2-9-2010

FPGA controlled reconfigurable antenna

Severn Shelley

Severn Robert Shelley
Candidate

Electrical and Computer Engineering
Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

[illegible], Chairperson

[illegible]

[illegible]

FPGA Controlled Reconfigurable Antenna

by

Severn Robert Shelley

B.S., Electrical Engineering, Brigham Young University, 2008

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2009

Dedication

to my

MOTHER and FATHER

with love

Acknowledgments

My deepest appreciation is to God for teaching me all things.

I am grateful for my parents who have loved, supported, and encouraged me.

I also thank my academic advisor Christos Christodoulou, who has motivated me and inspired me with his enthusiasm for the science of electromagnetics.

I am further thankful for my fellow students Joseph Costantine, Youssef Tawk, Manuel Rivera, Naga Devarapalli, Teofilo De la Mata Luque, Cassandra Mendonca, Maria Elizabeth Zamudio, Mohammed Husseini, Shawn Soh, Chris Leach, Rich Compeau, and many others who have personally tutored and helped me.

I am additionally grateful to my relatives Summer, Desta, Thomas, Roberta, Kenneth, Valerie, Robert, Bernice, Daniel, and many others for their care and encouragement.

Finally, I also thank The University of New Mexico Electrical and Computer Engineering Department professors and staff for all their hard work and dedication, providing me the means to complete my degree and prepare for a career as an Electrical Engineer.

NOTE: This thesis was submitted to my Supervising Committee November 10, 2009.

FPGA Controlled Reconfigurable Antenna

by

Severn Robert Shelley

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2009

FPGA Controlled Reconfigurable Antenna

by

Severn Robert Shelley

B.S., Electrical Engineering, Brigham Young University, 2008

M.S., Electrical Engineering, University of New Mexico, 2009

Abstract

At the present time, the advantages of reconfigurable antennas are numerous but limited by the method of controlling their configuration. This thesis proposes to utilize the advantages of Field Programmable Gate Arrays (FPGAs) to overcome this dilemma. Two experimental antennas are designed. The first reconfigurable antenna consists of two patches connected by two diodes. The second reconfigurable antenna has sixteen possible combinations and is designed with four perimeter patches also connected via diodes. The electromagnetic modelling software HFSS is utilized to predict the resulting radiation patterns and resonances of the possible configurations. A computer program is created to interface a user with the FPGA controlling the antenna. A module for receiving instructions and asserting biasing signals is programmed onto the FPGA. Finally, a prototype antenna is fabricated using a mechanical etching machine. Experimental results are examined using a network analyzer. The FPGA system is connected to the reconfigurable antenna. Both experimental and theoretical results show that configurable tuning is achieved.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

What is an FPGA? What are reconfigurable antennas? How can these two work together to make better antennas? These questions will be answered in this thesis, but first some background will be established. Reconfigurable antennas have a fascinating recent history and the principles that govern them have a much richer longer history. Microstrip antennas have been in use since the late 1960's and early 1970's but the concept of reconfigurable microstrip antennas has only recently been given greater attention [1]. Reconfigurability in an antenna system is a much-desired characteristic. It has been the focus of several research groups in the past decade because of renewed interest due to the rapid expansion of wireless technologies [2].

1.1 Motivation

It has become increasingly important to maximize the amount of information that can be transmitted wirelessly in the past decade. Because the frequency spectrum is limited as seen in Figure 1.1, it is important to develop technologies that can make the most use out of the available bandwidth. Reconfigurable antennas have become a technology that solves this problem because they can be instructed to automatically tune to unused frequency bands in the frequency spectrum. Additionally, despite being low-powered, inefficient, and having a narrow bandwidth, patch antennas can be used for a variety of purposes and can function over a number of frequency bands. Because of this ability, they are important in making the frequency spectrum more open and accessible, and are a good choice for use in a reconfigurable design.

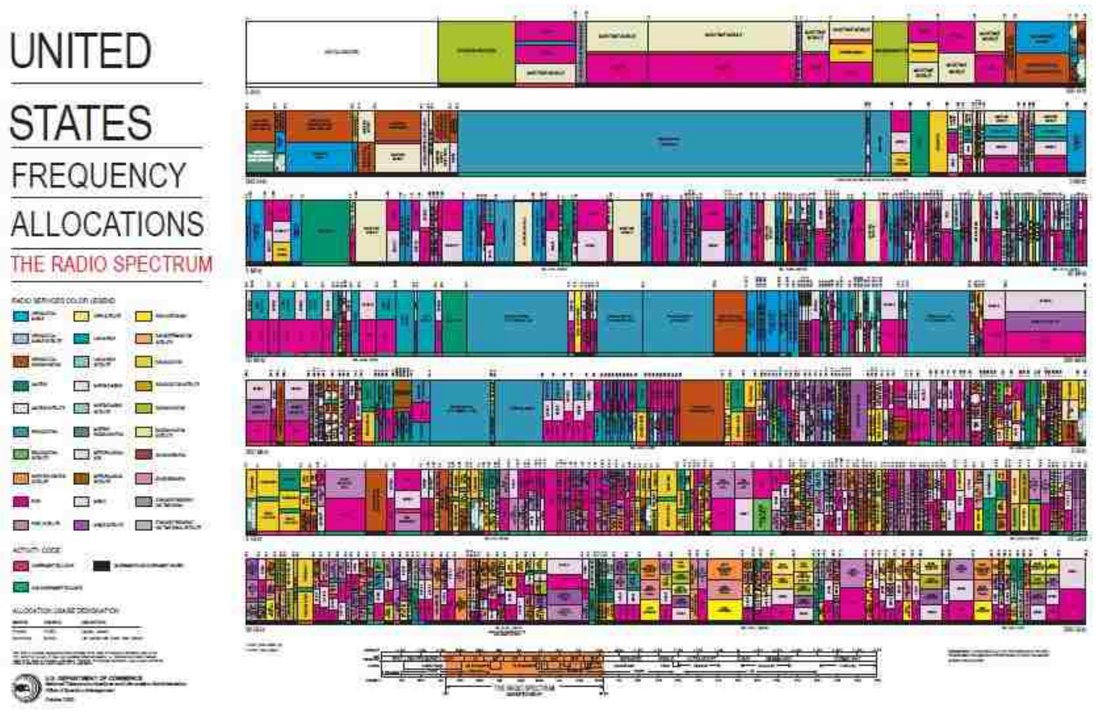

Figure 1.1: The United States Frequency Allocation Chart [3].

According to one author [2], a characteristic example of the application of reconfigurable antennas is the variety of frequencies used in everyday life by personal communication devices across different continents. For example, in Europe cellular phones are already being used to pay parking tickets, while in Asia they are being used to pay grocery bills, to reserve airplane tickets, and to remotely control PCs.

Since reconfigurable antennas are so versatile, they are the perfect answer to easily accessing these different frequencies in both commercial and government sectors.

Satellite, missile, and aerospace systems would benefit greatly due to the requirements for an antenna that is small and lightweight [4]. Personal communicators would also profit from additional reconfigurable antenna technology. Three present day examples of reconfigurable antennas are shown in Figure 1.2. Researchers continuously strive to integrate more services into a single device and the demand for reconfigurability increases.

(a) Smart Phone [5]

(b) Intel Metro Laptop [6]

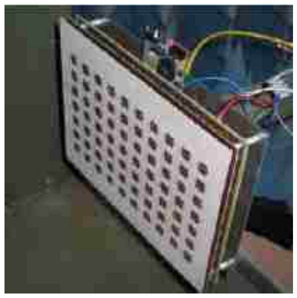
(c) Reconfigurable Antenna Array [7]

Figure 1.2: Three presently available reconfigurable antennas.

Chapter 2

Theoretical Framework and Characterization

In order to further develop the reconfigurable antenna, it is proposed that an FPGA be programmed to accept input from a computer and assert biasing signals to some of its output pins. In this chapter, a TAP controller which can accomplish this task is studied. Next, reconfigurable antennas are studied to outline how the biasing signals from the FPGA can configure the antennas.

2.1 Field Programmable Gate Arrays

The configuration of the antenna is accomplished by utilizing a Field Programmable Gate Array (FPGA). FPGAs are constructed using one basic “logic-cell”, duplicated thousands of times [8]. A logic-cell is simply a small lookup table (LUT), a D flip flop, and a two to one multiplexer for bypassing the flip flop.

The LUT is just a small Random Access Memory (RAM) cell and usually has four inputs, so it can in effect become any logic gate with up to four inputs. For example, an AND gate with three inputs, whose result is then sent through an OR gate with some other input would be able to fit into one LUT.

Every logic cell can be connected to other logic cells using interconnect resources. Interconnect resources are wires and multiplexers that are placed around the logic cells. While only one logic cell cannot accomplish much, lots of them connected together can accomplish complex logic functions. The FPGA's interconnect wires extend to the boundary of the device where Input Output cells are used to connect to the pins of the FPGA.

Besides the general purpose interconnect resources, FPGAs contain fast dedicated lines connecting neighboring logic cells. A technology somewhat unique to FPGAs, programmable technologies such as PAL and CPLD do not contain fast dedicated lines. These lines were implemented order to create arithmetic functions like counters and adders efficiently.

2.1.1 JTAG 1149.1 Standard

Voltages are asserted on four output lines from the FPGA by way of the Joint Test Access Group 1149.1 standard [9]. This standard began development in 1985 as part of the Joint European Test Action Group. The group collaborated with North America, became JTAG, and submitted a series of proposals for a standard form of boundary scan to the IEEE. The IEEE initially approved the standard in February 1990.

In this thesis, the standard is used to assert signals, but the standard's main function is the boundary scan. Without using physical test probes, boundary scan methodology allows one to test circuitry, interconnects, and cells of logic. By creating test cells which are then joined to every pin on the device, boundary scan can assert signals on specific pins by selection. The test cells are toggled using the JTAG serial scan chain line Test Data In (TDI). As outputs change on other pins, the test cell at that location can be read as Test Data Out (TDO). Thus, it is possible to verify proper circuit function. For example, if the circuit is shorted to another pin, the signal will not make it to the proper output pin and the short will be recognized. However, while using this technique on integrated circuits, test cells must be inserted around logic blocks in order to isolate them as separate circuits from the rest of the device.

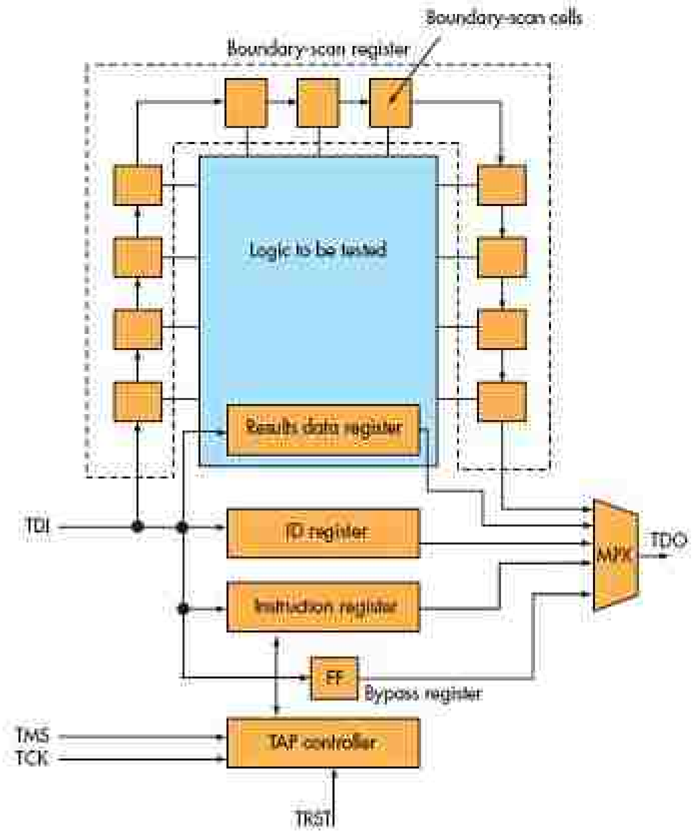

Figure 2.1: The boundary scan schematic.

A major advantage of using JTAG is that it will not interfere with circuit function when not in testing mode. While in testing mode however, specific test conditions can be chosen. This ability is taken advantage of in this thesis. In fact, Xilinx implements a proprietary version of a JTAG Test Access Port controller on their FPGAs in order to program the device. A serial scan chain is sent to the TAP controller and the logic cells and interconnects on the device are programmed using this information.

2.2 TAP Controller

A TAP controller module is programmed in VHSIC Hardware Description Language (VHDL) onto the FPGA. The state machine for the TAP controller is shown in Figure 2.2.

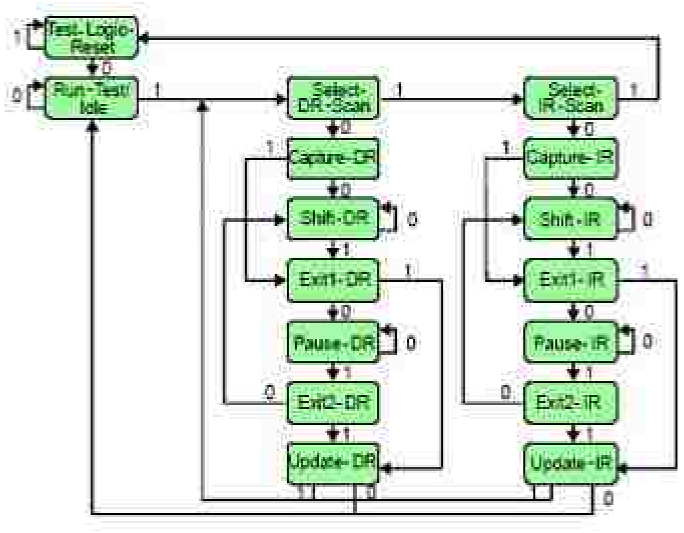

Figure 2.2: Test Access Port state machine.

The Test Access Port controller consists of five single-bit connections as can be seen in Table 2.1:

Boundary scan uses a couple methods to make sure that the TAP controller is

Table 2.1: Test Access Port Connections

Abbreviation	Name	IO	Description
TCK	Test Clock	Input	provides the clock for testing
TMS	Test Mode Select	Input	used to select testing modes
TDI	Test Data In	Input	line for sending data into chip
TDO	test Data Out	Output	line for reading data out of the chip
TRST*	Test Reset Signal	Input	used to reset the TAP controller

secure [10]. In order to prevent the boundary scan from running and allow the chip to run as designed, TRST* and TCK are held low and TMS is held high. Another facet of the controller is to sample the inputs on the rising edge of the TCK clock and ensure that the outputs are produced on the falling edge of the clock so that race conditions are avoided.

A general boundary scan testing architecture is shown in Figure 2.3. Several items of hardware make it up:

- The five lines that are part of the TAP interface
- For obtaining data from the device under test there are data registers (DRs)
- The instruction register (IR) dictates what the TAP controller should do
- The finite state machine or TAP controller, which is in charge of the inputting signal to the IR and DR

This finite state machine known as the TAP controller configures the system in two different modes. In one of its modes, the controller inserts an instruction into the instruction register to specify what boundary scan should do. In the other mode, it inserts data into and out of the data register. The IEEE standard demands the provision of at least two test data registers. These registers are known as the boundary scan register and the bypass register. The most important register for testing is the boundary scan register because it represents each input and each output on the device. This is behind the main idea that by controlling the device's IOs, boundary scan can find faults and shorts in the chip. The other register is known as the bypass register and is simply a single flip flop. It is also important since it is used to allow the TAP controller to skip the testing of idle devices and focus on testing one particular device. While not necessary, designers can also include configuration registers to add more functionality to a TAP controller.

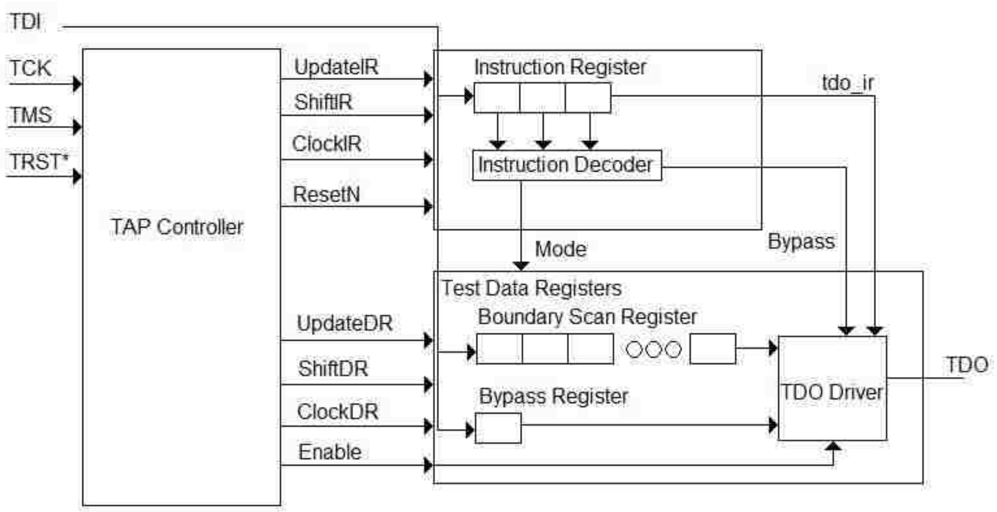

Figure 2.3: Architecture of the TAP controller.

There are sixteen states in the TAP controller's state machine [10]. The lines TCK and TMS determine how the machine goes from one state to another. These states send signals to the instruction register and data register. These signals include UpdateIR, ShiftIR, ClockIR, ResetN, UpdateDR, ShiftDR, ClockDR, and Enable. See Figure 2.2. At first, the TAP controller is initialized in the Test Logic Reset state by TRST*. Next, based on the value of TMS, the state machine may either move to the Run Test Idle state or remain in Test Logic Reset. If it were to transition to

Run Test Idle, it could move on to the DR Scan or the IR Scan states by holding TMS high for the next two rising edges of TCK.

Usually one begins by holding TRST* low for a few TCK clock cycles in order to reset the TAP controller state machine. Normally a testing sequence includes using TCK to regulate the system and using TMS to travel through the state machine and accomplish some procedure. Such procedures include either serially loading an instruction register or serially loading/reading the data registers which test the device.

The VHDL code which implements the TAP controller can be seen on the following page. The TRST* signal is named trstn. Often frowned upon in VHDL coding is the notion of gated clocks. However, it is common for TAP controllers to utilize such clocks in order to control the instruction register and data registers and also to drive the timing of the boundary scan correctly.

```
LIBRARY ieee;
   USE ieee.std_logic_1164.all;

ENTITY tapcontroller IS
   PORT (
      tms        : IN STD_LOGIC;--test mode select
      tck        : IN STD_LOGIC;--test clock
      trstn      : IN STD_LOGIC;--test reset signal
      ShiftIR    : OUT STD_LOGIC;--boundary scan register
      ShiftDR    : OUT STD_LOGIC;
      ClockIR    : OUT STD_LOGIC;
      ClockDR    : OUT STD_LOGIC;
      UpdateIR   : OUT STD_LOGIC;
      UpdateDR   : OUT STD_LOGIC;
      Resetn     : OUT STD_LOGIC;
      Enable     : OUT STD_LOGIC
   );
END tapcontroller;

ARCHITECTURE trans OF tapcontroller IS

   SIGNAL state        : STD_LOGIC_VECTOR(3 DOWNTO 0);
   --for where we are in the FSM

SIGNAL GSIR   : STD_ULOGIC;--instruction register
SIGNAL GSDR   : STD_ULOGIC;--data register
SIGNAL GRST   : STD_ULOGIC;--reset
SIGNAL GENB   : STD_ULOGIC;--enable

--states
   SIGNAL TEST_LOGIC_RESET : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL RUN_TEST_IDLE : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL SELECT_DR_SCAN : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL CAPTURE_DR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL SHIFT_DR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL EXIT1_DR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL PAUSE_DR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL EXIT2_DR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL UPDATE_DR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL SELECT_IR_SCAN : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL CAPURE_IR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL SHIFT_IR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL EXIT1_IR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL PAUSE_IR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL EXIT2_IR : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL UPDATE_IR : STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN

--defining a map of where each state will go to according to tms
TEST_LOGIC_RESET <= state WHEN (tms = '1') ELSE "1100";
   RUN_TEST_IDLE <= "0111" WHEN (tms = '1') ELSE state;
   SELECT_DR_SCAN <= "0100" WHEN (tms = '1') ELSE "0110";
   CAPTURE_DR <= "0001" WHEN (tms = '1') ELSE "0010";
   SHIFT_DR <= "0001" WHEN (tms = '1') ELSE state;
   EXIT1_DR <= "0101" WHEN (tms = '1') ELSE "0011";
   PAUSE_DR <= "0000" WHEN (tms = '1') ELSE state;
   EXIT2_DR <= "0101" WHEN (tms = '1') ELSE "0010";
   UPDATE_DR <= "0111" WHEN (tms = '1') ELSE "1100";
   SELECT_IR_SCAN <= "1111" WHEN (tms = '1') ELSE "1110";
   CAPURE_IR <= "1001" WHEN (tms = '1') ELSE "1010";
   SHIFT_IR <= "1001" WHEN (tms = '1') ELSE state;
   EXIT1_IR <= "1101" WHEN (tms = '1') ELSE "1011";
   PAUSE_IR <= "1000" WHEN (tms = '1') ELSE state;
   EXIT2_IR <= "1101" WHEN (tms = '1') ELSE "1010";
   UPDATE_IR <= "0111" WHEN (tms = '1') ELSE "1100";

--defining what state we are in
   PROCESS (tck, trstn)
   BEGIN
      IF ((NOT(trstn)) = '1') THEN
         state <= "1111";
      ELSIF (tck'EVENT AND tck = '1') THEN
         CASE state IS
            WHEN "1111" =>
               state <= TEST_LOGIC_RESET;
            WHEN "1100" =>
               state <= RUN_TEST_IDLE;
            WHEN "0111" =>
               state <= SELECT_DR_SCAN;
            WHEN "0110" =>
               state <= CAPTURE_DR;
            WHEN "0010" =>
               state <= SHIFT_DR;
            WHEN "0001" =>
               state <= EXIT1_DR;
            WHEN "0011" =>
               state <= PAUSE_DR;
            WHEN "0000" =>
               state <= EXIT2_DR;
            WHEN "0101" =>
               state <= UPDATE_DR;
            WHEN "0100" =>
               state <= SELECT_IR_SCAN;
            WHEN "1110" =>
               state <= CAPURE_IR;
            WHEN "1010" =>
               state <= SHIFT_IR;
            WHEN "1001" =>
               state <= EXIT1_IR;
            WHEN "1011" =>
               state <= PAUSE_IR;
            WHEN "1000" =>
               state <= EXIT2_IR;
            WHEN OTHERS => --1101
               state <= UPDATE_IR;
         END CASE;
      END IF;
   END PROCESS;

--logic for UpdateIR/DR
process (tck, trstn)
begin
if trstn = '0' then
UpdateIR <= '0';
UpdateDR <= '0';
```

```
elsif tck'event and tck = '0' then
UpdateIR<= state(0) and not(state(1)) and state(2) and state(3);
UpdateDR<=state(0)and not(state(1))and state(2)and not(state(3));
end if;
end process;

--logic for ClockIR/DR
process(state(0), state(1), state(3), tck)
begin
if (not(state(0)) and state(1) and state(3)) = '1' then
ClockIR <= tck;
else
ClockIR <= '1';
end if;
end process;

process(state(0), state(1), state(3), tck)
begin
if (not(state(0)) and state(1) and not(state(3))) = '1' then
ClockDR <= tck;
else
ClockDR <= '1';
end if;
end process;

--logic for ShiftIR/DR, Resetn, and Enable
GSIR<=(not(state(0))and state(1))and (not(state(2))and state(3));
GSDR<=(not(state(0))and state(1))
and (not(state(2))and not(state(3)));
GRST<=not((state(0)and state(1))and (state(2)and state(3)));
GENB<= GSIR or GSDR;
   PROCESS (tck, trstn)
   BEGIN
      IF ((NOT(trstn)) = '1') THEN
         ShiftIR <= '0';
         ShiftDR <= '0';
         Resetn <= '0';
         Enable <= '0';
      ELSIF (tck'EVENT AND tck = '0') THEN
         ShiftIR <= GSIR;
         ShiftDR <= GSDR;
         Resetn <= GRST;
         Enable <= GENB;
      END IF;
   END PROCESS;

END trans;
```

2.2.1 The Instruction Register

The instruction register (IR) must contain at least two bits because boundary scan employs two data registers: the bypass and boundary scan registers [10]. If the data register is chosen, the instruction register must select which of the data registers will be used for the boundary scan. In the CaptureDR state, the instruction register figures out where the data register will get its value from. It also determines if the values are to be sent to core logic or output cells. The designer of the TAP controller can specify instructions, but three instructions are necessary and two more are highly recommended:

The Bypass instruction puts the bypass register in the data register scan path in order to skip the testing of a specific device. This means that one flip flop passes along the bit stream from TDI to TDO. By using this instruction, a user can avoid devices that do not need testing and not have to send the bit stream scan through all the shift register stages of every device. It is recommended that this instruction be signified by ones in every cell of the instruction register.

The Sample/Preload instruction puts the boundary scan register in the data register scan path. This means that the bits the user wants tested on the inputs and outputs of the device are loaded into the data register. Sample/Preload simply copies the device IO values into the data registers in the CaptureDR state. Then the values can be subsequently moved out of the data register using the ShiftDR state. As this occurs, fresh values are loaded into the data registers but not activated on the inputs and outputs of the device.

The Extest instruction is a lot like Sample/Preload but makes it possible for the TAP controller to use boundary scan on the output cells of a device. This means that Extest can test the interconnecting lines between devices. Extest is accomplished by loading the values of the data register onto the output pads. Thus, by using a combination of ones and zeros on the output cells of one device and testing the input cells of another device, the connectivity between the devices can be determined.

The following two instructions are not required but are highly recommended:

The Intest instruction makes it possible to check circuitry within a device. Intest can more precisely pinpoint a discontinuity or short this way. By pacing through each part of a device, the core circuitry can be tested. The boundary scan registers are responsible for inserting ones and zeros not onto the input or output cells of the device, but into the core circuitry.

The RunBIST (Built In Self Test) instruction can trigger any Built In Self Tests that the device may be equipped with. While this instruction is dependent upon whether the device has a BIST or not, it can be invaluable in a thorough check of functionality.

The instruction register is made up of at least two implementations of Figure 2.4. A shift register is created by linking two ClockIR flip flops. The bit stream inserts data into this shift register when the TAP controller is in the CaptureIR state. Then in the ShiftIR state, new data and thus another instruction is loaded. The instructions that get loaded are determined by the user. However, in order to check the integrity of the scan chain, the last two bits must be a 01 combination. The UpdateIR state copies the instruction register data in parallel to the contents of the shift register so that the instruction register never contains an unrecognized instruction. It is recommended that when the TAP controller is reset the instruction register get loaded with an instruction that will not mess up the function of the device or the TAP controller.

The VHDL code for an instruction register is shown on the following page. The code creates the registers needed for a three bit instruction register. Depending on the combination of bits, the signals mode-in, mode-out, and bypass are toggled.

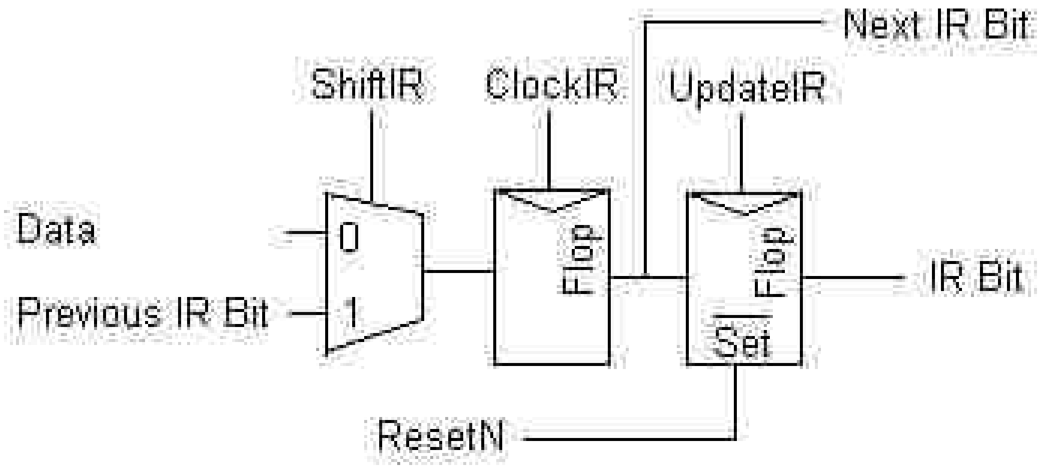

Figure 2.4: Typical IR bit.

```
LIBRARY ieee;
   USE ieee.std_logic_1164.all;

ENTITY inst_reg IS
   PORT (
      tdi       : IN STD_LOGIC;
      Resetn    : IN STD_LOGIC;
      ClockIR   : IN STD_LOGIC;
      UpdateIR  : IN STD_LOGIC;
      ShiftIR   : IN STD_LOGIC;
      tdo_ir    : OUT STD_LOGIC;
      mode_in   : OUT STD_LOGIC;
      mode_out  : OUT STD_LOGIC;
      bypass    : OUT STD_LOGIC
   );
END inst_reg;

ARCHITECTURE trans OF inst_reg IS

   SIGNAL shiftreg : STD_LOGIC_VECTOR(1 DOWNTO 0);
   SIGNAL instreg  : STD_LOGIC_VECTOR(1 DOWNTO 0);
   SIGNAL constantValue : STD_LOGIC_VECTOR(1 DOWNTO 0);

BEGIN

constantValue<=(tdi&shiftreg(1))WHEN(ShiftIR='1')ELSE"10";

   PROCESS (ClockIR)
   BEGIN
      IF (ClockIR'EVENT AND ClockIR = '1') THEN
         shiftreg <= constantValue;
      END IF;
   END PROCESS;

   PROCESS (UpdateIR, Resetn)
   BEGIN
      IF ((NOT(Resetn)) = '1') THEN
         instreg <= "11";
      ELSIF (UpdateIR'EVENT AND UpdateIR = '1') THEN
         instreg <= shiftreg;
      END IF;
   END PROCESS;

   tdo_ir <= shiftreg(0);

PROCESS (instreg)
   BEGIN
      IF (instreg = "11") THEN
         bypass <= '1';
ELSE
bypass <= '0';
END IF;
   END PROCESS;

PROCESS (instreg)
   BEGIN
      IF (instreg = "10") THEN --NOP SINCE WE DON'T HAVE INTEST
         mode_in <= '1';
ELSE
mode_in <= '0';
END IF;
   END PROCESS;

PROCESS (instreg)
   BEGIN
      IF ((instreg="10") OR (instreg="00")) THEN --NOP/EXTEST
         mode_out <= '1';
ELSE
mode_out <= '0';
END IF;
   END PROCESS;

END trans;
```

2.2.2 The Data Register

The data register, which consists of at least a bypass register and boundary scan register, is what determines how the test data will be input into a device and how to report the outcome of the test [10]. While the bypass register is only a single bit, the boundary scan register must be able to accommodate all of the input and output pins of a device. Additional internal registers for other purposes may also be added to the data register as can be seen in Figure 2.5. This way the boundary scan can easily use alternate built in self testers. A designer can simply add a multiplexer controlled by the TAP finite state machine to choose either the internal register or the regular boundary scan register. The resulting bit stream is then sent to the TDO output. However, care must be taken to make the instruction register wide enough to accommodate the number of different internal data registers. This is done so that each internal data register has a unique identification.

Every cell of the boundary scan register is actually linked to every pin on the device. See Figure 2.5 for what one of these cells looks like. It is simply a giant shift register that processes the bit stream inputs and reads the devices outputs. A mode signal selects whether or not the boundary scan is in an off state or is currently testing. This allows the IO data to be monitored at will.

As can be seen in Figure 2.6 the output and input cells are quite similar, but some key differences exist between the boundary scan register inputs and outputs. The input cell takes DataIn and sends Qout to the device. The output cell takes DataIn from the device and reports Qout back to the bit stream.

More complicated devices are bidirectional and have inputs that also function as outputs. A special set up for this can be seen in Figure 2.7. Also, sometimes devices operate with signals other that ones and zeros must be treated differently because of their tristate nature.

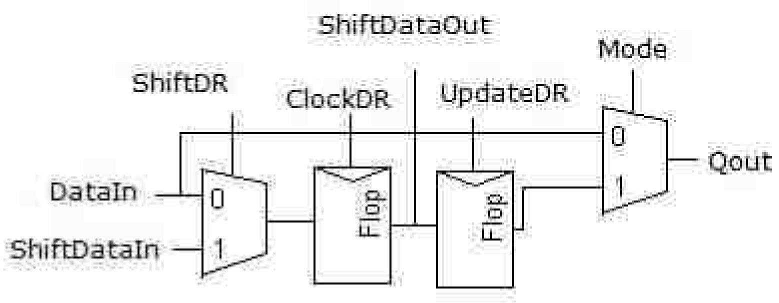

Figure 2.5: Boundary Scan Register.

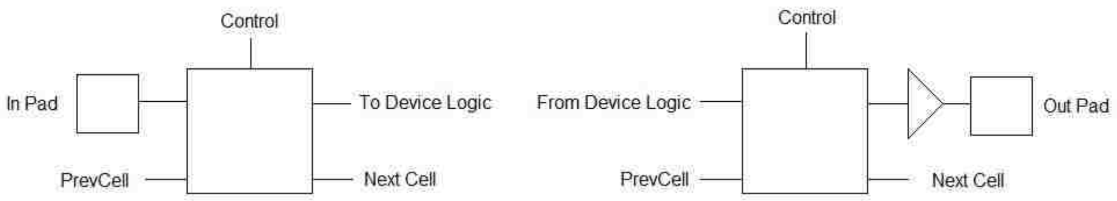

Figure 2.6: Input and Output Cells.

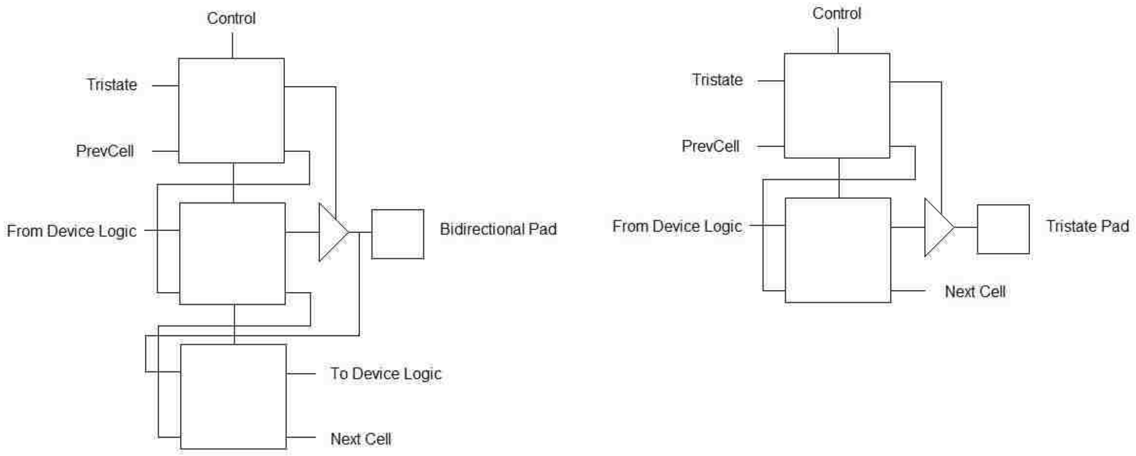

Figure 2.7: Bidirectional and Tristate cells.

Two differerent lines ‘mode-in’ and ‘mode-out’ are sent to the input cells and output cells respectively. These mode lines allow TDO to be driven by either the boundary scan register or straight from the bit stream. For example, if one wanted to utilize the TAP controllers functionality for the Intest instruction, both ‘mode-in’ and ‘mode-out’ would be driven high so that both inputs and outputs of the device could be monitored. If one wanted to bypass the TAP controller entirely and let the device function normally, both modes would be held low. However, for an Extest instruction the ‘mode-out’ line is driven high so that the outputs can be controlled.

When the TAP controller enters the ShiftIR or ShiftDR state, the last bit in the instruction register or data register is shifted into the TDO driver. Otherwise, TDO is driven in tristate so that race conditions are avoided and so that several TDO drivers can output to the same line.

In this thesis a TDO driver with the configuration seen in Figure 2.8 is used, except no alternative internal register is utilized. The Control bit automatically selects the boundary scan register, but this choice can be overridden by the Bypass multiplexer. If the Bypass bit does not override the boundary scan, another multiplexer chooses between the instruction register and the data register.

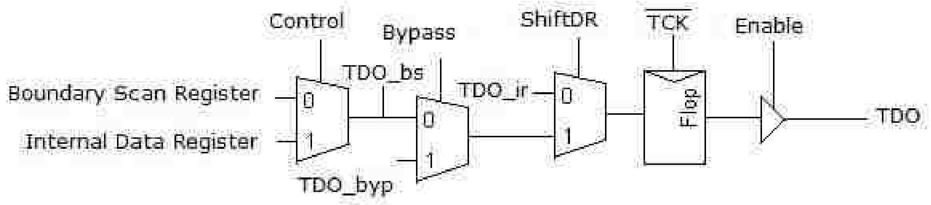

Figure 2.8: TDO Driver.

For this thesis, a simple circuit with four inputs and four outputs is used. The data register VHDL module can be seen on the following page. The boundary scan register is eight bits wide to accommodate the inputs and outputs.

```
LIBRARY ieee;
   USE ieee.std_logic_1164.all;

ENTITY data_reg IS
   PORT (
      a          : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      fromlogic  : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      tck        : IN STD_LOGIC;
      tdi        : IN STD_LOGIC;
      tdo_ir     : IN STD_LOGIC;
      ClockDR    : IN STD_LOGIC;
      UpdateDR   : IN STD_LOGIC;
      ShiftDR    : IN STD_LOGIC;
      Enable     : IN STD_LOGIC;
      mode_in    : IN STD_LOGIC;
      mode_out   : IN STD_LOGIC;
      bypass     : IN STD_LOGIC;
      y          : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      tologic    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      tdo        : OUT STD_LOGIC
   );
END data_reg;

ARCHITECTURE trans OF data_reg IS

   SIGNAL shiftreg     : STD_LOGIC_VECTOR(7 DOWNTO 0);
   SIGNAL datareg      : STD_LOGIC_VECTOR(7 DOWNTO 0);
   SIGNAL tdo_selected : STD_LOGIC;
   SIGNAL tdo_byp      : STD_LOGIC;
   SIGNAL tdo_delayed  : STD_LOGIC;
   SIGNAL shiftregNew : STD_LOGIC_VECTOR(7 DOWNTO 0);

BEGIN

shiftregNew<=(tdi&shiftreg(7 DOWNTO 1))WHEN(ShiftDR='1')--
ELSE(a&fromlogic);
   PROCESS (ClockDR)
   BEGIN
      IF (ClockDR'EVENT AND ClockDR = '1') THEN
         shiftreg <= shiftregNew;
      END IF;
   END PROCESS;

   PROCESS (UpdateDR)
   BEGIN
      IF (UpdateDR'EVENT AND UpdateDR = '1') THEN
         datareg <= shiftreg;
      END IF;
   END PROCESS;

   tologic <= datareg(7 DOWNTO 4) WHEN (mode_in = '1') ELSE a;
   y <= datareg(3 DOWNTO 0) WHEN (mode_out = '1') ELSE fromlogic;

   PROCESS (ClockDR)
   BEGIN
      IF (ClockDR'EVENT AND ClockDR = '1') THEN
         tdo_byp <= tdi AND ShiftDR;
      END IF;
   END PROCESS;

   tdo_selected <= tdo_byp WHEN (bypass = '1') ELSE
                   shiftreg(0) WHEN (ShiftDR = '1') ELSE
                   tdo_ir;
   PROCESS (tck)
   BEGIN
      IF (tck'EVENT AND tck = '0') THEN
         tdo_delayed <= tdo_selected;
      END IF;
   END PROCESS;

   tdo <= tdo_delayed WHEN (Enable = '1') ELSE 'Z';

END trans;
```

2.2.3 The Top Module

The Top module is made up of the TAP controller, the bypass register, the instruction register, and the boundary scan register (made up of input and output cells around the device). The Top module is controlled by the signals TDI, TCK, TMS, TRST, and TDO. The VHDL for the Top module is on the following page. Figure 2.1 shows a complete implementation of a Top module boundary scan. Figure 2.9 below is the Register Transfer Level (RTL) representation of the Top module.

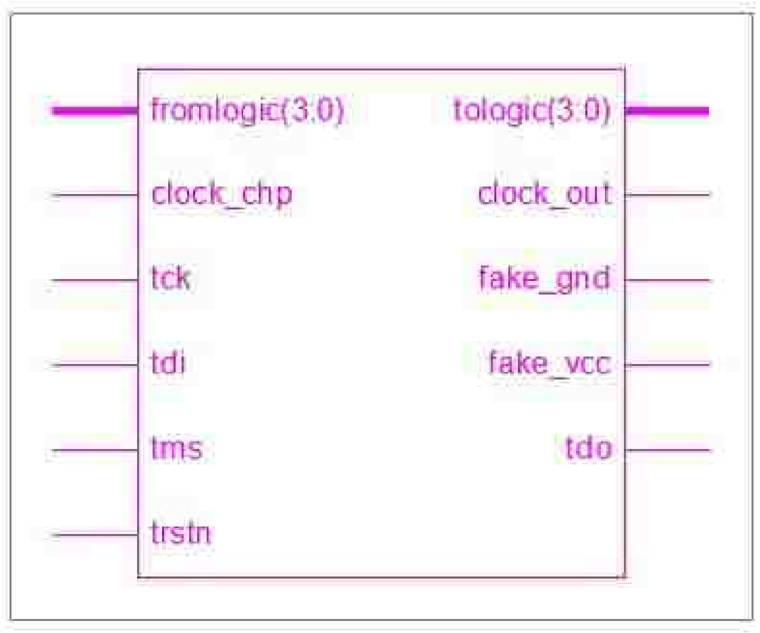

Figure 2.9: Top module RTL.

```
LIBRARY ieee;
   USE ieee.std_logic_1164.all;

ENTITY top IS
   PORT (
      tck        : IN STD_LOGIC;
      tms        : IN STD_LOGIC;
      tdi        : IN STD_LOGIC;
      trstn      : IN STD_LOGIC;
      a          : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      tdo        : OUT STD_LOGIC;
      y          : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
   );
END top;

ARCHITECTURE trans OF top IS
   COMPONENT tapcontroller IS
      PORT (
         tms        : IN STD_LOGIC;
         tck        : IN STD_LOGIC;
         trstn      : IN STD_LOGIC;
         ShiftIR    : OUT STD_LOGIC;
         ShiftDR    : OUT STD_LOGIC;
         ClockIR    : OUT STD_LOGIC;
         ClockDR    : OUT STD_LOGIC;
         UpdateIR   : OUT STD_LOGIC;
         UpdateDR   : OUT STD_LOGIC;
         Resetn     : OUT STD_LOGIC;
         Enable     : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT inst_reg IS
      PORT (
         tdi        : IN STD_LOGIC;
         Resetn     : IN STD_LOGIC;
         ClockIR    : IN STD_LOGIC;
         UpdateIR   : IN STD_LOGIC;
         ShiftIR    : IN STD_LOGIC;
         tdo_ir     : OUT STD_LOGIC;
         mode_in    : OUT STD_LOGIC;
         mode_out   : OUT STD_LOGIC;
         bypass     : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT data_reg IS
      PORT (
         a          : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         fromlogic  : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         tck        : IN STD_LOGIC;
         tdi        : IN STD_LOGIC;
         tdo_ir     : IN STD_LOGIC;
         ClockDR    : IN STD_LOGIC;
         UpdateDR   : IN STD_LOGIC;
         ShiftDR    : IN STD_LOGIC;
         Enable     : IN STD_LOGIC;
         mode_in    : IN STD_LOGIC;
         mode_out   : IN STD_LOGIC;
         bypass     : IN STD_LOGIC;
         y          : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
         tologic    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
         tdo        : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT core IS
      PORT (
         tologic    : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         fromlogic  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
      );
   END COMPONENT;

   SIGNAL tologic      : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL fromlogic    : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL UpdateIR     : STD_LOGIC;
   SIGNAL ShiftIR      : STD_LOGIC;
   SIGNAL ClockIR      : STD_LOGIC;
   SIGNAL UpdateDR     : STD_LOGIC;
   SIGNAL ShiftDR      : STD_LOGIC;
   SIGNAL ClockDR      : STD_LOGIC;
   SIGNAL Resetn       : STD_LOGIC;
   SIGNAL Enable       : STD_LOGIC;
   SIGNAL mode_in      : STD_LOGIC;
   SIGNAL mode_out     : STD_LOGIC;
   SIGNAL bypass       : STD_LOGIC;
   SIGNAL tdo_ir       : STD_LOGIC;

   --intermediate signals for outputs
   SIGNAL tdo_temp     : STD_LOGIC;
   SIGNAL y_temp       : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
   --drive referenced outputs
   tdo <= tdo_temp;
   y <= y_temp;

   c : core
      PORT MAP (
         tologic,
         fromlogic
      );

   tc : tapcontroller
      PORT MAP (
         tms,
         tck,
         trstn,
         ShiftIR,
         ShiftDR,
         ClockIR,
         ClockDR,
         UpdateIR,
         UpdateDR,
         Resetn,
         Enable
      );
   ir : inst_reg
      PORT MAP (
         tdi,
         Resetn,
         ClockIR,
         UpdateIR,
         ShiftIR,
         tdo_ir,
         mode_in,
         mode_out,
         bypass
      );

   dr : data_reg
      PORT MAP (
         a,
         fromlogic,
         tck,
         tdi,
         tdo_ir,
         ClockDR,
         UpdateDR,
         ShiftDR,
         Enable,
         mode_in,
         mode_out,
         bypass,
         y_temp,
         tologic,
         tdo_temp
      );

END trans;
```

An example sequence for a boundary scan is as follows:

- Reset the TAP controller
- Sample/Preload
- Bit stream loaded onto boundary scan cells
- Intest
- Bit stream shifted out
- New bit stream loaded
- next instruction
- ...repeat...

A simulation of the TAP controller proved invaluable in the development of this thesis. A circuit that simply took four inputs and sent them straight through as outputs was used. Figure 2.10 shows the waveform that was generated. To begin, the TRST* signal is used to reset the TAP controller. As can be seen tologic and fromlogic are equal, 0000. The instruction register is initially in Bypass, but then moves to 01 (Extest). Here we load information (a '7') from TDI into the data register. The test continues to process instructions and shift information.

2.3 Computer Program

A computer program developed by Alonzo Vera, Ph.D., is used to write JTAG instructions to the TAP controller module. The TAP controller module then interprets the instructions and outputs signals to pins on the FPGA. For instance, inputting

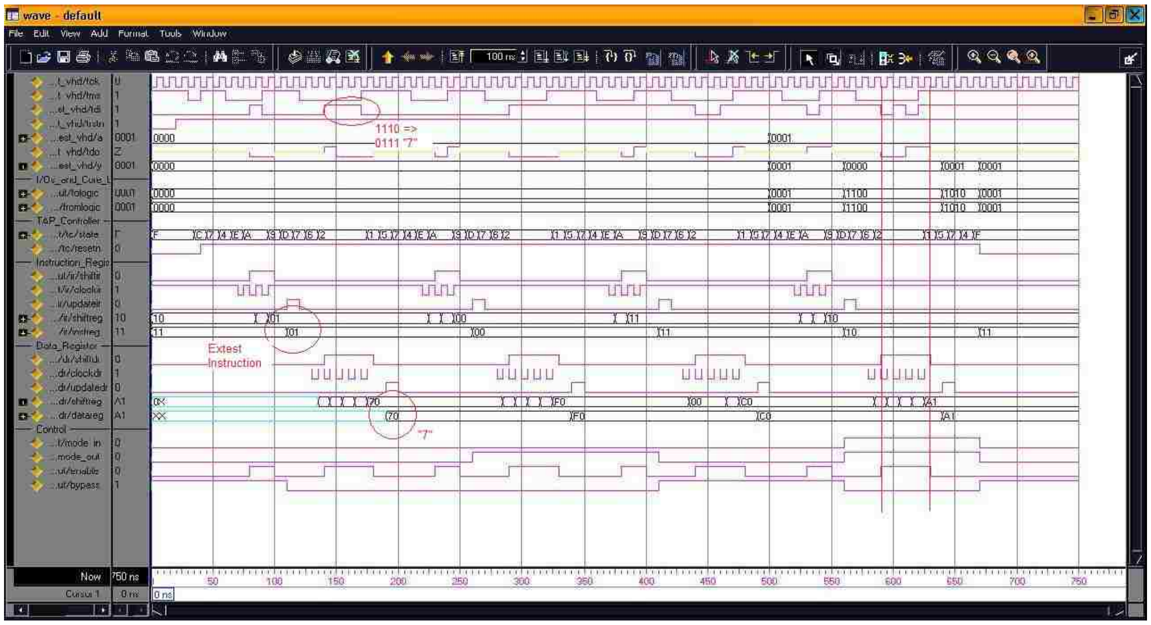

Figure 2.10: Two bit IR waveform.

the number “15” into the computer program results in voltage assertions on all four output lines while inputting the number “5” into the computer program results in voltage assertions on the second and fourth output lines.

A Xilinx Parallel Cable III cable administers the TCK, TMS, TDI, TDO, and TRST lines to pins on the FPGA board (see Figure 2.11).

2.4 Reconfigurable Microstrip Antennas

In general, the formulas used to calculate frequency, polarization, and other radiation effects are the same as those used for non-reconfigurable microstrip antennas. This is because p-i-n diodes and capacitors can be designed to have a relatively small impact on the radiation pattern. When designed appropriately, one can use the following formulas to approximate radiation effects. In practice, computer programs such as

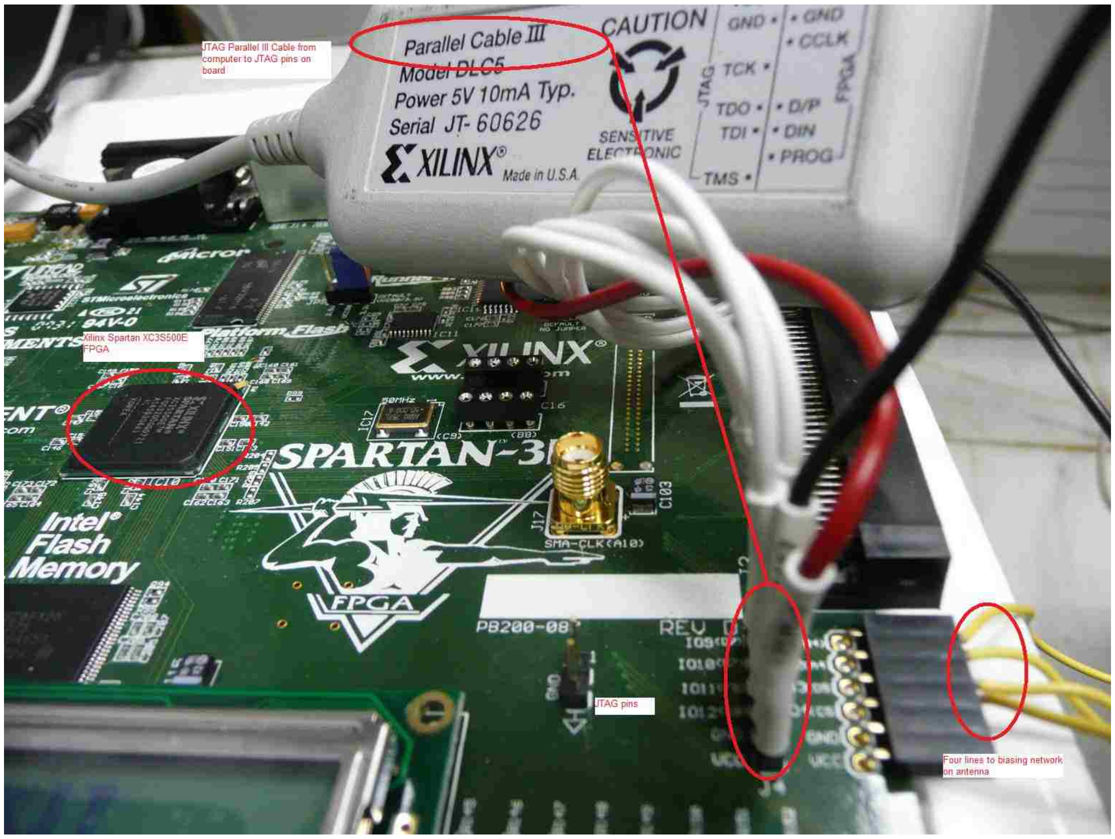
JTAG Parallel III Cable from computer to JTAG pins on board
Xilinx Spartan XC3S500E FPGA
JTAG pins
Four lines to biasing network on antenna

Figure 2.11: Parallel Cable III.

Advanced Design System (ADS) by Agilent and High Frequency Structure Simulator (HFSS) by Ansoft are invaluable in calculating the characteristics of antenna. Since microstrip patch antennas are commonly rectangular, this paper will examine the formulas that correspond with such antennas. More detailed explanations of these formulas can be found in [4].

2.4.1 Quarter Wave Transmission Line

In the biasing network of the reconfigurable antennas which this thesis proposes, the quarter wave transmission lines are terminated by radial stubs to minimize coupling.

The design of the quarter wave transmission line is mainly dependent upon the frequency of interest and other factors such as the dielectric constant. Quarter wave transmission lines represent a ninety degree electrical length which creates an effective open circuit for RF signals. However, for DC signals it is a short circuit. Thus, it is possible to create a biasing network.

In order to minimize losses and prevent current from flowing down the quarter wave bias lines, the width of the quarter wavelength transmission lines must be small. A better solution would be to utilize expensive high-tech resistive materials to inhibit current flow. However, in order to minimize costs and simplify design, the quarter wave biasing line approach is chosen as a valid solution.

Just as the width of the quarter wave transmission lines controls the RF signal on one side of the diode biasing network, a 100Ω resistive load is attached to the other side of the network to limit current flow toward the FPGA's output lines. This configures the current and voltage to more optimally bias the diode.

2.4.2 Radial Stubs

A radial stub is just a modification of an open circuit stub. It has lower impedance but still gives a good return loss [11]. Due to the large area at the open end of the radial stub, a large fringing capacitance is created [12] [13]. By connecting it to a thin, high impedance transmission line, a good biasing network can be created. Figure 2.12 shows the transmission line connecting to the radial stub [14].

2.4.3 p-i-n Diodes and Biasing Capacitors

p-i-n diodes are used to switch between states of the antenna [15] [16]. A DC bias voltage is created by using a shorting pin [17] on one side of the diode while connecting

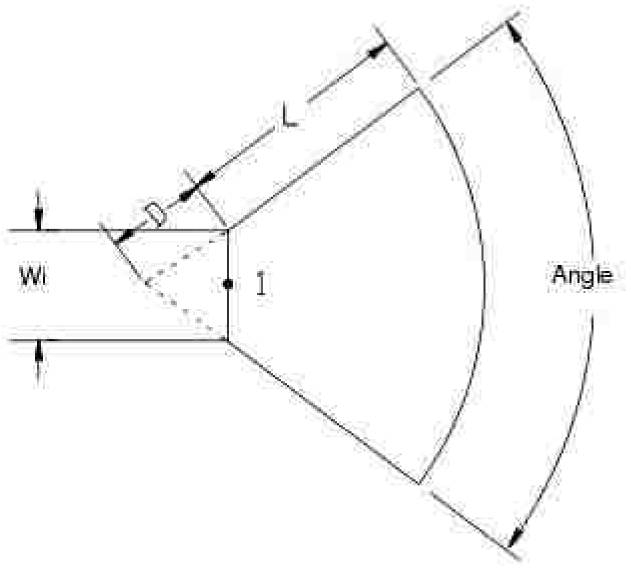

Figure 2.12: Radial stub diagram.

the other side of the diode to positive voltage. Gallium Arsenic p-i-n diodes operate well at high frequencies and can be fabricated on a small scale [18]. In this manner, the GaAs p-i-n diodes interfere minimally with the radiation of the antenna.

Another device that is used in the bias network is the biasing capacitor. These capacitors prevent DC currents from passing to other parts of the antenna, but allow the high frequency RF signal currents to flow through.

Chapter 3

Design and Simulation

3.1 Quarter Wavelength Transmission Lines

Quarter wavelength transmission lines were used to bias the p-i-n diodes at the correct voltage and current. This method of biasing diodes has proved useful in other research [19]. Selecting the width of the quarter-wave transmission lines allow the voltage from the FPGA to be scaled down to an optimal level for biasing the p-i-n diodes. Agilent ADS LineCalc was used to obtain the parameters for the transmission lines. These parameters are shown in Table 3.1 and Table 3.2. The impedance level of Antenna 1 was much too low to prevent current from flowing to the biasing network. This was changed in Antenna 2 and an impedance of 130.4Ω was achieved.

Table 3.1: Antenna 1 Transmission Line Parameters

Parameter	Value
Frequency	7.7GHz
Substrate Dielectric	$2.2\varepsilon_0$
Substrate Height	1.6mm
Length	9.279mm
Width	6.85mm
Impedance	32.4Ω

Table 3.2: Antenna 2 Transmission Line Parameters

Parameter	Value
Frequency	5GHz
Substrate Dielectric	$4.2\varepsilon_0$
Substrate Height	2.35mm
Length	8.959mm
Width	0.5mm
Impedance	130.4Ω

3.2 Radial Stubs

Antenna 1 and Antenna 2 radial stub parameters were calculated using equations developed by H. A. Atwater [13][20]. Table 3.3 and Table 3.4 show these parameters.

Table 3.3: Antenna 1 Radial Stub Parameters

Parameter	Value
Frequency	7.7GHz
Substrate Dielectric	$2.2\varepsilon_0$
Substrate Height	1.6mm
Radius	7.594mm
Angle	90deg

Table 3.4: Antenna 2 Radial Stub Parameters

Parameter	Value
Frequency	5GHz
Substrate Dielectric	$4.2\varepsilon_0$
Substrate Height	2.35mm
Radius	6.921mm
Angle	90deg

3.2.1 p-i-n Diodes and Biasing Capacitors

p-i-n diodes and capacitors were modelled and simulated using small rectangles in HFSS. To bias the diode, the rectangle was extended to touch the other patch and complete the circuit. Likewise to turn the diode off, the rectangle is shrunk until there is no connectivity between patches. Other researchers found that 47pF capacitors were able to bias diodes satisfactorily [21]. To minimize radiation effects, a small GC4271-152 GaAs p-i-n diode was used [22].

3.3 TAP Controller and Command Interface

A Digilent Spartan 3E board with a Xilinx Spartan XC3S500E FPGA is programmed with the TAP Controller module. A Xilinx Parallel III cable is connected from a parallel port on a Linux PC to the JTAG interface pins on the Spartan 3E board [23]. Four pins on the Spartan 3E board serve as the outputs for driving signals to the diode biasing network. The following instructions describe how to use the Linux program via a terminal window on the computer.

```
Log on to the linux machine using your user name and password.
Use the command CTRL+ALT+F4 to open up a terminal screen.
Type (login severn) to access the severn account.
Type in the appropriate password.
Type (sudo i) to become root.
Type in the appropriate password.
Type (lsmod) to list all the modules in the kernel.
Type (rmmod lp) to remove the lp module.
Type (cd ../home/severn/Desktop/jtagseu_v2_00/src/) for directory.
Type (./jtagseu_v2_00 s 5) runs program, 0101 to output lines.
```

3.4 FPGA Voltage and Current Control Circuit

In order to more optimally control the switching of the GaAs p-i-n diodes, four 100Ω resistors are placed on the four output lines of the FPGA for the Antenna 2 design. This brings the voltage down and holds current levels under 100mA, which is closer to what a GC4271 p-i-n diode typically deals with.

3.5 Antenna 1

The first experimental reconfigurable antenna 'Antenna 1' consists of two sheets of metal as seen in Figure 3.1. The antenna is fed on one of the patches of metal using a coaxial feed located at point 9. The substrate has a height of 1.6mm and permittivity $2.2\varepsilon_0$. Initially, four quarter wave transmission lines are connected to the patches. Two quarter wave transmission lines connect to one patch while two other quarter wave transmission lines connect to the other. These quarter wave transmission lines were designed to bias two diodes which span the gap between the two patches. However, it was realized that two of the quarter wave lines were redundant and unnecessary for biasing the diodes, since the diodes were in parallel. Also realized was the fact that to bias both diodes in parallel, the diodes must have the exact same electrical properties. Otherwise, only one diode would bias and the other would remain in the off state.

3.6 Antenna 1 Simulations

The following simulations and tests were used to predict how Antenna 1 would perform after it is fabricated.

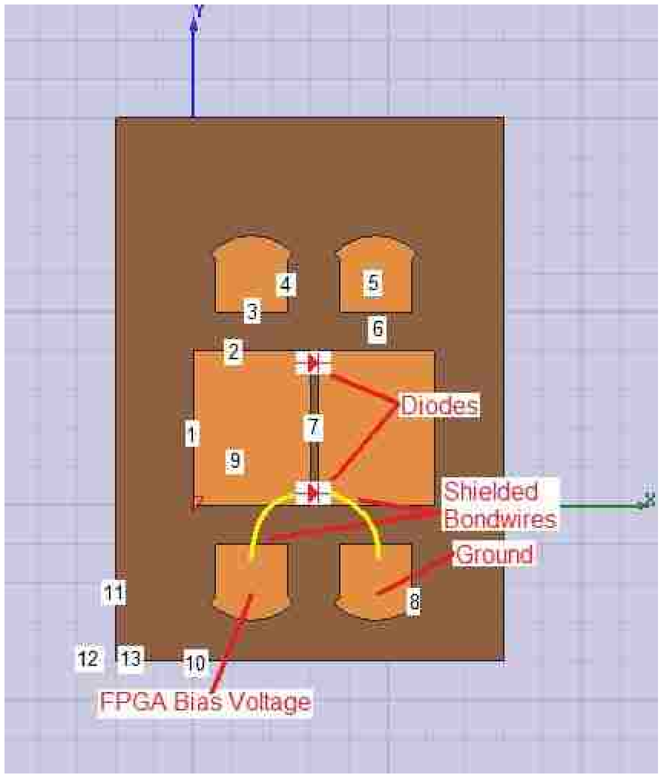

Figure 3.1: Antenna 1 diagram.

Table 3.5: Antenna 1 Dimensions

Part	Description	Length
1	Patch Y	20mm
2	Patch X	15mm
3	Quarter Wave TLine X	9.279mm
4	Quarter Wave TLine Y	6.85mm
5	Radial Stub Angle	90deg
6	Quarter Wave TLine Y Gap	5mm
7	Patch Gap	1mm
8	Radial Stub Radius	7.594mm
9	Feed Position	X: 5mm, Y: 5mm
10	Substrate X	50mm
11	Substrate Y	70mm
12	Substrate Dielectric	$2.2\varepsilon_0$
13	Substrate Height	1.6mm

3.6.1 Resonant Frequencies

The simulations in Figure 3.2 and Figure 3.3 demonstrate the S11 parameters for the case when the diodes are switched off and also for the case when the diodes are biased on.

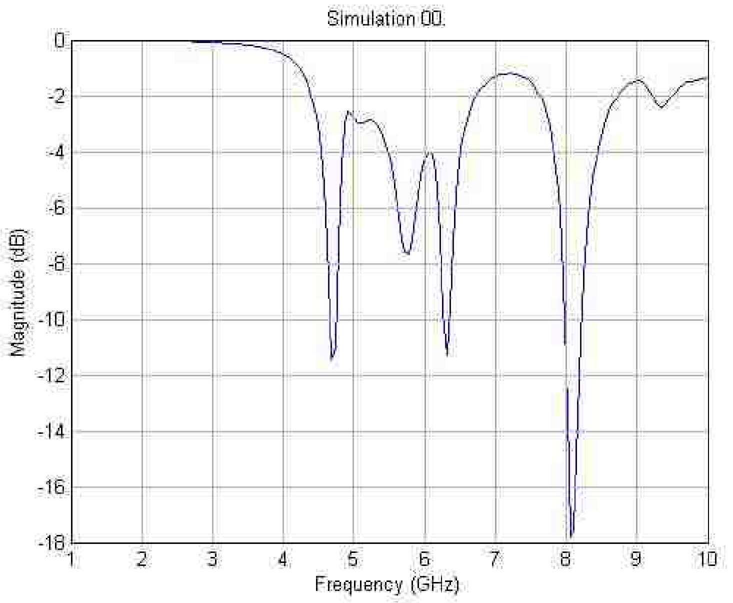

Figure 3.2: Simulation of Antenna 1 with diodes off.

3.6.2 Radiation Pattern

The radiation patterns of Antenna 1 at 5 GHz are shown in Figure 3.4 and Figure 3.5. The two lines represent the two configurations of the antenna, with diodes biased ($diodeX = 0.5mm$) and unbiased ($diodeX = 0.45mm$).

The figures show that reconfiguring the antenna drastically changes the radiation pattern. This means that the attachment of the large quarter wave transmission lines has altered the performance of the antenna. The lines act as radiating elements

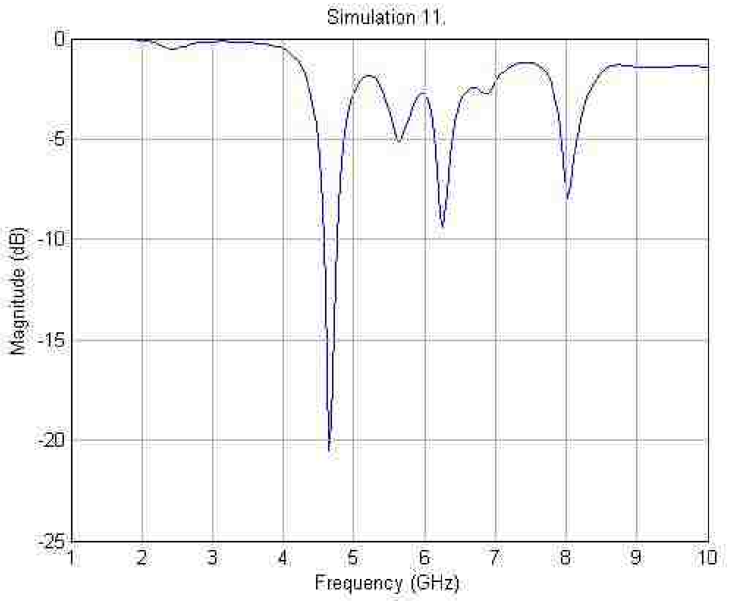

Figure 3.3: Simulation of Antenna 1 with diodes on.

and the result is propagation in unexpected directions.

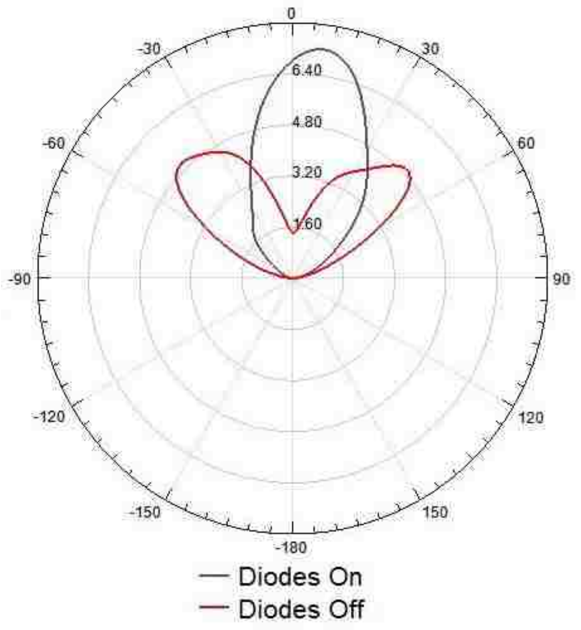

Figure 3.4: Antenna 1 ϕ plane radiation pattern.

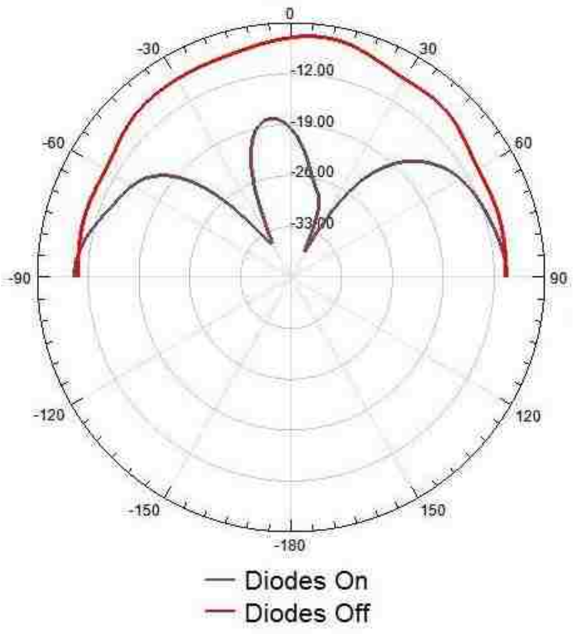

Figure 3.5: Antenna 1 θ plane radiation pattern.

3.7 Antenna 2

The second reconfigurable antenna 'Antenna 2'[24] is constructed with a central sheet of metal and four surrounding sheets of metal as shown in Figure 3.6. The antenna is fed on one of the arms of the central sheet of metal using a coaxial feed located at point 5. The substrate has a height of 2.35mm and permittivity $4.2\varepsilon_0$. The four surrounding patches have quarter wave transmission lines parallel to the X axis which end with radial stubs. Four pin diodes are connected to the four patches which are also parallel to the X axis but closer to the central patch. Capacitors parallel to the Y axis are then placed to connect the pin diodes to the central patch. The capacitors are required so that each diode can be controlled individually.

Biasing the pin diodes has different effects on the tuning of the antenna. When a diode is biased, the electric current flows from the central patch to the patch whose diode is biased. The radial stubs provide an effective decoupling network for the active components.

3.8 Antenna 2 Simulations

The following simulations and tests were used to predict how Antenna 2 would react after it was fabricated. As seen in Figure 3.7, cases 0010, 0110, 1010, and 1100 refer to different configurations of the antenna.

A zero in the case number represents an unbiased diode while a one in the case number represents a biased diode. The first binary 0 or 1 in the case number references the state of the diode in the first quadrant of the antenna (the upper right quadrant where x and y are positive). The second 0 or 1 in the case number references the state of the diode in the second quadrant of the antenna (the lower right quadrant where x is positive but y is negative). The third 0 or 1 references the third

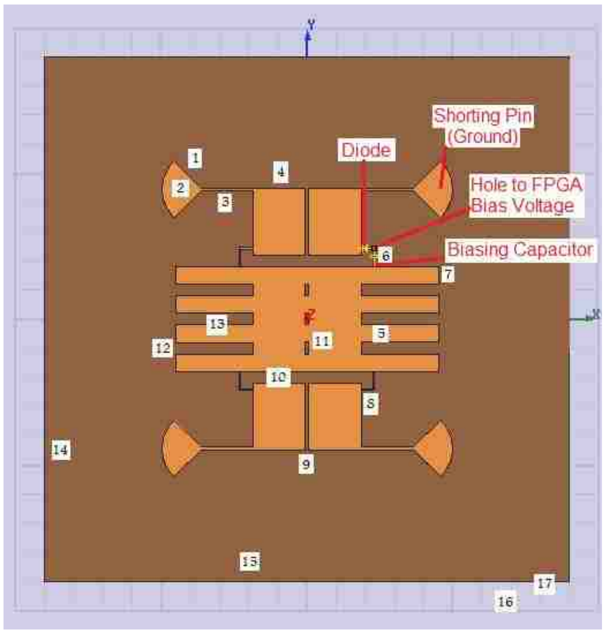

Figure 3.6: Antenna 2 diagram.

Table 3.6: Antenna 2 Dimensions

Part	Description	Length
1	Radial Stub	6.921mm
2	Radial Stub Angle	90deg
3	Quarter Wave TLine	X: 8.959mm, Y: 0.5mm
4	Outer Patch Width	9mm
5	Feed Position	X: 12.5mm, Y: -2.5mm
6	Capacitor	X: 0.25mm, Y: 3mm
7	Main Arm Y	3mm
8	Diode	X: 2mm, Y: 0.25mm
9	Outer Patch X Gap	0.6mm
10	Outer Patch Y Gap	2mm
11	Main Patch Hole	X: 0.6mm, Y: 2mm
12	Main Patch Arm Gap	Y: 2mm
13	Main Patch Arm X	X: 13.2mm
14	Substrate Y	90mm
15	Substrate X	90mm
16	Substrate Dielectric	$4.2\varepsilon_0$
17	Substrate Height	2.35mm

quadrant's diode and the fourth 0 or 1 references the fourth quadrant's diode.

3.8.1 Resonant Frequencies

The simulations of Antenna 2 demonstrate the S11 parameters for the cases 0010, 0110, 1010, and 1100 and are shown in Figures 3.8, 3.9, 3.10, and 3.11 respectively. Figure 3.12 then shows a comparison of these cases.

3.8.2 Radiation Pattern

The radiation patterns of Antenna 2 for cases 0010, 0110, 1010, and 1100 are shown in Figure 3.13, Figure 3.14, Figure 3.15, and Figure 3.16. The patterns are of the θ plane at the intersection of the ϕ plane equal to zero degrees. The different coloured traces represent the different resonant frequencies which are active in each case.

As can be seen from the figures, the radiation pattern changes according to frequency. However, it can be seen that the blue resonance around 4.8 GHz remains in the same general shape despite different configurations and cases. This is also true for the green resonance at 3.115 GHz and the red resonance around 1.8 GHz. This shows that the biasing network developed in Antenna 2 will mitigate the problem encountered in Antenna 1, where the quarter wavelength transmission lines were too large and acted as radiating elements. The yellow resonance at 2.305 GHz and the

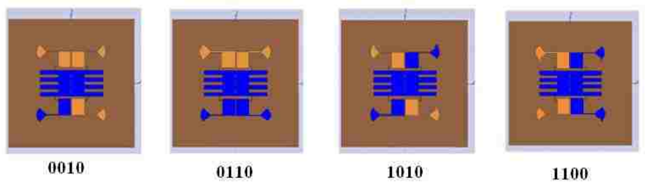

Figure 3.7: Antenna configuration for cases 0010, 0110, 1010, and 1100.

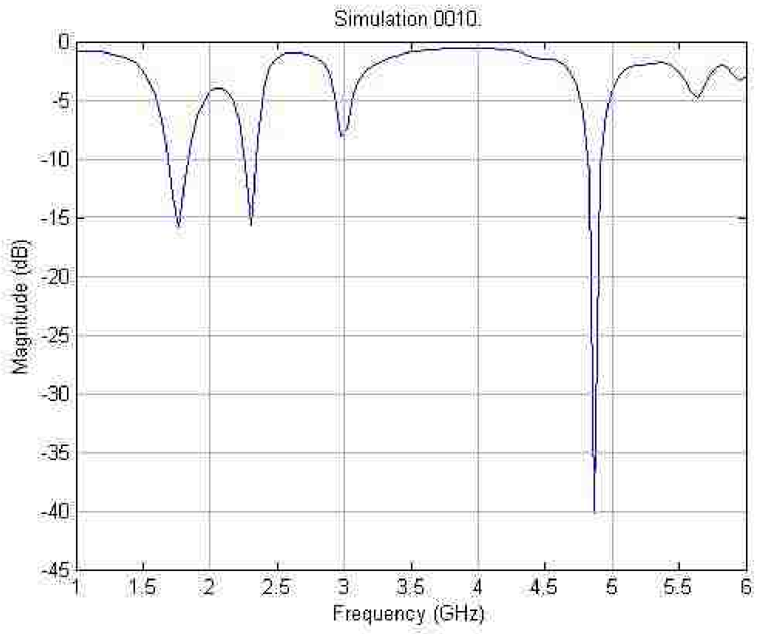

Figure 3.8: Simulation of Antenna 2 for case 0010.

purple resonance at 2.98 GHz are unique instances and it is presumed that other cases with one of these resonances will have similar radiation patterns.

The 3-D simulated radiation pattern for the 0010 configuration as well as the E and H plane cuts at 4.875 GHz are shown in Figures 3.17 and 3.18 respectively. This radiation pattern is plotted at a resonance frequency common to all the antenna configurations.

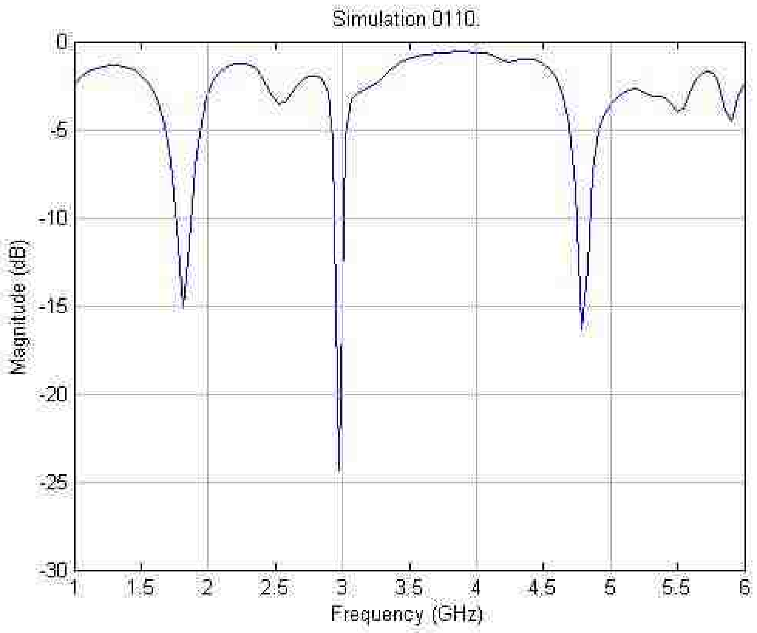

Figure 3.9: Simulation of Antenna 2 for case 0110.

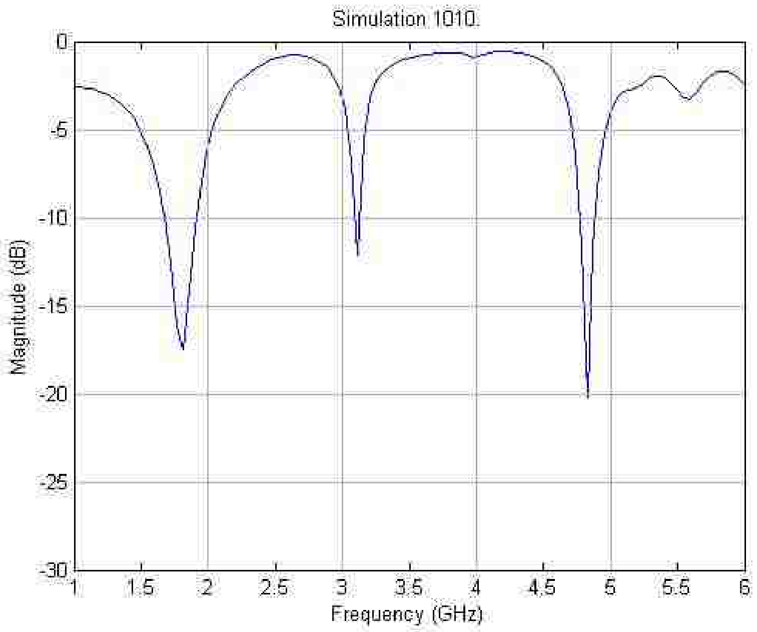

Figure 3.10: Simulation of Antenna 2 for case 1010.

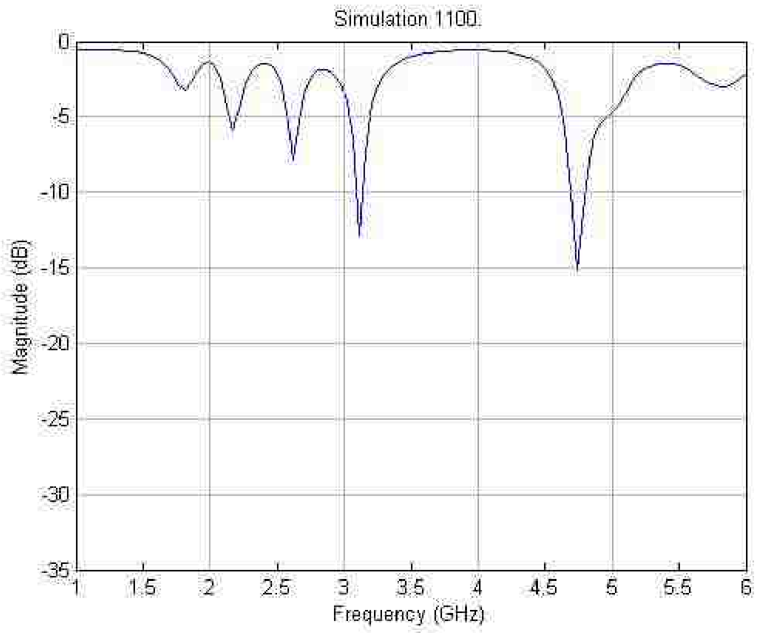

Figure 3.11: Simulation of Antenna 2 for case 1100.

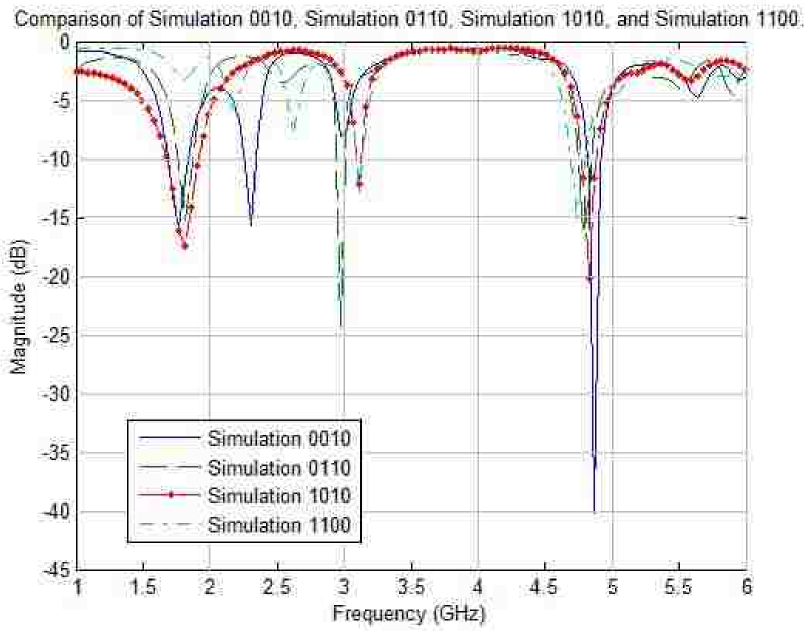

Figure 3.12: Comparison of cases 0010, 0110, 1010, and 1100.

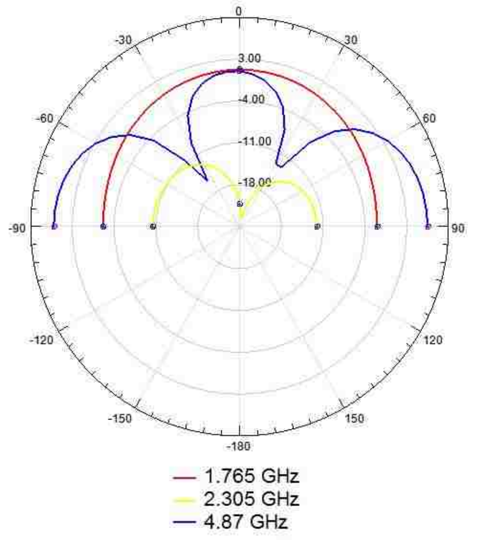

Figure 3.13: Antenna 2 Case 0010.

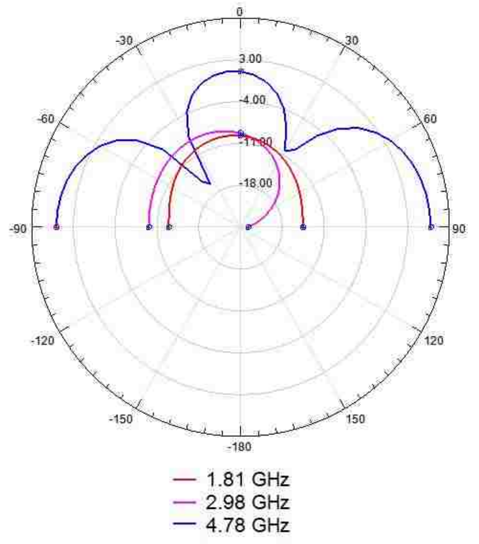

Figure 3.14: Antenna 2 Case 0110.

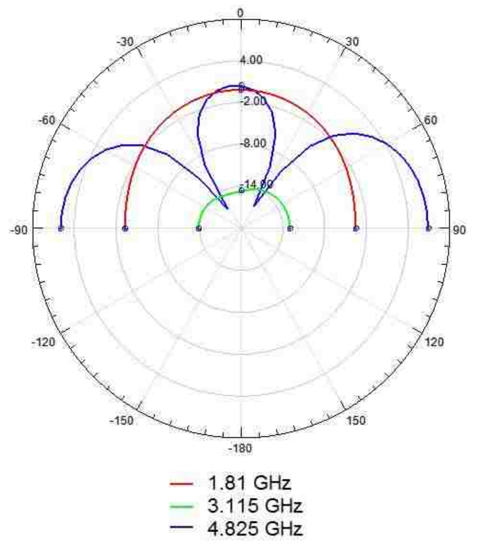

Figure 3.15: Antenna 2 Case 1010.

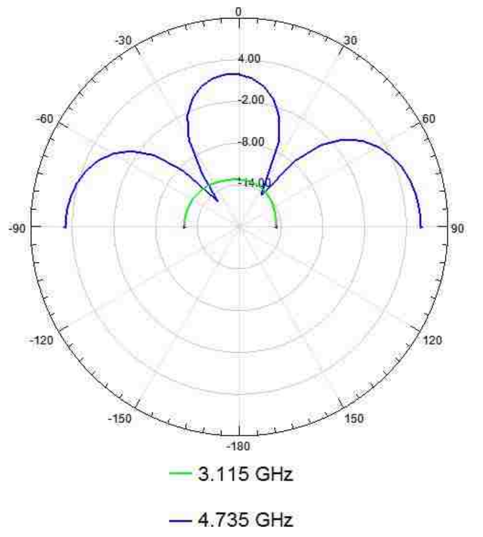

Figure 3.16: Antenna 2 Case 1100.

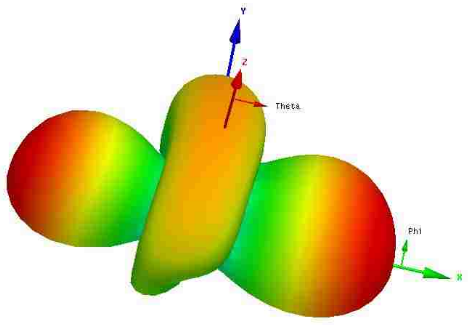

Figure 3.17: Simulated 3-D radiation pattern of case 0010 at 4.875 GHz.

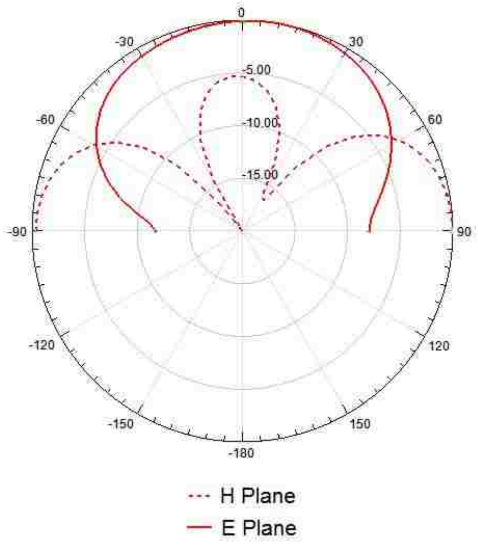

Figure 3.18: Case 0010 E and H plane cuts of the electric field at 4.875 GHz.

Chapter 4

Experimental Results

In order to understand the theory and simulations behind the previous work, two prototypes of Antenna 1 and Antenna 2 are fabricated and tested. The fabrication process includes the creation of a biasing network with diodes and capacitors. A Xilinx Spartan FPGA is programmed and interfaced with the biasing network on the antenna. A computer program controls the configuration of the antenna.

Several discrepancies between simulations and real results are noted. The reasons for theses differences can be attributed to capacitive effects, soldering, boundary conditions in software, and reflections.

4.1 PCB Fabrication

The base material used to create a reconfigurable microstrip antenna is the laminated dielectric board. This board is composed primarily of two materials. The first is the boards conductive laminate which covers the board's top and bottom and is typically made from copper. The second part of the board is the dielectric insulator that is sandwiched in between the copper ground plane and top copper layer. The dielectric is usually made of an epoxy resin that provides differing insulating values. Two companies that provide quality dielectric boards are Taconic Advanced Dielectric Division and the Rogers Corporation.

In addition to the microstrip materials, diodes must be chosen that can operate at the frequency required by the antenna's application. Often p-i-n or sometimes Schottky diodes that can operate on the order of several gigahertz are selected to function as biasing switches. Companies such as Microsemi and Avago make diodes that are well suited for reconfigurable antennas. Another element of some microstrip antennas are capacitors which are used to isolate DC biasing currents [21].

The laminated dielectric boards are 'printed' using computer files that are sent to an etching machine similar to the one made by Laser and Electronics in UNM's laboratory. The etching is done with diamond drill bits that etch away the conductive top layer and dielectric substrate as needed to produce the required pattern.

Two antennas were fabricated. The first antenna was an experiment to determine how the second antenna should be biased. In Antenna 1, two patches are connected via two GaAs p-i-n diodes in parallel. Antenna 2 elaborates on the first design and is used to learn more about the biasing network of the antenna.

4.1.1 Antenna 1

Antenna 1 was etched with the LPKF ProtoMat S62 machine onto Taconic substrate material with $2.2\varepsilon_0$ and height 1.6mm. A photo of this prototype is shown in Figure 4.2.

Figure 4.1: LPKF ProtoMat S62.

4.1.2 Antenna 2

Antenna 2 was also printed using an LPKF etching machine onto a Rogers substrate material with $4.2\varepsilon_0$ and height 0.235cm. A photo of the completed prototype is shown in Figure 4.3.

Holes about 0.75mm in diameter are drilled into the radial stubs of each surrounding patch. These stubs are shorted to the ground plane of the antenna.

Holes also about 0.75mm in diameter are drilled into the substrate at the intersection of the capacitor and diode. Four capacitors, each with a value of 47pF, are soldered from the main patch to the holes. Four of Microsemi's model GC4172 GaAs pin diodes are soldered from the capacitors to the outer patches. The FPGA's

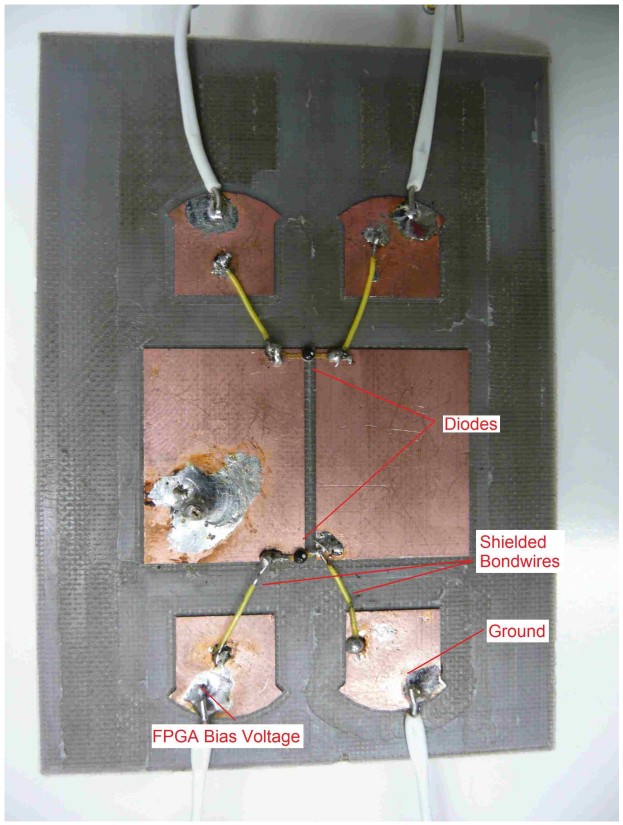

Figure 4.2: Photo of the fabricated Antenna 1.

output lines are then fed through the holes and soldered to the intersection of the capacitor and diode.

4.2 S-Parameters Measurement Set Up

Measuring the S-Parameters was done using an HP 8510C Network Analyzer, an Agilent 85056D Calibration Kit, and an Agilent Test Port cable. Similar equipment is shown in Figure 4.4 and Figure 4.5. A sweep from 1 to 10 GHz was used with a 201 point measurement set-up. Care was taken to minimize reflections from obstacles around the antenna.

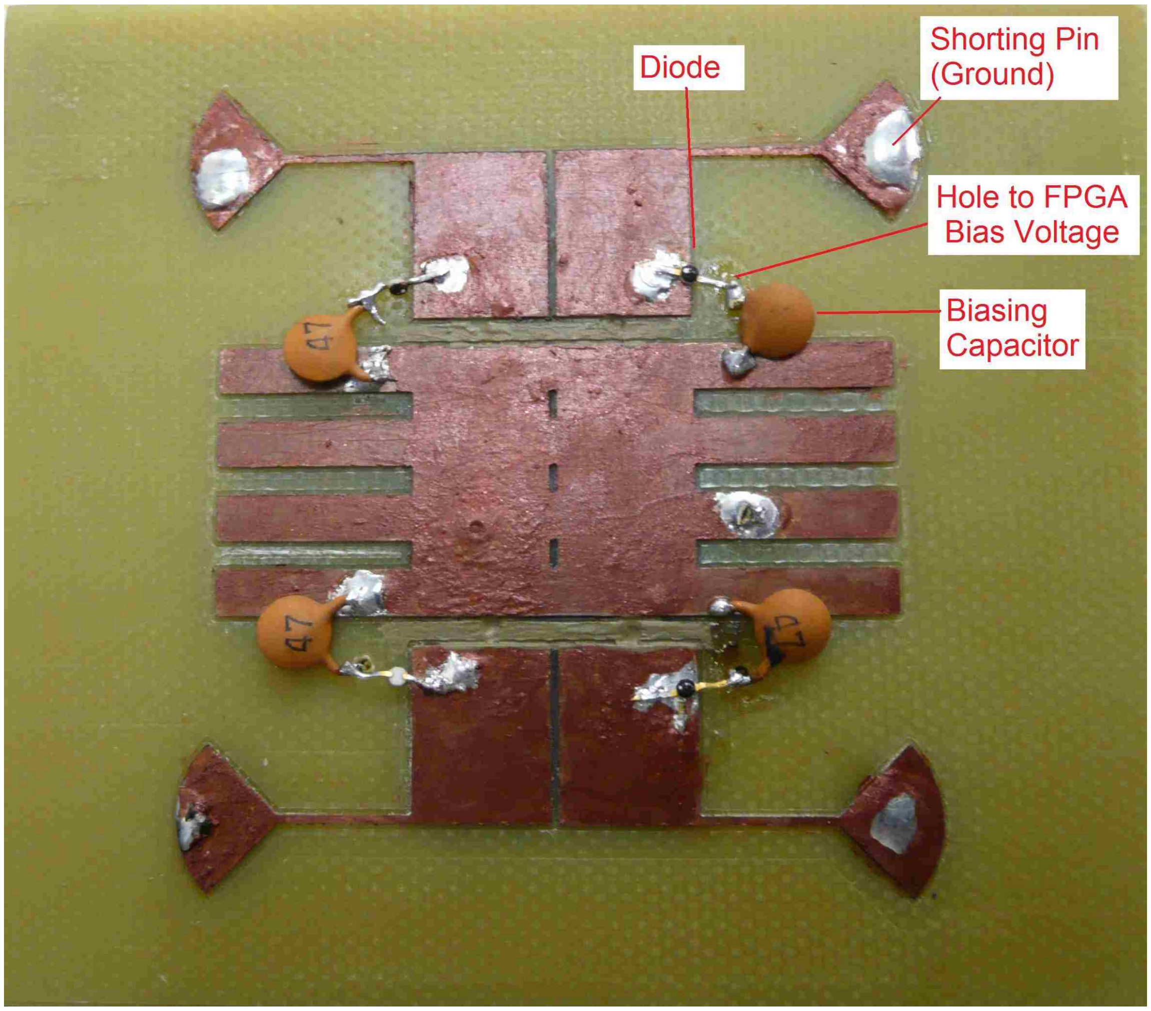

Figure 4.3: Photo of the fabricated Antenna 2.

4.3 Measurements

In order to test the reconfigurable antennas, the prototype antennas are reconfigured using the following set-up:

A Linux computer runs the JTAG software, issuing instructions to a Spartan 3E Xilinx FPGA over a Parallel III cable. The TAP controller programmed on the FPGA translates these instructions and asserts the associated signals to bias the pin

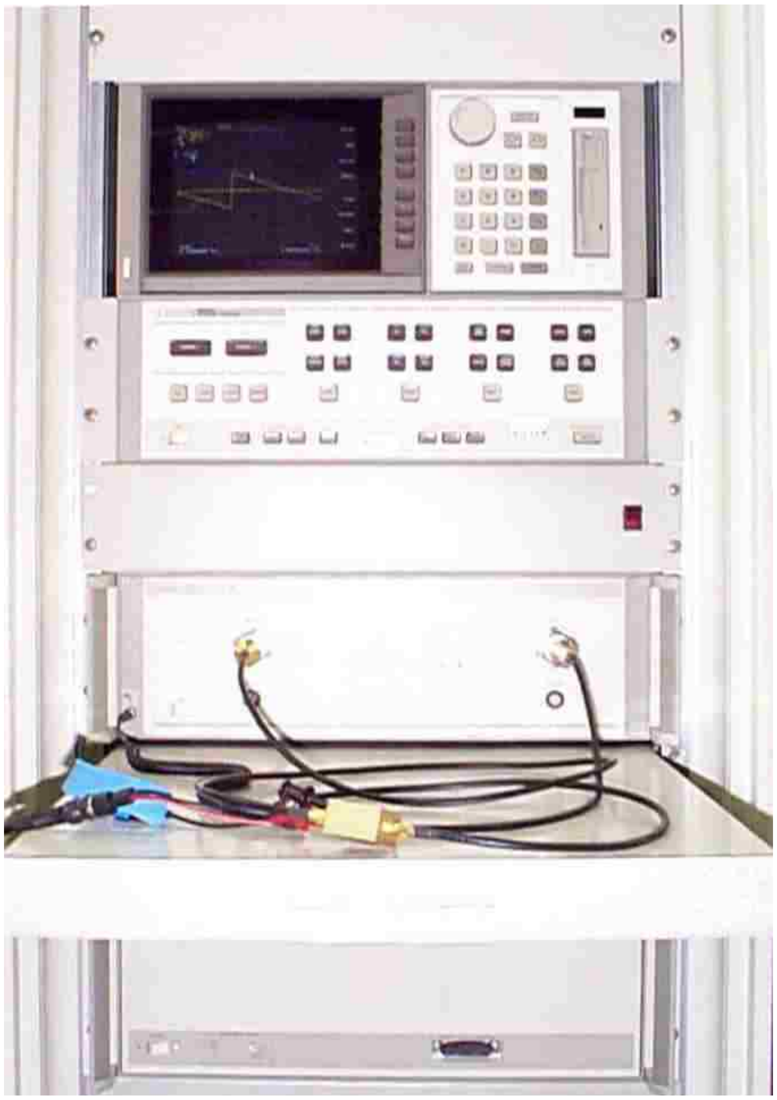

Figure 4.4: HP 8510C Network Analyzer

diodes on the antenna. With one of the pin diodes biased, the RF signal passes from the main patch through the capacitor and that diode to the outer patch.

(a) Agilent 85056D Calibration Kit

(b) Agilent Test Port Cables

Figure 4.5: Agilent test equipment.

4.4 Simulation and Actual Comparison

4.4.1 Antenna 1

The two patches on Antenna 1 are either connected via the two diodes (in parallel) or they are not connected. Figures 4.7 and 4.8 show the comparison between the unbiased 00 case and the biased 11 case.

4.4.2 Antenna 2

Since there are four diodes, sixteen configurations are possible on the antenna. Of these possible sixteen, we will continue to focus on the four cases 0010, 0110, 1010, and 1100. The return loss on these cases were obtained using a network analyzer and are presented in the Figures 4.9, 4.10, 4.11, and 4.12.

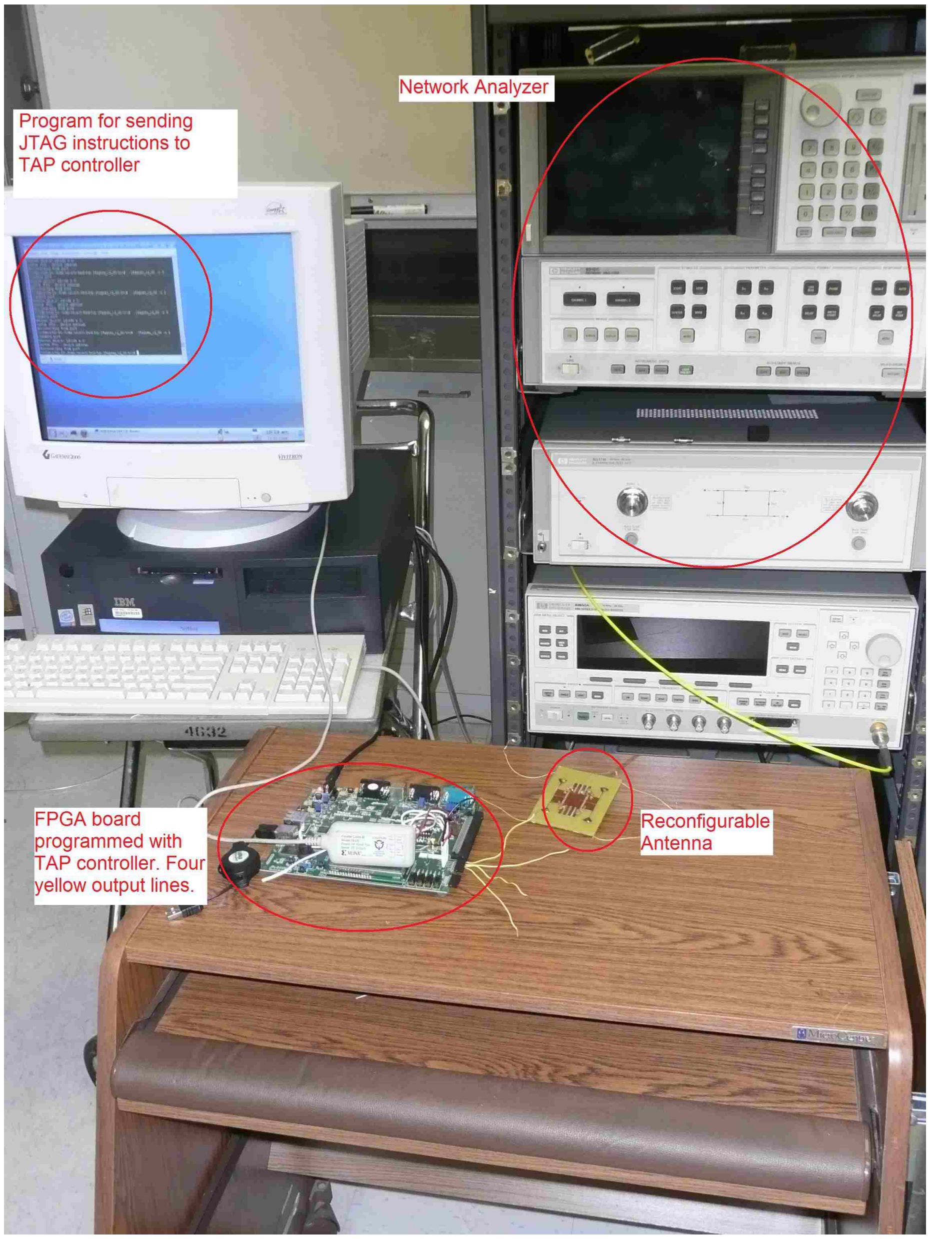
Network Analyzer
Program for sending JTAG instructions to TAP controller
FPGA board programmed with TAP controller. Four yellow output lines.
Reconfigurable Antenna

Figure 4.6: Test Setup

4.5 Actual Comparison

To see the reconfigurability of the two prototype antennas more clearly, it is helpful to compare one case against the other cases. Figure 4.13 is the comparison of actual

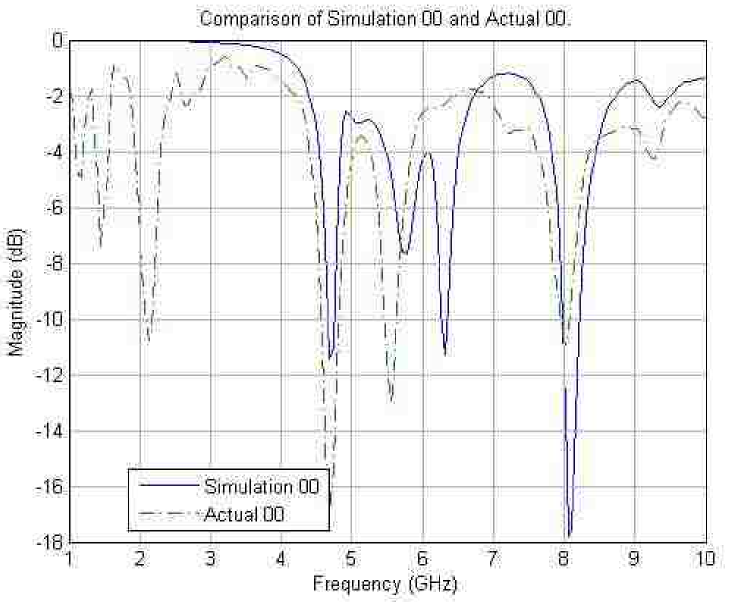

Figure 4.7: Case 00 comparison between simulated and actual.

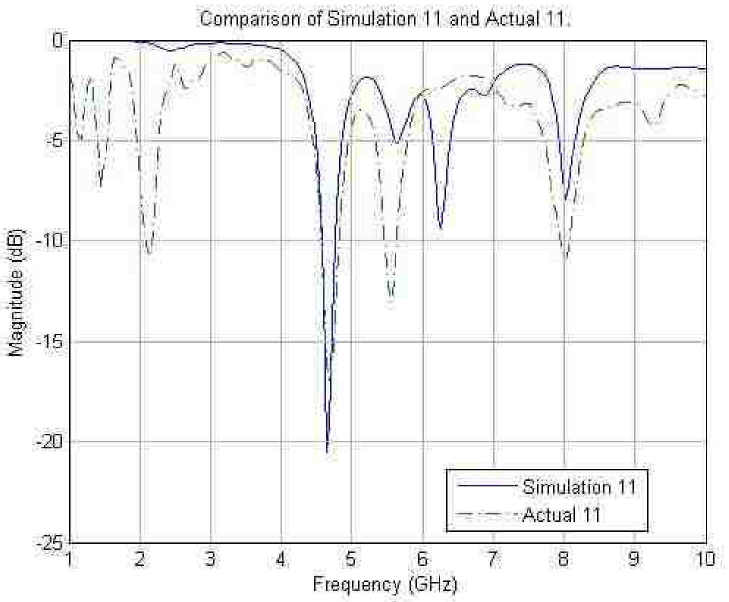

Figure 4.8: Case 11 comparison between simulated and actual.

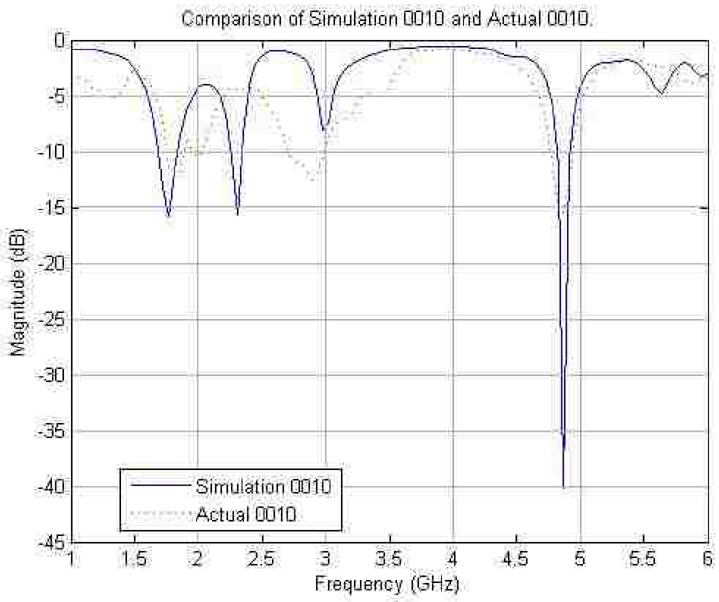

Figure 4.9: Case 0010 comparison between simulated and actual.

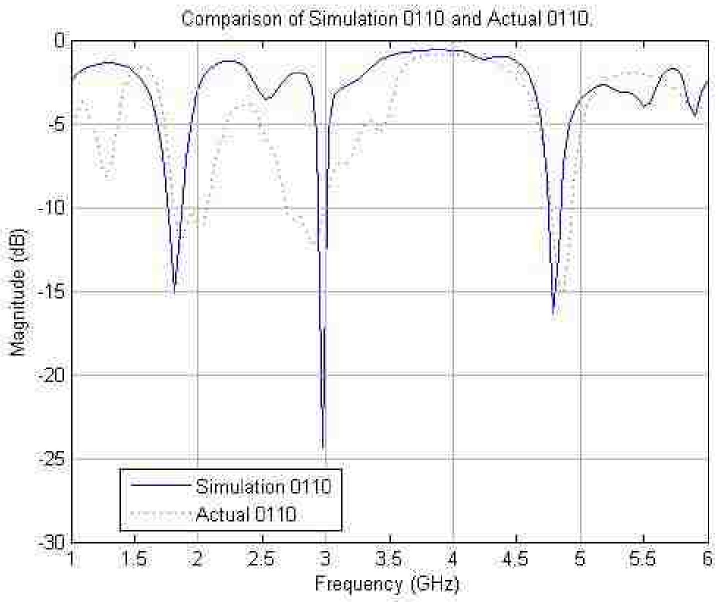

Figure 4.10: Case 0110 comparison between simulated and actual.

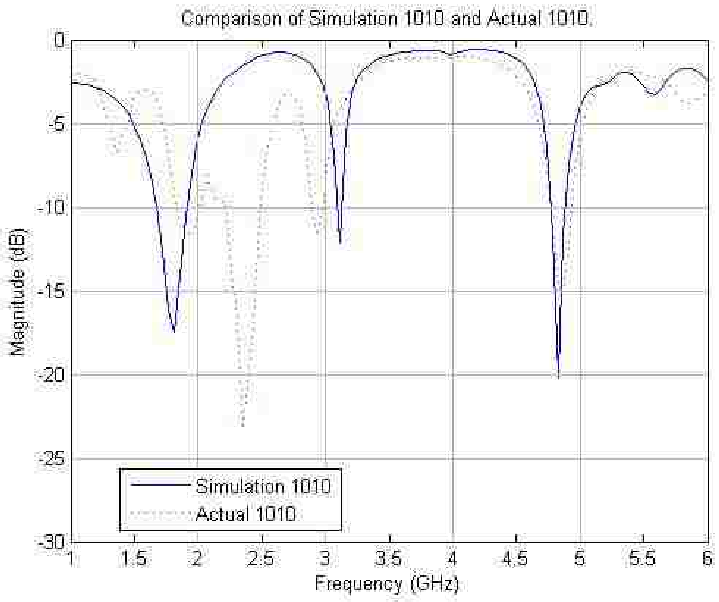

Figure 4.11: Case 1010 comparison between simulated and actual.

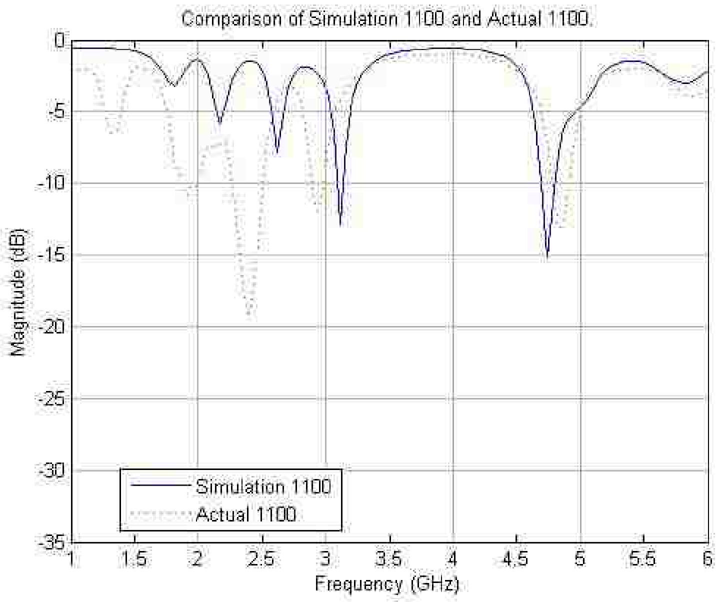

Figure 4.12: Case 1100 comparison between simulated and actual.

data of the cases 00 and 11 on Antenna 1. Figure 4.14 is the comparison of actual data of the four cases 0010, 0110, 1010, and 1100 on Antenna 2.

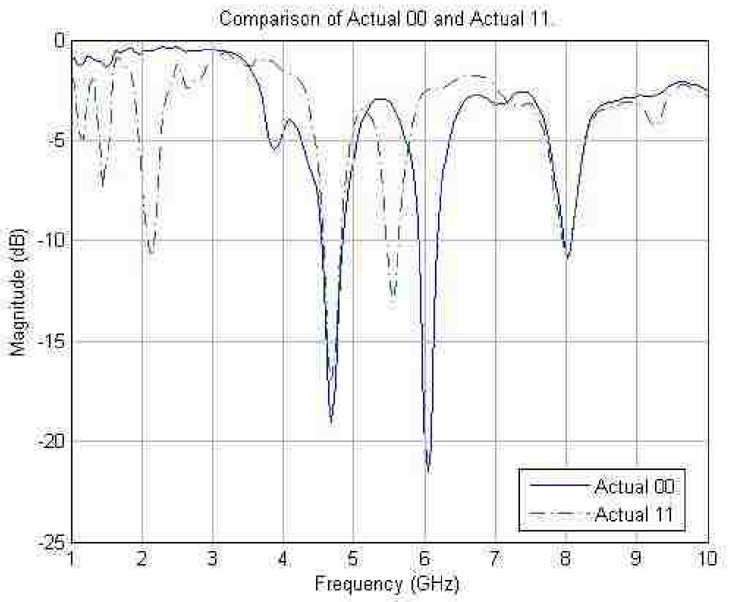

Figure 4.13: Actual data comparison of cases 00 and 11 on Antenna 1.

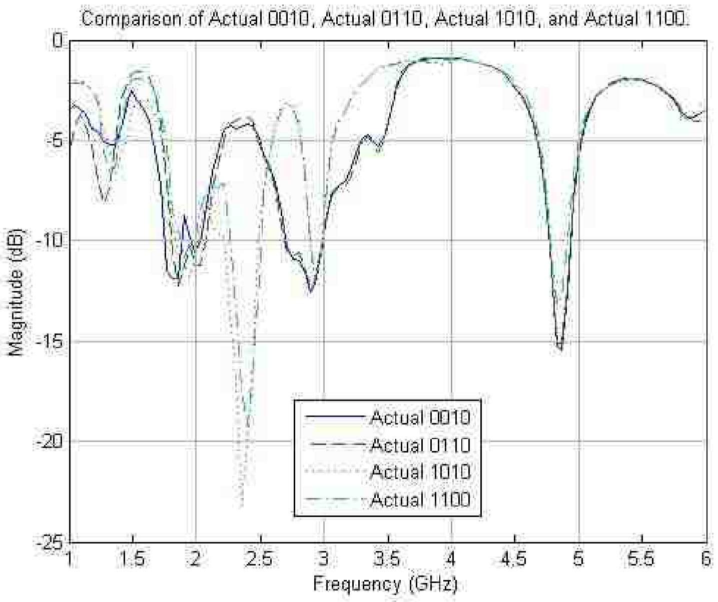

Figure 4.14: Actual data comparison of cases 0010, 0110, 1010, and 1100 on Antenna 2.

Chapter 5

Concluding Remarks

5.1 Significance of the Result

This groundbreaking development in the automated control of reconfigurable microstrip antennas will have a profound effect on future technologies. Antennas that can be controlled in this manner will be implemented on planes, satellites, communication devices, and computational machines. The ability to control an antenna via a computer is now more feasible. Although drawbacks such as low efficiency, low power, and narrow bandwidth still make reconfigurable microstrip antennas less desirable for some applications, the reconfigurable nature and low profile advantages of this type of antenna will make possible the wireless communication systems required today (see Figure 5.1). From the data reviewed and measurements obtained, this concept of reconfigurable antennas controlled via FPGAs is promising.

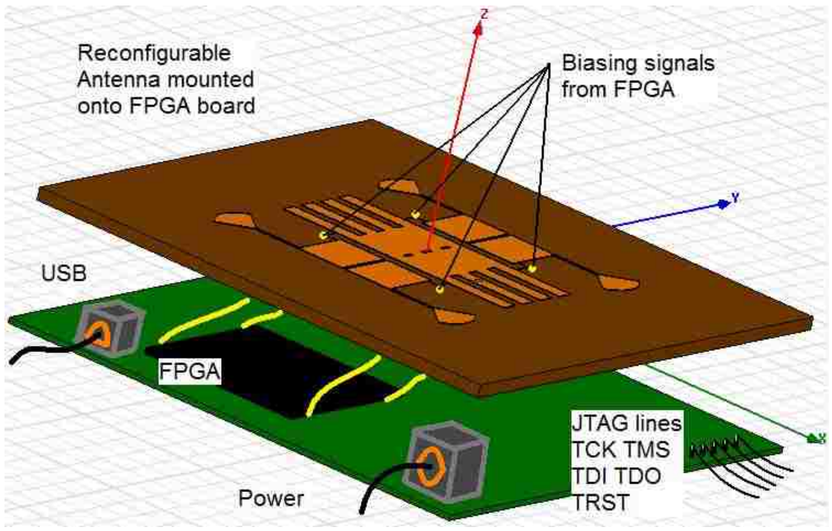
Reconfigurable Antenna mounted onto FPGA board
Biasing signals from FPGA
z
y
x
USB
FPGA
JTAG lines TCK TMS TDI TDO TRST
Power

Figure 5.1: Diagram of Antenna 2 packaged with an FPGA.

5.2 Future Work

Much research and investigation remains to be done in this field. Optimization of the biasing network and more precise soldering will make this type of reconfigurable antenna better.

Further, improved means of controlling the antenna can also be developed. While JTAG TAP controllers can be implemented on FPGAs, other programmable logic devices are suitable for this purpose. TAP controller modules for specific reconfigurable antenna applications could also easily be fabricated in large scale at foundries.

One idea for development is a generalized reconfigurable system which is shown in Figure 5.2. This system is made up of an analog to digital converter element and four reconfigurable elements: the antenna, the low-noise amplifier, IF conversion, and the FPGA. These reconfigurable elements can be managed with internal TAP controllers, which are accessed through a chained JTAG interface.

Graph theory techniques and neural networking algorithms will be instrumental

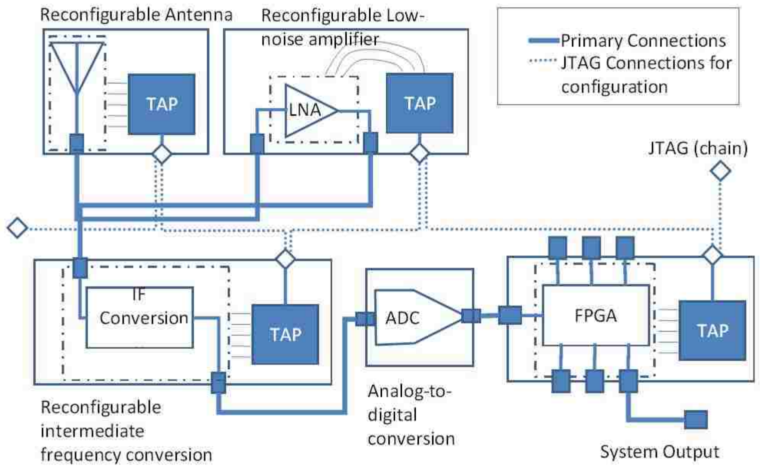
Reconfigurable Antenna
Reconfigurable Low-noise amplifier
Primary Connections
JTAG Connections for configuration
TAP
LNA
TAP
JTAG (chain)
IF Conversion
TAP
ADC
FPGA
TAP
Reconfigurable intermediate frequency conversion
Analog-to-digital conversion
System Output

Figure 5.2: Diagram of a generalized reconfigurable system.

in making reconfigurable antennas ‘smart’ enough to reconfigure themselves to unused bands of the frequency spectrum [25][26]. This will further optimize control of reconfigurable antennas.

Chapter 6

Appendix

`JTAG Computer Program (in attached CD)`

- `jtagseu_v2_00.zip`

  Description : For JTAG instructions to FPGA.

`HFSS Design Files (in attached CD)`

- `final1.hfss`

  Description : Final design of Antenna 1 in HFSS.

- `final1.dxf`

  Description : Final geometry of Antenna 1 in Autocad.

- `final2.hfss`

  Description : Final design of Antenna 2 in HFSS.

- `final2.dxf`

  Description : Final geometry of Antenna 2 in Autocad.

References

[1] F. Yang and Y. Rahmat-Samii, *Patch Antennas with Switchable Slots (PASS) in Wireless Communications: Concepts, Designs, and Applications*, IEEE Antennas and Propagation Mag., vol. 47, no. 2, pp. 13-29, Apr. 2005.

[2] D. E. Anagnostou, *RF-MEMS Reconfigurable Self-Similar Antennas*, Dissertation, UNM, May 2005.

[3] `http://www.ntia.doc.gov/osmhome/allochrt.pdf`

[4] C. A. Balanis, *Antenna Theory - Analysis and Design*, John Wiley and Sons, New York, 1997.

[5] `http://www.astri.org/en/ct_kti03a_3.php`

[6] `http://arstechnica.com/hardware/news/2007/04/intel-shows-off-ultraslim-concept-laptop-at-idf.ars`

[7] `http://www.trlabs.ca/trlabs/technology/technologybulletins/reconfigurableantenna.html`

[8] Pong P. Chu, *RTL Hardware Design Using VHDL*, John Wiley and Sons Inc., Hoboken, NJ, 2006.

[9] IEEE Standard Test Access Port and Boundary-Scan Architecture,

`http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-1990_desc.html`

[10] Neil H. E. Weste and D. Harris, *CMOS VLSI Design*, Pearson Education, Boston, MA, 2005.

[11] Franco Giannini, Robert Sorrentino, Jan Vrba, *Planar Circuit Analysis of Microstrip Radial Stub*, IEEE Transactions on Microwave Theory and Techniques, vol. MTT-32, no. 12, December 1984.

[12] Roberto Sorrentino, Luca Roselli, *A New Simple and Accurate Formula for Microstrip Radial Stub*, IEEE Microwave and Guided Wave Letters, vol. 2, no. 12, December 1992.

[13] H. A. Atwater, *The design of the radial line stub: A useful microstrip circuit element*, Microwave Journal, vol. 28, pp. 149-156, November 1985.

[14] Agilent ADS Manual,

`http://edocs.soco.agilent.com/display/ads2008U1/`
`MRSTUB+(Microstrip+Radial+Stub)`

[15] F. Yang and Y. Rahmat-Samii, *A reconfigurable patch antenna using switchable slots for circular polarization diversity*, IEEE Microwave Wireless Compon. Lett., vol. 12, no. 3, pp. 96-98, Mar. 2002.

[16] F. Yang and Y. Rahmat-Samii, *Patch antenna with switchable slot (PASS): Dual frequency operation*, Microwave Opt. Technol. Lett., vol. 31, no. 3, pp. 165-168, Nov. 2001.

[17] F. Yang and Y. Rahmat-Samii, *Switchable dual-band circularly polarized patch antenna with single feed*, Electron. Lett., vol. 37, no. 16, pp. 1002-1003, Aug. 2001

[18] A. Gopinath and H. Atwater, *Simulation of GaAs PIN Diodes*, IEEE Transactions on Electron Devices, vol. 35, no. 4, April 1988

[19] F. Yang and Y. Rahmat-Samii, *A Novel Patch Antenna With Switchable Slot (PASS): Dual-Frequency Operation With Reversed Circular Polarizations*, IEEE Transactions on Antennas and Propagation, vol. 54, no. 3, March 2006

[20] `http://www.flambda.com/php/stub/stub.php`

[21] Tai-Un Jang, B. Y. Kim, Young-Je Sung, Y. S. Kim, *Square Patch Antenna with Switchable Polarization using Spur-line and PIN Diode*, APMC2005 Proceedings.

[22] `http://www.microsemi.com/datasheets/GC4200_Series.pdf`

[23] Spartan-3E FPGA Starter Kit Board User Guide,

`http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf`

[24] Damien Ressiguier, Joseph Costantine, Youssef Tawk, Christos G. Christodoulou, *A Reconfigurable Multi-Band Microstrip Antenna based on open ended microstrip lines*, IEEE Transactions on Antennas and Propagation, vol. 54, no. 3, March 2006

[25] J. Costantine, C. G. Christodoulou, C. T. Abdallah, S. E. Barbin, *Optimization and Complexity Reduction of Switch-Reconfigured Antennas using Graph Models*, IEEE 2009

[26] J. Costantine, C. G. Christodoulou, *Analyzing Reconfigurable Antenna Structure Redundancy Using Graph Models*, IEEE 978