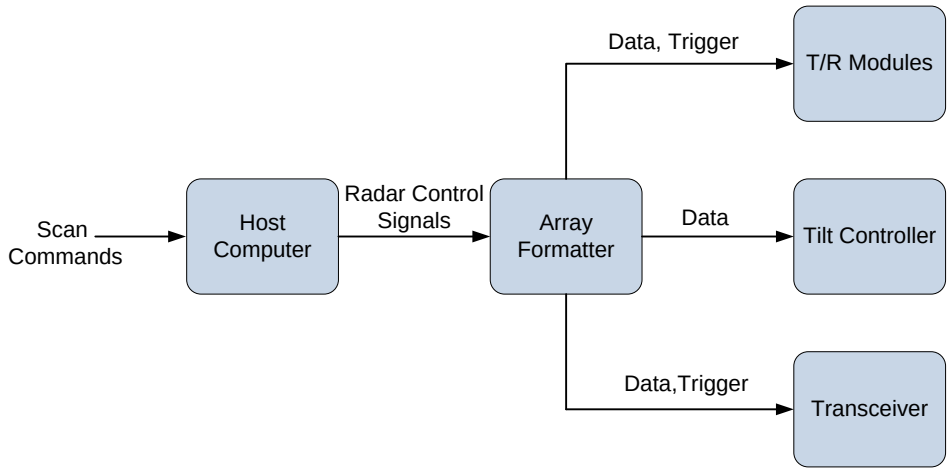

Figure 1.4. Phased-Array Controller/Formatter

Chapter 6. A typical Array Formatter requires hardware resources such as microcontrollers or microprocessors for data acquisition and control, peripheral interfaces for communicating with radar sub-systems, and sufficient memory for storing data and timing information [16].

Field Programmable Gate Arrays (FPGA)-based radar controllers are used these days by virtue of their fine grained and course grained parallelism, and support for high-speed interfaces [9]. FPGAs also offer a high degree of flexibility at an affordable cost. In this project, we use a soft microprocessor-based system on the FPGA to implement an Array Formatter for CASA's Phase-Tilt radar system. A soft microprocessor is a microprocessor that is implemented in the FPGA fabric. The soft microprocessor is the central control agent in the Formatter design. This FPGA-based system will be the first prototype used as the Array Formatter for phased-array antenna systems.

The rest of the thesis document is organized as follows: Chapter 2 describes the CASA Phase-Tilt Radar System. The Array Formatter Board specifications and features are described in Chapter 3. Chapter 4 introduces the MicroBlaze soft processor by Xilinx and also the environment used to create the embedded system on the

FPGA. Chapter 5 outlines the design methodology adopted in this thesis work, and also the FPGA system architecture of the Array Formatter. The Array Formatter software has been discussed in Chapter 6. Chapter 7 describes the ethernet interface between the Array Formatter and the Host Computer in detail. Chapter 8 concludes the thesis with an overview of the future work.

CHAPTER 2

BACKGROUND

For several decades, weather sensing systems have relied on the use of long-range, high-power radars to sense the atmosphere. Conventional radars transmit high-power radar beams that traverse several hundreds of miles in space in a straight path. While the beam continues to travel along straight lines, the earth's surface beneath the beam begins to curve away [16]. Radars are affected by the earth's curvature that limits their range of scanning, and also their ability to cover the lower regions of the atmosphere. The Center for Collaborative Sensing of the Atmosphere (CASA), an Engineering Research Center established by the National Science Foundation, is developing a dense network of small low-power radars to overcome coverage gaps due to earth's curvature and complex terrains [16]. CASA radars are separated by a few tens of kilometers apart over a large area, which enables them to get high spatial and temporal resolution views even in the lower atmospheric regions. The Center has currently deployed a network of four mechanically-scanned radars in Oklahoma. The next step in the evolution of distributed radar systems is the use of solid-state electronics and low-cost, electronically-scanned phased-array radars.

The active electronic circuits used in current phased-arrays are the solid-state Transmit/Receive Modules [11]. The T/R Modules control the antenna array by varying the antenna parameters such as phase, amplitude, and electric field polarizations. While the phase and amplitude adjust the antenna beam positions, the electric field polarization of the received signal gives structural characteristics of the scattering target. The electric field polarization of an electromagnetic (EM) wave refers to

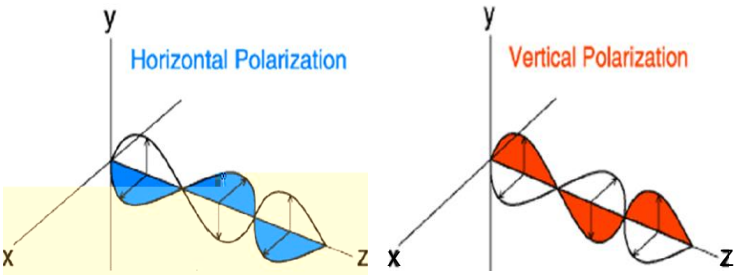

Figure 2.1. Horizontal and vertical polarizations of radar signals [10]

the locus of points of the electric field vector that lies in a plane perpendicular to the direction of propagation of the wave. Weather radars usually transmit radiations whose electric field vector lies either in the horizontal or vertical plane as shown in Figure 2.1 [10].

2.1 Dual-Polarized Radars

Dual-polarized radars have the ability to transmit or receive signals in either polarization. The variable transmit and receive electric field polarizations in the signals allows dual-polarized radars to measure hydrometeor characteristics, such as shape, size, and differentiate thermodynamic phase [6]. This helps in determining rainfall rates, and sometimes discriminate between between liquid or ice phases of the hydrometeor [6]. A weather radar usually transmits in one polarization and receives in the other or same polarization depending on the scattering characteristics of the target. The radar follows a sequence of Transmit (T) and Receive (R) modes in either Horizontal (H) or Vertical (V) polarization to determine the information about the hydrometeor. There are four possible polarimetric combinations for a dual-polarized radar which are:

1. TH - Transmit in Horizontal Polarization

2. RH - Receive in Horizontal Polarization

3. TV - Transmit in Vertical Polarization

4. RV - Receive in Vertical Polarization

The science of radar polarimetry is the analysis of these transmit and receive electric field polarization combinations, also called the radar polarimetric pulsing sequence. The sequence can implement different radar functionalities such as "Single," "Cross," or "Alternate." In Single Polarization the radar transmits and receives signals in the same polarization for multiple cycles; a cycle is a pair of Transmit-Receive states of the radar. Alternate Polarization refers to alternating the polarization of the radar between consecutive cycles. Cross Polarization refers to transmitting and receiving signals in different polarizations within the same cycle. Table 2.1 shows sample polarimetric pulsing sequences implementing different functionalities in a dual-polarized radar [10].

Table 2.1. Polarimetric pulsing sequences of dual-polarized radars

Cycle	**Single**	**Alternate**	**Cross**
1	T=H R=H	T=H R=H	T=H R=V
2	T=H R=H	T=V R=V	T=V R=H
3	T=H R=H	T=H R=H	T=V R=H
4	T=H R=H	T=V R=V	T=H R=V

2.2 Phase-Tilt System Overview

CASA has developed the X-band Phase-Tilt radar that operates at a frequency of 9.36 GHz in an effort to create a low-cost, electronically-scanned, dual-polarized, phased-array antenna scanning system. The Phase-Tilt radar system architecture is as shown in Figure 2.2.

The Phase-Tilt radar is used for distributed, collaborative, and adaptive sensing of the atmosphere. The radar is controlled by a central Meteorological Command and

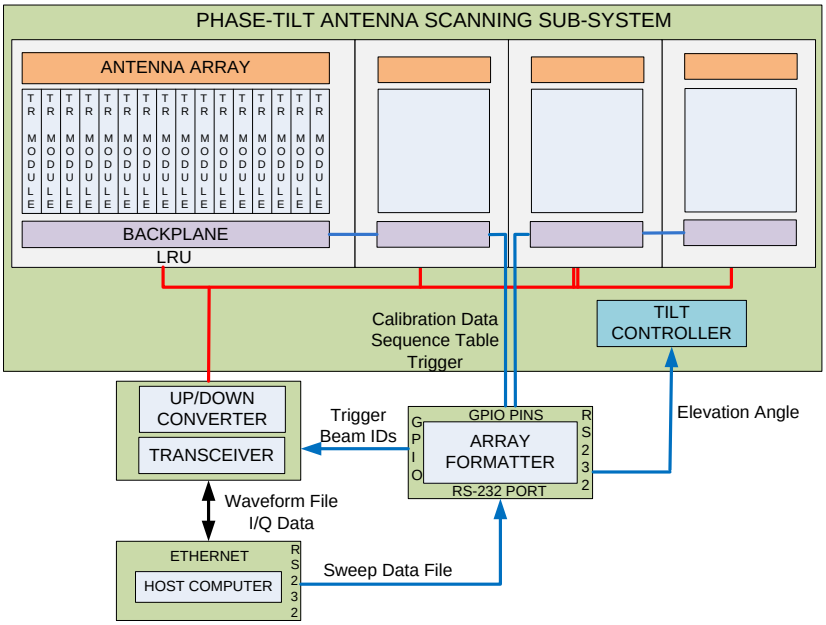

Figure 2.2. Phase-Tilt radar system architecture

Control Agent (MCC). The user scan preferences are encoded in the MCC. Depending on the preferences of different users and the state of the atmosphere, the MCC creates a "scan task." A scan task is typically a sub-volume of the atmosphere which is of particular interest to the user. The MCC creates the necessary scan command from the task, and issues the command to each one of the radar Host Computers in the network.

The Host Computer controls its radar by translating the scan commands to control files and control structures. The computer uploads these files to the AFB, and then the AFB becomes the master controller which synchronizes the different sub-systems of the radar. The next few sections describe each sub-system in detail.

2.2.1 Host Computer Sub-System

A block diagram of the Host Computer sub-system is shown in Figure 2.3. The Host Computer interfaces the human user to the rest of the system. In this project, the computer is interfaced with the Array Formatter and the Transceiver using the

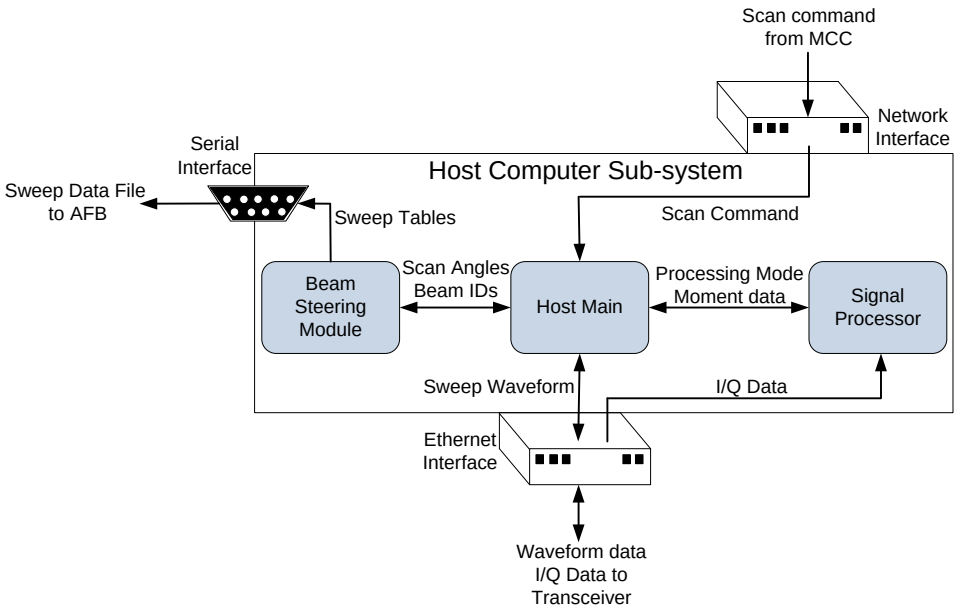

Figure 2.3. Host Computer sub-system

RS-232 and Ethernet ports respectively. The waveform data file is uploaded by the computer to the Transceiver. The signal processing component of the host computer sub-system performs real-time signal processing of the digital I and Q data from the Transceiver. The Beam Steering Module in the system records the current and future beam locations, and also calculates the "sweep" data structure (a sweep is a sequence of azimuth beam positions) from the scan command created by the end user. The sweep data file is uploaded to the Formatter via the computer's RS-232 serial interface.

2.2.2 Phase-Tilt Antenna Scanning Sub-system

The Phase-Tilt antenna scanning sub-system consists of 64 T/R Modules, antenna elements, and a Mechanical Actuator or Tilt Controller as shown in Figure 2.2. A picture of the passive antenna array is shown in Figure 2.5. The antenna scanning sub-system consists of four Line Replaceable Units (LRUs). An LRU is a black box of components that can be easily replaced or replicated depending on the needs of

the application. Each LRU consists of $0.25m \times 0.50m$ passive antenna array arranged as 16 columns and 32 rows, 16 T/R Modules that feed the 16 antenna columns, RF and DC power distribution backplane, and AC to DC power converters [14] [15]. The 64 T/R Modules are grouped into four LRUs. A T/R Module consists of three components: RF circuits, such as phase-shifters and attenuators, to set the amplitude and phase of the antenna columns, T/R and V/H switches to switch the the antenna columns between Transmit and Receive modes, and Vertical and Horizontal polarizations respectively, and an integrated Field Programmable Gate Array (FPGA) to control these circuits and interface with other radar components. The components of a T/R Module is shown in Figure 2.4 [11].

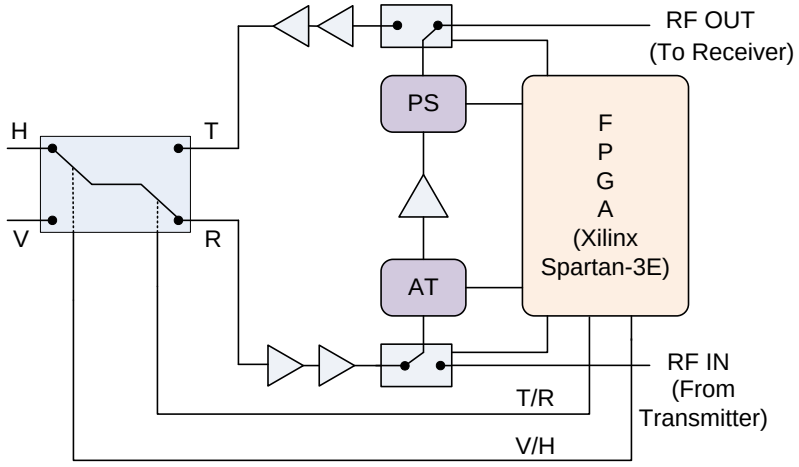

Figure 2.4. T/R Module block diagram [11]

The FPGA in the T/R Module has an internal memory, which is configured as a look-up table to store the amplitude and phase coefficients of the antenna column. The FPGA also maintains a sequence table which holds the address of the T/R Module memory. The sequence table is a data structure which contains the T/R Module states for the polarimetric pulsing sequence to be performed by the radar.

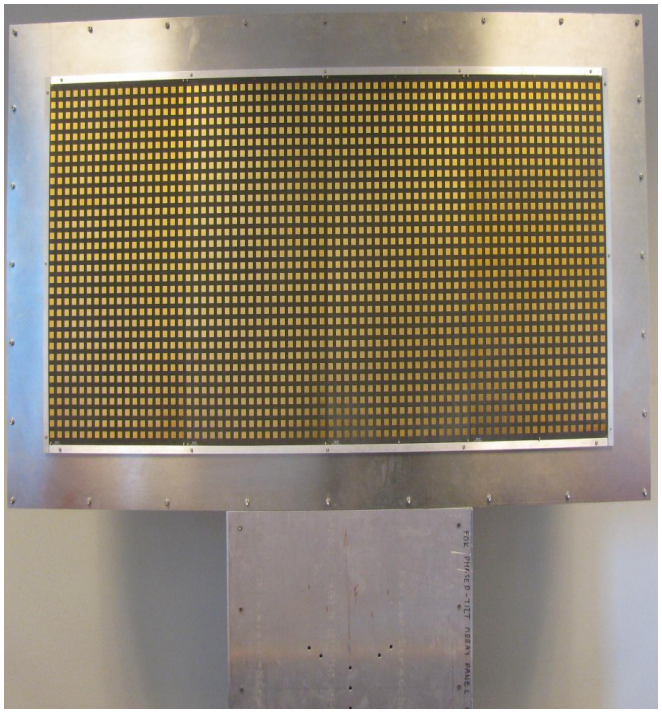

Figure 2.5. Phase-Tilt antenna array

2.2.3 Array Formatter Board (AFB)

A block diagram of the AFB is shown in Figure 2.6. The Array Formatter assumes master control of the entire radar system. The FPGA-based Formatter provides the required control and timing signals for the radar, and interfaces the Host Computer with the antenna scanning sub-system. The sweep data file from the computer is broken down by the Formatter to create sequence tables. The sequence table is broadcast to all the 64 T/R Modules. All the T/R Modules switch between their states by triggers sent by the AFB's Timing State Machine (TSM). The TSM also triggers the Transceiver sub-system. The Formatter stores the calibration data in its memory, which contains the phase and amplitude coefficients for each of the 64 T/R Modules. The calibration data is conveyed to each T/R Module individually by the Formatter. The Tilt Controller is also controlled by the AFB to update the current and future elevation angles

The Formatter is interfaced to the T/R Modules and the Transceiver via its General Purpose I/O (GPIO) pins on-board, which is capable of sending Low-Voltage Differential Signals (LVDS) for fast communication. The Formatter also interfaces with the Tilt Controller via its second RS-232 port. The elevation angle of the Tilt Controller is adjusted before pulsing the radar.

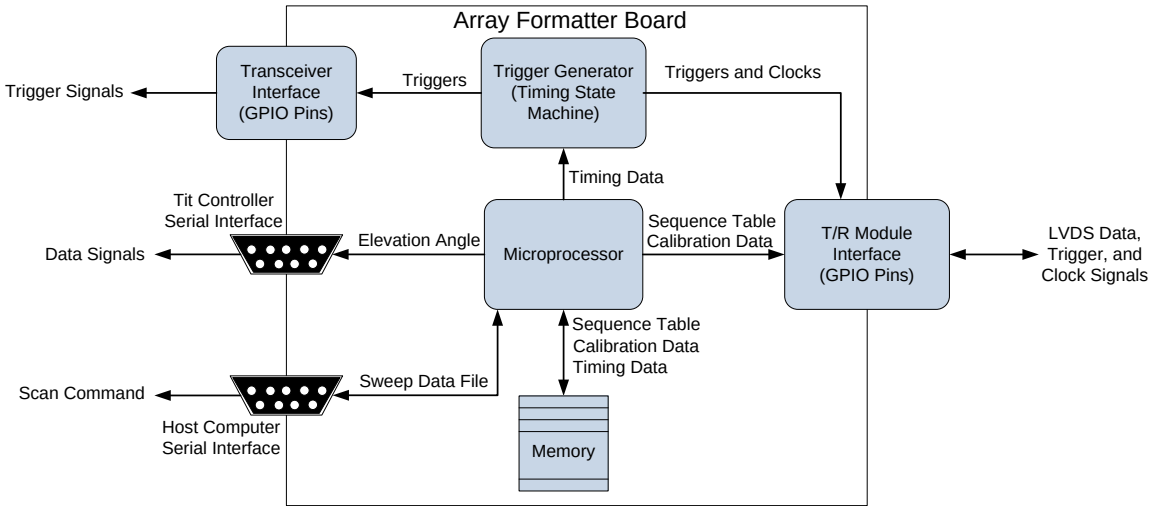

Figure 2.6. Block diagram of the Array Formatter Board

2.2.4 Transceiver Sub-system

The Transceiver sub-system consists of a commercially available Vaisala's RVP900 system [18], and an up-down converter as shown in Figure 2.7. The RVP900 creates the transmit-modulated waveform, digitizes the received signal, and demodulates the digitized signal [8]. The three inputs feeding the RVP900 are an external clock, differential TTL signals, and gigabit Ethernet as shown in Figure 2.7. The AFB sends the differential input TTL signal as a trigger to the transceiver. The gigabit Ethernet is used for interfacing the RVP900 system with the computer that uploads the waveform table and downloads the I/Q data.

The up-down conversion system is made from commercial off-the-shelf components, such as amplifiers, oscillators, mixers, etc. The up-down converter system

converts the Intermediate Frequency (IF) analog signals to Radio Frequency (RF) analog signals when the radar is in transmit mode, and does the reverse conversion when the radar is in the receive mode. The reference clock in the up-down conversion system is used as the clock input for the RVP900 as shown in Figure 2.7.

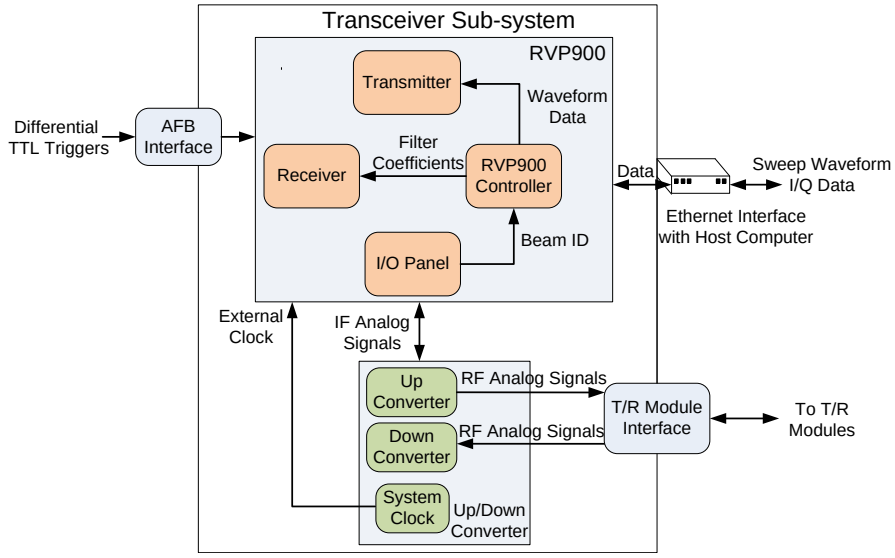

Figure 2.7. Transceiver sub-system

2.3 Phase-Tilt Modes of Operation

The Phase-Tilt radar fundamentally operates in two modes - Initialization and Operation.

Initialization: Communication paths between each sub-system and component in the radar system are set up. Registers are reset, all amplifiers are turned off, and each sub-system has its own power up sequence to be performed during the initialization phase. A major part of the initialization process is spent in updating the T/R Module memory with calibration data (phase and amplitude coefficients). The calibration data from the Host Computer is conveyed to the T/R Modules via the Array Formatter Board. The AFB stores the calibration data in its memory and routes it to its T/R Module interface through its LVDS bus in a point to point (unicast)

data format. The addressed T/R Module echoes the message back to the Formatter Receiver, which verifies the correctness of the data transmitted. The error check enables correct loading of the T/R Module memory as initialization is an important step towards successful and reliable radar operation. The T/R Module memory is a look-up table that is indexed by the sequence table to populate the antenna columns with phase, amplitude, and polarization diversity during the radar operation mode. The Formatter updates the T/R Module memory periodically or whenever the system performance changes due to external parameters.

Operation: The scan command which is a sequence of azimuth beam positions is translated to appropriate sweep data files and then uploaded to the AFB memory by the computer system. Each azimuth beam position has an associated 8 polarization sequences (sequence table), the number of times the sequence is repeated (the loop count), the transmit pulse length, and the Pulse Repetition Time (PRT), which is the time elapsed between the beginning of one pulse to the beginning of the next pulse. The AFB first loads the elevation to the Tilt Controller, before broadcasting the sequence tables to all the T/R modules in a point to multipoint (broadcast) data format. The pulse length, loop count, and PRT are given as inputs to the TSM. Once all the T/R Module sequence tables have been updated, the Formatter starts the TSM to pulse the radar. One pass through the TSM steps the T/R Modules through the sequence table once, generating 4 pulses. The TSM repeats the sequence until the loop count decrements to zero. The loop count number is chosen by the user depending on the number of pulses needed for a particular beam location. Once the loop count decrements to zero, the Formatter reads the second beam position from its memory, and broadcasts the associated sequence to the T/R Modules, and the whole process repeats. When all the azimuth beam positions for a particular elevation angle are read out of the memory, the computer uploads the next scan command to the Formatter.

CHAPTER 3

ARRAY FORMATTER BOARD

This chapter describes the Array Formatter Board in detail. The specifications from a radar standpoint are listed below first, followed by the features of the board used for implementing and testing the Formatter design.

3.1 Specifications

1. Support for real-time data acquisition and processing.

2. Support for high-speed and low-speed peripherals for communicating with different radar components.

3. Support for debug interfaces for the system development.

3.2 Array Formatter Board Features

The Spartan-3E Starter Kit board by Xilinx [25] is used for designing and testing the Array Formatter. The Spartan-3E Starter Kit as shown in Figure 3.1, is a development board that can be used for various embedded processing applications, and includes many peripherals, such as memory controllers, general purpose I/Os, and bus interfaces.

Some of the key components of the board are Spartan-3E FPGA (XC3S500E), 50 MHz on-board oscillator, 16 MB Parallel Flash (FLASH), 64 MB DDR SDRAM (DDR), 4 MB Xilinx Serial Platform Flash PROM, two RS-232 Serial Ports, Ethernet 10/100 Phy, JTAG USB download, and expansion connectors for I/Os.

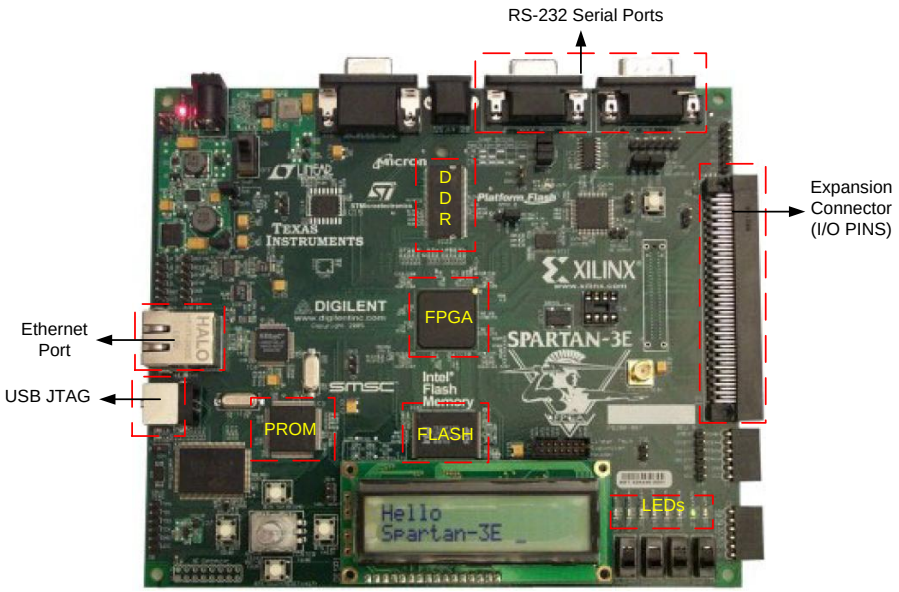

Figure 3.1. Spartan-3E Starter Kit Board [25]

The serial port is used for debugging and communicating with the computer. The other serial port is used for communicating with the Tilt Controller. The expansion I/O connector is used for interfacing the T/R Modules with the board. The 64 MB DDR SDRAM is used as the AFB memory that stores calibration data files, sequence tables, and also the timing information for the TSM. The DDR SDRAM is also used for executing the software application of the MicroBlaze processor, which is explained in chapter 6. 16 MB Parallel Flash is used to store the MicroBlaze software application. The 4 MB Serial Platform Flash PROM is used for storing the FPGA configuration file.

3.2.1 Spartan-3E XC3S500E FPGA

The XC3S500E FPGA is a part of the Spartan-3E family of FPGAs by Xilinx. The Spartan-3E FPGA family architecture fundamentally consists of five programmable logic elements [24]:

Configurable Logic Blocks (CLBs) contain flexible LUTs that implement logic and memory. CLBs can implement a wide variety of functions and also store data.

Input/Output Blocks (IOBs) control the signal flow between the I/O pins and the internal logic. It supports a variety of signaling standards including Low Voltage Differential Signaling (LVDS) standards for high speed signaling. The Formatter communicates with the T/R Modules using the LVDS standard.

Block RAMS that can be used for data storage.

Multiplier Blocks can accept two 18-bit inputs to calculate the product.

Digital Clock Managers (DCMs) provide high quality self-calibrating clock signals, and digital solutions for delaying, phase-shifting, multiplying, and dividing the clock signals. DCMs are used for providing three different clock frequencies in the system as required by the MicroBlaze system, DDR SDRAM memory controller, and the T/R Module interface bus. The details of the clocking architecture are explained in section 3.2.2.

The summary of the XC3S500E FPGA attributes is given in Table 3.1.

Table 3.1. Summary of Spartan-3E XC3S5000E FPGA Attributes

Feature	**Count**
Total Gates	500K
Total Cells	10,476
Total CLBs	1,164
Total Slices	4,656
Distributed RAM Bits	73K
Block RAM Bits	360K
Dedicated Multipliers	20
Total DCMs	4
User I/Os	232

3.2.2 Clocking Architecture

The Spartan-3E Starter Kit board supports up to three clock sources:

1. An on-board 50 MHz clock oscillator.

2. External clock can be given via the SMA connector.

3. Optionally install an 8-pin DIP-style clock oscillator.

A 100 MHz external clock signal is supplied to the Formatter via the SMA connector on the board.

Appropriate I/O and clock regions are chosen for interfacing the FPGA with the external memories, external clock oscillator, and the I/O expansion header to avoid excessive delays in the clock path. There are four DCM sites available in the FPGA. The four clock regions, global clock pins and I/O banks are shown in Figure 3.2.

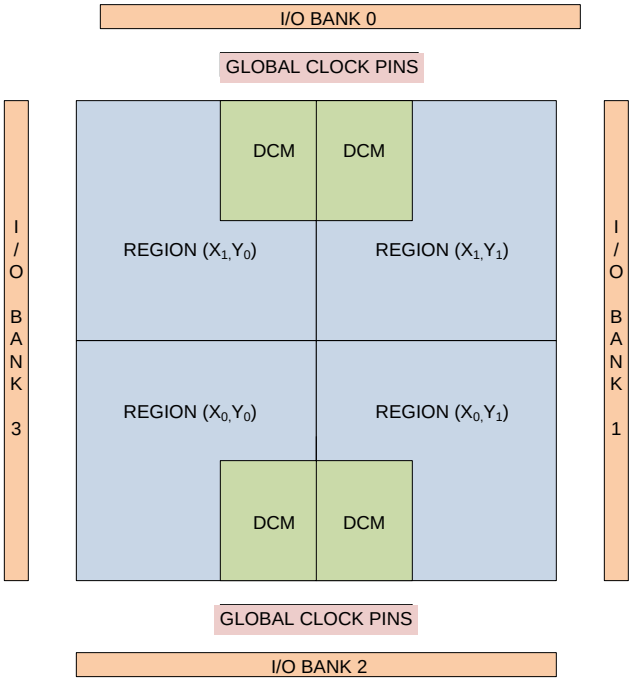

Figure 3.2. Clock Regions, DCM sites, and I/O banks in the FPGA

3.2.3 Expansion Connectors

The Spartan-3E Starter Kit board has support for several expansion connectors for conveniently interfacing with other off-board components. There is support for three I/O expansion headers:

1. A Hirose 100-pin expansion connector with support for 43 FPGA user-I/O pins, including 15 LVDS I/O pins and 2 input only pins.

2. Three 6-pin peripheral module connections.

3. Landing pads for Agilent connector-less probes.

The Hirose 100-pin expansion connector is used for interfacing the T/R modules and the Transceiver. Data and clock signals in LVDS format are exchanged serially between the T/R Modules and the board.

3.2.4 External Memories

The Spartan-3E Starter Kit board includes three external memory chips that are used in this project. The first is a 512Mbit ($32M \times 16$) DDR SDRAM by Micron [12] with a 16-bit data interface. The DDR SDRAM chip is used for storing calibration data, sequence tables, timing information, and the loop count value. The microprocessor in the system also executes its software code from the DDR SDRAM. All the DDR SDRAM pins are connected to the I/O Bank 3 of the FPGA [25].

The second memory chip used in this project is a 16 MB Intel StrataFlash (Parallel Flash). The non-volatile Parallel Flash provides a number of options to the designers as listed below.

1. Support for storing a single FPGA configuration file.

2. Support for storing two different configuration files to switch dynamically between two FPGA configurations using the board's multi-boot feature.

3. Support for storing the microprocessor software code, and also shadows the code to the DDR SDRAM, from where it can be executed by the processor.

4. Support for storing any non-volatile data from the FPGA.

In this project, we use the Parallel Flash for storing the processor code, and also to shadow the code to the DDR SDRAM memory. The processor eventually executes from the DDR SDRAM and more details about this process are given in Chapter 6. The third memory chip used in this project is on-board 4 MB Xilinx XCF04S Serial Platform Flash PROM. In this project, the PROM is used to store the FPGA configuration file. The FPGA is configured from the PROM in the Master Serial configuration mode [25]. Details about this configuration mode can be found in [25].

3.2.5 RS-232 Serial Ports

The Spartan-3E Starter Kit board has two RS-232 serial ports: a male DB9 DTE connector and female DB9 DCE connector as shown in Figure 3.3. The DTE connector is used for interfacing the AFB with the Host Computer using a null modem serial cable. The DCE connector is used for interfacing the AFB with the mechanical actuator. The output voltage levels of the FPGA is in LVCMOS or LVTTL standards, which are converted to the RS-232 voltage standards by the voltage converter IC as shown in Figure 3.3. Likewise, when the FPGA receives serial data from either of the ports, the RS-232 voltage levels are converted back to LVCMOS standard by the voltage translator IC.

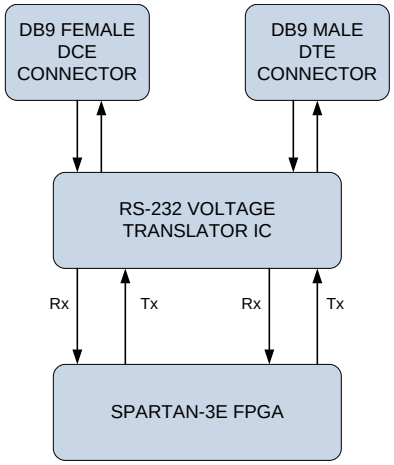

Figure 3.3. RS-232 serial ports

3.2.6 Ethernet Physical Layer Interface

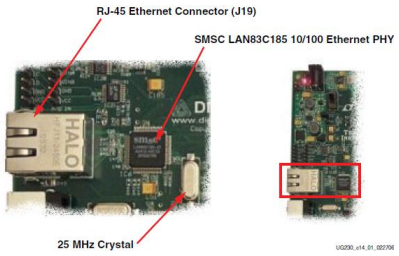

Figure 3.4. Ethernet PHY with RJ-45 connector [25]

The AFB also has support for an Ethernet physical layer to implement the standard Ethernet interface. The board includes a Standard Microsystems LAN83C185 10/100 Ethernet physical layer (PHY) interface and an RJ-45 connector as shown in Figure 3.4. All timing is controlled by the on-board 25 MHz crystal oscillator. The Ethernet interface is implemented in this work to enable high-speed communication between the Formatter and the Host Computer. This interface can replace the serial link to achieve a faster and more reliable communication channel. Details about the implementation of the ethernet interface are given in Chapter 7.

CHAPTER 4

MICROBLAZE AND EMBEDDED DEVELOPMENT KIT

This chapter gives a brief overview of the MicroBlaze-soft processor by Xilinx and also the environment used to create the embedded system on the FPGA.

4.1 MicroBlaze Soft Processor

The central processing component of the AFB is a soft microprocessor by Xilinx called MicroBlaze. The XC3S500E Field Programmable Gate Array has support for MicroBlaze, which is completely reconfigurable and built by combining various logic cores inside the FPGA. The MicroBlaze processor is an embedded soft 32-bit RISC processor [20]. The processor has high-speed instruction and data buses to guarantee access to instructions and data simultaneously both on and off the chip.

MicroBlaze supports both general purpose and special purpose registers. The processor has a three stage pipeline and a 32-bit high speed bus [20]. Two of the bus interfaces supported by MicroBlaze that are used in this project are:

Processor Local Bus (PLB): A fully synchronous bus that gives access to processor peripherals.

Local Memory Bus (LMB): A high-speed bus that connects MicroBlaze to peripherals like Block RAMs for high-speed access.

4.2 Embedded Development Kit

The Xilinx Embedded Development Kit (EDK) is a design suite that helps in designing a complete embedded system [22]. The two main tools used in EDK are Xilinx Platform Studio and Software Development Kit.

4.2.1 Xilinx Platform Studio

The Xilinx Platform Studio (XPS) facilitates the design of the hardware components of the embedded processor system on an FPGA. The tool provides a graphical user interface for interconnection of the various components of the system, such as processors, peripherals, and buses. Apart from the standard IP cores that are available in the IP catalog [19], the XPS allows the creation of custom peripherals that can be included in the system. The XPS uses IP Interface (IPIF) libraries to implement basic functionalities between processors and peripherals. These are optimized and parameterized to give a simplified Bus Protocol called IP Interconnect (IPIC) [22].

4.2.2 Software Development Kit

The Xilinx Software Development Kit (SDK) provides an environment for developing embedded software applications. In essence, the SDK helps in developing software for the hardware system developed in XPS. A Software Platform is the lowest level of the software stack that includes all the drivers and libraries of the components of the embedded system. Many applications can share the same software platform. A standalone software platform is created to execute single threaded applications on a single processor system.

CHAPTER 5

DESIGN METHODOLOGY AND HARDWARE SYSTEM ARCHITECTURE

5.1 Design Methodology

The hardware components of the embedded system are designed using XPS and the software application is developed using SDK as explained in section 4.2. The software application is developed using C. The custom hardware logic in the FPGA is designed in Verilog, and the embedded system designed in EDK is instantiated as a module in the main Xilinx ISE project.

5.2 AFB FPGA Hardware System Architecture

The Array Formatter design is implemented on the Xilinx Spartan-3E FPGA and the system architecture is illustrated in Figure 5.1. The FPGA system architecture has the following key components:

1. MicroBlaze Processor.
2. Xilinx IP Cores such as UART, Digital Clock Managers, Memory Controllers, and FIFOs [19].
3. Custom Peripheral (PF)
4. Custom Verilog Modules (T/R Module Interface, Timing State Machine, and Transceiver Interface)

Details of the MicroBlaze soft processor have been described in section 4.1. The next few sub-sections describe the other components in the system.

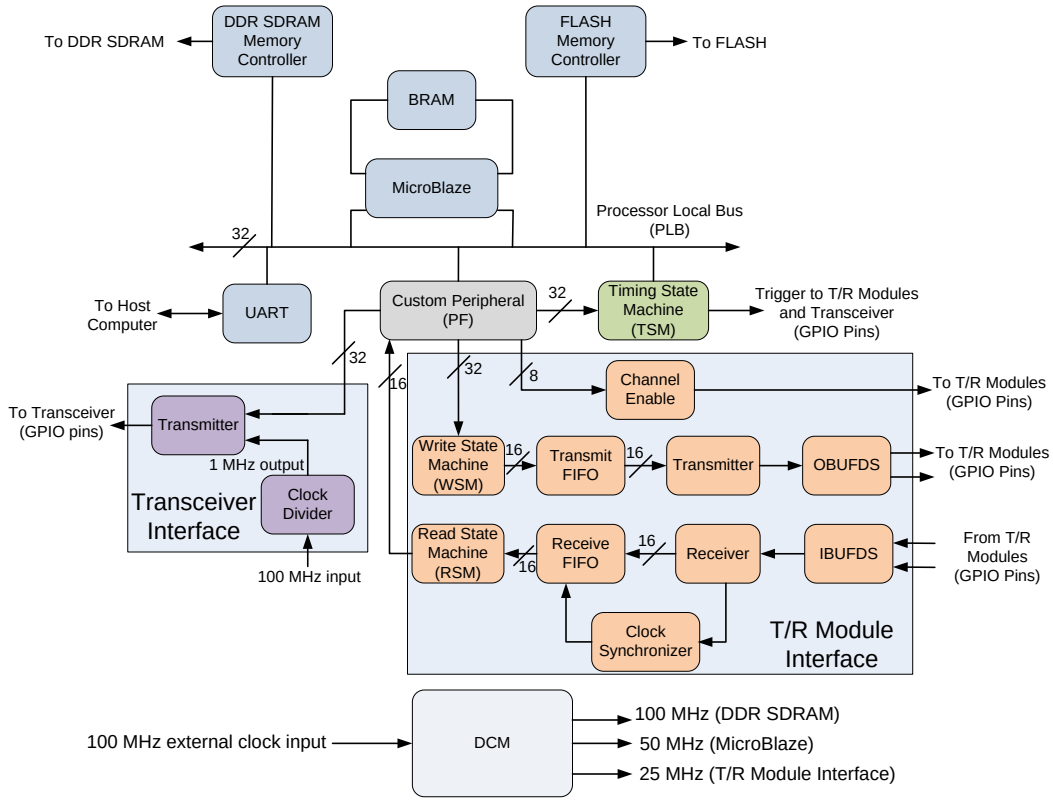

Figure 5.1. Array Formatter system architecture

5.2.1 Xilinx IP Cores

The Xilinx IP cores facilitate ease of design and integration into the system. This section describes the various IP blocks that have been used to design the system on the Formatter FPGA.

5.2.1.1 Memory Controllers

Xilinx IP cores such as Multi-Port Memory Controller for the DDR SDRAM and the External Memory Controller for Flash are used for controlling the two external memory chips [19]. These two cores are attached to the Processor Local Bus (PLB) of the system.

5.2.1.2 UART Core

The Xilinx UART core is used for communicating with the Host Computer. It has an adjustable baud rate and experiments are conducted at a rate of 115 Kbps.

5.2.1.3 Digital Clock Manager

The Digital Clock Managers are used to provide three different clock frequencies: 25 MHz, 50 MHz, and 100 MHz. The FPGA is triggered by an external 100 MHz clock oscillator, and the DCM generates three output frequencies: 25 MHz, 50 MHz, and 100 MHz. The MicroBlaze system runs at 50 MHz, while the T/R Module interface runs at 25 MHz to satisfy the serial bus load requirements connected to the T/R Modules, and the DDR SDRAM memory controller runs at 100 MHz as the operating frequency range of the memory chip is between 75 MHz and 133 MHz [12].

5.2.1.4 First In First Out Memories

Xilinx FIFO cores are used in a system whenever data or control signals are exchanged between two different clock domains. Two FIFO cores (Transmit and Receive FIFOs) are used in this system to facilitate communication between the MicroBlaze processor and the T/R Module interface. The Transmit FIFO buffers the data to be transmitted to the T/R Modules. Likewise, the Receive FIFO holds the data received from the T/R Modules before being read out by the processor.

5.2.2 Custom Peripheral (PF)

The Custom Peripheral, called PF in this system, is added to facilitate communication between the processor and the rest of the custom hardware Verilog modules (T/R Module Interface, Transceiver Interface and Timing State Machine). These Verilog modules are described in the next section. In order to add a custom peripheral to the PLB of the MicroBlaze system, the "Create Custom Peripheral" wizard is used in Xilinx XPS [22]. This wizard also generates the necessary device drivers that

are used in developing the software application for the MicroBlaze processor. The processor communicates with PF by making use of PF's software accessible registers. In this work, the PF is configured to have 30 software accessible registers. These registers are mapped to the different ports of the custom Verilog modules, and are listed in table A.1 in Appendix A. In order to access this peripheral in software, the applications makes use of the APIs defined in PF's device driver. The software APIs used for accessing this peripheral are described in Appendix A. After creating this peripheral, the Xilinx PLB interface modules are automatically included in the HDL files of PF. These PLB interface modules take care of all bus transactions between the processor and the peripheral.

5.2.3 Custom Verilog Modules

The custom Verilog modules in this system are used for communicating with the T/R Modules and the Transceiver. The three interfaces are described in the following sections.

5.2.3.1 T/R Module Interface

The Array Formatter Board communicates with the T/R Modules to convey the sequence table information, update each T/R Module memory with calibration coefficients, and trigger the switching of T/R Modules during the operation mode of the Phase-Tilt radar.

The Transmitter and Receiver blocks are used for transferring data between the Formatter and T/R Modules. A similar pair of Transmitter and Receiver blocks is configured in the FPGA of the T/R Modules. The Transmitter block on either side is the master, which sends data and clock using the LVDS standards via the I/O pins on each board.

Two Finite State Machines [13] are implemented to control the Transmit and Receive FIFOs in this interface. The PF loads the Write State Machine (WSM)

with a 32-bit data for the T/R Modules. The WSM then writes this data as two 16-bit words to the Transmit FIFO. The Transmitter module is a serial shift-out register that reads the 16-bit data from the Transmit FIFO, and sends the data serially as LVDS signals through the Xilinx OBUFDS (Output Buffer for Differential Signalling) primitive. Likewise, in the receive chain, the LVDS data signals from the T/R Modules are converted into a single ended signal by the Xilinx IBUFDS (Input Buffer for Differential Signalling) primitive. The Receiver module performs the conversion to a 16-bit data, and then writes it to the Receive FIFO. The Read State Machine (RSM), which is controlled by PF, reads the value out of the Receive FIFO. The Receiver operates at 25 MHz and the Receiver FIFO operates at 50 MHz with a clock synchronizer between the two modules.

Serial Data Format: The serial data transferred between the Formatter and the T/R Modules follows a simple protocol, where the message is represented as 18 bits as shown in Figure 5.2. The message includes 1 start bit, 16 data bits, and 1 stop bit. The start bit, which is a zero, indicates the start of data transmission. This is followed by the 16-bit data to be transmitted. The transmission ends with the stop bit, which is a one. Data is sampled at the negative edge of the clock cycle, and the last clock is used to manage the data flow in the Receiver.

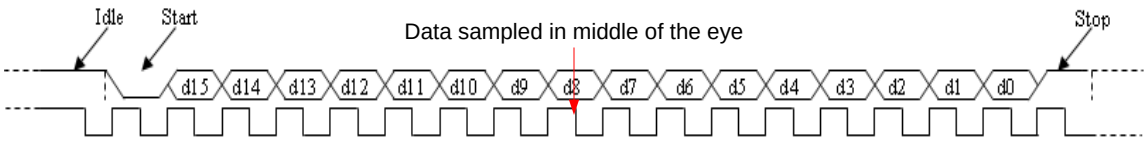

Figure 5.2. Serial data format

Channel Enable:

The array of sixty-four T/R Modules is arranged as two groups of 32 Modules each. Each group has its own channel through which data is transmitted to the Formatter. T/R Modules 1 to 32 are grouped to the left channel, while Modules 33 to 64 are

grouped to the right channel. The T/R Modules are configured in a way such that, the loaded calibration data words are echoed back to the Formatter during the radar initialization phase for error check. In order to enable one of the two channels during this phase, a channel enable module is implemented in the T/R Module interface of the Formatter's FPGA. This logic enables the left or right T/R Module channel, depending on its address. The channel enable module is also controlled by PF, and it gets the T/R Module address from one of the software accessible registers of PF.

5.2.3.2 Timing State Machine (TSM)

The TSM is implemented in the Formatter for triggering the T/R Modules and the Transceiver as explained in section 2.2. The pulse length, pulse repetition time, and Loop Count are read out from the DDR SDRAM memory by the MicroBlaze processor, and fed as inputs to the TSM for an azimuth beam position. The TSM goes through 32 states, and one iteration of the 32 states passes the T/R Modules through the sequence table once, thus generating 4 pulses. The 32 states are repeated for a particular beam location until the loop count decrements to zero. The Loop Count value determines the number of pulses needed by the Phase-Tilt radar in an azimuth beam position.

In order to pulse the radar, the TSM first sends a trigger to the Transceiver. This trigger switches the Transceiver to the Transmit state, which creates the output waveform. After a certain delay, the TSM triggers the T/R Modules to switch to the Transmit state. Once the Transmit time of the pulse elapses, the TSM triggers the T/R Modules to switch to the Receive state. The T/R Modules stay in this state until the Receive time elapses, and the TSM then triggers the Transceiver to start the next Transmit state. This sequence of triggers continues until the TSM Loop Count value decrements to zero. This corresponds to the Phase-Tilt radar transmitting a

fixed number of pulses in an azimuth beam position. The outputs of the TSM is shown in Figure 5.3.

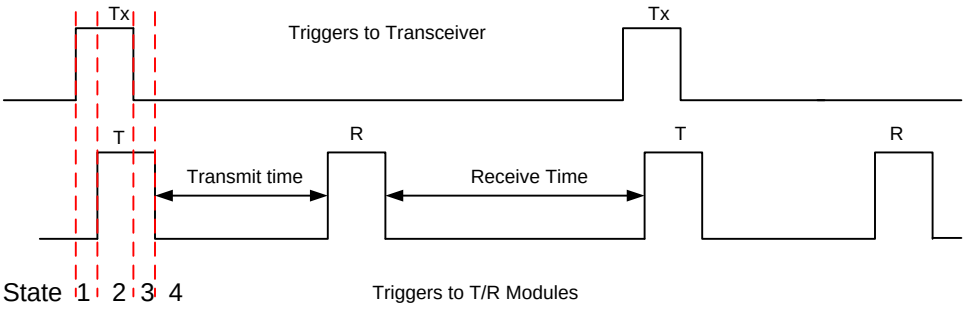

Figure 5.3. Timing state machine trigger outputs

5.2.3.3 Transceiver Interface

The Formatter communicates with the RVP900 Transceiver to convey the scan information before pulsing the radar. The Transceiver synchronizes the received radar data from the antenna panel with this scan information. This scan data contains information about the beam position and the polarization pulsing sequence performed by the radar.

The Transceiver interface consists of three lines: clock, enable signal, and a 32-bit serial data interface. The serial data interface is also another Transmitter, that shifts out a 32-bit data serially out through the GPIO pins on the board. The interface runs at 1 MHz, and hence, a clock divider is used to derive this clock rate from the external 100 MHz clock.

The implementation and verification of the custom Verilog modules are discussed in Appendix B. The AFB ports in these modules are used for communicating with the Phase-Tilt radar components. Except for clocks, all the data lines are sent serially. These ports are listed in Table 5.1.

The hardware utilization of the Spartan-3E FPGA for implementing the Array Formatter design is shown in Table 5.2.

Table 5.1. Array Formatter Board ports

Port	Data Width	Data Rate or Frequency
UART Transmit	1	115 Kbps
UART Receive	1	115 Kbps
T/R Module Interface Transmit	1	25 MHz
T/R Module Interface Receive	1	25 MHz
T/R Module Interface Clock	1	25 MHz
Transceiver Interface Data	1	1 Mbps
Transceiver Interface Clock	1	1 MHz
TSM Trigger	2	Variable

Table 5.2. Hardware utilization of the AFB FPGA

Logic Utilization	Used	Available	Percentage Utilization
Slices	3287	4656	70%
Flip Flops	3741	9312	40%
LUTs	3816	9312	40%
BRAMs	11	20	55%
DCMs	2	4	50%

5.3 T/R Module FPGA Architecture

The T/R Module includes a Spartan-3E FPGA, which controls the RF components and the switches in the T/R Module. The architecture of the T/R Module FPGA is shown in Figure 5.4.

The FPGA has an on-chip memory module, which contains phase and amplitude coefficients (calibration data) for an antenna column. During the radar initialization phase, the memory of each T/R Module is preloaded with calibration data. The look-up table memory is 16 bits wide as shown in Figure 5.5, with 6 bits each for setting the attenuator and phase-shifter, and 4 bits for holding the information about T, R, H, and V states. The memory has 1K entries, and is divided into four segments for

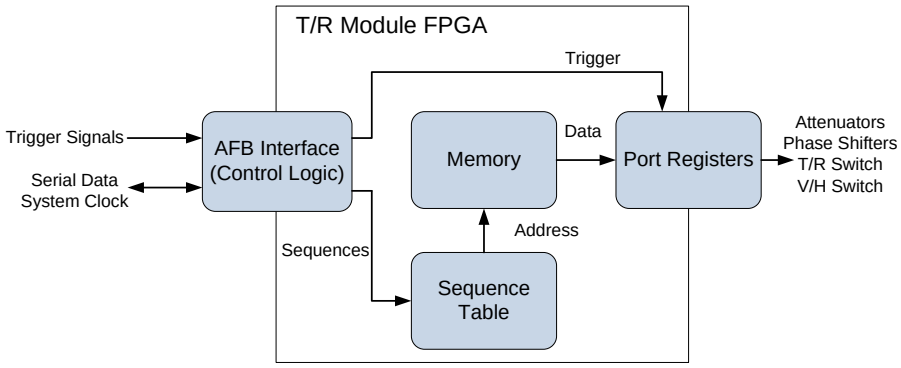

Figure 5.4. T/R Module FPGA

the four T/R Module states (TH, TV, RH, RV). The set of calibration data for each T/R Module state corresponds to 256 different beam positions in an azimuth plane.

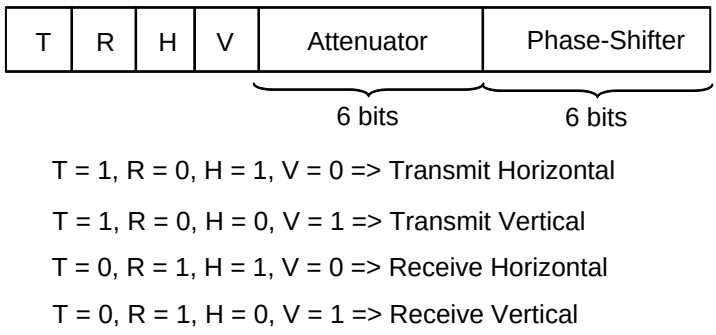

Figure 5.5. T/R Module memory bits

The FPGA also contains a sequence table that addresses the look-up table memory. The polarimetric pulsing sequence performed by the radar is determined by the eight entries in the sequence table. The sequence table, as shown in Figure 5.6, is 16 bits wide. The first two bits represent the mode. This instructs all the T/R Modules in the array to listen to the Array Formatter and update their sequence tables. The four temperature bits are latched by the temperature register in the T/R Module. These bits update the temperature of the on-board thermal sensor. The T/R and V/H bits select one of the four segments in the look-up table memory, which repre-

sents the T/R Module state. The eight beam position bits is used for selecting one of the 256 azimuth beam positions in the selected memory segment.

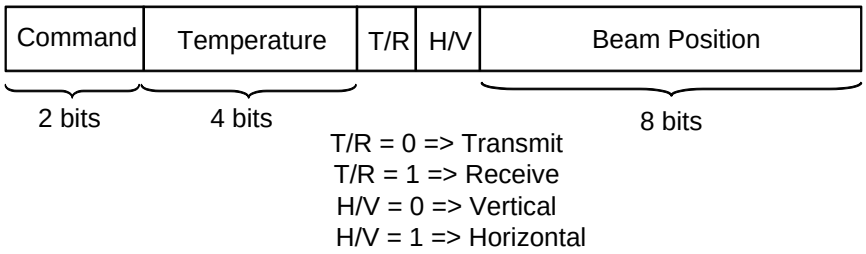

Figure 5.6. T/R Module sequence table bits

At the rising edge of the trigger pulse sent by the Array Formatter's TSM, the data in the memory that is addressed by the sequence table is latched by the port registers. The port registers are interfaced to the attenuators, phase-shifters, T/R and V/H switches to adjust the antenna column parameters. In this way, the T/R Module FPGA communicates with the Array Formatter and controls the operation of the Phase-Tilt radar.

CHAPTER 6

ARRAY FORMATTER SOFTWARE

This chapter describes the software application details of the Array Formatter that has been implemented and tested on the Spartan-3E FPGA.

6.1 Booting the System

The Xilinx Spartan-3E FPGA features the ability to configure from the Xilinx PROM available on-board in the Master Serial configuration mode [25]. On power up, the initial configuration loads the FPGA with the hardware components along with the MicroBlaze software application which gets loaded into the Block RAM. The MicroBlaze processor executes the application from the Block RAM as soon as the FPGA is configured. The capacity of the Block RAM is 8 Kilo Byte because of the limited available hardware resources. Hence the software bootloader technique is used in this thesis work to execute the Formatter software application that is larger than 8 KB.

6.1.1 Bootloader

A bootloader is a software piece of code that gets loaded along with the configuration bitstream while programming the PROM. The Intel Parallel Flash is used in this project to store the main software application code. The sequence of steps that take place during the bootloading operation is described below:

1. On power up, the FPGA gets configured by the PROM, and the processor executes the bootloader from the processor reset location.

2. Bootloader copies the image of the software code from the Parallel Flash to the DDR SDRAM as shown in Figure 6.1.

3. Processor executes the software code from the DDR SDRAM.

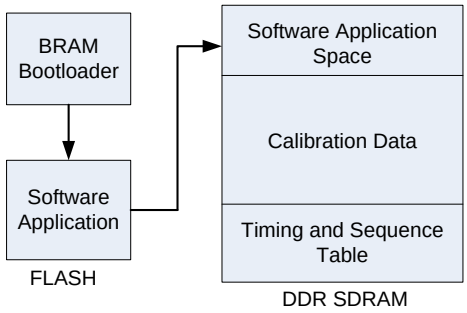

Figure 6.1. Bootloader illustration

The software application that is built from the project has the Executable and Linkable Format (ELF). ELF image files are generally not used for storing and bootloading as it increases the complexity of the bootloader code [21]. The ELF image is instead converted to SREC (Motorola S-record) format, which is one of the common bootloadable image formats, and hence, helps the process of bootloading. The concept of bootloading is shown in Figure 6.1.

6.2 MicroBlaze Software Application

The Array Formatter software application is developed in C. The Formatter prints the success of the bootloader process to the Host Computer, and then starts to execute the application code from the DDR SDRAM.

All peripheral device drivers are included in the code as header files. These files give access to the peripheral APIs. The drivers help users to treat the peripheral as a black box, and not deal with any complex bus transactions between the processor and the peripheral. For example, the user can instruct the MicroBlaze processor to write a 32-bit data to a specific address location in the DDR SDRAM, by calling the function

XIo_Out32(Address, Data). The processor recognizes the address space of the DDR SDRAM between its base and high addresses. The DDR SDRAM chip is accessed as a memory-mapped I/O, and any memory read or write operation is specified within this address range. Similarly, all other peripherals are accessed as memory-mapped I/Os by the processor, and the device driver APIs are used to communicate with them in the software application. These APIs are described in Appendix A.

The Formatter has three communication interfaces on the chip that are implemented in Verilog. These are the T/R Module interface, Transceiver interface, and the TSM. The interfaces are port-mapped to the software accessible registers of the Custom Peripheral (PF). The registers are accessed in software by the APIs defined in the PF device driver. The PF is configured to have 30 software accessible registers, and these are mapped to all the three communication interfaces. For example, a software write to the T/R Module interface corresponds to writing a 32-bit data to the FIFO WSM. In order to write the data to the FIFO WSM, the processor writes into the PF register that corresponds to the input data port of the WSM. The processor then enables the FIFO WSM in software by setting the corresponding PF register to one. Likewise, the processor performs a software read operation from the T/R Module interface by first starting the FIFO RSM in software, and then reading the data from the output port of the RSM. The processor uses the corresponding PF registers that are mapped to the RSM enable and data output ports. These functions along with other software communication interfaces have been described in Appendix A.

6.2.1 Software Functions

The MicroBlaze software functions can be grouped into two schemes: Unicast and Broadcast schemes. The Unicast scheme corresponds to the Phase-Tilt radar initialization mode, where the calibration data is loaded in the AFB's memory, and then routed to the corresponding T/R Modules. The Broadcast scheme, on the other

hand, is associated with the radar operation mode. In this scheme, the sequence table and timing information for multiple beam positions in an azimuth plane are loaded in the memory. The processor then reads them out from the memory, and writes to the T/R Module interface and the TSM respectively.

The MicroBlaze application program begins in a while loop that makes a call to access the UART receive buffers for the Host Computer command. The program enters a case statement that calls the different software functions depending on the 8-bit command decoded by the processor. If the command is 'z' or "exit," the application sets the command flag to 1. The application while loop terminates when the command flag is set to 1. The MicroBlaze software functions are explained below:

6.2.1.1 Unicast Scheme

In the unicast scheme, the Formatter addresses each T/R Module individually, and waits for the T/R Modules to echo the words back for error check. The Formatter enables the left or right channel of the T/R Modules to transmit data. The left channel is enabled if the Formatter receives data from T/R Modules 1 to 32, and the right channel is enabled for T/R Modules 33 to 64. The following are the five MicroBlaze software functions under the Unicast scheme.

Store Calibration Data:

If the command decoded by the Formatter is 's,' the store calibration data function is executed by the processor. The T/R Module calibration coefficients are stored in the Formatter's DDR SDRAM memory segment. The Formatter's memory is divided into segments of 4 KB to store the T/R Module calibration data. The steps taken by the MicroBlaze processor to implement this function in software is given below:

1. Receive T/R Module address as the next byte of data.

2. Calculate the DDR memory segment address for the T/R Module.

3. Calibration data are then formatted as 32-bit words by concatenating four consecutive 8-bit words from the UART receiver.

4. Store 32-bit words in memory, until the of command (0xFFFF) is received.

Transmit Calibration Data:

If the command decoded by the Formatter is 'f,' the application enters the "Transmit Calibration Data" function. The calibration data that is stored in DDR SDRAM is transmitted to the T/R Module Interface by loading each data word into the WSM. This function is used for loading the T/R Module look-up table memory with calibration data. A 16-bit command is created and transmitted by the Formatter. This command instructs the addressed T/R Module to update its memory with calibration data. The steps taken by the Formatter to implement the function are explained below:

1. Receive T/R module address and calculate the DDR memory segment address.

2. Create the command for the addressed T/R module to update its memory and write to the WSM.

3. Enable one of the T/R Module channels to receive the echoed data.

4. Read all the calibration data from DDR SDRAM for the addressed Module, and write to the WSM.

5. Wait until all words are read back from the RSM.

The data format for updating the T/R Module look-up table memory is shown in Figure 6.2. The Formatter creates and transmits the command for updating the T/R Module memory. The command is followed by another word that indicates the total number of data words, "N." This is followed by the "N" data words that are written into the T/R Module memory.

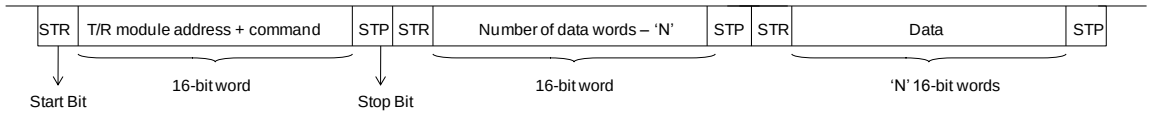

Figure 6.2. Data format for updating T/R Module memory

Write T/R Module Port Registers:

If the Host Computer command is 'p,' the application enters the "Write T/R Module Port Registers" function. The steps are explained below for this mode of operation.

1. Receive T/R module address and the port register data from the Host Computer.

2. Create the command for the addressed T/R Module to update port registers, and then write the command and data to the T/R Module interface (WSM).

3. Enable one of the T/R Module channels to receive the echoed data.

4. Wait until the the two 16-bit words (command and data) are read out from the RSM.

On receiving the data and command from the Formatter, the T/R Module FPGA bypasses the look up table memory to set the antenna column parameters directly into the port registers. The data format for communicating with a T/R Module during this mode of operation is shown in Figure 6.3. The first 16-bit word addresses the T/R Module, and also has the command for loading the port registers. The second word is the port registers data.

Figure 6.3. Data format for writing port registers

Read T/R Module Port Registers:

This function is executed when the Host Computer command is 'r.' The computer sends the command to instruct the Formatter to read from a T/R Module port registers. The steps for this mode are listed below.

1. Get T/R module address.
2. Create command to access the port registers for the addressed T/R Module, and then write the command to the WSM.
3. Enable one of the T/R Module channels to receive data.
4. Wait until the two 16-bit words (command and data) are read from the RSM.

The data format for communicating with the T/R Modules is shown in Figure 6.4. Only one 16-bit word is sent to the T/R Module, which contains both the command and the T/R Module address. Two 16-bit words are received by the Formatter, which contains both the T/R Module command and the 16-bit data from the port registers.

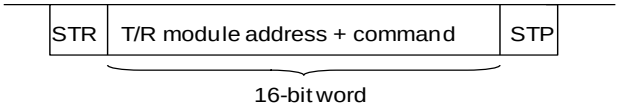

Figure 6.4. Data format for reading port registers

Write T/R Module Address Register:

The MicroBlaze processor executes this function on receiving command 'a' from the Host Computer. The purpose of this software function is to write the T/R Module port registers with the data contained in the addressed memory location of the T/R Module look-up table memory. The following are the set of steps:

1. Get T/R module address.

2. Create command for the T/R Module, and then write the command and T/R Module memory address to the WSM.

3. Enable one of the T/R Module channels to receive data.

4. Wait until the two 16-bit words (command and data) are read from the RSM.

The Formatter first addresses a T/R Module, and then addresses the location in the Module's look-up table memory. The data contained in the addressed memory location is loaded in the port registers of the T/R Module. The data format between the Formatter and the T/R Module for this function is shown in Figure 6.5.

Figure 6.5. Data format for writing address register

6.2.1.2 Broadcast Scheme

In the Broadcast scheme, the Formatter first stores the sequences and timing-related information in its memory, and then writes these data to the T/R Module interface and the TSM. The sequence table serves as the address bits for the memory look-up table in the T/R Module as shown in Figure 5.4. For each T/R Module in the antenna array, the set of amplitude, phase, T/R, and H/V values contained in the memory are loaded into the corresponding port registers at the rising edge of the trigger pulses generated by the Formatter's TSM. The T/R Module port registers are interfaced to the attenuators, phase-shifters, and the T/R and V/H switches. There are two software functions in the Broadcast scheme. The first function stores the data in the DDR SDRAM. The second function writes the data to the WSM of the T/R Module interface, and then starts the TSM.

Store Timing and Sequence Data:

If the command from the Host Computer is 'q,' the processor executes the "Store Timing and Sequence Data" function. A scan command is a set of azimuth beam position for a fixed elevation angle. Each azimuth beam position consists of Sequence Tables, Timing Information (PRT, Pulse length), and the Loop Count as explained in section 2.3. The Formatter stores the scan command in its memory. The sequence of steps is explained below.

1. Receive two 8-bit words from the computer, and create the 16-bit Loop Count variable of the TSM. Store Loop Count variable in memory as a 32-bit word.

2. Receive sixteen 8-bit words, and create four 32-bit words for the TSM timing registers. Store the four timing register values in memory.

3. Repeat step 2 for storing sequence table information as four 32-bit words in memory.

4. Repeat steps 1,2, and 3 for multiple beam positions until the Loop Count variable is received as 0xFFFF (end of command).

Begin:

The "Begin" function is executed by the processor when it receives the character 'b' from the computer as the command. This function corresponds to the radar operation mode. Sequence and timing related information from the memory are read out by the Formatter and appropriate actions are taken as explained below.

1. Read sequence table data out from the memory and write to the WSM of the T/R Module interface.

2. Create the 32-bit scan information for the Transceiver, and send the data to the Transceiver interface.

3. Read timing information and Loop Count variable from the memory, and load these information to the TSM.

4. Steps 1 to 3 are repeated for the other beam positions until all the sequence and timing information in memory have been read out to complete one scan in the azimuth plane.

Table 6.1. Host Computer commands for MicroBlaze software functions

MicroBlaze Software Function	**Scheme**	**Command**
Store Calibration Data	Unicast	's'
Transmit Calibration Data	Unicast	'f'
Write T/R Module Port Registers	Unicast	'p'
Read T/R Module Port Registers	Unicast	'r'
Write T/R Module Address Register	Unicast	'w'
Store Timing and Sequence Data	Broadcast	'q'
Begin	Broadcast	'b'

The Formatter software functions and the associated Host Computer serial commands are summarized in Table 6.1.

6.3 Formatter Data Rates

This section describes the Formatter data rates. The date rate for communicating with the T/R Modules is 25 Mbps. Each data or command sent by the Formatter is an 18-bit word which includes 16 data bits, 1 start bit, and 1 stop bit as explained in section 5.2.3. A timer test is conducted to estimate the data rates of the Formatter for communicating with the T/R Modules and the Host Computer.

Timer Test Results:

The Formatter's "Begin" function directly impacts the scanning performance of the radar, as discussed in section 6.2.1.2. The Formatter goes through the following steps before starting the TSM (pulsing the radar):

1. Read sequence table and timing information out of memory

2. Convey Sequence table information to T/R Modules

3. Update TSM registers

The above steps correspond to the minimum latency of the Formatter for switching the radar beam position, also known as the radar "Beam Loading Time." A test is conducted to measure the radar beam loading time. A timer is included in the circuit to count the number of cycles taken by MicroBlaze to execute the software code from the DDR SDRAM. It is observed that the beam loading time is approximately 15 μs.

The processor then starts the TSM, and the TSM triggers switch the T/R Module and Transceiver states. This corresponds to the radar scan time, which depends on the pulse length, PRT, and the loop count value of the TSM. In order to notify the Host Computer at the end of the first azimuth scan, the Formatter was programmed to send a 1 byte beam acknowledgment ('d' or done) signal to the computer. As the baud rate of the UART is set to 115 Kbps, the acknowledgment signal takes approximately 70 μs to reach the computer. Hence, to start the next scan from the AFB memory, the Formatter first sends the acknowledgment signal and then loads the next azimuth beam position from the memory. So the beam switching time is equal to the sum of the acknowledgment time of the completed scan and the beam loading time of the next scan. This does not include the latencies of the Host Computer software.

In order to obtain a more accurate estimate of the beam switching time, the Formatter is integrated with the Host Computer software, and the outputs of the TSM are observed on the GPIO pins of the board. The waveforms are recorded on the digital oscilloscope. One pass through the TSM generates 64 pulses. After observing 64 pulses, we observed a latency of 90 μs. This delay of 90 μs (marked as switching time in Figure 6.6) directly corresponds to the time taken to switch azimuth

beam positions. Figure 6.6 shows three waveforms, the middle green waveform is the trigger generated by the Formatter's TSM to the Transceiver. After a certain delay, the Transceiver responds by generating the IF 60 MHz signal, which is later converted to the 9.36 GHz RF signal. The purple waveform is the 100 MHz system clock.

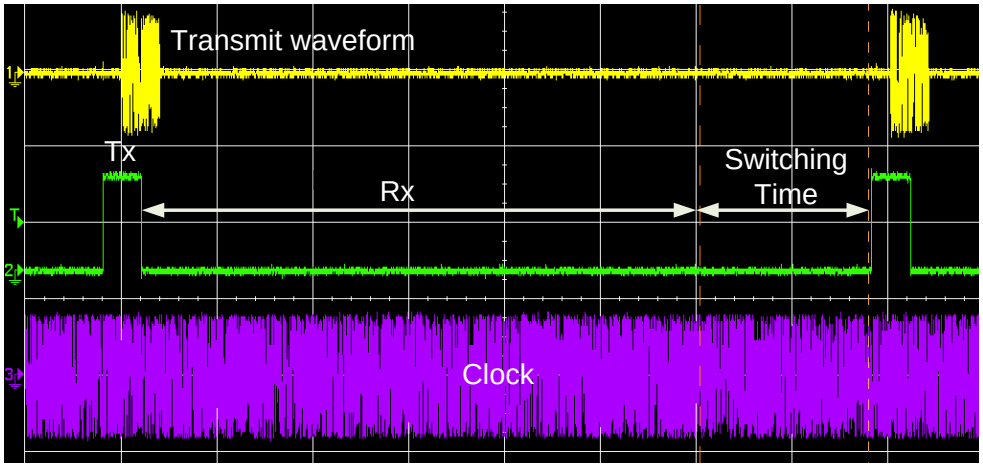

Figure 6.6. Timing state machine output trigger plot

The baud rate of the AFB UART controllers is set to 115 Kbps, and we can observe that the Host Computer is capable of sending 14720 commands (8-bit words) per second to the Formatter. The timer is also used to calculate the total time taken by the MicroBlaze processor to store the timing and sequence table information in its memory. It is observed that the MicroBlaze processor takes 30 ms to store the sequence table and timing information in its memory. This includes the transmission time through the serial cable.

A timing diagram showing the relative time periods for data transfer during the radar operation mode is shown in Figure 6.7. The polygons represent the relative time taken by the Formatter to perform the corresponding operations. The soft processor first executes the "Store Timing and Sequence Data" function to store multiple sets of azimuth beam positions in the memory and takes approximately 18 ms to store one beam position. The processor then executes the "Begin" function to switch azimuth

beam positions and also to pulse the radar by starting the TSM. The timing diagram as shown in Figure 6.7, clearly shows the different events taking place in the Formatter during the operation phase of the Phase-Tilt radar.

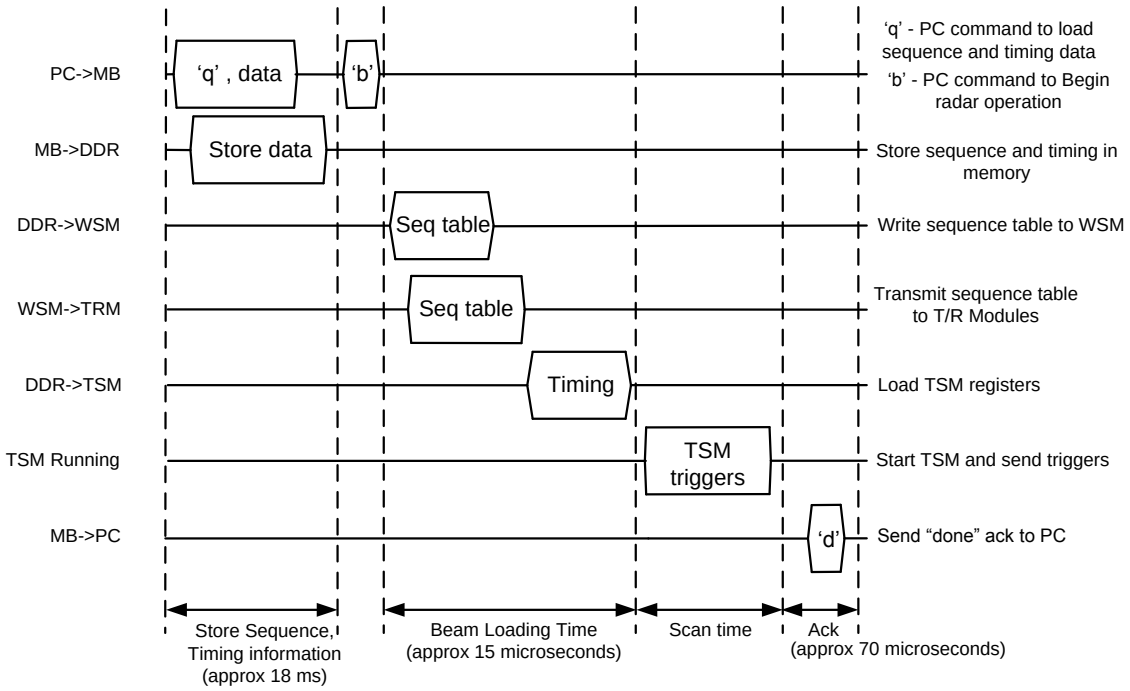

Figure 6.7. Diagram showing the relative time periods during the radar operation phase

CHAPTER 7

ETHERNET INTERFACE

CASA's long-term plan is to establish a network interface between the Host Computer and the Array Formatter in the Phase-Tilt radar system. An ethernet connection is preferable to employing long serial cables, since it allows communication to be faster, more reliable, and better suited where distances between the sub-systems are large. This chapter describes the ethernet interface between the two sub-systems in detail. A high-speed, full-duplex ethernet interface is implemented on the Formatter's FPGA. Xilinx Ethernet Lite Media Access Controller (MAC) [23] incorporates the appropriate features of the IEEE 802.3 standard [1]. The Ethernet lite core is interfaced to the Processor Local Bus of the MicroBlaze system.

The Ethernet core has a memory-mapped direct I/O interface to the 2K byte transmit and receive dual port memory (also called the Tx and Rx buffers). The Tx buffer holds the transmit data for one complete frame and the transmit interface control registers. Similarly, the Rx buffer holds the receive data for one complete frame and the receive interface control registers.

7.1 Lightweight IP (lwIP)

The lightweight IP (lwIP) [2] software is added in the software platform to implement the TCP/IP protocol stack. The focus of the lwIP stack is to reduce memory usage and code size, thus making it ideally suited for networking applications in embedded systems. Xilinx SDK provides the lwIP software that is customized to run on all MicroBlaze-based embedded systems. The lwIP applications can be developed

either using the raw API mode or the socket mode. The raw API mode provides a "callback" style interface to the application program. It is single threaded and registers callback to events like TCP read or write. Socket mode on the other hand, helps in the ease of programmability giving simple APIs to the application program developer, but requires support of the Xilinx kernal to handle all the functions. This makes the socket mode inherently slower than the raw API mode, and hence the raw API mode is used in this work to get better performance.

7.2 Test Ethernet Connection

In order to establish an ethernet connection between the host PC and the AFB, the FPGA in the AFB is configured as an echo server, which listens to the ethernet port for any data transfers. The standard lwIP software functions are used in this work to communicate with the PC over a TCP connection. The following steps are performed to establish and test the connection:

1. Designate IP address of the board and the PC. The PC IP address must be within the same subnetwork as the board IP address. In this work, the IP address of the board is set to 192.168.1.10 and the PC to 192.168.1.11.

2. The main function starts the user application which first creates the TCP Process Control Block (PCB) structure using the lwIP function pcb = tcp_new().

3. This structure binds to a local IP address and port number by calling the function tcp_bind(pcb, IP_ADDR_ANY, port). The argument IP_ADDR_ANY binds the connection to any local IP address.

4. The function pcb = tcp_listen(pcb) now sets up the port to listen to incoming connection.

5. The function tcp_accept(pcb, accept_callback) registers the accept_callback function to accept the incoming connection.

The above procedure creates a TCP connection between the board and the Host PC. This connection is then tested by pinging the board's IP from the Host PC.

The accept_callback function is called whenever a TCP connection is established. Since the FPGA is configured as an echo server, it responds on receiving data from the client (PC). A new communication protocol between the two is developed, which is described in detail in the next section.

7.3 Communication Protocol

Every command and the associated data to the Formatter is sent in an ethernet packet. The Formatter responds to the command and implements the different modes that are discussed in Chapter 6. The size of the Rx and Tx buffers of the ethernet core is limited to 2K bytes each. Hence, the maximum size of data that can be sent in one packet is 1500 bytes, and the rest is allocated for handling protocol information. The two Formatter modes that require heavy inflow of data from the Host PC are the Store Calibration Data and the Store Timing and Sequence Data modes. The implementation of these two modes with the ethernet interface are discussed in detailed in this section. The other modes are almost entirely the same as discussed in Chapter 6.

Whenever an ethernet packet is received by the Formatter, the first byte of the data field is ignored and the second byte is checked for the Formatter mode. This is because the Rx buffers are 32-bits wide, and the Formatter command is the fourth byte of the first data word. Succeeding data are then accessed as 32-bit words from the Rx buffers.

7.3.1 Store Calibration Data

As the maximum size of data in an ethernet packet is 1500 bytes, the store calibration command (which requires at least 512 4-byte words) is split into two-packet transmissions. The following are the steps to implement store calibration data mode:

1. If the second byte of the data field of the packet corresponds to 's,' the Formatter determines it as the first packet of this mode.

2. The next byte corresponds to the address of the T/R Module. The three bytes following the T/R Module address is ignored for the 32-bit data resolution in accessing the Rx buffers.

3. The calibration data is then accessed as 32-bit data in the Rx buffers and stored in the corresponding T/R Module DDR SDRAM address segment until it receives two different "End of commands."

4. The end of packet command, 0x7EE7, tells the Formatter that the remaining calibration data for the addressed T/R Module is contained in the second data packet. 0xFFFF, on the other hand, tells the Formatter that the store calibration data mode is done for the addressed T/R Module and there is no more data to store in the memory.

5. If the previous data packet contained 0x7EE7 as its end of command, the Formatter expects a character 'm' (more calibration data) as the command in the second packet to continue storing the calibration data for the addressed T/R Module.

6. The Formatter continues to access calibration data in the second packet as 32-bit words, and stores in its memory, until the end of command 0xFFFF is received.

The data fields in the two packets for this mode is shown in Figure 7.1

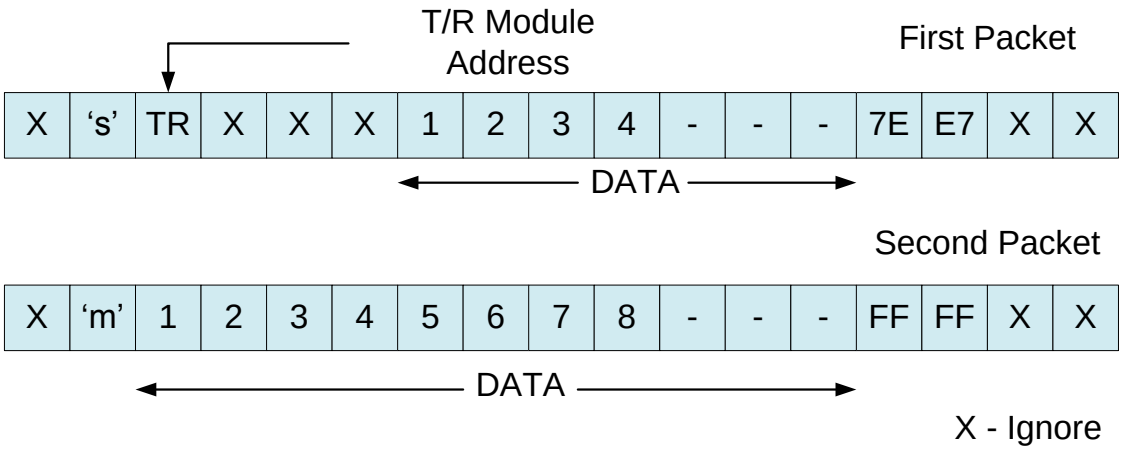

Figure 7.1. Ethernet packets for Store Calibration Mode

7.3.2 Store Timing and Sequence Data

This mode loads all the sequence and timing related information in Formatter's memory. A single packet transmission is used for loading multiple sets of sequence and timing data that constitute an azimuth scan. The protocol is described below:

1. If the second byte of data in the packet is 'q,' the Formatter implements this mode of operation.

2. The next two bytes correspond to the loop number of the TSM which is loaded in the Formatter's memory. The two bytes following the loop number is ignored.

3. The next 8 bytes correspond to timing and sequence table information and are read out from the buffer and stored in the memory.

4. Steps 2 and 3 are repeated until the loop number is read out as 0xFFFF, which is the "End of Command" for this mode.

The data fields in the packet for this mode is shown in Figure 7.2

The rest of the Formatter functions are implemented in a similar manner with each command requiring a separate packet transmission. All the functions have been

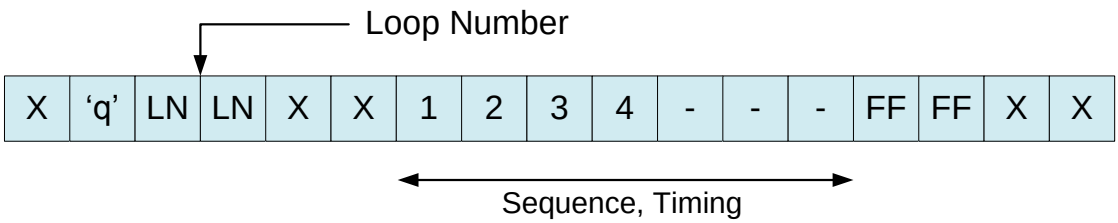

Figure 7.2. Ethernet packet for Store Timing and Sequence Data

implemented and verified on the Array Formatter Board. The communication protocol developed in this work can be used to completely replace the current serial link.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

A prototype Array Formatter for phased-array antenna systems has been designed and implemented for CASA. The requirements of the Formatter have been considered and a microprocessor-based system on an FPGA is designed and tested for functionality. The high-speed interfaces present in the Array Formatter Board are utilized for reliable and efficient radar control. A communication protocol between the Host Computer and the Formatter has been developed and tested. The Formatter along with the Host Computer software is capable of changing the azimuth beam positions of the Phase-Tilt radar in less than 90 μs. Successful integration of the Array Formatter with the array of T/R Modules has also been demonstrated in this work. Finally, the ethernet interface between the Host Computer and the Array Formatter has also been implemented. This serves as an ideal platform for CASA's goal to replace the current serial link with a faster and more reliable ethernet interface.

8.2 Future Work

1. Implementation of the Tilt Controller is necessary to change the elevation angle of the radar.

2. The Host Computer software must be capable of communicating with the Formatter over the ethernet. Once this is done, further testing between the two systems is necessary.

APPENDIX A

MICROBLAZE SOFTWARE APPLICATION USER GUIDE

This chapter serves as a reference manual for the MicroBlaze software application. In this application, the software APIs are used to facilitate communication between the MicroBlaze processor and the peripherals such as DDR SDRAM, Custom Peripheral (PF), and UART. The next few sections describes these APIs in detail.

A.1 UART

The UART core is used for implementing serial communication with the Host Computer. All the APIs are defined in the header file "xuartlite.h" which is added in this application. The functions, Xuartlite_SendByte(Base_Address, Data) and Xuartlite_RecvByte(Base_Address), are used for for sending and receiving a byte of data using the UART core. The processor recognizes the argument Base_Adress as the memory-mapped base address of the UART. Alternatively, the xil_printf function can also be used for sending a stream of data bytes to the computer. The UART core is the default "STDIO" for the processor system, and hence, this function can be used for sending data to the Host Computer.

A.2 DDR SDRAM Controller

The DDR SDRAM memory controller is used by the MicroBlaze processor for storing the calibration data, sequence, and timing information. The user can instruct the MicroBlaze processor to write a 32-bit data to a specific address location in the

DDR SDRAM by calling the function XIo_Out32(Address, Data). Similarly, for a read operation, the XIo_In32(Address) function is used to read out a 32-bit data from the Address argument. The two APIs are defined in the header file "xio.h," which is added to the software application. The processor recognizes the address space of the DDR SDRAM between its base and high addresses. The processor accesses the DDR SDRAM controller as a memory-mapped I/O, and any memory read or write operation is specified within this address range.

A.3 Custom Peripheral (PF)

The PF acts as a gateway for the processor to communicate with the rest of the custom hardware Verilog modules of the system. The PF APIs are contained in the header file "pf.h." This file is included in the software application. The PF has access to thirty 32-bit software registers, which are mapped to the various ports of the hardware modules. The modules that communicate with the soft processor via the PF are the T/R Module interface (Read and Write State Machines), the TSM, and the Transceiver Interface.

Table A.1 lists the software registers mapped to the different ports of the hardware modules.

A.3.1 T/R Module Interface Control

The PF communicates with the T/R Module interface primarily to write data into the WSM and read data from the RSM. It is also used to enable the channel receive pin of the the T/R Module for receiving the echoed data from the addressed Module during the radar initialization phase. As explained in section 5.2.3.1, the WSM writes the data into the Transmit FIFO. Data from the Transmit FIFO is then shifted out serially by the Transmitter, and finally transmitted as LVDS signals to the T/R Modules over the GPIO pins on the board. Likewise, the Receiver, which

Table A.1. PF software registers mapped to verilog module ports

Register Number	**Hardware Port**	**Hardware Logic**
0	Reset	Transmit, Receive FIFOs
1	Software Read Enable	RSM
2	Software Write Enable	WSM
3	Data In	WSM
4	Data Out	RSM
5	Write Ack	WSM
6	Read Ack	RSM
7	FIFO Empty	Receive FIFO
8	Loop Number	TSM
9	SW Channel Enable	Channel Enable
10	Timing Enable	TSM
11	Timing Reg 1	TSM
12	Timing Reg 2	TSM
13	Timing Reg 3	TSM
14	Timing Reg 4	TSM
15	Clock 2	TSM
16	Timing Ack	TSM
17	FIFO Empty	Transmit FIFO
18	T/R Module Address	Channel Enable
19	Software Enable	Transceiver Interface
20	Data In	Transceiver Interface
21	Transceiver Ack	Transceiver Interface

gets the serial data from the T/R Modules, converts it into a 16-bit data, and later writes the data into the Receive FIFO. The RSM reads the data from the Receive FIFO, and the PF gets the data from the RSM.

The PF software registers are mapped to the following ports of the T/R Module interface:

1. WSM enable
2. RSM enable
3. Input Data port of the WSM
4. Output Data port of the RSM
5. Address input pin of the Channel Enable Module
6. Software channel enable

The C functions below write a 32-bit data into the WSM:

Program 1 Write data to T/R Module interface

```
writef1(Xuint32 data32)
{
PF_mWriteSlaveReg3(XPAR_PF_0_BASEADDR, 0, data32);
//write data
PF_mWriteSlaveReg2(XPAR_PF_0_BASEADDR, 0, 1);
//sw write enable
PF_mWriteSlaveReg2(XPAR_PF_0_BASEADDR, 0, 0);
//sw write disable
}
```

The MicroBlaze processor first writes a 32-bit unsigned integer into the "Data In" port of the WSM, and later enables the WSM to latch the data into the Transmit FIFO as two 16-bit words. The processor disables the WSM by writing a zero to software register 2. The argument XPAR_PF_0_BASEADDR represents the base

address of PF, which is recognized by the processor as a memory-mapped I/O. All these peripheral paraments are listed in the header file "xparameters.h." This file is included in the application. The second argument "0" represents the register offset from the referenced register number. All the registers are directly referenced by their numbers and the offset argument is always zero.

The C functions below read a 16-bit value from the RSM:

Program 2 Read data from T/R Module interface

```
Xuint32 readf2()
{
PF_mWriteSlaveReg1(XPAR_PF_0_BASEADDR, 0, 1);
//sw read enable
Xuint32 data32 = PF_mReadSlaveReg4(XPAR_PF_0_BASEADDR, 0);
//read data from sw reg 4
PF_mWriteSlaveReg1(XPAR_PF_0_BASEADDR, 0, 0);
//sw read disable
}
```

The processor first enables the RSM, and then reads the data out from the "Data Out" port of the RSM, which is later disabled.

In order to receive data from the T/R Modules, the channel enable module is controlled in software by PF. The address is passed as a parameter to the channel enable module. This hardware logic receives the address from its "Address In" port and enables the left or right T/R Module channel to receive the echoed data during the Unicast scheme of operation. PF register 9 is used by the processor enable this module, and register 18 is used for passing the T/R Module address.

A.3.2 Timing State Machine Control

The MicroBlaze processor starts the TSM by controlling the appropriate PF software registers. This section describes the APIs used to load the timing registers and also enable or disable the TSM.

When the processor receives the 'b' command from the computer, it executes the "Begin" software function. In this function, the timing information (four 32-bit words) and the loop number (one 32-bit word) are read out of the memory using the XIo_In function. Each timing register is a 32-bit word, the first 16-bit word represents the transmit time of the pulse, and the second 16-bit word represents the receive time. As TSM contains four such timing registers, one pass through the TSM, generates four trigger pulses for the operation of the Phase-Tilt radar. The loop number variable repeats the TSM "Loop Number" times, and thus, each Phase-Tilt azimuth beam position generates 4×"Loop Number" of pulses.

Program 3 Update timing registers and enable TSM

```
PF_mWriteSlaveReg8(XPAR_PF_0_BASEADDR, 0, Loop_Number);
// load Loop Number variable
PF_mWriteSlaveReg11(XPAR_PF_0_BASEADDR, 0, timing_data1);
PF_mWriteSlaveReg12(XPAR_PF_0_BASEADDR, 0, timing_data2);
PF_mWriteSlaveReg13(XPAR_PF_0_BASEADDR, 0, timing_data3);
PF_mWriteSlaveReg14(XPAR_PF_0_BASEADDR, 0, timing_data4);
// load 4 timing regs
PF_mWriteSlaveReg10(XPAR_PF_0_BASEADDR, 0, 1);
// sw timing state machine enable
```

It is important to note that, before enabling the TSM, the processor also reads the sequence table information out of the memory, and writes them one-by-one to the WSM of the T/R Module interface. Section A.3.1 explains how data is written to the T/R Module interface. The sequence table information contains control bits that instruct all the T/R Modules to listen to their data bus. All the T/R Modules update their sequence table with the broadcasted information.

The processor then waits until Reg 16, which corresponds to the "Timing Ack" port of the TSM, is set to one. When this register is set to one, the processor is notified that the TSM has repeated "Loop Number" times for the current beam position, and it is time to load the next set of timing registers from the memory for the second azimuth beam position. The processor then sends a character 'd' (or done) to the

Host Computer by writing the character to the UART receiver. On receiving this command, the computer either sends another 'b,' to start the next beam position, or sends an 'e' to notify the processor to end the azimuth scan.

Program 4 Disable timing state machine

```
j = PF_mReadSlaveReg16(XPAR_PF_0_BASEADDR, 0);
// read timing ack
while (j!=1)
{
  j = PF_mReadSlaveReg16(XPAR_PF_0_BASEADDR, 0);
  // read timing ack
}
PF_mWriteSlaveReg10(XPAR_PF_0_BASEADDR, 0, 0);
// sw timing state machine disable
```

The processor sends the " done" message to the computer at the completion of one beam position, because the computer keeps track of the polarization pulsing sequences being performed by the radar. This helps in synchronization of the sequences with the processed data from the Transceiver. This acknowledgment by the processor is necessary if the Phase-Tilt radar employs a Pentek Transceiver [4]. The processor also conveys the scan information (beam position and polarization sequence) to the Transceiver interface. This is useful if the RVP900 Transceiver [18] is used, which is capable of synchronizing the scan information with the raw I/Q data samples that it receives from the antenna panel. Details of the implementation of this function is discussed in the next section. To add more flexibility to the design, while executing the "Begin" function, the processor sends the scan acknowledgment to the computer and also conveys the scan information to the Transceiver interface.

A.3.3 Transceiver Interface Control

The Transceiver interface is implemented in hardware to convey the scan information to the RVP900 Transceiver. This interface is also mapped to the software

registers of PF. During the "Begin" function, the processor first creates the scan information, (by extracting the beam position and polarization bits from the sequence table information). The processor then enables the Transceiver interface by setting Reg 19 to one. The scan information is then written out as a 32-bit word to PF Reg 20. This register is mapped to the "Data In" port of the Transmitter module in the Transceiver interface. The scan data is shifted out serially by the Transmitter via the GPIO pins on the board at a rate of 1 Mbps. The Interface is finally disabled in software by writing the value zero to PF Reg 19.

A.4 Integration Test

A sample experiment was conducted to check the functionality of the Array Formatter software application. The board was integrated with the Host Computer software as well as the T/R Modules to verify the different modes of operation. In order to test the "Store Calibration Data" function, a file that contains the T/R Module calibration data is uploaded by the computer. The file's data format is shown in Figure A.1.

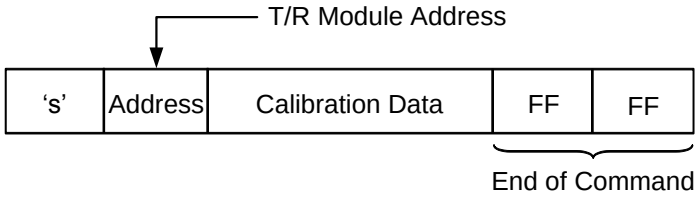

Figure A.1. Host Computer data format for storing calibration data in Formatter

The Formatter first receives the 's' character from the computer and enters the Store Calibration Data function. The next byte from the UART receiver is accessed by the processor, which corresponds to the T/R Module address, and the corresponding DDR memory segment address is calculated. The processor then receives sets of 4

bytes from the UART receiver. The processor stores the data as 32-bit words in the memory, until the end of command "0xFFFF" is received.

The computer sends multiple files for all the T/R Modules until all the 64 memory segments of the Formatter's DDR SDRAM is loaded with calibration data. The next step is to load the T/R Module look-up table memory with this data. The Host Computer sends the 'f' command followed by a byte of T/R Module address. The Formatter reads the data out from the DDR SDRAM memory segment, and transmits the data to the addressed T/R Module. The T/R Module updates its look-up table memory with the calibration data, which are then echoed back to the Formatter for error check. This process is repeated until all the T/R Modules are loaded with calibration data.

The computer then uploads a file to the Formatter, that contains multiple sets of timing and sequence table information for a complete azimuth scan. The data format for this file is shown in Figure A.2. The Formatter first receives the 'q' character and enters the "Store Timing and Sequence Data" function. The next two bytes are received as the Loop Count variable for the first beam position's TSM. The next 32 bytes are received as four 32-bit timing registers for the TSM, and four 32-bit words for the sequence table. The processor stores these information in the memory until the end of command "0xFFFF" is received. In this way, multiple beam positions are stored in the memory for an azimuth scan.

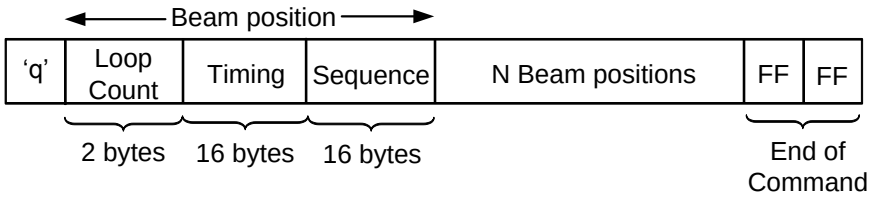

Figure A.2. Host Computer data format for storing sequence table and timing information

The next step is to execute the Formatter's "Begin" function. The Host Computer sends the character 'b' to the Formatter, which instructs the MicroBlaze processor to execute the "Begin" function. The timing information of the first beam position is read out from the DDR SDRAM memory and the TSM registers are updated by the Formatter. The beam position along with the polarization sequence information from the sequence table constitute the scan information. This information is written to the Transceiver Interface. The sequence table is then transmitted to the T/R Modules. The Formatter then starts the TSM to send the trigger pulses. Once the TSM Loop Count variable decrements to zero, the MicroBlaze processor sends the 'd' character to the computer to acknowledge the completion of the first beam position. If the computer sends another 'b,' the next beam position is fetched from the memory, and the same process repeats. In this way, multiple beam positions are read out from the memory, and pulsed by the Formatter TSM, until the 'e' character is received from the computer. The computer sends the 'e' character to mark the end of azimuth scan.

This test was conducted in the laboratory and the waveforms as shown in Figure 6.6 in Chapter 6 were recorded on the digital oscilloscope by probing the GPIO pins of the AFB. This test proves the successful integration of the Array Formatter Board with the Host Computer software and the array of T/R Modules. A picture of the laboratory setup used for testing the flow of commands from the Host Computer to the AFB to the T/R Modules is shown in Figure A.3.

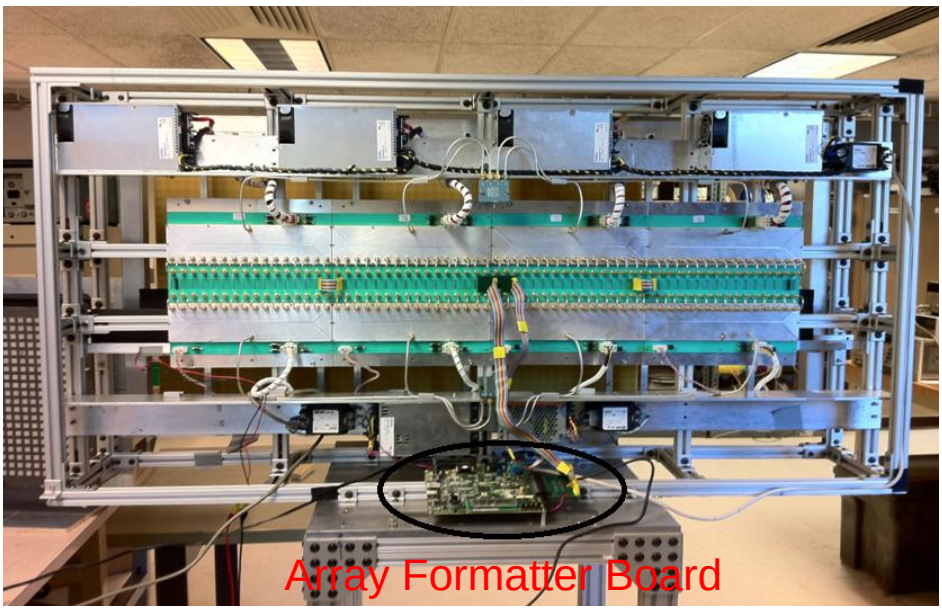

Figure A.3. Laboratory setup to test AFB communication

APPENDIX B

ARRAY FORMATTER CUSTOM VERILOG MODULES

This chapter shows the simulation results and describes the functionality of the custom Verilog modules implemented in the Formatter FPGA. All simulations are carried out using Modelsim [3].

B.1 T/R Module Interface Modules

The custom Verilog modules in this interface are used for communicating with the T/R Modules. This section discusses the implementation and verification of the serial Transmitter and Receiver in the T/R Module interface.

B.1.1 Serial Transmitter

The serial Transmitter module conveys the serial data to the T/R Modules. The WSM writes the 16-bit data to the Transmit FIFO (a Xilinx IP core), and the Transmitter sends the data out serially to the T/R Modules. This module monitors the Empty signal of the Transmit FIFO. When this signal goes low to indicate that the FIFO is not empty, the Transmitter module reads the FIFO data by asserting the FIFO "Read_enable signal" and copies the data into its internal shift register, and then enters the "Start" state. At this state, the Transmitter sends out a low signal to transmit the start bit. The module then enters the "Data" state, where each data bit is accessed from the shift register, and sent out serially through the "Tx" port. At the end of 16 clock cycles, the module enters the "Stop" state, where it transmits a high signal, to indicate the stop bit. The module then asserts the "Tx_done" signal.

As shown in Figure B.1, the data transmitted is 0xA55A, with the start bit being zero and the stop bit being one. This serial data is then sent out as LVDS signals by the Xilinx OBUFDS primitive.

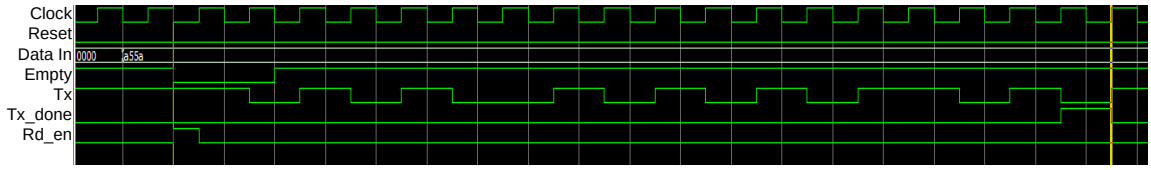

Figure B.1. T/R Module transmitter waveform

B.1.2 Serial Receiver

The serial receiver is a serial to parallel data converter. The module polls its input "Rx" port, and when it goes low, the module enters the "Data" state. In this state, the module receives the incoming data signals from the "Rx" port every cycle, and copies and shifts the data into its internal shift register. This value is copied into its "Data_out" port and appears as a 16-bit data. At the end of 16 cycles, the module asserts the "Rx_done" signal and enters the stop state, where it receives the stop bit. In Figure B.2, the data received is 0x8CC7, which is presented at the Data_out port in the end. The clock synchronizer which monitors the Receover's "Rx_done" signal, copies this data into the Receive FIFO by enabling the write port of the FIFO. The synchronizer is used because the Receiver operates at 25 MHz, whereas the FIFO operates at 50 MHz.

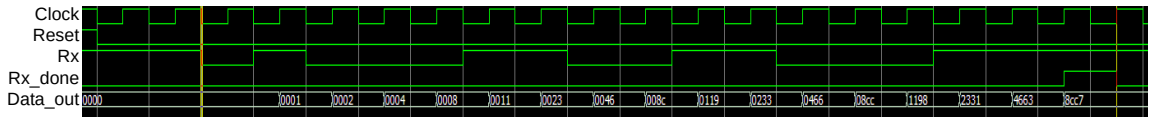

Figure B.2. T/R Module receiver waveform

B.2 Timing State Machine Module

The TSM is a 32-state FSM, which sends trigger pulses through two of its output ports for the Transceiver and the T/R Modules. The loop count variable, four timing registers, and software enable are given as inputs by the MircroBlaze processor's custom peripheral (PF). The TSM is tested and verified for functionality by probing the T/R Module and Transceiver triggers from the GPIO pins on-board. The waveforms are obtained in the oscilloscope and shown in Figure B.3. As seen in Figure B.3, the TSM output to the Transceiver is level sensitive. This is targeted for the Pentek Transceiver, which requires level sensitive triggers. The TSM outputs can be modified to derive edge sensitive Transceiver triggers.

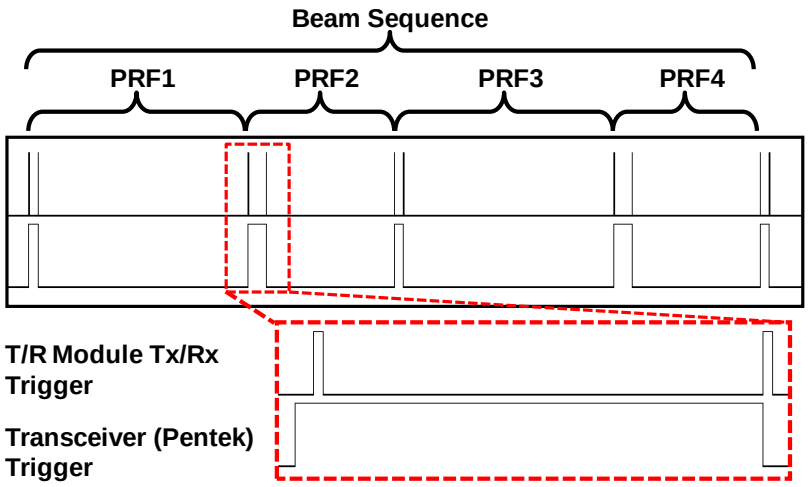

Figure B.3. Timing state machine waveform

B.3 Transceiver Interface Modules

The RVP900 Transceiver requires clock, enable signal, and a 32-bit scan data to synchronize the raw I/Q data samples with the scan information. The Transceiver requires the enable signal from the Formatter to latch the incoming data into its internal registers. The Formatter's Transceiver interface transmits the scan data serially over the GPIO pins on-board. The Transmitter module in this interface

enters the "Data" state when the software enable signal from PF register 19 is set to one. In this state, the out enable signal "v_en" is set to one, and the scan data from "Data_In" port is shifted out serially by the Transmitter module. At the end of 32 cycles, the module enters the "stop" state, where it de-asserts the "v_en" signal. Figure B.4 shows the Modelsim simulations for this module.

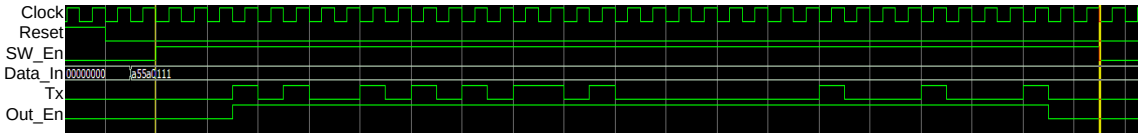

Figure B.4. Transceiver interface transmitter waveform

In order to operate this interface at 1 MHz, a clock divider function is implemented. This module takes in the external 100 MHz clock and generates the 1 MHz clock rate. Figure B.5 shows the output waveforms from this module.

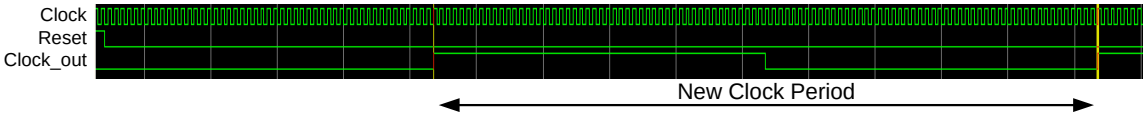

Figure B.5. Clock divider waveform

APPENDIX C

ARRAY FORMATTER BOARD USER MANUAL

This chapter serves as a manual for users operating the Array Formatter Board.

C.1 Loading the FPGA PROM

To load the PROM with a new FPGA configuration file, the first step is to create the PROM file from the generated "bit" file using Xilinx Impact Tool. This tool is invoked from the design pane in the Xilinx ISE project. The Xilinx PROM device is selected with a capacity of 4 MB in Impact. The PROM file is generated and the scan chain is initialized. The Xilinx PROM device is selected in the scan chain, and the generated PROM file is loaded into the PROM. For further details refer [25].

C.2 Loading the FLASH

The software application for MicroBlaze is loaded in the Parallel FLASH. This task is accomplished by Xilinx EDK (XPS or SDK). In the Xilinx XPS tool, the "Program FLASH" command in the GUI is used for loading the FLASH with an "ELF" file. The bootloader application is also created in this process by checking the "Create Bootloader" option. The entire process takes a few minutes to load the FLASH with the ELF file and also to create a bootloader for this application. In XPS, the bootloader application is "Marked to Initialize BRAMs." This loads the BRAM with the bootloader application when the FPGA gets configured from the PROM on power up. For further details refer [22].

C.3 AFB Operation sequence

In order to operate the AFB, the user has to go through a sequence of steps which is described below:

1. Turn on the supply to the AFB.

2. Turn on the supply to the external 100 MHz clock.

3. The Formatter goes through the bootlaoding process, and after this process, the computer can send serial commands to implement the different Formatter modes.

4. Once the Formatter software application exits, the supply to the clock is first turned OFF, before powering OFF the board.

BIBLIOGRAPHY

[1] IEEE 802.3 ethernet working group. `http://www.ieee802.org/3/`.

[2] lwip - a lightweight tcp/ip stack - summary. `http://www.lwip.org/`.

[3] Modelsim - advanced simulation and debugging. `http://www.model.com/`.

[4] Pentek, inc. `http://www.pentek.com/`.

[5] Phased-array radars. `http://www.noaa.gov/`.

[6] Doviak, Richard J., and Zrnic, Dusan S. *Doppler Radar and Weather Observations*. Dover Publications, Inc., 1993.

[7] Fenn, A.J., Temme, D.H., Delaney, W.P., and Courtney, W.E. Development of phased-array radar technology. Linoln Laboratory, Massachusetts Institute of Technology Journal, 2000.

[8] Hopf, A.P., Salazar, J.L., Medina, R., Venkatesh, V., Knapp, E.J., Frasier, S.J., and McLaughlin, D.J. Casa phased array radar system description, simulation and products. In *IGARSS 2009: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium* (Cape Town, South Africa, July 2009), vol. 2, pp. II–968 – II–971.

[9] Kuon, I., Tessier, R., and Rose, J. Fpga architecture: Survey and challenges. In *Foundations and Trends in Electronic Design Automation* (135-253), vol. 2, pp. 135–253.

[10] Marsili, Nathan. Dual-polarization radar. Dupage County Advanced Spotter Program, 2010.

[11] Medina, R.H., Knapp, E. J., Salazar, J. L., A.P., Hopf, and McLaughlin, D.J. T/R module for casa phase-tilt radar antenna array. In *Proceedings of the IEEE International Symposium on Phased Array Systems and Technology* (Waltham, MA, USA, October 2010).

[12] Micron Technology. *DDR SDRAM Datasheet*, 2000. http://www.micron.com.

[13] Palnitkar, Samir. *Verilog HDL: A Guide to Digital Design and Synthesis*. Prentice Hall, 1996.

[14] Salazar, J.L., Knapp, E.J., and McLaughlin, D.J. Dual-polarization performance of the phase-tilt antenna array in a casa dense network radar. In *IGARSS 2010: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium* (Honolulu, HI, USA, July 2010), pp. 3470–3473.

[15] Salazar, J.L., Medina, R., Knapp, E.J., and McLaughlin, D.J. Phase-tilt array antenna design for dense distributed radar networks for weather sensing. In *IGARSS 2008: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium* (Boston, MA, USA, July 2008), vol. 5, pp. V–318 – V–321.

[16] Seguin, Emmanuel. Low cost fpga based digital beamforming architecture for CASA weather radar applications. Master's thesis, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, May 2010.

[17] Skolnik, Merill. *Introduction to Radar Systems*. Tata McGraw-Hill, 2001.

[18] Vaisala. *Vaisala Sigmet Digital Receiver and Signal Processor RVP900 Datasheet*, 2009. http://www.vaisala.org.

[19] Xilinx Corporation. *CORE Generator User Guide*, April 2000. http://www.xilinx.com.

[20] Xilinx Corporation. *MicroBlaze Processor Reference Guide*, August 2004. http://www.xilinx.com.

[21] Xilinx Corporation. *MicroBlaze Platform Flash/PROM Boot Loader and User Data Storage*, June 2005. http://www.xilinx.com.

[22] Xilinx Corporation. *EDK Reference Guide*, December 2009. http://www.xilinx.com.

[23] Xilinx Corporation. *LogiCORE IP XPS Ethernet Lite Media Access Controller*, September 2010. http://www.xilinx.com.

[24] Xilinx Corporation. *Spartan-3E FPGAs Data Sheet*, January 2011. http://www.xilinx.com.

[25] Xilinx Corporation. *Spartan-3E Starter Kit User Guide*, January 2011. http://www.xilinx.com.