

---

**FPGA MISSION ASSURANCE CENTER (FMAC)  
SUPPORT ACTIVITY AT THE UNIVERSITY OF  
NEW MEXICO**

**Christos Christodoulou**

**Configurable Space Microsystems Innovations and Applications  
Center (COSMIAC)  
University of New Mexico  
Department of Electrical and Computer Engineering  
1700 Lomas Blvd NE, Suite 2200  
Albuquerque, New Mexico 87131**

**31 October 2013**

**Final Report**

**APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.**



**AIR FORCE RESEARCH LABORATORY  
Directed Energy Directorate  
3550 Aberdeen Ave SE  
AIR FORCE MATERIEL COMMAND  
KIRTLAND AIR FORCE BASE, NM 87117-5776**

---

**Notice Page for Public Release Report**

**NOTICE AND SIGNATURE PAGE**

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 377 ABW Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RD-PS-TR-2013-0047 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//Signed//

---

CASEY A. DERAAD, DR-IV  
Chief, Outreach Technology Branch

//Signed//

---

KEITH A. SHROCK, DR-IV, DAF  
Chief, Mission Planning and Support Division

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

\*Disseminated copies will show “//signature//” stamped or typed above the signature blocks.

# REPORT DOCUMENTATION PAGE

*Form Approved*  
*OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

<b>1. REPORT DATE (DD-MM-YYYY)</b> 31-10-2013		<b>2. REPORT TYPE</b> Final		<b>3. DATES COVERED (From - To)</b> 08/21/2008 to 07/31/2013	
<b>4. TITLE AND SUBTITLE</b> FGPA Mission Assurance Center (FMAC) Support Activity at the University of New Mexico				<b>5a. CONTRACT NUMBER</b> FA9453-08-2-0259	
				<b>5b. GRANT NUMBER</b>	
				<b>5c. PROGRAM ELEMENT NUMBER</b>	
<b>6. AUTHOR(S)</b> Christos Christodoulou				<b>5d. PROJECT NUMBER</b>	
				<b>5e. TASK NUMBER</b>	
				<b>5f. WORK UNIT NUMBER</b> V033	
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Configurable Space Microsystems Innovations and Applications Center (COSMIAC) University of New Mexico Department of Electrical and Computer Engineering 1700 Lomas Blvd NE, Suite 2200 Albuquerque, New Mexico 87131				<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> Air Force Research Laboratory 3550 Aberdeen Avenue, SE Kirtland Air Force Base, New Mexico 87117- 5776				<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b> AFRL/RDMX	
				<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b> AFRL-RD-PS-TR-2013-0047	
<b>12. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release; distribution unlimited.					
<b>13. SUPPLEMENTARY NOTES</b> 377ABW-2014-0345 Government Purpose Rights					
<b>14. ABSTRACT</b> This report covers the period of performance of the FPGA Mission Assurance Center (FMAC) Cooperative Agreement at the University of New Mexico (UNM). The organization has changed its name from the FMAC to the Configurable Space Microsystems Innovations and Applications Center (COSMIAC) to better reflect the work that is being accomplished. The main emphasis of this work was to develop and implement new principles related to reconfigurable electronics as applied to the space environment. Issues such as radiation hardening, complexity, reliability and Space Plug-n-Play are presented and discussed in detail. This report includes theory, analysis, design and implementation of several reconfigurable devices that can be used on a space vehicle or platform. Several recommendations and future ideas are presented and discussed in detail.					
<b>15. SUBJECT TERMS</b> FPGAs, Reconfigurable electronics, cubesats, adaptive wiring, complexity, reliability					
<b>16. SECURITY CLASSIFICATION OF:</b> Unclassified			<b>17. LIMITATION OF ABSTRACT</b>  SAR	<b>18. NUMBER OF PAGES</b>  169	<b>19a. NAME OF RESPONSIBLE PERSON</b> Casey DeRaad
<b>a. REPORT</b>  U	<b>b. ABSTRACT</b>  U	<b>c. THIS PAGE</b>  U			<b>19b. TELEPHONE NUMBER (include area)</b>

**This page intentionally left blank**

# 1. Table of Contents

Summary.....	1
Introduction.....	1
<b>1. Methods, Assumptions and Procedures: Establishing, verifying, and demonstrating open design guidelines for best-practice application of FPGAs and other configurable devices in space and defense systems.....</b>	<b>3</b>
1.1 AFRL’s OpenASIM.....	3
1.2 Hardware & Software development.....	8
1.1.1 Support for latest FPGA devices .....	9
1.3 Support for new development platforms .....	12
1.4 Creation of simulation framework and verification suite .....	13
1.5 New hardware features to support ASIM requirements.....	15
1.6 SpaceWire integration.....	15
1.7 I <sup>2</sup> C integration .....	20
1.8 Creation of ASIM core libraries for SSM support .....	20
1.9 Framework for code documentation .....	22
1.10 Streamlined porting of OpenASIM to other platforms.....	23
<b>2. Improving the reliability of FPGAs and other configurable devices in space and defense systems including operation in demanding environments such as the radiation environment of space .....</b>	<b>25</b>
2.1 FPGA-based Dynamically Reconfigurable Systems.....	25
2.1.1 A Dynamic Dual Fixed-Point Arithmetic Architecture for FPGA .....	28
2.1.2 Conclusions.....	42
2.2 A DPR capable space-borne processor with SEU mitigation .....	43
2.2.1 Scrubbing an FPGA’s configuration memory for SEU mitigation .....	43
2.2.2 Space-borne processor architecture .....	44
2.2.3 LEON3FT .....	44
2.2.4 OpenRISC-FT.....	45
2.2.5 COSMIAC’s Alternative space-borne processor: DR-OpenRISC .....	45
2.2.6 Enabling DPR on OpenRISC.....	46
2.3 Additional Board Development.....	47
2.3.1 Radiation Testing .....	48
<b>3. Adapting, augmenting, verifying, and demonstrating commercially available FPGA and configurable systems design tools to accommodate unique space and defense requirements, including the verification of correct design, and ensuring of high system-level productivity.....</b>	<b>50</b>
3.1 FPGA-based Reconfigurable Adaptive Wiring.....	50
3.1.1 Adaptive Wiring Panel (AWP).....	50
3.1.2 Conceptual Architecture .....	50
3.1.3 Example of Adaptive Panel Design and Implementation .....	52
3.1.4 The AWP Cell Unit.....	54
<b>Cell Management Unit .....</b>	<b>57</b>
3.1.5 Module .....	57
3.1.6 Prototype Demonstration .....	59

3.2	Space Plug-and-play Architecture (SPA).....	59
3.3	Commercial High Speed Digital Test Structures.....	60
3.4	New Materials Developed for Advanced Reconfigurability .....	61

**4. Results for Assessing the future development of reconfigurable electronics, devices, and architectures and their potential impact on overall system performance and reliability and planning for future space and defense needs.....62**

4.1	Development of Configurable Electronics and Cubesats.....	62
-----	---	----

**In this section we show what COSMIAC has developed in the last two years, under this agreement, in terms of cubesat development, related instrumentation, and the ground station developed to track and download satellite mission data for UNM, AFRL and the University Nanosat Program’s (UNP’s) FASTRAC satellites ..... 62**

4.1.1	<i>Trailblazer</i> .....	62
4.1.2	<i>ORS Squared</i> .....	65
4.2	Additive Manufacturing .....	66
4.3	Software Defined Radio (SDR).....	67
4.4	Langmuir Probe Design .....	68
4.5	Global Educational Network for Satellite Operations (GENSO) .....	70
4.6	The Use of FPGAs to Control Reconfigurable Antennas .....	70
4.6.1	<i>Reconfigurable Antenna Structure, Design and Tuning</i> .....	70
4.6.2	<i>Reconfigurable Antenna Fabrication, Measurement and FPGA control</i> .....	74
4.7	Using Neural Networks embedded in an FPGA to achieve Device Reconfiguration .....	78
This section gives a basic theoretical background of Neural Networks (NN), it shows how these Neural Networks can be embedded on FPGAs, and how reconfigurable systems can be controlled by an FPGA.		
.....		78
4.7.1	<i>Basic Review of Artificial Neural Networks</i> .....	78
4.7.2	<i>NN-FPGA Controller Design</i> .....	83
4.7.3	<i>Simulations and Results</i> .....	91
4.7.4	<i>Conclusion and future work</i> .....	105

**5. Results for developing and demonstrating performance-level and mission reliability models that will permit quick assessment of complex electronic architectures for space and defense systems..... 106**

5.1	Graph Modeling.....	106
5.1.1	<i>Graph Modeling of Reconfigurable Antenna Arrays</i> .....	107
5.2	Correlation Between Complexity and Reliability .....	111
5.2.1	<i>The General Complexity of reconfigurable Antenna arrays</i> .....	111
5.2.2	<i>Example 1</i> .....	112
5.2.3	<i>Example 2</i> .....	113
5.2.4	<i>Example 3</i> .....	113
5.3	Prioritization of Frequency Dependent Configurations and Antenna Performance. ....	116
5.3.1	<i>Example 4</i> .....	117
5.4	Practical and Design Aspects .....	120

**6. Facilitating the exchange of information between interested organizations on the use of configurable devices and presenting results of the above research and development at technical conferences and meetings ..... 121**

**7. Conclusions ..... 125**

<b>8</b>	<b>References.....</b>	<b>126</b>
<b>9</b>	<b>Key Personnel .....</b>	<b>129</b>
<b>10</b>	<b>List of Publications .....</b>	<b>131</b>

## Index of Figures

Figure 1. Generation 1 ASIM's block diagram.....	4
Figure 2. Current state of OpenASIM .....	8
Figure 3. Code for a Virtex 4 memory structure .....	9
Figure 4. Code for a Virtex 5 memory structure .....	9
Figure 5. Memory collision detected during simulation. This error implies a hardware malfunction and required analysis.....	10
Figure 6. Original code for memory generation the collision error .....	10
Figure 7. Apparent fix for the collision error. Cause still not resolved .....	10
Figure 8. Final error detection.....	10
Figure 9. Clock manager code for Virtex 5.....	11
Figure 10. Clock manager code for Virtex 7 .....	11
Figure 11. Undefined signals inserted during simulation .....	11
Figure 12. Same simulation after reset states were correctly introduced in simulation code.....	12
Figure 13. Differences between the ML505 and the Data Design Gen2 hardware platforms .....	13
Figure 14. Newly defined simulation framework .....	14
Figure 15. Original MinSOC console output of the simulation testing the system's UART .....	15
Figure 16. Console output for simulation testing the system's UART using the newly created simulation framework. ....	15
Figure 17. NASA's Goddard Space Flight Center SpaceWire core.....	16
Figure 18. SpaceWire Light block diagram. ....	17
Figure 19. Simulation of a bus transaction on the original adaptation of the core. The red circle with the number 1 represents the action of sending a word to the TX register in the core. Number2 is a control operation to set a txwrite flag high in order to trigger the actual transmission. Number 3 sets the same flag low to allow a new transaction to take place. The overall operation took ~1920 ns @ 40ns clock period .....	17
Figure 20. Simulation of a bus transaction on the improved adaptation of the core. The red circle with the number 1 represents the action of sending a word to the TX register in the core. Number 2 shows txwrite flag being automatically generated. The total time this transaction takes is ~640ns @ 40ns clock period (1/3 original latency). ....	17
Figure 21. SpaceWire Analyzer's Graphical User Interface snapshot showing traffic.....	18
Figure 22. SpaceWire Analyzer in Loop-back mode [7].....	18
Figure 23. SpaceWire brick architecture .....	19
Figure 24. COSMIAC's setup to test SpaceWire integration.....	19
Figure 25. Interruption handler routine.....	19
Figure 26. I2C core block diagram. ....	20
Figure 27. ASIM Core libraries Layered Architecture.....	21
Figure 28. Snapshot for I2C drivers' documentation .....	23
Figure 29. PERL application developed to centralize the generation of configuration files needed to port the OpenASIM design to different FPGA-based hardware platforms.....	24
Figure 30. F+V computer prototype [8].....	26
Figure 31. FPGA architecture's block diagram.....	27
Figure 32. Evolution of Xilinx's FPGAs in terms of density. First green line shows the time where the tools supporting DPR in a beta version first appear. Second green line shows the time when they were finally commercially supported. ....	28
Figure 33. Virtex 4 family partial reconfiguration times. Points were calculated using the bitstream sizes reported in the device's datasheets and a theoretical maximum reconfigurable speed of 3.2 Gbits/sec [16] .....	30
Figure 34. Precision variation curve for equation 2.1.....	31
Figure 35. DDFX Adder .....	33
Figure 36. DDFX Multiplier.....	35
Figure 37. Multi-DDFX core performance for Virtex II Pro MicroBlaze (a) and Virtex 4 PowerPC405 (b).....	37
Figure 38. Performance results for a single DDFX core versus reconfiguration frequency for Virtex II Pro MicroBlaze (a) and Virtex 4 PowerPC405 (b). Here, 0% can not be represented in our logarithmic scale, thus it was included as	



the leftmost point in the graph .....	38
Figure 39. Performance results for 4 DDFX cores versus reconfiguration frequency for Virtex II Pro MicroBlaze (a) and Virtex 4 PowerPC405 (b). Here, 0% cannot be represented in our logarithmic scale, thus it was included as the leftmost point in the graph .....	39
Figure 40. Dynamic power consumption for Virtex II Pro (a) and Virtex 4 (b) .....	40
Figure 41. Virtex II Pro and Virtex 4 static power consumption and interpolated power consumption of commonly used peripherals in embedded systems. The red circles and blue squares represent data points obtained through direct measurements .....	41
Figure 42. Virtex II Pro (a) and Virtex 4 (b) total power consumption (dynamic + static).....	42
Figure 43. AAC Microtec’s OpenRISC-FT .....	45
Figure 44. Initial setup of the processor used for UNM to build a space version.....	46
Figure 45. WBICAP’s block diagram .....	47
Figure 46. Development board for the Radiation Hazard Awareness Sensor .....	48
Figure 47. Basic concept of an adaptive wiring cell. (a) Substrate (yellow cloud) having a number of inputs and outputs defining a ‘wiring problem’. (b) Depiction of a possible wiring solution .....	51
Figure 48. A physical wiring problem example. (a) Unprogrammed substrate, containing four sockets for components (modules). (b) Example placement of two modules and wiring of a two-net netlist .....	51
Figure 49. Extended adaptive wiring system by using two cells.....	52
Figure 50. Adaptive Wiring Panel (AWP) concept using 64 cells in a 8x8 array .....	53
Figure 51. Partial adaptive wiring panel (AWP) system, containing six cells and two modules. (top left) Graphical user interface (GUI) snapshot used to set-up the netlist. (bottom left) Top view of the AWP. (right) Side view of demonstration system revealing the .....	54
Figure 52. Cell unit logical architecture .....	55
Figure 53. Cell unit of the AWP .....	56
Figure 54. Block diagram of the hardware design in each FPGA cell unit. It consists of a main control unit, memory blocks, and I2C blocks.....	57
Figure 55. AWP in use. (a) Modules are placed. (b) Connection details revealed. (c) Virtual wire formation. (d) Connection to redundant module.....	58
Figure 56. AWP with the circuit connected using two modules. The bottom module has a battery and the top module with a resistor and a LED .....	59
Figure 57. ORS Squared Development with VSI .....	60
Figure 58. The Trailblazer Satellite.....	63
Figure 59. Environmental Testing of Trailblazer .....	63
Figure 60. Thermal or Vibrational Testing Configuration .....	64
Figure 61. ORS2 Satellite .....	65
Figure 62. ORS Squared for 3D Printing .....	66
Figure 63. 6U Prototype 3D Printed Structure .....	67
Figure 64. Langmuir Probe Design.....	68
Figure 65. Data Design Nano CDH and Langmuir Probe on the AFRL vibration table.....	69
Figure 66. Z-axis vibration testing.....	69
Figure 67. Langmuir Probe in Plasma Chamber.....	69
Figure 68. Schematic representation of the reconfigurable antenna .....	71
Figure 69. Antenna resonance for different diodes states 0- Diode OFF, 1 Diode ON.....	73
Figure 70. Simulated 3-D radiation pattern at 4.875 GHz.....	73
Figure 71. E and H plane cuts at 4.875 GHz.....	74
Figure 72. The fabricated prototype.....	75
Figure 73. The parallel III cable with FPGA board.....	76
Figure 74. The S11 measurement setup .....	76
Figure 75. A comparison between actual measurements and simulation for an antenna state .....	77
Figure 76. A comparison between actual measurements and simulations for an antenna state.....	77
Figure 77. Entire reconfigurable antenna system .....	78
Figure 78. Biological Neural connections [29].....	79
Figure 79. Artificial neural network topology .....	79

Figure 80. A Neuron model.....	81
Figure 81. Sigmoid function segments.....	85
Figure 82. Segment 1 .....	85
Figure 83. Segment 2 .....	86
Figure 84. Segment 3 function.....	86
Figure 85. Segment 4 function.....	87
Figure 86. Segment 5 .....	87
Figure 87. Sigmoid Simulink model .....	88
Figure 88. System Generator block.....	89
Figure 89. A NN-FPGA controller hardware architecture .....	90
Figure 90. FPGA board connections where The FPGA board output is connected to the antenna input.....	90
Figure 91. Filtenna antenna structure .....	92
Figure 92. Simulink NN_FPGA model of the antenna.....	93
Figure 93. Hidden layer of filtenna .....	94
Figure 94. Output layer of filtenna .....	95
Figure 95. Output neuron structure.....	95
Figure 96. NN output vs measured antenna response.....	96
Figure 97. Antenna Structure .....	97
Figure 98. Neural network representation for this antenna.....	97
Figure 99. NN output vs measured antenna response for shape 1 .....	98
Figure 100. NN output vs measured antenna response for shape 2.....	99
Figure 101. NN output vs measured antenna response for shape 3.....	99
Figure 102. Broadband Tapered Slot Antenna.....	100
Figure 103. The 3 models used for modeling .....	101
Figure 104. NN output vs measured antenna response for Model 1 .....	101
Figure 105. NN output vs measured antenna response for Model 2 .....	102
Figure 106. NN output vs measured antenna response for Model 3 .....	102
Figure 107. Star antenna structure.....	103
Figure 108. A NN model for the Star antenna.....	103
Figure 109. NN output (red) vs measured antenna response (blue) and determination of number of hidden neurons .....	104
Figure 110. NN output vs measured antenna response for case 111000 .....	105
Figure 111. NN output vs measured antenna response for case 110000 .....	105
Figure 112. A two element reconfigurable antenna array with dimensions in "mm" .....	107
Figure 113. The return loss for the array in Figure 3 for three different switch configurations. S3=S21S'21 .....	108
Figure 114. The Fabricated antenna array prototype .....	108
Figure 115. A Comparison between the measured and simulated reflection coefficient when S1 is activated .....	109
Figure 116. General form of a graph model for the antenna array.....	110
Figure 117. Simplified graph model for the array antenna .....	110

## Index of Tables

Table 1. Summary of currently known ASIMs and their more significant features.....	4
Table 2. AFRL's general requirements for OpenASIM.....	5
Table 3. Antenna Dimensions.....	72
Table 4. Neural network parameters for Filtenna antenna.....	92
Table 5. NN parameters of Rotatable Microstrip Antenna .....	98
Table 6. NN parameters of Broadband Tapered Slot Antenna.....	100
Table 7. The star antenna neural network parameters .....	104
Table 8. Equivalent graphs representing equivalent configurations at 2.205 GHz .....	111

## Summary

This report covers the period of performance of the FPGA Mission Assurance Center (FMAC) Cooperative Agreement at the University of New Mexico (UNM). The organization has changed its name from the FMAC to the Configurable Space Microsystems Innovations and Applications Center (COSMIAC) to better reflect the work that is being accomplished. The main emphasis of this work was to develop and implement new principles related to reconfigurable electronics as applied to the space environment. Issues such as radiation hardening, complexity, reliability and Space Plug-n-Play are presented and discussed in detail. This report includes theory, analysis, design and implementation of several reconfigurable devices that can be used on a space vehicle or platform. Several recommendations and future ideas are presented and discussed in detail.

## Introduction

This report completes the period of performance of the Cooperative Agreement entitled “FPGA Mission Assurance Center (FMAC) Support Activity at the University of New Mexico (UNM)”. The UNM organization that performed the work was originally called the Field Programmable Gate Array (FPGA) Mission Assurance Center but changed its name to the Configurable Space Microsystems Innovations and Applications Center (COSMIAC). This change better reflected the work that was accomplished under the revised statement of work issued on September 8, 2011 and new work being performed under separate sponsorship. COSMIAC is the Tier-2 Research Center at UNM’s School of Engineering.

COSMIAC was devised as a way of adding value to the Air Force and the nation at large by ensuring the success of FPGAs and reconfigurable systems in a variety of application areas. Particular emphasis was placed on the key application areas for FPGAs that include space environment and its unique reliability requirements, high-performance embedded processing, and system-on-a-chip designs. COSMIAC brings together New Mexico participants from across Industry, Academia and government laboratories in order to assure the greatest benefit from FPGAs and other reconfigurable systems to the U.S. military and the nation as a whole.

The development of the next generation responsive systems can significantly benefit from the use of real-time reconfigurable systems. Such systems form the next generation FPGAs, or plug-and-play electronic components, allowing production of scalable components that optimize performance in terms of both speed and energy consumption (or power). Although the initial work was focused on FPGAs, other systems were studied under this contract. Reconfigurability became an important desired feature of modern, agile, radio frequency (RF) systems for wireless and satellite communications, sensing, and imaging. There is a shift towards incorporating smart and agile RF devices that both sense the surrounding RF environment and communicate at the same time in any contested/congested environment. These concepts significantly reduce the number of components and thus hardware complexity and cost compared to today’s technology, which relies on inflexible hardware. FPGAs and Microcontrollers are particularly useful and cost effective for military systems, particularly in space where a few parts can be qualified and then reused for many purposes such as space applications.

The scope of this work was to conduct and manage direct research activities on the use of

FPGAs and other reconfigurable systems in space systems, with emphasis on: (a) establishing open design guidelines for best-practice application of FPGAs in space and defense systems, (b) improving the reliability of FPGAs and reconfigurable systems, (c) adapting and augmenting commercially available FPGA design tools to accommodate unique space and defense requirements, (d) verification of correct design, (e) high system-level productivity assurance, (f) assessment of the future development of reconfigurable electronics and architectures, and (g) planning for future space and defense requirements. Additionally, COSMIAC was tasked to facilitate the exchange of information between interested organizations on the use of configurable devices and present results of research and development at technical conferences and meetings.

This report covers six primary scope topics that are presented individually that are addressed in detail, specifically work on FPGAs, analog systems, RF, antennas, adaptive wiring, and the design and testing of several devices to be fielded in space. The COSMIAC team included researchers from many different areas, including engineering, physics and biology.

# 1. Methods, Assumptions and Procedures: Establishing, verifying, and demonstrating open design guidelines for best-practice application of FPGAs and other configurable devices in space and defense systems

This section describes the work done on hardware and software development to support the latest FPGA devices. The goal was to create the concepts of plug-and-play used in the computer industry and apply them to the concepts to satellite design. The plug-and-play architecture developed by AFRL to address this challenge is called Monarch, for “modular open network architecture”. The architecture is open, not proprietary, and uses publicly available interfaces akin to the USB and Ethernet standards found in computers.

## 1.1 AFRL’s OpenASIM

Nine years ago the Air Force Research Laboratory (AFRL) set out to study the concepts of plug-and-play used in the computer industry to apply them to satellite design. The goal was to reduce the time needed to build a working satellite from years to only 6 days using Monarch. With Monarch, a spacecraft’s components can be thought of as ‘black boxes’. These components, all of which have standard connectors, can be easily combined to form a Monarch system. A larger, more complex spacecraft would have more black boxes than a smaller, simpler one. Monarch allows for three classes of black boxes: (1) *endpoints*, which are components that perform a function, such as thermometers, cameras, and radios, (2) *routers*, which connect two or more components, and (3) *hosts*, which are comparable to the central processing unit of a PC [1].

The connectivity between components in a Monarch network is done through a variety of standards and a defined protocol. In cases where designers want to integrate a legacy sensor or actuator into a Monarch network, a specialized interface needs to be added to it. Such an interface is called *Appliqué Sensor Interface Module (ASIM)*. ASIMs serve as a bridge between a module’s native communication channel and a Monarch network. ASIM’s generally consist of a microprocessor with several communication channel alternatives and general purpose input/output ports. Figure 1 depicts a generation 1 ASIM block diagram.

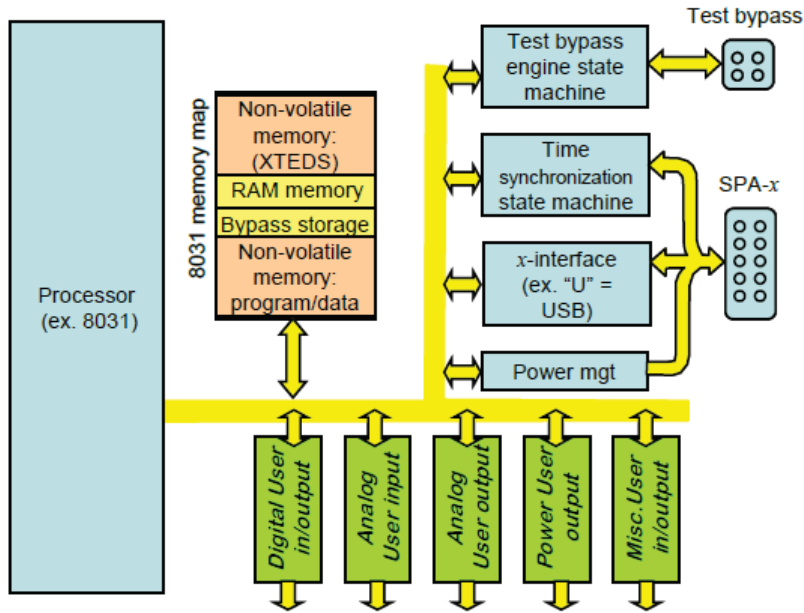


Figure 1. Generation 1 ASIM's block diagram

There exist several ASIMs reported in the literature and known within the scientific community [2][3]. All of the current ASIMs are proprietary, costly and closed designs. Each has a different development tool chain to incorporate different processors. Table 1 summarizes currently known ASIMs and their more significant features.

Table 1. Summary of currently known ASIMs and their more significant features

	CPU	FP G A	A S I C	I 2 C	G P I O	U A R T	S P W	W D G	T B P	E T H	U S B	NV	RAM	CLK	Software
NanoRTU	PIC16	√	√	√	√	√		√				128KB	4K x 14 bit	16MHz	Proprietary
MicroRTU	Open RISC	√	√	√	√	√	√	√		√	√	8Gb	54Mb		GCC
PnPGen3	8051	√		√	√	√	√	√	√		√	48Kb	8Kb	50MHz	Keil

<b>DataDesign</b>	8051	√		√	√	√	√	√	√	√	√	64kb		48MHz	Keil
<b>MiniPnP</b>	PIC16		√	√	√	√		√			√			50MHz	Keil
<b>ASIM</b>	8051		√	√	√	√	√	√	√		√		32Kb	50MHz	Keil Custom C
<b>OpenASIM</b>	Open RIS C	√	√	√	√	√	√		√			Up to 1Gb	Up to 1Gb	100MHz	GCC

Although functional and standard compliant, ASIMs can differ significantly over their wide range of features. AFRL recognized the need for an open source ASIM design that could be shared at no cost and modified as needed. The task of creating such a design was assigned to COSMIAC under this agreement. OpenASIM is the result of COSMIAC’s efforts over the last year. Table 2 defines AFRL’s general requirements for OpenASIM.

**Table 2. AFRL’s general requirements for OpenASIM**

<b>Category</b>	<b>Description</b>	<b>Goal</b>
Memory	OpenASIM shall provide the capability to address up to 2GB of instruction memory.	2GB
Memory	Up to 64kB of instruction memory should be addressable internally (memory implemented on the chip). The actual size should be defined by a parameter the user can change.	64kB
Memory	OpenASIM shall provide the capability to address up to 1GB of data memory.	1GB
Memory	Up to 64kB of data memory should be addressable internally (memory implemented on the chip). The actual size should be defined by a parameter the user can change.	64kB

Memory	Data cache should be parameterizable between 0 and 8kB.	8kB
Memory	Instruction cache should be parameterizable between 0 and 8kB.	8kB
Memory	Non-volatile storage should be available to store application code and/or xTEDs data.	1GB
GPIO	OpenASIM shall provide 32 general digital I/O channels, whose direction should be individually controlled by the microprocessor.	32
I2C	OpenASIM shall have an I2C master peripheral attached to its internal bus.	-
SpW	OpenASIM shall have an SpW peripheral attached to its internal bus, capable of operation at least 10Mbps.	10M bps
SpW	The SpW peripheral should be able to reset the microprocessor upon command, independently of the microprocessor operations and state.	-
SpW	We shall seek to avoid the need of using an external converter or PHY.	-
<b>Category</b>	<b>Description</b>	<b>Goal</b>
RS232	The OpenASIM shall provide a parameterizable number of asynchronous serial ports, between 1 and 4. Serial ports shall use eight data bits, 1 stop bit and no parity bits (8N1).	4
RS232	OpenASIM shall support a minimum of 9.6kbauds and a maximum of 115.2kbauds.	115.2k
RS232	Serial ports shall provide a buffer of at least 128 bytes.	128
Ethernet	OpenASIM shall include an Ethernet core in its internal bus. (Many features still uncertain, this is a complex core.)	-
Timers	The OpenASIM shall provide a parameterizable number of hardware timers (between 0 and 4) capable of being programmed to trigger at a specified rate.	4



CRC	The OpenASIM shall have a CRC hardware calculator in its internal bus.	1
CRC	The CRC implemented shall be CCITT X.25 polynomial.	-
PWM	The OpenASIM shall provide a parameterizable number of PWM cores (between 0 and 4) attached to its internal bus.	4
PWM	PWM outputs should be programmable in duty cycle from 0 to 100%.	-
Debugger	The OpenASIM shall provide ways to perform software debugging, including step by step execution and the ability to use break points.	-
Time_keeping	The OpenASIM shall keep time from a local oscillator. Time shall be available to the microprocessor from a memory mapped location composed of 4 bytes seconds & 4 bytes microseconds.	-
Time_keeping	The OpenASIM shall be capable of receiving a 1 Hz signal. This signal shall be used to load a new time into the time register.	-
Watchdog	The OpenASIM shall incorporate a watchdog timer, which will reset the microprocessor upon expiration or trigger an interruption.	-

OpenASIM is an OpenRISC based system with I<sup>2</sup>C and Spacewire interfaces to serve as an adaptors for non-SPA compatible sensors/actuators to connect into a MONARCH network. Figure 2 depicts the block diagram of the system in its current state.

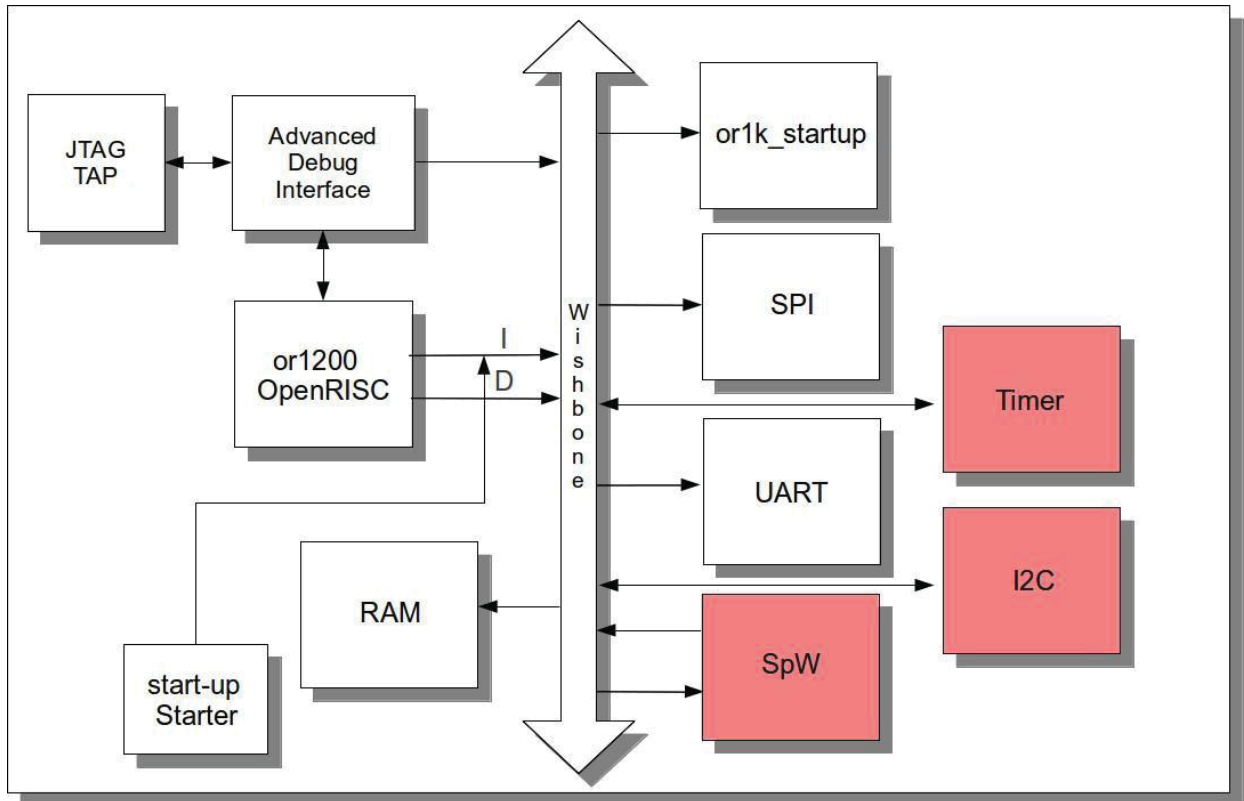


Figure 2. Current state of OpenASIM

## 1.2 Hardware & Software development

OpenASIM was developed based on an open source project called Minimal System-On-Chip (MinSOC). MinSOC consists of the minimal hardware necessary for a microprocessor system based on the OpenRISC 1200. The original system included a startup module, memory, an Ethernet and a UART core. The COSMIAC project supported several hardware development platforms, and its RTL was coded to support different FPGA manufacturers, different FPGA families as well as ASIC implementations. Compiling of the software and hardware module is accomplished through extensive use of MakeFiles across the directory structure of the project.

The following is a summary of the tasks to modify MinSOC and create OpenASIM as defined and executed by COSMIAC. The definition of these tasks was based on the specific requirements in Table 2.

1. Add support for latest FPGA devices from Xilinx: Virtex 5, Virtex 6 and Virtex 7.
2. Add support for new development platforms used to implement ASIMs.
3. Create a fully functional simulation framework and verification suite based on System Verilog to support the hardware development.
4. Modify hardware synthesis process to obtain higher frequencies of operation.
5. Add new hardware components as peripherals to support all ASIM features. New cores identified were: SpaceWire, I<sup>2</sup>C, SPI and a Real Time Timer.

6. Create a set of ASIM core libraries to support communication with SSM, the standard sensor software management running on Monarch networks.
7. Create a framework for code documentation that could be used for future developers to expand on the software and hardware capabilities of OpenASIM.
8. Streamline the process of porting OpenASIM to different FPGA-based hardware platforms.

### 1.1.1 Support for latest FPGA devices

The original MinSOC project supported FPGAs from Xilinx up to the Virtex 4 family. For OpenASIM requirements it was necessary to extend support for Virtex 5, Virtex 6 and the Series 7 families. Although MinSOC's RTL is written in a very portable form, a few key components of the system had to be coded in a family-dependent way in order to guarantee performance and an efficient implementation. For instance, memory structures had to be hard coded to specific primitives in each particular family for efficient and fast implementations. Important memory structures defined this way belonged to the main memory space (RAM for instruction and data memories) of the microprocessor. Other memories, such as cache, were implemented on a portable across families register file fashion, not requiring any modification. Figures 3 and 4 show a code snippet with modifications made to support newer families.

```
// Instantiation of FPGA memory:
//
// SPARTAN3/SPARTAN3E/VIRTEX2
// SPARTAN3A/VIRTEX4 are automatically reallocated by ISE
//
// Added By Nir Mor
//
wire [dw-1:0] doq_internal; // output data bus

RAMB16_S9 ramb16_s9_inst(
    .CLK(clk),
    .SSR(rst),
    .ADDR(addr),
    .DI(dt),
    .DIP(1'b0),
    .EN(ce),
    .WE(we),
    .DO(doq_internal),
    .DOP()
);

assign doq = (oe) ? (doq_internal) : { dw{1'bZ} };
```

Figure 3. Code for a Virtex 4 memory structure

Figure 3(Above), code for a Virtex 4 memory structure as originally defined by MinSOC, Figure 4 (right), code for a Virtex 5 memory structure, as added to provide support for latest FPGA families.

```
// Instantiation of FPGA memory:
//
// VIRTEX5 are automatically reallocated by ISE
//
// Added By alnz
//
wire [dw-1:0] doq_internal; // output data bus

BRAM_SINGLE_MACRO #(
    .BRAM_SIZE("18Kb"), // Target BRAM, "18Kb" or "36Kb"
    .DEVICE("VIRTEX5"), // Target Device: "VIRTEX5", "VIRTEX6",
                        // "SPARTAN6"
    .DO_REG(0), // Optional output register (0 or 1)
    .INIT(36'h00000000), // Initial values on output port
    .INIT_FILE ("NONE"),
    .WRITE_WIDTH(8), // Valid values are 1-72 (37-72 only valid
                    // when BRAM_SIZE="36Kb")
    .READ_WIDTH(8), // Valid values are 1-72 (37-72 only valid
                   // when BRAM_SIZE="36Kb")
    .SIM_MODE("SAFE"), // Simulation: "SAFE" vs. "FAST", see "Synthesis
                       // and Simulation Design Guide" for details
    .SRVAL(36'h00000000), // Set/Reset value for port output
    .WRITE_MODE("WRITE_FIRST") // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
)
    ramb16_s9 (
        .DO(doq_internal), // Output data
        .ADDR(addr), // Input address
        .CLK(clk), // Input clock
        .DI(dt), // Input data port
        .EN(ce), // Input RAM enable
        .RST(rst), // Input reset
        .WE(we) // Input write enable
    );

assign doq = (oe) ? (doq_internal) : { dw{1'bZ} };
```

Figure 4. Code for a Virtex 5 memory structure

Additionally, issues in the simulation of some memory structures were detected. Although these issues did not represent a risk of hardware malfunction, they did make the interpretation of simulation results more complex. Some issues are simulator dependent and are associated with the differences in mixed language simulations. An example is depicted in Figure 5. In ModelSim, VHDL and Verilog are compiled using the "single big file" strategy. However, SystemVerilog is not compiled the same way by default, unless the -mfcu flag is used. This

“minor” detail caused substantial issues in our simulation as the *‘include’* approach of having parameters did not scale properly across all the RTL files. The first warning was a significant memory collision problem that did not occur with original simulations.

```
# Memory Collision Error on RAMB16_S36_S36:test_minsoc.TheMinsoc.dut.or1200_top.or1200_cpu.or1200_rf.rf_b.ramb16_s36_s36.display_ra_wb at simulati
3146450.000 ns
# A read was performed on address 009 (hex) of Port A while a write was requested to the same address on Port B. The write will be successful howe
read value on Port A is unknown until the next CLKA cycle.
# Memory Collision Error on RAMB16_S36_S36:test_minsoc.TheMinsoc.dut.or1200_top.or1200_cpu.or1200_rf.rf_b.ramb16_s36_s36.display_ra_wb at simulati
3148450.000 ns
# A read was performed on address 003 (hex) of Port A while a write was requested to the same address on Port B. The write will be successful howe
read value on Port A is unknown until the next CLKA cycle.
```

Figure 5. Memory collision detected during simulation. This error implies a hardware malfunction and required analysis.

```
RAMB16_S36_S36 ramb16_s36_s36(
    .CLKA(clk_a),
    .SSRA(1'b0),
    .ADDRA({4'b0000, addr_a}),
    .DIA(32'h00000000),
    .DIPA(4'h0),
    .ENA(ce_a),
    .WEA(1'b0),
    .DOA(do_a),
    .DOPA(),
    .CLKB(clk_b),
    .SSRB(1'b0),
    .ADDRB({4'b0000, addr_b}),
    .DIB(di_b),
    .DIPB(4'h0),
    .ENB(ce_b),
    .WEB(we_b),
    .DOB(),
    .DOPB()
);

RAMB16_S36_S36 #(.SIM_COLLISION_CHECK("NONE")) ramb16_s36_s36(
    .CLKA(clk_a),
    .SSRA(1'b0),
    .ADDRA({4'b0000, addr_a}),
    .DIA(32'h00000000),
    .DIPA(4'h0),
    .ENA(ce_a),
    .WEA(1'b0),
    .DOA(do_a),
    .DOPA(),
    .CLKB(clk_b),
    .SSRB(1'b0),
    .ADDRB({4'b0000, addr_b}),
    .DIB(di_b),
    .DIPB(4'h0),
    .ENB(ce_b),
    .WEB(we_b),
    .DOB(),
    .DOPB()
);
```

Figure 6. Original code for memory

Figure 7. Apparent fix for the collision error. Cause still not resolved.

```
#!/bin/bash

#vlog -sv -mfcu -incr -work minsoc_test -f minsoc_sysverilog.src testbench.sv
vlog -sv -mfcu -incr -work minsoc_test -f minsoc_verilog.src testbench.sv
vcom -work minsoc_test -f minsoc_vhdl.src
```

Figure 8. Final error detection

The error depicted in Figures 6-8 was associated with mixed language files that are dealt with by ModelSim. The Default treatment for SystemVerilog files created problems in the way ‘default’ statements scaled across Verilog files in the design. By adding the “mfcu” precompiler flag, the collision error disappeared.

Another example of a key element hard coded in a family-dependent FPGA is the main clock manager. Figures 6 and 7 depict the differences in clock manager instantiations for the different families. Similar to the memory case, issues were found during simulation. In this particular case, the issue did not represent a risk of hardware malfunction, but made the interpretation of simulation results particularly difficult by inserting undefined signals. Initializing signals is not considered a good coding practice; instead, inserting reset states are recommended. Figures 9-12 depict an example of simulation differences before and after those reset states are included.

```

DCM_ADV #(
    .CLKDV_DIVIDE(divisor),           // Divide by: 1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,6.5
    .CLKFX_DIVIDE(FX_DIV),           // Can be any integer from 1 to 32
    .CLKFX_MULTIPLY(FX_MULT),        // Can be any integer from 2 to 32
    .CLKIN_DIVIDE_BY_2("FALSE"),     // TRUE/FALSE to enable CLKIN divide by two feature
    .CLKIN_PERIOD(10.0),             // Specify period of input clock in ns from 1.25 to 1000.00
    .CLKOUT_PHASE_SHIFT("NONE"),     // Specify phase shift mode of NONE, FIXED,
    .CLK_FEEDBACK("1X"),             // Specify clock feedback of NONE, 1X or 2X
    .DCM_AUTOCALIBRATION("TRUE"),    // DCM calibration circuitry "TRUE"/"FALSE"
    .DCM_PERFORMANCE_MODE("MAX_SPEED"), // Can be MAX_SPEED or MAX_RANGE
    .DFS_FREQUENCY_MODE("LOW"),      // HIGH or LOW frequency mode for frequency synthesis
    .DLL_FREQUENCY_MODE("LOW"),      // LOW, HIGH, or HIGH_SER frequency mode for DLL
    .DUTY_CYCLE_CORRECTION("TRUE"),  // Duty cycle correction, "TRUE"/"FALSE"
    .FACTORY_JF(16'hf0f0),           // FACTORY JF value suggested to be set to 16'hf0f0
    .PHASE_SHIFT(0),                // Amount of ftxed phase shift from -255 to 1023
    .SIM_DEVICE(XILINX_DCM_COMPONENT), // Set target device, "VIRTEX4" or "VIRTEX5" or "VIRTEX7"
    .STARTUP_WAIT("FALSE")          // Delay configuration DONE until DCM LOCK, "TRUE"/"FALSE"
) DCM_ADV_inst (
    .CLK0(CLK0_BUF),                // 0 degree DCM CLK output
    .CLK180(),                      // 180 degree DCM CLK output
    .CLK270(),                      // 270 degree DCM CLK output
    .CLK2X(FASTCLK_BUF),           // 2X DCM CLK output
    .CLK2X180(),                   // 2X, 180 degree DCM CLK out
    .CLK90(),                       // 90 degree DCM CLK output
    .CLKDV(CLKDV_BUF),             // Divided DCM CLK out (CLKDV_DIVIDE)
    .CLKFX(CLKFX_BUF),             // DCM CLK synthests out (M/D)
    .CLKFX180(),                   // 180 degree CLK synthests out
    .DDI(),                        // 16-bit data output for Dynamic Reconfiguration Port (DRP)
    .DRDY(),                       // Ready output signal from the DRP
    .LOCKED(),                     // DCM LOCK status output
    .PSDONE(),                    // Dynamic phase adjust done output
    .CLKFB(CLKFB_IN),             // DCM clock feedback
    .CLKIN(CLKIN_IBUFG),          // Clock input (from IBUFG, BUFG or DCM)
    .DADDR(7'h00),               // 7-bit address for the DRP
    .DCLK(1'b0),                  // Clock for the DRP
    .DEN(1'b0),                  // Enable input for the DRP
    .DI(16'h0000),                // 16-bit data input for the DRP
    .DWE(1'b0),                  // Active high allows for writing configuration memory
    .PCLK(1'b0),                 // Dynamic phase adjust clock input
    .PSEN(1'b0),                 // Dynamic phase adjust enable input
    .PSINCDEC(1'b0),             // Dynamic phase adjust increment/decrement
    .RST(1'b0)                   // DCM asynchronous reset input
);

```

Figure 9. Clock manager code for Virtex 5.

```

MMCME2_ADV
#(.BANDWIDTH ("OPTIMIZED"),
    .CLKOUT4_CASCADE ("FALSE"),
    .COMPENSATION ("ZHOLD"),
    .STARTUP_WAIT ("FALSE"),
    .DIVCLK_DIVIDE (1),
    .CLKFBOUT_MULT_F (5.000),
    .CLKFBOUT_PHASE (0.000),
    .CLKFBOUT_USE_FINE_PS ("FALSE"),
    .CLKOUT0_DIVIDE_F (5.000),
    .CLKOUT0_PHASE (0.000),
    .CLKOUT0_DUTY_CYCLE (0.500),
    .CLKOUT0_USE_FINE_PS ("FALSE"),
    .CLKOUT1_DIVIDE (2),
    .CLKOUT1_PHASE (0.000),
    .CLKOUT1_DUTY_CYCLE (0.500),
    .CLKOUT1_USE_FINE_PS ("FALSE"),
    .REF_JITTER1 (0.010))
mmcm_adv_inst
(
    .CLKFBOUT (clkfbout),
    .CLKOUT0 (clkout0),
    .CLKOUT1 (clkout1),
    .CLKOUT2 (clkout2),
    .CLKFBIN (clkfbout_buf),
    .CLKIN1 (clkln1),
    .CLKIN2 (1'b0),
    .CLKINSEL (1'b1),
    .DADDR (7'h0),
    .DCLK (1'b0),
    .DEN (1'b0),
    .DI (16'h0),
    .DO (do_unused),
    .DRDY (drdy_unused),
    .DWE (1'b0),
    .PCLK (1'b0),
    .PSEN (1'b0),
    .PSINCDEC (1'b0),
    .PSDONE (psdone_unused),
    .LOCKED (LOCKED),
    .PWRDN (1'b0),
    .RST (1'b0));

```

Figure 10. Clock manager code for Virtex 7.

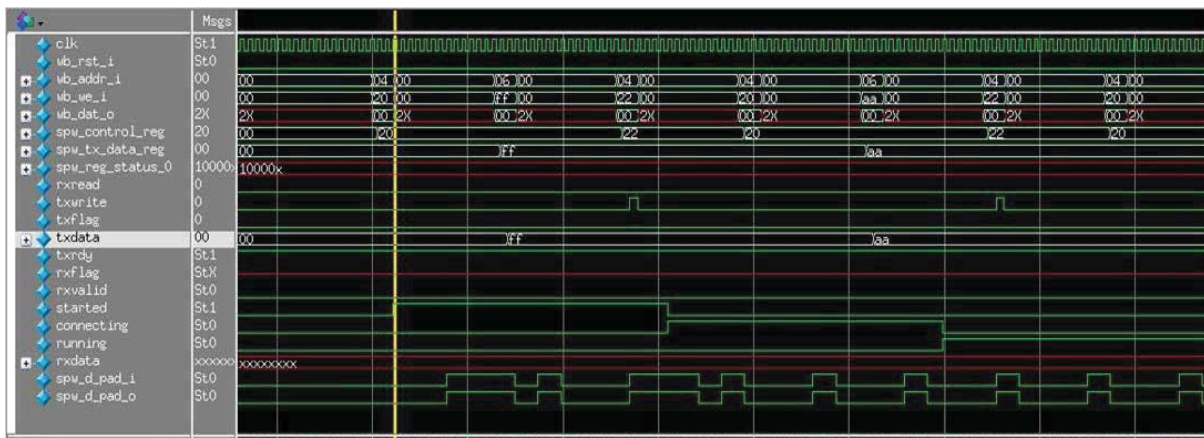


Figure 11. Undefined signals inserted during simulation.

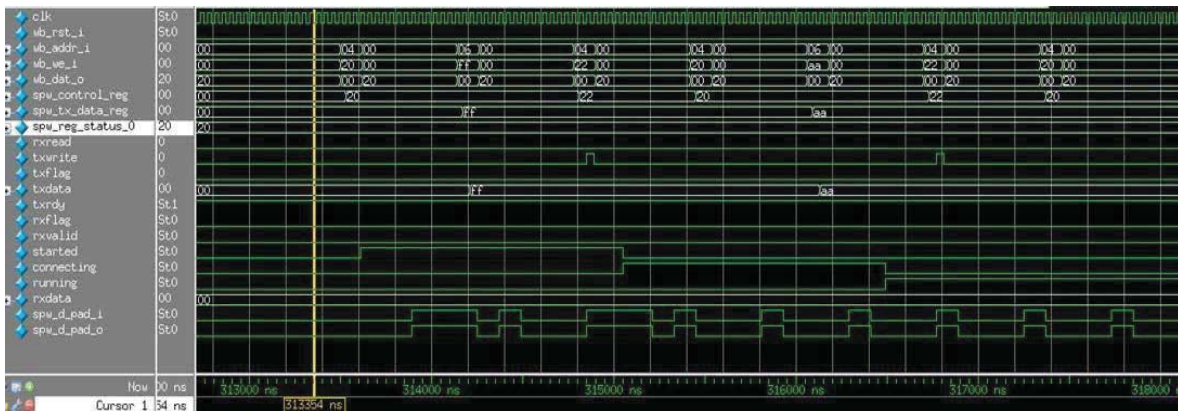
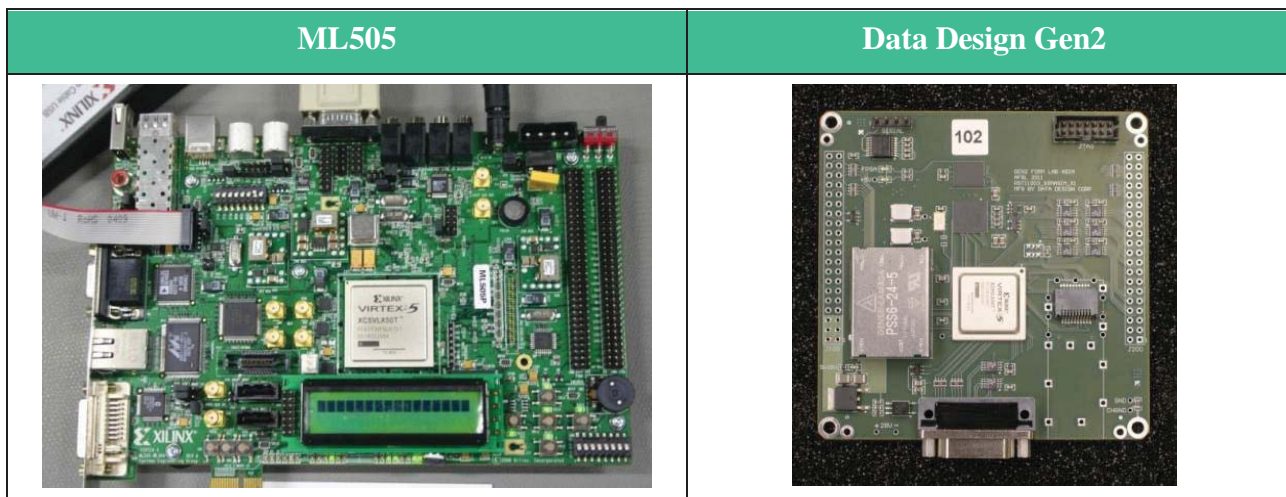


Figure 12. Same simulation after reset states were correctly introduced in simulation code.

### 1.3 Support for new development platforms

Support for new development platforms was necessary to facilitate the development, testing, and deployment of OpenASIM using hardware platforms that have traditionally been used for ASIMs (e.g. DataDesign Corporation’s Generation 2 ASIMs based on the Virtex 5 device). Two new platforms were identified for development and testing: Xilinx’s ML505 and ML510 development boards (both based on the Virtex 5 family). One new platform was identified for deployment in current space applications: DataDesign’s Generation 2 ASIM, also based on the Virtex 5 family.



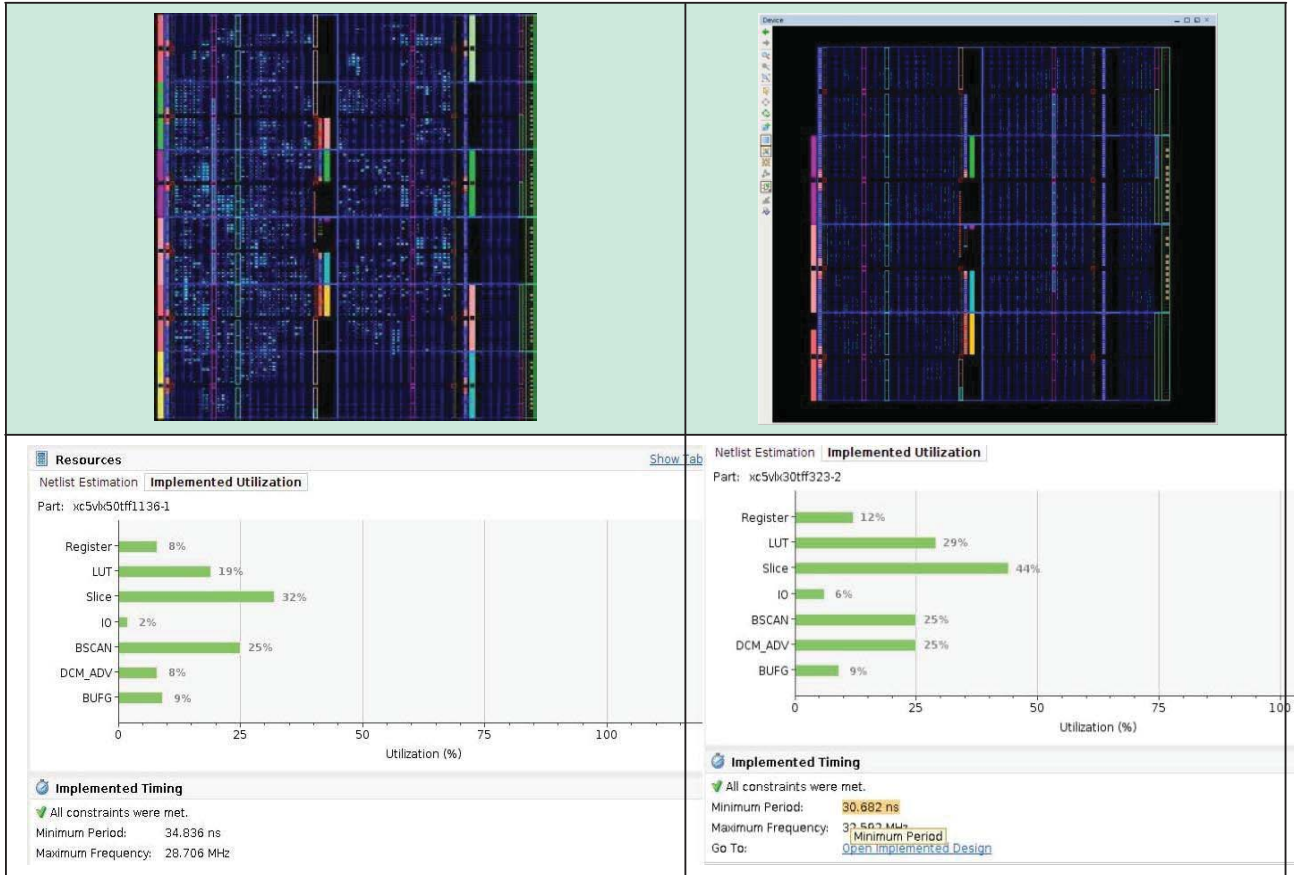


Figure 13. Differences between the ML505 and the Data Design Gen2 hardware platforms.

Software support for new development platforms is done by creating new configuration files for each new platform added. These configuration files include information such as the FPGA device the platform uses, default clock speeds, peripherals and communication channels the platform supports, pin assignments for all I/Os, reset signals polarities, etc. Figure 13 depicts differences in resources usage between the ML505 and the DataDesign Corporation’s Generation 2 hardware platforms. An interesting observation is that even though both are based on the Virtex 5 family, they use different chips and implemented features due to platforms differences.

#### 1.4 Creation of simulation framework and verification suite

The original MinSOC project included a set of simple test benches that are used to manually verify hardware functionality. MinSOC also includes an architectural simulator useful to simulate software or operating systems. For OpenASIM purposes, a more complete, self-checking set of test benches was needed. The new simulation framework developed for the OpenASIM allows the user to simulate complex behavioral processes (such as communication over serial channels like UART, I2C, SpW, etc) and perform self-checking tests. To accomplish this, COSMIAC’s team decided to use System Verilog as a language to develop the new test benches. This decision was based on the extensive use of System Verilog across the industry to develop verification suites for complex FPGA and ASIC designs.

This simulation framework is based on the definition of Bus Functional Models (BFM) to model

the high level behavior of the system. BFM have the additional advantage of a more scalable simulation framework as they enable the user to create a “harness” to interface with the device under test (DUT). The DUT can change the nature and number of its I/Os over the development time frame. New test benches can be written to exercise and monitor the changing DUT without breaking test benches that were written for an older version of the DUT. As such, BFMs provide a more scalable approach to building a comprehensive verification suite.

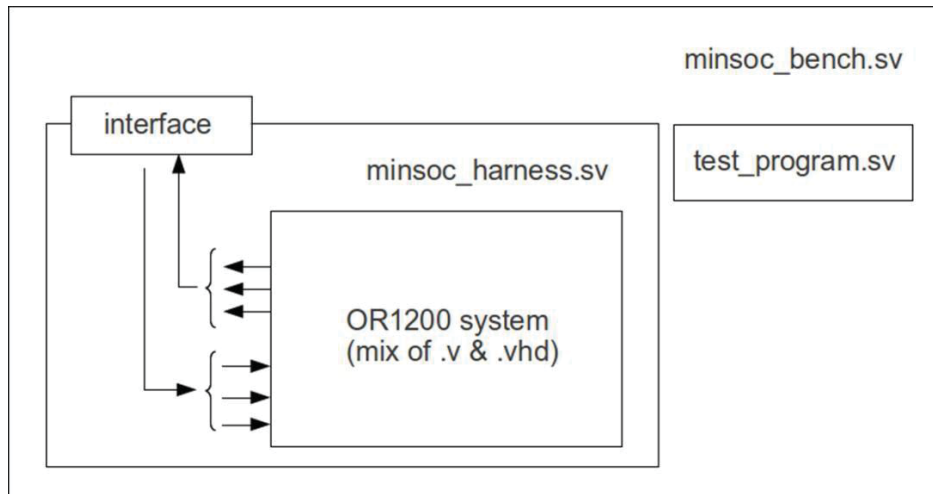


Figure 14. Newly defined simulation framework

Figure 14 depicts a block diagram with all the components in the newly created simulation framework. To simulate a mixed language design using System Verilog test benches, advance simulator software was used: Riviera-Pro from the Aldec Corporation. COSMIAC has acquired a commercial license for a complete package (all features included) of this software.

To illustrate the full potential of the simulation framework created, Figures 15 and 16 depict the console output of the original test bench provided with the MinSOC project. The reader can appreciate a simple and deterministic approach to evaluate the functionality of the UART. Although this approach works, it scales poorly and becomes impractical when the verification engineer attempts to increment the overall coverage of the tests. The same Figures also depict the output of the UART test using the newly created simulation framework. In this case, the number of tests is an input parameter and the stimulus provided to the DUT is stochastic in nature, providing better tools to increase the test coverage.



```

# (minsoc_bench.minsoc_top_0.spw_light_top) SPW INFO: Data bus width is 32. Debug Interface present.
#
# Running simulation: if you want to stop it, type ctrl+c and type in finish afterwards.
# Testing UART firmware, this takes a while (~1 min. @ 2.53 GHz dual-core)...
# Hello World.
# UART data received.
# Testing UART interrupt...
# UART interrupt failed. B was expected, A was received.
# UART firmware test completed, behaving correctly.
# Stopping simulation.
# ** Note: $finish      : /opt/minsoc/prj/./bench/verilog/minsoc_bench.v(216)
#   Time: 3557560 ns  Iteration: 1  Instance: /minsoc_bench

```

Figure 15. Original MinSOC console output of the simulation testing the system's UART.

```

# (test_minsoc.TheMinsoc.dut.spw_light_top) SPW INFO: Data bus width is 32. Debug Interface present.
#
# Initializing FPGA memory to 0s
# Memory model initialized with firmware:      ../.././sw/uart/uart.hex
#   7888 Bytes loaded from      7888 ...
# Testing UART firmware, this takes a while (~1 min. @ 2.53 GHz dual-core)...
# Controller have been reseted
# @test_uart.
# @uart_decoder.
# Hello World.
# UART data received.
# Testing UART interrupt. A total of 6 test chars will be send.
# Test 0: Send D, Received D. Passed
# Test 1: Send R, Received R. Passed
# Test 2: Send J, Received J. Passed
# Test 3: Send U, Received U. Passed
# Test 4: Send X, Received X. Passed
# Test 5: Send Q, Received Q. Passed
# UART firmware test completed, behaving correctly.
# ** Note: $finish      : random_uart_test.sv(60)
#   Time: 6261040 ns  Iteration: 1  Instance: /test_minsoc/test
[4]+ Done      ./run_sim.sh console ../.././sw/uart/uart.hex

```

Figure 16. Console output for simulation testing the system's UART using the newly created simulation framework.

## 1.5 New hardware features to support ASIM requirements

Given the requirements defined in Table 2, a set of new hardware peripherals was created and integrated with the original MinSOC system. MinSOC is based on the OpenRISC OR1200 microprocessor which uses WishBone as its main bus standard. WishBone is a widely accepted bus standard and many of the cores available under the open source license and community are compliant with it. The main cores that COSMIAC integrated to build the OpenASIM system are SpaceWire and I2C. These cores are of particular importance as they represent the two major standards for Monarch networks communication channels.

## 1.6 SpaceWire integration

SpaceWire is a newly created AIAA standard [SPW\_AIAA] that supports data rates up to 400Mb/sec. Two existing cores were investigated as candidates to form part of the OpenASIM system. The first was a core designed by NASA's Goddard Space Flight Center that is provided at no cost for government supported projects. COSMIAC had access to the RTL description of the core for analysis and possible integration into the OpenASIM system. Figure 17 depicts a block diagram of NASA's core.

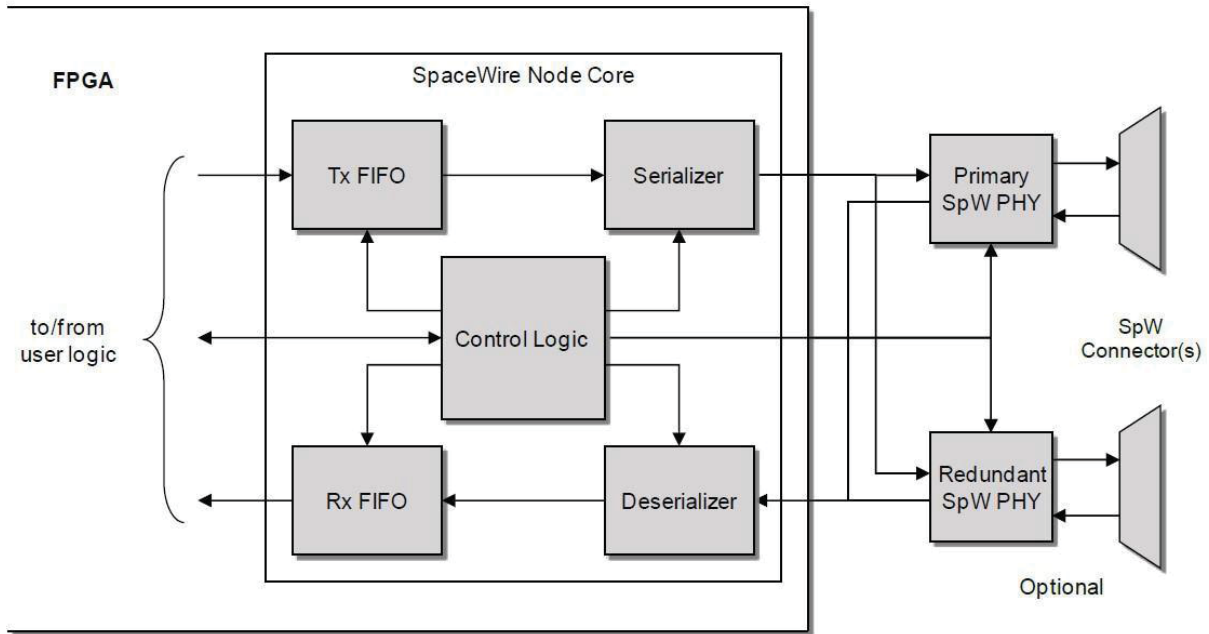


Figure 17. NASA's Goddard Space Flight Center SpaceWire core.

This core was successfully integrated into the OpenASIM system after creating a bus interface compatible with WishBone. Further testing and development was abandoned because the core, which relies on a particular PHY chip outside the ASIM platform, is expensive and is not present on the hardware platforms the OpenASIM project focused on. Despite the fact that development in this path was discontinued, the current OpenASIM RTL supports integration of this core.

The second core alternative analyzed was part of an open source project known as "SpaceWire Light" [4]. SpaceWire Light is a VHDL core implementing a SpaceWire encoder-decoder, synthesizable for FPGA targets. A goal of this project was to provide a complete, reliable and fast implementation of a SpaceWire encoder-decoder according to ECSS –E–ST–50–12. The core is "light" in the sense that it does not provide additional features such as RMAP, packet routing, etc. [5]. This core included an interface for an AMBA bus, which is the bus standard for other microprocessors such as Leon3 [6].

To complete the integration of this core into the OpenASIM system, a WishBone compliant interface was created and added. In the process of simulating and verifying the integration some improvements were made to the WishBone interface to obtain better data rates and reduce latency. Figure 19 depicts the simulation of a bus transaction on the original adaptation of the core with obvious latency due to the numerous control-related transactions involved in the transfer of a byte. Figure 20 depicts an improved version of the same interface where many of the control related operation were eliminated, obtaining lower latency and better data transfer rates.

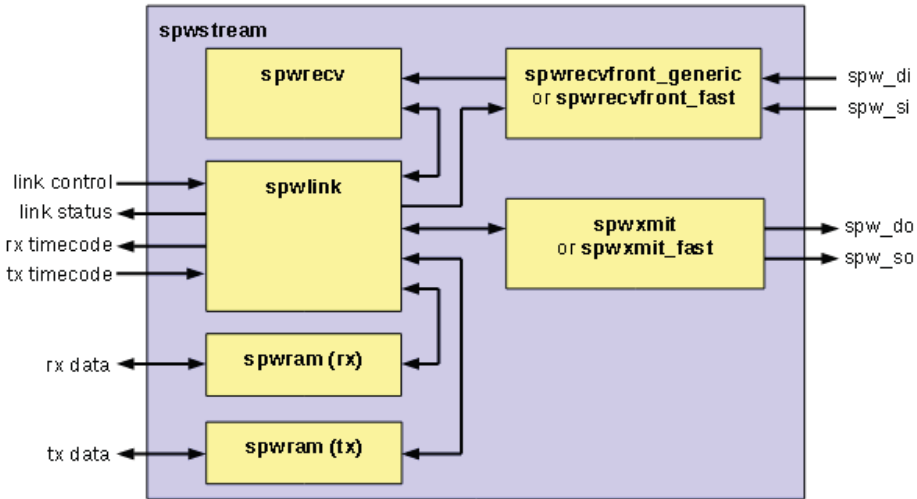


Figure 18. SpaceWire Light block diagram.



Figure 19. Simulation of a bus transaction on the original adaptation of the core. The red circle with the number 1 represents the action of sending a word to the TX register in the core. Number 2 is a control operation to set a txwrite flag high in order to trigger the actual transmission. Number 3 sets the same flag low to allow a new transaction to take place. The overall operation took ~1920 ns @ 40ns clock period.

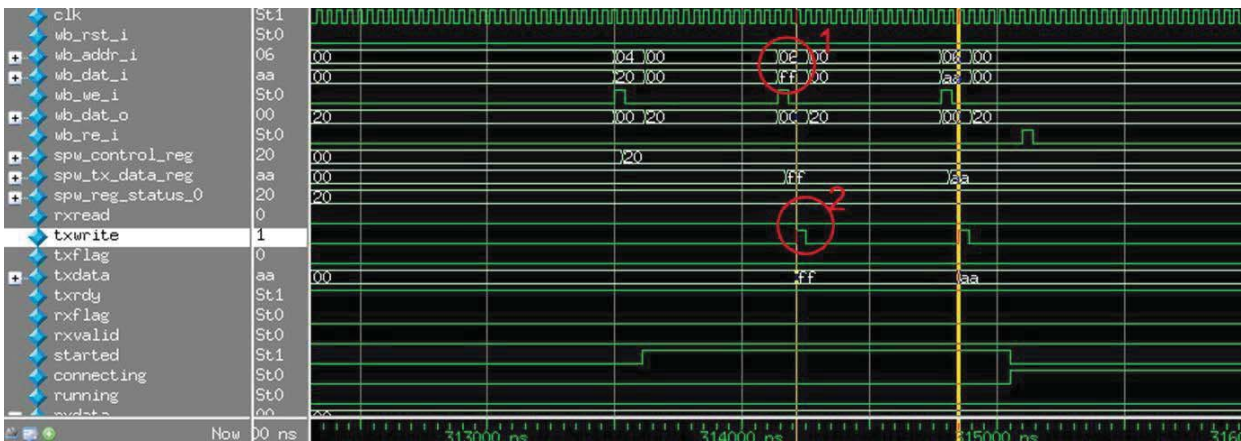


Figure 20. Simulation of a bus transaction on the improved adaptation of the core. The red circle with the number 1 represents the action of sending a word to the TX register in the core. Number 2 shows txwrite flag being automatically generated. The total time this transaction takes is ~640ns @ 40ns clock period (1/3 original latency).

To perform further testing, two Star-Dundee devices were obtained on loan from AFRL; a SpaceWire Link Analyzer MK2 [7] and a SpaceWire brick. The Analyzer consists of a 4 SpaceWire port box with a USB port to connect to a host computer and can operate in two modes: Loop-through and Loop-back. The analyzer was used in the loop-back mode, where it basically routes all traffic without modification while capturing and presenting it on a graphical user interface at the host computer for analysis and debugging. Figures 21 and 22 depict a block diagram of this mode's operation and a snapshot of its graphical user interface showing traffic.

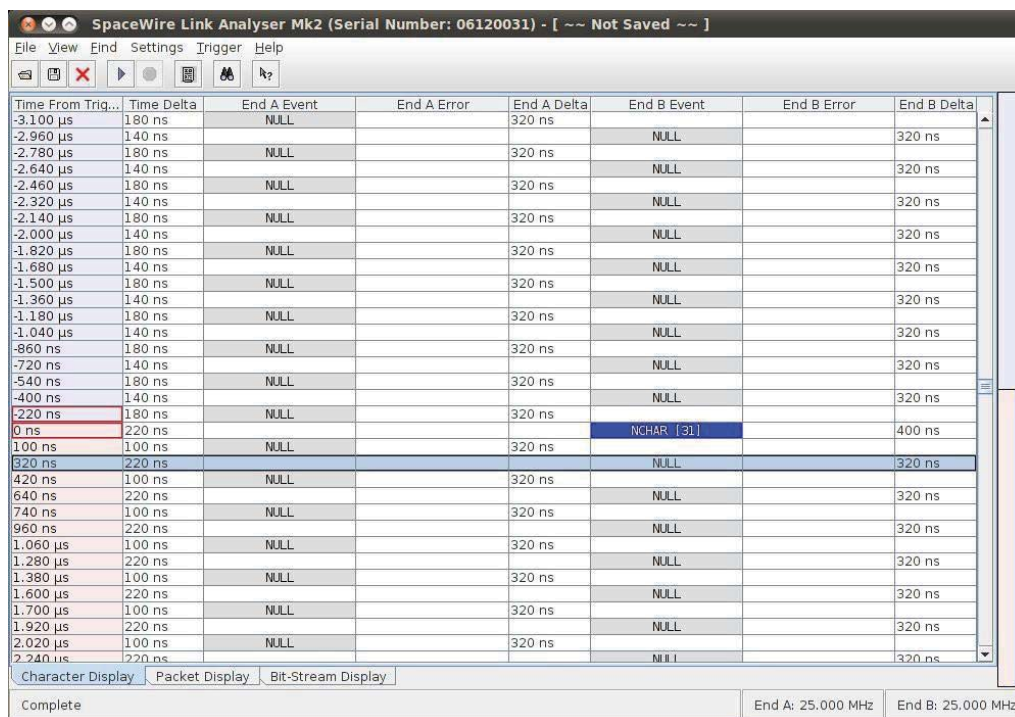


Figure 21. SpaceWire Analyzer's Graphical User Interface snapshot showing traffic.

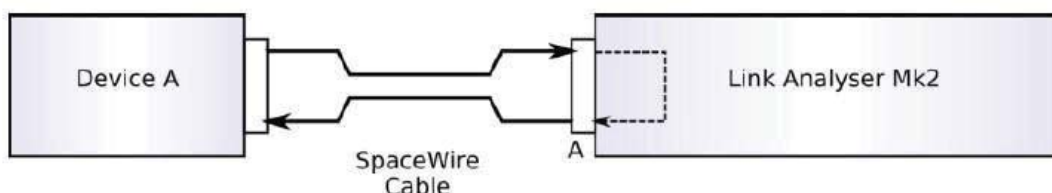


Figure 22. SpaceWire Analyzer in Loop-back mode [7].

The second device, the SpaceWire brick, is a USB-to-SpaceWire adapter that allows any computer with a USB port to connect to a SpaceWire network. Figure 23 depicts the SpaceWire brick architecture.

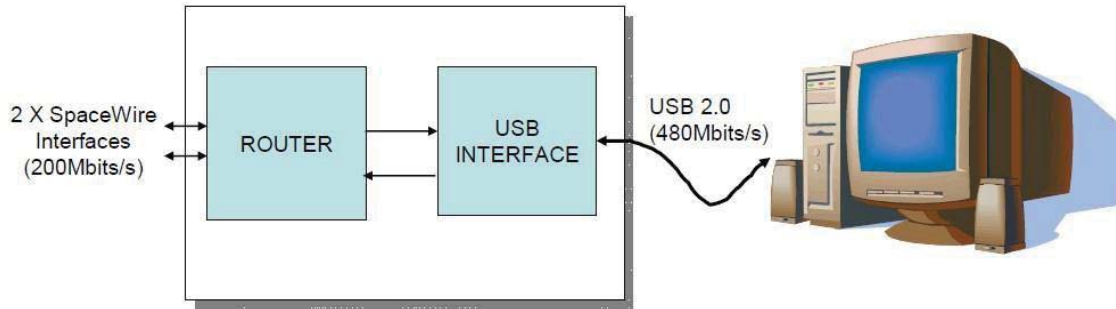


Figure 23. SpaceWire brick architecture.

Both of these devices were used by COSMIAC’s team to perform further testing on the newly integrated SpaceWire core. The actual setup is depicted in Figure 24. Testing was done using specialized test routines and drivers provided by the manufacturer of both the SpaceWire Analyzer and the SpaceWire Brick. Testing focused on obtaining the maximum data transfer rate possible with the hardware platforms at hand and compliance with the SpaceWire standard. Performance improvements were obtained by optimizing the RTL code and also simplifying some of the software driver routines of the SpaceWire core. Figure 25 shows a code snippet of the SpaceWire interruption handler routine, which was simplified in order to allow higher data transfer rates.



Figure 24. COSMIAC’s setup to test SpaceWire integration.

```

void spw_interrupt()
{
    char spw_byte_rcved;
    char lsr;
    do {
        lsr = REG8(spw_base + SPW_STAT0);
        spw_byte_rcved = REG8(spw_base + SPW_RXDAT);
        REG8(spw_base + SPW_TXDAT) = spw_byte_rcved;
    } while ((lsr & DATA_VALID) == DATA_VALID);
}

```

Figure 25. Interruption handler routine.

COSMIAC obtained an average data transfer rate of 5.44Mbits/sec and a bus clock frequency of 25MHz, which is within the margins set as goals in Table 2. It is possible to increase the overall

performance of the SpaceWire channel by creating further clock domains within the OpenASIM system. Currently, the entire system (CPU plus peripherals including the SpaceWire core) runs at 75MHz. It is possible to split the system into 2 clock domains: the CPU and most of its peripherals on one clock domain and the SpaceWire core on a separate clock domain. The SpaceWire core supports this option. Since the slowest path is currently through the CPU, it is expected that such separation will allow the SpaceWire core to operate at higher clock frequency, potentially improving the bus clock frequency and the average data transfer rate.

### 1.7 I<sup>2</sup>C integration

Inter-Integrated Circuit (I<sup>2</sup>C) is a standard that supports bus clock speeds up to 400KHz. An I<sup>2</sup>C node requires a master and a slave. A core labeled master-slave-i2c, in the open source community, was adapted for its integration on the OpenASIM system. The core is WishBone compliant and only minor modifications were required to perform the integration (the core supports an 8 bit data bus, while the OpenASIM has a 32-bit data bus). The integrated core is capable of I<sup>2</sup>C clock rates over 400KHz. Figure 26 depicts a block diagram of the I<sup>2</sup>C core and its Wishbone interface. To complete the integration, a set of low level software routines or drivers were written to operate the core.

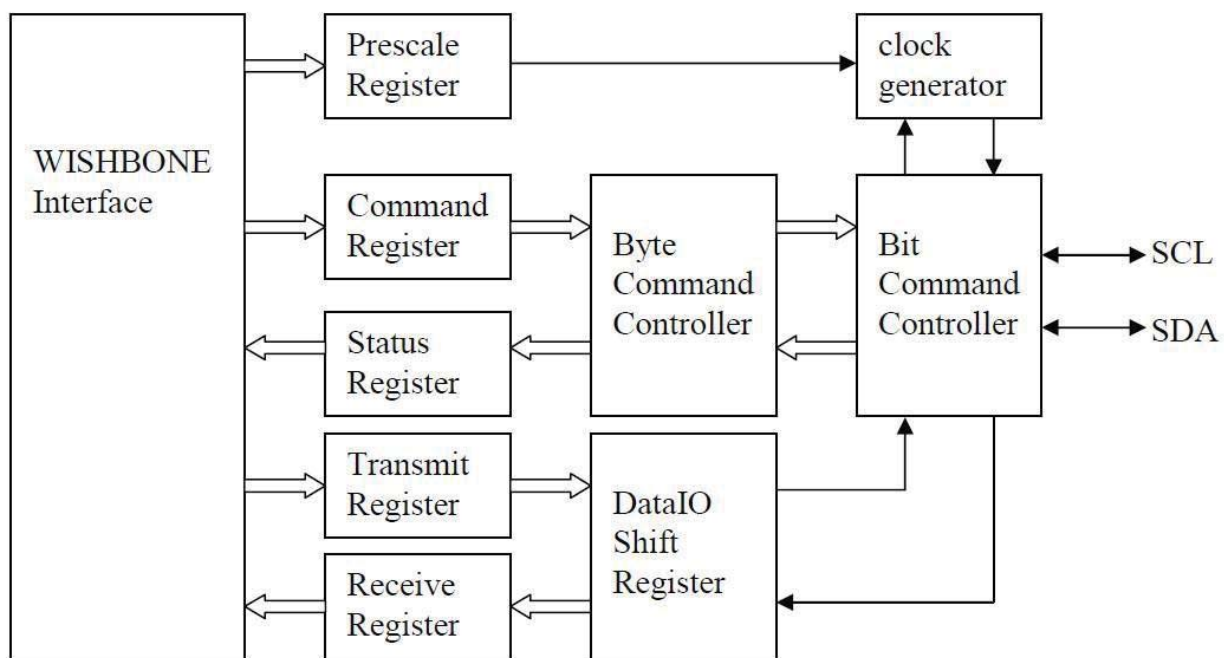
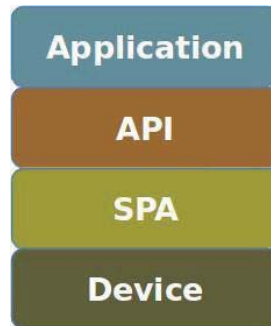


Figure 26. I2C core block diagram.

### 1.8 Creation of ASIM core libraries for SSM support.

The COSMIAC team developed the lower, hardware-dependent-layer of the ASIM core libraries. Figure 27 depicts the 4 layer architecture of the proposed libraries. The lower layer, the closest to the hardware, has an implementation that is strongly driven by the details of the hardware implementation. Several AFRL sponsored teams we collaborated with undertook the task of developing this lower layer for different hardware platforms. COSMIAC is responsible for the OpenASIM implementation. Seven functions are defined to be the center of the lower layer:

1. `UInt16 send (UInt8* pBuf, UInt16 length);`
2. `UInt16 receive (UInt8* pBuf);`
3. `Int8 createTimer (void* (*funcPtr) (void*), UInt8 rate);`
4. `Int8 watchdog_checkIn (void);`
5. `void debugPrint (char* pMsg);`
6. `Time_t getTime (void);`
7. `Int8 setTimeAtTone (Time_t val);`



**Figure 27. ASIM Core libraries Layered Architecture.**

The “send” and “receive” functions are used to transfer data through either of the communication channels supported by OpenASIM, I<sup>2</sup>C or SpaceWire. Selection of which channel used is defined by using #define statements within the libraries.

The function “createTimer” initializes a periodic interruption that an application developer uses to perform periodic tasks. The number of “timers” that can be created is specific to the ASIM used and should be defined within the documentation. Currently, OpenASIM supports the creation of only one timer using the onboard “Tick Timer”. If the function is called a second time, it returns a negative value indicating error.

The function “watchdog\_checkIn” resets the processor watchdog to avoid a general reset from occurring. Watchdog timers are set in order to allow a microprocessor to reset itself in case it gets stuck into an infinite loop. The time it takes for a watchdog timer to reset the processor is application and platform dependent. COSMIAC’s team decided that hardware developers are free to set a time based on the characteristics of the ASIM and best practices. For OpenASIM, this time is set to 1 second.

The function “debugPrint” is a generic printout function whose output is re-directed to a system console. For the vast majority of ASIMs and simple microprocessor systems, this console is represented by an UART.

“getTime” and “setTimeAtTone” are complementary functions. The first function returns a time structure with the current system time based on an onboard timer. The second function sets the value of that timer at a particular point in time. ASIM’s that support SpaceWire may have an external signal called PPS (pulse per second) used to signal the onboard timer when to start counting, starting at a count previously loaded with the function setTimeAtTone. Other ASIMs, without the PPS signal, will receive a command and a time through the Monarch network. The

time is loaded immediately into the timer and is used to set the overall time of the system. OpenASIM currently supports the second non PPS signal option. Although a PPS is a relatively simple signal to add, it was not accomplished at the time this report is written. Currently, the set of ASIM core libraries for the OpenASIM is under test.

### **1.9 Framework for code documentation.**

The cores used for OpenASIM have their hardware description well documented, but software routines have no documentation besides some comments in the actual source files. Since an important effort of the OpenASIM project was dedicated to write software, COSMIAC decided that a good documentation method should be selected from the beginning. This facilitates transferring the code to other developers, as well as to facilitate the tasks of debugging and development of higher level software routines. Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D. OpenASIM uses Doxygen to generate its software code documentation. A configuration file for the process of generating documentation was created and has gone through several iterations to its current state. Figure 28 depicts an example of OpenASIM software documentation. In this particular case the documentation for I<sup>2</sup>C drivers is depicted.



### **i2c\_ack\_interrupt ( unsigned long i2c\_base\_addr )**

Acknowledges interrupt has been serviced

Definition at line 210 of file `i2c_master_slave.c`.

```
{  
    i2c_write_reg(i2c_base_addr, I2C_CR, I2C_CR_IACK);  
    return 0;  
}
```

### **i2c\_deact\_as\_slave ( unsigned long i2c\_base\_addr )**

Disables slave mode for this I2C core and deasserts slave enable bit in control register

Definition at line 125 of file `i2c_master_slave.c`.

```
{  
    // Clear slave enable bit  
    i2c_write_reg(i2c_base_addr, I2C_CTR, i2c_read_reg(i2c_base_addr, I2C_CTR) &  
        ~I2C_CTR_SLAVE_ENABLE);  
    return 0;  
}
```

### **i2c\_deact\_core ( unsigned long i2c\_base\_addr )**

Deactivates I2C core, clear core enable and interrupt enable bits

Definition at line 97 of file `i2c_master_slave.c`.

```
{  
    i2c_write_reg(i2c_base_addr, I2C_CTR, i2c_read_reg(i2c_base_addr, I2C_CTR) &
```

Figure 28. Snapshot for I2C drivers' documentation.

## **1.10 Streamlined porting of OpenASIM to other platforms.**

One of the goals of the OpenASIM project is to facilitate or streamline the process of adapting the code for new and different FPGA-based hardware platforms. This process is currently being accomplished manually by modifying multiple configuration files. Under the OpenASIM project, an application in PERL has been created to perform the process of porting the code to new platforms by means of an intuitive graphical user interface. Figure 29 depicts an early version of the application. Currently, the application is still under development and testing.

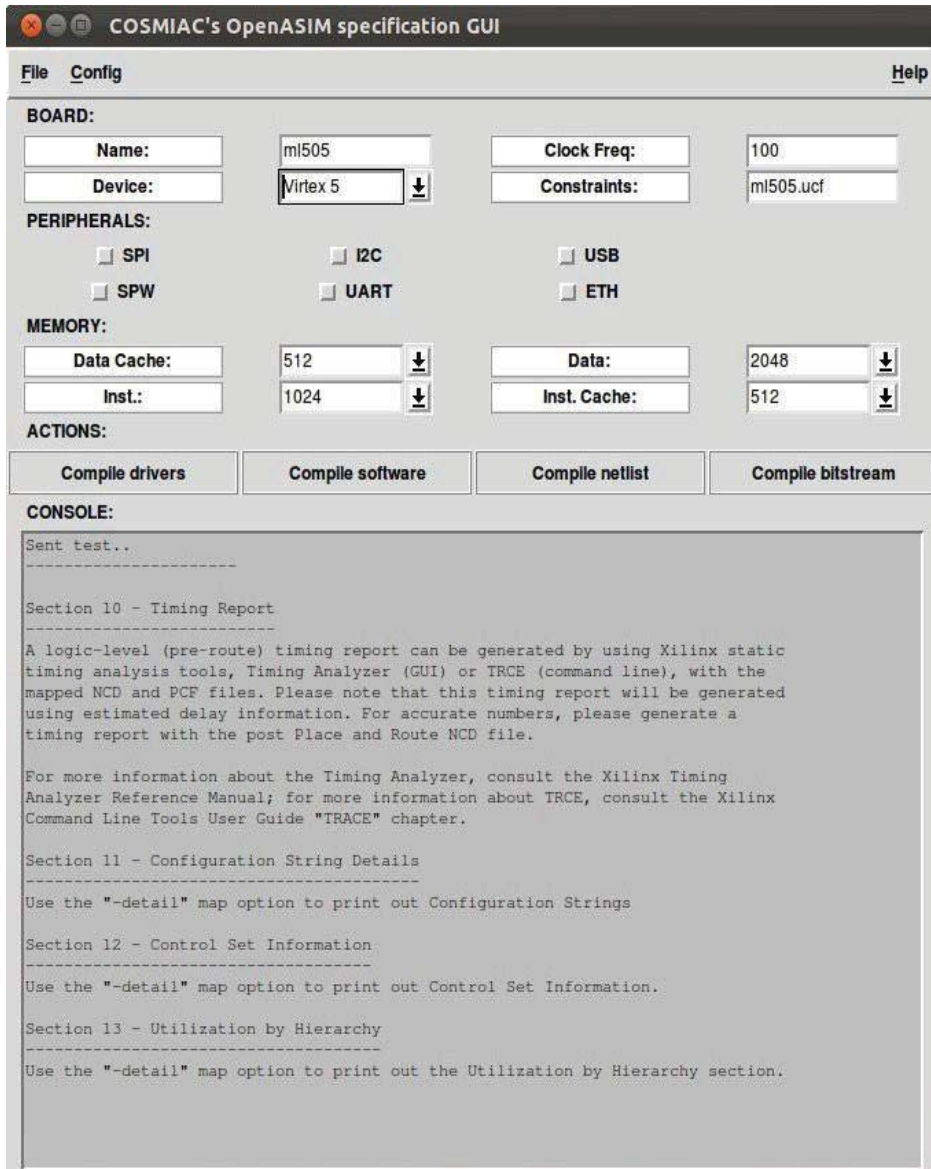


Figure 29. PERL application developed to centralize the generation of configuration files needed to port the OpenASIM design to different FPGA-based hardware platforms.

## **2. Improving the reliability of FPGAs and other configurable devices in space and defense systems including operation in demanding environments such as the radiation environment of space**

The traditional approach for Single Event Upset (SEU) mitigation on commercial parts is triple modular redundancy (TMR). Although proven effective, this method adds logic overhead and a penalty in power consumption and processing speed. This approach is also vulnerable to multiple-bits upset that are becoming more frequent as geometries decrease in modern devices. An alternative approach – called “scrubbing” - relies on simply reloading the configuration memory frames at defined time intervals. This approach is possible with FPGA devices that support dynamic Partial Reconfiguration (DPR). Scrubbing provides protection against the accumulation of upsets in the configuration memory and, in combination with TMR, improves the overall system's reliability. Work done towards this effort using FPGA-based Dynamically Reconfigurable Systems is described below.

### **2.1 FPGA-based Dynamically Reconfigurable Systems**

Dynamical partial reconfiguration (DPR) is a relatively new feature of FPGAs. DPR allows the designer to partially reconfigure an FPGA without stopping or affecting the other parts of the device are not being reconfigured. This feature opens endless possibilities in the realm of digital systems, including the Holy Grail of *Reconfigurable Computing*: a computer capable of changing itself in response to an external stimulus. The idea of a reconfigurable computer was first published in the 1960's by Gerald Estrin, in his landmark paper “Organization of Computer Systems – The Fixed Plus Variable Structure Computer” [8].

The idea was first triggered by an apparent lack of interest from commercial computer manufacturers to develop solutions for many vital computational problems, focusing only on conventional computer systems. The initial concept of reconfigurable computing was to allow the acceleration of computational processes by using variable configurations of specialized hardware modules to a classic sequential processing unit. A prototype of the F+V computer described in Reference [8] was developed at UCLA (See Figure 30). Although the concept didn't have much resonance at its time, it started a revolution in the early 1990's when technique finally caught up with this visionary concept.

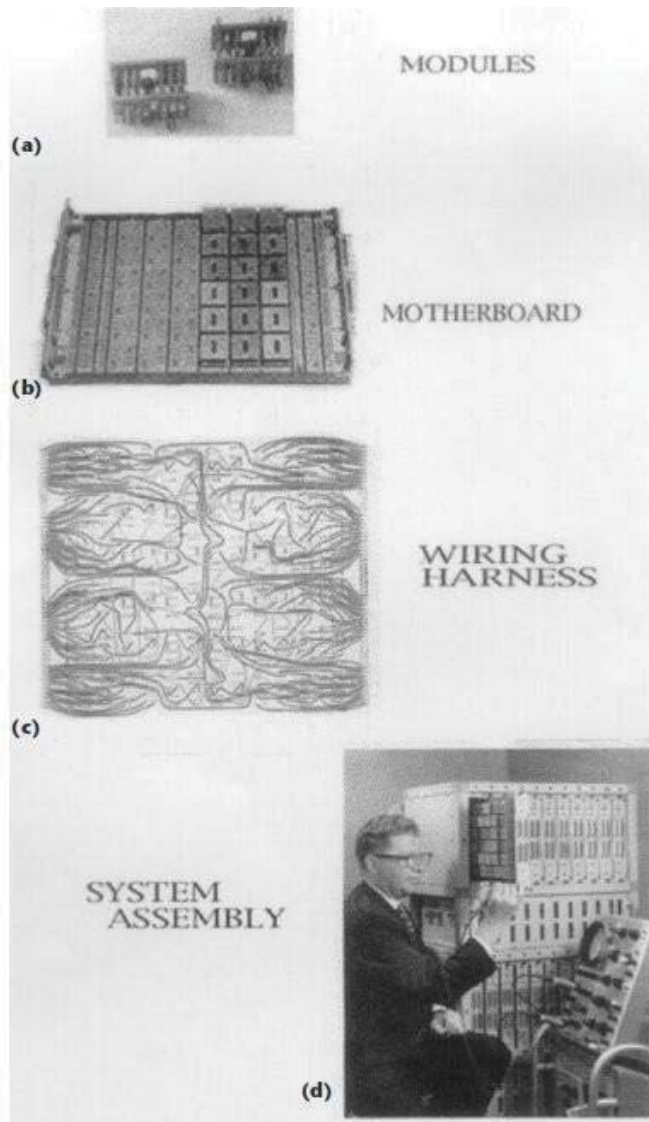


Figure 30. F+V computer prototype [8].

The world's first commercial reconfigurable computer, the Algotronix CHS2X4, was completed in 1991. Although not a commercial success, it promised enough that Xilinx bought the technology. Xilinx started to commercialize FPGAs in the early 1990's with an architecture whose basic ideas persist in today's devices. FPGAs are organized as heterogeneous (in the beginnings FPGAs were more homogenous) arrays of primitives to perform specific tasks (arithmetic operations, memory controllers, CPUs, digital signal processing operations, etc) and generic/programmable tasks (LUTs and FFs within CLBs). Figure 31 depicts a generic and simplified block diagram of an FPGA's architecture.

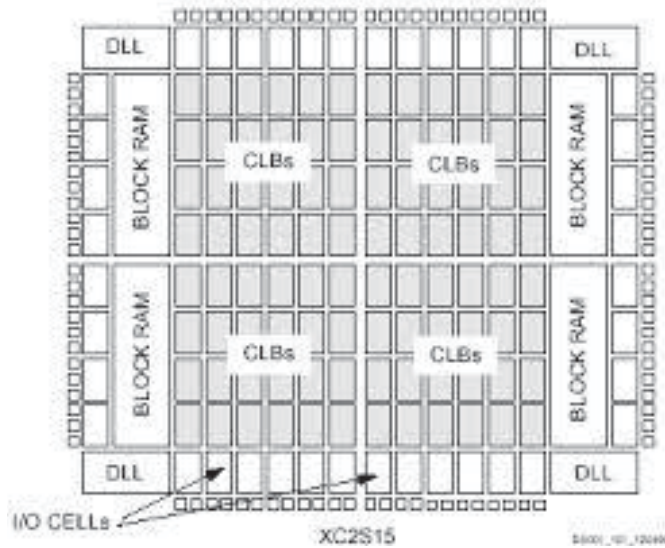


Figure 31. FPGA architecture's block diagram.

FPGAs more distinctive feature is its SRAM-based configuration memory. The SRAM nature of the configuration memory is the origin of FPGA's main disadvantages and advantages. SRAM memory loses its contents if powered off, requiring FPGAs to be configured each time they are powered up. This impedes some applications. SRAM memories usually consume more power than other alternatives. FPGAs are very power intensive devices and powering them on and off is not desirable. This is a substantial issue, especially for portable applications. Finally, SRAM memories are susceptible to radiation effects and their contents can be "flipped" in radiation harsh environments. These events are labeled Single Event Upsets (SEUs) and are a major problem for space-borne applications. Conversely, SRAM memories can be read/write at faster rates than other alternatives and can be addressed to the bit level. This characteristic, together with a technology that Xilinx labeled "glitch less configuration memory cell", enabled today's DPR capabilities.

The "glitch less" configuration memory cell allows a write process to happen in a configuration memory cell without generating glitches in the underlying logic, as long as no change in state is produced. This is particularly important since the FPGA's configuration memory smallest accessible unit is a frame (41 words of 32 bits for Virtex 4 and newer devices).

configuration memory is accessed externally or within its reconfigurable fabric (internally). Internal reconfiguration provides devices with the capability of reconfiguring themselves. Internal access to the configuration memory is accomplished through the Internal Configuration Access Port (ICAP). Although the ICAP was available in early devices (Virtex 2 are the first devices we are aware of that included this primitives), using it to perform DPR was not yet possible because the design flow was not yet supported by the tools.

DPR generated new design challenges. Two important challenges, physical partitioning and routing planning, were not supportable because of the absence of tools. Physical partitioning for DPR became more complicated because the tools needed to assure that two time concurrent, reconfigurable modules were not going to interfere with each other. Routing planning also became more complicated because tools had to assure that any physical circuitry was placed and

routed such that all partial configurations interface correctly with each other. It was not until late 2011 that tools started to support DPR commercially. Figure 32 depicts the evolution of Xilinx's FPGAs in terms of density and shows the time when the tools supporting DPR in a beta version first appear and when they were commercially supported.

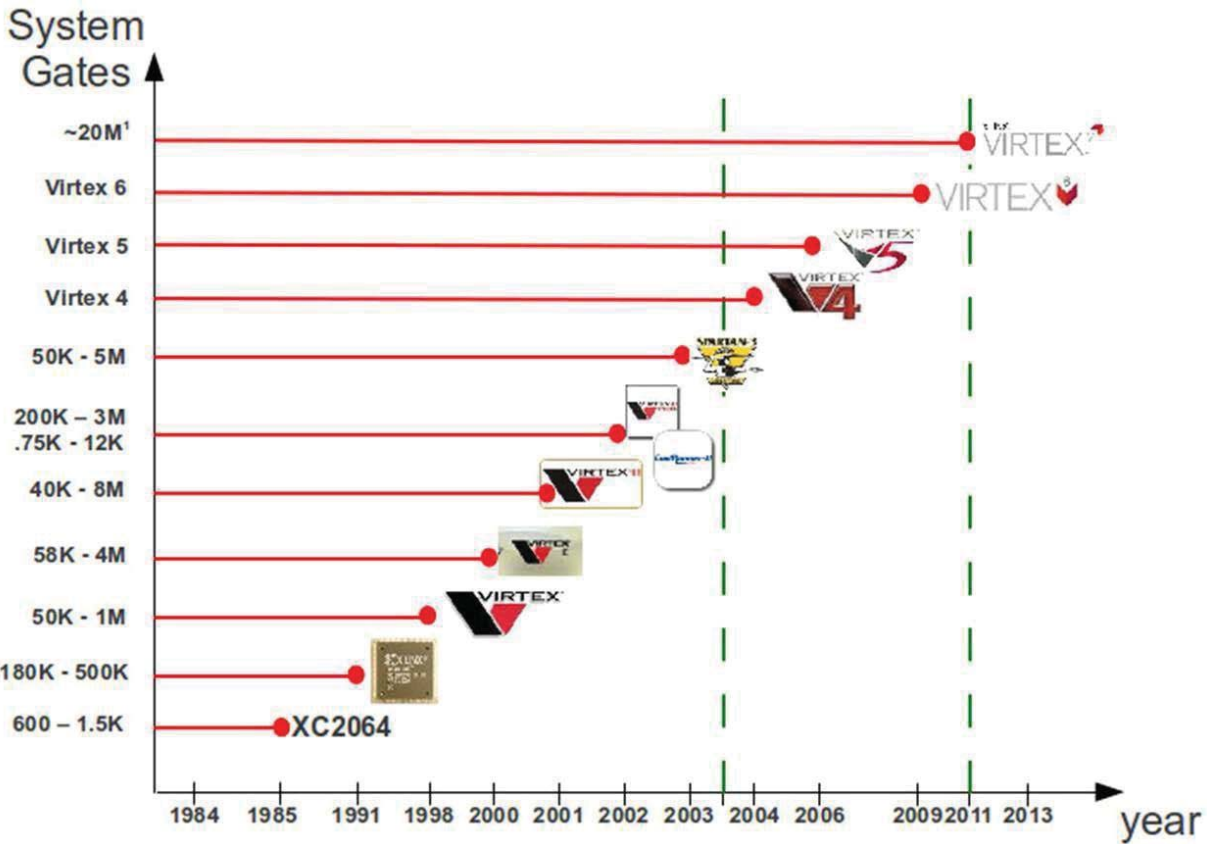


Figure 32. Evolution of Xilinx's FPGAs in terms of density. First green line shows the time where the tools supporting DPR in a beta version first appear. Second green line shows the time when they were finally commercially supported.

DPR has many implications and applications. The first implication is that it allows multiplexing in time functional units within the FPGAs, as long as their use doesn't intersect in time. In theory, this extends the FPGA's physical resources infinitely. The second implication is that DPR can mitigate FPGA's high power consumption (by trading off performance and power) and the occurrence of SEU's.

UNM has explored some of the applications that both of these implications opened up. A PhD dissertation explored the options of trading off performance and power consumption while implementing basic arithmetic operations [9]. Also, an ongoing project is exploring DPR's capabilities to mitigate SEUs. Both projects efforts are described in detail in the following sections.

### 2.1.1 A Dynamic Dual Fixed-Point Arithmetic Architecture for FPGA

In the realm of embedded systems, a designer often faces the decision of what numerical representation to use and how to implement it. Particularly when using programmable logic devices, constraints such as power consumption and area resources lead to traded offs with performance requirements. Floating point is still too expensive in terms of resources to be intensively used in programmable logic devices. Fixed-point is less expensive but lacks the flexibility to represent numbers in a wide range. In order to increase the numerical range, several fixed-point units—supporting different number representations—are required. Alternatively, the numerical range can be increased by a single fixed-point unit that is capable of changing its binary point position.

In this work, DPR is used to dynamically change an arithmetic unit's precision, operation, or both. This approach requires intensive use of partial reconfiguration making it particularly important to take into consideration the time it takes to reconfigure. This time is commonly referred to as the reconfiguration time overhead. Usually, runtime reconfigurable implementations involve the exchange of relatively large functional units that have large processing times. This, along with low reconfiguration frequencies, significantly reduces the impact of the reconfiguration time overhead on performance.

Reconfiguration time overhead is quantified by taking into consideration the bitstream size and the data transfer speed of the configuration circuitry [10-12]. SelectMap and ICAP are the external and internal parallel reconfiguration ports for Xilinx FPGAs, respectively [12]. SelectMAP provides an 8-bit or 32-bit bidirectional data bus interface to the Virtex 4 configuration logic that can be used for configuration and read back at an operation frequency of 100 MHz. ICAP has 32-bit wide input and output data buses and is also set to run at a maximum frequency of 100 MHz. A maximum theoretical speed at which data can be transferred into the configuration memory using ICAP or SelectMAP is 3.2 Gb per second.

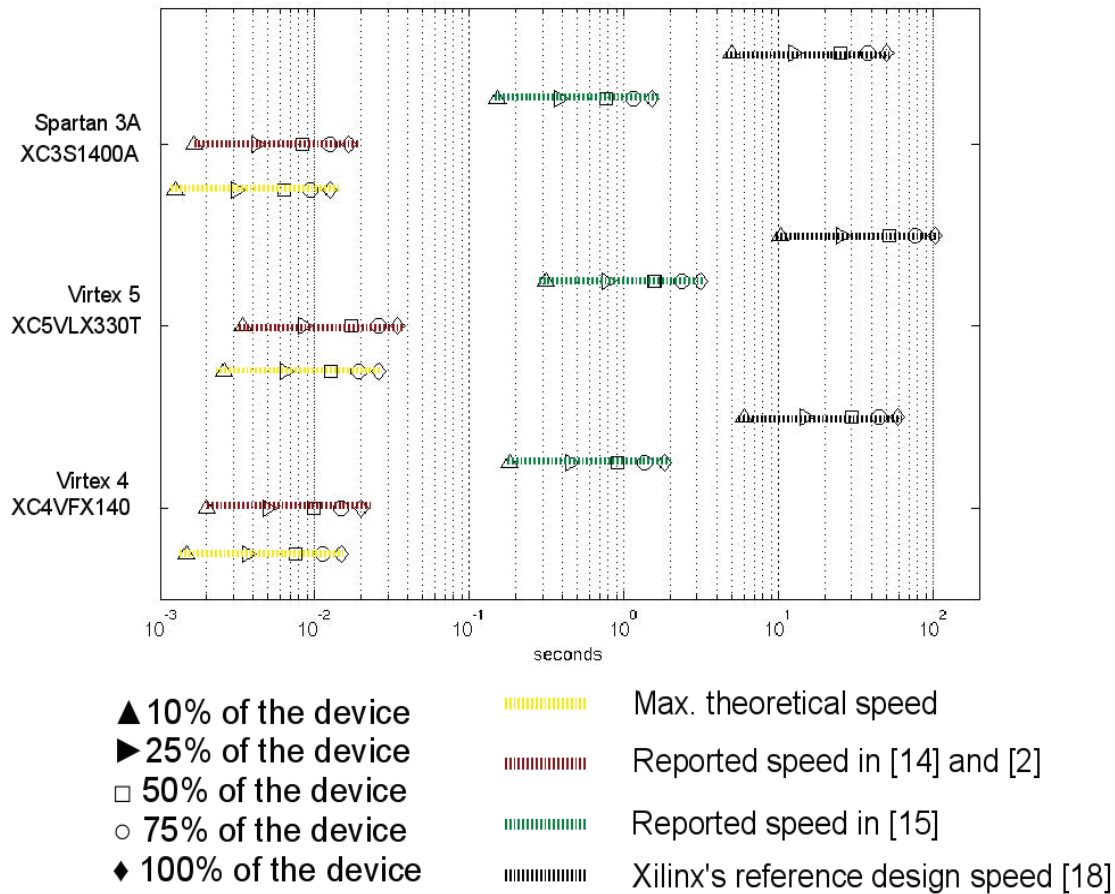


Figure 33. Virtex 4 family partial reconfiguration times. Points were calculated using the bitstream sizes reported in the device's datasheets and a theoretical maximum reconfigurable speed of 3.2 Gbits/sec [16].

The smaller unit of reconfiguration is called a frame. In Virtex 4 devices, a frame corresponds to a bit-wide column of 16 CLBs. All Virtex 4 configuration frames consist of forty-one 32-bit words resulting in a total of 1312 bits per frame. The bitstream size per each reconfigurable region in a device is calculated by the number of frames (CLB columns) it contains. Figure 33 depicts the time it takes to reconfigure fully or partially (horizontal axis) different devices in the Virtex 4 family (vertical axis), assuming the maximum reconfiguration speed of to be 3.2 Gb per second.

Unlike common runtime reconfigurable implementations, the exchangeable functional units in the approach taken by this project are smaller and reconfiguration frequencies are larger. Smaller exchangeable functional units are possible by using a dual fixed-point (DFX) numerical representation [17]. This provides larger dynamic range than classical fixed-point representations with little extra cost in terms of hardware. In this project, the COSMIAC team introduced a dynamic dual fixed-point (DDFX) architecture that allows changes in precision (binary point position) based on relatively small changes at runtime in the hardware implementation. Although at a higher reconfiguration time cost, DDFX allows the dynamic swap between different arithmetic operations. The improvements in dynamic range and precision allow this approach to find potential applications in problems where only a floating-point solution made



sense. Numerical optimization algorithms are examples of such applications. The iterative nature of these algorithms makes them especially susceptible to numerical issues arising from the use of fixed-point arithmetic as their required precision is dependent on the number of iterations or loops. Figure 34 depicts a precision variation curve introduced in References [18] and [19].

$$XXXXXXXXXXXXXXXXXX (XX) \diamond XXXXXXXXXXXXXXXXXXXX (YY) + \log(NN+1) \quad (1)$$

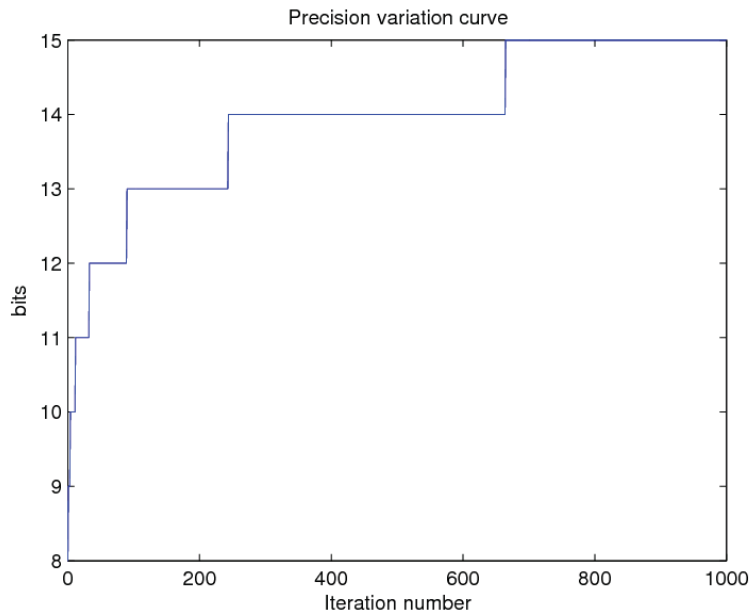


Figure 34. Precision variation curve for equation 2.1.

Furthermore, these algorithms usually require extensive calculations, making them optimal candidates for performance speed-up through parallelization. In that sense, the smaller hardware footprint of COSMIAC's approach is an advantage as it allows a larger number of DDFX units as opposed to a reduced number of larger floating-point units.

### 2.1.1.1 Dual Fixed-Point (DFX)

In this format, an n-bit number is divided into two components. The most significant bit is used to represent the exponent while the other bits are its significant complement.

$$DD = \diamond \begin{matrix} -pp0 \\ XX * 2^{-pp1} \end{matrix} \quad PPii \quad EE = l \diamond \quad (2)$$

Where

$$EE = \diamond \begin{matrix} 0 \quad PPii < -3/3 \quad \diamond DD < /3/3 \\ 1 \quad PPii \quad DD < -/3/3 \quad PPPPDD \diamond /3/3 \end{matrix} \quad (3)$$

### 2.1.1.2 *Dynamic Dual Fixed-Point (DDFX)*

COSMIAC's team researched the implementation of cores to perform arithmetic operations using the DFX format. This core library was labeled Dynamic Dual Fixed-Point (DDFX) as the cores' architecture was optimized to support DPR, to morph into different arithmetic operations and different radices, supporting a wide range of numbers.

#### 2.1.1.2.1 DDFX Adder

When performing addition of DFX numbers, both operands must be aligned with respect to the binary point. Thus, a prescaler section is required to analyze the operands and to determine in which range  $\mathbb{P}_o$  or  $\mathbb{H}$  one must perform the addition. Depending on the range in which addition is performed, one may need to scale an operand to a different range. This is done by executing an arithmetical shift to the right on the operand. A set of multiplexers are then used to select between an unchanged operand or its shifted version. Note that a truncation is performed when an operand is scaled. The bits discarded by the truncation are saved using a third multiplexer. These bits can later be recovered into the final result if the output range allows it. A block diagram is depicted in Figure 35.

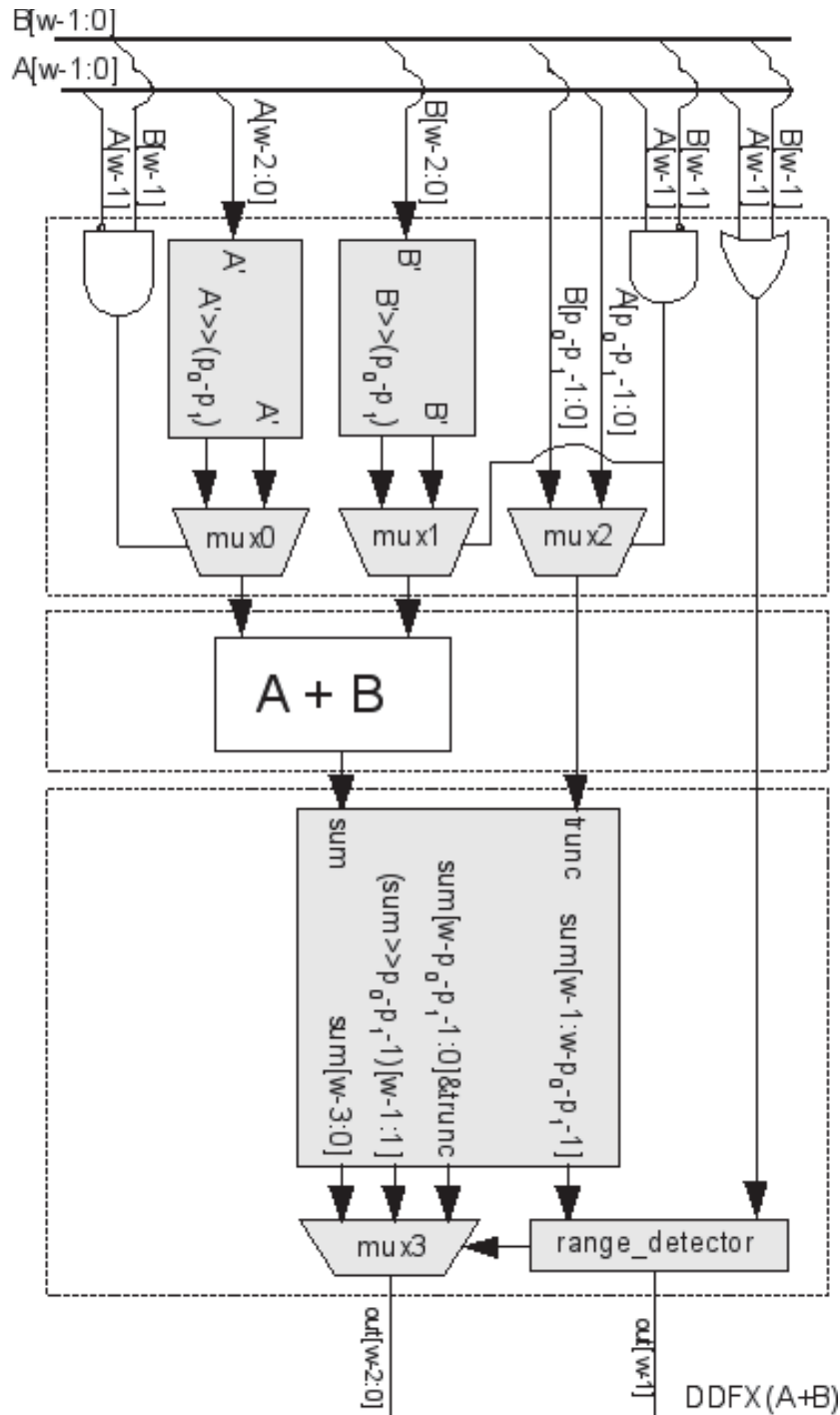


Figure 35. DDFX Adder

The prescaler produces two operands that are handed to the second stage, a full binary adder. This adder does not allow for a carry in, but it does produce an overflow signal. The adder's output is given to the third and last stage, the postscaler. In this stage, one decides in which radix the result will be presented and scales the output if needed. Scaling is again performed via shifting and multiplexers. Note that the adder described in this section can easily be transformed into a subtractor by including a 2's complement block for the subtrahend operand. When analyzing the block diagram in Figure 35, one realizes that precision is handled only by

the control blocks shown in gray. The adder itself does not deal with precision and only requires the operands' binary point to be aligned. Thus, it is possible to change the adder's precision and dynamic range by dynamically changing the control blocks. The segmentation of the functional unit in control (dynamic) and operational (static) parts is the basic idea behind the architecture. Since the dynamic part represents less than 40% of the whole adder, this segmentation allows for the reduction in the amount of logic required to reconfigure the precision. As a consequence, the reconfiguration time overhead is also reduced.

#### 2.1.1.2.2 DDFX Multiplier

In the case of multiplication, DFX operands do not need to be aligned with respect to the binary point before performing the operation. Thus, a prescaler section as the one in addition is not required.

A block diagram of the multiplier is depicted in Figure 36. The diagram is divided in two stages. The first stage is a full precision 2's complement, binary multiplier. The second stage, a postscaler, takes the multiplier output and performs an analysis similar to the adder's postscaler case. However, in the case of the multiplier, no shifting is required. Only bit slicing is performed. Similar to the adder, the blocks that control precision are shown in gray. The multiplier itself does not deal with precision. The section of the multiplier that one needs to change in order to change the multiplier's precision represents less than 30% of the overall logic resources used by the multiplier.

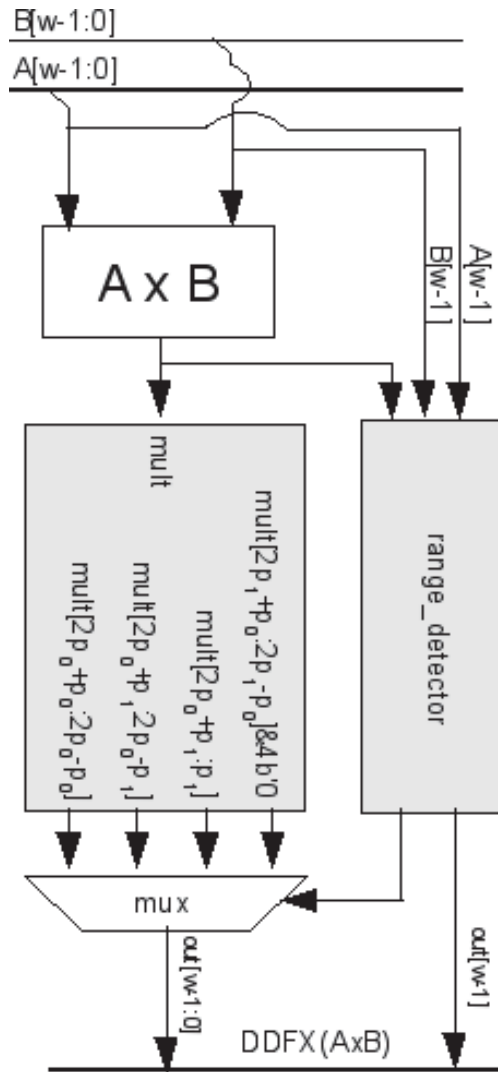


Figure 36. DDFX Multiplier.

### 2.1.1.3 DPR and the tradeoff between performance and power consumption

A DDFX core library was setup to perform experiments to explore the trade off between performance and power consumption using DPR and the impact of the reconfiguration time overhead while doing so. The goal of the experiments was to quantify the potential savings in power consumption, the impact of reconfiguration time overhead and the range maximum performance attainable.

Results are presented in terms of performance, and power consumption. Comparisons are made against hardware implementations of single-precision floating point (SFPU) and fixed-point (FX) with similar precision and fixed data width (32 bits). Results for GCC software emulation of single-precision floating point (SWFP) are also presented when appropriate.

The systems under test (SUT) were implemented in Virtex II Pro and Virtex 4 devices for the purpose of gaining insight on the impact that the differences between both families have over

reconfiguration time overhead. The general characteristics of the SUTs defined are:

- CPU speed of 100 MHz,
- CPU's cache enabled,
- DDR2 memory as main memory,
- UART for communication and control,
- Access to the device's internal configuration ports (ICAP),
- CPU's capacity to support user-defined coprocessors,
- Floating-point unit (FPU) support,
- Peripheral bus speed of 100 MHz, and
- Timer with one clock cycle resolution.

The speed of 100 MHz for the CPU and the peripheral bus was chosen to facilitate fair comparisons between the devices. Virtex 4 devices support a larger frequency of operation than Virtex II Pro. A more accurate comparison is feasible at lower operating frequencies since the level of effort the compiler requires to accomplish this frequency is similar for both devices. A major difference between the Virtex II Pro and the Virtex 4 SUTs is that a MicroBlaze microprocessor was used for the Virtex II Pro while a PowerPC was used for the Virtex 4. The MicroBlaze was picked in the case of the Virtex II Pro because the PowerPC 405D5 available on these devices does not support as tight coprocessor integration as the PowerPC 405F6 on the Virtex 4 does.

#### 2.1.1.4 Performance

Performance was of the arithmetic units set as coprocessors running a series of test programs targeted towards vector operations (addition, subtraction, and multiplication). The average performance is measured as

$$\frac{\#00ppPPPPff00PPPPPPPP}{PPPPPPiiPPPPffffPPPPPP} \quad 00_{opp} + 00_{rrrrroorrrriir} \quad (4)$$

$$=$$

where  $00$  represents the time required to perform the arithmetic operations while  $00_{rrrrroorrrriir}$  represents the time it takes to reconfigure the precision or the operation. First, the static performance or  $00_{rrrrroorrrriir} = 0$  was measured. In order to assess the setup's scalability, the measurements were taken under different load distribution schemes. Next, the whole computation was performed by a single DDFX arithmetic co-processor. Then the computation was distributed among 2, 4, and 8 coprocessors. A coarse interleaving scheme was used for distributing the data among the coprocessors. Note that in terms of logical resources used, one Single Point Floating Unit (SFPU) core is equivalent to 4 DDFX core [9]. Thus, a like-for-like comparison in terms of performance, keeping resources fixed should be done between an SFPU core and 4 DDFX cores. Figure 37 depicts the performance measurements for all four setups (1, 2, 4, and 8 DDFX cores) and all three operations implemented (addition, subtraction, and multiplication) and compares them to the SFPU and software-emulated FP alternatives. In the case of DDFX, the latency for addition, subtraction, and multiplication was

the same. Thus, the performance, and the curves presented in Figure 37, are the same—in each device family—for all three operations. Note that the speedup was not linear with respect to the number of cores. This was due to saturation in the FSL bus as more cores connect to it.

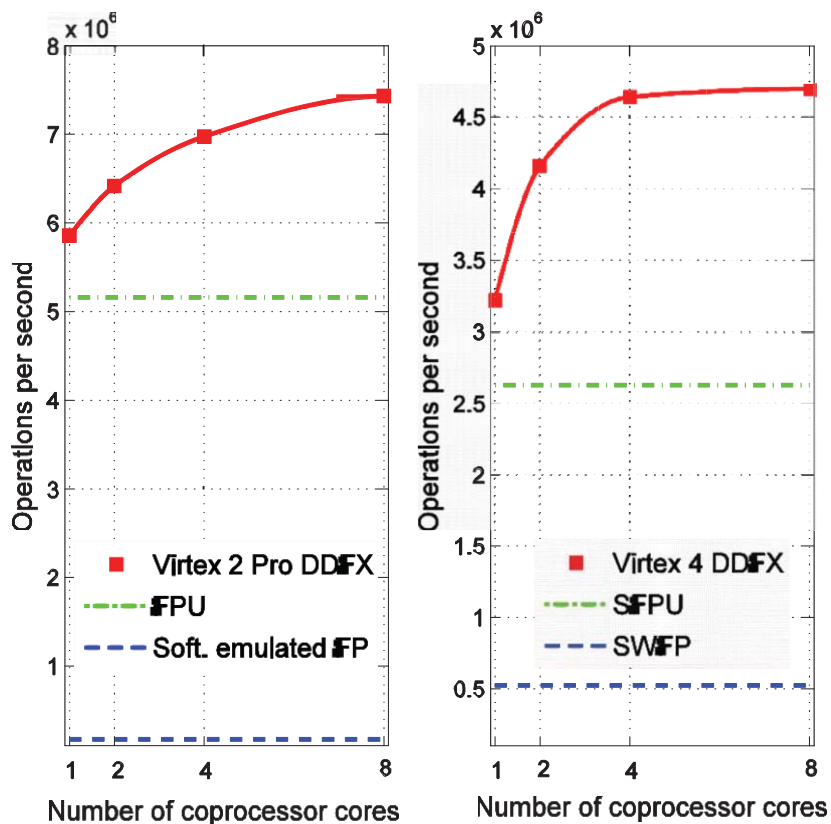


Figure 37. Multi-DDFX core performance for Virtex II Pro MicroBlaze (a) and Virtex 4 PowerPC405 (b).

In these graphs, performance for SWFP and SFPU are shown as constants at their best single core's performance. In the case of SFPU, the multiplication was on average  $\approx 5.5\%$  slower than addition. In the case of software-emulated floating point, the difference between addition and multiplication was almost imperceptible for the PowerPC. However, in the MicroBlaze case, the difference was significant. This was primarily due to differences in the compiler (ways to emulate floating point) and the differences between the instructions available to each processor. Also of note was that the MicroBlaze implementation outperforms the PowerPC. The reason for this difference lies in the level of integration between the processors and the FSL bus used to connect the coprocessors. The FSL bus was originally designed for the MicroBlaze, and it is tightly integrated with this core. In the case of the PPC, a bridge is needed for adding latency to FSL operations. DDFX outperforms the SFPU and SWFP alternative in both families. Moreover, since an SFPU unit is equivalent to 4 DDFX cores in terms of resources (i.e., in the case of the adder), a DDFX outperforms SFPU by about 40% in the Virtex II Pro case. For Virtex 4, 4 DDFX cores outperform SFPU by about 80%.

Next, reconfiguration was added, and the performance of a single core was measured. As mentioned before, all three operations have the same latency. Also, the bitstreams to partially reconfigure both, their precision control section and the operation itself, are the same size.

Although counter-intuitive, the multiplication requires more logic resources. The bitstream sizes are the same because the partial reconfiguration regions allocated for these sections are similar in area. This was independent of the percentage of resources used in each region. However, the architectural changes between the Virtex II Pro and the Virtex 4 families made possible for the areas (i.e. the bitstreams) to be smaller in the case of Virtex 4. This, as Figure 38 depicts, represents a significant difference in the way performance drops for both families, when reconfiguration was included in the measurements. Dynamic reconfiguration can be triggered by the arithmetic cores asserting a signal that will be treated as an interrupt by the microcontroller. In these examples, reconfiguration was hardwired in the testing code to occur after a fix number of operations. This allowed for emulating different reconfiguration frequencies.

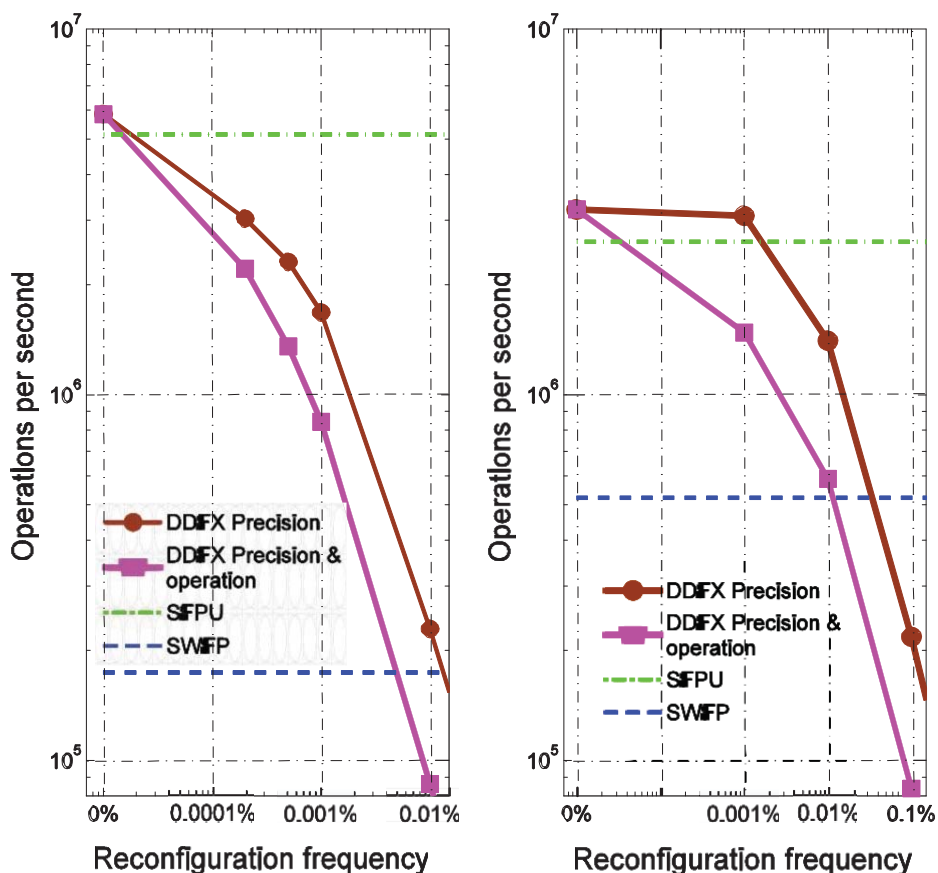


Figure 38. Performance results for a single DDFX core versus reconfiguration frequency for Virtex II Pro MicroBlaze (a) and Virtex 4 PowerPC405 (b). Here, 0% can not be represented in our logarithmic scale, thus it was included as the leftmost point in the graph.

Several observations are made from the graph. First, when both precision and the operation itself are reconfigured, the performance drops faster than when only the precision is changed. This is a straightforward consequence of the larger bitstream used when both precision and operation are reconfigured. Secondly, a sharper drop in performance is observable for the Virtex II Pro as compared to the Virtex 4. This is a direct consequence of the architectural changes that allowed smaller bitstreams on the Virtex 4. Finally, the graph shows that in the case of Virtex II Pro, the solution is outperformed by the SFPU even with very small reconfiguration frequencies.



In the case of the Virtex 4, changes of precision can be made with a frequency of 0.001% while still outperforming the SPFP alternative. In context, one could calculate a matrix vector multiplication  $AA_{xx}$ , where A is a matrix  $100 \times 100$  and x is a vector  $100 \times 1$ , reconfigure the precision of the operation once, and still outperform the SFPU alternative. In terms of the number of operations, 0.001% is the maximum reconfiguration frequency possible while still outperforming the SPFP alternative. This term is used throughout the paper as an upper bound.

Similar to the static case, a like-for-like comparison in terms of resource consumption requires comparison of performances between one SFPU core and about 4 DDFX cores. Compared performance results where the load was distributed among 4 DDFX coprocessors and reconfiguration was performed. The results depicted in Figure 39 show that the maximum reconfiguration frequency has been reduced, thus more reconfiguration instances are possible while still outperforming the SFPU alternative.

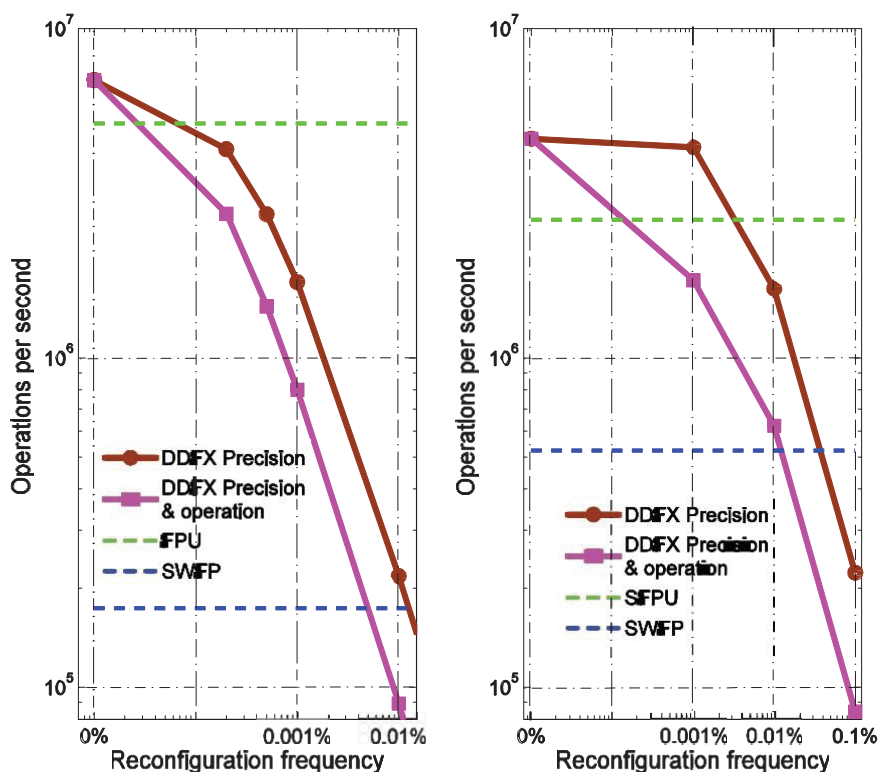


Figure 39. Performance results for 4 DDFX cores versus reconfiguration frequency for Virtex II Pro MicroBlaze (a) and Virtex 4 PowerPC405 (b). Here, 0% cannot be represented in our logarithmic scale, thus it was included as the leftmost point in the graph.

### 2.1.1.5 Power Consumption

Three types of power were considered: dynamic power, static power, and reconfiguration power. The dynamic power is due to the energy consumed by the device when it is doing useful work. Dynamic power is mainly dependent on the number of transitions in the logic gates' transistors (frequency of operation) and the gate count of the design in question. Some design decisions can help to keep this power in budget (including different number formats). Consider

the performance curves shown in Figure 40 (left) and the corresponding power consumption curves presented in Figure 40 (right). The relationship between these curves suggests that the dynamic approach can be exploited by a system to dynamically tradeoff performance for power consumption.

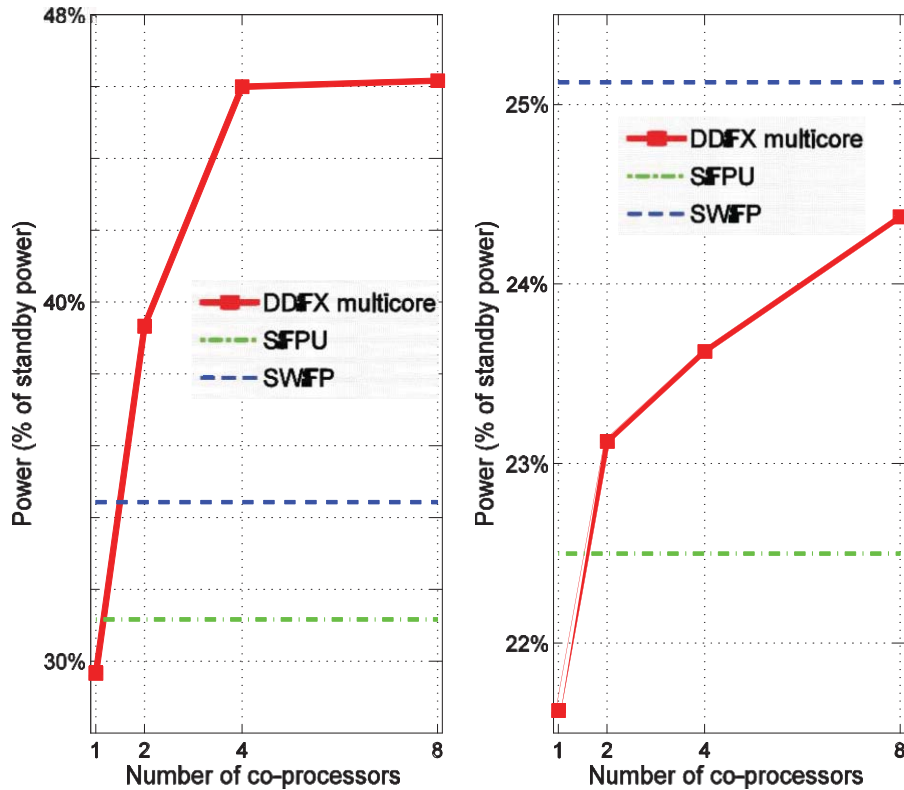


Figure 40. Dynamic power consumption for Virtex II Pro (a) and Virtex 4 (b).

The data in Figure 41 was obtained by physically measuring the current at the FPGA core during the execution of especially crafted test programs. In both graphs, the values in the axis Y are expressed as a percentage of the processor power consumption while in standby.

Static power is due to the current drawn by the device after being powered up and configured but while doing nothing. This power is due to the leak current in the transistors' thin oxide layers. As the number of transistors increases, and fabrication techniques allow for smaller geometries, the current usually increases. The increase in current will result in heat dissipation that will in turn result in more leak current. Static power has been and still is a major problem for the FPGA technology.

Experimentation was conducted using partial reconfiguration to shut off different sections of a system at different times, without affecting the rest of the FPGA. For this purpose, shift registers of different sizes, built upon the SRL16 primitive (basically LUTs), were connected to both SUTs as customized cores. The experiment consisted of shutting off one peripheral at a time and measuring the current at the FPGA core every time. The shut-off of a core is done by reconfiguring the section that the core occupies with a “blank” bitstream (equivalent to having

nothing programmed in that section).

The test results showed a linear relationship between the amount of resources used and static power consumption. Figure 41 depicts the test results interpolated with the resources used by common peripherals in an embedded system. As in the case of dynamic power, these results suggest the dynamic approach could be exploited to tradeoff performance for power consumption. The idea is extendable to any reconfigurable system. Considering the amount of time a peripheral could be idle, the savings in static power are considerable.

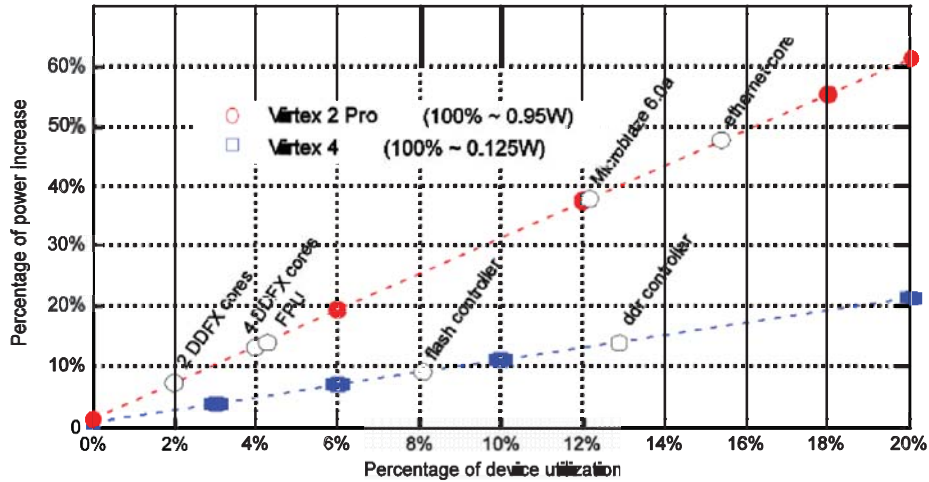


Figure 41. Virtex II Pro and Virtex 4 static power consumption and interpolated power consumption of commonly used peripherals in embedded systems. The red circles and blue squares represent data points obtained through direct measurements.

The overall power consumption is depicted in Figure 42. This graph was obtained unifying the results of dynamic and static power consumption. A definite advantage of DDFX over SWFP and SFPU alternatives can be observed in the case of the Virtex 4 while, in the case of the Virtex II Pro, no clear advantage is observable. This discrepancy can be explained by the architectural and technology differences between both families.

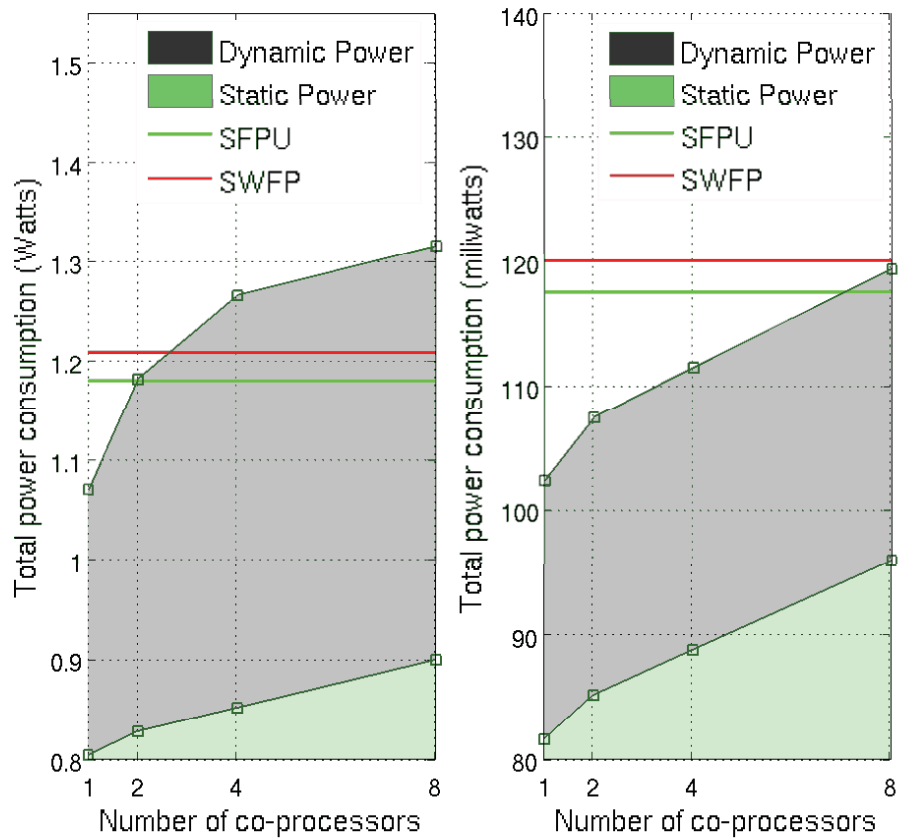


Figure 42. Virtex II Pro (a) and Virtex 4 (b) total power consumption (dynamic + static).

### 2.1.2 Conclusions

An arithmetic architecture that used DPR to dynamically morph and adapt performance and power consumption was evaluated. For the newer Virtex 4 FPGA devices, testing results show significant advantages of the proposed approach over alternative approaches. Keeping logical resources and precision equivalent, DDFX shows a performance advantage in excess of 50% with respect to alternative floating point and software-emulated floating point when no reconfiguration is performed. The maximum reconfiguration frequency, while keeping performance comparable to a floating point unit, is 0.001% (in the case of the Virtex 4).

For equivalent logical resources, precision, and performance (equivalent upper bounds), the proposed solution consumes 10% less power (in the case of the Virtex 4) than its alternatives. Experiments also showed power savings by dynamically managing the number of processing cores, effectively trading off performance by power consumption. Measurements show that the savings in overall energy consumption are significant. The approach is extendable to more general embedded systems, where a dynamic reconfiguration scheme can be used to shutdown unused cores.

## **2.2 A DPR capable space-borne processor with SEU mitigation**

FPGA-based reconfigurable systems' popularity for space-based applications has grown considerably due to their flexibility and the ability to multiplex in real time, different hardware configurations. A wider utilization of commercial off the shelf (COTS) FPGAs – especially in aerospace applications - is limited by their susceptibility to Single Event Upsets (SEUs). The second project performed by COSMIAC consisted of exploring DPR to mitigate the effects of SEUs.

The traditional approach for Single Event Upset (SEU) mitigation on commercial parts consists of triple modular redundancy (TMR). Although proven effective, this method adds a certain amount of logic overhead and a penalty in power consumption and processing speed. This approach is also vulnerable to multiple bits upset that are becoming more frequent as geometries decrease in modern devices. An alternative approach, called “scrubbing”, relies on simply reloading the configuration memory frames at defined time intervals. This approach is possible in the case of FPGA devices that support DPR. Scrubbing provides protection against the accumulation of upsets in the configuration memory and, in combination with TMR, improves the overall system's reliability.

### **2.2.1 Scrubbing an FPGA's configuration memory for SEU mitigation**

Different alternatives for the design of a scrubbing system can be devised by varying the configuration memory access port used, the scrubber position with respect to the system being scrubbed, and the strategies used to monitor and scrub the configuration memory. The suitability of any of these design variations to a certain application is driven by factors such as mission criticality, hardware, and power limitations. The ideal scrubbing solution would be extremely flexible to support its deployment on every possible scenario. [3]

The basic functionality of a scrubber is to access the configuration memory to overwrite it periodically with a known good copy of the original bitstream to correct any possible bit flip. This method is called “blind scrubbing”. A more advanced method is to read back the configuration memory and detect errors to selectively overwrite sections of the configuration memory instead of blindly scrubbing. Blind scrubbing is the most common implementation for scrubbers reported in literature. Its main advantage is simplicity, since overwriting the configuration memory without discrimination is a relatively simple task. This method has two substantial drawbacks. First, blindly scrubbing has a significant overhead of power consumption (writing the configuration memory consumes measurable power and performance). Secondly and more importantly, since no discrimination is done as to what sections of the configuration memory are scrubbed, some bits that are upset may remain in the wrong state for long times while waiting for the scrubber to complete its turnaround of the whole configuration memory. In some instances, blind scrubbing may also prolong a system's down time while the memory space is scrubbed.

Reading back the configuration memory to detect errors for scrubbing is a more complex but efficient task. Selective scrubbing saves power and allows the designer to implement scrubbing

strategies that provide higher reliability to critical sections of a system and lower reliability to less important sections. Additionally, reading back the configuration memory to find errors provides valuable statistics in space borne applications[3].

### **2.2.2 Space-borne processor architecture**

Several processors have been used in space-borne applications, with different architecture performance characteristics. A common denominator of all processors used is that they are all licensed and have a cost. Within all reported processors, the COSMIAC team was particularly interested in RTL descriptions that can be portable to different implementation platforms (ASICs, FPGAs). Two more notable examples in this category of space-borne processors are OpenRISC and Leon3. The first is a 32-bit RISC processor developed as an open source project. The second is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. Both processors' full source code descriptions are available under the GNU GPL license, allowing free and unlimited use for research and education. OpenRISC is free of cost even for commercial applications. LEON3 is available under a low-cost commercial license, allowing it to be used in any commercial application with a relatively low cost. For space-borne applications, both processors have been modified to include SEU mitigation features in their architectures.

### **2.2.3 LEON3FT**

Aeroflex Gaisler's LEON3 commercializes a Fault Tolerant (FT) version of the core. It has been designed for operation in the harsh space environment, and includes functionality to detect and correct (SEU) errors in all on-chip RAM memories. The LEON3FT processor supports most of the functionality in the standard LEON3 processor, and adds the following features:

- Register file SEU error-correction of up to 4 errors per 32-bit word
- Cache memory error-correction of up to 4 errors per tag or 32-bit word
- Autonomous and software transparent error handling
- No timing or performance impact due to error detection and correction

LEON3's fault-tolerance in LEON3FT is implemented using ECC coding of all on-chip RAM blocks. The general scheme is to detect and correct up to four errors per 32-bit RAM word. In RAM blocks where the data is mirrored in a secondary memory area (e.g. cache memories), the ECC codes are tuned for error-detection only. A correction cycle consists of reloading the faulty data from the mirror location. In the cache memories, this equals an invalidation of the faulty cache line and a cache line reload from main memory.

In RAM blocks where no secondary copy of the data is available (e.g. register file), the ECC codes are tuned for both error-detection and correction. The focus is placed on fast encoding/decoding times rather than minimizing the number of ECC bits. This approach ensures that the FT logic does not affect the timing and performance of the processor, and that LEON3FT can reach the same maximum frequency as the standard non-FT LEON3. The ECC encoding/decoding is done in the LEON3FT pipeline in parallel with normal operation, and a correction cycle is fully transparent to the software without affecting the instruction timing. [20]

## 2.2.4 OpenRISC-FT

AAC Microtec took the OpenRISC architecture and added fault tolerant features meant to mitigate the effects of SEUs. There is very little public information available on what those features are, but it is likely to incorporate similar features than LEON3-FT. Additionally, AAC Microtec offers a myriad of peripherals as a core library, to build different configurations of the processor. Figure 43 shows a block diagram of the OpenRISC-FT offered by the ACC Microtec Corporation.

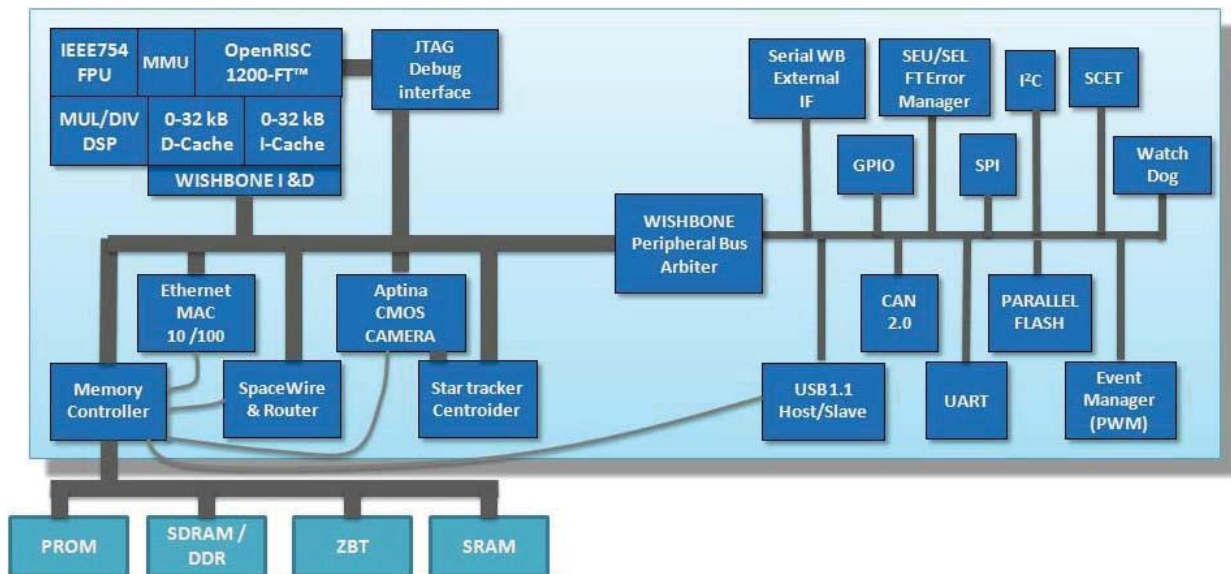


Figure 43. AAC Microtec's OpenRISC-FT.

## 2.2.5 COSMIAC's Alternative space-borne processor: DR-Open RISC

The fault tolerant versions of both AAC Microtec processors just described are available under a costly license. COSMIAC initiated a project to develop a fault tolerant processor that would be open sourced (zero cost). The logic behind having a space-borne processor that is open source, like in any open-source project, is to access a large community of developers that could help improve the overall quality of the processor's hardware description. Additionally, a no-cost design might become a de-facto standard within the community, promoting interoperability between components and possibly a large eco-system designers can pick from when designing for a new space mission. All these advantages have the potential of allowing the community to reduce the time it takes to design, develop, and integrate a spacecraft.

COSMIAC chose the OpenRISC architecture as it has the larger community support and the most solid tool-chain currently available as open source. The initial setup used to develop COSMIAC's space processor is depicted in Figure 44. This setup is the one used for OpenASIM (see Section 1.1).

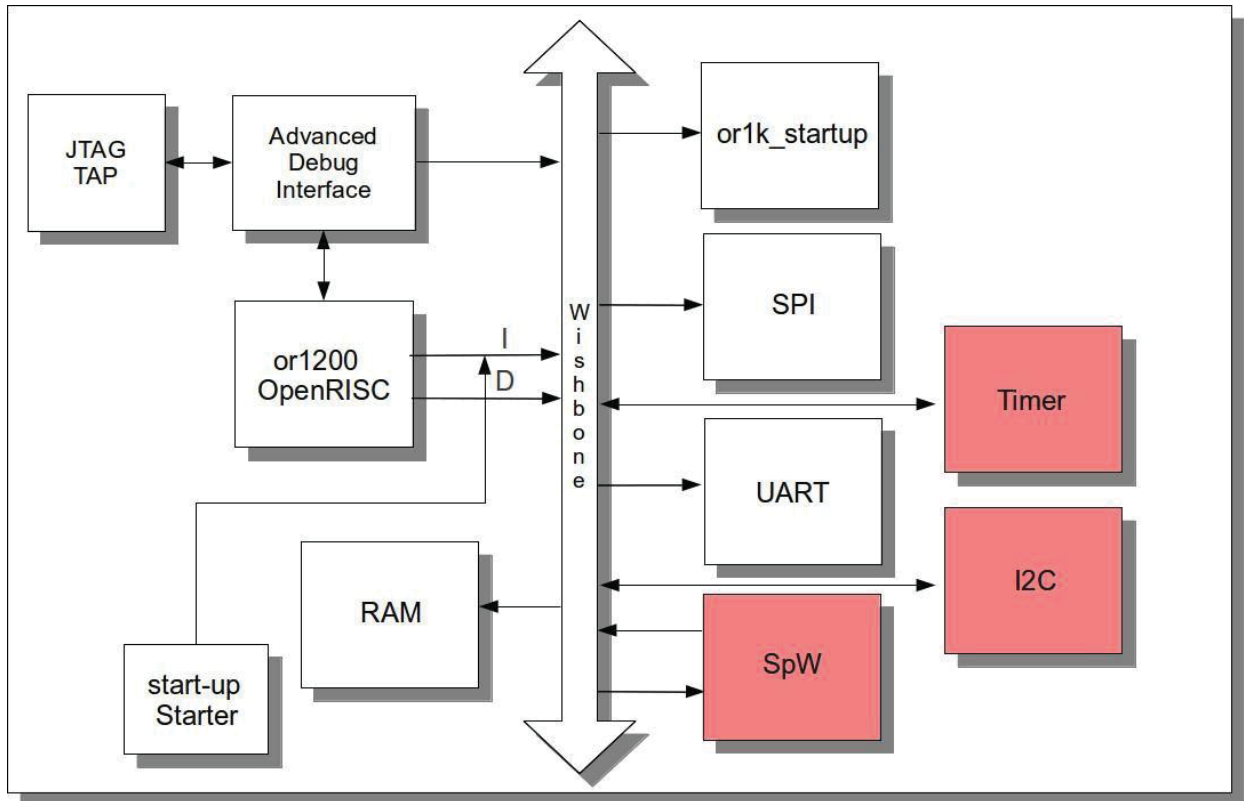


Figure 44. Initial setup of the processor used for UNM to build a space version.

Developing the space version of this processor consists of two major tasks: (1) addition of a fault tolerant mechanism in its memory structures to mitigate the effects of SEUs, and (2) addition of DPR capabilities to the processor to deal with SEUs within an FPGA’s configuration memory (classic hardware platform for processors whose hardware description is available) and to allow power consumption and performance tradeoff on the fly.

The second task is substantially different from the processor described and other available cores, besides the cost. Using DPR to deal with SEUs in the configuration memory is akin to the scrubbing process described in earlier sections of this report. Using DPR to allow the processor to adapt to different performance requirements and different power availability levels is akin to the COSMIAC Dynamic Dual Fixed-Point Arithmetic study. The processor UNM is building is the result of years of research in the realm of DPR. It is because of this second task that UNM’s processor is labeled Dynamically Reconfigurable (DR) OpenRISC.

### 2.2.6 Enabling DPR on OpenRISC

Currently COSMIAC is focusing on enabling DPR on an OpenRISC architecture. The first step in this direction is allowing the CPU to access the Internal Configuration Access Port (ICAP) of the host FPGA. This allows the processor to deal with errors in its own memory via a scrubbing process using DPR. It also allows the processor to swap in and out processing units in response to variation on performance requirements or power availability. Xilinx provides a means for a processor to access the ICAP through a core named HWICAP as part of their libraries. HWICAP is proprietary and only compatible with the bus architecture used by Xilinx’s embedded



microprocessors. OpenRISC uses an open standard named WishBone as a bus architecture, which is not compatible with HWICAP. UNM has developed a HWICAP core compatible with WishBone. Figure 45 depicts a block diagram of this core, labeled WBICAP.

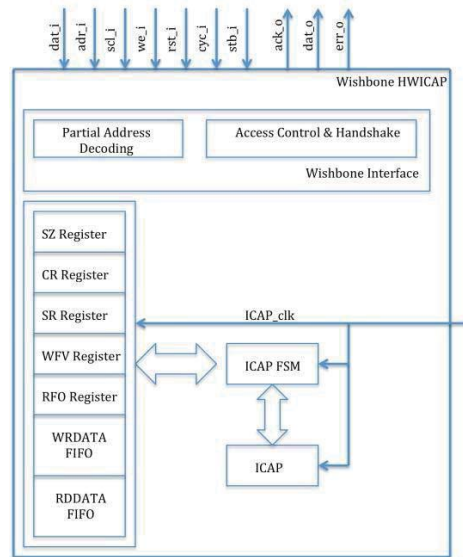


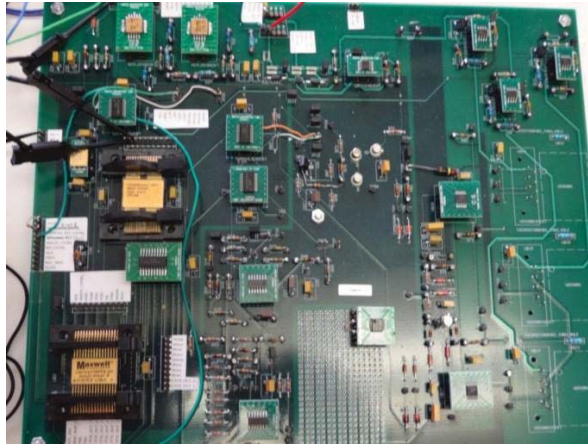
Figure 45. WBICAP's block diagram

UNM is currently developing the low level software routines that would enable a designer to seamlessly exploit the full potential of DPR.

### 2.3 Additional Board Development

COSMIAC was an active participant in the Xilinx Radiation Test Consortium (XRTC) and its associated radiation test planning. COSMIAC's role focused on the use of open-design guidelines such as Space Plug-n-Play, as well the use of small spacecraft to validate the designs.

The design and fabrication of the development board for the Radiation Hazard Awareness Sensor (RHAS) for detecting ionizing radiation in a geosynchronous orbit was completed in 2012. Figure 46 is a photograph of the board as configured for testing. Results of the tests indicated that the microcontroller had insufficient drive strength to reliably activate any portion of the instrument with an impedance less than 500 Kohms. There were enough unused gates in the design to buffer the microcontroller output pins without adding additional part packages. Other design changes included increasing the value of the voltage divider resistors used in the power switching networks to reduce the current required to turn-on the instrument and increasing the value of capacitance in the turn-on delay circuit. Two errors were found in the layout: (1) a net was misnamed causing a faulty connection within the AND gate buffer, and (2) the “minus” and “adjust” pins on the LM136 analog reference were interchanged. Both errors were corrected with wiring patches on the board. One section of the power-on delay circuit was found to be unnecessary; consequently, one LM139 analog comparator was eliminated from the design. All the changes were incorporated in an engineering board.



**Figure 46. Development board for the Radiation Hazard Awareness Sensor**

Another (RHAS) completed design was composed of three Teledyne UDS001 sensors arranged on a modified CubeSat PUMPKIN printed circuit board. The outputs of the sensors are converted to digital signals with Texas Instrument's ACD128S102 analog-to-digital converters controlled by an Aeroflex UT69RH051 microcontroller based on the 8051 instruction set. The microcontroller also communicates with the ARCS system through HCPL6530 opto-isolators. Other functions included on the RHAS include temperature and voltage monitoring, self-calibration circuitry, and voltage control. Six layers were required to fully route the board due to the UDS001 placements, which blocked surface routing channels.

### **2.3.1 Radiation Testing**

A test plan and test software were created for a latch-up screening test for the Cypress Semiconductor PSOC-3 system on a chip. The chip is under consideration for use as the controller for the RHAS dosimeter discussed above. The test that was conducted in early August 2011 included a latch-up screening test on the AFRL/RVSE flash X-ray machine followed by a Co-60 test.

A reference design was developed for a processing capability, based on the Actel ProASIC-3 FPGA, to control and take data for a space-borne ionized particle detector that will fly on C/NOFS. COSMIAC provided supporting material on the reference design for the spacecraft PDR. This effort was particularly valuable for three reasons: (1) it provided an opportunity for COSMIAC to gain expertise in a new line of FPGA, (2) it allowed COSMIAC to evaluate and prepare for the possibility of using the ProASIC line in small spacecraft electronics in the future because of its power and flexible implementation advantages, and (3) it provided an opportunity future collaboration with Angstrom Aerospace and their developments of SPA-compliant architectures (which run on Actel ProASIC).

The COSMIAC team developed an "experimental protocol" that was used to independently extract at least three different components:

1. Recoverable Charges
2. Field Recoverable Charges

### 3. Interface State Generation” of Negative Bias Temperature Instability (NBTI).

Measurement of NBTI (normally on a p-channel MOSFET) on n-channel MOSFETs were made to better understand these mechanisms. The results were submitted to the Nuclear and Space Radiation Effects Conference in 2013 and a paper for review for Journal of Vacuum Science and Technology. In addition, work was done in partnership with Center for Integrated Technology (CINT) at Sandia National Laboratory to develop the Junctionless Nano-Transistor.

The research was expanded to include organic photovoltaics by measuring the relaxation time, radiation effects, spectroscopy imaging, and photocurrents. The devices that were analyzed have expanded to different anode and cathode materials as well as different active blends. The AFRL/UNM team working on this problem published a paper in the Institute of Electrical and Electronics Engineering in December of 2012 and presented a poster at NSREC in July 2012.

### **3. Adapting, augmenting, verifying, and demonstrating commercially available FPGA and configurable systems design tools to accommodate unique space and defense requirements, including the verification of correct design, and ensuring of high system-level productivity.**

A concept was explored for an adaptive wiring manifold (AWM) which is a wiring harness that is reconfigurable and scalable for general applications. In principle, an AWM is similar to an FPGA that can be programmed to implement wiring patterns, except that the proposed adaptive harness allows for the routing of continuously variable analog, power, and microwave signals. In general, FPGAs can only manipulate Boolean signals.

Additionally, this section includes examples of using available FPGAs and configurable systems for satellite applications, and the work done in developing new materials and the design and fabrication of basic high speed digital test structures.

#### **3.1 FPGA-based Reconfigurable Adaptive Wiring**

##### **3.1.1 Adaptive Wiring Panel (AWP)**

In vehicular platforms, the network of wires that connect sensors and actuators to other electrical boxes is referred to as a wiring assembly or wiring harness. The wiring harness plays a critical role in distributing signals and power throughout the platform. These wiring harnesses are intensively custom, often expensive, and can take time to build (i.e. months). As such, the wiring harness can be a limiting factor in the time necessary to build a new reconfigurable system from scratch, even if all components and software are available for the new design. Conventional spacecraft wiring harnesses are built with architectures that are fixed at the manufacturer. By implementing reversible (meaning they can be changed repetitively) and dynamically programmable software wires, an “adaptive wiring manifold” can be formed. Adaptive wiring systems have many useful reconfigurable properties. They can, for example, be customized within seconds if the wiring configuration is known. They have tremendous flexibility in that they can be changed up to the last moment of a system’s development without removing components and performing painstaking rework. They also have the potential of self-healing and enhanced diagnostics through soft-definable probe signals.

##### **3.1.2 Conceptual Architecture**

This subsection describes a number of basic principles for adaptive wiring systems. To introduce the basic idea, an abstract adaptive wiring structure (Figure 48) that could be referred to as a panel or substrate contains a number of input/output (I/O) termini. These are shown in Figure 47(a) as connection points on the left (AI, BI, ...) and right (EO, DO, ...) edges. In the case of adaptive wiring, we can form connections between the termini “on demand”. We can supply a wiring “problem” that we wish to solve, a set of termini that we wish to connect together. Through some (not shown) control mechanism, we can convey commands into the panel that cause it to form “virtual” wire connections as desired. In Figure 47(b), for example, we show the solution of virtual wires needed to connect termini together bearing the same pre-fix label input

on the left edge to the corresponding label output on the right edge (e.g., “AI” connecting to “AO”, etc.). In general, the wiring problem to be solved can be referred to as a netlist specification. In fixed wiring systems, netlists are implemented physically in the form of a wiring harness. In this example, the adaptive wiring system, represented as a “cloud” within the substrate, forms the wiring dynamically (under program control). At this point, the adaptive wiring concept is notional, and we have not suggested how the “cloud” is implemented.

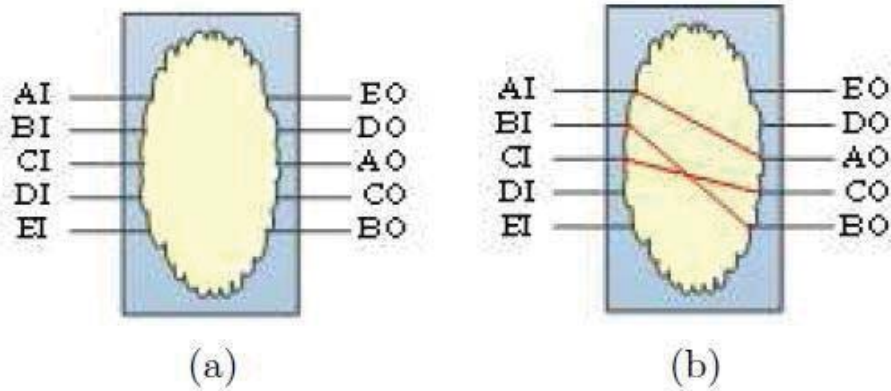


Figure 47. Basic concept of an adaptive wiring cell. (a) Substrate (yellow cloud) having a number of inputs and outputs defining a ‘wiring problem’. (b) Depiction of a possible wiring solution

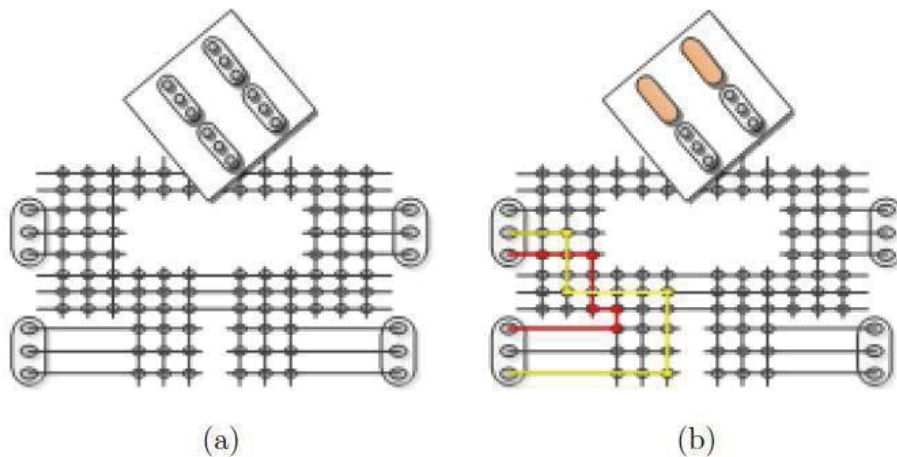


Figure 48. A physical wiring problem example. (a) Unprogrammed substrate, containing four sockets for components (modules). (b) Example placement of two modules and wiring of a two-net netlist

Figure 48 depicts a notional implementation concept to provide some intuition about how an adaptive wiring system might actually be implemented. In this case, the substrate takes on the aspect of a physical panel featuring four sockets where components or “modules” can be mounted (Fig. 48). To implement the amorphous “cloud” of wiring resources in Figure 47, a deliberate configuration is depicted consisting of a matrix of wires in rows and columns, with circles shown at the intersection points [21]. The circles represent electrical switches that, when closed, short together the associated row and column. We are not immediately concerned over the specific medium for switches. They could be, for example, metallic relays, solid-state switches, microelectromechanical systems (MEMS) devices, or combinations of these and other

switch types [22], [23]. Using such a fabric, implementing a solution to a particular wiring netlist amounts to closing a number of switches, as shown in Figure 48(b), which shows how a netlist problem having two “virtual wires” or nets (involving connections between two placed modules) might be solved (through a total of eight switch closures).

We can extend the concept through an approach analogous to the segmenting previously described. A number of substrates could be tiled together, connected through some of the available external termini as suggested in Figure 49. In this case, two adaptive wiring substrates or “cells” form an extended system. Now, a netlist solution is compound in nature, involving a global specification (such as “connect AI to AO”) and local specifications (the specific solutions of each “cloud”). The local specification involves allocating terminals between cells, and then defining subnet problems for each cell. It is then necessary to compute local solutions within each cell to implement the implied sub-netlist. It is obvious upon inspection that there are many non-unique solutions for a particular global netlist problem, both in terms of the allocation of nets between cells and implementations of the sub-netlists within each cell.

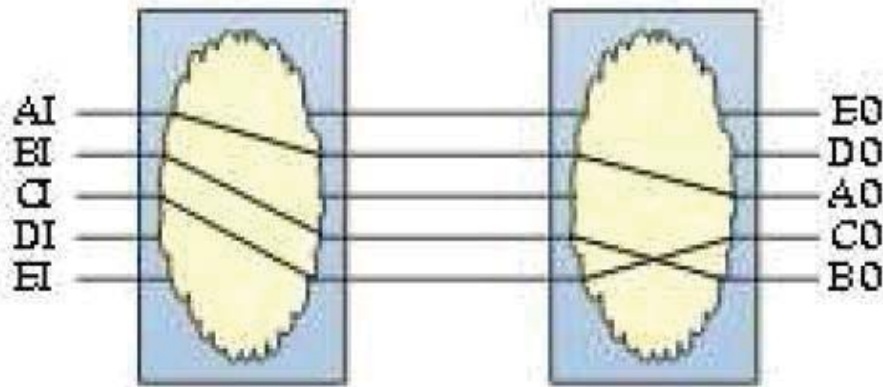


Figure 49. Extended adaptive wiring system by using two cells

Since the adaptive wiring substrates (or panels) may be pre-built and inventoried until use, it is possible to retrieve them as needed and configure them on demand. Rather than wait for custom-defined wiring harnesses to be developed and delivered, a process that could take weeks or months, the adaptive versions can be configured very quickly. Unlike custom wiring harnesses, whose wiring pattern is permanently locked in, adaptive panels can be altered as needed to accommodate late-point changes.

Furthermore, this architecture has the ability to adapt to faults that occur after a system is placed in the field. Since wiring patterns can be software-definable, defects could conceivably be fixed by computing an alternate configuration.

### 3.1.3 Example of Adaptive Panel Design and Implementation

In this section, we describe the implementation of a demonstration adaptive wiring panel (AWP) system shown in Figures 50 and 51. The demonstration system in Figure 51(right) contains six (of 64 planned) cell units, a few simple modules (for plugging into the adaptive panel), and a laptop as the controlling cell management unit (CMU). As a programmable fabric, the AWP

requires tools to generate specifications for implementing particular wiring solutions to interconnect “modules” (shown lower left). A simple graphical user interface (GUI), shown upper left, was created for this purpose.

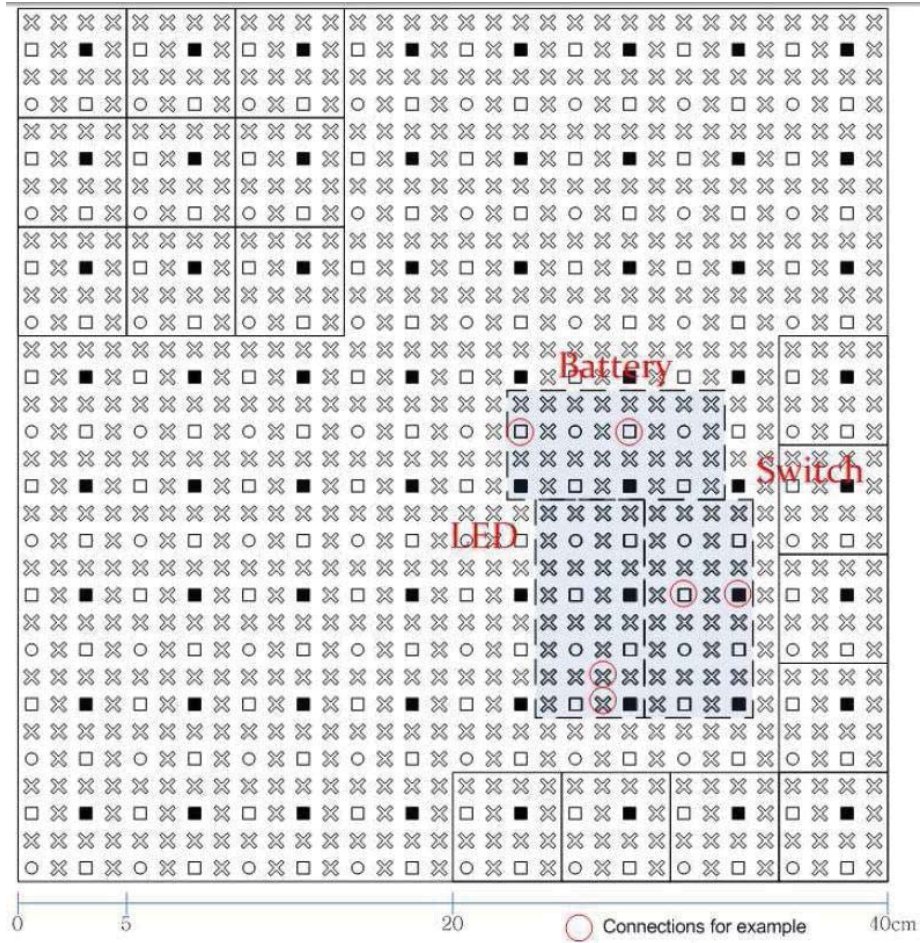


Figure 50. Adaptive Wiring Panel (AWP) concept using 64 cells in a 8x8 array

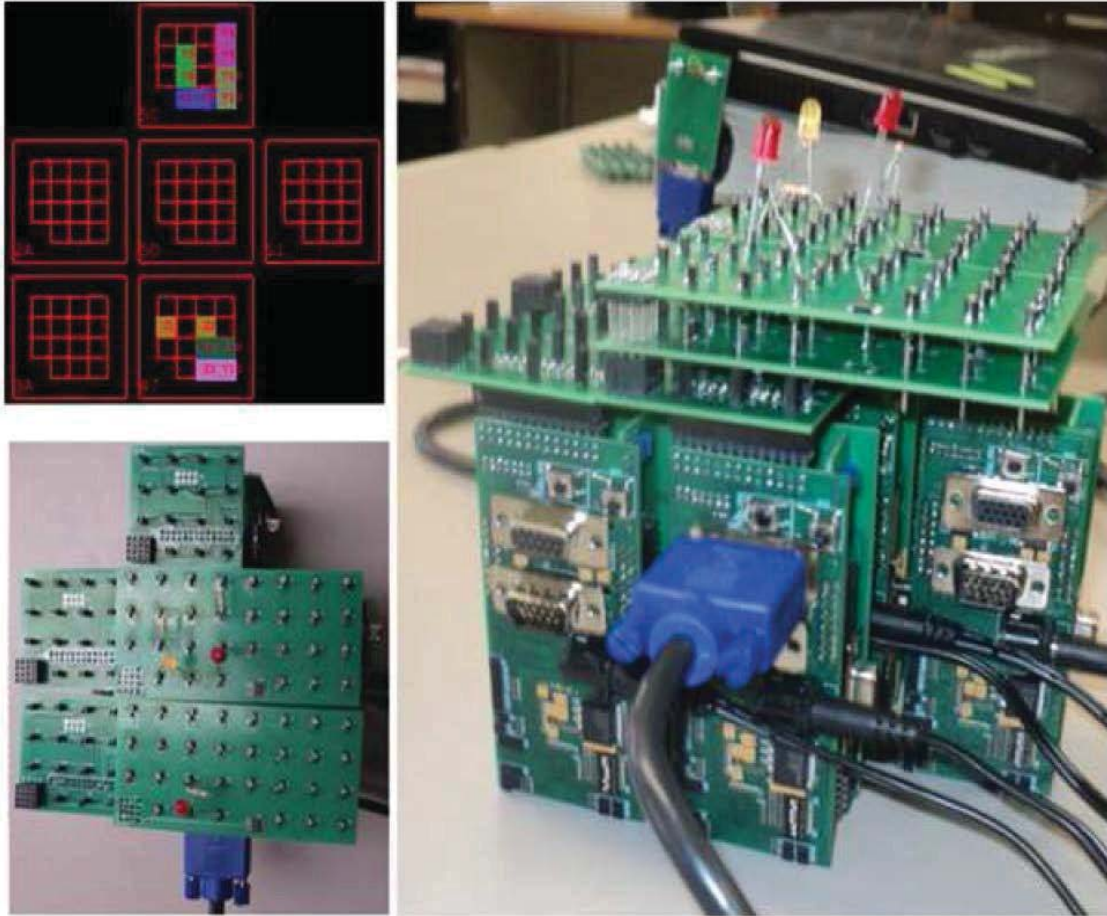


Figure 51. Partial adaptive wiring panel (AWP) system, containing six cells and two modules. (top left) Graphical user interface (GUI) snapshot used to set-up the netlist. (bottom left) Top view of the AWP. (right) Side view of demonstration system revealing the

The main objective was to design, develop and build a 40×40cm adaptive wiring panel (AWP) and the modules to be connected in there. For a first stage, both signal and power connections were programmable. The mechanical connections were fixed. The panel shown in Figure 51 is square-shape with three types of connectors on it: (i) signal connectors (for transmission and reception of the signals, labeled as ‘X’ connectors), (ii) power connectors (for connecting to the source powers, labeled as o connectors), and (iii) “white box” is for mechanical connectors (for holding the modules). The power connectors with black color ‘■’ represents fixed connections to ground (GND).

Figure 51 shows an example for connecting three modules on the adaptive wiring panel: (i) a module with a LED, (ii) a module with a switch (to control the LED from (i)) and (iii) a battery module. The modules are connected on the surface of the panel. They can be placed at any location within the panel. Each module needs to be connected in the locations shown with a red circle. We will next describe the cell unit, the modules, and the CMU that manages the configuration of the overall system.

### 3.1.4 The AWP Cell Unit

Each cell unit (CU) is an “atomic” element, a minimum independent unit required to create the



AWP. Its logical architecture is shown in Figure 52. A rectilinear tile that connects to its nearest neighbors in all four directions is referred to as a “NEWS” (north-east-west-south) network. Each edge (detailed only for the “east” port) contains a local communications port (for inter-tile information sharing), as well as pins for routing wiring connections between other edges and the primary surface array of contact pins. The surface array is the primary set of termini that are user-accessible. These are intended to support connections to matching pins present on “modules”, which are to be surface mounted onto an AWP. “Modules”, as will be discussed, are intelligent assemblies, and as such, require communications. The cell-module I<sup>2</sup>C port provides support for this purpose. Finally, a single “cell common I<sup>2</sup>C port” is provided to support communications to the cell management unit. Unlike the other I<sup>2</sup>C ports, which are implemented as point-to-point interfaces, the common I<sup>2</sup>C port is connected to all cells in an AWP.

The cell functions are managed by a “cell local processing unit” (fully implemented in hardware using FPGAs), including the six communications ports, cell status functions (such as maintaining a globally unique identification code), and configuring the switches connecting the wiring resources in the cell. The functions of each CU are:

1. Control the programmable connections of the AWP.
2. Communicate with (up to) four neighbors: each CU needs to communicate with its physical neighbors to recognize spatial orientations.
3. Read “electronic datasheet” information from modules (each module has a probe pin which sends module specifications to the cell it is plugged into).
4. Provide a low current power supply to the modules to enable transmission of electronic data sheet information through predefined probe pins.
5. Communication with the cell management unit: each cell unit transmits and receives information to/from the cell management unit (i.e., cell units aside from neighbor recognition cannot communicate directly with each other). Each cell block, upon system power up, will send identification information such as ID of the CU, the IDs of its neighbors and relative orientations, and module Electronic Data Sheets, if connected to that CU.

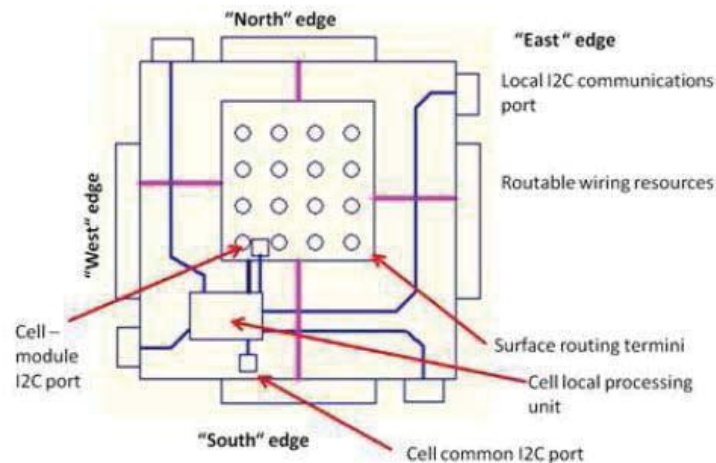


Figure 52. Cell unit logical architecture

Figure 53 illustrates the physical embodiment of an AWP cell unit in our prototype. Each AWP cell consists of 5 boards:

1. Top board is where the modules are placed.
2. South board is the main board where the main hardware is placed: a FPGA with all the logic control, the relays to close the connections, and extra hardware (for example, to reconfigure the FPGA for updates of the system). This board controls the rest of the boards. It includes a connector that connects to the North board of a neighboring AWP cell.
3. East board is connected to the West board of a neighboring AWP cell.
4. North board is connected to the South side of a neighboring AWP cell.
5. West board is connected to the East side of a neighboring AWP cell.

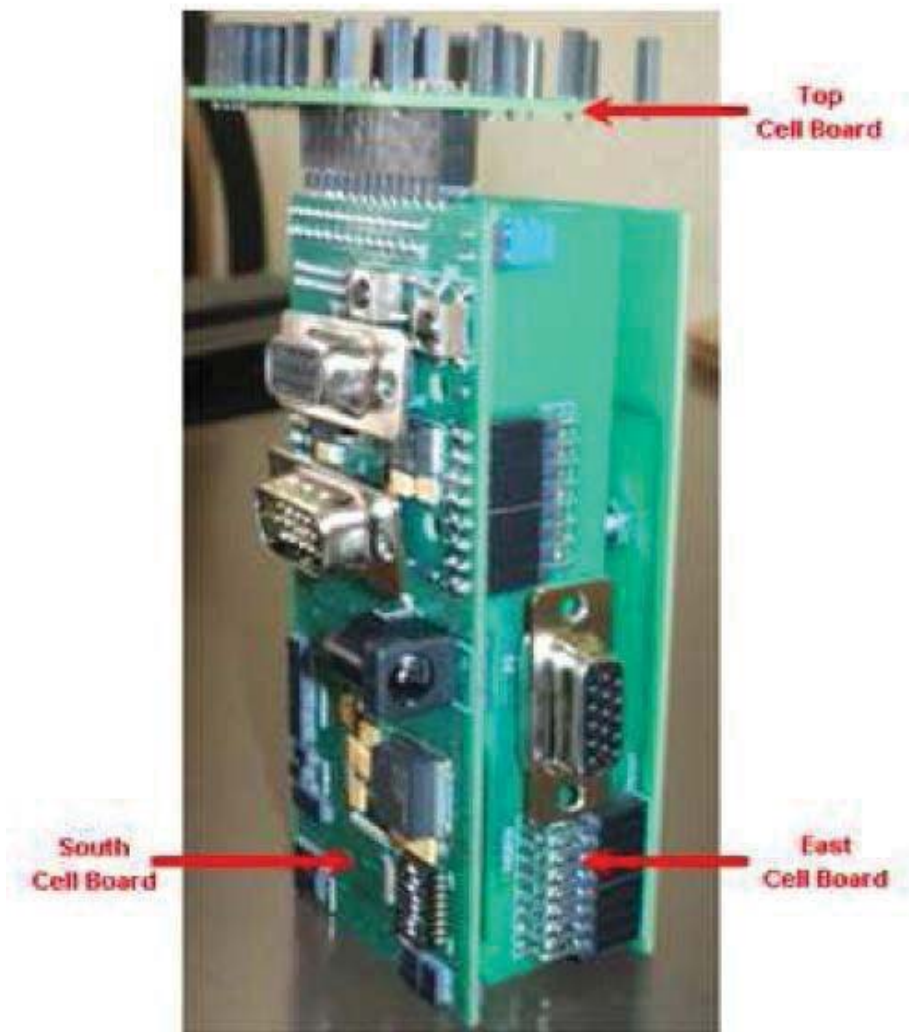


Figure 53. Cell unit of the AWP

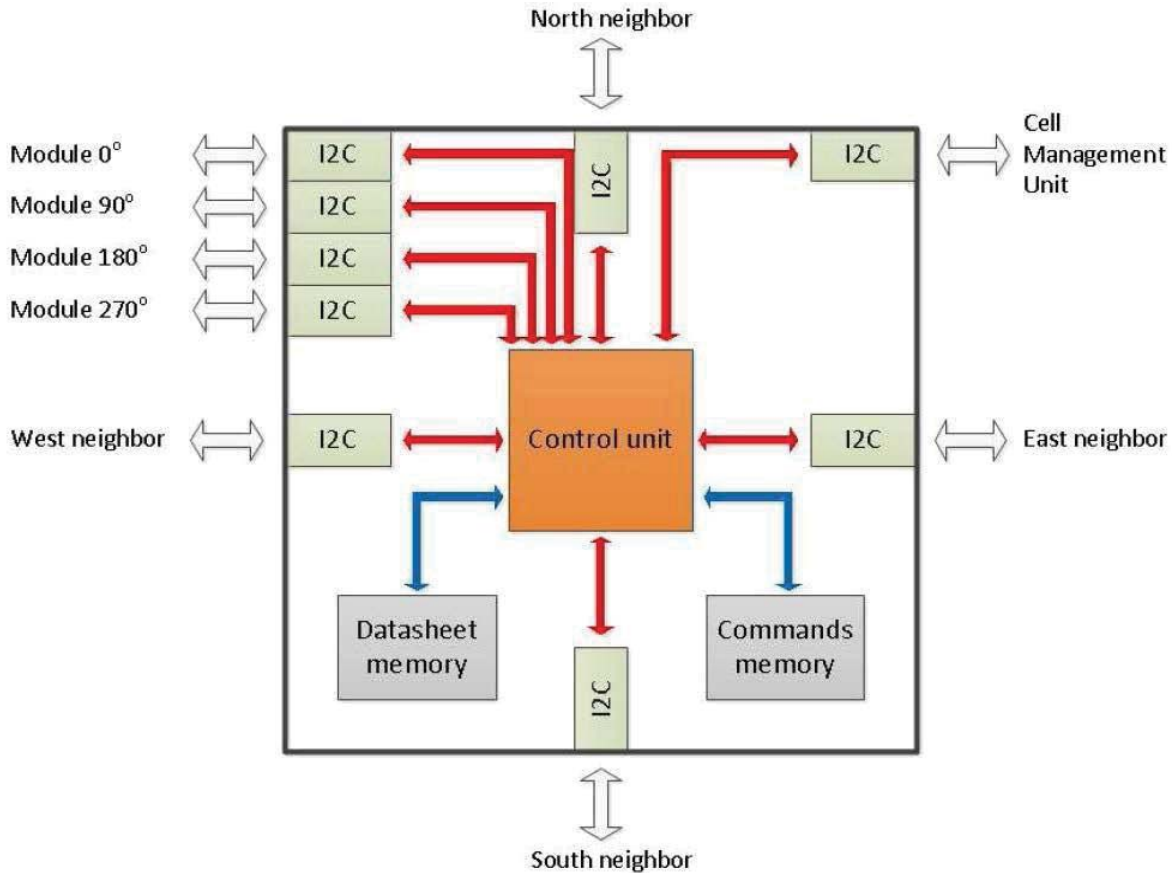


Figure 54. Block diagram of the hardware design in each FPGA cell unit. It consists of a main control unit, memory blocks, and I2C blocks

The CU has been fully implemented in a FPGA using VHDL. The CU hardware design is depicted in Figure 54. It consists of a main control unit, internal memory units, and I<sup>2</sup>C blocks to communicate with every other component of the AWP.

### **Cell Management Unit**

The cell management unit (CMU) manages global communications and routing configurations of all cell units on the AWP. An architectural diagram depicting the connection to a typical cell unit indicates the chain between the cell management unit (global), the cell unit (local), and particular relays (switches) controlled by particular cells.

### **3.1.5 Module**

Modules refer to components that are plugged into AWP assemblies. Before we describe them, it is insightful to consider how an AWP might be used in a simple design example shown in Figure 55. A prospective AWP is shown with three modules in Figure 55 (a) (a light bulb, a switch, and a battery) placed on the panel. The placed modules cover a number of pin locations and mechanical attachment points (revealed in Fig. 55 (b)). These modules can be placed in any Manhattan direction and in any linear position so long as the mechanical attachment grids of the module align to those on the AWP. At this point, the AWP does not “know” what to do with these modules. Rather, the user placing the modules must supply this

information in the form of a netlist. When this is done, the AWP can connect the modules by forming virtual wires, as shown in Figure 55 (c). If a second copy of a module (e.g., an extra light bulb) is placed on the AWP, it intrinsically has the ability to connect to this second copy when the failure is detected (Fig. 55d)).

We now move from this abstract description of an AWP with modules to describe our demonstration implementation. The modules we built were 5×10cm (or 2 cell units) with 24 signal connectors, 6 power connectors and 2 mechanical connectors (most of which need not be connected for a specific module type).

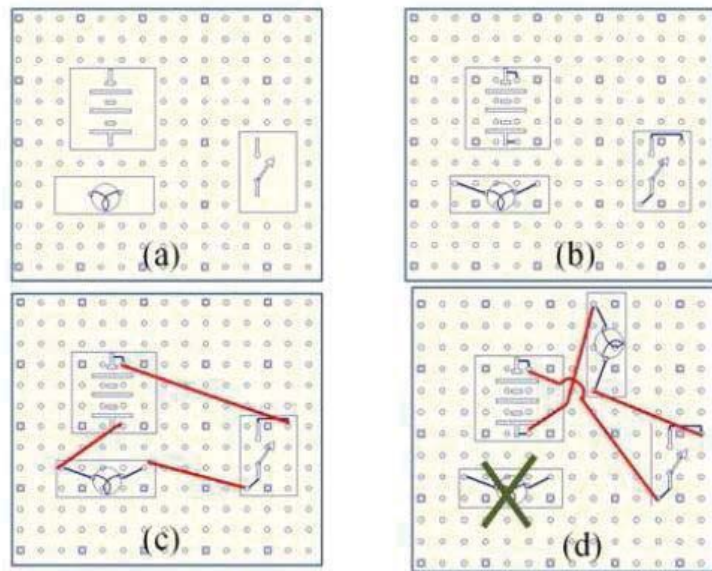


Figure 55. AWP in use. (a) Modules are placed. (b) Connection details revealed. (c) Virtual wire formation. (d) Connection to redundant module

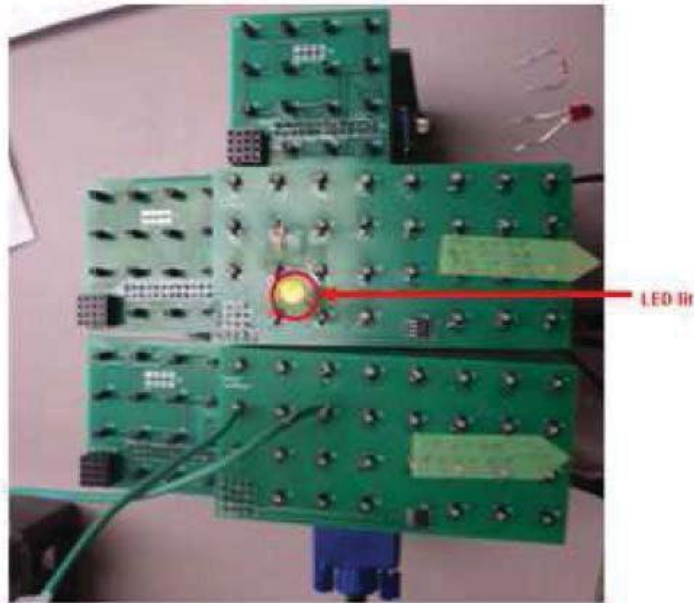


Figure 56. AWP with the circuit connected using two modules. The bottom module has a battery and the top module with a resistor and a LED

### 3.1.6 Prototype Demonstration

Our prototype, a partial panel containing six cells and two modules, has demonstrated all of the elements of the AWP described in this section. Module 1 has been prototyped as a “compound” consisting of a battery source (V1), a resistor (R3) and a LED (L3). Module 2, also a “compound”, has two resistors (R1 and R2) and two LEDs (L1 and L2). Simple circuits of the form shown in Figure 4.13 can be composed, in which subsets of modules can be connected. Even these simple demonstrations have considerable underlying complexity, as each cell contains a dedicated processor, internal wiring and 70 relays to implement local connections. The modules also have internal microprocessors to “explain” modules to the adaptive system.

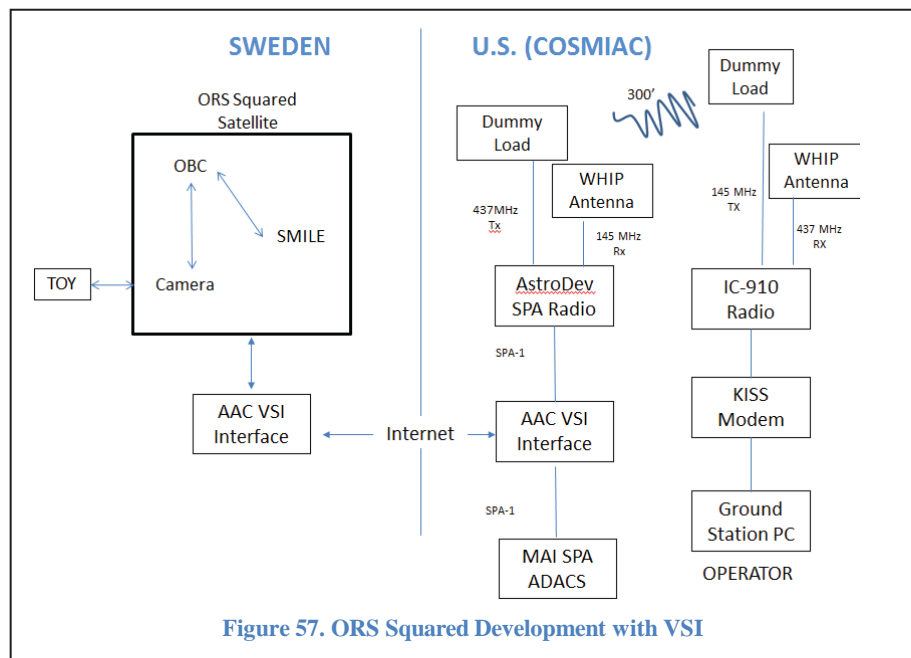
We have successfully created a GUI that can manage the creation of circuits on the AWP. Once a circuit is set-up, the user can re-arrange either the cell units or the modules. The AWP will look for the new locations of the components previously defined and adapt the routing to connect the circuit.

## 3.2 Space Plug-and-play Architecture (SPA)

This activity has been expedited by the use of the Virtual Satellite Integrator (VSI) system that allows COSMIAC personnel to be able to integrate and test systems we are developing including the encryption and radio systems into the main onboard computer located in Sweden (as shown in Figure 57). The VSI is a commercial product developed by the AAC Microtec Corporation. The true power of this is taking advantage of the time difference. The Swedish team can work on the satellite all day. Based on the time difference overseas, the Swedish developers leave for the day as COSMIAC engineers arrive at work. As long as they leave the satellite running, we can turn on our satellite ground station on one side of the building. We can transmit across the building (emulating a space link). It goes into the satellite radio (SPA compatible) and then is

connected to the onboard computer in Sweden (as shown in Figure 57). COSMIAC can then command the satellite's camera to take a picture and download the image to the COSMIAC ground station for processing. This allows for global design and integration without the need for delivering any information to foreign nationals.

All of these efforts are precursors to the future Nanosatellite And Plug-and-play Architecture (NAPA), a five year project between AFRL and the Swedish Ministry of Defense.



COSMIAC provided support for a highly successful sounding rocket project in 2011 that was designed to test the SPA capability of the Vulcan Wireless Corporation new radio as well as to test a student led activity related to SPA-1 connector testing. The desire is to test the ruggedness of a variety of small and inexpensive series of connectors for use in space related activities. The sounding rocket was launched from southern New Mexico and was used to test the radio and connector project during flight. Analysis of the flight was completed and presented at the CubeSat workshop in 2012.

### 3.3 Commercial High Speed Digital Test Structures

The purpose of this project was to design and fabricate basic high speed digital test structures using different technology nodes to be used for further analysis. The first test chip was designed and fabricated using 0.35um (Taiwan Semiconductor Manufacturing Company) TSMC process. The test chip was also exposed to static and low frequency tests. The frequency Range is DC to

20GHz. There are possible technology node options including: 0.35um, 0.25um, 0.18um, 0.13um, 90nm, and 65nm (all from TSMC).

### **3.4 New Materials Developed for Advanced Reconfigurability**

We have also worked on developing new materials for advanced reconfigurability for space systems. Some of this work involved coordination and cooperation with RVSE at AFRL. In collaboration with Dr. Rod Devine, we conducted a series of experiments on advanced semiconductor devices with feature sizes as small as 32 nm. The objective of the experiments was to determine the characteristics of the bias temperature instability, a device reliability concern. Also, thin film organic based electronic devices were characterized in collaboration with Air Force Office of Scientific Research (AFOSR) and Dr. Rod Devine.

Some work was also done in processing several substrate material that can be used in solar devices for space applications This work was performed at the UNM Center for High Technology Materials. The resulting devices will be used in the future for a variety of experiments including the Haynes-Schockley experiment to determine carrier mobilities and diffusion constants, radiation experiments, and spectral analyses.

## **4. Results for Assessing the future development of reconfigurable electronics, devices, and architectures and their potential impact on overall system performance and reliability and planning for future space and defense needs**

An excellent way of assessing future development of reconfigurable electronics and their impact in space applications is to design and incorporate these electronics on cubesats, launch them and then monitor their overall performance. This section describes the various devices developed for that purpose and some of the new techniques such as additive manufacturing, neural networks embedded on FPGAs to rapidly design these reconfigurable electronics and control them.

### **4.1 Development of Configurable Electronics and Cubesats**

In this section we show what COSMIAC has developed in the last two years, under this agreement, in terms of cubesat development, related instrumentation, and the ground station developed to track and download satellite mission data for UNM, AFRL and the University Nanosat Program's (UNP's) FASTRAC satellites.

#### **4.1.1 Trailblazer**

This spacecraft (shown in Figure 58) was delivered in September, 2013 to the National Aeronautics and Space Administration (NASA) with a launch date of November, 2013 aboard a Minotaur rocket from Wallops, Virginia. It is being launched under NASA's Educational Launch of NanoSatellites (ELaNa) program. All parts were received and integrated. Environmental testing occurred during the week of February 11, 2013 at AFRL on Kirtland AFB. Two other cubesats were tested at the same time; DragonSat from Drexel University, and PrintSat from Colorado Sciences Corporation. These three 1U CubeSats will fit into a single deployer for vibration testing. An image from the vibrational testing is shown in Figure 59. No problems or failures were found in the Thermal or Vibrational testing that was accomplished per the configuration shown in Figure 60.





Figure 58. The Trailblazer Satellite

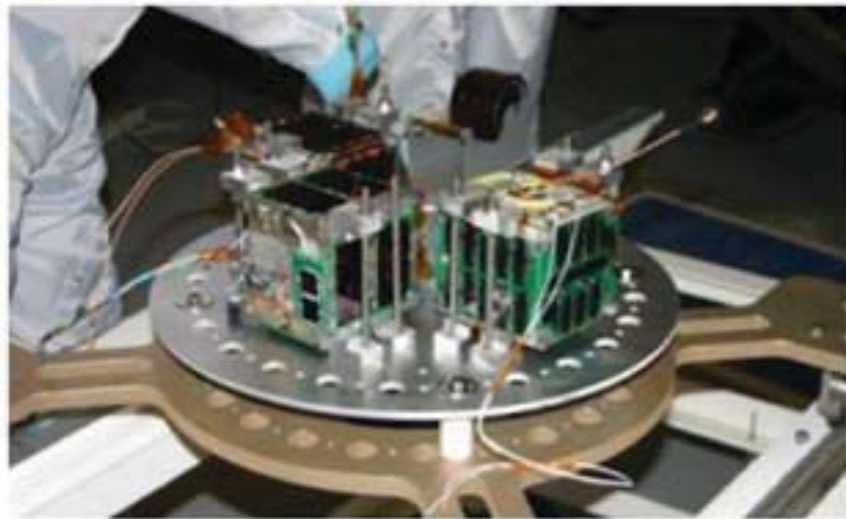


Figure 59. Environmental Testing of Trailblazer

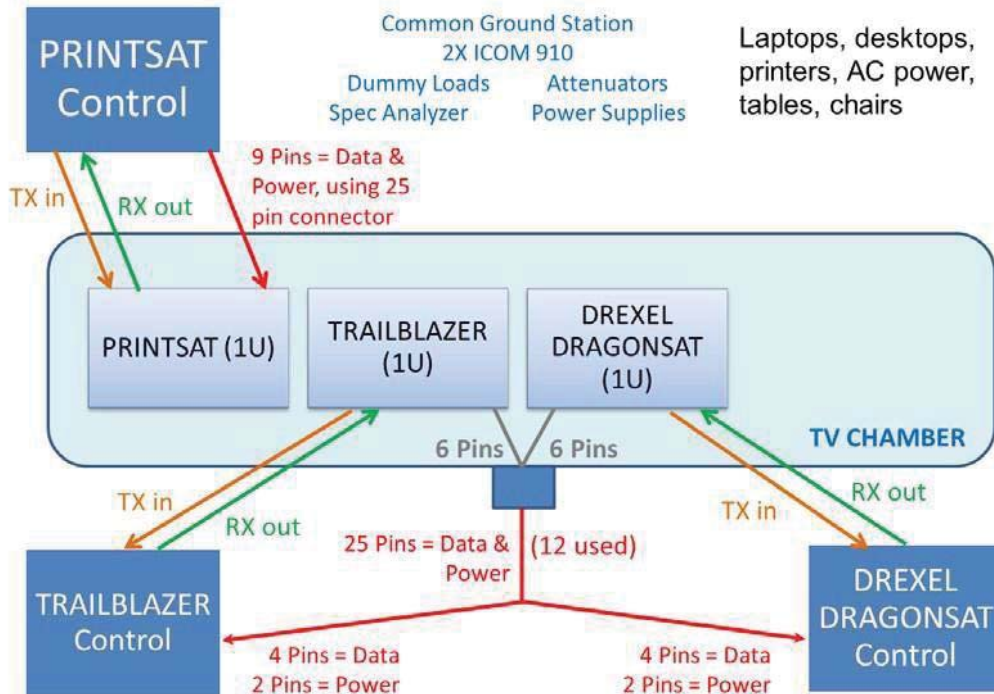


Figure 60. Thermal or Vibrational Testing Configuration

The individual components on Trailblazer are:

- Radio: This is an AstroDev Radio that transmits down in the UHF band and receives in the VHF band. We successfully transmitted and received from the satellite radio through the COSMIAC ground station. Full conversion to SPA-1 was completed as part of the integration.
- Command and Data Handler (C&DH): Mr. Jesse Mee designed the Pluggable Processor Module (PPM). A compiler was purchased to develop the PPM's required programming files.
- ADACs: The 1515 magnetic stabilization unit is incorporated into the structure.
- Structure: It was noted that we needed harness clips for securing the solar panels to the structure that were not part of the original purchase. We purchased the clips and then incorporated a wiring harness from the Pumpkin Corporation.
- Dosimeter: The hardware is complete. The Appliqué Specific Integration Module (ASIM) code has been written. We flew this module on a high altitude balloon as a functional test. Dosimeter ASIM and flight module were run through a 40-day test to check functionality and for calibration.
- University of Texas El Paso (UTEP) 3D module: UTEP has delivered a "fit check" module. The final flight unit was delivered and incorporated.
- Power: We stored, routinely checked and charged the batteries. Electrical Power System SPA-1 conversion was completed. One issue that arose during the building of the Trailblazer spacecraft was the inability to obtain a U.S. built Nanosat power system. COSMIAC developed its own power system for future space activities

- Environmental Testing: A mounting plate has been developed to act as the interface between the satellite launcher and the AFRL test fixture.
- Ground Station Software: Although we are confident about the ability of the Global Educational Network for Satellite Operations (GENSO) system to support our satellite downlink capabilities (and to provide more downlink options), we will need a more robust system to do command and control. Mr. Jim White (Colorado Satellite Services) has provided us with the control system he used for his RAMPART satellite. It was written in VB6 software. We are transitioned it into a .NET format.

#### 4.1.2 ORS Squared

This is a joint activity with NASA, UNM, AFRL, AAC Microtec, and a variety of other developers to build a SPA 6U satellite. UNM is serving as the Program Manager for this effort. All modules have been created and delivered. The final structural design was completed, milled in aluminum and anodized. The satellite is scheduled to be delivered in 2014 for launch aboard the ORS-4 mission to a 97 degree orbit. A very good benefit from this effort has been the use of the AAC Microtec Virtual Satellite Integrator (VSI). This VSI has provided the capability to test interchangeable modules from across the globe. This type of testing has piqued the interest of the NASA Glenn and Ames Research Centers. NASA has requested COSMIAC to provide more information and possibly a training course on the VSI, 3D printing and SPA. This type of capability will have a great benefit to the future satellite missions between AFRL and NASA.



Figure 61. ORS2 Satellite

A subset of some of the missions objectives are:

- Test flight of Space Plug-and-play Architecture (SPA)
- Flight of an AFRL space weather boom and related magnetometer
- Flight of the MMA Corporations deployable solar power system

- Flight of a dosimeter space weather experiment developed jointly by COSMIAC and AFRL

## 4.2 Additive Manufacturing

COSMIAC has developed a working relationship with the University of Texas El Paso (UTEP) related to their Additive Manufacturing (AM) process. We believe that this type of technology will help the future development of reconfigurable circuits and devices for all future satellite missions. This approach has the capability of being very cost effective and time saving. Where this has come to pay a great benefit is in the ORS Squared development. The team was able to print a plastic version of the structure (as shown in Figure 62) that had extremely high quality tolerances. This allowed the team to hold the satellite structure in their hands during the prototyping phase. This capability allowed the team to make critical design decisions before the actual structure was milled in aluminum (and at a fraction of the cost of aluminum).

COSMIAC has been working with UTEP to develop a design for future 6U spacecraft. This joint project with the Maryland Aerospace Corporation is to 3D print a spacecraft bus that will allow developers to focus on their payloads. An initial design is shown in Figure 61. Another example of how this technology could be used is in the ORS Squared mission. The Engineering Model structure will be 3D printed in plastic while the main structure is being milled in aluminum.

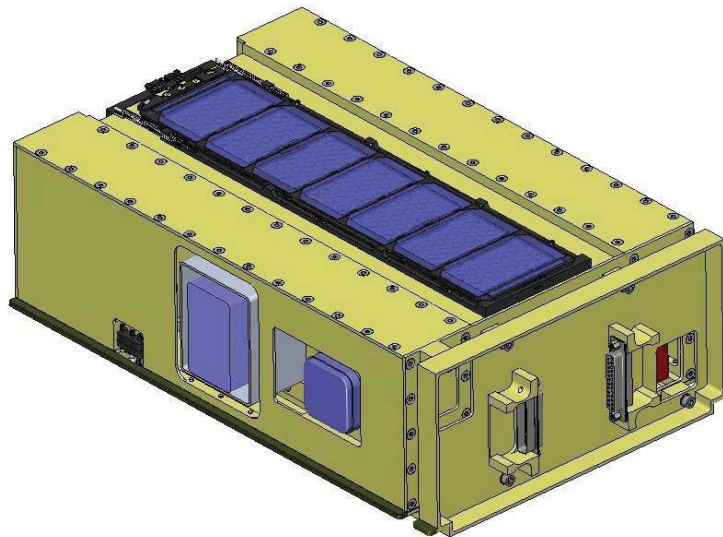


Figure 62. ORS Squared for 3D Printing

This will allow the design team to be able to do form and fit checks prior to the actual structure being complete. Modifications can be quickly made in plastic that can allow for optimization and design in a parallel fashion.

Various opportunities are currently under investigation for advancing this area of space research. The current effort is to print a 6U CubeSat for demonstration purposes. In addition, future efforts

revolve around how to create much more elaborate items that can be tested for the rigors of space. COSMIAC is working with the Planetary Systems Corporation to ensure that the developed satellite will be compatible with their 6U deployer. Figure 63 shows a first iteration of the printed technology. The system shown is a 6U CubeSat structure. The tolerances that were achieved with this process were stellar. Tolerances were found to be within  $\frac{1}{2}$  of  $1/1000^{\text{th}}$  of an inch.



Figure 63. 6U Prototype 3D Printed Structure

### 4.3 Software Defined Radio (SDR)

The current research revolves around the ability to process AX25 packets at 9,600 bps. One of the projects during the summer of 2011 was related to developing a SDR platform for GENSO. To achieve this, the team gained expertise in the area of creating spread spectrum radios utilizing commercial SDR platforms. SDR development, utilizing the USRP2 platform, has progressed through the use of two primary tools. The first tool, GNU Radio Companion (GRC), allows the user to drag and drop Digital Signal Processing (DSP) blocks to a workspace and connect different blocks into a graph. This graph is then “compiled” and translated into Python. The other development tool is Simulink. Simulink blocks are translated into C++. Simulink and GRC are very similar in that both allow the user to drag-and-drop DSP blocks to a workspace. The collection of DSP blocks on the workspace is then used to control, and process the data of, a USRP2 SDR. Utilizing these tools, flow diagrams have been generated that resulted in the following modulation schemes, Direct Sequence Spread Spectrum (DSSS), Frequency hopping (FH), and a combination of both, Direct Sequence Spread Spectrum with Frequency hopping (DSSS-FH). The DSSS scheme was implemented in both Simulink and GRC. In order to spread the data signal, the data signal was mixed with a pseudo-random noise sequence (PN code) before being modulated. Synchronization was achieved through the use of a custom block containing a signal correlator, an artificial delay, and persistent storage for the most common signal offset. The algorithm allowed a frequency hop every 5 seconds and every 20 seconds the frequency range is changed with limits of 51 MHz to 987MHz. The pattern never repeats within a 24 hour period. This can be altered to fit any frequency range and hop rate by changing the user defined function. These models have been simulated in Matlab/Simulink. The models have resulted in a Bit Error Rates (BER) of  $\sim 0.01\%$  and very good signal to noise ratios.

#### 4.4 Langmuir Probe Design

New Mexico Institute of Mining and Technology (NMT) developed a Langmuir Probe that AFRL wishes to fly in a future space mission. As the subject matter experts in SPA and its implementation in configurable electronics, UNM was essential in providing guidance on the development of the control system used for this probe. UNM also used these same skills to provide environmental testing assistance to AFRL and NMT in performing testing of this probe. Environmental testing consisted of two activities. The first was vibration and vacuum testing in AFRL/RVEI. To accomplish this testing it was necessary to develop test plans according to the launch provider specifications. Then it was necessary to create a metal mount to attach the probe to the RVEI vibration table. The next phase was to complete plasma chamber testing at AFRL/RVB. RVB would only accept the probe once it was completely out gassed. Cables and test routines were created to interface to the RVB plasma chamber so that the probe could be tested in the plasma induced environment. UNM provided real time interfacing to the configurable control system during all testing. Cables and test routines were created to interface to the RVB plasma chamber so that the probe could be tested in the plasma induced environment. The device completed vibration testing as well as a complete bake-out in a thermal chamber. The probe successfully completed all testing and has been delivered to RVSE for shipment to Sweden. Figures 64 to 67 show the details of the designed Langmuir probe before being installed on the Quadsat satellite.

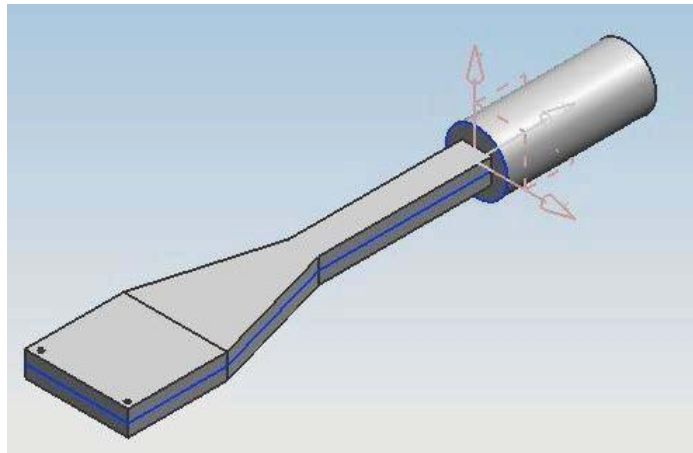


Figure 64. Langmuir Probe Design

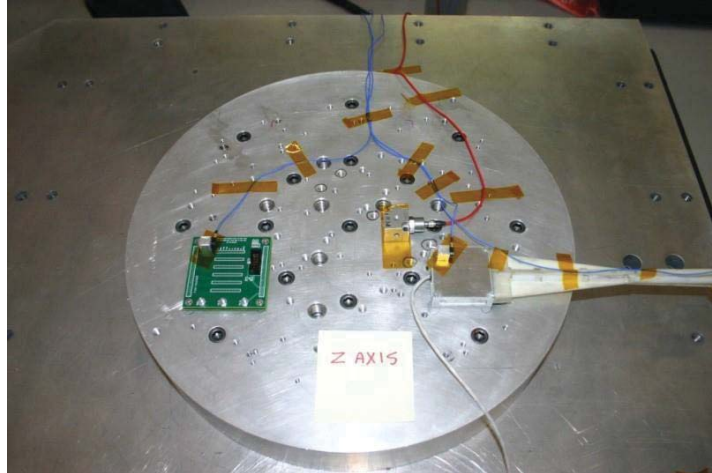


Figure 65. Data Design Nano CDH and Langmuir Probe on the AFRL vibration table



Figure 66. Z-axis vibration testing



Figure 67. Langmuir Probe in Plasma Chamber

## 4.5 Global Educational Network for Satellite Operations (GENSO)

A GENSO station was developed to track and download satellite mission data for UNM, AFRL and the University Nanosat Program's (UNP's) FASTRAC satellites. SDR solutions were implemented in the GENSO ground station system to replace the older HAM radios. SDR allows the system to take full advantage of the RF frequency spectrum. Many of the other GENSO stations we work with are at current or prior UNP schools so this work assists in providing a more robust download and upload capacity for the UNP program.

The COSMIAC GENSO station continues to provide a facility for active communications research. Past upgrades and improvements have created a very robust and reliable ground station capability. This allows the center to log into the ground station from anywhere and remotely perform experiments. The COSMIAC ground station is quickly becoming the model for others to emulate. We have been approached by several schools to obtain our lists of parts to help guide them in creation of their ground stations. The station continues to track satellites around the clock. In addition, the Center is beginning coordination with NASA Glenn on their CoNNeCT program. This program will place SDR assets in orbit on the International Space Station (ISS). The COSMIAC ground station has been selected as a potential site for performing research related to SDR on the NASA CoNNeCT program. All preliminary work performed to date has built the required expertise to make this possible.

## 4.6 The Use of FPGAs to Control Reconfigurable Antennas

Under this agreement research was conducted to demonstrate that FPGAs can be used to control reconfigurable antennas that can be used on cubesats or on a ground station by controlling various switches on the antenna to vary the frequency of operation and to also achieve RF spectrum sensing. Figure 5.11 shows an example of such an FPGA reconfigurable antenna system. This system was the first of its kind. It was a result of research work conducted by Mr. Severn Shelley as part of his Master's Thesis and was published in 2010 [24].

### 4.6.1 Reconfigurable Antenna Structure, Design and Tuning

The antenna structure shown in Figure 68 consists of 3 layers. The bottom layer constitutes the square ground plane that covers the entire substrate. The middle substrate has a dielectric constant  $\epsilon_r=4.2$  and a height of 0.235 cm. The patch on the upper layer is composed of a main mid-section and four surrounding smaller sections. This antenna is fed through a 50  $\Omega$  coaxial cable. The feeding position and the antenna dimensions are shown in Table 3.



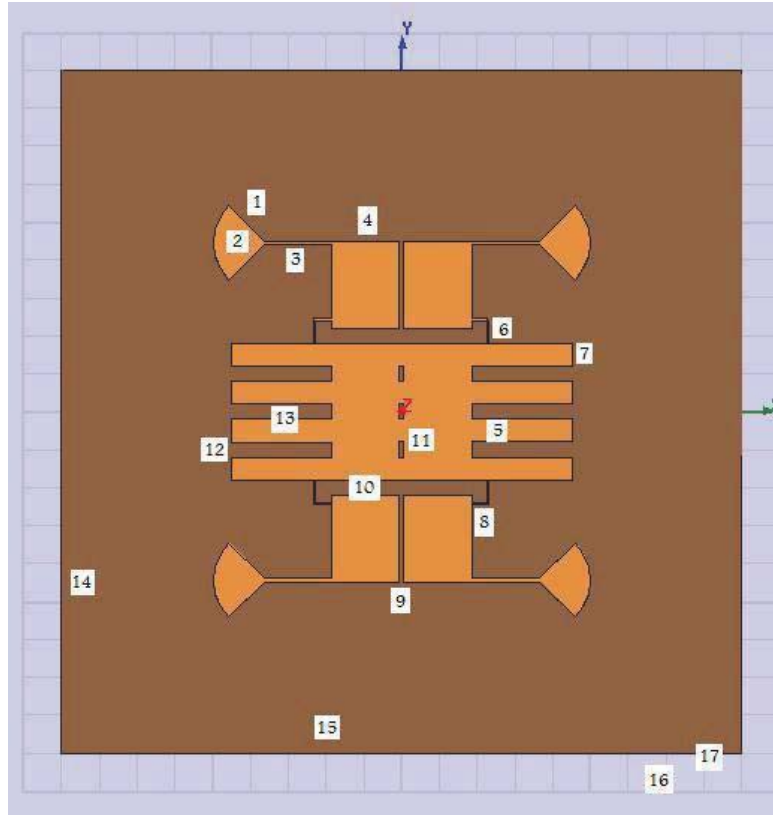


Figure 68. Schematic representation of the reconfigurable antenna

Microsemi's GC 4712 GaAs p-i-n diodes are used to connect the small sections to the main section as shown in Figure 68. These p-i-n diodes operate up to 18 GHz and are oriented in the X direction. 47 pF capacitors connect the p-i-n diodes to the main section of the antenna. These capacitors are oriented in the Y direction. The capacitors are used to prevent the DC current from crossing into the main section while passing the RF current. Quarter wave length transmission lines designed at 7.7 GHz are used to bias the p-i-n diodes. These  $\lambda/4$  lines are terminated by  $\lambda/4$  radial stubs in order to eliminate interference of the DC biasing network with the radiating structure. Biasing these p-i-n diodes separately can connect the corresponding side sections to the mid-section respectively.

Table 3. Antenna Dimensions

Part	Description	Length
1	Radial Stub	6.921mm
2	Radial Stub Angle	90
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	4.2ε0
17	Substrate Height	H: 2.35mm

To reduce fabrication costs, the biasing lines are etched on copper. These copper lines will resonate at a frequency where the length of the bias lines is approximately  $0.45\lambda_{\text{eff}}$  and at its odd multiples;  $\lambda_{\text{eff}}$  being the effective wavelength at the frequency of operation [25]. In this case these lines radiate around 9 GHz and its odd multiples which are outside the desired operating band (1-6GHz) of this antenna. The directions of the biasing lines are also optimized to contribute constructively to the radiation pattern of the antenna that is reconfigured.

The antenna achieves multi-frequency resonance tuning which as shown in Figure 69. The 4 configurations shown are a sample of the 16 possible frequency change configurations. The shift is noticed at frequencies lower than 3.5 GHz where many wireless communications applications can be found. In Figure 69, “0” represents the OFF state of a diode and 1 represents the ON state.

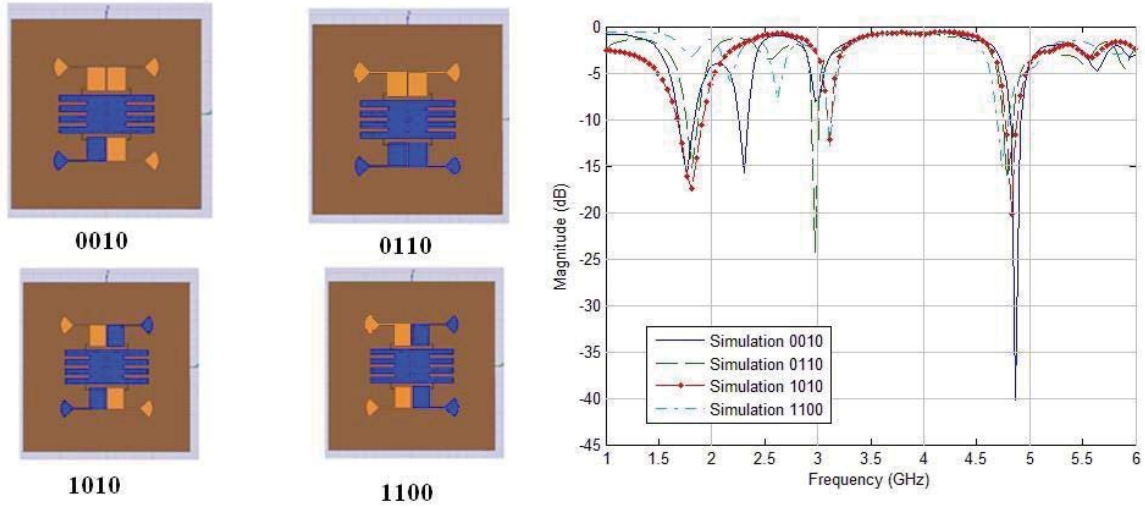


Figure 69. Antenna resonance for different diodes states 0- Diode OFF, 1 Diode ON

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 Figs. 70 and 71 respectively. This radiation pattern is plotted at a resonance frequency common to all the antenna configurations.

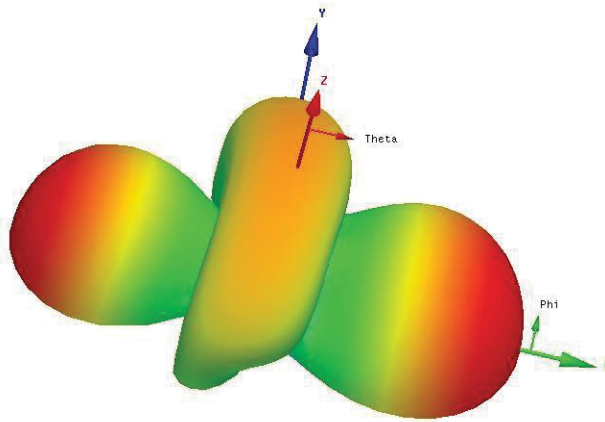


Figure 70. Simulated 3-D radiation pattern at 4.875 GHz

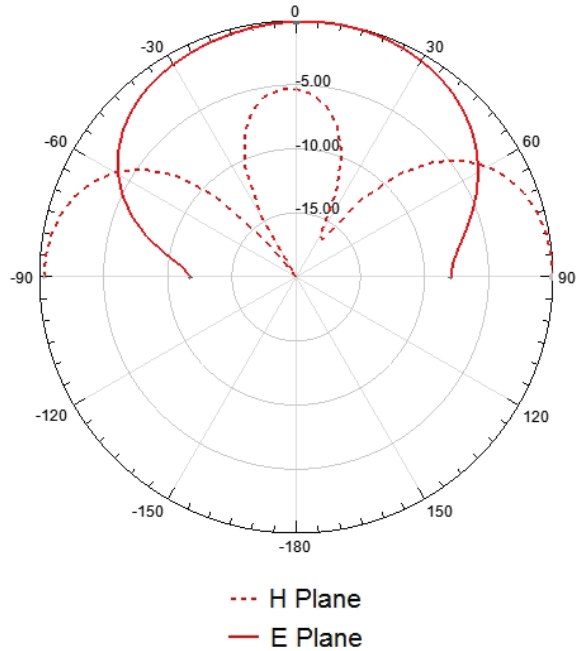
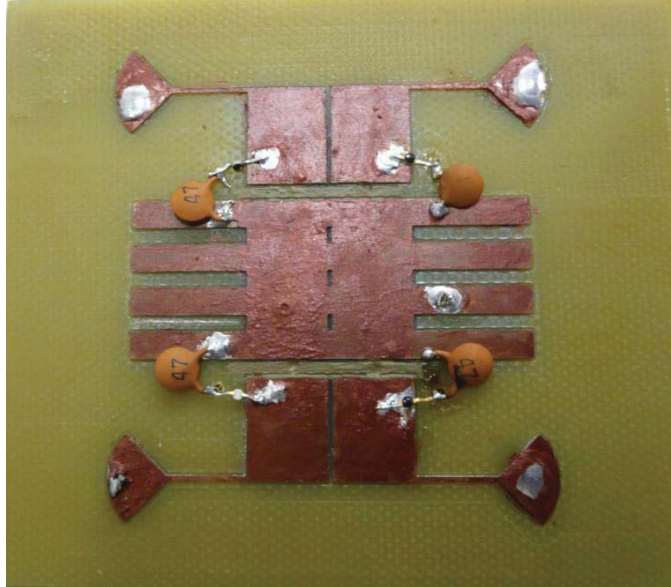


Figure 71. E and H plane cuts at 4.875 GHz

#### 4.6.2 Reconfigurable Antenna Fabrication, Measurement and FPGA Control

The fabricated prototype is shown in Figure 72. Shorting pins were inserted into 0.75 mm diameter holes drilled at the point of intersection between the p-i-n diodes and the 47 pF capacitors. These shorting pins are used to connect the Microsemi’s GC 4712 GaAs p-i-n diodes to ground. The VCC is connected to the through pins across holes drilled in the radial stubs. The FPGA’s output lines feed these pins to activate and de-activate the p-i-n diodes.



**Figure 72. The fabricated prototype**

For the FPGA to activate the diodes, activation, voltages are asserted on four output lines from the FPGA by way of the Joint Test Access Group 1149.1 standard [26]. 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 [27].

Four pins on the Spartan 3E board serve as the outputs for driving signals to the diode biasing network. The parallel III cable as well as the board is shown in Figure 74. The antenna reconfiguration is achieved through the following setup. 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 p-i-n diodes on the antenna. With one of the p-i-n diodes biased, the RF signal passes from the main patch through the capacitor and that diode to the outer patch.

Since there are four diodes, sixteen antenna configurations are possible. The measurement and reconfiguration setup is shown in Figure 74. Two comparisons between the simulated and fabricated S11 results are shown in Figures 75 and 75, respectively, where analogy can be noticed. A diagram showing the whole system is shown in Figure 77.

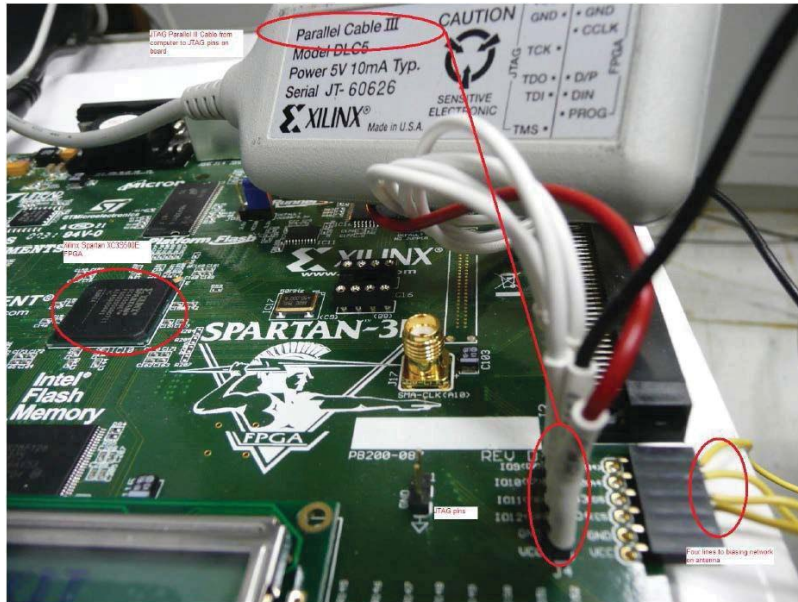


Figure 73. The parallel III cable with FPGA board

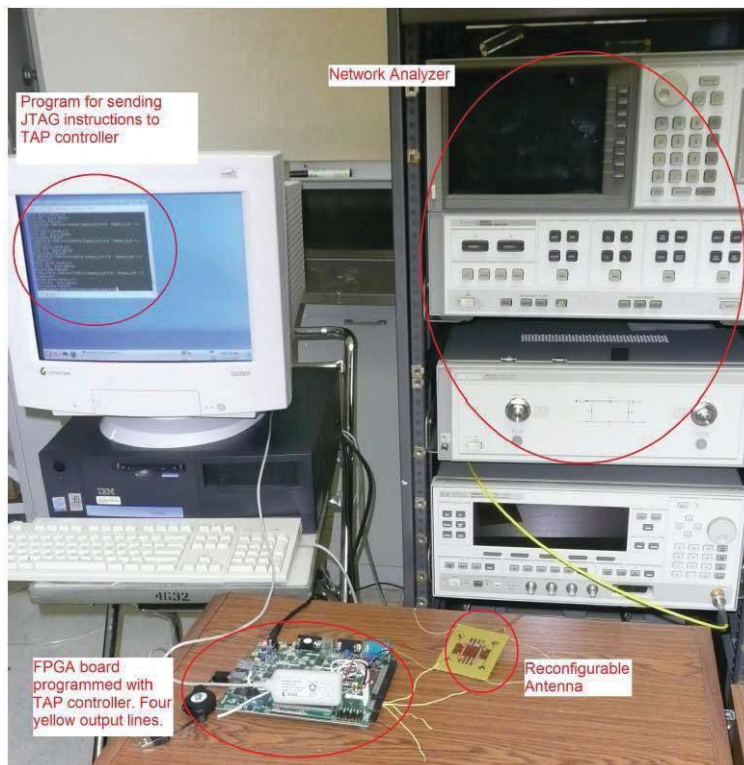


Figure 74. The S11 measurement setup

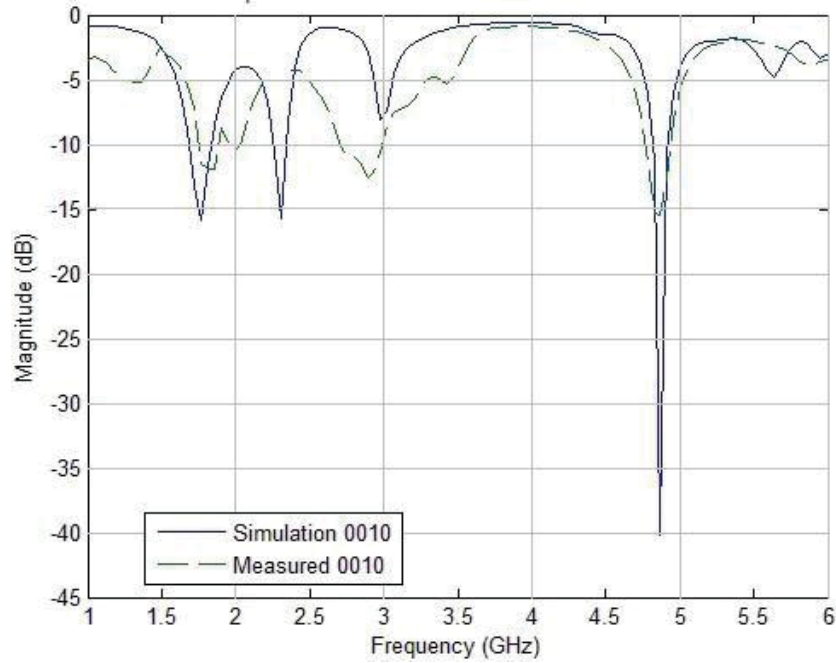


Figure 75. A comparison between actual measurements and simulation for an antenna state

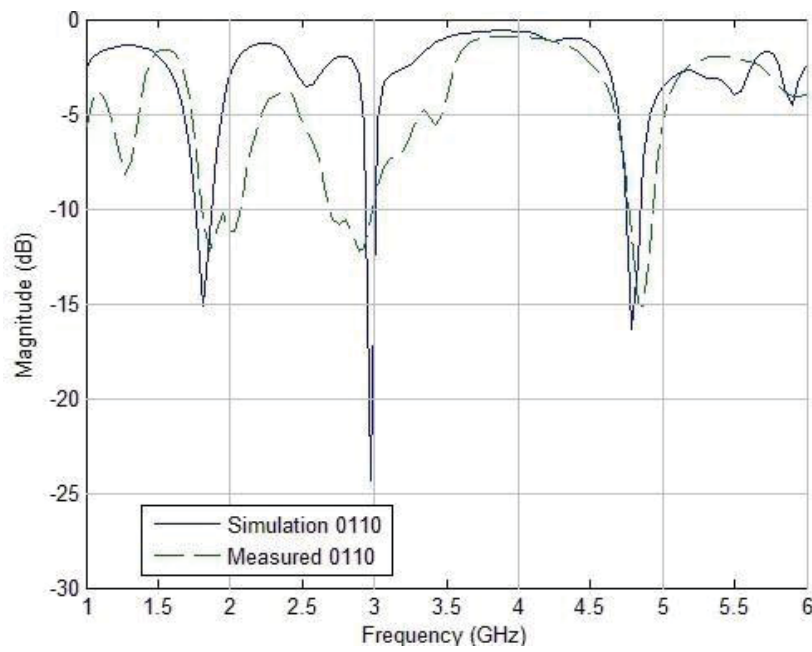


Figure 76. A comparison between actual measurements and simulations for an antenna state

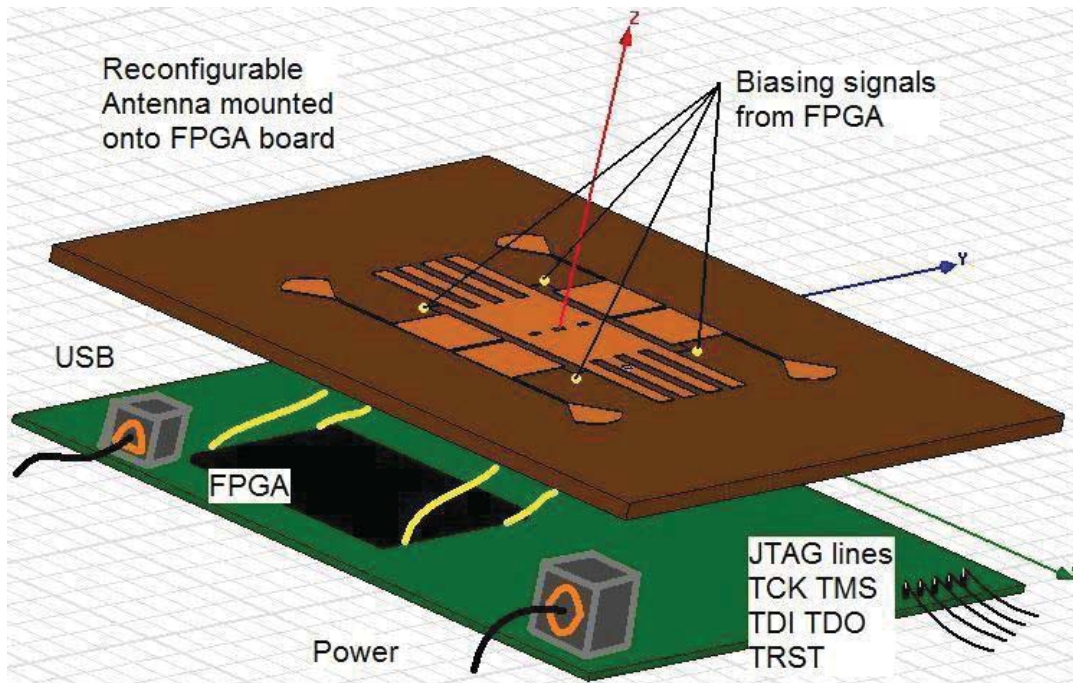


Figure 77. Entire reconfigurable antenna system

## 4.7 Using Neural Networks embedded in an FPGA to achieve device reconfiguration

This section gives a basic theoretical background of Neural Networks (NN), it shows how these Neural Networks can be embedded on FPGAs, and how reconfigurable systems can be controlled by an FPGA.

### 4.7.1 Basic Review of Artificial Neural Networks:

A Neural Network (NN) is a highly interconnected network of information-processing elements that mimics the connectivity and functioning of the human brain. Neural networks provide some insight into the way the human brain works. One of the most significant strengths of neural networks is their ability to learn from a limited set of examples.

#### 4.7.1.1 Neural Network Concepts

Neural networks are subdivided into two categories, artificial neural networks and biological neural networks. Artificial neural networks are used to simulate the structure and functioning of biological neural networks. The most familiar biological neural network is the human brain.

The basic element in a NN is the neuron. A biological neuron (as shown in Figure 78) is a nerve cell with all of its processes. Neurons have three main parts [28]. A central cell body, called the soma, and two different types of branched, treelike structures that extend from the soma, called dendrites, and axons. In a form of electrical impulses, information from other neurons enters the dendrites at connection points called synapses. The information flows from the dendrites to the



soma, where it is processed. The output signal is then sent down the axon to the synapses of other neurons.

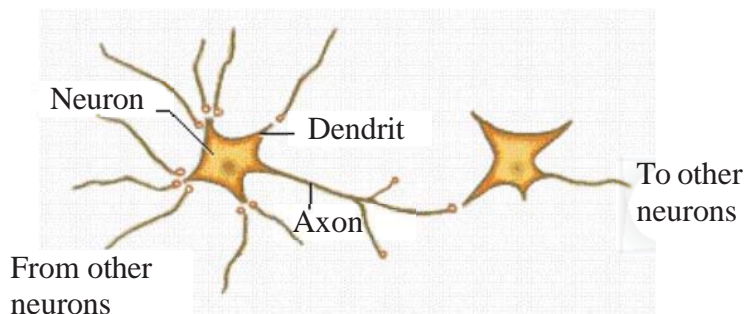


Figure 78. Biological Neural connections [29]

Artificial neurons (Figure 79) are designed to mimic the function of biological neurons.

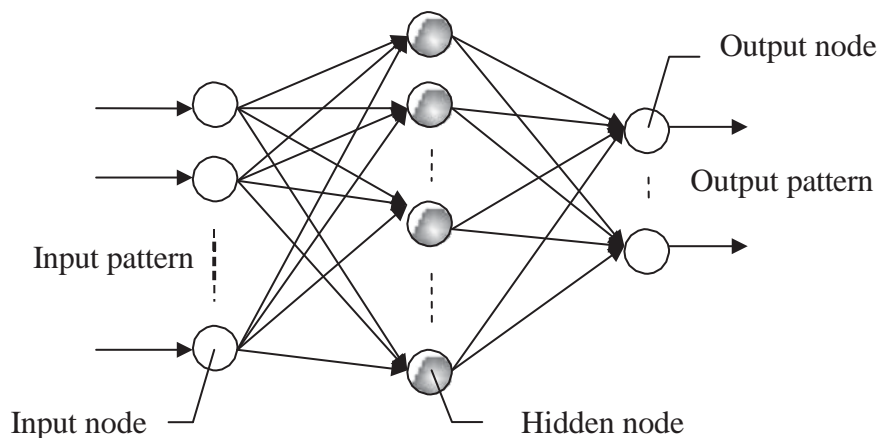


Figure 79. Artificial neural network topology

Artificial neurons have structures that are designed to mimic the function of biological neurons. The main body of an artificial neuron is called a node or unit. Artificial neurons may be physically connected to one another by wires that mimic the connections between biological neurons, if, for instance, the neurons are simple integrated circuits. However, neural networks are usually simulated on traditional computers, in which case the connections between processing nodes are not physical but instead they are virtual connections.

Usually, a Neural network architecture has three layers. The first layer is called the input layer and is the only layer exposed to external signals. The input layer transmits signals to the neurons in the next layer, which is called a hidden layer. The hidden layer extracts relevant features or

patterns from the received signals. Those features or patterns that are considered important are then directed to the final layer of the network, the output layer.

#### **4.7.1.2 *Neural Network Learning***

The ‘strength’ of the connections between relevant neurons determines the ‘strength’ of the memory. Important information that needs to be remembered may cause the brain to constantly reinforce the pathways between the neurons that form the memory, while relatively unimportant information will not receive the same degree of reinforcement.

##### **4.7.1.2.1 Connections Weights**

To simulate the way in which biological neurons reinforce certain axon-dendrite pathways, the connections between artificial neurons in a neural network are given adjustable connection weights. When signals are received and processed by a node, they are multiplied by a weight, and then added up.

##### **4.7.1.2.2 Back-propagation**

The most widely used scheme for adjusting the connection weights is called error back-propagation. The back-propagation learning scheme compares a neural network’s output to a target output and calculates an error adjustment for each of the nodes in the network. The neural network adjusts the connection weights according to the error values assigned to each node, starting with the connections between the last hidden layer and the output layer. After the network has made adjustments to this set of connections, it calculates error values for the previous layer and makes adjustments. The back-propagation algorithm continues in this way, adjusting all of the connection weights between the hidden layers until it reaches the input layer. At this point it is ready to calculate another output.

##### **4.7.1.2.3 Mathematical model of a neuron**

A Neuron is the information processing unit in a neural network. The mathematical part is extracted from Haykin’s ‘Neural Networks’ book [30]. Figure 80 shows the model of a neuron, which represents the basis for building neural networks. As shown in Figure 80, a neuron has three main components:

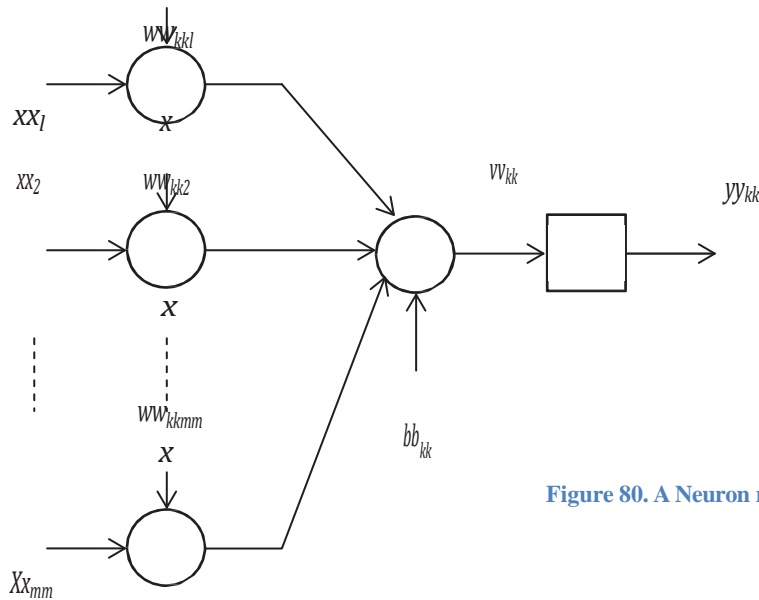


Figure 80. A Neuron model

1. Synaptic links, which are characterized by the synaptic weights ( $w_{kki}$ ). Each input signal ( $x_{ii}$ ) is multiplied by the synaptic weight ( $w_{kki}$ ).
2. Adder, for adding up the input signals, weighted by synaptic weights.
3. Activation function, for limiting the amplitude of neurons output to some finite limit.

A neuron  $k$  can be mathematically described using the following pair of equations,

$$u_{kk} = \sum_{i=1}^m w_{kki} x_{ii} \quad (5)$$

and

$$y_{kk} = f(u_{kk} + b_{kk}) \quad (6)$$

Where  $x_{jj}$ ,  $j=1,2,\dots,m$  are the input signals,  $w_{kji}$ ,  $j=1,2,\dots,m$  are the synaptic weights of neuron  $k$ , activation function.

Neurons in each layer compute the function signal that appears on their output and calculate an estimate of the gradient vector needed for the backward pass of the signal that determines the change in the weights of each neuron.

The function of an output neuron can be described using equations (4.3-4.5). The instantaneous error at the neuron's output can be written:

$$d_{pp}(PP) = d_{ii}(PP) - y_{ii}(PP) \quad (7)$$

where  $d_{pp}(PP)$  is the desired output for the neuron  $PP$  and  $y_{ii}(PP)$  is the output function signal of neuron  $PP$  at the  $n$ th iteration. The total instantaneous error of all neurons is then:

$$EE(PP) = \frac{1}{I} \sum_{ii=1}^I P_{ii}^2(PP) \quad (8)$$

where I is the number of neurons in the network's output layer. If N is the total number of examples contained in the training set, then the average squared error energy is:

$$E_{aaaa} = \frac{1}{NN} \sum_{rr=1}^{NN} EE(PP) \quad (9)$$

The weights of each neuron are updated on a pattern-by-pattern basis. The purpose of the neural network is to change the weights of its neurons in such a way that the total network's error will be minimized; the new weights are expressed as:

$$ww_{iikk}(PP+1) = ww_{iikk}(PP) + \Delta iikk(PP) \quad (10)$$

Where,

$$\Delta ww_{iikk}(PP) = -\mu \frac{\Delta EE(PP)}{iikk(PP)} \quad (11)$$

and  $\mu$  is the learning rate parameter of the back-propagation algorithm.

The learning rate is a parameter that controls how small or large are the changes to the synaptic weights in the network from one iteration to the next. The improvement in the learning is attained at the cost of a slower rate of learning. A too large  $\mu$  may speed up the rate of learning, but the resulting changes in the synaptic weights assume such a form that the network may become unstable.

#### 4.7.1.3 Activation functions

The activation function defines the output of a neuron in terms of the induced local field  $x$ . Through this work, we use two types of activation functions:

1. A linear activation function:

$$f(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \quad (12)$$

The output of a neuron  $k$  using such an activation function is expressed as

$$y_k = \begin{cases} 1 & \text{if } v_{kk} \geq 0 \\ 0 & \text{if } v_{kk} < 0 \end{cases} \quad (13)$$

where  $v_{kk}$  is the induced local field of the neuron, that is,

$$v_{kk} = \sum_{mm} ww_{kmm} x_{mm} + b_{kk} \quad (14)$$

2. The sigmoid function, the sigmoid function can be defined using the following equation,

$$f(v) = \frac{1}{1 + \exp(-av)} \quad (15)$$

where  $a$  is the slope parameter of the sigmoid function. By changing the parameter  $a$ , we can obtain a sigmoid function of different slopes.

## 4.7.2 NN-FPGA Controller Design

### 4.7.2.1 NN Implementation

Implementing a neural network in reconfigurable architectures like FPGAs is an efficient way to calculate weights and network topologies. Network parameters are mapped into a hardware structure that improves the performance and the efficiency. The size of the implemented NN is limited by the block RAM capacity of the FPGA board.

The size of a NN is the amount of neurons synaptic weights available. The number of synaptic weight connections ( $N_{sw}$ ) available for the FPGA board is given by [31]:

$$N_{sw} = \frac{(\text{FPGA Block RAM Size})(\# \text{ of BlockRAMs})}{\text{Size of Synaptic Weights}} \quad (16)$$

Network efficiency is the percentage of processing elements operating in parallel.

$$\text{Eff} = \frac{\# \text{ of neurons in the smallest layer}}{\# \text{ of neurons in the largest layer}} \times 100\% \quad (17)$$

Maximum network efficiency is achieved when the network layers have the same number of neurons. Thus, to increase the efficiency, neurons in the largest layer are multiplexed in groups.

### 4.7.2.2 NN modeling on FPGA procedure

Here we use a reconfigurable antenna with several switches as the device to be controlled. The controller design starts by first building and training a neural network model for the reconfigurable antenna using Matlab, then, Xilinx ISE, the model is sent to an FPGA board. The whole process can be summarized as follows:

1. Measured antenna S11 (input scattering parameters) data are collected, sampled, and normalized.
2. Matlab m-code is written to build and train the NN.
3. A Matlab Simulink model is built for the NN using Xilinx System Generator blocks.
4. VHDL code for the design is generated using Xilinx System Generator.

5. The VHDL code is sent to ISE, where it synthesized, implemented, and sent to the FPGA board.
6. FPGA board output is connected to the antenna input.

Each of these steps are explained thoroughly below.

Step 1) Measured antenna scattering parameter S11 data are collected, sampled, and normalized:

The antenna measured data from a network analyzer are saved as a look up table for the NN and the data are saved in an excel format. The output of the antenna (S11) is the input to the neural network, and the input of the antenna (switch configurations, or voltage level) represents the desired output from the NN.

Step 2) Matlab m-code is written to build the NN:

A Matlab code is written with help of NN toolbox. The code reads the xls data, then, builds an input/output arrays for the NN model. Next a NN model is formed with the following three layers:

- Input layer: has N neurons, where N is the number of points required to reproduce the antenna's S11 values with all resonance frequencies at all possible switch configurations.
- Hidden layer: a single hidden layer is used with a sigmoid activation function. The number of neurons in this layer is determined by trial and error.
- Output layer: the number of neurons in the output layer is equal to number of switches in the antenna.

Step 3) - A Matlab Simulink model is built for the NN using the Xilinx System Generator blocks. In order to generate a valid HDL code, XSG requires a Matlab model to be built in Matlab Simulink using Xilinx blocks. Therefore, the m code is manually built, block by block in Simulink. Luckily, due to the NN structure, most of the design is formed using adders and multipliers. However, although the Matlab Simulink library provided by XSG, has all blocks necessary for the design of an artificial neural network (ANN), there are still a few functions such as the sigmoid function that has to be configured.

Approximations for the sigmoid function were suggested before using a Taylor series expansion [32], or polynomial functions [33]. These approximations however dealt with the sigmoid as a whole function which results in either a complicated costly approximation to implement, or an approximation that has poor accuracy in parts of the function's curve. Here, a different approach is presented. Instead of approximating the sigmoid as one function, it is approximated by dividing the sigmoid in multiple segments and each segment is represented by a simpler function.

The idea is to divide the domain  $x$  in the sigmoid function in equation 4.11 into segments. The more the segments the more accuracy we can obtain but at a higher cost, so the number of segments has to be optimized to an accepted level of accuracy with available FPGA board resources. For this work, five segments was enough to yield accurate antenna modeling on the FPGA board used. Figure 81 shows how the sigmoid function is divided in 5 segments.

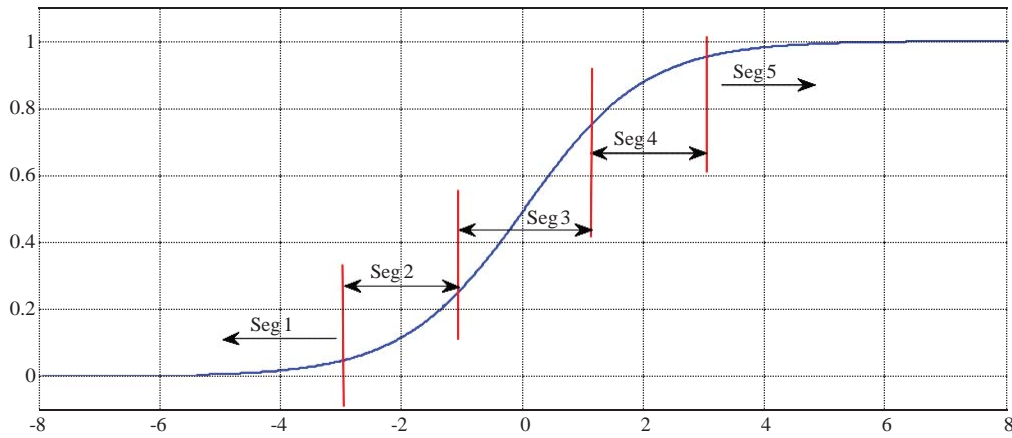


Figure 81. Sigmoid function segments

The segments are as follows:

- I. Segment 1:  $-\infty < x < -3$   
 In this interval, the function is approximated to have the fixed value of 0.02 as shown in Figure 82

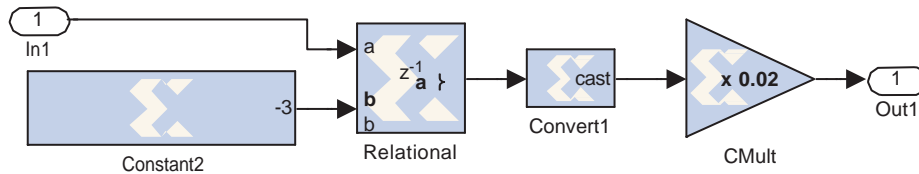


Figure 82. Segment 1

- II. Segment 2:  $-3 < x < -1.1$   
 In this interval, the function is approximated as a quadratic function:

$$q:2(x) = 0.03809x^2 + 0.2617x + 0.4893 \quad (18)$$

Segment's 2 function model is shown in Figure 83

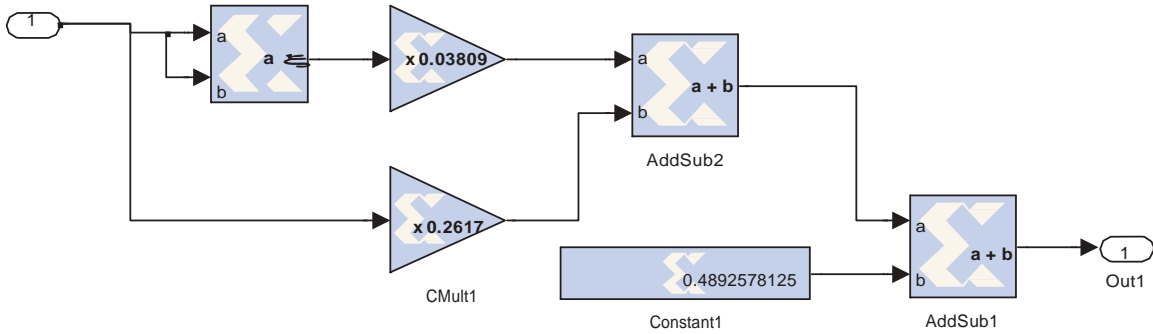


Figure 83. Segment 2

III. Segment 3:  $-1.1 \leq x \leq 1.1$

In this interval, the function is approximated as a linear function:

$$q:3(x) = 0.2314x + 0.5 \tag{19}$$

Segment's 3 function model is shown in Figure 84

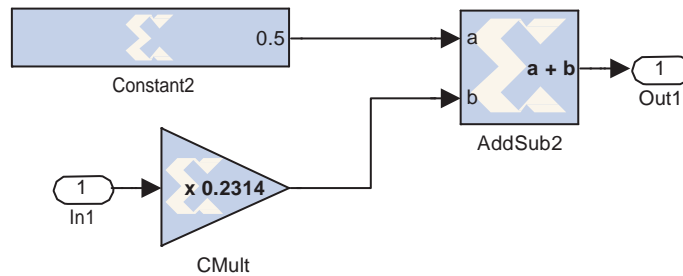


Figure 84. Segment 3 function

IV. Segment 4:  $1.1 < x < 3$

In this interval, the function is approximated as a quadratic function:

$$q:4(x) = -0.03809x^2 + 0.2617x + 0.1507 \tag{20}$$

Segment's 4 function model is shown in Figure 85.



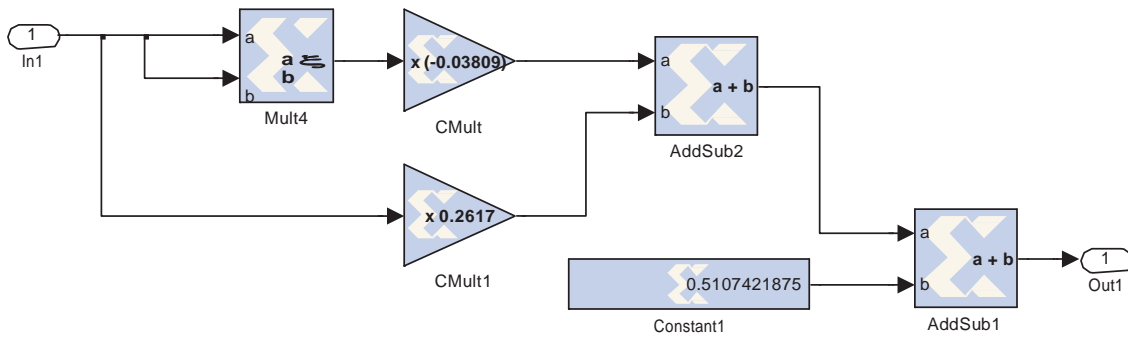


Figure 85. Segment 4 function

V. Segment 5:  $3 \leq x < \infty$

In this interval, the function is approximated to have fixed value of 0.98 as shown in Figure 86.

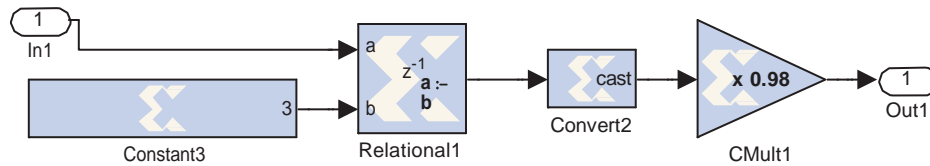
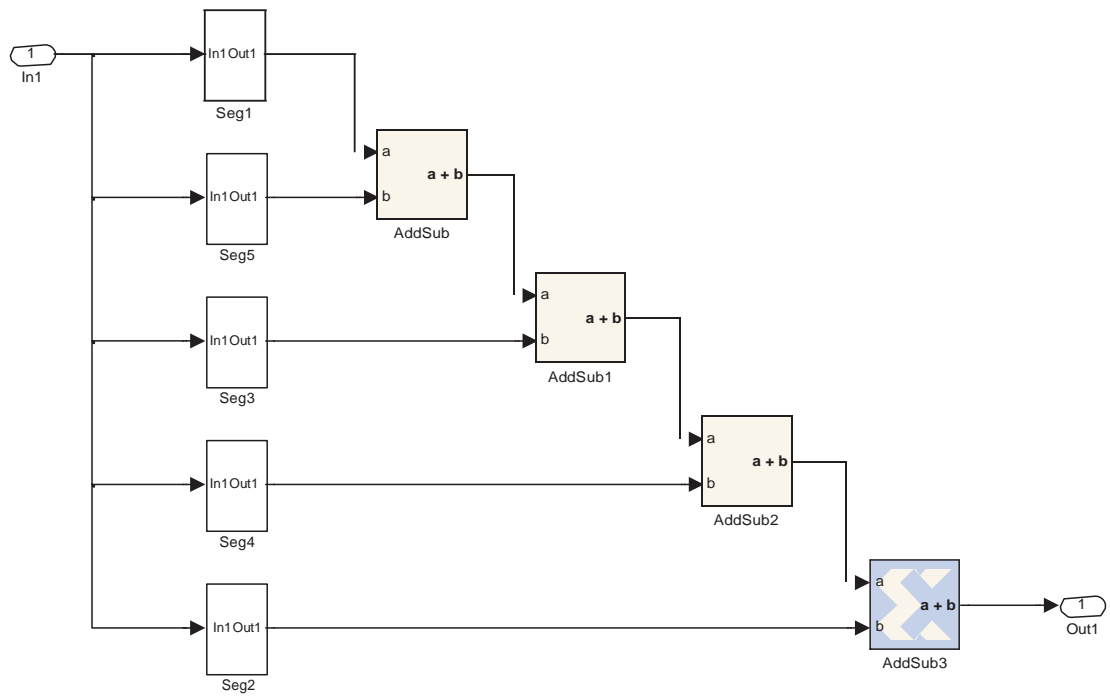


Figure 86. Segment 5

The overall approximation function is given by:

$$q:_{App}(x) = \begin{cases} 0.02 & , -0.0 < x < -3 \\ 0.03809x^2 + 0.2617x + 0.4893 & , -3 < x < -1.1 \\ 0.2314x + 0.5 & , -1.1 \leq x \leq 1.1 \\ -0.03809x^2 + 0.2617x + 0.1507 & , 1.1 < x < 3 \\ 0.98 & , 3 \leq x < \infty \end{cases} \quad (21)$$

Figure 87 shows the entire sigmoid function as a Simulink model.



**Figure 87. Sigmoid Simulink model**

The VHDL code for the design is generated using the Xilinx System Generator. The System Generator block (Figure 88) provides the control of the system and simulation parameters, and is used to invoke the code generator.

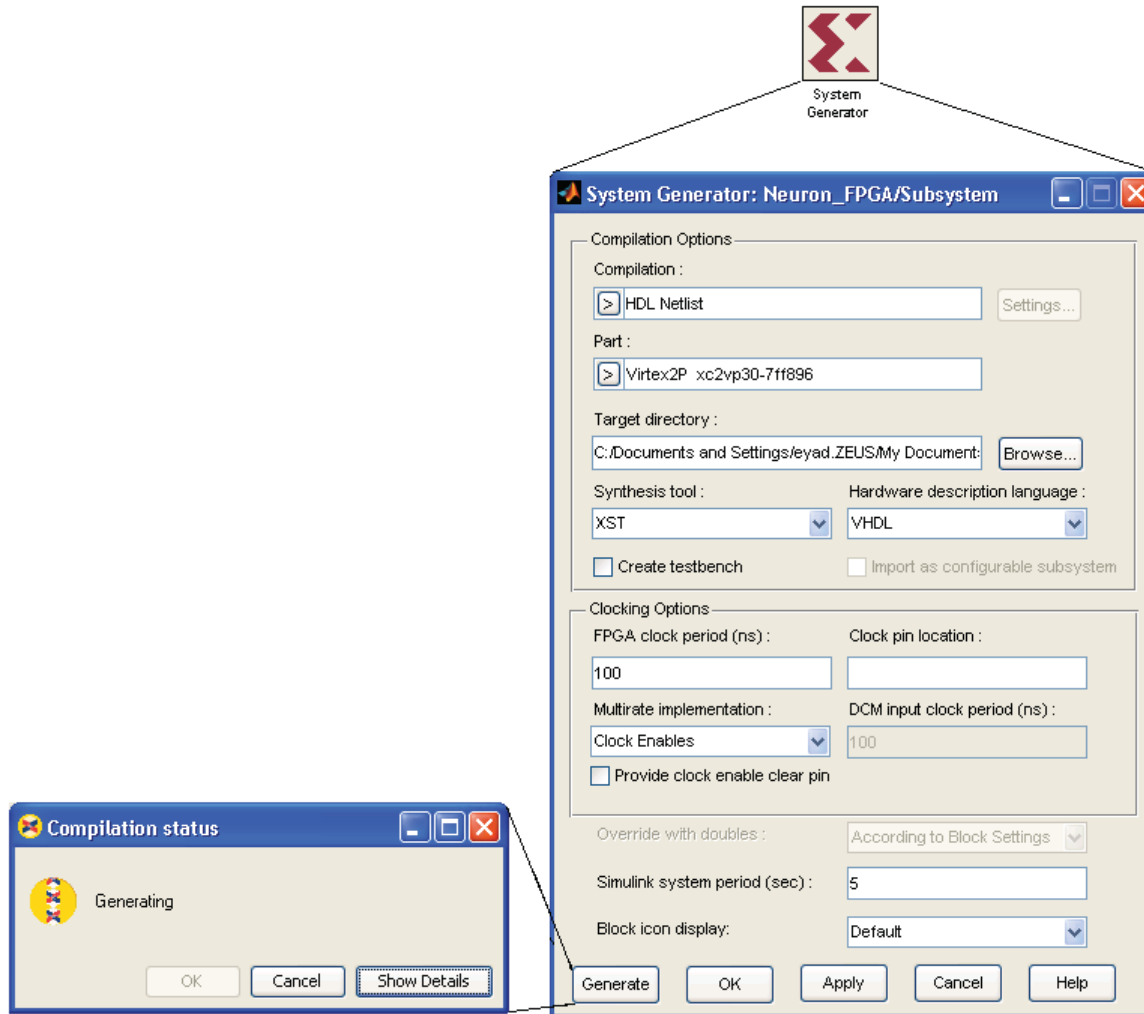


Figure 88. System Generator block

Next the VHDL code is sent to ISE. These generated files are eventually transferred to the FPGA board.

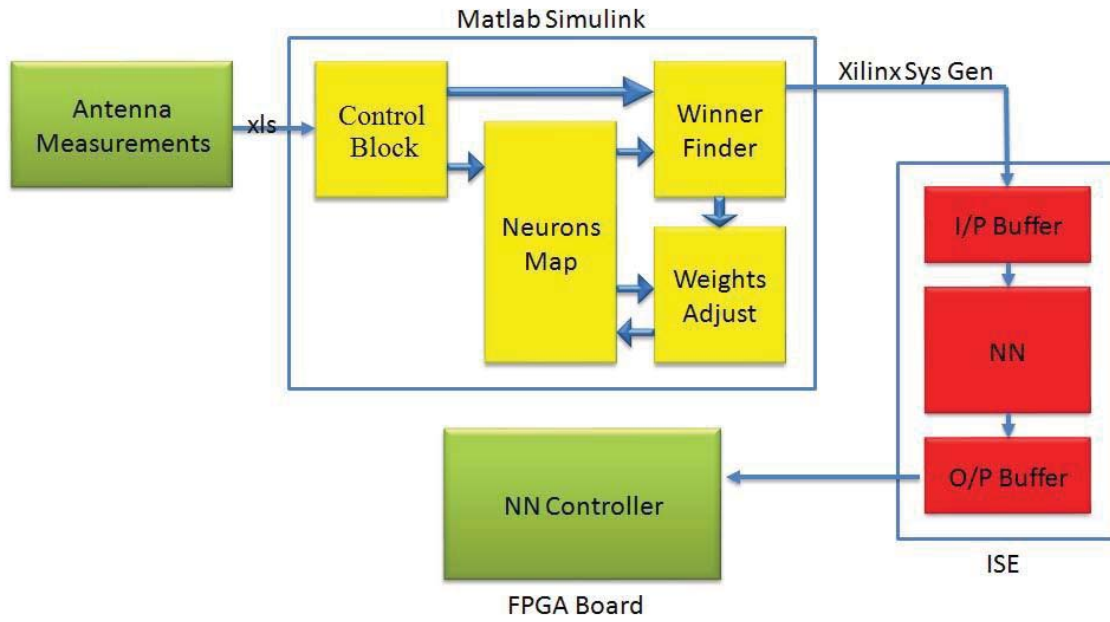


Figure 89. A NN-FPGA controller hardware architecture



Figure 90. FPGA board connections where The FPGA board output is connected to the antenna input

### 4.7.3 Simulations and Results

The designed NN-FPGA controller was applied to different reconfigurable antennas. For each antenna, a NN-FPGA model was built and analyzed. The FPGA board used was the Xilinx ML403. Next, a few examples are presented to demonstrate the validity of controlling reconfigurable systems using neural networks embedded in an FPGA, such as the ones shown in Figures 89 and 90.

#### 4.7.3.1 *Reconfigurable Filtenna Antenna*

The first example is applied on a reconfigurable filtenna. The idea behind this was to find out if the NN network can control a reconfigurable band-pass filter within the feeding line of the antenna [34]. Thus, the antenna is able to tune its frequency based on the filter's operation.

The filtenna structure consists of a dual sided Vivaldi antenna which is a wideband structure [35]. It is fed via a 50 ohms microstrip line which corresponds to a width of 5 mm. The antenna has a partial ground which is the ground plane of the filter of dimensions 30 mm x 30 mm. The structure is printed on a Taconic TLY substrate of dimension 59.8 mm x 30 mm. The inner and outer contours of the antenna radiating surface are based on an exponential function. The top layer constitutes the first side of the antenna radiating surface as well as the feeding line where the reconfigurable filter is located. On the bottom layer of the design resides the ground plane of the filter connected to the second radiating part of the Vivaldi antenna. The filtenna top and bottom layers are shown in Figure 91.

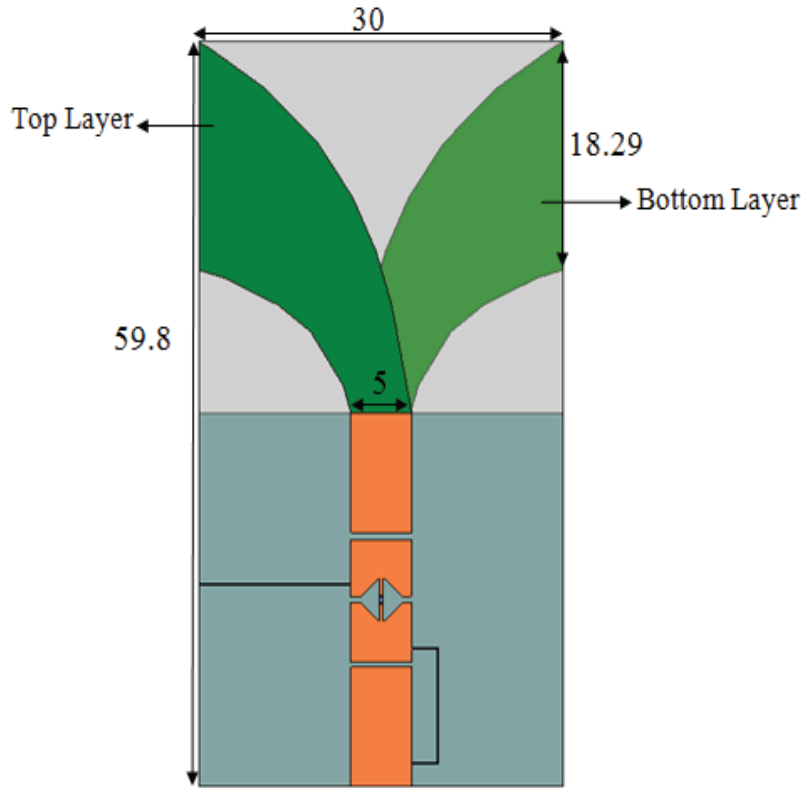


Figure 91. Filtenna antenna structure

Table 4. Neural network parameters for Filtenna antenna

Iterations	3
Input Neurons	1
Output Neurons	1
Hidden Layers	1
Hidden Neurons	2

As shown in Table 4, the NN structure of this antenna requires only one input neuron, one output neuron, and two hidden neurons. The input of NN is the desired resonant frequency of the antenna, and the output is the voltage level required to bias the varactors

Figure 92 shows the Simulink NN\_FPGA model of the antenna. Layer 1 represents the hidden layer, while layer 2 is the output layer. Subsystems 1 and 2 are used for pre- and post-processing for the input and output signals.

Figures 93 to 95 show the breakdown of the hidden and output layers.

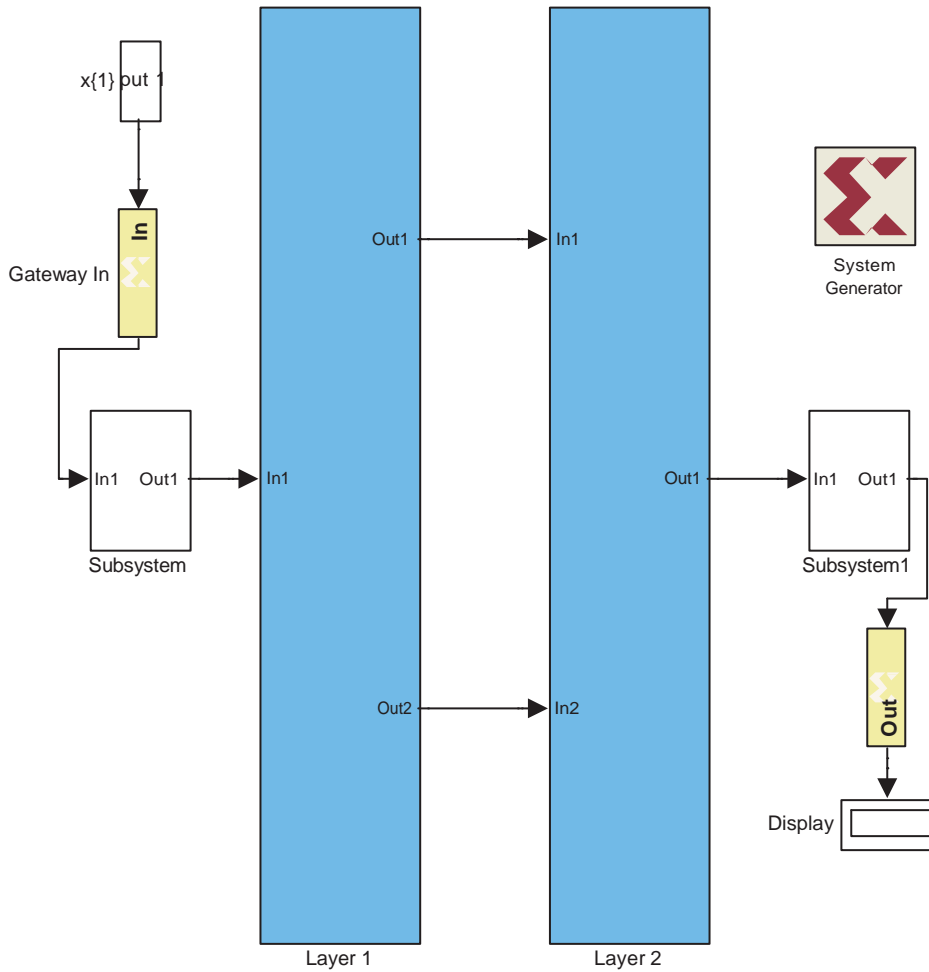


Figure 92. Simulink NN\_FPGA model of the antenna

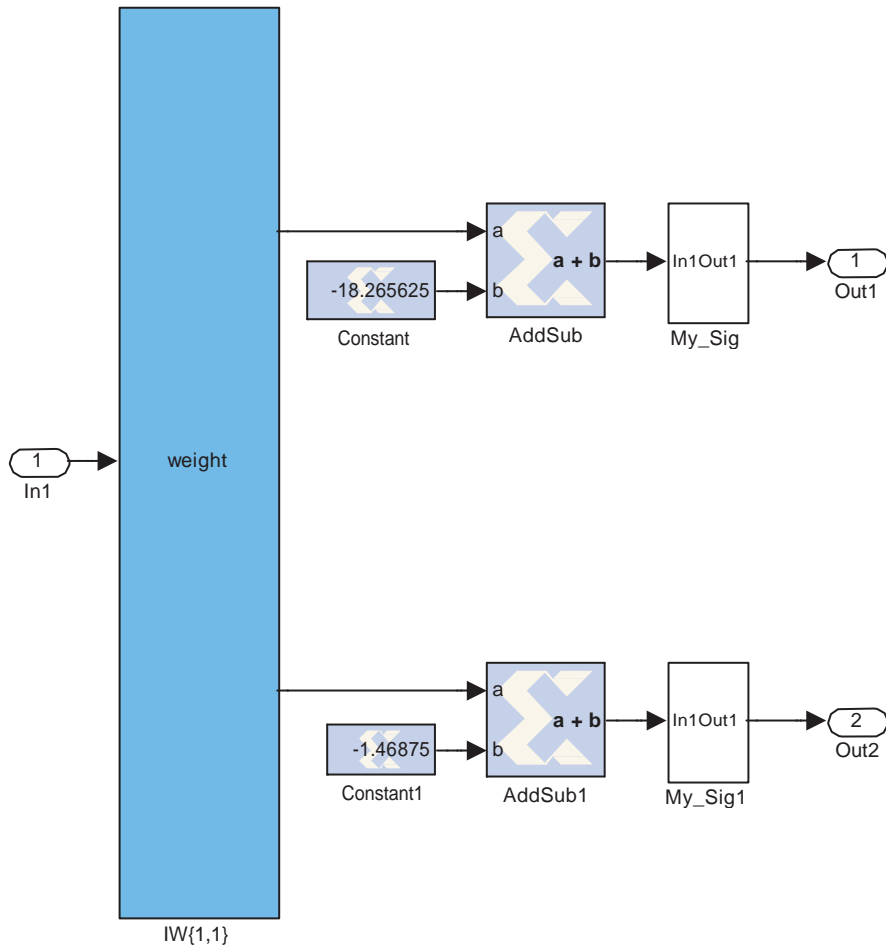


Figure 93. Hidden layer of filtenna



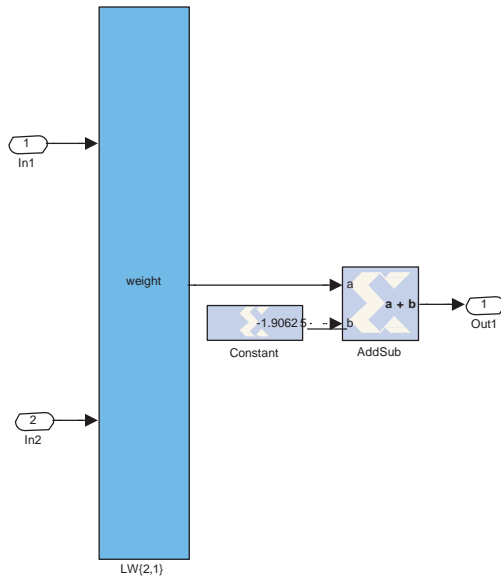


Figure 94. Output layer of filter

Figure 5.37 Output layer of filter

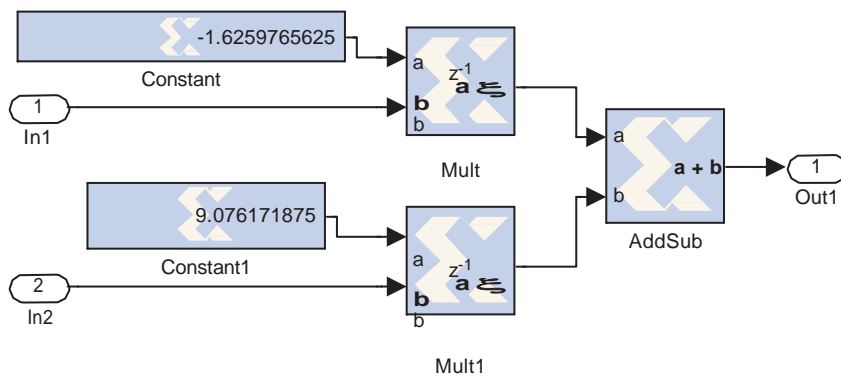


Figure 95. Output neuron structure

Figure 96 shows the neural network output compared to the measured antenna response. Each point on the curve represents the resonance frequency at that specific input voltage. The training error was  $10^{-5}$ .

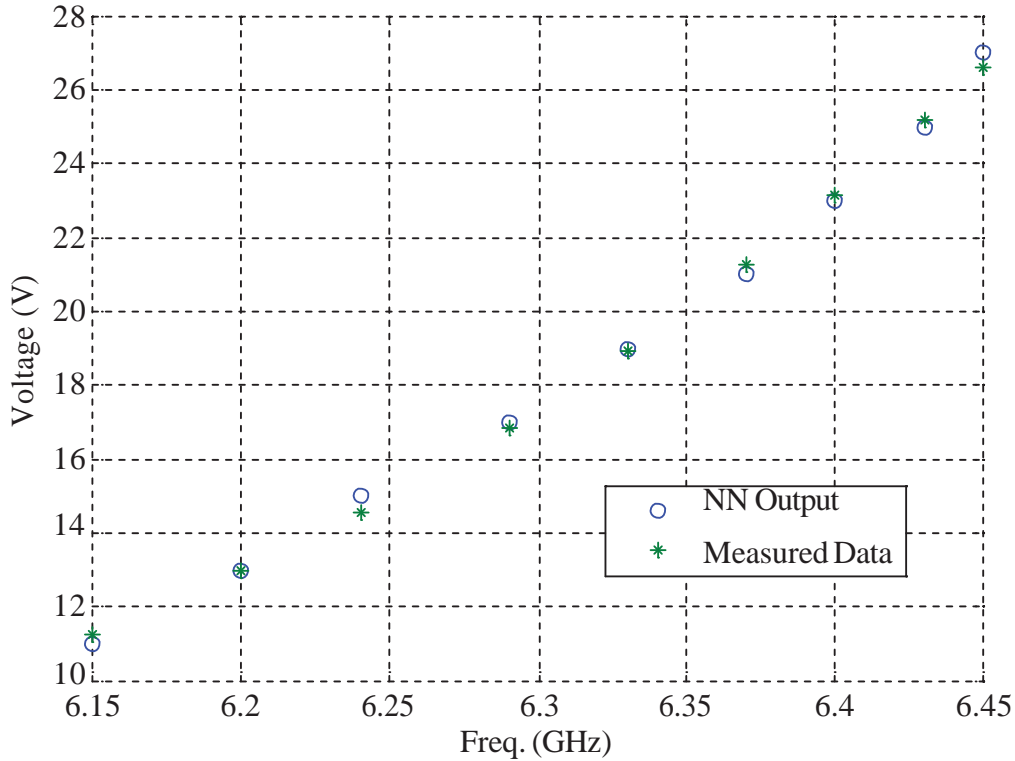


Figure 96. NN output vs measured antenna response

#### 4.7.3.2 A Rotatable Microstrip Antenna

In this example, a rotatable microstrip antenna [36] was modeled. Reconfigurability is achieved via a rotational motion of a part of the antenna patch while maintaining the same omnidirectional radiation pattern in both the E and H planes. The rotating part has the form of a circle and contains four different shapes. Each shape corresponds to a different antenna structure. With every rotation, a different antenna structure is fed in order to produce a different set of resonant frequencies. Four different rotations can be done making the antenna cover five different bands (from 2 GHz up to 7 GHz) correspondingly.

The corresponding antenna structure is shown in Figure 97. It consists of two layers. The bottom layer is a partial ground to allow radiation above and below the substrate. The top layer is a rotating circular shape. The chosen substrate is Taconic TLY with a dielectric constant of 2.2 and a thickness of 1.6 mm. Figure 98 shows NN representation of the antenna.

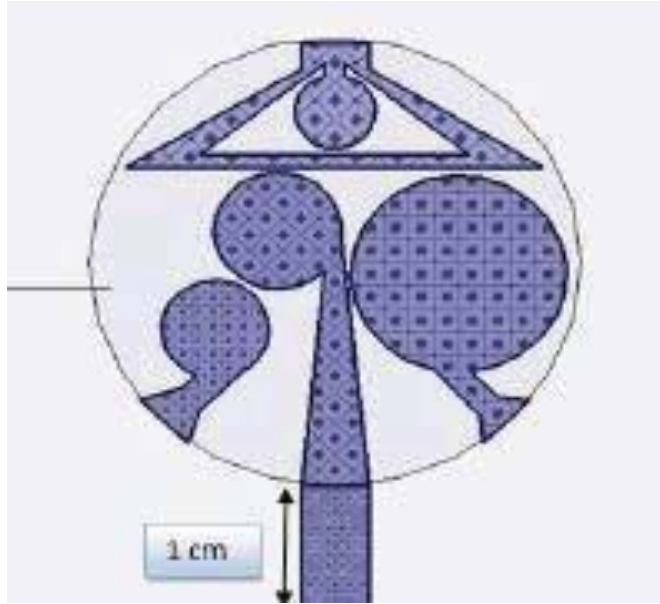


Figure 97. Antenna Structure

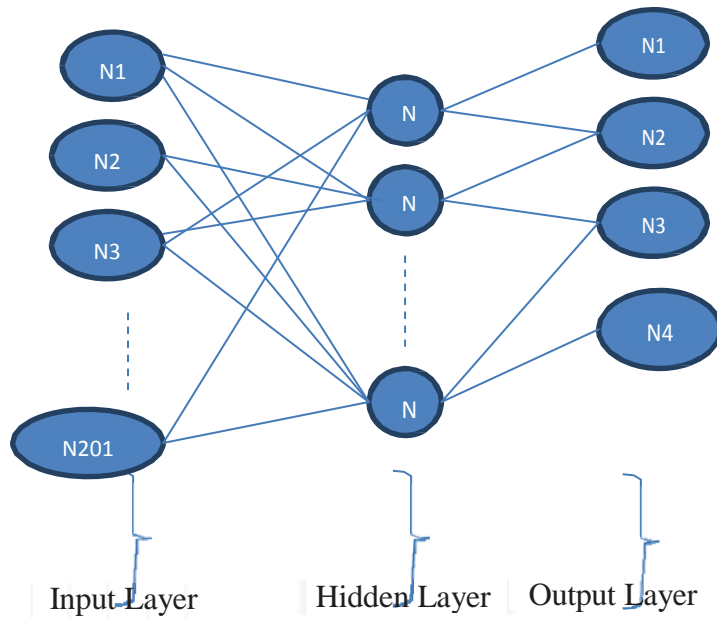


Figure 98. Neural network representation for this antenna

Table 5. NN parameters of Rotatable Microstrip Antenna

Iterations	6
Input Neurons	201
Output Neurons	4
Hidden Layers	1
Hidden Neurons	7

As shown in Table 5, the neural network training took 6 iterations to achieve the required accuracy with 201 input neurons, 7 hidden neurons and 4 output neurons. Figures 99 to 101 show the neural network output compared to the measured antenna response for different shapes.

**Shape 1:**

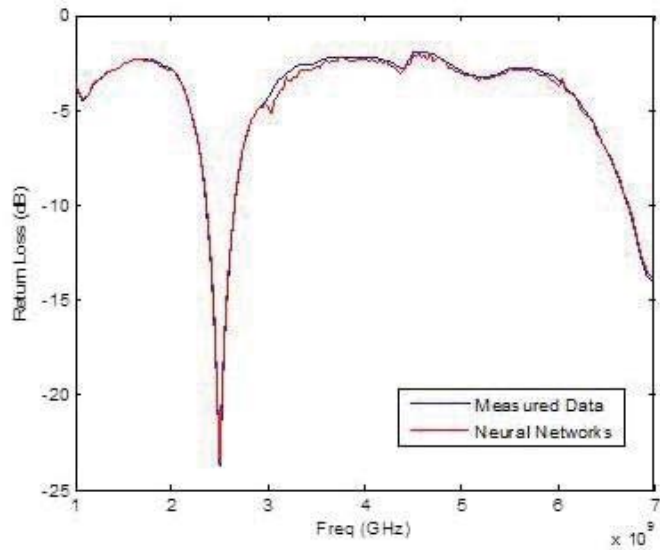
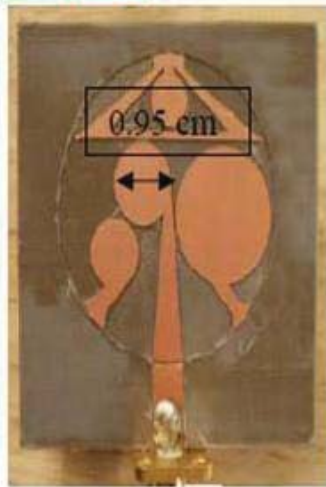


Figure 99. NN output vs measured antenna response for shape 1

**Shape 2:**

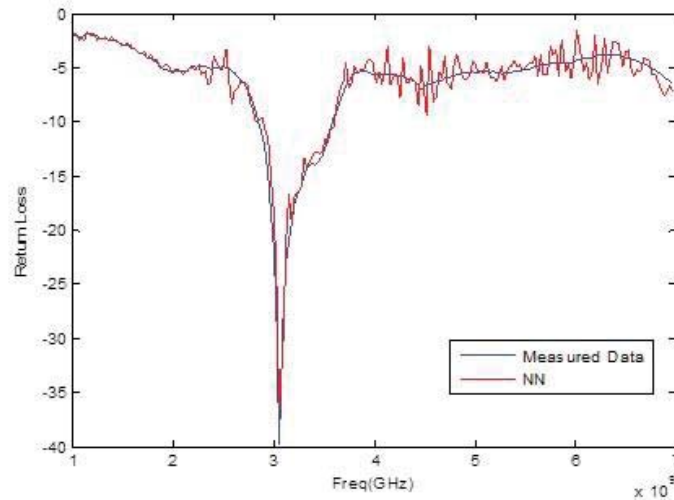


Figure 100. NN output vs measured antenna response for shape 2

**Shape 3:**

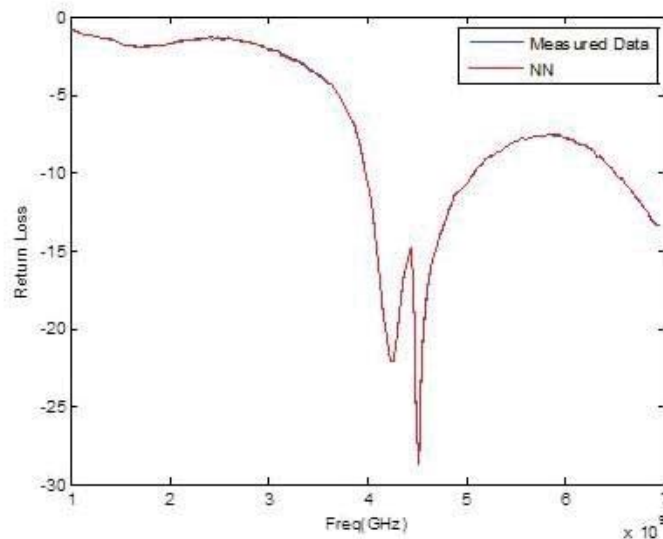


Figure 101. NN output vs measured antenna response for shape 3

#### 4.7.3.3 Broadband Tapered Slot Antenna

In this antenna, a reconfigurable defected microstrip structure (DMS) band pass filter is integrated with a broadband antenna [37]. The filter has a T-shape slot of dimension 2.25 mm x 2.8 mm and a coupling gap of dimension 0.2 mm. The purpose of the gap is to allow the filter to have the “band-pass” feature by functioning as a parallel-series resonance. The proposed design was printed on the Taclam plus substrate with a dielectric constant  $\epsilon\epsilon_{rr} = 2.1$  and a thickness 1.6 mm. The fabricated prototype and the dimensions of the T-shape DMS bandpass filter are

shown in Figure 102.

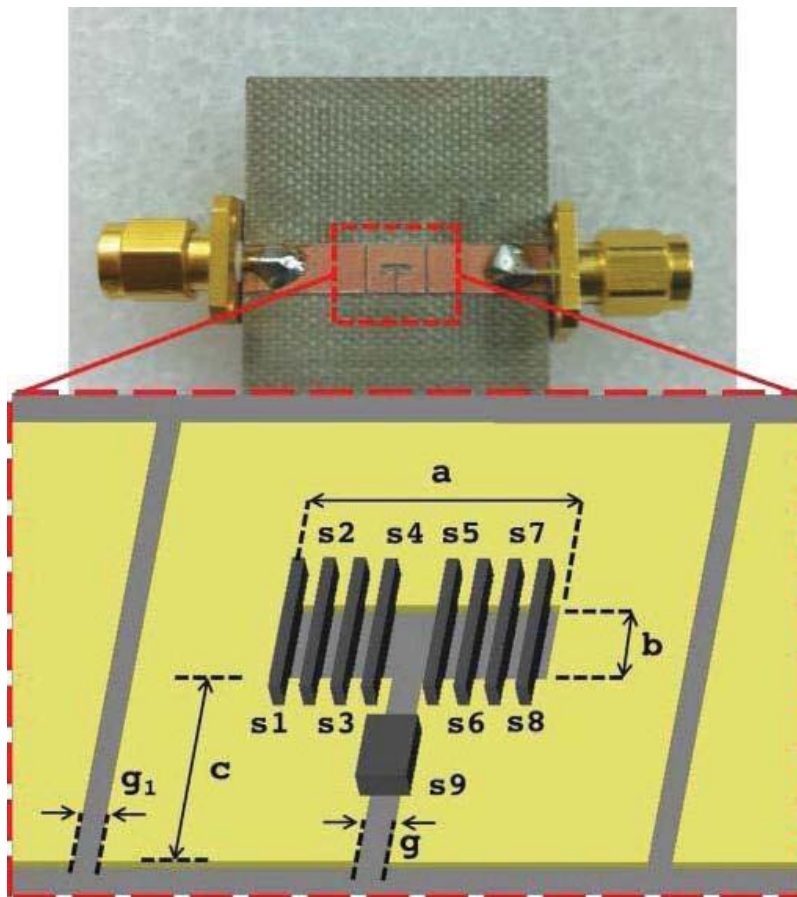


Figure 102. Broadband Tapered Slot Antenna

The reconfigurability of the filter was achieved by integrating 9 switches within the T-slot. The switches are activated in pairs of two from the two edges of the T-slot. The purpose of the switches is to change the dimension of the slot (labeled 'a' in Figure 5.45) in order to produce a reconfigurable band pass filter.

Table 6. NN parameters of Broadband Tapered Slot Antenna

Iterations	15
Input Neurons	105
Output Neurons	9
Hidden Layers	1
Hidden Neurons	11

As shown in Table 6, the NN model requires 15 iterations to converge. The NN structure has 105 input neurons, 9 output neurons, and 11 hidden neurons in a single hidden layer. The performance of the fabricated DMS filter was measured. Three different cases were taken as shown in Figure 103.

- Model 1: All Switch ON
- Model 2: All Switch OFF
- Model 3: 8 Switches ON

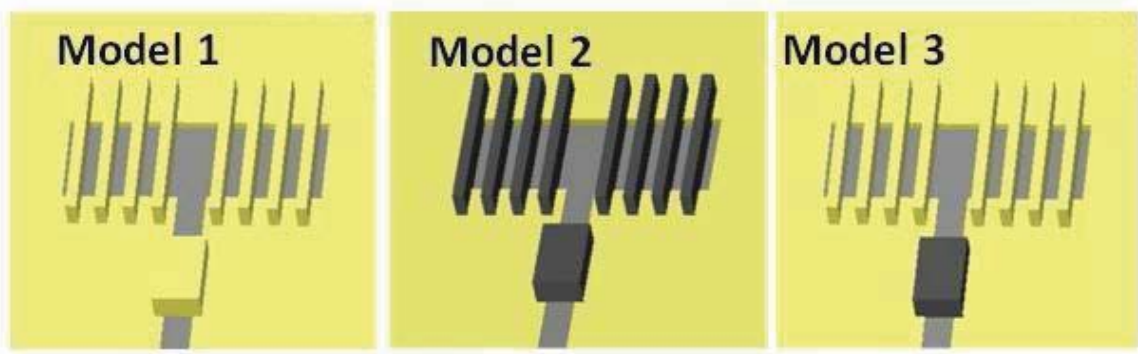


Figure 103. The 3 models used for modeling

The antenna response compared to the NN results for the three models are shown in Figures 104 to 106.

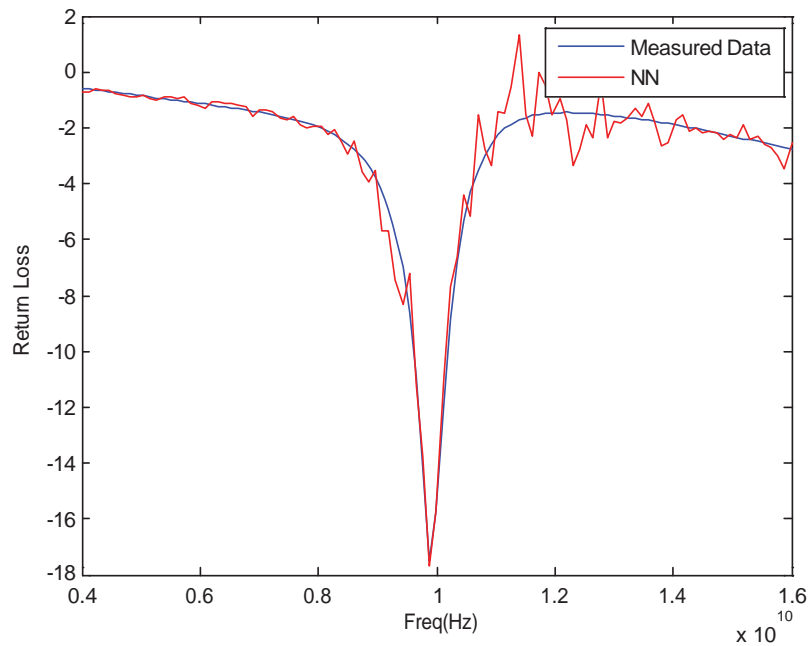


Figure 104. NN output vs measured antenna response for Model 1

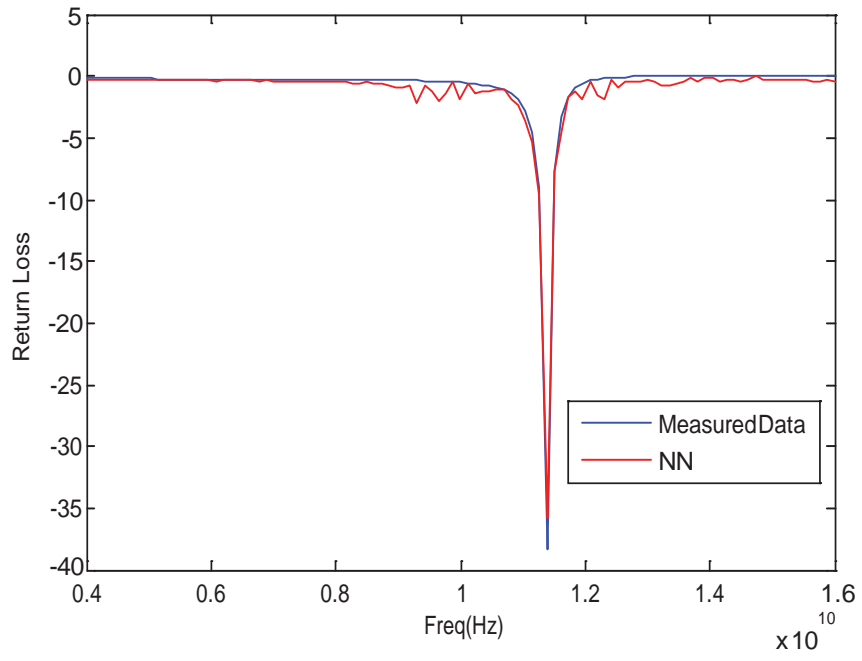


Figure 105. NN output vs measured antenna response for Model 2

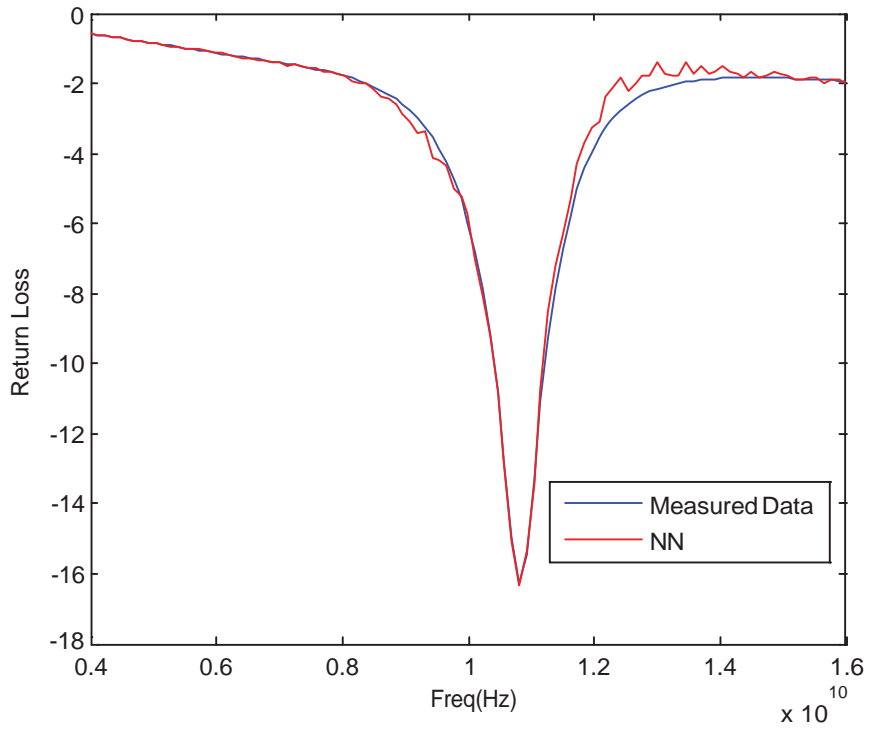


Figure 106. NN output vs measured antenna response for Model 3



#### 4.7.3.4 Star Antenna

Applying NNs on a reconfigurable star antenna with 6 switches connecting different parts together [38] results in 6 input neurons, 11 hidden neurons, and 51 output neurons. The antenna structure is shown in Figure 107, and its NN structure is represented in Figure 108.

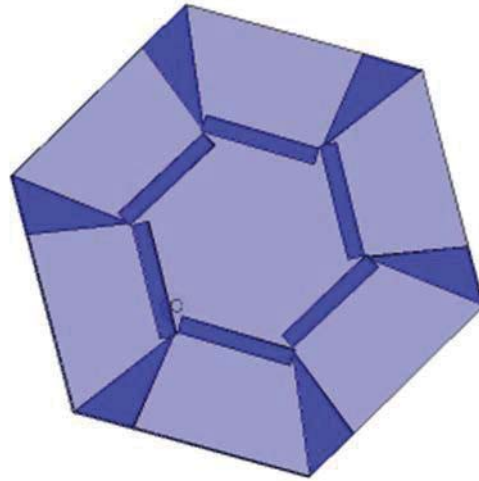


Figure 107. Star antenna structure

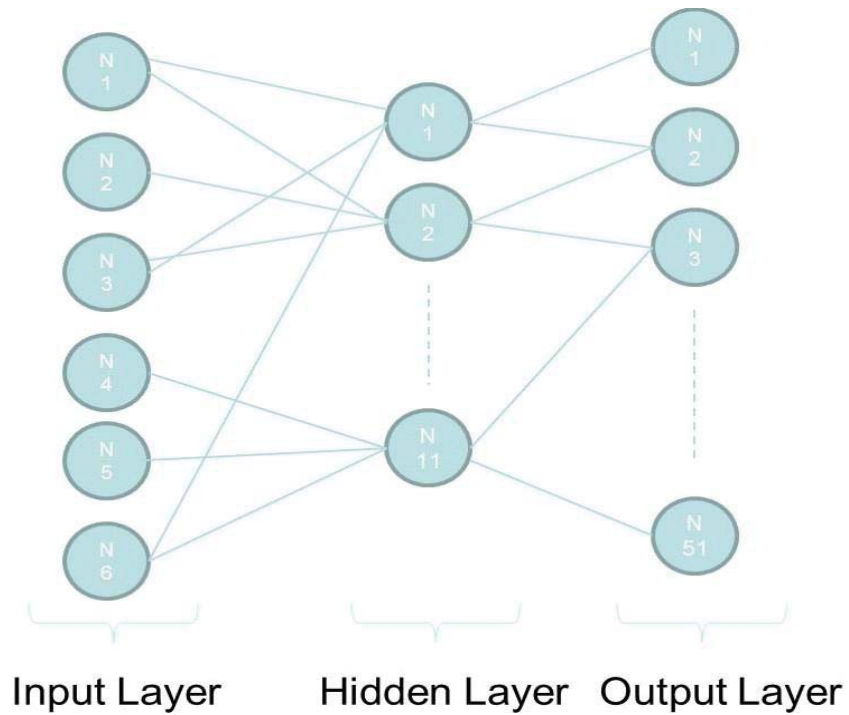
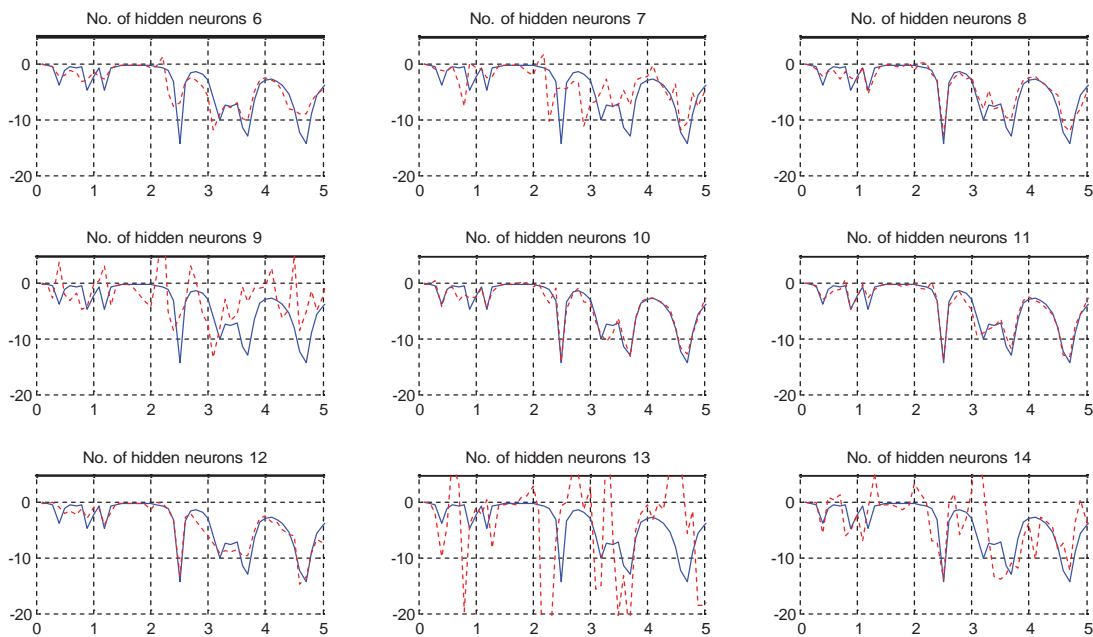


Figure 108. A NN model for the Star antenna

**Table 7. The star antenna neural network parameters**

Iterations	18
Input Neurons	6
Output Neurons	51
Hidden Layers	1
Hidden Neurons	11

As shown in Table 7, the neural network training took 18 iterations to achieve the required accuracy with 11 hidden Neurons and 51 output neurons. The number of hidden neurons is found to be 11, as shown in Figure 109.



**Figure 109. NN output (red) vs measured antenna response (blue) and determination of number of hidden neurons**

Figures 110 and 111 show the neural network output compared to the measured antenna response for some different switch configurations.

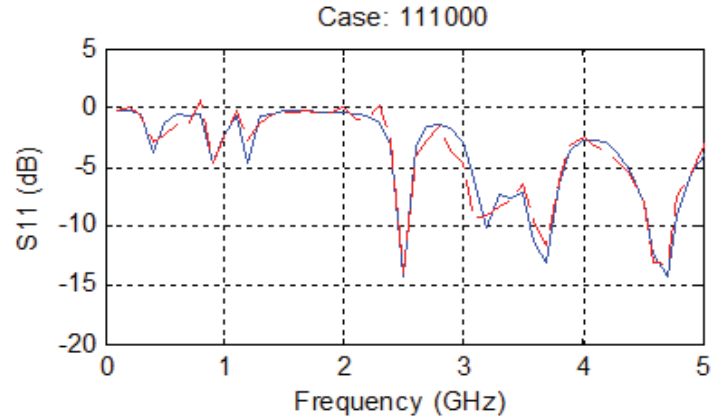
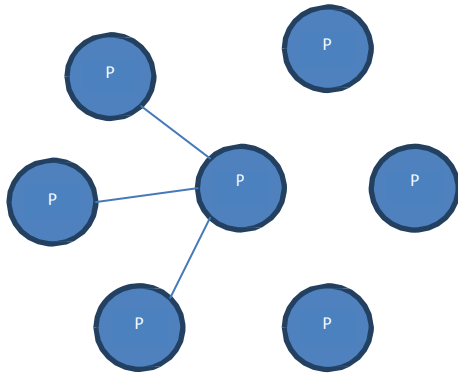


Figure 110. NN output vs measured antenna response for case 111000

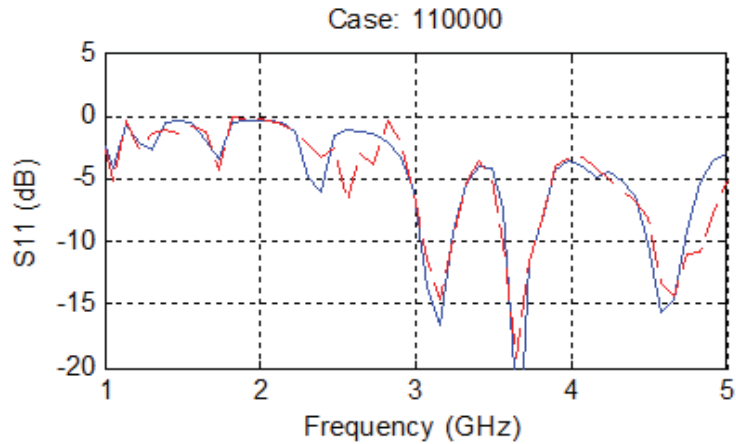
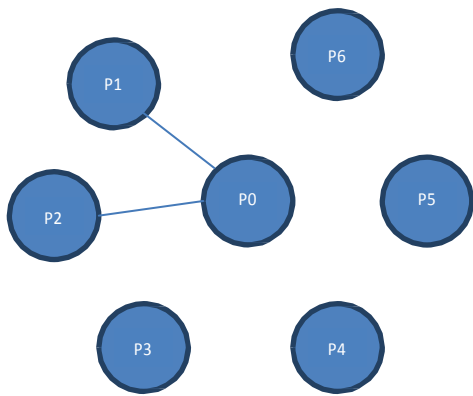


Figure 111. NN output vs measured antenna response for case 110000

#### 4.7.4 Conclusion and future work

A design approach for embedding a neural network FPGA controller with a reconfigurable antenna was described. The network blocks were built in the Matlab Simulink environment, and by using basic Xilinx System Generator blocks. The NN network parameters were mapped into a hardware structure that improves, both the performance and efficiency of the network. Several examples of reconfigurable antennas were given, showing the procedure of building the NN-FPGA controller that represents antenna parameters.

The design presented requires the user to first build and train a NN in a Matlab environment. In the future, NN training and building could be done inside the FPGA board itself. Thus, a general NN code can be sent to the FPGA board, and then the reconfigurable system that is connected to the board can be programmed to take input/output pairs from the reconfigurable system to actually train the NN. The NN will then be able to build and train itself by itself. This can yield a highly reconfigurable, real-time, controller that is able to adapt with any changes in the reconfigurable system.

## **5. Results for Developing and demonstrating performance-level and mission reliability models that will permit quick assessment of complex electronic architectures for space and defense systems.**

The aim of this part of the work was to develop not only new reconfigurable system designs but to also establish guidelines for the design and optimization of these types of systems. We applied some new guidelines in the area of reconfigurable antennas that can be extended to other reconfigurable space electronics and devices. In addition, we demonstrated the correlation between reliability and complexity of reconfigurable antennas mathematically. Information theory was used to predict the probability of error in such systems. This work also presents the different methods that can be utilized to adjust to failures of an antenna system and to ensure a smooth functioning of the defected reconfigurable antenna.

### **5.1 Graph Modeling**

Graphs are symbolic representations of relationships between different components of a system. They are mathematical tools used to model complex systems in order to organize them and improve their status. A graph is defined as a collection of vertices that are connected by lines called edges [39]. A graph can be either directed or undirected. The edges in a directed graph have a certain determined direction, while this is not the case in an undirected graph. Vertices may represent physical entities while the edges between them in the graph represent the presence of a function resulting from connecting these entities. In a switch reconfigurable antenna, an edge represents the connection that occurs once a switch is activated. Edges may have weights associated with them. These weights represent costs or benefits that are to be minimized or maximized.

A graph is used as an abstract model to represent physical structures [39]. Graph modeling reconfigurable antennas transform them into software accessible devices [24,38] that are easy to optimize, control and automate. In addition to these functionalities the modeling of reconfigurable antennas using graphs leads to a redundancy reduction approach that eliminates unnecessary components from the structure. Thus, graph models are utilized to formulate a reconfigurable antenna's complexity [40]. The overall complexity of an antenna system increases with the number of p-i-n diodes [41,42], RF MEMS [8-10], varactors [43,44] or optical switches employed [45,46]. Additional details on graph theory and what guidelines can be used to design and optimize reconfigurable antennas can be found in Appendix A.

In here we show how previous formulations done on single element reconfigurable antenna [40] can be extended to evaluate the complexity and reliability of reconfigurable antenna arrays using graphs. This is essential to address the continuous functioning of these arrays in unknown conditions and harsh environment conditions such as space.

A technique is proposed to improve the performance and reliability of reconfigurable antenna arrays. This technique is based on rearranging antenna configurations to ensure a higher reliability. The improvement that such rearrangement introduces to the design efficiency and operation is discussed.

### 5.1.1 Graph Modeling of Reconfigurable Antenna Arrays

A graph model of an array of reconfigurable antennas is very different than that of a single reconfigurable element. In an antenna array all elements are fed through a feeding network and are placed at certain spacing from each other [47,48]. As an example, let us consider the antenna array shown in Fig. 1. The array is composed of three layers. The bottom layer constitutes a common ground plane for the different elements. The middle layer constitutes the substrate Taconic TLX with a dielectric constant  $\epsilon_r=2.55$  and height 2.9 mm. The upper layer constitutes the different element patches as well as the corporate feeding network. Each element is a rectangular patch with 2 rectangular slots dividing it into 2 sections connected constantly. Two switches are placed in each element to bridge over the upper and lower part of the slots. The end-points of each switch are indicated by the nodes  $\{P_{ij}, P'_{ij}\}$  (equation 5.1) where  $i$  refers to the element number and  $j$  refers to the switch position. For example  $P_{21}P'_{21}$  represent the end-points of the lower switch (S4) in element 2. The array is fed with a corporate feed as shown in Figure 112. The array's reflection coefficient for 3 different switch combinations is shown in Figure 113. The array's fabricated prototype is shown in Figure 114.

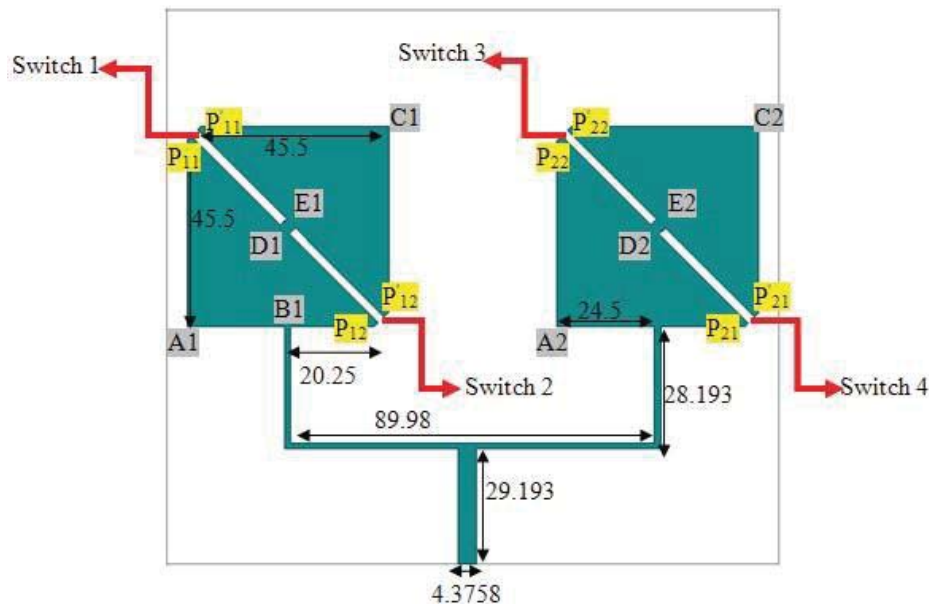


Figure 112. A two element reconfigurable antenna array with dimensions in “mm”

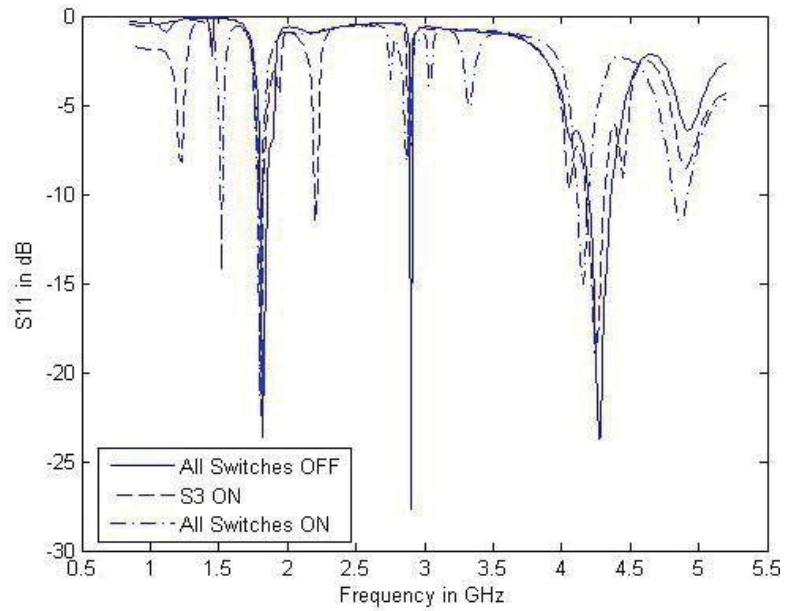


Figure 113. The return loss for the array in Figure 3 for three different switch configurations. S3= S21S'21

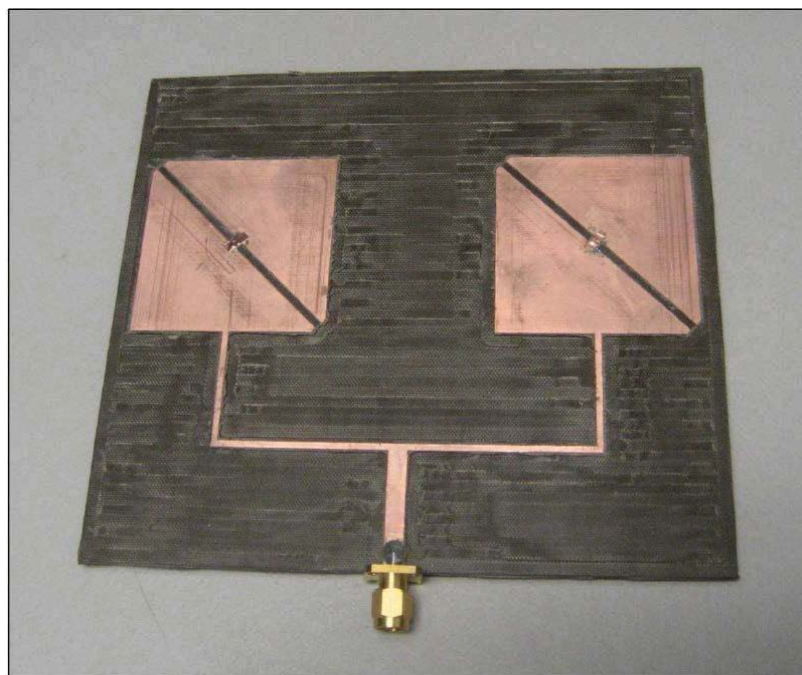


Figure 114. The Fabricated antenna array prototype

A comparison between the measured and simulated reflection coefficient of the array antenna when S1 is ON, is shown in Figure 115 indicating good agreement.

$$\begin{aligned}
 \{P_{ij}, P'_{ij}\}: & \begin{array}{l}
 I_{i=1, j=1} \quad \text{Element1, upper switch, Switch 1 (S1)} \\
 I_{i=1, j=2} \quad \text{Element1, lower switch, Switch 2 (S2)} \\
 I_{i=2, j=1} \quad \text{Element2, upper switch, Switch 3 (S3)} \\
 I_{i=2, j=2} \quad \text{Element2, lower switch, Switch 4 (S4)}
 \end{array}
 \end{aligned}
 \tag{22}$$

The graph model of this array is shown in Figure 116 where each vertex represents the intersection of any two lines forming the array structure. For example, the connection between vertices  $D_1$  and  $E_1$  represents the constant connection between the two triangular parts of the first array element; while the edge  $P_{21}A_2$  represents the connection between one end-point of the switch  $S_3$  and the right lower corner of the second array element as shown in Figure 112. The end-points of each switch are also represented by vertices where an edge between them represents the activation of that switch.

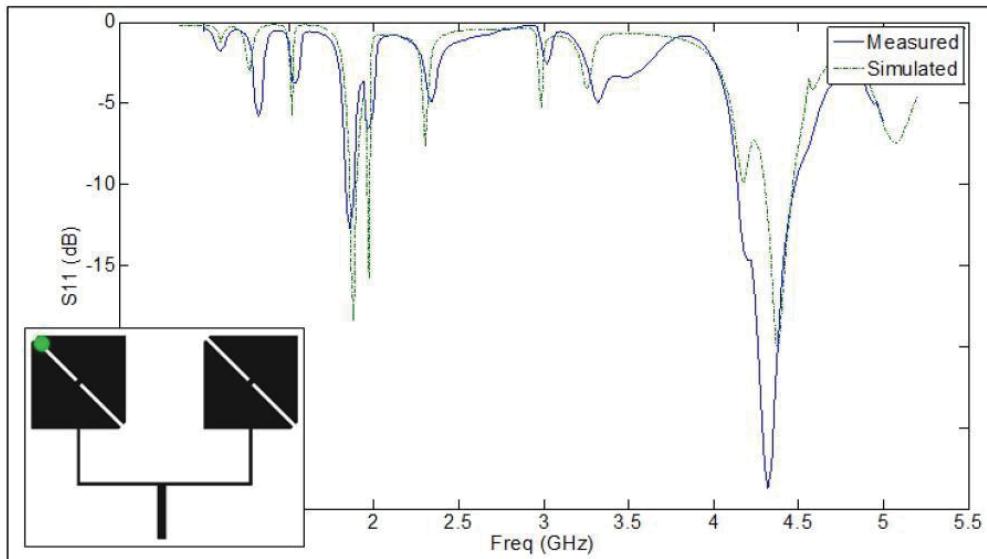


Figure 115. A Comparison between the measured and simulated reflection coefficient when S1 is activated

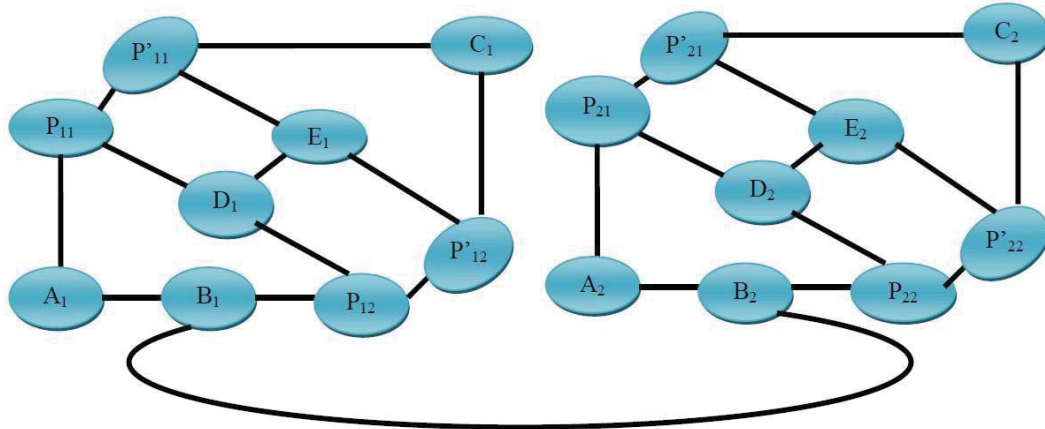


Figure 116. General form of a graph model for the antenna array

Figure 6.6 General form of a graph model for the antenna array

The graph model shown in Figure 116 is a general representation of modeling an antenna array where all the parameters that go into the antenna array design are taken into consideration. In this work we were only interested in studying the complexity of a reconfigurable antenna array and since this complexity is directly related to the number of switching elements in such structure, the graph model shown in Figure 116 can be simplified as shown in Figure 117. The simplified graph models only the connections between the end-points of the four switches.

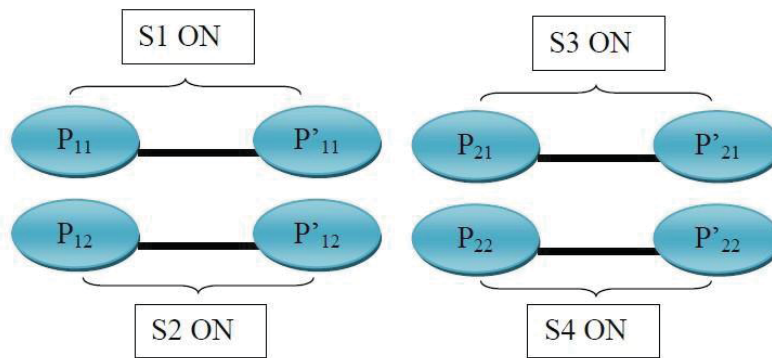


Figure 117. Simplified graph model for the array antenna

The graphs representing arrays are frequency dependent and each graph represents a different antenna behavior based on the frequency of operation. This frequency dependence is considered to be dependent on single narrowband frequencies. For example this array resonates at 2.205 GHz when {S1, S2, S3}, {S2, S3, S4}, or {S3} are ON and at the same time the array preserves the same elliptical polarization for all three cases. Thus at  $f=2.205$  GHz the array is graph modeled in three graphs  $\{G_1, G_2, G_3\}$  that represent the operation of the antenna at the same frequency while preserving same radiation characteristics. These graphs are considered equivalent. The equivalence is purely based on the antenna behavior in different configurations. Different switch configurations that yield same antenna operation in terms of frequency and radiation characteristics are modeled by graphs that are considered equivalent. Thus, equivalent

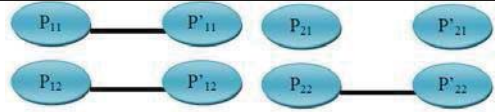
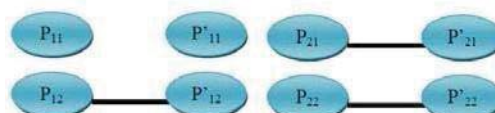



graphs are defined as different models that represent the antenna in various configurations operating at the same frequency with the same polarization and radiation properties. These graphs shown in Table 8 are equivalent at 2.205 GHz however they are not at other frequencies. Such frequency dependence can be summarized in equation 5.2. The use of frequency dependent graphs facilitates the formulation of the antenna's complexity and reliability that are also frequency dependent [40].

$$G_n(f_1) \} G_m(f_2) \text{ iff } f_1 = f_2 \text{ and same radiation characteristics} \quad (23)$$

$G_n$  and  $G_m$  model the same antenna array.

**Table 8. Equivalent graphs representing equivalent configurations at 2.205 GHz**

Config.1: Graph $G_1$ (2.205GHz)	
Config.2: Graph $G_2$ (2.205GHz)	
Config.3: Graph $G_3$ (2.205GHz)	

## 5.2 Correlation Between Complexity and Reliability

In this section we discuss the different complexity and reliability parameters governing the various reconfigurable antenna array functions.

### 5.2.1 The General Complexity of reconfigurable Antenna arrays

The general complexity of reconfigurable antennas has been previously defined in [40] as equivalent to the number of edges (NE) in a graph model. This definition is represented in equation 5.3 where the complexity (C) can be defined as the size of the graph.

$$C_{\text{single element}} = NE \quad (5.3) \quad (24)$$

The general complexity of an M\*N element reconfigurable antenna array can be extended from equation 5.3 to take into consideration the total number of elements. Hence the array's general complexity is the summation of each element's general complexity represented as shown in equation 5.4

$$C_{\text{reconfigurable array}} = \bigwedge_{i=1, j=1}^{M, N} NE_{ij} \quad (25)$$

where M represents the number of rows in an array, N the number of columns,  $NE_{ij}$  represents the number of edges for all possible configurations in the elements existing in row i and column j.

The general complexity of the array in Figure 6.2 is computed from equation 5.4 as follows:

$$C_{\text{array in Fig. 1}} = \sum_{i=1, j=1}^{1, 2} NE_{ij} = NE_{11} + NE_{12} = 2 + 2 = 4$$

Based on the fact that a reconfigurable antenna can have different equivalent configurations at the same frequency of operation, a frequency dependent complexity is defined in [40] for a single element reconfigurable antenna as shown in equation 5.5.

$$C_{\text{single element}}(f) = \text{Max}_{i=1, N_c(f)} (NE_i(f)) \quad (26)$$

For a reconfigurable antenna array the frequency dependent complexity is expanded from equation 5.5 and represented in equation 5.6.

$$C_{\text{reconfigurable array}}(f) = \bigwedge_{i=1, j=1}^{M, N} \text{Max}_{K=1, N_c(ij)(f)} (NE_K(f)) \quad (27)$$

where

$C(f)$  represents the array's complexity at a frequency f

$N_c(f)$  represents the number of equivalent configurations at a frequency f

$NE_{ij}(f)$  represents the number of edges at the configuration i,j for a frequency f

### 5.2.2 Example 1

$$C(2.205 \text{ GHz}) = \bigwedge_{i=1, j=1}^{1, 2} \text{Max}_{K=1, 4} (NE_K(2.205)) = 2 + 2 = 4 \quad (28)$$

One way of calculating reliability is to relate it to the number of alternative configurations that the antenna has at a certain frequency and the probability of achieving all of these configurations. This correlation is also inversely proportional to the number of edges required to achieve these configurations. In other words, the reliability is higher if the number of configurations is higher, but it gets reduced by the number of edges or connections employed to achieve these configurations. The optimum solution is to have as many alternative configurations as possible with the smallest possible number of connections. Equation 5.7 shows this relationship for a single element reconfigurable antenna.

$$R_{\text{single element}}(f) = \frac{\prod_{i=1}^{N_c(f)} \prod_{j=1}^{NE_i(f)} P(E_{ij})}{\sum_{i,j} 1} \quad (29)$$

where:

R(f)=The reconfigurable antenna reliability at a given frequency f

N<sub>c</sub>(f)=The number of configurations achieving the frequency f

NE(f)=The number of edges for different configurations at the frequency f

P(E)= Probability of achieving the edge E

Extending this relationship to a reconfigurable antenna array requires taking into consideration the M\*N elements as shown in equation 5.8.

$$R_{\text{reconfigurable array}}(f) = \prod_{i,j}^{M,N} \frac{\prod_{k=1}^{N_{cij}(f)} \prod_{L=1}^{NE_k(f)} P(E_{KL})}{\sum_{k=1}^{N_{cij}(f)} \sum_{L=1}^{NE_k(f)} 1} \quad (30)$$

### 5.2.3 Example 2

The antenna array presented in Figure 115 is assumed to be designed with the p-i-n diodes described in [49]. These p-i-n diodes have an RF switching failure rate of  $1.5 \times 10^{-5}$  (1/h). The probability of failure is then considered to be 0.0015. The probability of achieving an edge in the graph model of Figure 115 is 0.9985 for all switches. The reliability of the array at 2.205 GHz can now be calculated according to Table 8 and equation 5.8 as follows:

$$R(2.205) = \sum_{i,j}^{1,2} \frac{\sum_{k=1}^{N_g(f)} \sum_{L=1}^{NE_k(f)} P(E_{KL})}{\sum_{k=1}^{N_g(f)} NE_k(2.205)} \times 100 =$$

$$\frac{\sum_{K=1}^{N_{\alpha}(1.8)} \sum_{L=1}^{NE_K} P(E_{KL}) + \sum_{K=1}^{N_{\alpha}(1.8)} \sum_{L=1}^{NE_K} P(E_{KL})}{\sum_{K=1}^{N_{\alpha}(1.8)} NE_K(2.205) + \sum_{K=1}^{N_{\alpha}(1.8)} NE_K(2.205)} =$$

$$\frac{2 \times 0.9985 + 1 \times 0.9985 + 1 \times 0.9985 + 2 \times 0.9985 + 1 \times 0.9985}{(2+1+1+2+1)} \times 100$$

$$= \frac{7 * 0.9985}{(4+1+1+1)}$$

$$= 99.85\%$$

### 5.2.4 Example 3

We now incorporate the RF MEMS presented in [50] on the antenna array. The probability of switching failure is 0.0357 for the switches in the first element of the array and 0.026 for the switches in the second element [50]. Using these numbers we can determine the probability of switching success in the first element as 0.9643 and in the second element to be 0.974 which

leads to the reliability at 2.205 GHz according to equation 5.8 as follows:

$$\begin{aligned}
 R(2.205) &= \frac{\prod_{i,j} N_{C_{ij}}(f) \prod_{k=1}^{N_{C_{ij}}(f)} NE_k(f)}{\prod_{i,j} N_{C_{ij}}(f) \prod_{k=1}^{N_{C_{ij}}(f)} NE_k(2.205)} \xi 100 = \\
 &= \frac{\prod_{K=1}^{N_{C_{11}}(1.8)} NE_K(2.205) + \prod_{K=1}^{N_{C_{12}}(1.8)} NE_K(2.205)}{\prod_{K=1}^{N_{C_{11}}(1.8)} NE_K(2.205) + \prod_{K=1}^{N_{C_{12}}(1.8)} NE_K(2.205)} P(E_{KL}) + \frac{\prod_{K=1}^{N_{C_{11}}(1.8)} NE_K(2.205) + \prod_{K=1}^{N_{C_{12}}(1.8)} NE_K(2.205)}{\prod_{K=1}^{N_{C_{11}}(1.8)} NE_K(2.205) + \prod_{K=1}^{N_{C_{12}}(1.8)} NE_K(2.205)} P(E_{KL}) \\
 &= \frac{2 \xi 0.9643 + 1 \xi 0.9643 + 1 \xi 0.974 + 2 \xi 0.974 + 1 \xi 0.974}{(2+1+1+2+1)} \xi 100 \\
 &= \frac{3*0.9643 + 4*0.974}{(4+1+1+1)} \\
 &= 96.98\%
 \end{aligned} \tag{32}$$

The **reliability** of an antenna array at a frequency  $f$  can be expressed in terms of its complexity as shown in equation 5.9. We can deduce that the reliability is inversely proportional to the complexity of a reconfigurable antenna array at the same frequency  $f$ .

$$\begin{aligned}
 R_{\text{reconfigurable array}}(f) &= \frac{\prod_{i,j} N_{C_{ij}}(f) \prod_{k=1}^{N_{C_{ij}}(f)} NE_k(f)}{\prod_{i,j} N_{C_{ij}}(f) \prod_{k=1}^{N_{C_{ij}}(f)} NE_k(f)} \xi 100 \\
 &= \frac{\prod_{i,j} N_{C_{ij}}(f) \prod_{k=1}^{N_{C_{ij}}(f)} NE_k(f)}{\prod_{i,j} N_{C_{ij}}(f) \prod_{k=1}^{N_{C_{ij}}(f)} NE_k(f)} \xi 100
 \end{aligned} \tag{33}$$

$C(f)$  is calculated in equation 5.6

$N'_C(f)$  is the number of equivalent configurations at a frequency  $f$  without the configurations with maximum edges.

### **5.3 Prioritization of Frequency Dependent Configurations and Antenna Performance**

In any reconfigurable antenna array, equivalent configurations exist for the same frequency. The failure of any switch, deeply affects the complexity and reliability of that particular array. However the designer is required to prioritize the equivalent frequency configurations based on minimizing the use of switches to improve the overall reliability. For example in Table 8, three

different configurations are equivalent at 2.205 GHz. The primary configuration that achieves this frequency should be the one with highest reliability.

Previously, the complexity and reliability of the entire antenna array was discussed at a certain frequency of operation. Several parameters have to be taken into consideration for all the different equivalent configurations that the antenna array supports. One way of prioritizing the array configurations is by calculating the complexity and reliability of each particular configuration independently from the entire array. The *complexity* of each particular configuration is defined based on equation 5.6 as :

$$C_{Config.}(f) = NE(f) \quad (34)$$

where NE is the total number of edges in a particular configuration.

The configuration reliability can be formulated based on equation 5.8 and is shown in equation 5.11 as:

$$R_{Config.}(f) = \frac{\prod_{i=1}^{NE} P(E_i)}{\sum_{NE} 100} \quad (35)$$

where NE is the total number of edges in a particular configuration and P(E) is the probability of switching success which is equivalent to the probability of achieving an edge.

Based on equation 5.10, equation 5.11 can be rewritten as:

$$R_{Config.}(f) = \frac{\prod_{i=1}^{NE} P(E_i)}{C_{Config.} \sum 100} \quad (36)$$

where  $C_{Config.}$  is defined in equation 5.11

For example, the operation of the reconfigurable antenna array at 2.205 GHz according to Table 8 has three equivalent configurations. To calculate the reliability of the three configurations we consider the array presented in the previous Example 3 and using RF MEMS from [50]. The probabilities of switching success are distributed as 0.9643 for switches in element 1 and 0.974 for switches in element 2. The reliability of each configuration according to equation 5.12 becomes:

### 5.3.1 Example 4

$$R_{G1}(2.205) = \frac{\prod_{i=1}^{NE} P(E_i)}{C_{Config.} \sum} = \frac{2 \times 0.974}{\sum 100} \quad (37)$$

=96.75%

0.9643

3



$$R_{G_2}(2.205) = \frac{\prod_{i=1}^{NE} P(E_i)}{\xi} = \frac{0.9643^2 \cdot 0.974}{\xi} = \frac{0.974}{\xi} = 97.07\% \quad (38)$$

$$R_{G_3}(2.205) = \frac{\prod_{i=1}^{NE} P(E_i)}{\xi} = \frac{0.974}{\xi} = 97.4\%$$

G<sub>3</sub> is designated as the primary configuration with the highest priority since it has the highest reliability. In the same manner, G<sub>2</sub> and G<sub>1</sub> follow respectively. The configurations represented by G<sub>1</sub> and G<sub>2</sub> are used whenever the configuration represented by G<sub>3</sub> fails. For example in the case where switch S<sub>3</sub> fails, configurations G<sub>3</sub> and G<sub>2</sub> cannot be achieved and thus G<sub>1</sub> has to be used with a reliability of 96.75 %. This insures the continuous operation of the array at 2.205 GHz.

The primary configuration in this example is also the one with the lowest complexity. Lower complexity means better reliability only when all the probabilities of switching success are equal. However in an array where the switching success varies between switches, lower complexity won't necessarily mean a better reliability.

For example if the probability of switching success for S<sub>3</sub> was 0.91 while the other probabilities are the same as in Example 3 (0.9643 for {S<sub>1</sub>,S<sub>2</sub>} and 0.974 for {S<sub>4</sub>}), then the reliability for G<sub>1</sub> remains the same whereas the reliabilities for G<sub>2</sub> and G<sub>3</sub> change as:

$$R_{G_1}(2.205) = \frac{\prod_{i=1}^{NE} P(E_i)}{\xi} = \frac{0.9643^2 \cdot 0.974}{\xi} = 96.75\% \quad (39)$$

$$\begin{aligned}
R_{G2}(2.205) &= \frac{\sum_{i=1}^{NE} P(E_i)}{C_{config2}} \times 100 = \\
&= \frac{1 \times 0.9643 + 1 \times 0.974 + 1 \times 0.91}{3} \times 100 = \\
&= 94.94\%
\end{aligned} \tag{40}$$

$$\begin{aligned}
R_{G3}(2.205) &= \frac{\bigwedge_{i=1}^{NE} P(E_i)}{C_{config3}} \times 100 = \frac{0.91}{1} \times 100 \\
&= 91\%
\end{aligned} \tag{41}$$

In this case the primary configuration has to be G1 since it has the highest reliability and probability of success. Even though it does not exhibit the lowest complexity  $C_{G1} = 3 > C_{G1} = 1$

#### 5.4 Practical and Design Aspects

Standard operating conditions for common commercialized switches are predefined by the switch manufacturing companies, where temperature, humidity and other environmental factors are well known, and taken into consideration. Companies clearly specify that they will provide the probability of failure of any of their switching products under unusual conditions to any antenna designer based on specific information and requests from the designer.

An antenna designer needs to follow five steps to insure better system reliability with less complexity while maintaining versatility and adaptability of the reconfigurable antenna array system. These steps are based on the fact that in unknown and harsh environments such as in space, standard conditions for switch failures don't apply and sudden changes in the surrounding conditions are common. The proposed steps are discussed below:

**Step1:** identify the environment where the antenna array will be installed and predict possible surrounding abnormalities

**Step2:** Study and customize the switching components with the manufacturing company

**Step3:** Start the antenna design process while maximizing equivalent configurations, graph modeling each configuration, minimizing the number of connections and thus minimizing the number of switches required.

**Step4:** Optimize the design by prioritizing the configurations as described in section 6.6

**Step5:** Finalize the design to satisfy all the required constraints

## 6. Facilitating the exchange of information between interested organizations on the use of configurable devices and presenting results of the above research and development at technical conferences and meetings.

This area primarily focuses on the requirement to ensure success for programs using microcontrollers and FPGAs, as well as the need to build the reconfigurable systems business base in New Mexico. All COSMIAC partners participate fully in this regard, although UNM also carries the management responsibility for the overall consortium. To that end, COSMIAC created the necessary infrastructure through this grant and companion activities to allow a community to grow. These overarching goals included:

- **Shared Organization and Management:** COSMIAC provided key communications and shared resources for the collaboration. This includes enhancement of the Website at: <http://cosmiac.org>, where information and shared code (software and cores) are made available to the group at large. It also includes organizing meetings and planning activities within the consortium. Meetings and presentations were facilitated to allow cross pollination of ideas and capabilities.
- **Consulting services:** COSMIAC consulted on programs to ensure their successful use of FPGAs and microcontroller based systems through the combination of its own staff as well as efforts that it coordinated through the member organizations such as the Air Force Research Laboratory.
- **Professor Workshops:** COSMIAC provided many workshops each year for community college and university professors across the U.S. These courses are developed in conjunction with the Xilinx Corporation, the largest maker of FPGAs and a COSMIAC partner. The courses provided the basics needed to ensure faculty had the education and training to understand what was required to infuse this type of technology into their programs. The courses provide a standardized basis for FPGA, microcontroller and Hardware Descriptive Language (HDL) education throughout the nation and enhance the likelihood that students will be aware of the advantages of using the parts, as well as the likelihood that they will have a university program that properly prepares them to contribute successfully to advanced projects and subsequent research.
- **Industrial Workshops:** COSMIAC provided many workshops over the period of performance for engineers and scientists in local New Mexico Industry. These maximized exposure of reconfigurable technologies, as well as introduce the COSMIAC consortium as a potential partner in solving industrial problems. These workshops cover subjects ranging from basic introduction and use of parts to advanced topics such as radiation hardened design, power management and partial reconfiguration. In addition to reconfigurable solutions, workshops related to an open source bus architecture called the AFRL Space Plug-and-play Architecture (SPA) were also routinely provided.
- **K-12 Education and Outreach:** A major focus at the University of New Mexico is the support of AFRL in a conscious series of activities to increase the pipeline of quality

students and later, researchers entering the Science, Technology, Engineering and Mathematics (STEM) fields. COSMIAC supports many activities, to include classroom instruction at the Career Enrichment Center (CEC). CEC is an Albuquerque High School devoted to new technology. COSMIAC helped them to begin teaching FPGAs. COSMIAC also sponsored a two-week high-school summer camp (“Digital Academy”) at UNM. This Academy allowed UNM to recruit extremely bright young students. Many of which later went on to become AFRL Phillips Scholars. This opportunity allowed COSMIAC to be able to align itself with outreach activities within AFRL and the National Laboratories to seize opportunities such as summer internships. Our goal was to take training, information, and the excitement of working with digital systems to schools throughout New Mexico to expand the pool of available students who will later be our expert designers.

- **Development of Shared Infrastructure:** COSMIAC located and prepared facilities with adequate laboratory capabilities and easy access to all members, such that all COSMIAC members may easily work together (in a collaborative environment) on a variety of projects. These areas were also available for those needing COSMIAC’s technical services to come and share with any member. This included four facility efforts:
  - o Enhanced laboratories within the UNM ECE Department. COSMIAC was responsible for the creation of four new laboratories at the UNM ECE Department. The first was the ECE238L Introduction to Digital Logic Laboratory. This lab is used by approximately 100 engineering undergraduates each semester. It is designed to teach basic digital concepts (synchronous and asynchronous) using FPGA technology. The second is the ECE344L Microprocessor Laboratory. This lab is used to teach microprocessors using the advanced Virtex-4 device. The third laboratory is the Senior Design facility. It is used by undergraduate students during their completion of their capstone project. The final laboratory is the Advanced Microsystems Laboratory. This lab is used by graduate students for the design and development of advanced platforms for performing graduate level studies.
  - o Preparation of an off-campus facility with convenient access for all COSMIAC consortium members. This 11,000 square foot facility has a complete range of FPGA and microcontroller based technologies to include a wide range of test equipment for performing analysis on the available prototyping platforms as well as on the platforms developed at COSMIAC.
  - o Assistance to community colleges and universities across the state of NM and across the country. This included creation of the FPGA learning facility at the New Mexico State University (Las Cruces, NM) as well as upgrading the FPGA learning capability and laboratory at New Mexico Institute of Mining and Technology (Socorro, NM) and Highlands University (Las Vegas, NM). Further assistance was provided to community colleges to include Dona Ana Community College (Las Cruces, NM), San Juan Community College (Farmington, NM), Southwest Polytechnic Institute (Albuquerque, NM), Central New Mexico Community College (Albuquerque, NM) and Luna Community College (Las Vegas, NM).

- **Mobile Training Equipment:** Instrumental to all of these activities was the mobile teaching laboratory, which consisted of a set of laptop computers and hardware platforms that can be reconfigured at will to work with several FPGA platforms. These include:
  - o An inexpensive FPGA-based board (Spartan-III) for simple logic instruction and introduction to HDLs.
  - o A more elaborate Virtex-4 Pro board by the Digilent Corporation that can be used to instruct nearly all possible FPGA programming concepts. This includes an advanced FPGA with two embedded PowerPCs (PPCs) that allow students to design systems containing both embedded software and digital logic. They can also be used to provide instruction for concepts such as Single Event reliability and triple modular redundancy.
  - o A state-of-the-art Virtex-V board (Xilinx ML-501) to explore concepts requiring the most advanced and newest devices.

The following is a summary of some of the activities performed under this task during the period of performance:

- COSMIAC personnel coordinated with NASA Goddard Space Flight Center (GSFC) to jointly host their Military/Aerospace Programmable Logic Devices (MAPLD) conference in collaboration with COSMIAC's Reconfigurable Space (ReSpace) conference in 2010 and 2011. This joint conference was named ReSpace/MAPLD. This joint activity was designed to address relevant experiences of FPGAs in space. COSMIAC took the main organizing role to include program management; local events coordination and scheduling of presentations. Conference Schedules for 2010 and 2011 are included in Appendix B.
- The Center has established an online repository for quick access information related to radiation testing of commercial electronics. This site also has resources to other materials and radiation data from other organizations such as NASA. Much of what has been accomplished by COSMIAC is related to the use of Microsystems in space. To accomplish this, the Center is highly involved with testing and publishing the results of this type of activity. Often, COSMIAC personnel will create development platforms that are then taken into the on base Cobalt or Cesium sources for further testing and documentation of the results.
- COSMIAC worked for two years with AFRL/RVSE to create a quarterly newsletter related to SPA, CubeFlow and small satellites. This newsletter was handed out at a variety of different conferences and workshops. SPA was a major part of many of our activities over the five years. The only way the nanosatellite community could succeed is through the use of new systems and technologies. SPA presented just such a technology breakthrough. As such, SPA presented a way to develop a PnP methodology for CubeSats.
- NASA Glenn funded COSMIAC to travel to Ohio to teach a two-day SPA course. This

course was attended by engineers from NASA Glenn and the Jet Propulsion Laboratory. Over the five years, COSMIAC became known as the “go to” place for all things SPA. Activities such as this SPA course at NASA were routine as they provided a mechanism for exposing other federal organizations to this system. COSMAIC provided three short courses on SPA to the Navy Research Laboratory and several other organizations. To date, it is estimated that COSMIAC and their partners have trained over 700 individuals on SPA and its design methodology.

- B. Zufelt traveled to Space Dynamics Laboratory in 2012 to attend the SPA working group meeting related to the upgrade of the Satellite Data Model (SDM) to the SPA Services Manager (SSM). This allowed COSMIAC to stay aware of the changes and direction SPA standards are moving. COSMIAC is beginning to play a more active role in the area of SPA-1 standards and we have been asked to present our SPA-1 work at the next SPA working group meeting.
- COSMIAC hosted a four-day Advanced Plug and Play Technology (APT) meeting. The meetings resulted in the completion of the first draft of the AIAA SPA standards. These standards were submitted for review through the AIAA standards system and have now been formally accepted. AFRL personnel felt that only through the creation of formalized standards would SPA ever be truly accepted by the large satellite and space industry for utilization into their systems.
- Course material and guidelines for Aerospace design were developed. Materials developed by Dr. Heather Quinn of Los Alamos National Lab on FPGA Design for Aerospace are now being used to train graduate students and she routinely offers an undergraduate/graduate level course on this type of design methodology. The materials cover subjects ranging from reliability requirements and challenges of the space environment to mitigation techniques to assure designs. Students must demonstrate their knowledge through hands-on lab projects.
- COSMIAC (in conjunction with California Polytechnic Institute) developed a one-hour educational presentation on the Global Educational Network for Satellite Operations (GENSO) system for distributed ground station support for satellites that can be delivered over the Internet via skype or Ooooo. This allowed COSMIAC to inform others quickly about the features and capabilities of GENSO. We have already been requested by the AFRL UNP Manager to provide this briefing to other schools. In 2011, this presentation was delivered to AFRL UNP, NASA, Boston University, Kentucky University, Michigan Technological University and Stanford. In addition, at the CubeSat workshop in April 2012, COSMIAC and Cal Poly performed a joint presentation. COSMIAC is also serving a key role as the lead academic liaison for GENSO in the US for the European Space Agency. The Center continues to host weekly developers meetings on the package. As part of the process, B. Zufelt was asked to attend a Plug and Play Workshop at NASA Ames. GENSO, in conjunction with SPA provided a powerful capability to be able to rapidly create and deploy space assets.

## 7. Conclusions

This report described the research and educational activities in the area of reconfigurable systems in a variety of application areas. Hardware and software development to support the latest FPGA devices for creating the concepts of plug-and-play that can be applied to satellite design were presented and discussed. The plug-and-play architecture was developed to address a modern “modular open network architecture “ that is akin to the USB and Ethernet standards found in computers.

A new approach, called “scrubbing”, that relies on reloading the configuration memory frames in an FPGA, at defined time intervals, was analyzed and several results presented. It was shown that the proposed approach is possible and can easily be implemented in the case of FPGA devices that support dynamic Partial Reconfiguration (DPR).

The development of adaptive wiring manifold (AWM), a wiring harness that is reconfigurable and scalable for general applications was shown that it can be used in space applications. In principle, it is like an FPGA that can be programmed to implement wiring patterns, except that the proposed adaptive harness allows for the routing of continuously variable analog, power, and microwave signals.

Various devices developed for reconfigurable space applications and some of the new techniques such as additive manufacturing, neural networks embedded on FPGAs, to rapidly design these reconfigurable electronics and control them, were included in this report.

One of the aims of the agreement was to develop not only new reconfigurable system designs but to also establish guidelines for the design and optimization of these types of systems. Some new guidelines in the area of reconfigurable antennas that can be extended to other reconfigurable space electronics and devices were introduced. In addition, the correlation between reliability and complexity of reconfigurable antennas mathematically was demonstrated. Information theory was used to predict the probability of error in such systems.

Finally, COSMIAC facilitated the exchange of information between interested organizations on the use of configurable devices by organizing conferences, several workshops on FPGA design, and by introducing courses in the area of satellite design and development.

## 8 References

- 1) J. Lyke, "U.S. Air Force's Plug-and-Play Satellites", IEEE Spectrum, July 2012.
- 2) J. Scott, J. Lyke, P. McGuirk, M. Shaw, D. Fronterhouse, "Appliqué Sensor Interface Module: An Enabling Technology for Space Plug-and-Play Systems", proceedings of Small Satellite Conference, 2007.
- 3) A. Vera, M. Sibley, S. Ardalán, K. Avery, and J. Lyke, "Appliqué Sensor Interface Module Based on 90nm Rad- Hard Structured Application- Specific Integrated Circuit", Proceedings of the AIAA Infotech@Aerospace 2010 Conference and Exhibit, Atlanta, Georgia, April 2010.
- 4) Space Plug-and-Play Architecture Standard, SpaceWire Subnet Adaptation, AIAA, available at [http://aiaa.kavi.com/public/pub\\_rev/SPA\\_S-133-9-201X\\_PR.pdf](http://aiaa.kavi.com/public/pub_rev/SPA_S-133-9-201X_PR.pdf)
- 5) OpenCores.org project. Available at [http://opencores.org/project,spacewire\\_light](http://opencores.org/project,spacewire_light)
- 6) LEON3, Multiprocessing CPU core by Aeroflex Gaisler, Datasheet available at [http://www.gaisler.com/doc/leon3\\_product\\_sheet.pdf](http://www.gaisler.com/doc/leon3_product_sheet.pdf)
- 7) Star-Dundee SpaceWire Analyzer User manual. Software version 4.01. Available upon request at [www.star-dundee.com](http://www.star-dundee.com)
- 8) G. Estrin et al., "Parallel Processing in a Restructurable Computer System", IEEE Transactions on Electronic Computers, 1963, Issue 5.
- 9) G. A. Vera, "A dynamic arithmetic architecture: precision, power and performance considerations", PhD Dissertation, The University of New Mexico, 2008.
- 10) A. Dehon, "Reconfigurable architectures for general-purpose computing", Ph.D. thesis, Massachusetts Institute of Technology Artificial Intelligence Laboratory, October 1996.
- 11) J. Resano, D. Mozos, F. Catthoor, and D. Verkest, "A reconfiguration manager for dynamically reconfigurable hardware", IEEE Design and Test of Computers, vol. 22, no. 5, pp. 452–460, 2005.
- 12) P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Invited paper: Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs", Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '06), pp. 12–17, August 2006.
- 13) C. Claus, B. Zhang, W. Stechele, L. Braun, M. Hubner, and J. Becker, "A Multi-Platform Controller allowing for maximum dynamic partial reconfiguration throughput", Proceedings of FPL'2008, Heidelberg, Germany, Sept. 2008, pp. 535-538.
- 14) C. Claus, J. Zeppenfeld, F. Muller, and W. Stechele, "Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System", *Design, Automation & Test in Europe Conference & Exhibition*, pp.1-6, April 16-20, 2007
- 15) Partial Reconfiguration Early Access software tools for ISE 9.1 SP2. Design examples targeting an ML403 Development board.
- 16) Xilinx, "Virtex-4 Configuration Guide (UG071)," 2006.
- 17) C. T. Ewe, P. Y.K. Cheung, and G. A. Constantinides, "Dual fixed-point: an efficient alternative to floating-point computation", Proceedings of International Conference on Field Programmable Logic, vol. 3203 of Lecture Notes in Computer Science, pp. 200–208, 2004.
- 18) K. Bondalapati and V.K. Prasanna, "Reconfigurable computing systems", Proceedings of the IEEE, vol. 90, no. 7, pp. 1201–1217, 2002.



- 19) K. Bondalapati and V. K. Prasanna, "Dynamic precision management for loop computations on reconfigurable architectures", Field Programmable Custom Computing Machines, FCCM'99, 1999.
- 20) Fact sheet, available: <http://www.gaisler.com/index.php/products/processors/leon3ft>
- 21) Lyke, J., Wilson, W., and Contino, P., MEMS-based reconfigurable manifold, in NASA MAPLD Conference, 2005.
- 22) J. Lyke, W. Wilson, and R. Broyles, Adaptive manifold, October 2002.
- 23) S. Thompson, and I. Mycroft, "Self-healing reconfigurable manifolds", Proceedings DCC'06: Designing Correct Circuits, 25-26 March 2006.
- 24) S. Shelley, J. Costantine, C. G. Christodoulou, D. E. Anagnostou, and J. C. Lyke, "FPGA Controlled Switch-Reconfigured Antenna", IEEE Antennas and Wireless Propagation Letters, pp. 355 – 358, 2010.
- 25) D. E. Anagnostou, G. Zheng, M. T. Chryssomallis, J. C. Lyke, G. E. Ponchak, J. Papapolymerou and C. G. Christodoulou, "Design, fabrication and measurements of an RF-MEMS-based self-similar reconfigurable antenna", IEEE Transactions on Antennas and Propagation, V.54, I. 2, pt. 1, Feb. 2006, pp. 422-432.
- 26) IEEE Standard Test Access Port and Boundary-Scan Architecture, available at: [http://standards.ieee.org/reading/ieee/std\\_public/description/testtech/1149.11990\\_desc.html](http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.11990_desc.html)
- 27) Spartan-3E FPGA Starter Kit Board User Guide, available at [http://www.xilinx.com/support/documentation/boards\\_and\\_kits/ug230.pdf](http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf)
- 28) K. Chung, Y. Nam, T. Yun, and J. Choi, "Reconfigurable Microstrip Patch Antenna with Switchable Polarization", ETRI Journal, Volume 28, Number 3, June 2006.
- 29) O. Bernander. Neural Network. Microsoft Encarta 2006, Redmond, WA, Microsoft Corporation, 2005.
- 30) S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd edition, Prentice Hall, 1998.
- 31) M. Bonnici, E. J. Gatt, J. Micallef, I. Grech, "Artificial Neural Network Optimization for FPGA", 13th IEEE International Conference on Electronics, Circuits and Systems, pp 1340-1343, Dec 2006.
- 32) M. Renovell, P. Faure, J. M. Portal, J. Figueras, and Y. Zorian, "IS-FPGA: a New symmetric FPGA architecture with implicit scan," Proceedings of Test Conference, pp. 924-931, 2001
- 33) E. Djalal, A. Kheldoun, and L. Refoufi, "Sigmoid function approximation for ANN implementation in FPGA devices", CSECS '10 Proceedings of the 9th WSEAS International conference on Circuits, systems, electronics, control & signal processing, 2010.
- 34) Y. Tawk, J. Costantine, and C. G. Christodoulou, "A Varactor Based Reconfigurable Filtenna", IEEE Antennas and Wireless Propagation Letters, pp. 716-719, 2012
- 35) S. Nikolaou, G. E. Ponchak, J. Papapolymerou, and M. M. Tentzeris, "Conformal double exponentially tapered slot antenna (DE TSA) on LCP for UWB applications," IEEE Transactions on Antennas and Propagation, vol.54, no.6, pp. 1663- 1669, June 2006.
- 36) Y. Tawk, J. Costantine, and C. G. Christodoulou, "A frequency reconfigurable rotatable microstrip antenna design" , Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE, pp. 1-4, 11-17, July 2010

- 37) M. E. Zamudio, J. –H. Kim, Y. Tawk and C. G. Christodoulou, “Integrated Cognitive Radio Antenna Using Reconfigurable Band Pass Filters”, European Conference on Antennas and Propagation 2011, Rome, Italy, pp. 11-15, April 2011.
- 38) J. Costantine, S. al-Saffar, C. G. Christodoulou, K. Y. Kabalan, and A. El-Hajj, “The analysis of a reconfiguring antenna with a rotating feed using graph models”, IEEE Antennas and Wireless Propagation Letters, vol. 8, pp. 943-946, 2009.
- 39) J. Costantine, S. al-Saffar, C. G. Christodoulou, and C. T. Abdallah, “Reducing Redundancies in Reconfigurable Antenna Structures Using Graph Models,” IEEE Transactions on Antennas and Propagation, Vol.59, Issue 3, pp.793-801, 2011
- 40) J. Costantine, Y. Tawk, C. G. Christodoulou, and C. T. Abdallah, “Reducing Complexity and Improving the Reliability of Frequency Reconfigurable Antennas”, Proceedings of the Fourth European Conference on Antennas and Propagation (EuCAP 2010), pp.1-4, 2010
- 41) J. Sarrazin, Y. Mahe, S. Avrillon, and S. Toutain, “Pattern reconfigurable cubic antenna”, IEEE Transactions on Antennas and Propagation, vol. 57, no. 2, pp. 310-317, Feb. 2009.
- 42) J. Perruisseau-Carrier, P. Pardo-Carrera, and P. Miskovsky, “Modeling, design and characterization of a very wideband slot antenna with reconfigurable band rejection”, IEEE Transactions on Antennas and Propagation, vol. 58, no. 7, pp. 2218-2226, July 2010.
- 43) C. R. White and G. M. Rebeiz, “Single and dual-polarized tunable slot-ring antennas”, IEEE Transactions on Antennas and Propagation, vol. 57, no. 1, pp. 19-26, Jan. 2009.
- 44) H. Jiang, M. Patterson, C. Zhang, and G. Subramanyan, “Frequency tunable microstrip patch antenna using ferroelectric thin film varactor”, IEEE National Aerospace & Electronics Conference, pp. 248-250, July 2009.
- 45) C. J. Panagamuwa, A. Chauraya, and J. C. Vardaxoglou, “Frequency and beam reconfigurable antenna using photoconductive switches”, IEEE Transactions on Antennas and Propagation, vol. 54, no. 2, Feb. 2006.
- 46) Y. Tawk, A. R. Albrecht, S. Hemmady, G. Balakrishnan, and C. G. Christodoulou, “Optically pumped frequency reconfigurable antenna design”, IEEE Antennas and Wireless Propagation Letters, vol. 9, pp. 280-283, Mar. 2010.
- 47) D. Piazza, N. J. Kirsch, A. Forenza, R.W. Heath, K.R.. Dandekar, “Design and Evaluation of a Reconfigurable Antenna Array for MIMO Systems”, IEEE Transactions on Antennas and Propagation, Vol.56, Issue 3, pp.869-881, 2008.
- 48) M. T. Ali, T. A. Rahman, M. R. Kamarudin, R. Sauleau, M. N. M. Tan, and M. F. Jamlos “A Reconfigurable Planar Antenna Array (RPAA) With Back Lobe Reduction”, International Workshop and Antenna Technology (IWAT), pp.1-4, 2010
- 49) [http://documentation.renesas.com/doc/products/diode/rej27g0006\\_diode.pdf](http://documentation.renesas.com/doc/products/diode/rej27g0006_diode.pdf)
- 50) J. G. Teti, F. P. Dareff “ MEMS 2- bit Phase Shifter Failure Mode and Reliability Considerations For Large X Band Arrays”, IEEE Transactions on Microwave Theory and Techniques, Vol.52, Issue 2, pp.693-701.

## 9 Key Personnel

Christos Christodoulou is the Director for the Configurable Space Microsystems Innovations and Applications Center (COSMIAC) and a Professor in the Department of Electrical and Computer Engineering at the University of New Mexico. His research interests are in the areas of modeling of electromagnetic systems, FPGA reconfigurable systems, and smart RF/photonics. He is responsible for leading the COSMIAC collaboration between Government, Industry, and Academia to ensure design success and deployment of programmable logic in space, military and civil applications as well as to develop the long-term plans for the Center.

Dr. Christodoulou received his Ph.D. in Electrical Engineering from North Carolina State University in 1985. He served as a faculty member in the University of Central Florida, Orlando, from 1985 to 1998. In 1999, he joined the faculty of UNM's Electrical and Computer Engineering Department, where he served as the Chair of the Department through 2005. Dr. Christodoulou is an IEEE Fellow and a member of Commission B of USNC/URSI, Eta Kappa Nu, and the Electromagnetics Academy.

Craig Kief serves as Deputy Director of COSMIAC. Mr. Kief serves as the lead Program Manager for the Air Force Research Laboratory's Cubeflow training program and is a Research Scholar on the faculty at the University of New Mexico. In this capacity, he is responsible for overseeing curriculum and training development, teaching short courses, and coordinating the scheduling and registration of COSMIAC and NSF courses.

Mr. Kief has over 32 years experience in computer and satellite communications, including voice and data networks, testing, troubleshooting, debugging, system administration, embedded software, software/hardware integration, and network monitoring. Mr. Kief has an extensive background in programmable logic involving FPGA and CPLD technologies. He is currently developing a 1U and 6U CubeSats to perform space weather analysis. Mr. Kief retired from the Air Force in 1998 following 20 years of military service. His final military assignment was at the Air Force Operational Test and Evaluation Center (AFOTEC) at Kirtland Air Force Base.

Mr. Kief has a B.S. and M.S. in Computer Engineering from the University of New Mexico. He has published and taught in the areas of digital and programmable logic, satellite design and in verification and validation of systems. He is also an IEEE senior member.

Steven Suddarth is the Chief Research Officer of COSMIAC. He serves as Research Faculty at the University of New Mexico. A retired Air Force Colonel with 24 years of service, Dr. Suddarth served his last military assignment as the liaison from U.S. Strategic Command to the National Laboratories. Based in Los Alamos, NM, Col. Suddarth initiated and oversaw a development team of 60 people spanning four sites to develop key technologies for the war fighter. Dr. Suddarth has overseen several substantial computer engineering/embedded systems projects. These include the development of a first-ever 3-dimensional mixed analog/digital image processor which advanced the State-of-the-Art by 3 orders of magnitude. Dr. Suddarth also built and tested several airborne optical sensing systems, unmanned aerial robotics systems, and software systems for large military space programs. Dr. Suddarth is a 1982 graduate of the U.S.

Air Force Academy (B.S. in Electrical Engineering), and he holds M.S. and Ph.D. degrees in Electrical Engineering from the University of Washington.

**Alonzo Vera** received M.S. and Ph.D. in Electrical Engineering from the University of New Mexico in 2004 and 2008 respectively. Since then, he has specialized in embedded system design for aerospace applications and dynamic partial reconfiguration applications using FPGAs. He has worked on radiation effects mitigation techniques for FPGA-based systems and custom ASICs as well as radiation effects testing at different facilities around the country. Dr. Vera has numerous publications in conferences and journals. Dr. Vera has also been an invited lecturer on digital design and digital signal processing using FPGAs in numerous countries across Latin America. His current areas of interest are reconfigurable computing, digital signal processing and cognitive radio.

**Brian Zuffelt** is an electrical engineer and graduated from the University of New Mexico. He has written many tutorials on Microcontroller and FPGA design. He has also assisted many community colleges in the creation of FPGA based curriculum. Over the last three years, Zuffelt has been working with the Air Force Research Lab to create Space Plug-and-play Architecture (SPA) components, and provided training on SPA tools and architecture. To date, Zuffelt has trained over 700 engineers in this new approach to spacecraft design using an open source bus architecture. Recent work involves the use of different types of Mini Plug-and-Play network managers that can control various Appliqué Sensor Interface Modules. This approach builds on the current SPA-1 standard while remaining ITAR free. This architecture will be flown on the Trailblazer Cubesat mission that is in current development. Trailblazer will be one of the first satellites to completely run on the new Plug and Play concept. Zuffelt is the lead engineer for the Trailblazer satellite effort. It will fly a dosimeter to test the radiation levels of the South Atlantic Anomaly in addition to testing a new type of 3D Printed Circuit Board – rapid prototyping design

## 10 List of Publications

### A) Journal papers published or submitted in the area of transistors, devices, photocells, and materials:

- 1) K. E. Kambour, C. Kouhestani, D. Nguyen, N. Rosen, and R. A. B. Devine, "Tunneling discharge of positive trapped oxide charge in p-channel field effect transistors", *App. Phys. Letts.*, 99 (8), art. 083506, 2011.
- 2) D. D. Nguyen, C. Kouhestani, K. E. Kambour, H. P. Hjalmarson, and R. A. B. Devine, "Extraction of recoverable and permanent trapped charge resulting from negative bias temperature instability", *Physica Status Solidi [C]*, 2013.
- 3) K. Kambour, N. Rosen, C. Kouhestani, D. Nguyen, C. Mayberry, R. A. B. Devine, A. Kumar, C-C. Chen, G. Li and Y. Yang, "Modeling of the X-irradiation Response of the Carrier Relaxation Time in P3HT:PCBM Organic-Based Photocells", *IEEE Transactions on Nuc. Sci*, 2012.
- 4) C. Mayberry, D. D. Nguyen, C. Kouhestani, K. E. Kambour, H. P. Hjalmarson and R. A. B. Devine, "Measurement and Identification of Three Contributing Charge Terms in Negative Bias Temperature Instability", *ECS Transactions* 2013.
- 5) K. E. Kambour, D. D. Nguyen, C. Kouhestani, and R. A. B. Devine "Measurement and Modeling of Duty-Cycle Effects Due to NBTI", *IEEE IIRW Final Report*, 2013.
- 6) D. D. Nguyen, C. Kouhestani, K.E.Kambour, and R.A.B Devine, "Insight into the multicomponent nature of negative bias temperature instability", *JVST B*, 2013.
- 7) K. E. Kambour, D. D. Nguyen, C. Kouhestani, and R. A. B. Devine, "Comparison of NBTI and irradiation induced interface states", *IEEE IIRW 2013* [in publication]
- 8) D. D. Nguyen, C. Kouhestani, K. E. Kambour, and R. A. B. Devine, "On the nature of "permanent" degradation of NBTI", *IEEE IIRW 2013* [in publication]
- 9) D. D. Nguyen, K. E. Kambour, C. Kouhestani, H. P. Hjalmarson, and R. A. B. Devine, "High temperature annealing of the interface state component of negative bias temperature instability (NBTI) in MOSFET devices", *ECS Transactions* 2013 [in publication]
- 10) H. P. Hjalmarson, D. D. Nguyen, K. E. Kambour, C. Kouhestani, and R. A. B. Devine, "Similarity between ionizing radiation effects and negative bias temperature instability (NBTI) MOSFET devices", *ECS Transactions* 2013 [in publication]
- 11) D. D. Nguyen, C. Kouhestani, K. E. Kambour, and R.A.B Devine, "Direct evidence for interface state annealing in negative bias temperature instability response", *JVST B*, 2013 [in publication]
- 12) K. Kambour, C. Kouhestani, D. Nguyen, and R. A. B. Devine "The effect of radiation induced charging on gate all around NMOS devices," *IEEE Trans. on Nuc. Sci*, 2013 [in publication]

### B) Journal papers published or submitted in the area of reconfigurable antennas, FPGA control, graphs, complexity and reliability :

- 13) J. Costantine, C. G. Christodoulou, C. T. Abdallah, and S. E. Barbin, "Optimization and Complexity Reduction of Switch-Reconfigured Antennas using Graph Models",

- IEEE Antennas and Wireless Propagation Letters, vol. 8, pp. 1072 – 1075, 2009
- 14) Y. Tawk and C. G. Christodoulou, “A New Reconfigurable Antenna Design for Cognitive Radio”, IEEE Antennas and Wireless Propagation Letters, vol. 8., pp. 1378-1381, 2009
  - 15) S. Shelley, J. Costantine, C. G. Christodoulou, D. E. Anagnostou, and J. C. Lyke. “FPGA Controlled Switch-Reconfigurable Antenna”, IEEE Antennas and Wireless Propagation Letters, pp. 355 – 358, 2010
  - 16) J. Costantine, Sinan al-Saffar, C. G. Christodoulou, and C. T. Abdallah, “Reducing Redundancies in Reconfigurable Antenna Structures Using Graph Models”, IEEE Transactions on Antennas and Propagation, pp. 793-801, March 2011
  - 17) J. Costantine, C. G. Christodoulou, J. C. Lyke, F. De Flaviis, A. Grau, and S E. Barbin, “Analyzing the Complexity and Reliability of Switch-Frequency-Reconfigurable Antennas Using Graph Models”, IEEE Transactions on Antennas and Propagation, vol. 60, pp. 811 – 820, Feb. 2012
  - 18) J. Costantine, Y. Tawk, and C. G. Christodoulou , “CubeSat Deployable Antenna Using Bi-Stable Composite Tape Springs”, IEEE Antennas and Wireless Propagation Letters, pp. 285-288, 2012
  - 19) Y. Tawk, J. Costantine, and C. G. Christodoulou, “A Varactor Based Reconfigurable Filtenna”, IEEE Antennas and Wireless Propagation Letters, pp. 716-719, 2012
  - 20) J. Costantine, Y. Tawk, and C. G. Christodoulou, “Complexity versus Reliability in Arrays of Reconfigurable Antennas”, IEEE Transactions on Antennas and Propagation, pp. 5436-5441, Nov. 2012

## Appendix A - Basic Rules for Using Graphs Models with Reconfigurable Antennas

### The Definition of a Graph

A graph can be defined as the collection of *vertices* that may be connected together with lines called *edges*. A graph  $G=(V(G),E(G))$  consists of two finite sets:

- $V(G)$ , the vertex set of the graph, often denoted by just  $V$ , which is a nonempty set of elements called Vertices.
- $E(G)$ , the edge set of the graph, often denoted by just  $E$ , which is a possibly empty set called Edges. Each edge  $e$  in  $E$  is assigned an unordered pair of vertices  $(u, v)$ , called the end vertices of  $e$ .

Vertices are also sometimes called points, nodes or just dots. If  $e$  is an edge with end vertices  $u$  and  $v$  then  $e$  is said to join  $u$  and  $v$ . It is important to note that the definition of a graph allows the possibility of the edge  $e$  having identical end vertices, i.e. it is possible to have a vertex  $u$  joined to itself by an edge; such an edge is called a loop.

### The Properties of a Graph

A graph can be either *directed or undirected*. A directed graph (often called digraph) is a finite vertex set  $V$  together with a non reflective relation  $R$  on  $V$ . The edges in a directed graph have a certain determined direction while this is not the case in an undirected graph.

If two or more edges of a graph  $G$  have the same end vertices then these edges are called parallel. A vertex of  $G$  which is not the end of any edge is called isolated. Two vertices joined by an edge are said to be adjacent or neighbors. The set of all neighbors of a fixed vertex  $v$  of  $G$  is called the neighborhood set of  $v$  and is denoted by  $N(v)$ .

*Vertices may represent physical entities and edges between them in the graph represent the presence of a function* resulting from connecting these entities. If one is proposing a set of guidelines for a reconfigurable antenna design, then a possible modeling rule may be to create an edge between two vertices whenever their physical connection results in a meaningful antenna function.

*Edges may have weights* associated with them to represent costs or benefits that are to be minimized or maximized. For example, if a capacitor is connecting two end points of a system and these end points are represented by two vertices in a graph, then the edge connecting these two vertices has a weight equal to the capacitance of that capacitor. Figure 1 shows an example of an undirected as well as a weighted directed graph.

An empty graph is a graph with no edges. An edge  $e$  of a graph  $G$  is said to be incident with the vertex  $v$  if  $v$  is an end vertex of  $e$ . In this case we also say that  $v$  is incident with  $e$ . Two edges  $e$  and  $f$  which are incident with a common vertex  $v$  are said to be adjacent. The degree  $d$  of a vertex  $v$ ,  $d(v)$ , is the number of edges of  $G$  incident with  $v$ , counting each loop twice, i.e. it is the number of times  $v$  is an end vertex of an edge. A vertex is called odd or even depending on whether its degree is odd or even.

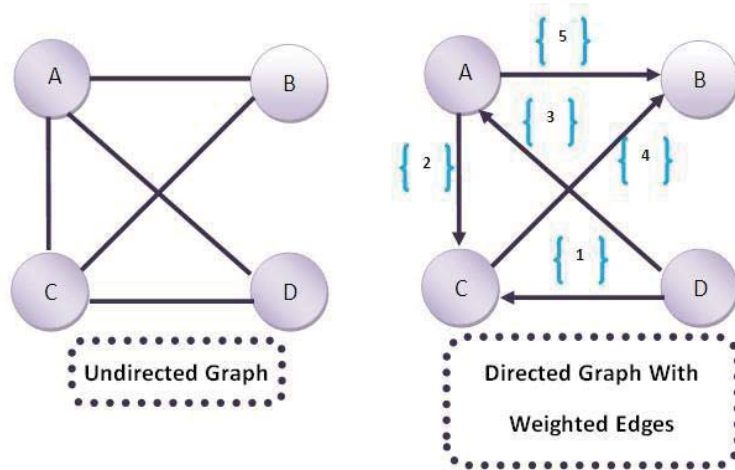


Figure 1 An example of an undirected as well as directed graph with weighted edges

**The Adjacency Matrix Representation of a Graph**

The adjacency-matrix representation of a graph  $G$ , assuming that the vertices are numbered  $1, 2, \dots, |V|$  in some arbitrary manner, consists of a  $|V| \times |V|$  matrix  $A = (a_{ij})$  such that:

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

Example

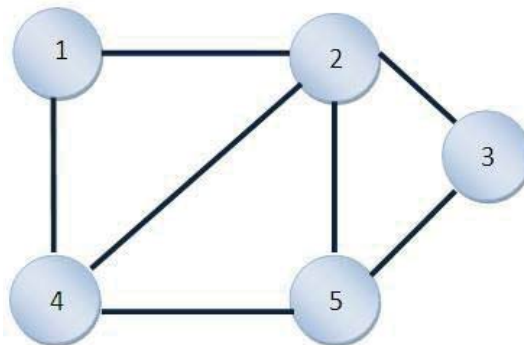


Figure 2 A graph representation with 6 vertices

The adjacency matrix of the graph shown in Figure 2 is presented by the matrix  $A$  below:



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (2)$$

The adjacency-matrix representation can also be used for weighted graphs. *The corresponding weights in a graph are grouped into the adjacency matrix.* For example, if  $G = (V, E)$  is a weighted graph with edge-weight function  $w(u, v)$  of the edge  $(u, v)$ .  $W$  is simply stored as the entry in row  $u$  and column  $v$  of the adjacency matrix. If an edge does not exist, a NIL value can be stored as its corresponding matrix entry, though for many problems it is convenient to use a value such as 0 or  $\infty$ . Moreover, if the graph is un-weighted, there is an additional advantage in storage for the adjacency-matrix representation. Rather than using one word of computer memory for each matrix entry, the adjacency matrix uses only one bit per entry.

### **Paths and Cycles of a Graph**

A walk in a graph  $G$  is a finite sequence :

$$W = v_0 e_1 v_1 e_2 v_2 \dots v_{k-1} e_k v_k \quad (3)$$

Whose terms are alternatively vertices and edges such that, for  $1 \leq i \leq k$ , the edges  $e_i$  has ends  $v_{i-1}$  and  $v_i$ . Thus each edge  $e_i$  is immediately preceded and succeeded by the two vertices with which it is incident. The vertex  $v_0$  in the above walk is called the origin of the walk  $W$ , while  $v_k$  is called the terminus of  $W$ . Note that  $v_0$  and  $v_k$  need not be distinct. A trivial walk is one containing no edges.

If the vertices  $v_0, v_1, \dots, v_k$  of the walk  $W$  are distinct then  $W$  is called a

*Path.* Any two paths with the same number of vertices are isomorphic [1]. *The weight of a path is defined as the sum of the weights of its constituent edges.* In some cases it is useful to find the *shortest path* connecting two vertices. This notion is used in graph algorithms in order to optimize a certain function. The shortest path distance in a non-weighted graph is defined as the minimum number of edges in any path from vertex  $s$  to vertex  $v$ . Otherwise, if the graph is weighted then the shortest path corresponds to the least sum of weights in a particular path.

### **Rules and Guidelines for Graph Modeling Reconfigurable Antennas**

Here, some rules to graph model reconfigurable antennas are set. These *rules are not unique*; however they are required any reconfigurable antenna design steps. A designer following the proposed graph modeling rules should end up with an optimal reconfigurable antenna.

The graph modeling of a certain antenna is governed by its structure and the reconfiguration techniques used in that particular structure. *Each rule is valid for a certain reconfigurable*

*antenna group.* The main six groups to be analyzed here are:

- *Group 1:* Antennas using switches
- *Group 2:* Antennas using capacitors or varactors
- *Group 3:* Antennas using physical angular alteration
- *Group 4:* Antennas using different biasing networks
- *Group 5:* Antenna arrays
- *Group 6:* Antennas using reconfigurable feeding networks [2]

Here, some constraints are set for each rule to facilitate the graph modeling process.

***Rule 1.a: The graph modeling of a multi-part antenna whose parts are connected by switches is undirected with weighted edges connecting the vertices that represent the different parts.***

Valid for:

This rule is valid for group 1 with multiple metallic parts.

Constraints:

The connection between each two parts has a distinctive angular direction. The designer defines a reference axis that represents the direction that the majority of parts have with each other or with a main part. The connections between the parts are represented by the edges. The edge weights represent the angles that the connections make with the reference axis. A weight  $W=1$  is assigned to an edge representing a connection that has an angle  $0^\circ$  or  $180^\circ$  with the reference axis, otherwise a weight  $W=2$  is assigned to the edge as shown in equation 4.

$$W_{ij} = P_{ij} \tag{4}$$

$$\text{Where } P_{ij} = \begin{cases} 1 & A_{ij} = 0^\circ \text{ or } 180^\circ \\ 2 & \text{Otherwise} \end{cases}$$

Where  $A_{ij}$  represents the angle that the connection  $i,j$  form with the reference axis.

Example

As an example the antenna shown in Figure 3 is modeled using a graph following rule 1.a. The basic antenna is a  $130^\circ$  balanced bowtie. A portion of the antenna corresponds to a two iteration fractal Sierpinski dipole. The remaining elements are added (three on each side) to make the antenna a genera; reconfigurable structure.

Following rule 1.a, the different adjacent triangular parts of the antenna (triangles added) are interconnected by MEMS switches as shown in Figure 3 [3-6]. The vertices in the graph model represent the triangles added. The edges connecting these vertices represent the connection of the corresponding triangles by MEMS switches. If a switch is activated to connect triangle T1 to triangle T'1 shown in Figure 3, then an edge appears between the vertex T1 and the vertex T'1 as shown in the 1<sup>st</sup> state of the graph model in Figure 4. In this design the connection between T1 and

T2, T2 and T4, T'1 and T'3, T'3 and T'6 are collinear with the reference axis and as a result the edges representing these connections are weighted with  $W=1$  and  $W=2$  for the other connections. The cost of connecting parts at the same direction is less ( $w=1$ ) than connecting parts at a deviated direction ( $w=2$ ).

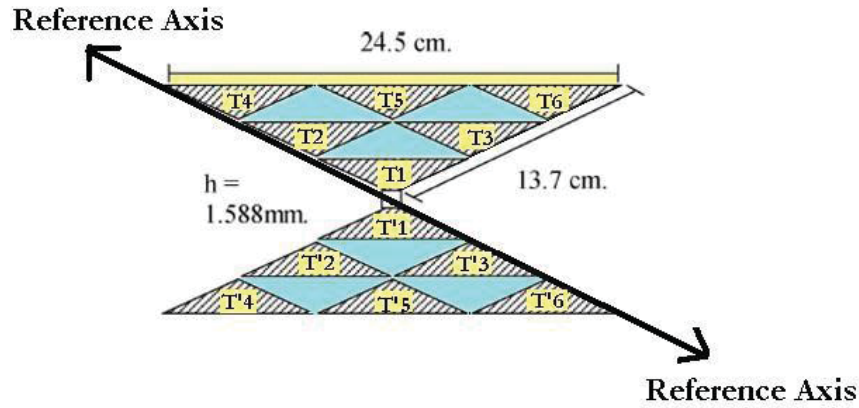
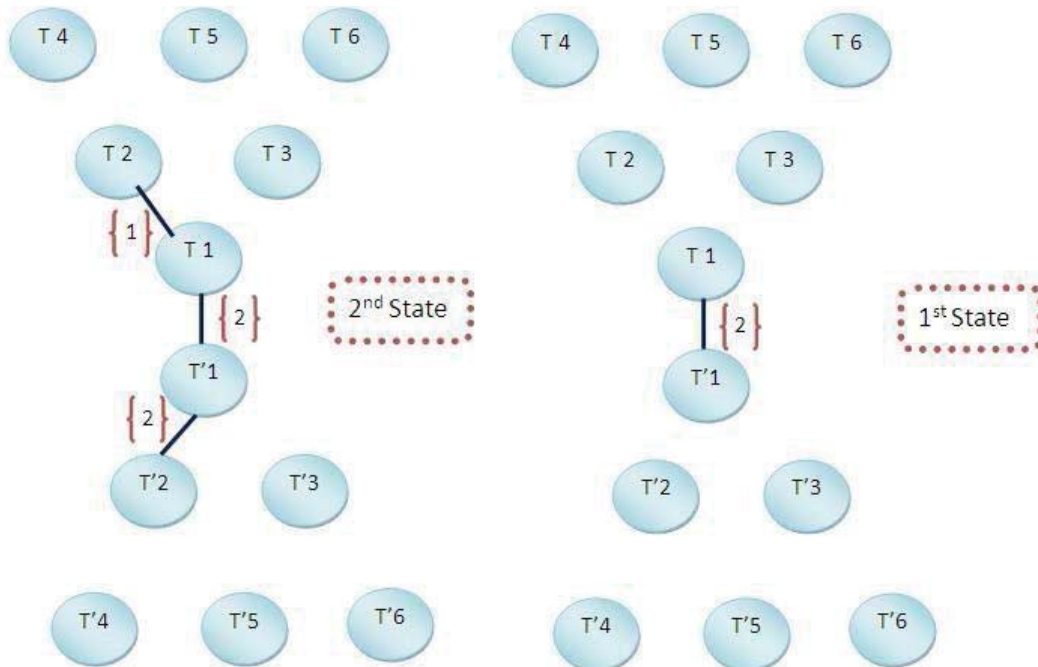


Figure 3 The antenna structure in [3]



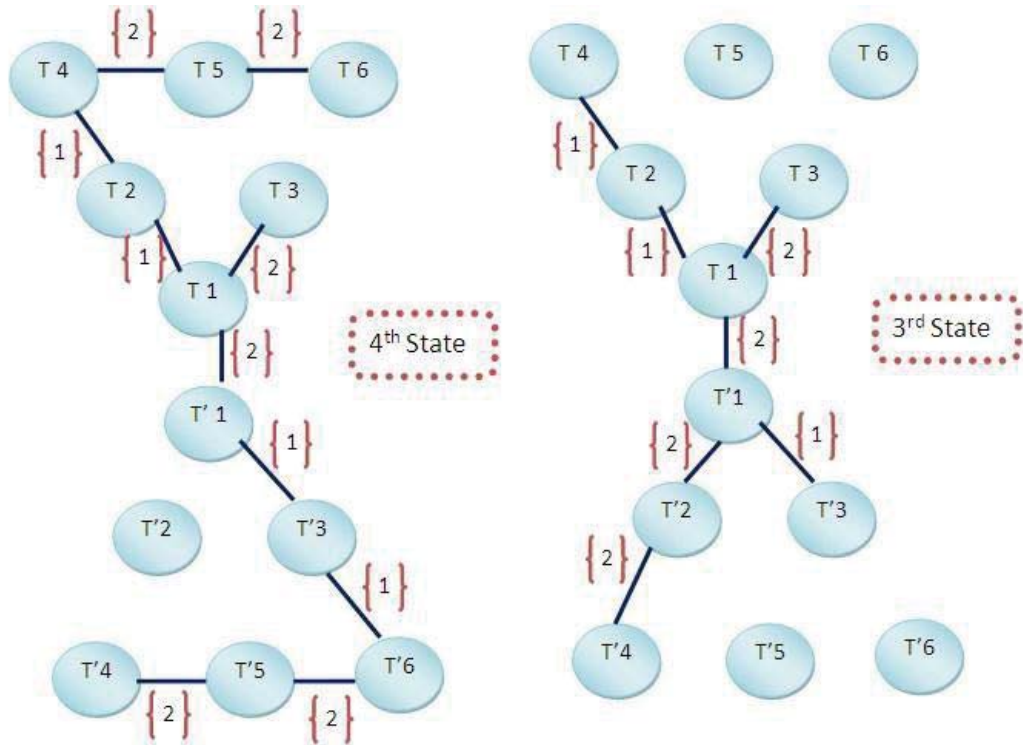


Figure 4 Graph model for different configurations of the antenna in Figure 3

**Rule 1.b:** *The graph modeling of a single part antenna with switches bridging over slots is undirected with non-weighted edges connecting vertices that represent the end points of each switch.*

Valid for:

This rule is valid for group 1 type of antennas with a single part but multiple slots.

Constraints:

In the case of switches bridging over multiple slots in one antenna structure the graph model handles one slot at a time.

Example

As an example, the antenna shown in Figure 5a is analyzed [7]. This antenna is a triangular patch antenna with 2 slots incorporated in it. The authors suggested 5 switches to bridge over each slot to achieve the desired required functions.

The graph modeling of this antenna following rule 1.b is shown in Figure 5b where vertices represent the end points of each switch and edges represent the connections between these end points. When switch 1 is activated an edge appears between N1 and N'1 representing the 2 end-points of switch 1. The graph model of Figure 5b represents each slot at a time. For example N1 represent end-point 1 for switch 1 in slot 1 and end-point 1 for switch 1 in slot 2.

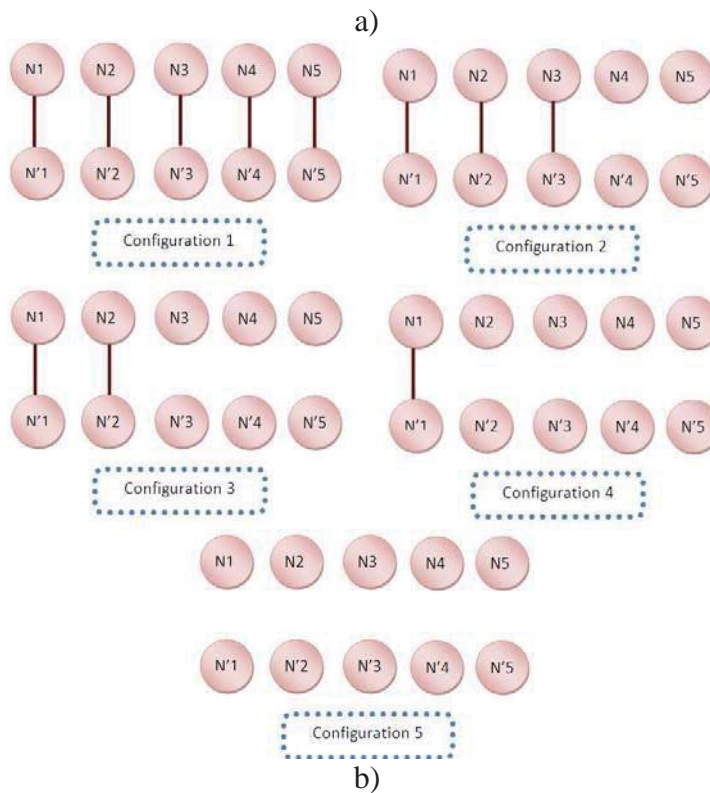
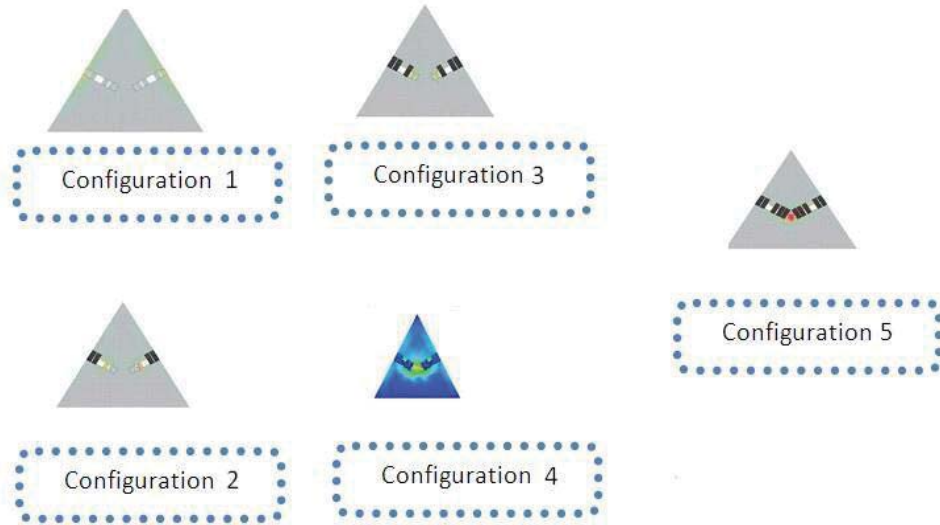


Figure 5 a) Antenna structure with different configurations. b) Graph modeling

***Rule 2.a: The graph modeling of a multi-part antenna with parts connected by capacitors or varactors is undirected with weighted edges connecting vertices that represent the different parts connected.***

Valid for:

This rule is valid for multi-part group 2 antennas.

Constraints:

The edges' weights in this case are calculated according to equation 5. All the capacitances of the different capacitors connecting the parts should be transformed to the same unit and then they should be normalized with respect to the largest capacitance. The weights represent the addition of the normalized capacitances values with the values of  $P_{ij}$  as shown in equation 5.  $P_{ij}$  was discussed in rule 1.a.

$$W_{ij} = P_{ij} + C_{ij}^{normalized} \quad (5)$$

$$\text{Where } P_{ij} = \begin{cases} 1 & A_{ij} = 0^\circ \text{ or } 180^\circ \\ 2 & \text{Otherwise} \end{cases}$$

Where  $A_{ij}$  represents the angle that the connection  $i,j$  form with the reference axis.  $C_{ij}$  represents the normalized capacitance of the capacitor connecting parts  $i$  and  $j$ . In Figure 6 a graph with edges connecting 4 vertices is presented. The weights of the edges are calculated according to equations 6 a,b, and c, assuming that the capacitors have the same unit as:

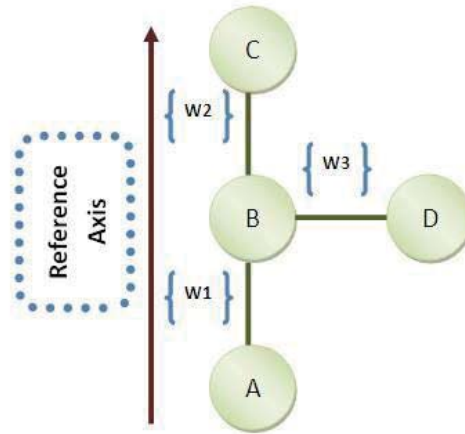


Figure 6 Illustration of Rule 2.a

$$W_1 = \frac{C1}{\text{Max}(C1, C2, C3)} + P_{AB} = \frac{C1}{\text{Max}(C1, C2, C3)} + 1 \quad (6.a)$$

$$W_2 = \frac{C2}{\text{Max}(C1, C2, C3)} + P_{BC} = \frac{C2}{\text{Max}(C1, C2, C3)} + 1 \quad (6.b)$$

$$W_3 = \frac{C3}{\text{Max}(C1, C2, C3)} + P_{BD} = \frac{C3}{\text{Max}(C1, C2, C3)} + 2 \quad (6.c)$$

In this case the antenna shown in Fig. 6 is examined [8]. The antenna is a 2x2 reconfigurable planar wire grid antenna designed to operate in free space. Variable capacitors were placed in the centers of 11 of the 12 wire segments that comprise the grid. The values of the variable capacitors were constrained to lie between 0.1pF and 1 pF. The graph modeling of this antenna follows rule 2.a and is shown in Figure 7. The vertices in this graph model represent the different parts of the lines that are connected together via a capacitor.

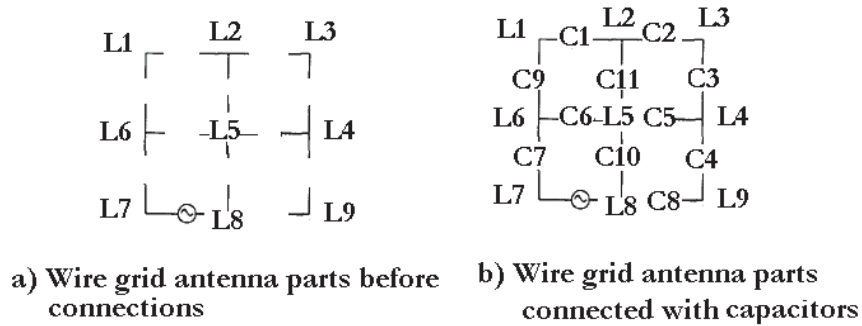


Figure 7 Antenna structure in [8]

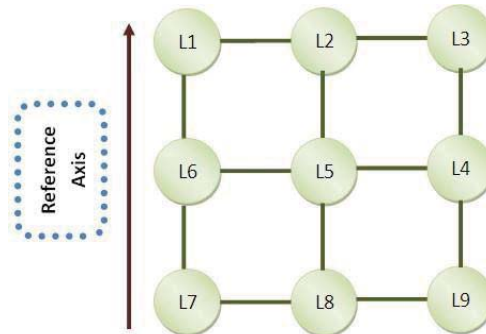


Figure 8 Graph model of the antenna in [8]

The values of the capacitors were not specified in [8]. The weights governing the edges are defined according to equation 5 and are shown in the adjacency matrix

$$A = \begin{bmatrix} 0 & W_{12} & 0 & 0 & 0 & W_{16} & 0 & 0 & 0 \\ W_{21} & 0 & W_{23} & 0 & W_{25} & 0 & 0 & 0 & 0 \\ 0 & W_{32} & 0 & W_{34} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & W_{43} & 0 & W_{45} & 0 & 0 & 0 & W_{49} \\ 0 & W_{52} & 0 & W_{54} & 0 & W_{56} & 0 & W_{58} & 0 \\ W_{61} & 0 & 0 & 0 & W_{65} & 0 & W_{67} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & W_{76} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & W_{85} & 0 & 0 & 0 & W_{89} \\ 0 & 0 & 0 & W_{94} & 0 & 0 & 0 & W_{98} & 0 \end{bmatrix} \quad (7)$$

Where

$$W_{12} = \frac{C1}{\text{Max}(C1, \dots, C11)} + 2$$

$$W_{23} = \frac{C2}{\text{Max}(C1, \dots, C11)} + 2$$

$$W_{34} = \frac{C5}{\text{Max}(C1, \dots, C11)} + 1$$

$$W_{49} = \frac{C6}{\text{Max}(C1, \dots, C11)} + 1$$

$$W_{45} = \frac{C8}{\text{Max}(C1, \dots, C11)} + 2$$

$$W_{56} = \frac{C10}{\text{Max}(C1, \dots, C11)} + 2$$

$$W_{98} = \frac{C11}{\text{Max}(C1, \dots, C11)} + 2$$

$$W_{85} = \frac{C7}{\text{Max}(C1, \dots, C11)} + 1$$

$$W_{52} = \frac{C9}{\text{Max}(C1, \dots, C11)} + 1$$

$$W_{76} = \frac{C7}{\text{Max}(C1, \dots, C11)} + 1$$

$$W_{61} = \frac{C9}{\text{Max}(C1, \dots, C11)} + 1$$

**Rule 2.b:** The graph modeling of a single part antenna where capacitors or varactors are bridging over slots is undirected with weighted edges connecting vertices that represent the end points of each capacitor.

Valid for:

This rule is valid for single part antennas of group 2.

Constraints:

The graph should be undirected and weighted where the weights are defined in equation 8

$$W_{ij} = C_{ij \text{normalized}} \quad (8)$$

Where  $C_{ij}$  represents the normalized capacitance of the capacitor connecting end-points  $i$  and  $j$ . The capacitances values are calculated as discussed in rule 2.a. In the case of multiple slots, rule 1.b applies with the addition of equation 8

Example

As an example consider the antenna shown in Figure 9 [9]. Different variable capacitance diodes (varactors) values are used, and these varactor values are obtained by changing the biasing voltages. The graph modeling of this antenna follows rule 2.b. where the vertices represent the end points of the different varactors. The undirected edges are weighted with different varactor



values. The graph model is shown in Figure 10.

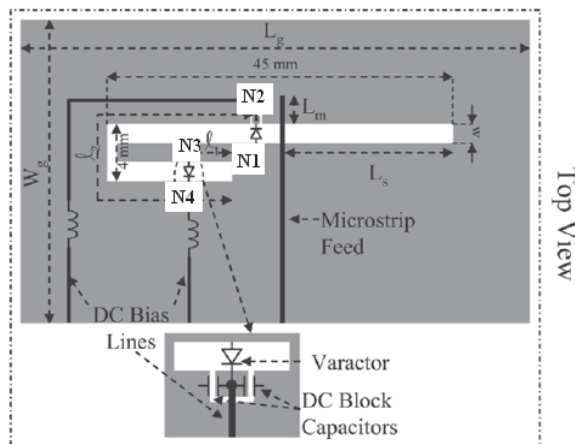


Figure 9 Antenna structure [9]

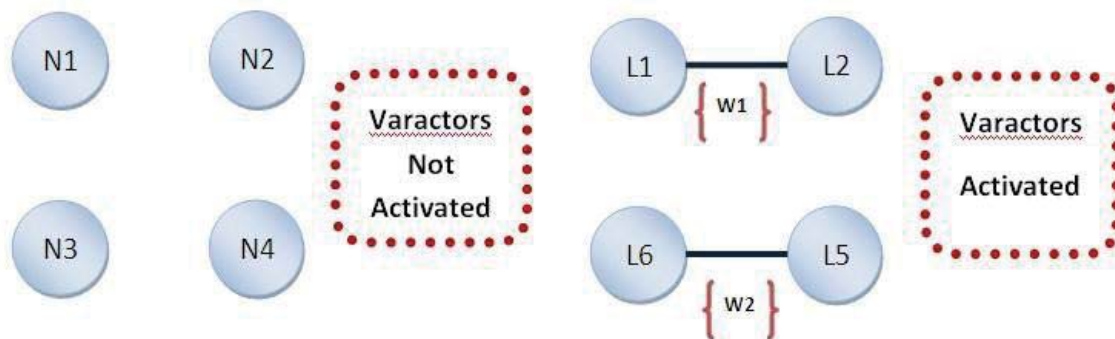


Figure 10 Graph modeling for the antenna in [9]

**Rule 3:** *The graph modeling of an antenna using angular change in its structure is undirected with weighted edges connecting vertices that represent the different angles of the physical action.*

Valid for:

This rule is valid for antennas of group 3.

Constraints:

The graph modeling of this type of antennas is undirected since the angular change (bending or rotation) is reversible. The vertices represent the angles of this physical action. The weighted edges connecting the vertices represent the rotation or the bending process that is the state change from one angle to another. The weights represent constraints related to the system controlling the angular change like the rotation or the bending process i.e. time of rotation...

Example

The antenna shown in Figure 11 falls under this rule [10]. This antenna exhibited a return loss tuning and a reconfigurable radiation pattern. The graph modeling follows rule 3 where the

bending angles are considered as vertices. The physical bending is occurring as a response to an external field applied then removed when the antenna reaches a rest angle. The time it takes for an antenna to reach that rest angle is very important in the antenna's applications. The edges' weights which are the costs that a designer must pay may represent in this case the time of bending. The different weights can be evaluated as in equation 9:

$$w_{ij} = T(A_i \rightarrow A_j) \quad (9)$$

The weight  $W_{ij}$  in this case represents the time it takes to bend the antenna from position  $i$  into position  $j$ . The adjacency matrix  $A$  shown below can be evaluated numerically however the exact numerical values depend on the fabricated system.

$$A = \begin{bmatrix} 0 & T(A_1 \rightarrow A_2) & T(A_1 \rightarrow A_3) & T(A_1 \rightarrow A_4) \\ T(A_2 \rightarrow A_1) & 0 & T(A_2 \rightarrow A_3) & T(A_2 \rightarrow A_4) \\ T(A_3 \rightarrow A_1) & T(A_3 \rightarrow A_2) & 0 & T(A_3 \rightarrow A_4) \\ T(A_4 \rightarrow A_1) & T(A_4 \rightarrow A_2) & T(A_4 \rightarrow A_3) & 0 \end{bmatrix} \quad (10)$$

This antenna is shown in Figure 11 and the graph modeling is shown in Figure 12.

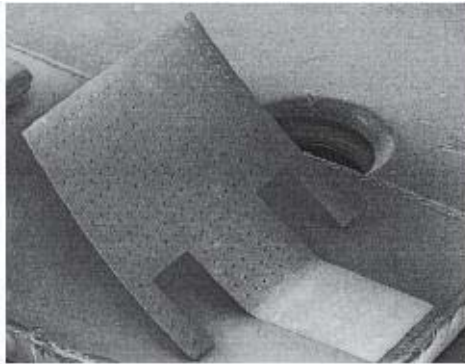


Figure 11 Antenna structure used for rule 3 [10]

In the graph model of Figure 12,  $A_1$  represents  $0^\circ$ ,  $A_2$  represents  $15^\circ$ ,  $A_3$  represents  $45^\circ$  and  $A_4$  represents  $90^\circ$ . Bending from  $0^\circ$  to  $45^\circ$  has to pass by  $15^\circ$  then the path from  $A_1$  to  $A_3$  has to pass by  $A_2$ .

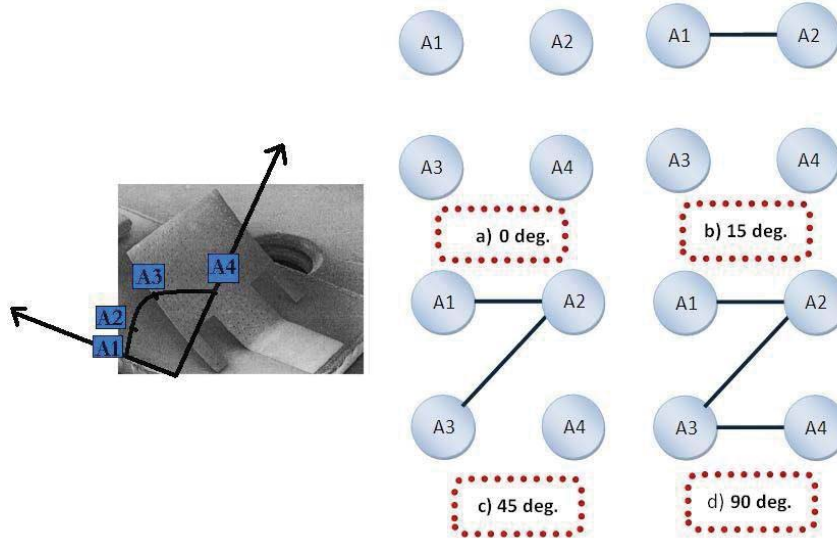


Figure 12 Graph modeling of the antenna in [10]

**Rule 4:** *The graph modeling of an antenna using biasing networks to attach additional parts to each other is undirected with weighted edges connecting vertices that represent the parts of the whole antenna system.*

Valid for:

This rule is valid for antennas of group 4.

Constraints:

The same constraints as rule 1.a apply.

Example

In this case the antenna shown in Figure 13 is studied [11]. Reconfiguration is achieved by turning ON or OFF various sections, to change the active length of the assembled monopole structure. The graph modeling of this antenna follows rule 4 where the different parts are the vertices and the edges represent the connection of these parts by the activation of the different biasing networks. The antenna's graph model is shown in Figure 14.

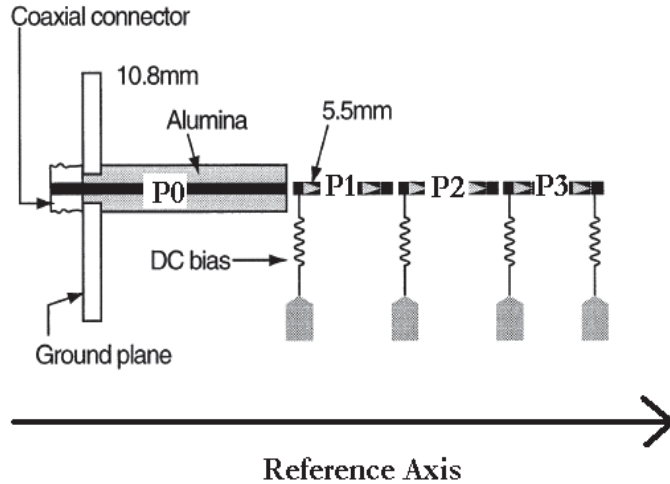


Figure 13 The antenna structure in [11]

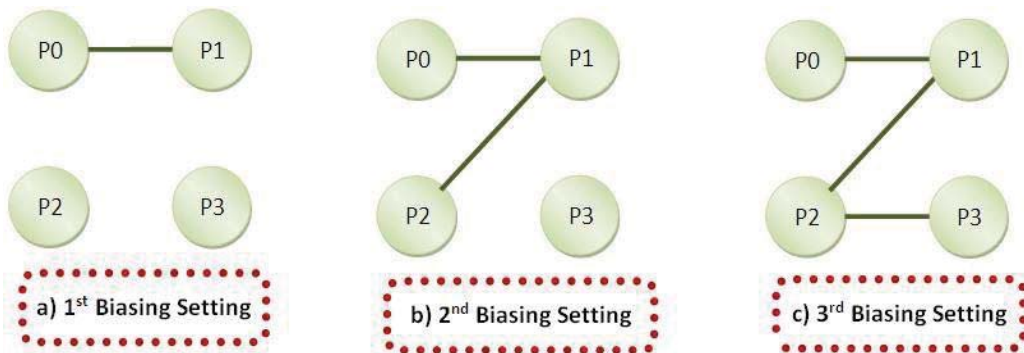


Figure 14 Antenna graph model in [4]

**Rule 5:** *The graph modeling of an antenna array where different antennas are excited at different times is undirected with weighted edges connecting vertices that represent the different antennas forming the array.*

Valid for:

This rule is valid for antenna of group 5.

Constraints:

In the case where different antennas in an array system are excited at different times, the vertices in the modeling graph represent the different antennas. Undirected edges connecting different vertices represent the excitation presence of the corresponding different antennas at the same time. The angle that the antennas form with each other is of importance in the array function. The corresponding graph should be undirected with weighted edges where weights correspond to the antennas' positions relatively to each other in addition to the absolute value of the mutual coupling in the case where mutual coupling is counted.  $M$  is the mutual coupling. All of the

mutual coupling values should be expressed in the same unit and then they should be normalized with respect to the highest value. The weights are calculated according to equation 11.

$$W_{ij} = P_{ij} + |M_{ij \text{ normalized}}| \quad (11)$$

$$\text{Where } P_{ij} = \begin{cases} 1 & A_{ij} = 0^\circ \text{ or } 180^\circ \\ 2 & \text{Otherwise} \end{cases}$$

$A_{ij}$  represents the angle that the antennas have with each other.  $M_{ij}$  represents the normalized mutual coupling amount between antennas  $i$  and  $j$ . If there isn't any mutual coupling between the antennas and  $j$  then  $M_{ij}=0$ .

Example

Consider the antenna shown in Figure 15 [12]. This antenna is a 3-Dimensional model and it exhibits reconfigurable radiation pattern. The graph modeling of this antenna follows rule 5 where the vertices are the different antennas on the different cube faces and the edges between them occur when the corresponding antennas are activated at the same time simulating their connection by the same feeding network and their radiated field coupling connection. The exact mutual coupling values between each 2 antennas were not specified in [12], however the weights are calculated according to equation 11 and are shown in the adjacency matrix  $A$  below. The graph model is shown in Fig. 16.

$$A = \begin{bmatrix} 1 & 2+|M_{12}| & 1+|M_{13}| & 2+|M_{14}| \\ 2+|M_{21}| & 2 & 2+|M_{23}| & 2+|M_{24}| \\ 1+|M_{31}| & 2+|M_{32}| & 1 & 2+|M_{34}| \\ 2+|M_{41}| & 2+|M_{42}| & 2+|M_{43}| & 2 \end{bmatrix} \quad (12)$$

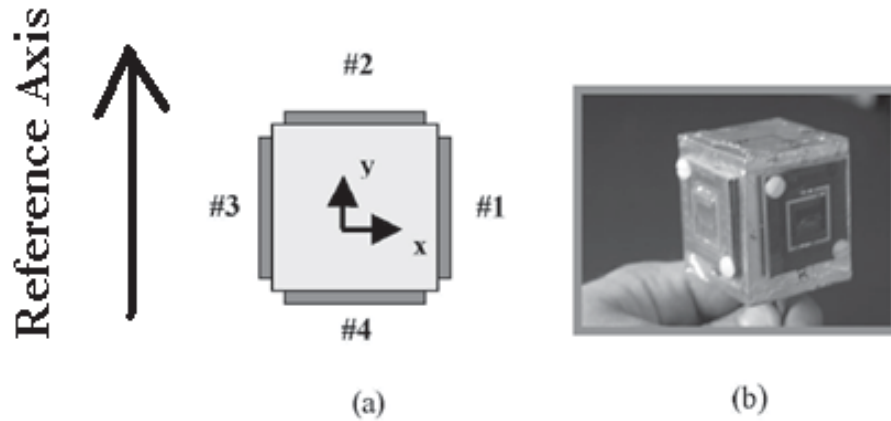


Figure 15 The antenna array in [12]

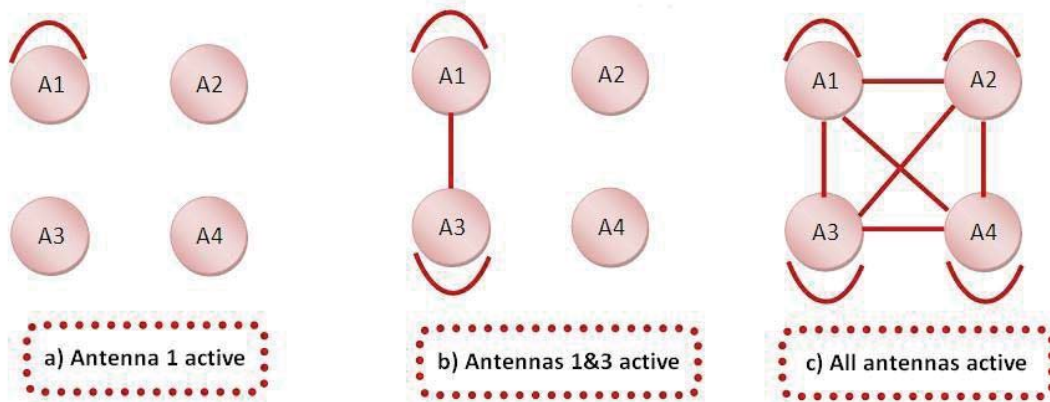


Figure 16 Graph model of the array antenna in [12]

**Rule 6:** *The rules defined previously in this section apply for the graph modeling of a reconfigurable feeding antenna where the reconfiguration is achieved in the feeding network.*

Valid for:

This rule is valid for antennas of group 6.

Constraints:

The graph components in this case represent the feeding components. All the rule constraints defined previously apply correspondingly. For example if the feeding reconfiguration is through switches then rule 1.a applies and so on.

Example

The antenna under consideration is shown in Figure 17 [2]. This antenna is utilizing a parasitic antenna concept to realize pattern diversity. To realize the pattern diversity, each of the slot pairs in

the parasitic patches is loaded by a switchable stub. The stub lengths are adjusted by p-i-n diodes which allow four different patterns for one of the polarization state. By switching ON a diode while the other is OFF, the antenna can switch between horizontal or vertical polarization states with a single feeding port. Figure 17 shows the feeding configuration connected by different diodes. The graph model according to rule 6 leads us to rule 1.a. where the vertices are the different lines in the feeding network connected together. The graph model is shown in Figure 18 where the edges weights are calculated according to equation 4.

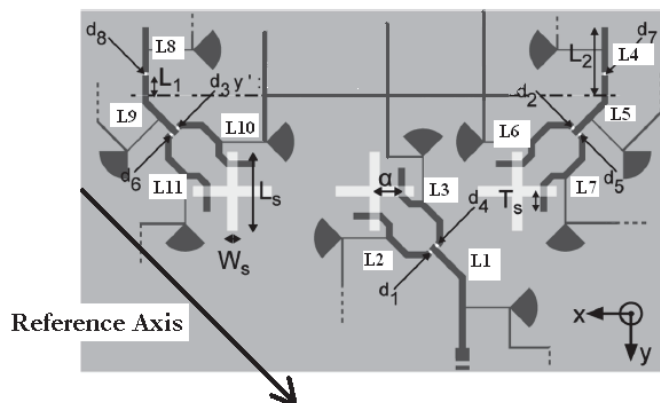
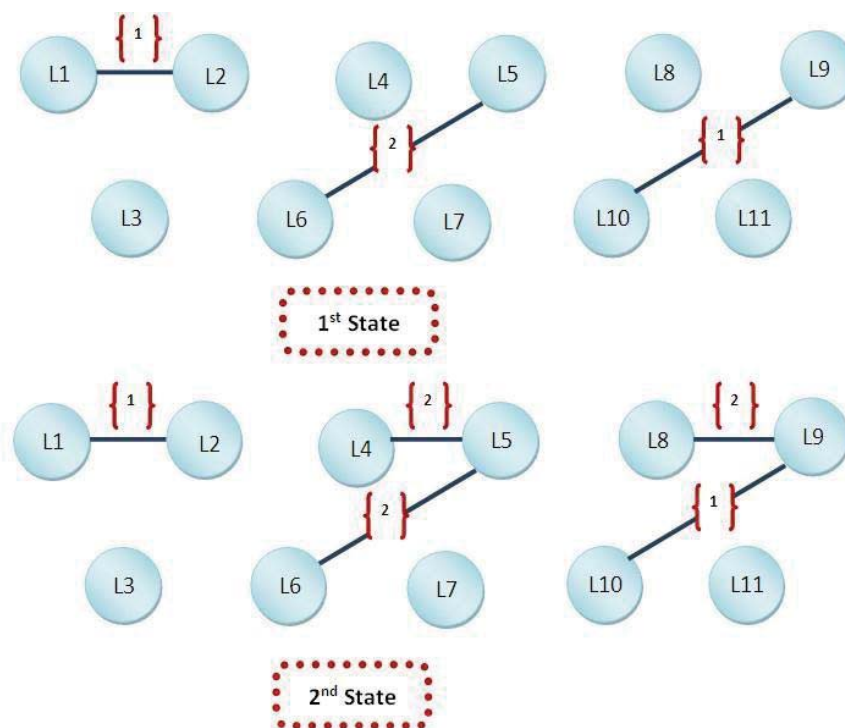


Figure 17 The feeding network of the antenna in [2]



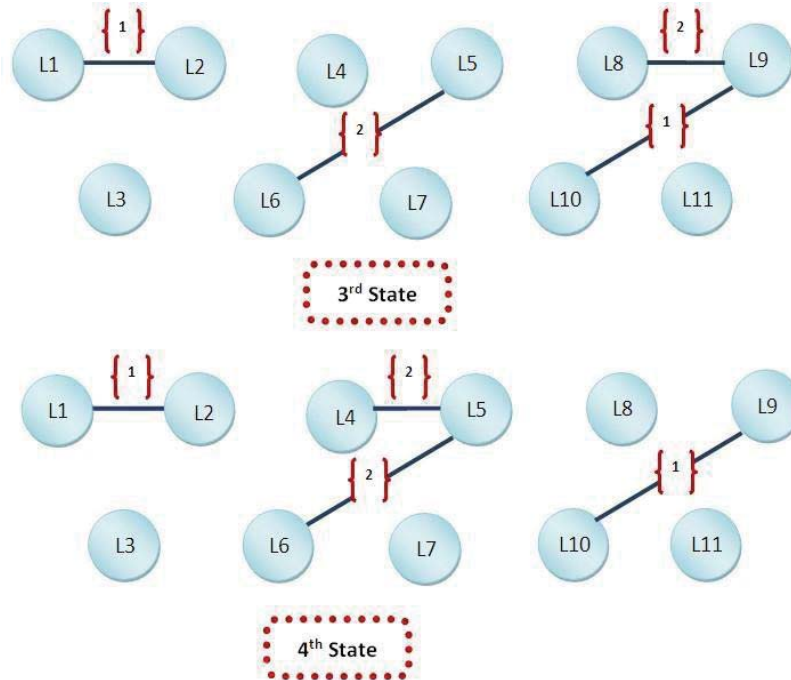


Figure 18 The graph model for the antenna in [2]

A summary of all the previous rules is shown in Table 1.

Table 1 A summary of rules for using graph modeling for reconfigurable antennas

	Multi-Part	Vertices	Directed	Weighted
Switches	YES	Parts	NO	YES
	NO	End-Points	NO	NO
Capacitors	YES	Parts	NO	YES
	NO	End-Points	NO	YES
Angular Change	N/A	Angles	NO	YES
Many Biasing Networks	YES	Parts	NO	YES
Antenna Arrays	N/A	Antennas	NO	YES
Reconfigurable Feeding	YES	Parts	NO	YES



## Dijkstra's Shortest Path Algorithm

### Introduction to Dijkstra's algorithm

Dijkstra's algorithm [13] solves the single source shortest-paths problem on a weighted, directed graph for the case in which all edge weights are nonnegative. The algorithm repeatedly selects the vertex with the minimum shortest-path estimate. For a given node in the graph, the algorithm finds the lowest cost path between that vertex and every other vertex. The algorithm can also be used to find the costs of shortest paths from one node to a certain destination node. The algorithm needs to stop once the shortest path has been identified. A summary of Dijkstra's algorithm operating procedure is shown in Figure 19

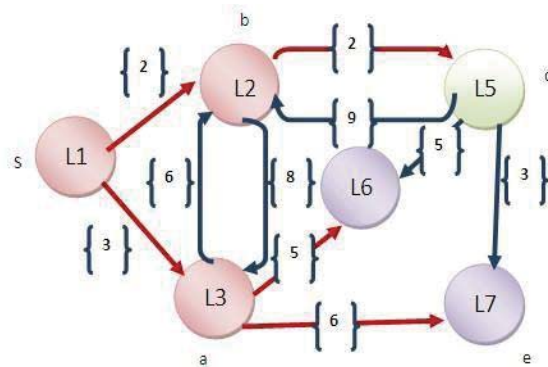


Figure 19 The shortest path shown in red as calculated by Dijkstra's algorithm for a weighted graph

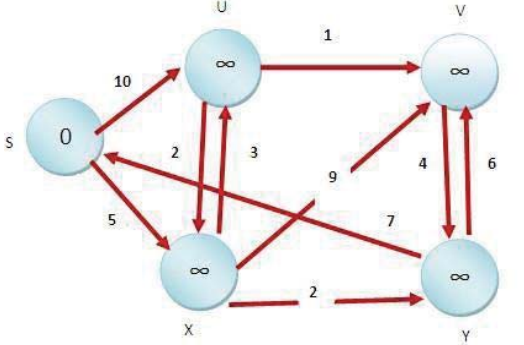
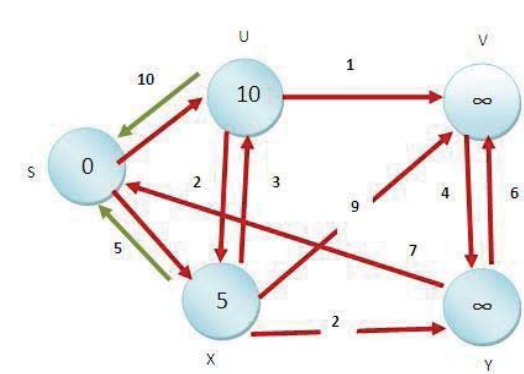
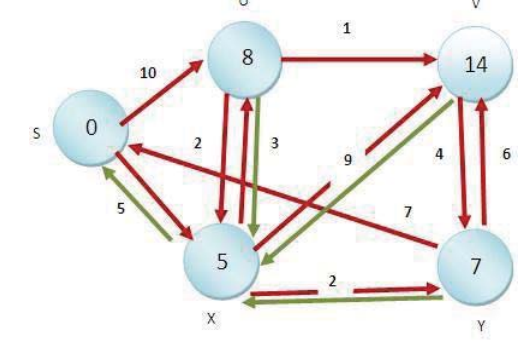
### Dijkstra's Algorithm Mechanism

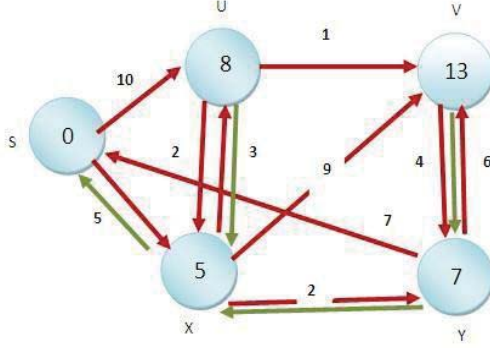
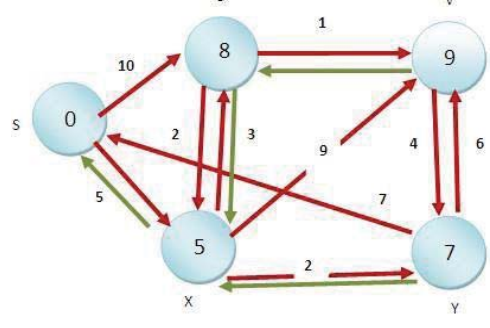
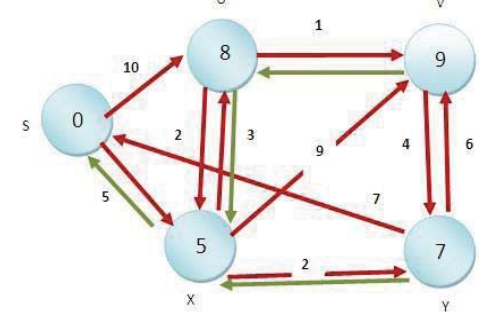
Dijkstra's algorithm starts at the source vertex,  $s$ , it grows a tree,  $T$ , that ultimately spans all vertices reachable from  $S$ . Vertices are added to  $T$  in order of distance i.e., first  $S$ , then the vertex closest to  $S$ , then the next closest, and so on. The distance between two vertices is affected by the edge weight connecting these two vertices. Dijkstra's algorithm can be described in the following 8 steps [13]:

1. INITIALIZE SINGLE-SOURCE ( $G, s$ )
2.  $S \leftarrow \{ \}$  //  $S$  will ultimately contains vertices of final shortest-path weights from  $s$
3. Initialize priority queue  $Q$  i.e.,  $Q \leftarrow V[G]$
4. while priority queue  $Q$  is not empty do
5.    $u \leftarrow \text{EXTRACT\_MIN}(Q)$  // Pull out new vertex
6.    $S \leftarrow S \dot{\cup} \{u\}$  // Perform relaxation for each vertex  $v$  adjacent to  $u$
7.   for each vertex  $v$  in  $\text{Adj}[u]$  do
8.     Relax ( $u, v, w$ )

An illustration of the step by step operation of Dijkstra's algorithm is shown in Table 2

Table 2 The illustration of the step by step operation of Dijkstra's algorithm

<p>Step 1: Given initial graph <math>G=(V, E)</math>. All nodes have infinite cost except the source node, <math>s</math>, which has 0 cost.</p>	
<p>Step 2: First we choose the node, which is closest to the source node, <math>s</math>. Initialize <math>d[s]</math> to 0. Add it to <math>S</math>. Relax all nodes adjacent to source, <math>s</math>. Update predecessor for all nodes updated. (check green arrows)</p>	
<p>Step 3: Choose the closest node, <math>x</math>. Relax all nodes adjacent to node <math>x</math>. Update predecessors for nodes <math>u, v</math> and <math>y</math>. (check green arrows)</p>	

<p>Step 4: Now, node y is the closest node, so add it to S.</p> <p>Relax node v and adjust its predecessor. (green arrows)</p>	 <p>The graph shows nodes s (0), u (8), v (13), x (5), and y (7). Edges are labeled with weights: s-u (10), s-x (5), u-v (1), u-x (2), u-y (3), v-x (9), v-y (4), x-y (2), and x-v (7). Green arrows indicate predecessor adjustments: from u to s, from x to u, from y to x, and from v to y.</p>
<p>Step 5: Now we have node u that is closest.</p> <p>Choose this node and adjust its neighbor node v. (green arrows)</p>	 <p>The graph is the same as in Step 4. A green arrow now points from u to v, indicating that v's predecessor is updated to u.</p>
<p>Step 6: Finally, add node v. The predecessor list now defines the shortest path from each node to the source nodes. (green arrows)</p>	 <p>The graph is the same as in Step 5. A green arrow now points from v to u, indicating that u is the predecessor of v. The final shortest paths are: s to u (10), s to x (5), u to v (1), u to x (2), u to y (3), v to x (9), v to y (4), x to y (2), and x to v (7).</p>

### Applying Dijkstra's Algorithm to the Control of Reconfigurable Antennas

In certain reconfigurable antennas a shorter path may mean a shorter current flow and thus a certain resonance associated with it. A longer path may denote a lower resonance frequency than the shorter path. In reconfigurable antennas resorting to physical and angular alterations a shorter path means a faster response and a swifter reconfiguration.

The antenna in [15] resorts to slot rotation to achieve reconfiguration. Several commercial rotary switches can be used to automatically rotate the slot. Rotary switches can also be customized for this design and implemented through an FPGA (Field Programmable Gate Array) to control the rotation of the slots on the antenna. The graph model of this antenna follows rule 4. Vertices correspond to the antenna's different angles of rotation as shown in Figure 20. The edges connecting these vertices are undirected and they represent the rotation process between 2 angles.

The cyclic flow of the graph is due to the fact that the graph is modeling the rotation process controlled by rotary switches. The mode of operation of rotary switches ensures a sequential rotation. For example if A1 represents 0 degree, A2 represents 30 degrees and A3 represents 60 degrees. The rotation from 0 degree to 60 degrees is represented by edges connecting A1 (0 degree) to A2 (30 degrees) and A2 (30 degrees) to A3 (60 degrees). The edges in the graph are weighted. In this model the weights represent the time of rotation from one angle into another and they are calculated according to equation 9. The graph modeling of this antenna with all possible edges is shown in Fig. 20.

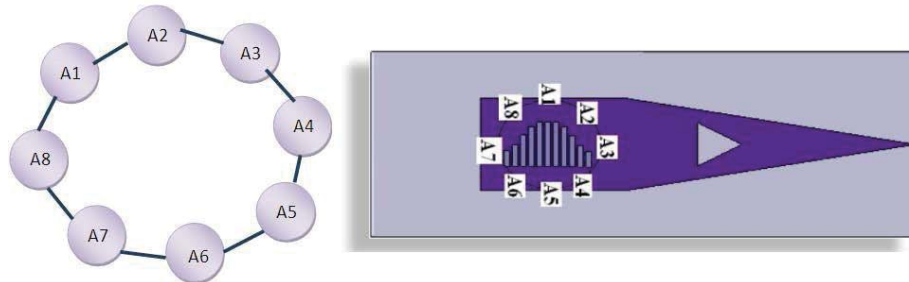


Figure 20 All possible configurations represented by all possible edges

In the case where this antenna is implemented on a time sensitive platform like a satellite, arriving at the desired position with the shortest time possible, regardless of the system constraints is very important. Finding the shortest path for each possible scenario is a major requirement.

The complication of the rotational slotted antenna may vary from one system into another. For example: In a particular system, going from 0 degree to 30 degrees might be more costly than going from 0 to 230 degrees and thus the importance of application of algorithms like Dijkstra's algorithms. This algorithm can be used to program an FPGA to control the rotation of the slots through a rotary switch. The direction of rotation of the slots will be chosen according to the direction of the shortest path to move from one position into another.

The adjacency matrix A showing all the graph weights calculated according to equation 9 is shown below. This matrix represents only four rotation processes. It can be evaluated numerically however the exact numerical values depend on the fabricated system and the rotary switch used.

$$A = \begin{bmatrix} 0 & T(A_1 \rightarrow A_2) & T(A_1 \rightarrow A_3) & T(A_1 \rightarrow A_4) \\ T(A_2 \rightarrow A_1) & 0 & T(A_2 \rightarrow A_3) & T(A_2 \rightarrow A_4) \\ T(A_3 \rightarrow A_1) & T(A_3 \rightarrow A_2) & 0 & T(A_3 \rightarrow A_4) \\ T(A_4 \rightarrow A_1) & T(A_4 \rightarrow A_2) & T(A_4 \rightarrow A_3) & 0 \end{bmatrix} \quad (13)$$

## References

- 1) J. Clark, and D. A. Holton, *A First Look At Graph Theory*, World Scientific Publishing Company, 1991.
- 2) B. Poussot, J. M. Laheurte, L. Cirio, O. Picon, D. Delcroix and L. Dussopt, "Diversity measurements of a reconfigurable antenna with switched polarizations and patterns", *IEEE Transactions on Antennas and Propagation*, Vol. 56, Issue 1, pp. 31-38, Jan. 2008
- 3) D. E. Anagnostou, Z. Guizhen, M.T. Chrysomallis, J. C. Lyke, G.E. Ponchak, J. Papapolymerou, and C. G. Christodoulou "Design, fabrication and measurement of an RF-MEMS-based self –similar reconfigurable antenna", *IEEE Transactions on Antennas and Propagation*, Vol. 54, Issue 2, Part 1, pp. 422-432, Feb 2006.
- 4) A. Patnaik, D. E. Anagnostou, C.G. Christodoulou and J. C. Lyke, "Neurocomputational analysis of a multiband reconfigurable planar antenna", *IEEE Transactions on Antennas and Propagation*, Vol. 53, Issue 11, pp. 3453-3458, Nov. 2005.
- 5) A. Patnaik, D. E. Anagnostou, C.G. Christodoulou and J.C. Lyke "A frequency reconfigurable antenna design using neural networks", *IEEE Antennas and Propagation society international symposium*, Vol. 2A, pp.409-412, July 2005.
- 6) N. Kingsley, D. E. Anagnostou, M. Tentzeris and J. Papapolymerou, "RF MEMS sequentially reconfigurable sierpinski antenna on a flexible organic substrate with novel DC-biasing technique", *Journal of Microelectromechanical Systems*, Vol. 16, Issue 5, pp. 1185-1192, Oct. 2007.
- 7) L. M. Feldner, C. D. Nordquist, and C. G. Christodoulou, "RFMEMS Reconfigurable Triangular Patch Antenna", *IEEE AP/URSI International Symposium*, Vol. 2A, pp. 388-391, July 2005.
- 8) C. S. Deluccia, D. H. Werner, P. L. Werner, M. Fernandez Pentoja, and A.R. Bretones, "A novel frequency agile beam scanning reconfigurable antenna", *IEEE Antennas and Propagation society international symposium*, Vol. 2, pp. 1839-1842, June 2004.
- 9) N. Behdad and K. Sarabandi, "Dual-Band reconfigurable antenna with a very wide tunability range", *IEEE Transactions on Antennas and Propagation*, Vol. 54, Issue 2, Part 1, pp. 409-416, Feb. 2006.
- 10) J. C. Langer, J. Zou, C. Liu, and J. T. Bernhard, "Micromachined Reconfigurable Out Of Plane Microstrip Patch Antenna Using Plastic Deformation Magnetic Actuation", *IEEE Microwave and Wireless Components Letters*, Vol. 13 No 3, pp. 120-122, March 2003.
- 11) A. E. Fathy, A. Rosen, H. S. Owen, F. McGuinty, D. J. Mcgee, G. C. Taylor, R. Amantea, P. K. Swain, S. M. Perlow, and M. Elsharbiny, "Silicon-based reconfigurable antennas-concepts, analysis, implementation and feasibility", *IEEE Transactions on Microwave Theory and Techniques*, Vol. 51, Issue 6, pp. 1650-1661, June 2003.
- 12) T. L. Roach, G. H. Huff and J. T. Bernhard, "Enabling high performance wireless communication systems using reconfigurable antennas", *Military Communications Conference*, pp. 1-6, Oct. 2006.
- 13) T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introductions to Algorithms", MIT Press, 2001.
- 14) J. Costantine, K. Y. Kabalan, A. El Hajj and C. G. Christodoulou, "New Multi\_Band Design for a Microstrip Patch Antenna", *The Second European Conference on Antennas and Propagation*, pp. 1-4, Nov. 2007.

## Table of Acronyms

<b>Acronym</b>	<b>Full</b>
1U-1	CubeSat Unit
ADAC	Attitude Determination and Control
AFB	Air Force Base
AFRL	Air Force Research Laboratory
AFOSR	Air Force Office of Scientific Research
AIAA	American Institute of Aeronautics and Astronautics, Inc.
AM	Additive Manufacturing
AMBA	Advanced Microcontroller Bus Architecture
ANN	Artificial Neural Network
API	Application Programming Application
APT	Advanced Plug and Play Technology
ASIC	Application Specific Integrated Circuit
ASIM	Appliqué Sensor Interface Module
AWM	Adaptive Writing Manifold
AWP	Adaptive Writing Panel
BER	Bit Error Rates
BFM	Bus Functional Models
C&DH	Command and Data Handler
Cal Poly	California Polytechnic Institute
CEC	Career Enrichment Center
CINT	Center for Integrated Technology
CLBs	Configurable Logic Blocks
CLK	Clock
CMU	Cell Management Unit
C/NOFS	Communications/Navigation Outage Forecasting System
CoNNeCT	Communications, Navigation, and Networking reConfigurable Testbed
COSMIAC	Configurable Space Microsystems Innovations and Applications Center
COTS	Commercial off-the-shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CU	Cell Unit
DDFX	Dynamic Dual Fixed-Point
DDR2	Double Data Rate 2
DFX	Dual Fixed-Point
DMS	Defected Microstrip Structure
DPR	Dynamic Partial Reconfiguration
DR	Dynamically Reconfigurable
DSP	Digital Signal Processing
DSSS	Direct Sequence Spread Spectrum
DSSS-FH	Direct Sequence Spread Spectrum with Frequency Hopping
DUT	Device under Test
ELaNa	Educational Launch of NanoSatellites

ETH	Ethernet
FH	Frequency Hopping
FMAC	FPGA Mission Assurance Center
FPGA	Field Programmable Gate Array
FP	Floating-point
FPU	Floating-point unit
FSL	Fast Simple Link
FT	Fault Tolerant
FX	Fixed-point
GCC	GNU Compiler Collection
GENSO	Global Educational Network for Satellite Operations
GNU GPL	GNU General Public License
GPIO	General Purpose Inputs/Outputs
GRC	GNU Radio Companion
GSFC	NASA Goddard Space Flight Center
GUI	Graphical User Interface
HDL	Hardware Descriptive Language
HWICAP	HardWare Internal Configuration Access Port
I <sup>2</sup> C	A well-known standard that supports bus clock speeds up to 400KHz
I/O	Input/Output
ICAP	Internal Configuration Access Port
ISE	Integrated Software environment
ISS	International Space Station
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LUT	Lookup Table
MAPLD	Military/Aerospace Programmable Logic Devices
MEMS	Micro-electromechanical systems
MinSOC	Minimal System-On-Chip
Monarch	Modular Open Network Architecture
NAPA	Nanosatellite and Plug-and-play Architecture
NASA	National Aeronautics and Space Administration
NBTI	Negative Bias Temperature Instability
NCHARs	National Characters (Fixed-length Unicode string data)
NN	Neural Network
NSREC	Nuclear and Space Radiation Conference
OpenRISC	Open Source Reduced Instruction Set Computing (RISC) CPU architecture
ORS	Operationally Responsive Space
PDR	Preliminary Design Review
PIC	Peripheral Interface Controller
PHY	Physical Layer
PnP	Plug and Play
PPC	PowerPC
PPM	Pluggable Processor Module
PPS signal	Pulse-Per-Second

RHAS	Radiation Hazard Assessment System
RAM	Random-Access Memory
RF	Radio Frequency
RHEME	RadHard Electronic Memory Experiment
RMAP	Reference mapping
RTL	Register Transfer Language
RTU	Remote Terminal Unit
FIFO	First In, First Out
SCL	I <sup>2</sup> C clock rates over 400KHz
SDA	Serial Data Line
SDM	Satellite Data Model
SDR	Software Defined Radio
SEU	Single Event Upset
SFPU	Single Point Floating Unit
SPA	Space Plug-and-play Architecture
SPARC	Scalable Processor Architecture
SPFP	Single-precision Floating-Point
SPI	Serial Peripheral Interface.
SPW	Space Wire
SRAM	Static Random-Access Memory
SSM	Satellite Systems Manager
STEM	Science, Technology, Engineering and Mathematics Fields
SUT	Systems under test
SWFP	Software emulated Floating Point.
TBP	Test-By-Pass
TMR	Triple Modular Redundancy
TSMC	Taiwan Semiconductor Manufacturing Company
UART	Universal Asynchronous Receiver/Transmitter
UCLA	University of California, Los Angeles
UHF	Ultra High Frequency
UNM	University of New Mexico
UNP	University Nanosat Program
USRP	Universal Software Radio Peripheral
USB	Universal Serial Bus
USRP 2	Universal Software Radio Peripheral Number 2
UTEP	University of Texas El Paso
VHDL	VHSIC Hardware Description Language
VHF	Very High Frequency
VHSIC	Very-High-Speed Integrated Circuits
VSI	Virtual Satellite Integrator
WDG	WatchDog
XRTC	Xilinx Radiation Test Consortium
XSG	Xilinx System Generator



## DISTRIBUTION LIST

DTIC/OCP

8725 John J. Kingman Rd, Suite 0944

Ft Belvoir, VA 22060-6218

1 cy

AFRL/RVIL

Kirtland AFB, NM 87117-5776

2 cys

Official Record Copy

AFRL/RDMS/Casey DeRaad

1 cy