NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

RESEARCH, DEVELOPMENT AND TESTING OF A FAULT-TOLERANT FPGA-BASED SEQUENCER FOR CUBESAT LAUNCHING APPLICATIONS

by

Lucas S. Parobek

March 2013

Thesis Co-Advisors: Herschel H. Loomis, Jr.
James H. Newman

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

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 instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the 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 Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE RESEARCH, DEVELOPMENT AND TESTING OF A FAULT-TOLERANT FPGA-BASED SEQUENCER FOR CUBESAT LAUNCHING APPLICATIONS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Lucas S. Parobek			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis concerns various means of implementing fault tolerance in logic for use in a general payload processor design. The first specific application of this research is a sequencer developed for deploying CubeSats. The sequencer shall be capable of the timing and accurate deployment of multiple CubeSats from a host spacecraft and shall have the capability for quick reconfiguration prior to launch. This research considers a variety of hardware for suitability toward sequencer construction; field programmable gate arrays (FPGAs) are chosen as the primary device. The design further evolves to selection of the Actel ProASIC3 series of FPGAs. Initial logic test configurations are implemented on a development kit with analysis of results provided. Fault-tolerant techniques are compared with a set of experiments to determine optimum resource utilization and timing schemes. Triple modular redundancy (TMR) is selected as the technique for fault-tolerant logic implementation in the sequencer. Preliminary test boards are built via schematic design and printed circuit board layout. The manufacturing, integration and testing of the 'ProASIC3 Test Board' is fully discussed. A follow-on flight prototype board is designed with more extensive hardware allowing for implementation of fault-tolerant techniques and future growth capability. Recommendations for future work are discussed.

14. SUBJECT TERMS Single-Event Effect (SEE), Single-Event Upset (SEU), Multiple-Bit Upset (MBU), Field Programmable Gate Array (FPGA), Fault Tolerance, Triple Modular Redundancy (TMR), Quadruple Force Decide Redundancy (QFDR), Quadded Logic, CubeSat, Satellite, Altium, Actel, Microsemi, ProASIC3, Xilinx, Virtex			**15. NUMBER OF PAGES** 224
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

RESEARCH, DEVELOPMENT AND TESTING OF A FAULT-TOLERANT FPGA-BASED SEQUENCER FOR CUBESAT LAUNCHING APPLICATIONS

Lucas S. Parobek
Lieutenant, United States Navy
B.S., The Citadel, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2013

Author: Lucas S. Parobek

Approved by: Herschel H. Loomis, Jr.
Thesis Co-Advisor

James H. Newman
Thesis Co-Advisor

Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis concerns various means of implementing fault tolerance in logic for use in a general payload processor design. The first specific application of this research is a sequencer developed for deploying CubeSats. The sequencer shall be capable of the timing and accurate deployment of multiple CubeSats from a host spacecraft and shall have the capability for quick reconfiguration prior to launch. This research considers a variety of hardware for suitability toward sequencer construction; field programmable gate arrays (FPGAs) are chosen as the primary device. The design further evolves to selection of the Actel ProASIC3 series of FPGAs. Initial logic test configurations are implemented on a development kit with analysis of results provided. Fault-tolerant techniques are compared with a set of experiments to determine optimum resource utilization and timing schemes. Triple modular redundancy (TMR) is selected as the technique for fault-tolerant logic implementation in the sequencer. Preliminary test boards are built via schematic design and printed circuit board layout. The manufacturing, integration and testing of the 'ProASIC3 Test Board' is fully discussed. A follow-on flight prototype board is designed with more extensive hardware allowing for implementation of fault-tolerant techniques and future growth capability. Recommendations for future work are discussed.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
- **A. PURPOSE 1**
- **B. NAVAL POSTGRADUATE SCHOOL CUBESAT LAUNCHER (NPSCUL) BACKGROUND 3**
  - **1. NPSCuL Objective and History 3**
  - **2. NPSCuL Physical Design Characteristics 3**
  - **3. History of Sequencer Requirement 5**
  - **4. Concept of Sequencer Development 6**
  - **5. Sequencer Requirements and Specifications 7**
- **C. RELATED WORK AND DOCUMENTATION 9**
- **D. CONCEPT OF DEVELOPMENT STRATEGY 10**
- **E. THESIS ORGANIZATION 11**

II. SEQUENCERS, RADIATION HARDENING, AND FAULT-TOLERANCE 13
- **A. SEQUENCER DESIGN METHODOLOGY 13**
  - **1. Timing Concerns Specific to Initiating P-POD Deployment 13**
  - **2. Chosen Approach 14**
- **B. RADIATION HARDENING 15**
  - **1. Single-Event Effects (SEEs) 16**
    - ***a. Single-Event Upsets (SEUs) 17***
    - ***b. Single-Event Transients (SETs) 18***
    - ***c. Single-Event Latch-ups (SELs) 19***
  - **2. Total Ionizing Dose (TID) 19**
  - **3. Testing of FPGA Hardware in Radiation Environments 20**
  - **4. Radiation Protection Approaches 20**
- **C. FAULT TOLERANCE 23**
  - **1. Scrubbing / Re-programming at Intervals 23**
  - **2. Triple Modular Redundancy (TMR) 24**
    - ***a. Block TMR (BTMR) 24***
    - ***b. Local TMR (LTMR) 25***
    - ***c. Global TMR (GTMR) 25***
    - ***d. Distributed TMR (DTMR) 26***
  - **3. Quadded Logic 27**
  - **4. Triplicated Interwoven Redundancy (TIR) 28**
  - **5. Quadruple Force Decide Redundancy (QFDR) 30**
  - **6. Advantages and Disadvantages of Various Fault-Tolerant Techniques 31**
- **D. CHAPTER SUMMARY 32**

III. FIELD PROGRAMMABLE GATE ARRAYS AND SELECTION OF SEQUENCER TECHNOLOGY 33
- **A. FIELD PROGRAMMABLE GATE ARRAYS 33**
  - **1. Physical Construction 33**

a. Internal Fabric *34*
b. Programmability and Reconfiguration *36*
2. Historical Overview of FPGA Technology 37
3. Types of Modern FPGA Technology 39
a. Antifuse-based FPGAs *39*
b. SRAM-based FPGA Implementations *40*
c. Flash-based FPGAs *41*
4. Design Concerns 42
B. FINAL SELECTION OF ACTEL PROASIC3 SERIES FPGA 43
1. Required FPGA Characteristics 43
2. Overall Sequencer and Processor Characteristics 44
3. Comparisons of Xilinx and Actel FPGAs 44
C. SELECTION OF DEVELOPMENT BOARDS FOR LOGIC DESIGN 47
D. CHAPTER SUMMARY 48

IV. HARDWARE DEVELOPMENT – PROASIC3 TEST BOARDS AND PRELIMINARY FLIGHT PROTOTYPE 49
A. PROASIC3 TEST BOARD 49
1. Derived Requirements 50
2. Selection of Board Size and Layer Count 50
3. Selection of Components 51
B. SAD VERSION 3 – FLIGHT PROTOTYPE BOARD 54
1. Discussion of Specific FPGA 55
2. Additional Modifications Considered 58
C. CHAPTER SUMMARY 58

V. COMPARISON OF FAULT-TOLERANCE METHODS FOR USE IN SEQUENCER LOGIC DEVELOPMENT 61
A. TMR VERSUS OTHER FAULT TOLERANT APPROACHES 61
1. Maintaining Logic Structures within HDL Languages (Verilog / VHDL and Schematically) 61
a. Baseline Design – Seven-Segment LED Counter Example *62*
b. Triplicated Design – Seven-Segment LED Counter Example *63*
2. Implementation of Various Redundant Techniques in Combinational Logic Designs 65
a. Baseline Design – 16×16-bit Array Multiplier *65*
b. Triple Modular Redundancy (TMR) *68*
c. Quadded Logic *69*
d. Triplicated Interwoven Redundancy (TIR) *72*
e. Stacked Techniques – Quad-TMR & TIR-TMR *74*
3. Data Results and Analysis 75
4. Discussion of Results 77
B. SELECTION OF A FAULT-TOLERANT TECHNIQUE 78
1. Xilinx TMRTool and Mentor Graphics Precision Hi-Rel 79

2. Manual Insertion of TMR with Schematic and HDL Techniques 80
C. APPLICATION OF TMR TO SEQUENTIAL LOGIC DESIGNS 80
1. Manual Insertion of Faults 82
a. Floating-Input Case 82
b. 'Stuck-at-0' Logic Fault 83
c. 'Stuck-at-1' Logic Fault 84
2. Modification of FSM Design to TMR Representation 85
3. Timing and Resource Comparison for Sequential Circuits 87
D. PRELIMINARY SEQUENCER LOGIC DESIGN 89
1. Conversion of Xilinx HDL to Actel HDL 89
2. Synplify Pro Representations of FSM Logic 90
3. Designation of Timing Constraints within Logic 92
a. Parameter Constants in Verilog HDL 93
b. Constant Declarations in ProASIC3 FlashROM 93
E. CHAPTER SUMMARY 95

VI. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH 97
A. CONCLUSIONS 97
B. FOLLOW-ON RESEARCH AND TESTING 100

APPENDIX A. FPGA HARDWARE PRODUCTS AND SOFTWARE DEVELOPMENT TOOL DIFFERENCES 103
A.1 COMPARISON OF XILINX AND ACTEL PRODUCTS 103
1. Software Feature Set 103
a. Xilinx ISE 14.3 104
b. Actel Libero SoC 10.1 107
c. Results of Software Comparison 109
2. Hardware Feature Set 109
a. Xilinx Virtex Series FPGAs 110
b. Actel ProASIC3 Series FPGAs 112
3. Associated Hardware Requirements 115
A.2 APPENDIX SUMMARY 115

APPENDIX B. PROASIC3 TEST BOARD – REFERENCE, SCHEMATICS, AND PCB LAYOUT DIAGRAMS 117
B.1 PROASIC3 TEST BOARD DEVELOPMENT METHODOLOGY 117
1. Detailed Discussion of ProASIC3 Test Board Design Decisions 117
2. Discussion of Schematic 119
3. Discussion of PCB Layout and Component Placement 121
4. Physical Construction and Testing 123
B.2 BILL OF MATERIALS 127
B.3 MECHANICAL DIMENSIONS AND ELECTRICAL SCHEMATICS 128
B.4 2-D PCB LAYER DIAGRAMS 131
B.5 3-D PCB COMPONENT PLACEMENT 134

B.6 APPENDIX SUMMARY 135

APPENDIX C. SAD VERSION 3 – FLIGHT PROTOTYPE BOARD – REFERENCE, SCHEMATICS, AND PCB LAYOUT DIAGRAMS 137

C.1 SAD VERSION 3 – FLIGHT PROTOTYPE BOARD DEVELOPMENT METHODOLOGY 137

1. Selection of Board Size and Layer Count 137

2. Selection of Components 141

3. Discussion of Schematic 144

4. Discussion of PCB Layout and Component Placement 146

C.2 BILL OF MATERIALS 149

C.3 MECHANICAL DIMENSIONS AND ELECTRICAL SCHEMATICS 151

C.4 2-D PCB LAYER DIAGRAMS 157

C.5 3-D PCB COMPONENT PLACEMENT 163

C.6 APPENDIX SUMMARY 164

APPENDIX D. SEQUENCER HDL PRESENTATION 165

D.1.1 SEVEN-SEGMENT LED COUNTER (BASYS2USERDEMO.VHD) 165

D.1.2 SEVEN-SEGMENT LED COUNTER (SIMPLESSEGLEDDEMO.VHD) 167

D.1.3 SEVEN-SEGMENT LED COUNTER (VOTER.V) 169

D.1.4 SEVEN-SEGMENT LED COUNTER (TMR SCHEMATIC) 170

D.2 LAUNCH SEQUENCE GENERATOR (LAUNCH_SEQUENCER.VHD) 171

LIST OF REFERENCES 181

INITIAL DISTRIBUTION LIST 191

LIST OF FIGURES

Figure 1. Final payload configuration and integration of NPSCuL with eight P-PODs. 4
Figure 2. NPSCuL with associated Splitter Auxiliary Device (SAD) and external wiring visible. 5
Figure 3. Internal wiring harness structure for SAD depicting power and data line split to eight output ports. 5
Figure 4. Neutron radiation strike of silicon atom resulting in ejection of heavy ions leading to firm error in transistor (From [21]). 15
Figure 5. Single-event effects classification diagram illustrating difference between recoverable and non-recoverable errors (From [25]). 17
Figure 6. MBU of six bits triggered by proton induced oxygen heavy-ion in 130-nm CMOS memory cells (From [27]). 18
Figure 7. Inner and outer torus-shaped layers of plasma of the Van Allen radiation belts. Energetic electrons form the outer belt; the inner belt consists of protons and electrons (From [39]). 21
Figure 8. Local TMR with all sequential elements tripled and followed by a majority voter (From [44]). 25
Figure 9. Global TMR with all sequential elements, combinatorial logic, majority voters, clocking signals and global buffers triplicated (From [44]). 26
Figure 10. Distributed TMR with all sequential elements, combinatorial elements and majority voters triplicated. Clock signal is shared among resources (From [44]). 27
Figure 11. Quadded logic illustrating error propagation and masking (From [46]). 28
Figure 12. Design of TIR half-adder implementation illustrating interwoven connections (From [47]). 29
Figure 13. QFDR substituting LUTs and FFs constructs for logic gates (From [48]). 30
Figure 14. Typical FPGA internal architecture consisting of configurable logic blocks, I/O blocks and programmable interconnects (From [54]). 34
Figure 15. Typical FPGA logic cells with associated LUTs, gates, adders, memory devices, and interconnect buffers visible (From [55]). 35
Figure 16. Internal flash transistor configuration with charge build-up in floating gate allowing for reconfiguration and nonvolatile storage (From [71]). 41
Figure 17. Altium Designer interface with separate schematic, 2-D PCB layout, and 3-D component placement views visible on separate monitors (From [73]). 49
Figure 18. Four-layer stack of the ProASIC3 Test Board with internal power planes and external signal layers. 51
Figure 19. ProASIC3 nano VQ100 package of VQFP-type (From [74]). 52
Figure 20. ProASIC3 Test Board 2-D PCB layout with all component footprints, signal lines, and vias visible. 53
Figure 21. ProASIC3 Test Board 3-D isometric view with component models indicating approximate sizing and spacing of integrated board. 54

Figure 22. FG484 BGA package of the Actel ProASIC3L A3PE600L FPGA (From [75]).....56
Figure 23. Initial 2-D PCB layout of FPGA, three 40-pin I/O headers, prototyping area, and ribbon-cable connector for relay board attachment.....57
Figure 24. Digilent BASYS2 development board counting with proper state logic.....62
Figure 25. Default gate layout for baseline seven-segment LED counter.....62
Figure 26. Triplicated seven-segment LED counter example with voting logic.....64
Figure 27. Triplicated gate layout with voting logic - seven-segment LED counter.....64
Figure 28. FPGA layout view illustrating placement of three counter structures.....65
Figure 29. 4×4-bit array multiplier internal structure using partial product generators and full adders.....66
Figure 30. Full adder gate structure for the 16×16-bit array multiplier, baseline design.....67
Figure 31. FPGA physical layout of default 16×16-bit array multiplier logic.....67
Figure 32. 16×16-bit array multiplier implemented in TMR with voting logic.....68
Figure 33. FPGA physical layout of 16×16-bit array multiplier using TMR logic.....69
Figure 34. Baseline NAND implementation of half adder.....70
Figure 35. Quadded logic version of NAND gate.....70
Figure 36. Quadded logic version of half adder.....71
Figure 37. FPGA physical layout of 16×16-bit array multiplier using quadded logic.....72
Figure 38. TIR logic version of half adder.....73
Figure 39. FPGA physical layout of 16×16-bit array multiplier using TIR logic.....73
Figure 40. FPGA physical layout of 16×16-bit array multiplier using stacked TIR-TMR logic.....74
Figure 41. Timing analysis for worst-case paths in various fault-tolerant implementations.....76
Figure 42. FPGA LUT and slice resource utilization in fault-tolerant implementations.....76
Figure 43. Moore design, clocked synchronous state machine utilizing positive-edge triggered D flip-flops (After [76]).....81
Figure 44. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (fault-free case).....82
Figure 45. Floating-input case of AND gate second input disconnected, simulating break in circuit or disruption in input.....82
Figure 46. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (floating-input case).....83
Figure 47. Second input of AND gate tied to logic ‘0’, simulating constant logic zero or short in circuit to ground plane.....83
Figure 48. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (constant logic ‘0’ case).....84
Figure 49. Second input of AND gate tied to logic ‘1’, simulating constant logic one or short in circuit to positive voltage plane.....84
Figure 50. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (constant logic ‘1’ case).....85
Figure 51. TMR representation of the Moore-type FSM with voting logic.....86

Figure 52. Timing diagram for the Moore-type, TMR version of the FSM developed showing output signal correction regardless of errors (constant logic '1' case). 86
Figure 53. Timing diagram for the Moore-type, TMR version of the FSM developed showing output signal correction regardless of errors (floating-input case)....87
Figure 54. Actel ProASIC3 development board showing sequencer LED testing with TMR logic implemented. 89
Figure 55. Synplify Pro RTL view of instantiated FSM logic. 90
Figure 56. Synplify Pro technology view of instantiated FSM logic. 91
Figure 57. Synplify Pro generated state diagram for FSM logic....92
Figure 58. FlashROM storage of eight bytes defining sequencing timing; the third value (0x2) is highlighted containing decimal value '15.' 95
Figure 59. Interface of Xilinx ISE 14.3 showing schematic layout and design flow.....105
Figure 60. ISE 'Design Summary' window listing FPGA utilization and design flow. .106
Figure 61. Interface of Libero SoC 10.1 showing Verilog HDL and design flow. 108
Figure 62. Each Virtex-5 CLB slice contains four LUTs and four FFs (From [79]). 111
Figure 63. Configuration of low-power ProASIC3 device core VersaTile (From [93]).113
Figure 64. Degradation of propagation delay versus TID for RT ProASIC3 (From [83])....114
Figure 65. Footprint of Molex Micro-Fit 3.0 6-pin header for power supply to FPGA core and I/O bank voltages. 118
Figure 66. Footprint of FlashPro4 JTAG 10-pin header for programmer interface. 118
Figure 67. Schematic view of PA3TB 'Bank 2' connections to USB connector and BIT0 through BIT7 LEDs. 120
Figure 68. Schematic wiring diagram of JTAG interface header to ProASIC3 nano connection points on PA3TB. 121
Figure 69. Manufactured four-layer PCB with gold surface pads and through-hole vias. 123
Figure 70. External camera view of ProASIC3 nano soldering to PCB board as seen through the microscope. 124
Figure 71. All three ProASIC3 Test Boards soldered and integrated – ready for testing. 125
Figure 72. Test bench setup of ProASIC3 Test Board wired to power supply; FlashPro4 interface via USB to desktop computer station and programmer interface....126
Figure 73. Current 2-D layout and 3-D component placement of SADv3 Relay daughter-board with size and shape for proper enclosure fit (From [98]). 138
Figure 74. Eight-layer stack of the SAD Version 3 – Flight Prototype Board with internal power planes and external signal layers. 139
Figure 75. Final fanned-out configuration of the ProASIC3EL FPGA with multiple signal layers and all vias visible....140
Figure 76. Bottom plane pattern of vias within FPGA footprint on eight-layer PCB with room for center-channel power decoupling capacitors. 141
Figure 77. Logic block diagram of Cypress Semiconductor 4-Mbit (256k words × 16-bit) SRAM devices illustrating address and data line use (From [99])....142

Figure 78. Logic block diagram of Numonyx™ 64-Mbit embedded flash-memory devices illustrating address, data, and signal line use (From [107]). 143
Figure 79. TPS79601 adjustable LDO regulator programming schematic for proper specification of R_1 and R_2 values (From [109]). 145
Figure 80. Schematic wiring of SRAM and flash-memory devices with address, control, and data lines as specified by design documents. 146
Figure 81. 40-pin general digital and LVDS headers as placed on the schematic. 146
Figure 82. Location of SRAM and flash-based memory chips relative to BGA FPGA device; JTAG header visible in lower-right corner. 147
Figure 83. Auto-route testing of SRAM and flash-memory connections to ProASIC3EL FPGA across four-layer signal planes. 148
Figure 84. Initial 3-D footprint placement of FPGA and memory devices in isometric view of PCB top plane. 148

LIST OF TABLES

Table 1. Overall hardware comparison between Actel ProASIC3 and Xilinx Virtex-5 series FPGAs illustrating strengths and weaknesses. 45
Table 2. Approximate relationship between different vendors FPGA logic blocks (After [72]).....46
Table 3. Comparison of FPGA development boards used in logic design and testing.....48
Table 4. Characteristics of ProASIC3 nano FPGA (A3PN250) utilized in development of ProASIC3 Test Board (After [74]). 52
Table 5. Characteristics of ProASIC3L BGA (A3PE600L) utilized in development of SAD Version 3 – Flight Prototype Board (After [75]).....56
Table 6. FPGA worst-case path timing for various fault-tolerant designs.....75
Table 7. FPGA resource utilization for various fault-tolerant design methods. 75
Table 8. Synplify Pro generated state table for FSM logic.....92

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

2-D	Two-Dimensional
3-D	Three-Dimensional
AES	Advanced Encryption Standard
AIAA	American Institute of Aeronautics and Astronautics
APIC	Auxiliary Payload Integrating Contractor
ASIC	Application-Specific Integrated Circuit
BGA	Ball Grid Array
BTMR	Block TMR
C	Celsius
CAD	Computer-Aided Drafting
CAM	Computer-Aided Manufacturing
CCC	Clock Conditioning Circuit
CFTP	Configurable Fault-Tolerant Processor
CLB	Configurable Logic Block
cm^2	Square Centimeter(s)
CMOS	Complimentary Metal-Oxide Semiconductor
Co	Cobalt
COTS	Commercial Off-The-Shelf
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
CSEWI	California Space Education and Workforce Institute
DC	Direct Current
DSP	Digital Signal Processing

DTMR	Distributed TMR
EDA	Electronic Design Automation
EELV	Evolved Expendable Launch Vehicle
EEPROM	Electrically Erasable PROM
EHP	Electron-Hole-Pair
EMI	Electromagnetic Interference
ESPA	EELV Secondary Payload Adapter
F	Fahrenheit
FF	Flip-Flop
FPD	Field Programmable Device
FPGA	Field Programmable Gate Array
FROM	Flash ROM
FSM	Finite-State Machine
GHz	Gigahertz
GND	Ground
GTMR	Global TMR
HDL	Hardware Description Language
I/O	Input/Output
IC	Integrated Circuit
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
in^2	Square Inches
IP	Intellectual Property
ISE	Integrated Software Environment
ISim	ISE Simulator

ITAR	International Traffic in Arms Regulations
JTAG	Joint Test Action Group
kbit	Kilobit(s)
kbyte	Kilobyte(s)
kg	Kilogram(s)
km	Kilometer(s)
kΩ	Kilo-Ohm(s)
krad	Kilorad(s)
lb	Pound(s)
LDO	Low-Dropout
LED	Light-Emitting Diode
LFSR	Linear Feedback Shift Register
LTMR	Local TMR
LUT	Lookup Table
LVDS	Low-Voltage Differential Signaling
Mbit	Megabit(s)
Mbyte	Megabyte(s)
MBU	Multiple Bit Upset
MeV	Mega-Electron Volt
µA	Microamp(s)
µF	Microfarad(s)
µm	Micrometer(s)
mg	Milligram(s)
MHz	Megahertz
mi	Mile(s)

mil	Thousandth(s) of an Inch
mm	Millimeter(s)
Mrad	Megarad(s)
NASA	National Aeronautics and Space Administration
NEA	Non-Explosive Actuator
NPS	Naval Postgraduate School
NPSCuL	NPS CubeSat Launcher
NRE	Non-Recurring Engineering
NRO	National Reconnaissance Office
ns	Nanosecond(s)
OTA	Over-The-Air
OUTSat	Operationally Unique Technologies Satellite
P&R	Place and Route
PA3TB	ProASIC3 Test Board
PAL	Programmable Array Logic
Pb	Lead
PCB	Printed Circuit Board
PLA	Programmable Logic Array
PLD	Programmable Logic Device
PLL	Phase-Locked Loop
P-POD	Poly-Picosatellite Orbital Deployer
PROM	Programmable Read-Only Memory
PSRR	Power Supply Rejection Ratio
QFDR	Quadruple Force Decide Redundancy
Rad-Hard	Radiation Hardened

RAM	Random-Access Memory
RFI	Radio Frequency Interference
RHBD	Radiation Hardening by Design
RNG	Random-Number Generator
RTL	Register-Transfer Level
RTOS	Real-Time Operating System
SAD	Splitter Auxiliary Device
SADv3	SAD Version 3 - Flight Prototype Board
SBU	Single Bit Upset
SDK	Software Development Kit
SDR	Software-Defined Radio
SDRAM	Synchronous Dynamic Random-Access Memory
SEE	Single-Event Effect
SEL	Single-Event Latchup
SET	Single-Event Transient
SEU	Single Even Upset
SMT	Surface-Mount Technology
Sn	Tin
SoC	System on Chip
SOI	Silicon on Insulator
SOS	Silicon on Sapphire
SRAM	Static Random-Access Memory
STEP	Standard for the Exchange of Product Data
SWAP	Size, Weight, Area, and Power
TID	Total Ionizing Dose

TIR	Triplicated Interwoven Redundancy
TMR	Triple Modular Redundancy
TSOP	Thin Small-Outline Package
TTL	Transistor-Transistor Logic
USB	Universal Serial Bus
V_{CC}	Positive Voltage Source
V_{DC}	Volt(s) Direct Current
VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuits
VLSI	Very-Large-Scale Integration
VQFP	Very Thin Quad Flat Package
XPA	Xilinx Power Analyzer
XST	Xilinx Synthesis Tool

EXECUTIVE SUMMARY

The research work contained in this thesis documents a means of implementing fault-tolerant logic for use in a sequencer application developed for deploying CubeSats. The sequencer being developed at the Naval Postgraduate School (NPS) in Monterey, CA shall be capable of the timing and accurate launch of multiple CubeSat platforms from a host spacecraft and shall have the capability for quick reconfiguration prior to launch. This sequencer is one particular application of the general payload processor architecture developed in this work. This payload processor architecture is flexible enough to accommodate future needs and applications with few design alterations required.

The NPS CubeSat Launcher (NPSCuL) currently utilizes host spacecraft resources to control the timing and execution of CubeSat deployment through a simple splitter architecture. Previously manufactured Splitter Auxiliary Device (SAD) versions incorporate wiring harnesses or printed circuit board (PCB) traces to accomplish power and signal splitting and recombining. The developed SAD Version 3 – Flight Prototype Board (SADv3) utilizes a field-programmable gate array (FPGA) device to accomplish the sequencing with primary and secondary power provided by the host spacecraft. Although SADv3 expects to get power from the launch vehicle bus, future versions may be designed with a capability for an internal battery power supply. By utilizing various fault-tolerant design techniques, the need for a fully radiation hardened (rad-hard) FPGA can possibly be avoided. This would save a great deal of money and could result in a product capable of missions beyond low Earth orbit (LEO). While LEO is of primary interest, there may be opportunities to send the general payload processor hardware to geosynchronous (GEO) or perhaps even interplanetary orbits.

Several sequencer design methodologies are investigated. Since the sequencer requires only primary or secondary power, a finite-state machine (FSM) architecture is the appropriate choice. To limit the amount of logic required in the general payload processor architecture, a decision to avoid central processing unit (CPU) designs, such as an ARM processor core, was made. The need for embedded processing lends itself well to an FPGA-based implementation. Types of FPGAs compared include antifuse, static

random-access memory (SRAM), and flash-based designs. The hybrid approach of flash-based FPGAs allows for several benefits such as reconfiguration and live-at-power-on capability, with no boot sequence or secondary support components required, along with increased tolerance to certain radiation effects.

The overall goal of this research is to allow for the operation of a processing circuit in the radiation environment of space. Many of the various single-event effects (SEEs) that may affect the operation of the sequencer logic are considered. Of these, single-event upsets (SEUs), single-event transients (SETs), and single-event latch-ups (SELs) are investigated in further detail. These SEEs along with the amount of total ionizing dose (TID) tolerable by the various classes of FPGAs are an important driver to selection of a particular FPGA. Various forms of radiation fault-tolerant methods are considered for use within the sequencer logic—including scrubbing at set intervals, triple modular redundancy (TMR), quadded logic, triplicated interwoven redundancy (TIR), and quadruple force decide redundancy (QFDR). Since radiation hardening by design (RHBD) is the primary focus of this thesis, further investigation into these types of fault-tolerant methods is justified, and the results become the rationale for selection of a particular fault-tolerant implementation.

A comparison between Xilinx and Actel software and hardware products is made to determine the best applicable development suite for design of the sequencer logic. Of the two software products, Xilinx Integrated Software Environment (ISE[1]) 14.3 is the more capable product due to its incorporation of a schematic editor. In addition, the design flow of Xilinx software is more user-friendly and supports rapid re-targeting to different Xilinx FPGA products. Ultimately, the choice of software will depend upon the particular vendor of FPGA product chosen; although, there is a work-around for porting Xilinx schematically designed circuits to Actel instantiations. Since Xilinx devices are primarily SRAM-based, and Actel FPGAs are either antifuse or flash-based, selection of the vendor is primarily dependent on choice of FPGA architecture. In addition to the previously mentioned hybrid advantages of flash-based FPGAs, the Actel ProASIC[2]3

[1] ISE® is a registered trademark of Xilinx, Inc.

[2] ProASIC® is a registered trademark of Microsemi Corporation.

series has the added benefit of pin-to-pin and timing compatibility between military-grade and rad-hard components. Even with the added difficulty of adopting Actel Libero[3] System on Chip (SoC) software over Xilinx ISE, the associated hardware attributes inherent to the ProASIC3 series are beneficial. Therefore, an Actel ProASIC3 is chosen as the FPGA for the sequencer design. Various development boards used in the performance of logic testing included designs from the Xilinx Spartan[4]-3E, Xilinx Virtex[5]-5, and Actel ProASIC3 series FPGAs; these were purchased to allow for all software development platforms to be tested.

Prior to development of the more complex SADv3, a ProASIC3 Test Board (PA3TB) was designed and implemented. This provided significant experience with the Altium Designer[6] electronic design automation (EDA) development environment. The Altium product suite allows for front-end schematic development, physical printed circuit board (PCB) layout, and FPGA hardware design. Testing of the Joint Test Action Group (JTAG) interface was possible on the PA3TB prior to implementation of an identical JTAG connector interface on the SADv3. The PA3TB incorporates a ProASIC3 nano series FPGA using a core voltage of 1.2 V_{DC}. Bank input and output (I/O) voltages are set at 2.5 or 3.3 V_{DC} levels with CubeSat deployment simulated using light-emitting diode (LED) indicators. The four-layer PCB utilized in the PA3TB serves as a precursor to the eight-layer SADv3 design. Additional experience was realized through surface-mount soldering in the construction of the PA3TB. All power supply circuitry for the PA3TB is located off-board for ease of troubleshooting the main FPGA and JTAG design. JTAG programming is accomplished via a Microsemi FlashPro[7]4 USB-to-JTAG external programmer. A standard USB port is also provided on the PA3TB for interfacing with the design and possible future data-logging capability.

---

[3] Libero® is a registered trademark of the Microsemi Corporation.

[4] Spartan® is a registered trademark of Xilinx, Inc.

[5] Virtex® is a registered trademark of Xilinx, Inc.

[6] Altium Designer® is a registered trademark of Altium Limited.

[7] FlashPro is a trademark of the Microsemi Corporation.

The implementation of a ball grid array (BGA) FPGA in the prototype flight-hardware design is a more involved process than the development of the PA3TB. The incorporation of a BGA into the design limits manufacture to companies with the necessary equipment to accomplish the integration of the PCB. To provide for future expandability and features, SRAM and flash-based memory were added to the design. This added memory allows for additional processing features and data storage. Military-grade Actel ProASIC3E/L FPGAs are used in the design of the SADv3; these devices allow for pin-to-pin replacement with a rad-hard Actel RT ProASIC3 FPGA should a mission require more robust hardware. The complexity of the PCB design is markedly increased due to an increased layer count and the requirement to manage the 484 pins of the BGA FPGA. Due to the increased risk of design error, extensive design review is necessary to prevent errors propagating to the final product. The PCB shape is tailored for optimum fit within the SAD enclosure utilized on NPSCuL. The design is a daughter board configuration for pairing with an associated relay board in the SAD box.

In addition to the memory devices, the increase in SADv3 board area allowed for the inclusion of all power supply and power conditioning circuitry on the PCB. The increase in the number of FPGA I/O pins allow for the inclusion of three 40-pin headers for general-purpose use. A breadboard prototyping area is also provided to allow for future capability development using the PCB. An external crystal oscillator is provided for increased timing accuracy and directly connects to the FPGA for utilization in logic designs. Auto-routing of the design proved difficult as the decisions of the auto-router are not always in the best interest of the overall PCB layout. The fan-out features of Altium work well, after a considerable amount of setup time, to properly distribute the signals lines from the BGA FPGA to the four signal layers and four voltage planes. The majority of signal lines on the SADv3 are manually routed.

To test various fault-tolerant techniques for inclusion in the sequencer, different combinational and sequential logic circuits were implemented. These designs were tested in the Xilinx and Actel software environments. An initial test of redundancy techniques is accomplished by constructing a seven-segment LED counter upon a Xilinx Spartan-3E development board. The design is triplicated schematically with code directives to the

synthesis tool. These directives prevent undesired logic removal by the automatic synthesis tool. Proper operation and timing of the design were seen in the default and triplicated cases of logic. A 16×16-bit array multiplier was constructed to further compare the various fault-tolerant schemes implemented. Fault-tolerant techniques are altered by varying the format of the overall design, mechanisms of full adder instantiation, and the inclusion of voter circuits as necessary for operation. A baseline scheme, TMR, quadded logic, triplicated interwoven redundancy (TIR), and stacked Quad-TMR / TIR-TMR fault-tolerant techniques are developed as versions of the array multiplier. In all cases, the outputs of the array multipliers implemented were identical. Although TIR uses the least resources of any fault-tolerant scheme, TMR has the best combination of device utilization and timing delay for logic design. There is future opportunity for research into different TMR schemes using commercial software such as Precision[8] Hi-Rel.

Manual insertion of fault injection techniques leads to the comparison of floating-input, 'stuck-at-0', and 'stuck-at-1' faults. Preliminary research into a method of automatic and random fault injection is conducted, with several methods identified for further inspection. While the baseline FSM produces false timing signals or non-valid signal levels during fault injection runs, the TMR version of the FSM is able to overcome any errors in each of the three fault cases. Timing and resource utilization differences between the baseline and TMR versions of the FSM logic are small. In the future, the additional fault-tolerant techniques that need to be researched are variations in TMR design methods and clocking generation.

Preliminary sequencer logic design is accomplished by developing code and schematics within Xilinx ISE that are subsequently ported to Actel Libero SoC. The initial sequencer logic includes the possibility of setting timing constants either as Verilog parameters or in the FlashROM area of the Actel ProASIC3 FPGA. The logic to be triplicated must be appropriately marked to reduce the possibility of trimming in Libero SoC. The resultant logic design can utilize either case or state switching within

[8] Precision® is a registered trademark of Mentor Graphics.

Verilog to more fully define the operation of the sequencer logic. Schematic design within Xilinx ISE may be avoided once Actel Verilog techniques are fully developed. The switch from the Xilinx to Actel software environment is an important step toward verification of the design and ease of reconfiguration.

Future mission features of the general purpose payload processor are envisioned to incorporate desired logic in a TMR representation. Due to the remaining logic available on the ProASIC3 FPGAs, there are a considerable number of projects which may be researched for inclusion. These may consist of telemetry and data-logging by the FPGA, two-way communication and charging of the CubeSats within the launcher, video feed and recording of CubeSat launches, and potential networking features to the host spacecraft bus – in addition to the already in-progress sequencer development. A complete signal path verification of the current eight-layer, BGA SADv3 should be accomplished prior to manufacture of the board. This will consist of a series of design reviews by multiple engineers. Future testing within a radiation environment, prior to launch, is also an important step in verifying the chosen fault-tolerant technique of TMR.

In conclusion, there are a number of applications for which the ProASIC3 general purpose payload processor is suitable. The sequencer work contained in this thesis is one application in which a significant gain in the state of current CubeSat deployment technology may be realized. The ability to deploy CubeSats using TMR logic should be quite cost effective when compared to custom, rad-hard components, possibly with no loss of fault tolerance. TMR is useful for protecting both combinational and sequential circuits, especially given the available fault-tolerant software toolsets. Future designs using Actel flash-based FPGA technology may have wide applicability due to its configurability and the flexibility with which the I/O pairs can be assigned. The compactness of the ProASIC3 circuit design, when compared to the more traditional Xilinx Virtex series FPGAs, is readily apparent. It is anticipated that this design, or some evolution of it, will be quickly developed into a flight-ready product.

ACKNOWLEDGMENTS

The author would like to thank Dr. Hersch Loomis and Dr. Jim Newman for their time and devotion to this project. Our bi-weekly meetings were instrumental in keeping this project on track and providing insight and suggestions toward the work. There were several times that a perceived roadblock was avoided, due to the consistently excellent mentorship you both provided. Additionally, thank you for the career guidance and inspiration throughout my time at the Naval Postgraduate School.

Additionally, I would like to thank all the personnel present within the Small Satellite Laboratory—specifically David Rigmaiden and Ernesto Yzquierdo. The time you both spent giving helpful tips on learning Altium Designer and methods of soldering under the microscope were very influential to the success of this project. You are both instrumental to the success of the laboratory as a whole and NPS is lucky to have people like you on staff.

Thank you to all the other excellent professors and Space Systems Academic Group personnel who both instructed and guided my time here in Monterey. Many of the classes were excellent and led to methods and tactics used throughout the performance of this research. I honestly feel as if I have an excellent foundation for future success, based on the knowledge obtained in the Space Systems Engineering curriculum.

I would like to thank my fellow classmates in the program—Matt DeMartino, Pete Firenze, Nolan Semrau, Al Perez, Allan Phillips, Jessie Hallan, Kerri Ackman, Marta Savage, Mike Becker, Travis Bateman, Jason Crane, and Maddie Studholme. You have all become great friends and it was wonderful to feel like a team during our many shared classes and design project. Also, thanks to my friends Mike Smith and Rich Ray. I wish you all continued success in the future. Fair winds and following seas!

Most importantly, I would like to thank my family—namely my wife, Jessica, and son, Michael, for the countless time they donated toward allowing me to finish my work and the thesis writing process. You two are my foundation and reason for every decision I make. I love you both, more than you could ever imagine.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE

The purpose of this research is to determine an effective solution for implementing radiation fault tolerance utilizing inexpensive hardware coupled with a robust logic design. The need for fault tolerant processing drives the development of solutions targeted to the space environment, with particular interest in payload processing designs. By careful exploration of various logic implementations, the developed product will be capable of full operation in a radiation environment through redundancy in logic design.

This research led to the development of a hardware and logic implementation suitable for a general payload processor package. The first specific application of the payload processor is a sequencer for CubeSat launching applications. This sequencer is of primary interest because a low-cost, flight-ready sequencer does not exist. Similar commercial products cost upwards of hundreds of thousands of dollars for space-related applications. The primary cost drivers in the selection of these devices tend to be the amount of non-recurring engineering (NRE) development time and manufacturing that must accompany the design. The additional cost of fully radiation hardened components are not even factored in to these designs. The cost of such devices can easily exceed the budget of all but the largest CubeSat[1] projects. Due to the expense of purchasing radiation hardened (rad-hard), commercial off-the-shelf (COTS) technology, an alternative lower-cost solution is desired. It is expected that development of a sequencer will provide a cost-effective solution for government CubeSat launches.

The technology behind fault tolerance in processors as applied to timing and sequencing devices is presented in this thesis, leading to the development of a sequencer. This sequencer is expected to have many of the same characteristics of more expensive radiation hardened products while using less costly commercial and military-grade

[1] CubeSats are nanosatellites. The term "nanosatellite" is applied to a satellite with a mass between 1 kg and 10 kg (2.2 and 22 lbs) [1].

components. The lack of hardware robustness against upsets is compensated for with fault-tolerant techniques in the logic design itself. Various fault-tolerant techniques are explored leading to the selection of the best logic design suitable for the radiation environment of space. This sequencer must be capable of executing a pre-programmed deployment sequence when the launch vehicle initiates power and deploy commands. The CubeSats may be deployed sequentially at specified intervals, or, in some cases, simultaneously.

The need for the sequencer to be re-programmable prior to launch, to have the capability for future logic growth and the redundancy necessary to deal with space radiation upsets, has led to the use of field programmable gate arrays (FPGAs) in the project. Multiple manufacturers and product lines are considered to determine the particular design configuration for the sequencer design. The developed logic designs are synthesized and implemented on commercial development boards to provide testing and experience with chosen FPGAs and secondary components.

Development of the hardware is carried through the research and schematic design phases. Full printed circuit board (PCB) layout is accomplished with techniques and difficulties discussed along the way. Initial test circuitry was assembled to gain familiarity with chosen components prior to final circuit board manufacture. The final circuit board design will be easily expandable for future projects or substitution into designs requiring a general-purpose processing capability in an easily modified form-factor. In all cases, the design will complement work already completed prior to the initiation of the FPGA-based sequencer research. Though the chosen FPGA technology and fault-tolerant techniques apply specifically to deployment sequencer development in this thesis, the implemented methods have broader applications to space-based processors in general.

B. NAVAL POSTGRADUATE SCHOOL CUBESAT LAUNCHER (NPSCUL) BACKGROUND

1. NPSCuL Objective and History

The Naval Postgraduate School (NPS) in Monterey, CA, uses small satellites for focused research projects of national interest. NPS has several established CubeSat projects along with associated lab support. The Small Satellite Laboratory is equipped with capabilities such as a 'Class 10k' cleanroom, thermal vacuum chambers, vibration table, radio frequency interference (RFI) / electromagnetic interference (EMI) testing anechoic chamber, microscope-equipped soldering stations, and a large commercial-scale three-dimensional (3-D) printer. In particular, NPS has recently been involved in the development, integration, and launch of the NPS CubeSat Launcher (NPSCuL). NPSCuL launched on September 13, 2012, as part of the Operationally Unique Technologies Satellite (OUTSat) mission—carrying eleven CubeSats into orbit on NROL-36 [2]. NPSCuL was developed with support of the California Space Education and Workforce Institute (CSEWI) and the CubeSat Program Office at the National Reconnaissance Office (NRO).

2. NPSCuL Physical Design Characteristics

The NPSCuL design uses existing standards such as the Poly-Picosatellite Orbital Deployer (P-POD) from the California Polytechnic University and the EELV Secondary Payload Adapter (ESPA) mechanical interface. As illustrated in Figure 1, NPSCuL is capable of carrying eight P-PODs; each P-POD is capable of deploying up to three 1U CubeSat spacecraft [3]. An ESPA interface ensures compatibility with multiple launch vehicles. NPSCuL itself is a five sided box comprised of a base-plate, four walls, four brackets and the ESPA-compatible, non-separating adapter ring. The non-separating adapter ring could be replaced by a Planetary Systems Mark II Motorized Lightband[2] [4], allowing NPSCuL to be deployed.

[2] MKII Motorized Lightband is a registered trademark of the Planetary Systems Corporation.

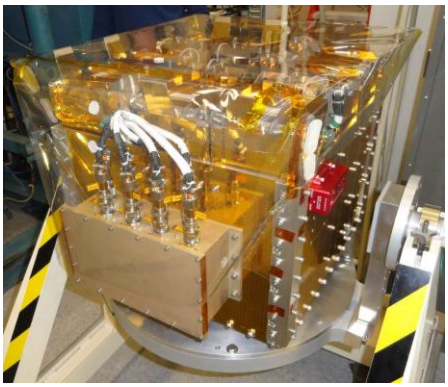


Figure 1. Final payload configuration and integration of NPSCuL with eight P-PODs.

Deployment signals, either directly from the launch vehicle or via an on-board sequencer box, are required to activate the non-explosive actuators (NEAs), open the P-POD doors, and deploy the CubeSats. To support both these scenarios, NPSCuL can be configured as follows:

- If the launch vehicle has adequate resources (i.e., redundant power and signal lines) then it may be outfitted with a Splitter Auxiliary Device (SAD) to accept eight primary and secondary power signals from the launch vehicle and pass them through to the P-PODs. The SAD also has a pass through for lines monitoring the status of the doors on each of the P-PODs [5].
- If the launch vehicle does not have adequate resources (i.e., only a single primary power, secondary power, and data line are available) then a sequencer is required. This device can be powered by the launch vehicle or by an internal battery and must be capable of providing deploy signals and monitoring the status of the P-POD doors switches [5].

The development of the sequencer is the primary concern of this thesis in order to satisfy the limited resources scenario. Such a sequencer will first be useful for missions to low Earth orbit (LEO). Beyond-LEO applications could include deployment of CubeSats in geostationary Earth orbit (GEO) or for use as a payload processor for interplanetary missions in a greatly increased radiation environment. In any of these scenarios, a fault-tolerant design must prevent radiation-induced upsets in logic.

3. History of Sequencer Requirement

Currently, NPSCuL utilizes the host-launcher resources for the timing and execution of the deployment of the individual CubeSats from their P-POD. The SAD takes the eight lines from each of the launcher-provided primary power, secondary power, and data harnesses and splits them out to eight separate harnesses, each containing one primary power line and one secondary power line for redundancy in actuating the P-PODs NEA, and one data line for monitoring the P-POD door microswitches. Each of eight harnesses is connected to one of the P-PODs, as illustrated in Figure 2. The first iteration of the SAD consisted of an internally harnessed, splitter configuration, as shown in Figure 3.

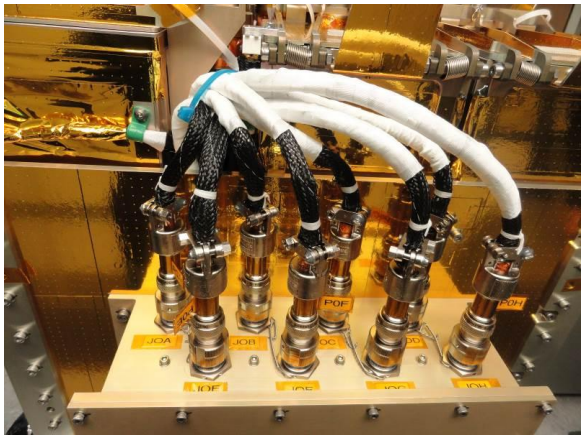


Figure 2. NPSCuL with associated Splitter Auxiliary Device (SAD) and external wiring visible.

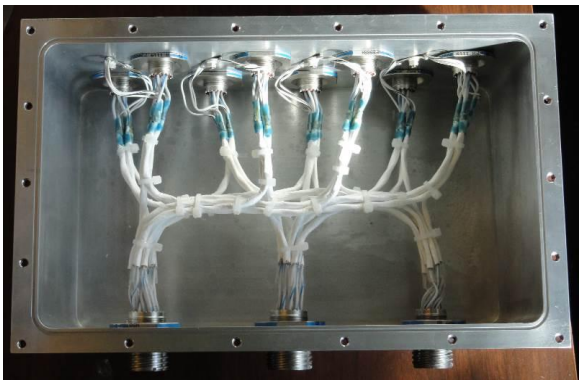


Figure 3. Internal wiring harness structure for SAD depicting power and data line split to eight output ports.

4. Concept of Sequencer Development

The need for a sequencer was realized shortly after the development of NPSCuL was initiated. The design of the sequencer, though simple in principle, should take into account the radiation environment in LEO. There were two approaches envisioned – one, that a radiation and fault-tolerant circuit be designed with the latest in integrated circuit (IC) and manufacturing processes; two, a robust approach involving individual discrete components with feature sizes and voltages that are inherently resistant to radiation induced upsets. This second approach was tailored to industrial parts such as automotive relays and other lower cost components and is the study of a separate, parallel research project. For the purposes of this thesis, only the first approach was considered. The constraints of the fiscal environment precluded the use of the most expensive rad-hard hardware, although, an upgrade path to that technology was considered. Thus, through utilizing lower-cost, commercial and military-grade components, drop-in compatibility of the more expensive rad-hard technology has been maintained. Future processing growth potential was also considered, with emphasis placed on both logic space available and incorporation of subsequent electrical connections.

In modern FPGA applications, especially those dealing with space-based applications, the topic of fault detection and tolerance is often discussed. While there have been a myriad of approaches to sustain the operability of such devices in the past, recent advancements in both FPGA size and application have required updated, or in some cases new, techniques to be developed. Typical methods such as triple modular redundancy (TMR) have been further subdivided into categories based upon application [6]. New methods such as quadruple force decide redundancy (QFDR) [7] are being looked at as a means toward further refinement without the associated overhead of voter circuits. In all of these cases, the aim is to successfully protect the FPGA device from the effects of radiation-induced single-event upsets (SEUs) and multiple-bit upsets (MBUs) which can seriously impact, or even end, a mission.

Various FPGA-specific cost metrics are utilized in comparing the fault-tolerant techniques evaluated. These include device-resource utilization, path-delay timing, ease of implementation, proper logic output, and availability of commercial solutions. The

metrics are assessed after concluding work on a particular fault-tolerant scheme. Logic design software allows for calculation of resource utilization and timing delays. Ease of implementation is more of a subjective realization regarding the time to manually insert the required fault tolerance means. In all cases, the designs are tested for logic accuracy in output under normal operating conditions. Fault injection is used to test for proper logic output in circuit degraded scenarios. Certain fault-tolerant schemes such as TMR are available in many commercial synthesis products; others, like quadded logic, have no commercial implementation and require intense amounts of time and effort to implement. Future testing with commercial, fault-tolerant synthesis software, under a radiation environment, will yield more data for assessing the effectiveness of a particular fault-tolerant technique.

5. Sequencer Requirements and Specifications

For the sequencer application of the general purpose payload processor being developed, there are several requirements that must be considered prior to complete specification of the design and the logic approaches investigated. Specifically, the sequencer will be capable of initiating the door opening of the P-PODs either individually or simultaneously. The application of power, either through primary or secondary power supplies, serves to start the internal timing of the sequencer application. The inputs to the sequencer are minimized, in that application of power serves as both the 'Arm' and 'Deploy' signal necessary to facilitate the deployment of the payload CubeSats. The removal of power to the sequencer device is sufficient to stop the deployment process. The timing sequencer and deployment of CubeSats will be reinitiated by again supplying power to the device.

There are secondary requirements for the CubeSat launch sequencer which are important in the context of both the environment and application involved. These include radiation tolerance, reconfigurability, low-cost, and suitability toward differing applications. Specifically, the functional requirements of this application require that the general purpose payload processor be capable of operation in the radiation environment of space. This requires careful study of the various methods involved in targeting a

design with fault-tolerant aspects and an understanding of the relevant radiation environment. Several different logic designs will be investigated to determine the appropriate technique for fault-tolerance necessary for the sequencer application. Functionally, the design needs to be capable of operation even in the radiation environment of space. This operation should be indistinguishable from the operation of fully radiation-hardened hardware while allowing for an upgrade path to such hardware if desired. Additionally, the design is targeted to produce the same performance of such hardware, without the overhead of associated expense.

The reconfiguration aspect of the design leads to the choice of FPGA technology, with the associated requirement for the sequencer to be adjusted for timing at any point prior to launch when physical access to the device is still possible. Thus, given the requirements for operation and design, the following specifications for the sequencer can be derived:

- The general purpose payload processor (sequencer application) should fit in the same relative size and form-factor as the previous NPSCuL SAD enclosure space.
- Operation of the sequencer should be indistinguishable from that occurring if the launch vehicle had initiated P-POD door openings.
- A significant reduction in the number of incoming signal and power lines is required occur due to on-board logic taking the place of required external interfaces and data lines.
- The sequencer shall be capable of opening the doors of any of the P-PODs individually, or simultaneously, as the application warrants.
- The sequencer shall be capable of timing reconfiguration at any point prior to launch, when physical access to the device is still possible. This capability may be refined in future revisions to allow for remote reconfiguration. (Low Priority)

- Application of primary or secondary power is sufficient to start the deployment sequencer. Likewise, the removal of these power sources is sufficient to halt the deployment process and power down the sequencer.
- The sequencer shall possess the proper fault-tolerant logic constructs to allow for continued operation in the presence of a radiation environment, particularly while in low earth orbit and possibly out to geosynchronous orbit.
- The sequencer shall utilize commercial or military-grade, off-the-shelf hardware whenever possible to lead to an overall lower-cost. Hardware upgrade to radiation hardened versions should be allowed for with minimal re-design effort.
- The sequencer should have sufficient logic overhead to allow future, secondary functions to be added with no change to the hardware design. This will allow for future expansion and/or the application of the hardware to designs other than the sequencer application.

C. RELATED WORK AND DOCUMENTATION

There have been many theses researching various fault-tolerant techniques, along with past development of FPGA-based hardware [8]–[15]. These works are related in their applicability toward the space environment and various developmental approaches toward the design of FPGA logic and circuitry. In particular, the previous thesis material on the development of the Configurable Fault-Tolerant Processor (CFTP) project was a good starting point for the research contained herein. CFTP was a multi-year, multiple-student research project aimed at the orbital flight of an experimental package designed to test for radiation tolerance properties and mitigation techniques in earlier-generation FPGA technology [8]–[15]. The work on CFTP is still on-going with a second flight scheduled to take place in August 2015, with the launch of NPSAT1 [16]. The CFTP design uses SRAM-based FPGAs and provides a comprehensive guideline for the use of TMR within that type of technology. In this thesis, all available FPGA technologies are compared and evaluated for use in the final sequencer implementation.

Differences between this research and earlier CFTP theses include the type of technology utilized, number of secondary component requirements, transition from combinational to sequential logic, method of power-on configuration, and the transition from a test experiment to practical application. CFTP showed that logic redundancy was an important aspect of design when applied to the LEO radiation environment. It is not a question of if, but of how many, events would occur in LEO due to the radiation environment. The experiment was designed to report radiation events and CFTP's response to them so as to benefit future fault-tolerant logic designs and hardware designs. Lessons learned from that research have led to an evolved selection of technology and logic design for this thesis.

D. CONCEPT OF DEVELOPMENT STRATEGY

Goals of this thesis research include the manufacture and integration of an initial sequencer test board, hardware schematic design and preliminary PCB layout of the flight prototype board, and selection of a particular fault tolerance technique for implementation into the sequencer. Since the sequencer is one specific application of the general payload processor design, the board will incorporate features to allow for easy expansion to other mission sets. All devised logic and prototype hardware will require extensive design review prior to board manufacture. In the future it is hoped to test the design at a radiation test facility to determine fault tolerance effectiveness prior to exposure to radiation in the space environment.

To maximize the amount of time available for sequencer design research, including hardware and logic development, concurrent tasks were undertaken in the course of this research. From comparison of FPGA technologies and fault-tolerant schemes, choices are made regarding applicable devices and logic for implementation. After down-selection to a particular product line, a decision was made to quickly manufacture a sequencer test board. This resulted in considerable design experience with the hardware design software prior to implementation of the more complex flight prototype board. Since the lead time for design of a flight prototype board is vastly greater than the sequencer test board, work on the logic design occurred simultaneously.

This allowed for logic error testing of the chosen fault-tolerant schemes at the same time final hardware layout occurred.

E. THESIS ORGANIZATION

In Chapter I, the history of the sequencer requirement and a concept of application have been presented. This research relates to NPSCuL and follow-on missions, with information provided concerning related projects. The concepts of sequencer design, radiation hardening, and fault-tolerance techniques are discussed in Chapter II. A survey of field programmable gate arrays and selection of the particular FPGA and development boards used are presented in Chapter III. The preliminary hardware design of the sequencer test board and modification of the hardware design to a final sequencer logic board are presented in Chapter IV. A comparison of fault-tolerance methods and implementations, fault injection techniques, and preliminary sequencer logic design, is presented in Chapter V. Conclusions and recommendations for follow-on research are presented in Chapter VI. A comparison of vendor products, including existing software suites, hardware solutions and development kits, are contained in Appendix A. All schematics, circuit board layouts, and data sheets for the sequencer test board are contained in Appendix B. All schematics, circuit board layouts, and data sheets for the final sequencer logic board are contained in Appendix C. Documentation of the HDL source coding and test bench setups are contained in Appendix D.

THIS PAGE INTENTIONALLY LEFT BLANK

II. SEQUENCERS, RADIATION HARDENING, AND FAULT-TOLERANCE

A. SEQUENCER DESIGN METHODOLOGY

1. Timing Concerns Specific to Initiating P-POD Deployment

The sequencer design is such that a pre-determined, periodic set of events is triggered in response to the power supplied by the launch vehicle bus. The clocked synchronous state machines within the sequencer must be robust enough to withstand the radiation environment of space, neither permitting premature deployment nor failing to deploy at all. The sequencer must be capable of energizing and de-energizing multiple outputs for variable amounts of time. Additionally, the re-programmable aspect of the sequencer demands that the implementation of the design not be dependent upon the wiring configuration of the overall circuit board or the associated supporting components such as resistors or capacitors. Many commonly used timing circuits are based on technology such as oscillators or 555 timer integrated circuits (ICs) whose timing outputs are fixed once inserted in a certain configuration [17]. These are insufficient for the design of a reconfigurable sequencer, as one should be able to re-program the deployment sequence as long as access to the SAD interface remains available.

There are several options available which would meet the demands of sequencing. The use of a central processing unit (CPU), such as an ARM processor core, running some form of real-time operating system (RTOS) would be applicable to the sequencer development [18]. The resource demands, both in power consumption and protection of the CPU against radiation upset, are limiting factors in the use of this approach toward the sequencer development. Many of the rad-hard designed FPGAs preclude the inclusion of processing units (disabling them in hardware) to prevent these shortcomings in space applications [19].

The other option involves selection between various field programmable devices (FPDs). FPGAs are a type of FPD; ‘field programmable’ refers to configuration capability once installed without disassembly of the circuit. Programmable logic devices

(PLDs) are purely combinational devices while FPGAs provide both combinational and flip-flop logic. PLDs typically consist of two basic types – simple PLDs, which are usually either programmable logic array (PLA) or programmable array logic (PAL[3]), or complex PLDs (CPLDs), which allow more than one logic block as well as configurable interconnections [20]. The configurable PLA, consisting of logical connections in its and-plane and or-plane, is a good analogue for the manner in which FPGAs are configured [20]. Since simple and complex PLDs are purely combinational and lack the memory devices of FPGAs, they are not suitable for implementing the sequencer finite-state machine (FSM). The mechanisms of operation and the physical characteristics of FPGAs are discussed in Chapter III and Appendix A.

2. Chosen Approach

The FSM to be implemented in the sequencer will allow for the power-on signal to provide an 'Arm' situation when the sequencer is first turned on. The sequencer will initiate the 'Deploy' signal from its internal logic to begin the CubeSat deployments. Once the command to deploy has been given, the machine architecture will be fully capable of moving from state to state without additional inputs from the host platform. Should there be a desire to stop the sequence for a contingency, power can simply be cut to the sequencer device, which will place it back in a fail-safe configuration. This may be further reinforced with secondary hardware, such as relays for an added layer of protection against misfire. This approach is in line with the derived requirements specified in Chapter I.

The risk of logic upsets in a microprocessor in a radiation environment leads to the decision to utilize FPGA-based technology within the sequencer. FPGAs are a highly flexible means for implementing the sequencer FSM, where complex sequential timing outputs are necessary. The FPGA-based approach gives more flexibility than an embedded microprocessor and is expandable to perform other mission functions besides sequencing. The various PLDs previously discussed are all of combinational logic design and would require external memory devices to allow sequential operation. This would

[3] PAL is a trademark of Advanced Micro Devices.

greatly add to the complexity of the design and introduce points of failure between the added memory and interfaces. Since FPGA-based technology is capable of many of the functions of a CPU, without the added requirement of an operating system, FPGAs are the logical choice for incorporation into the sequencer. Some background knowledge about FPGAs and their operation is important before down-selection to a particular product.

B. RADIATION HARDENING

Radiation can ionize atoms and disrupt a semiconductor's crystal structure. High-energy neutrons, which are present in the radiation environment of space, arise from the subatomic particles ejected from the Sun and deep space. A neutron that strikes a silicon atom in a semiconductor IC results in the ejection of heavy ions, as seen in Figure 4 [21]. These heavy ions can further lead to device effects, SEEs, as the possibility of firm or soft errors in logic blocks or the routing interconnect may result. Even in today's low-alpha compounds in FPGA packaging materials, alpha particle emission due to uranium and thorium present in molding compounds can present another significant challenge by making it difficult to protect the devices fully from SEEs [21].

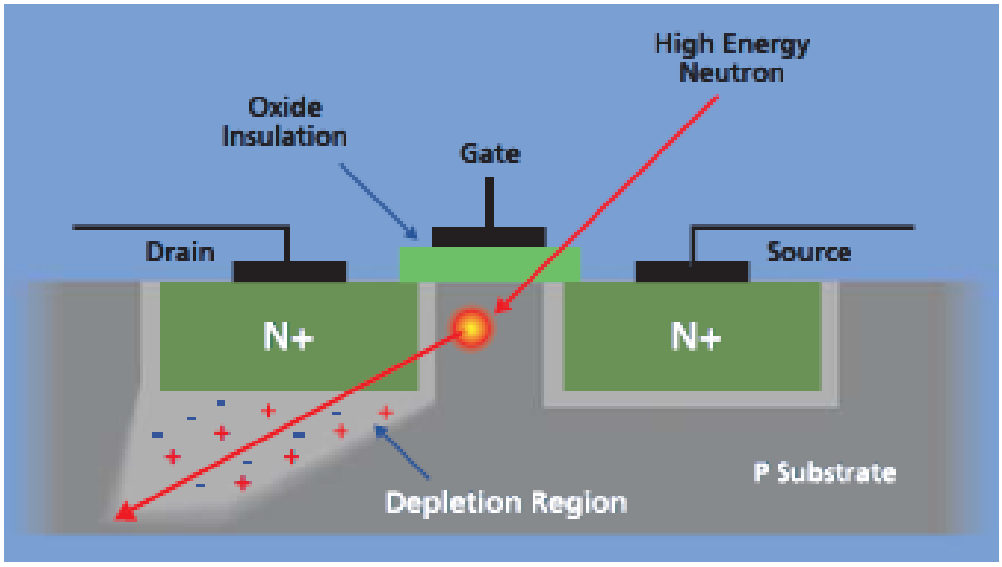


Figure 4. Neutron radiation strike of silicon atom resulting in ejection of heavy ions leading to firm error in transistor (From [21]).

While there has always been a problem of radiation within the space environment, older technologies had the benefit of largely segregated and independent components. The large can-size transistors and low-number integrated circuit counts of the 1960s and 1970s served to protect the computers and equipment placed into space based on their feature sizes alone. As technology-level integration increased, and the number of transistors on ICs grew, the problem of SEEs was seen to manifest itself in ways which were wholly unfamiliar to logic designers of a generation before [22]. New methods of physical shielding and protection from the radiation environment were devised—and differed greatly from those previously developed to combat the strong blast type effects from a nuclear attack. Many of these schemes centered on proper logic operation of the circuit under varying environmental conditions. Such techniques included the use of voter circuits within TMR to correctly provide for functioning of the circuit with erroneous conditions present [23]. TMR has been successfully employed on computers for fault-tolerant behavior since the design of the Saturn V Launch Vehicle Digital Computer in the early 1960s [24].

1. Single-Event Effects (SEEs)

The general term for the effect caused by the interaction of high-energy particles with circuit elements in ICs is the single-event effect (SEE). The charged particles created by the ionization trail along the path of an incoming radiation particle can induce upsets in the states of transistors [25]. These may take the form of logic state changes, current and voltage spikes within the circuit elements, or permanent damage to the transistor itself. In general, SEEs fall into two categories – soft errors and hard errors, as seen in Figure 5. Soft errors are those events that have no damaging effects and are cleared by subsequent device operation, while hard errors are those events resulting in lasting damage [25].

The recoverable soft errors are further categorized into several groups, of which, SEUs (including single-bit errors and MBUs) and single-event transients (SETs) are of the most interest. Likewise, hard errors can be classified into several categories; single-event latch-up (SEL) events are discussed further to provide an example of non-

recoverable errors. Both of these categories of events are of primary interest for the performance of the sequencer.

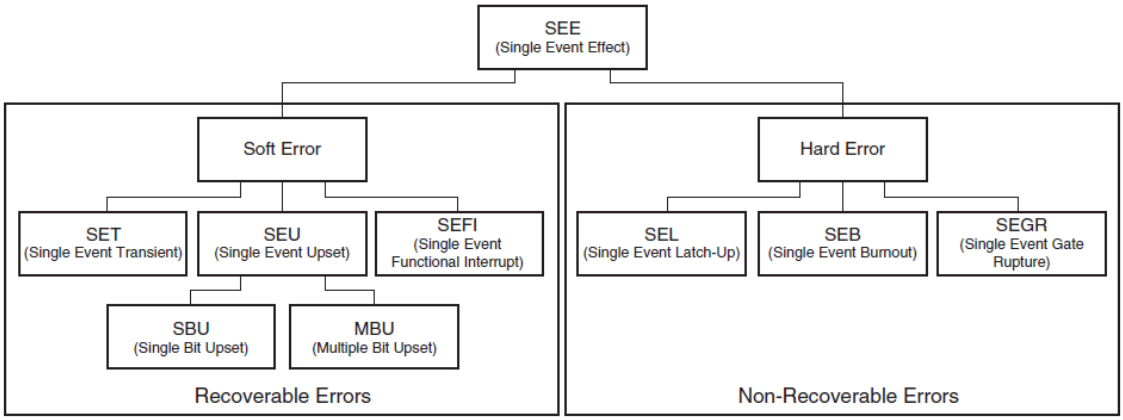


Figure 5. Single-event effects classification diagram illustrating difference between recoverable and non-recoverable errors (From [25]).

a. Single-Event Upsets (SEUs)

Single-event upsets are the result of high-energy particles causing a change in the state of a memory element (SRAM, flash memory, FF, or latch) [25]. These can be further subdivided into single-bit upsets (SBUs) or MBUs. SBUs are by far the most common SEE seen in space-related applications [25]. The results of a SEU can lead to the erroneous output of data, which uncorrected can provide little faith in the correctness of the logic circuit in operation.

MBUs are becoming more of an issue with later generations of hardware ICs, including the most recent revisions of FPGAs. As the transistor count has increased on devices, the associated transistor size has decreased. The probability of a particle crossing an individual bit might decrease, but the charge threshold usually also falls [26]. In addition, the number of bits on a device increases due to the increased transistor count, making the device as a whole more sensitive. Thus, a radiation impact upon the devices lattice structure has more of a potential to create cascading effects in the transistors surrounding the impact zone. This is illustrated in Figure 6, where five additional bit cells are upset due to the generation of an oxygen ion by an incident proton in 130-nm

complimentary metal-oxide semiconductor (CMOS) memory [27]. This may lead to MBU issues which are difficult to correct with legacy techniques. Studies have indicated that in one to five percent of SEU cases, MBUs may also be triggered [28].

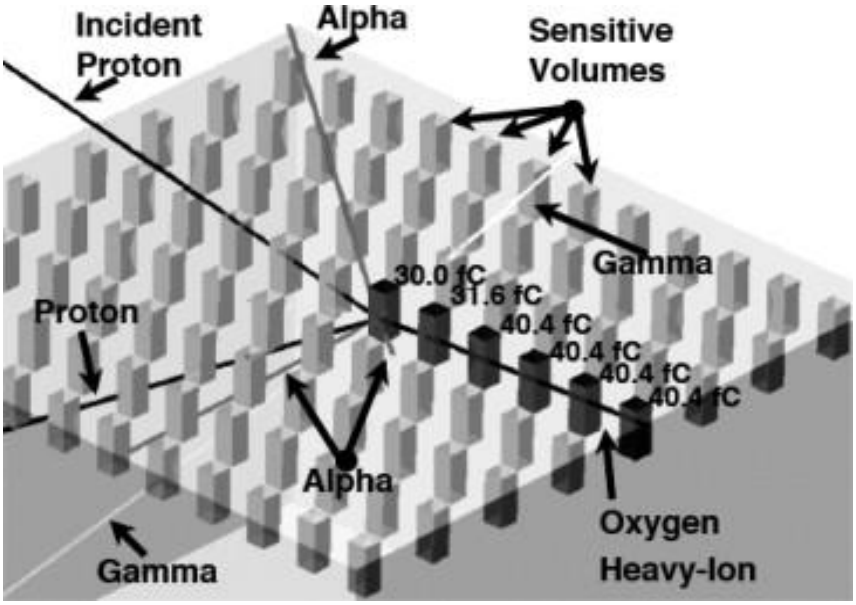


Figure 6. MBU of six bits triggered by proton induced oxygen heavy-ion in 130 nm CMOS memory cells (From [27]).

b. Single-Event Transients (SETs)

SETs result when a high-energy particle impacts a combinatorial path of a device and can induce a voltage or current spike. Although SETs are temporary in nature, they can lead to long-term effects if they trigger undesirable actions such as power resets. If the pulse width of the spike is sufficient and occurs at the right time, it can propagate throughout the circuit [25]. SETs differ from SEUs in affecting combinational rather than sequential logic. An SET may be indiscernible from a SEU if the transient propagates to the input of a FF during the clocking window of the FF. As a result, SETs can become the primary source of observable errors if hardening is incorporated into an FPGA's sequential logic [29].

SETs pose a potentially serious threat to spacecraft as evidenced by system malfunctions in operational missions. These failures include NASA's

TOPEX/Poseidon satellite, TDRS, Cassini-Huygens and others. In most cases, the SETs caused the satellites to switch into a failsafe mode, in which all nonessential systems were temporarily powered down until the cause was identified [30]. For the sequencer, such an event could be detrimental to the deployment sequence, as a power-down event could greatly impact the mission. Although the sequencer could be resumed after a transient event, this would delay the deployment timing, possibly in an unacceptable manner. For this reason, one must look into not only the SEU mitigation techniques for FPGAs but also the SET tolerance.

c. Single-Event Latch-ups (SELs)

A single-event latch-up (SEL) is a type of circuit latch-up induced by radiation [25]. This is one of the most common CMOS IC failures in the space environment and can degrade IC performance and potentially cause permanent failures [31]. Many space systems cannot tolerate even one SEL. Recent studies indicate that SELs may be caused by nuclear recoil interaction of protons with Tungsten plugs commonly used in high-density ICs [31]. These errors may be cleared by power cycling but in many cases may result in permanent damage and may be non-recoverable. For the FPGA-based sequencer, SELs must be avoided at all costs, with priority given to the technology with the greatest inherent resistance to SELs.

2. Total Ionizing Dose (TID)

Gamma ray photoinduced ionization damage is initiated when electron-hole-pairs (EHPs) are generated along the track of secondary electrons emitted via photo-material interactions such as that with silicon. The density of EHPs generated along the tracks of charged particles is proportional to the energy transferred to the target material. The stopping power or linear energy transfer (LET) expresses the energy loss per unit length (dE / dx) of a particle [32]. It is a function of the energy of the gamma ray photon and the target material's density. The total amount of energy deposited by a gamma ray photon that results in EHP production is commonly referred to as total ionizing dose (TID) [32]. The typical unit of TID is the rad, which denotes the energy absorbed per unit mass of a material. One rad is equivalent to 100 ergs absorbed by one gram of the target material

[32]. An excellent background discussion of the physics behind EHP generation leading to accumulated TID is provided in [32] and [33]. The most common method of protection from TID effects is physical shielding of the PCB. This is accomplished by design of the overall enclosure or protective measures around individual components. Other means of protection from TID effects include the manufacture of specifically targeted radiation-tolerant FPGAs, such as the Xilinx Virtex QV space-rated series of devices. Methods to increase the tolerance of these devices include manufacturing on a substrate such as silicon on insulator (SOI) or silicon on sapphire (SOS) and an increase in feature size to aid in resistance to radiation induced failures.

3. Testing of FPGA Hardware in Radiation Environments

Common means of producing various types of radiation for both SEE and TID testing include the use of a nuclear source, commonly cobalt-60 (Co-60), to produce a collimated gamma-ray beam [34]. Methods to generate broad-spectrum neutrons include bombarding a thick beryllium target with energetic deuterons from a cyclotron [35]. This is a highly-efficient process for testing for neutron SEEs and has been employed by light-ion accelerators for many years.

A common metric for measuring the radiation tolerance of an FPGA device is failures-in-time (FIT). One FIT represents a single failure in one billion hours of operation; a system that experiences one failure in 13,158 hours has a failure rate of 1×10^9 / 13,158 = 76,000 FITs. Acceptable FIT rates for commercial applications are fewer than 100, while acceptable FIT rates for system-critical applications (including the sequencer) are fewer than 20 [36]. Acceptable FIT rates for the sequencer could possibly be as high as one failure in one hundred thousand hours, equivalent to one failure in about thirteen years of run time. This corresponds to 10^4 FITs, producing a success rate of 99.999% or better for the sequencer's roughly one hour of operation.

4. Radiation Protection Approaches

In looking to safeguard a system against the environment of space, multiple solutions are available to protect electronic systems from radiation in particular. These may be used independently or in combination to provide an optimum solution for the

radiation environment. Commonly utilized techniques include physical shielding of the device or radiation avoidance by mission design [37]. Since the CubeSat Ride Share missions are currently relegated to LEO, the radiation in the outer toroid of the Van Allen radiation belts are beyond the altitude of the mission. The South Atlantic Anomaly, due to asymmetry of the inner Van Allen belt, is closest to the Earth's surface at 200 km (124 mi) in altitude and presents the majority of the radiation received in LEO [38]. These two radiation belts are illustrated in Figure 7. Thus, radiation avoidance by mission design will be somewhat limited based on the planned orbit of the host spacecraft prior to deployment. For future missions extending beyond LEO, the radiation environment of GEO and deep space can be expected to further challenge the sequencer design.

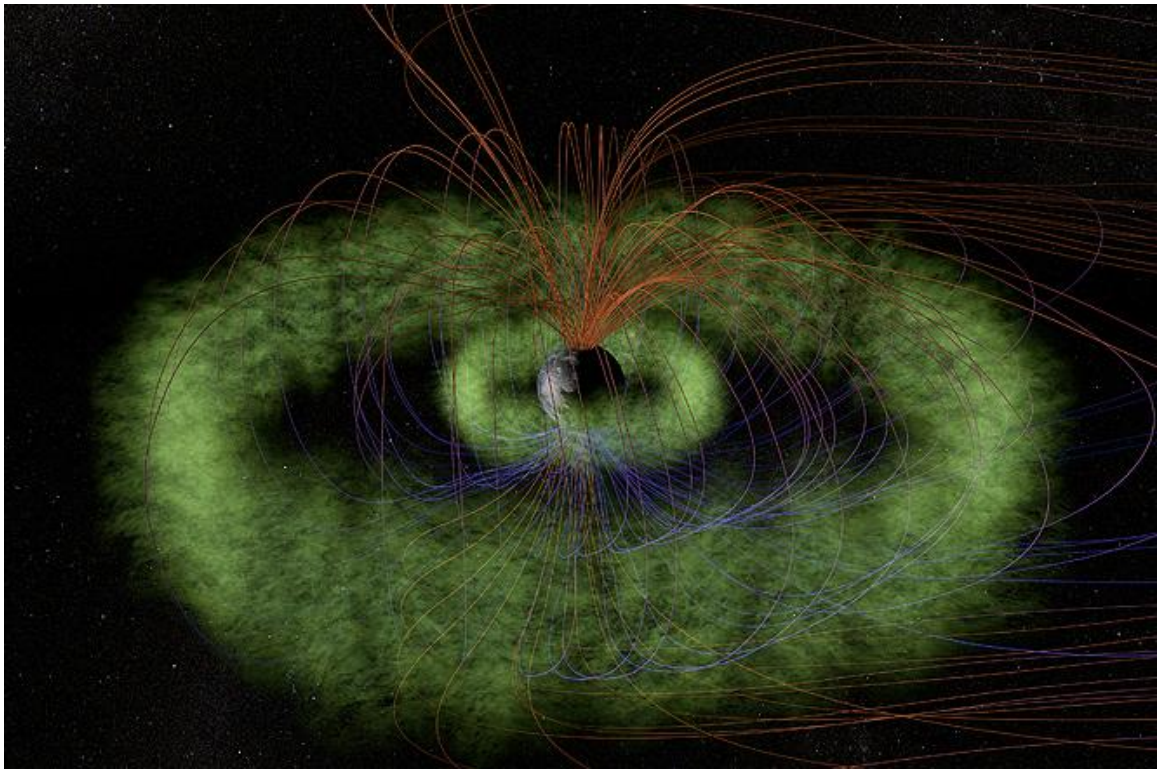


Figure 7. Inner and outer torus-shaped layers of plasma of the Van Allen radiation belts. Energetic electrons form the outer belt; the inner belt consists of protons and electrons (From [39]).

Other radiation protection approaches are beyond the scope of the sequencer project. These protection approaches include radiation hardening by process which

employs specific materials and non-conventional processing techniques and is usually performed on dedicated rad-hard fabrication lines [37]. The protection approaches which are under design control include radiation hardening by architecture and radiation hardening by design. The radiation hardening by architecture technique typically incorporates redundant configurations and component selection. This can be implemented at the component, board, subsystem, or spacecraft level [37]. Some of this will be utilized in the development of the FPGA-based sequencer; the overall architecture of the developed PCB will incorporate as many duplicative protection features to support the FPGA as possible.

Certain forms of radiation faults are protected against, depending on the level of protection required, by selecting the appropriate device line within a vendor's product offering. Devices are often classified as radiation-tolerant when they offer limited resistance to SEL, SEB, and TID effects. Some resistance to these effects is afforded in the commercial and military-grade lines of FPGAs depending on the vendor and production processes. In general, commercial grade devices provide almost no radiation-tolerant properties and are rarely suitable for critical aerospace applications. Certain military-grade level FPGAs provide limited radiation-tolerant properties and can be used with varying degrees of success for critical aerospace applications when combined with other fault-tolerant logic methods. Typically, these military-grade components are used as prototyping devices before purchase of the more robust radiation-hardened FPGAs.

In contrast, the full radiation-hardened devices typically provide much better resistance to SEU and SET effects. This resistance is due to the production processes inherent in the design and manufacture of the FPGAs. The logic cells within the devices are designed with a larger feature size and replicated in hardware to allow for the greatest level of resistance to SEEs as a whole. Coupled with the radiation tolerance afforded by the design of internal fault-tolerant logic, these physical hardware differences combined with fault-tolerant logic development are called radiation hardening by design (RHBD).

The majority of this thesis research considers the protection approach afforded in RHBD. A portion of this technique incorporates fault-tolerant logic design techniques within the FPGA layout to overcome many of the SEU and SET challenges in the space

environment, as covered in Section C of this chapter. After introduction to the techniques in this chapter, the various schemes will be implemented in logic and tested further in Chapter V. Additionally, error correcting memory, parity bits, and watchdog timers are often utilized to provide even more protection. In Appendix A, TID tolerances provided by certain vendors and types of FPGAs are explored with a final selection based upon the best combination of hard and soft fault tolerances.

C. FAULT TOLERANCE

Fault tolerance is the ability of a system to continue to perform its tasks after the occurrence of faults [40]. The ultimate goal of fault-tolerant design is to prevent system failure from occurring. Various requirements satisfied by the introduction of fault tolerance to a system include: dependability, reliability, availability, safety, performance, maintainability, and testability [40].

1. Scrubbing / Re-programming at Intervals

Within the FPGA, the internal logic matrix most vulnerable to SEU effects is the configuration memory. It is this portion of the device that provides the configurable Boolean constructs and memory for operation of the developed FPGA logic. Depending upon the area of the configuration memory affected by a SBU or MBU, the device may have no indications of erroneous output other than transient invalid data being passed. In the worst cases, the FPGA may be altered in such a manner that consistently wrong outputs are generated, or the FPGA may cease to function altogether. Further discussion of FPGA physical configuration memory construction is provided in Chapter III.A.

One of the most common and easiest ways to safeguard an FPGA's configuration memory from SEUs involves the use of scrubbing. To 'scrub' an FPGA, as it is known, involves the complete re-flashing or re-programming of the device at a specified interval to mitigate or prevent the effects of SEUs upon the device [41]. Unfortunately, this requires either that the rate be such that it will not affect the functionality of the FPGA or that it be done regionally, in real time, so that the portion of the FPGA with the upset is reconfigured while other sections continue work. This partial reconfiguration capability is important in real-time operational scenarios. In the case of digital signal processing

(DSP) functions, the complete halt of a processing capability to allow for complete reconfiguration would be unacceptable. Thus, a capability to partially reconfigure based upon the partitioning scheme and fault-tolerance method implemented is desired. If errors are detected in a portion of logic, that section of logic should provide an event trigger to cause partial reconfiguration of the specific partition. This can be implemented via an error detection line with a counter technique to determine repeated errors. Both of these reconfiguration approaches may greatly cut down on the allowable resources of the device, or the quantity of time in which it is fully utilized – therefore, their use should be carefully evaluated before implementation.

2. Triple Modular Redundancy (TMR)

In simplest terms, TMR involves triplicating the logic functioning of the device (in the critical path sections) and including a set or series of voter circuits to determine majority output for proper operation [42]. In majority voting, the best two of three wins the vote and is considered the correct output. Unfortunately, if there is an error in the voter circuits themselves or the output path, then the voting scheme can lead to an overall logic failure. Triplicating voters guards against this type of failure. As such, the voters should be designed with sufficient logic to detect errors when compared to the other voting logic to trigger complete or partial reconfiguration of that portion in the TMR scheme. To better design for the environment of space, and to reflect the increased probabilities of SEUs and MBUs in newer FPGA series, new methods of TMR have been developed [43].

a. Block TMR (BTMR)

The oldest form of TMR is known as Block TMR (BTMR), whereby the three levels of replicated logic are majority voted in one voter circuit [43]. Signal tracing between the triplicated logic outputs and the output of the voter circuit is useful in eliminating the voter as a suspect error source. By limiting the voter circuits to one copy, there is an overhead circuitry requirement to continuously monitor triplicated logic outputs in order to provide this troubleshooting method. Additionally, if one of the triplicated logic circuits should fail completely, then the ability to correct errors in any of

the remaining logic is lost. When coupled with the ever-present radiation environment of space, this leads to a highly increased probability of total failure for a mission.

b. Local TMR (LTMR)

As depicted in Figure 8, an improvement over BTMR is Local TMR (LTMR). In this case, the redundancy and triplication is done only at the Flip-Flop (FF) level where the results of the voted logic are transferred back to the FFs in feedback paths [43]. This has the advantage of masking and correction and greatly increases the overall reliability of the circuit in question. Unfortunately, the problems of clock skew and reset lines are not handled, and Single-Event Transients (SETs), which can lead to SEUs are still a possibility.

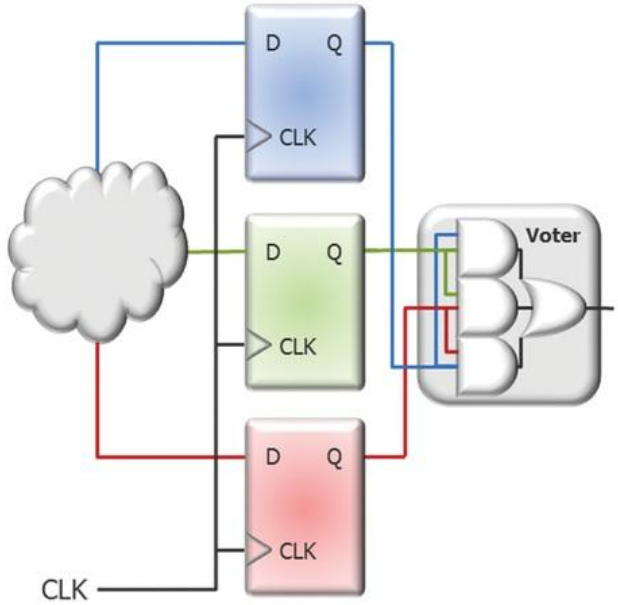


Figure 8. Local TMR with all sequential elements tripled and followed by a majority voter (From [44]).

c. Global TMR (GTMR)

As illustrated in Figure 9, Global TMR (GTMR) has the largest overhead of FPGA utilization and is also the most complex. In it, all sections of the FPGA are triplicated – three separate clocks with three separate domains [43]. The upset rate in

such a circuit is very low. This would be at first glance the most advantageous; however it is the most power hungry and wasteful of FPGA resources. It can be difficult to implement if large object libraries are used and difficult to verify and test such large designs. Additionally, the problem of clock-skew and mismatch between the three disparate clocks must be considered in the design methodology [43].

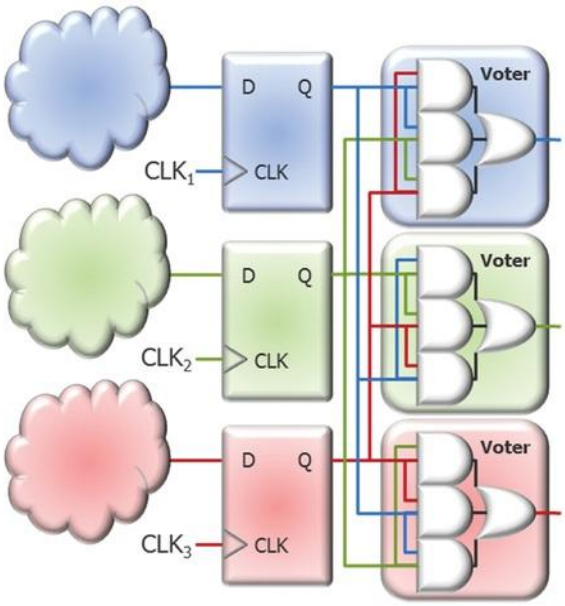


Figure 9. Global TMR with all sequential elements, combinatorial logic, majority voters, clocking signals and global buffers triplicated (From [44]).

d. Distributed TMR (DTMR)

Lying somewhere in the middle of all these approaches is Distributed TMR (DTMR) – while similar to GTMR in that all logic and sequential elements are triplicated – the clock and resets are not [43]. That is to say, there is only one clocking and reset path. This is illustrated in Figure 10. This is seen to have many of the advantages of GTMR without the clock skew and mismatch concerns of the former. The clock line is still susceptible to effects. Often though, clock drivers are sufficiently less susceptible to SEEs. In all of the TMR techniques, the problem of verification of all nodes is seen to exist and becomes more challenging as feature sizes shrink and logic usage increases [43].

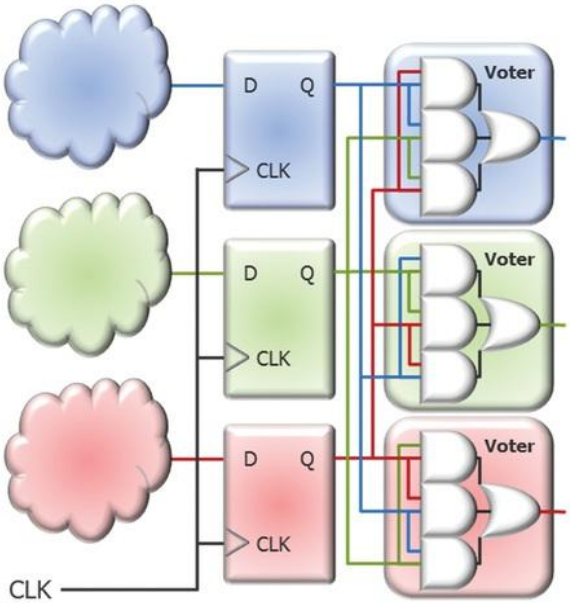


Figure 10. Distributed TMR with all sequential elements, combinatorial elements and majority voters triplicated. Clock signal is shared among resources (From [44]).

3. Quadded Logic

In quadded logic, all involved gates are replicated four times. Errors are corrected soon after they occur, since neighboring good values correct faulty values [45]. Quadded logic does not use voting schemes to determine error values and positions. Rather, it masks errors using logic gates with a particular interconnection pattern.

As illustrated in Figure 11, Tryon conceived of a notion where an error may initially spread through the first series of logic gates but would be corrected further down the circuit. It is in this way that quadded logic can be thought of as a 'self-healing' circuit. The absence of voters occurs due to the fact that the logic values at the outputs should all have identical values by the time they reach that stage. The only conceivable failure point would occur should one of the last stage logic gates provide the wrong output due to error in that stage. For that reason, and the desire to recombine the quadded logic signals back into a singular value, voters or other recombination techniques are sometimes reintroduced to provide both of these desired properties. One difficulty with quadded logic is the lack of an error signal to trigger reconfiguration. Periodic scrubbing is necessary.

The interconnection patterns between gates are chosen such that the same pattern is not encountered twice consecutively. This is done in the cases when going from an AND gate to an OR gate or vice versa, as seen in Figure 11. There is an exception to this rule when connecting like gates. When driving an AND gate by an AND gate or an OR gate by an OR gate, the connecting patterns to the first and second identical gates must also be identical. This is not the case, however, when connecting NAND or NOR gates. In these cases, like NAND or NOR, repeating gates should have differing interconnection patterns.

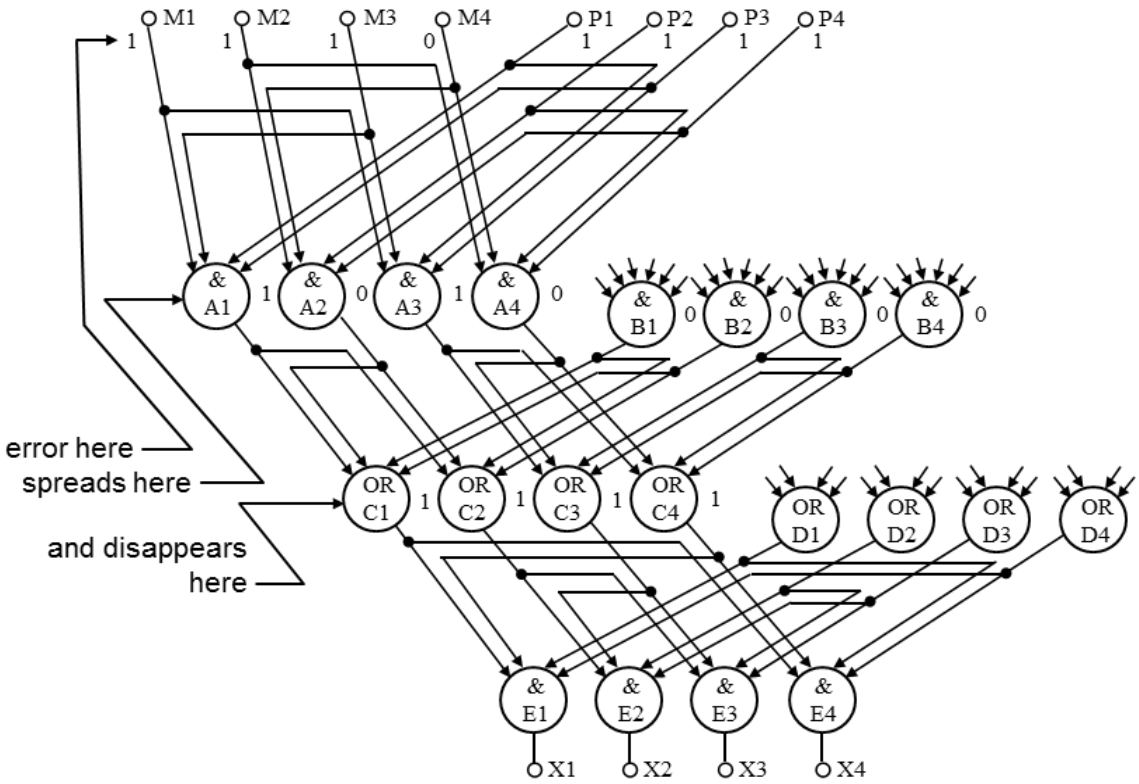


Figure 11. Quadded logic illustrating error propagation and masking (From [46]).

4. Triplicated Interwoven Redundancy (TIR)

As the simplest form of random interwoven redundancy, triplicated interwoven redundancy (TIR) exists as a useful comparison to the techniques of TMR and quadded

logic. There are some of the interwoven benefits of quadded logic combined with the triplicated nature of TMR. Similar to TMR designs, TIR requires a voter at the outputs as a restoring device. The design of a TIR half-adder, without the associated voting logic, is represented in Figure 12.

The general procedure for constructing a TIR circuit involves starting with the non-redundant form of the circuit, triplicating each gate, and then using the interconnection pattern of the non-redundant circuit – randomly selecting a gate from each triplet pair to use as an input for a gate that has no other inputs from the same triple. That is to say, the output of a gate in a pair of triplets is paired with the output of a gate in another pair of triplets to form the inputs of a gate in the next stage. This last step is repeated until all of the gates are connected in the TIR circuit. The method would be identical to TMR if the connection pattern utilized repeating inputs from the same triplet. Thus, TIR is a generalization of TMR to allow for random interconnections.

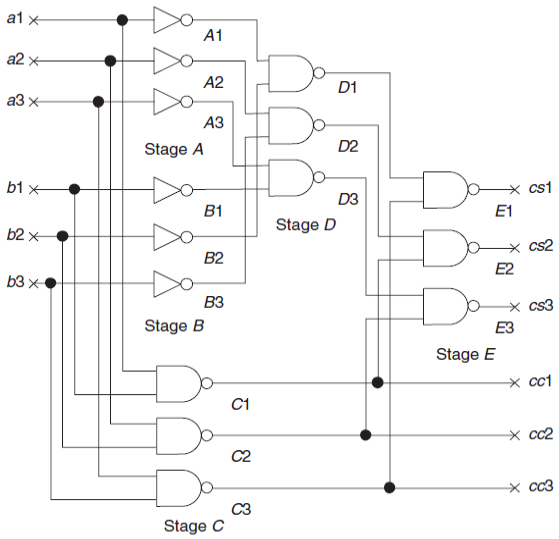


Figure 12. Design of TIR half-adder implementation illustrating interwoven connections (From [47]).

Previous work and simulations have shown that the reliability of TIR circuits with random interconnections are comparable to their TMR equivalent circuits [47]. For certain interconnection patterns, the TIR structure would present weaker performance

than TMR where a single error's effect in a circuit is not confined only to one set of outputs. The interconnection pattern used for experimental testing in the thesis was selected to alleviate much of these concerns. As with quadded logic, no error detection signal is provided, so periodic scrubbing is necessary.

5. Quadruple Force Decide Redundancy (QFDR)

Much of past logic research has dealt with the concept of quadded logic as it applies in the historical context of interest—dealing with AND and OR gates. Quadruple Force Decide Redundancy (QFDR) takes this idea further by applying it to larger general Boolean constructs—namely, the LUTs and FFs present on FPGA devices, as illustrated in Figure 13. The differences between quadded logic and QFDR allow for the FPGA resources to be allocated manually in a more efficient manner than allowing synthesis to auto-allocate resources. By specifying both the number and attachment point of the individual LUTs and FFs, one has greater control of the synthesis effort. This is accomplished by taking the output of the map phase and manually making edits to the layout before the place and route phase occurs [48]. This can be a very time intensive application of fault-tolerance. There are some software tools to aid in this approach; although, none are fully autonomous.

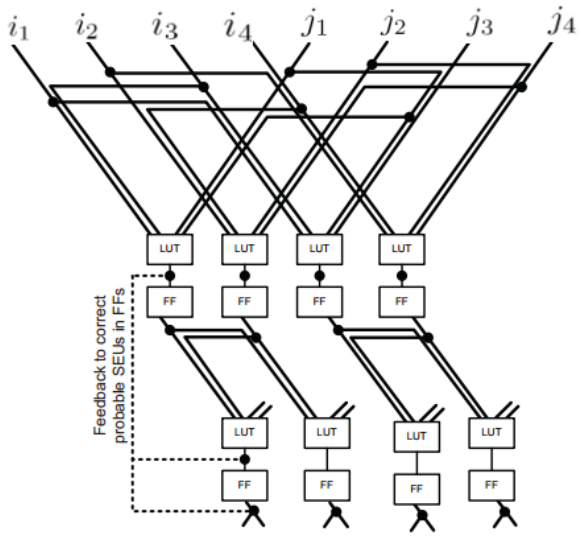


Figure 13. QFDR substituting LUTs and FFs constructs for logic gates (From [48]).

6. Advantages and Disadvantages of Various Fault-Tolerant Techniques

There are many inherent advantages and disadvantages of the above techniques when implemented in logic. Scrubbing of an FPGA usually guarantees a clean working state of the device; however, it typically incurs large time penalties. These can be somewhat mitigated if partial reconfiguration is applicable to a certain device. TMR has the advantage of a long history of application and testing in the space environment. While the utilization requirements for the circuit are slightly more than three times the amount of logic when compared to a non-TMR design, the timing requirements for the design should only incur a modest penalty due to the addition of the voter circuit. The various subsets of TMR will be further studied to determine the appropriate instantiation for the particular type of circuit in development.

Quadded logic and QFDR techniques have the disadvantage of utilizing even more logic resources than TMR. The elimination of a required voter circuit may be an advantage since that eliminates a single point of failure from the design, as the implementations mask any faults by the time the logic values reach the output state. Another advantage may come from increased MBU resiliency. Overall, both quadded logic and QFDR appear to be viable techniques; although, the automated methods of implementation have yet to be developed. TMR insertion using software such as Xilinx TMRTool[4] or Mentor Graphics Precision[5] Hi-Rel remains the only automated means for fault-tolerance insertion into existing logic designs [49], [50]. These logic design software packages are discussed in Chapter V.

TIR exists as a hybrid approach between TMR and quadded logic, with the TMR design being a special case of a TIR solution. An inherent disadvantage is the uncertainty in the proper output unless the proper interwoven pattern is chosen. Timing conclusions for TIR, as compared to the TMR-case, remain to be seen. The suitability of the various fault-tolerant techniques compared in this study is assessed fully in Chapter V.

---

[4] TMRTool® is a registered trademark of Xilinx, Inc.

[5] Precision® Hi-Rel is a registered trademark of Mentor Graphics Corporation.

D. CHAPTER SUMMARY

The concepts of sequencer design, radiation hardening and fault tolerance were discussed in Chapter II. An FSM logic design was chosen for its suitability in the sequencer. Considerations unique to the CubeSat launching application are contributing factors to the FSM approach. Different types of programmable logic devices were analyzed with the FPGA being chosen as the most applicable hardware type. Trends in FPGA development were discussed with an emphasis placed on the evolution of the technology into different subcategories of underlying architecture.

Background information regarding radiation effects in semiconductors was provided with a full description of recoverable versus non-recoverable errors. SEEs are subdivided into categories of SEUs, SETs, and SELs with an emphasis placed on those events which could impact the FPGA-based CubeSat deployer in a LEO radiation environment. TID resiliency is seen as another common metric for comparison between FPGA vendors' products with the means of conducting testing for SEEs and TID introduced. RHBD is considered the focus of research within this thesis.

Various means of fault-tolerant logic implementation were discussed with a breakdown of particular approaches. The particular advantages and disadvantages of scrubbing, TMR, quadded logic, TIR, and QFDR were presented. Initially, TMR is seen as the preferred choice due to software availability and a long history of use in space applications.

III. FIELD PROGRAMMABLE GATE ARRAYS AND SELECTION OF SEQUENCER TECHNOLOGY

A. FIELD PROGRAMMABLE GATE ARRAYS

The field programmable gate array is an IC designed to be configured by the end-user after manufacture of the chip is completed [51]. The configuration process is aided by the use of various hardware description languages (HDLs) including VHDL, Verilog and schematic design. These HDLs are translated to actual FPGA layout configuration via various logic computer-aided drafting (CAD) software suites. These CAD software tools are usually vendor specific; although, generic toolsets are available. FPGAs have largely superseded custom application-specific integrated circuits (ASICs) as the method of choice for rapid prototyping of logic circuits [52]. Though custom ASICs are still often designed for large-quantity or application critical situations, FPGAs are rapidly becoming the primary device all the way to manufacture. This is largely due to the low-cost of FPGAs relative to the development costs inherent in ASIC production [52]. When ASICs are still required, the FPGA is an important tool in prototyping logic circuits that will eventually be laid out during very-large-scale integration (VLSI) logic design. There are even FPGAs onboard the most recent Mars rover, Curiosity [53].

1. Physical Construction

The typical FPGA is internally configured as a matrix of logic blocks, input / output blocks (I/O), and programmable interconnects [54], as illustrated in Figure 14. In all cases, the internal configuration is programmable and, in many FPGA hardware types, re-programmable. This reconfiguration may take place as a wholesale re-programming of the device or a partial reconfiguration of certain sections of the FPGA. The partial reconfiguration capabilities of a particular FPGA device are dependent on the vendor and particular series of device utilized. In all cases, the particular reconfiguration characteristics of a device should be fully understood to allow for a combination of partial reconfiguration simultaneously with other fault-tolerant techniques.

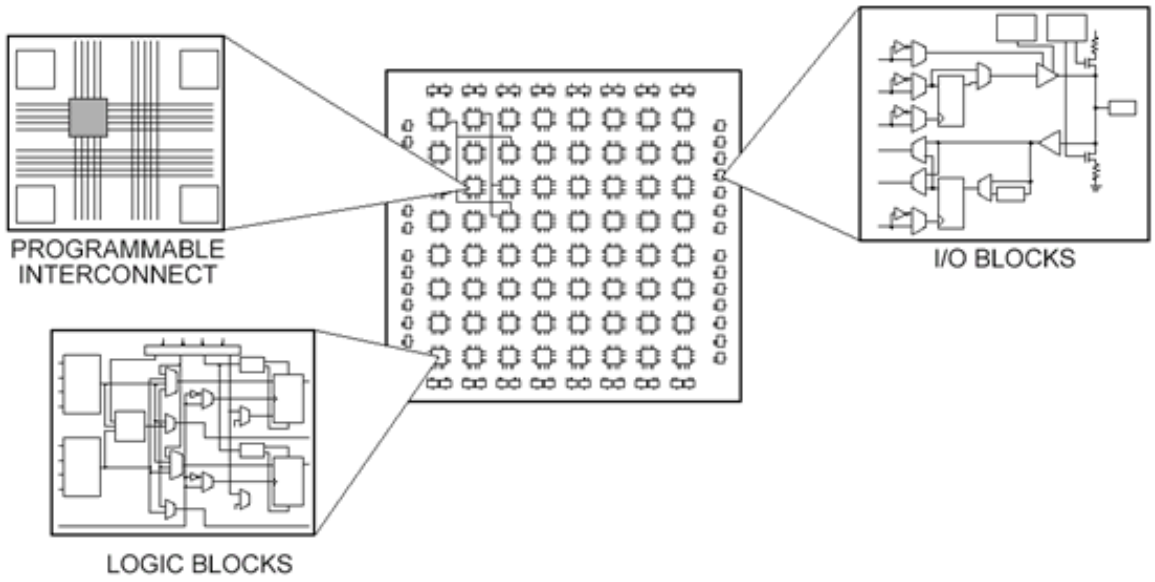


Figure 14. Typical FPGA internal architecture consisting of configurable logic blocks, I/O blocks and programmable interconnects (From [54]).

a. Internal Fabric

FPGAs are usually structured in a two-dimensional array with corridors dividing the rows and the columns used for global interconnects between the logic cells of the array [20]. Each cell consists of a combination of logic functions and flip-flops that can be programmed to perform a desired function. Logic functions are often implemented as lookup tables (LUTs) since they are basically small, programmable random-access memory (RAM). Newer FPGAs may contain more sophisticated building blocks such as adders and RAM. RAM blocks can be further used to build register files [20] as shown in Figure 15. Together these cell clusters of LUTs, gates, adders and RAM blocks can be termed configurable logic blocks (CLBs) or 'slices.' FPGAs that contain 16 and 32-bit CPU cores can often be found in commercial use; however, the use of FPGAs with built-in CPU cores are extremely limited in space-based applications. This is due to the previously mentioned difficulties with radiation and power consumption.

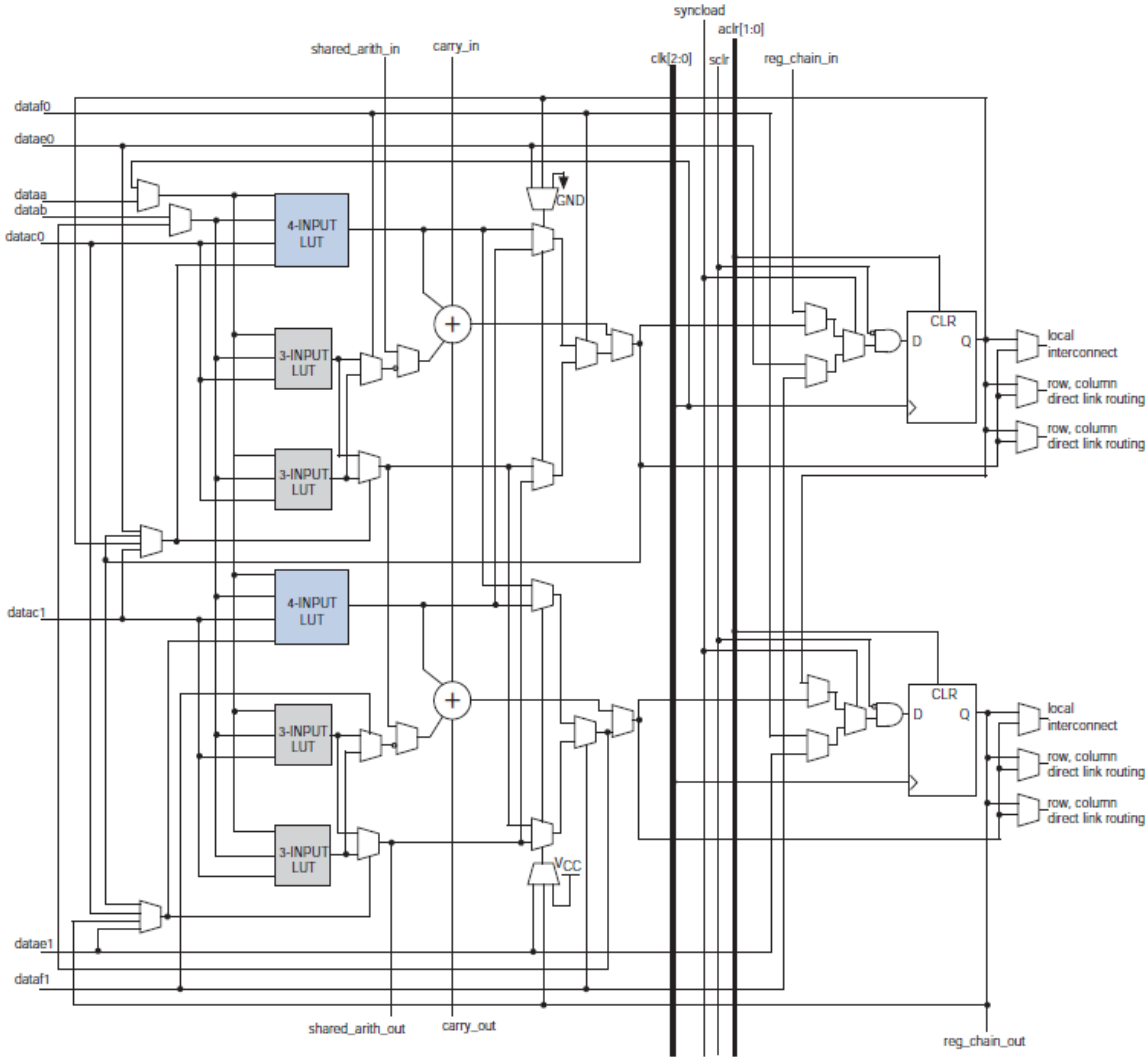


Figure 15. Typical FPGA logic cells with associated LUTs, gates, adders, memory devices, and interconnect buffers visible (From [55]).

In addition to the ability to program individual cells for various functions, the interconnections between cells are also programmable. This allows FPGAs with hundreds of blocks and hundreds of thousands of gates to be utilized for complex logic functions. Interconnect is a major challenge in custom chips—this also holds true in the case of FPGAs. In many FPGAs, 90% of the area is reserved for the interconnection matrix and only 10% is dedicated toward logic and memory blocks [20]. The perimeter of the FPGA is typically surrounded by I/O blocks which are also programmable. On higher density devices, these I/O blocks can also be found running parallel to the corridors

onboard the device, allowing for connection to ball grid array (BGA) device pins [56]. These I/O blocks allow for various data lines to be routed to the appropriate pins of the device with reconfiguration possible at a later point. I/O blocks provide buffering and the appropriate voltage and current levels necessary to drive externally interfaced components at the signal level required. Often, these I/O blocks are grouped into I/O banks with differing voltage levels possible for various bank clusters.

b. Programmability and Reconfiguration

In general, the method of programming FPGAs involves writing HDL such as Verilog or VHDL and is known as configuration. Schematic diagrams may also be developed to do the same function. In most cases, the schematic designed is translated into an intermediate HDL format through the programming interface. This allows one to create a schematic representation of a logic circuit and later instantiate it through a code representation. After synthesis of the HDL code, the developed design is ready for translation, mapping, and place and route phases of implementation. It is at this point that a programming file can be generated for construction of the logic upon the desired device. The actual configuration of FPGAs takes place when the bitstream format of the programming file is loaded onto the device. All of the logic block, I/O block, and interconnect logic is configured on the device at this point in the development cycle. This configuration cycle may be repeated at any point in the future, dependent upon the nature of the technology with which the FPGA is constructed.

Arguably, the largest benefit stemming from the development and rapid adoption of FPGA technology is the short time-span in which reconfiguration of devices is possible. Akin to a software upgrade within a computer, or firmware upgrade within a flash memory device, the re-programmable aspect of FPGAs allows for quick and easy maintenance of hardware logic. This ‘field programmable’ capability changes much of the traditionally held notion that once hardware was deployed, one would have to replace that hardware in order for capabilities upgrade or bug corrections. Many of the current approaches to FPGA reconfiguration involve “over-the-air” (OTA) re-programming, often at a remote location from the actual physical location of the FPGA device [57].

Partial reconfiguration allows an FPGA to implement many concurrent functions and to change those functions while the system is running [58]. This is done by partitioning the device into several regions which may be re-programmed without impacting the other sections of the device. This dynamic reconfiguration during runtime is the real advantage of partial reconfiguration. Each vendor's FPGAs have significant differences in the partial reconfiguration technique, advantages and perceived benefits due to the close coupling of the underlying FPGA fabric with the ability to partially reconfigure [59]. One of the largest benefits for partial reconfiguration would come in communications—the field of software-defined radios (SDRs). The partial reconfiguration of such a device, to allow for more modes or channels in specific frequency ranges, while the radio continues to operate on other areas of the FPGA, is extremely attractive.

Applied to the CubeSat deployer, such a reconfiguration technique could be utilized to reconfigure the sequencer portion of the FPGA while another region of the device is still operating. This would be useful in future scenarios where expected feature upgrades to the sequencer will allow for multiple secondary functions. These functions could include such features as remote CubeSat monitoring via telemetry, video feed of the launch sequence occurring, and overall monitoring and data collection for NPSCuL. The ability to execute these mission areas simultaneously while still allowing for remote OTA re-programming is recommended for future study. For the first iteration of the sequencer, only the ability to execute the deployment sequence reliably is researched. This initial design will be reconfigurable through an interface on the SAD, with future designs possibly incorporating OTA reconfiguration. Another function of partial reconfiguration is configuration-fault repair, which was presented in more detail in Chapter II.

2. Historical Overview of FPGA Technology

PLD development was a new industry in the early to mid-1970s, developed by companies such as Motorola, Texas Instruments, and IBM [60]. The ability of PLDs to have not only programmable logic but programmable interconnection fabric gave rise to

the FPGA industry. Much of the groundwork for programmable logic arrays, gates and logic blocks are found in the two patents by David W. Page and LuVerne R. Peterson filed on January 11, 1983 [61], [62]. These patents were the first to incorporate the aspects of dynamic re-programmable PLAs. This gave way to the notion of devising a high-density programmable logic device from an SRAM cell in the mid-1980s. Ross Freeman founded the Xilinx Corporation in 1984 and successfully obtained a patent for a 'Configurable Electrical Circuit Having Configurable Logic Elements and Configurable Interconnects' on September 26, 1989—the first FPGA [63]. Seiko Epson—Semiconductor Division manufactured the first FPGAs for Xilinx in 1985 with a 1,000 gate ASIC equivalency and clock speed of 18 MHz. This FPGA, the XC2064[6] had both programmable gates and programmable interconnects—64 configurable logic blocks (CLBs) each with two 3-input lookup tables (LUTs) [64].

Recent advancements in IC manufacturing techniques have led to a series of devices with feature sizes that were unthinkable in years before. These advancements have extended to FPGA families with some of the more advanced series (Xilinx Virtex[7] 6 or 7) containing millions of logic devices onboard. In contrast to the XC2604, a Virtex 7 with a clock speed of 1.866 GHz, may contain up to 305,400 CLBs with a total logic cell count of 1,954,560 and 1,221,600 6-input LUTs [65].

With the reduction in feature size comes the added burden of an increased probability of SEUs occurring. Indeed, the smaller the feature sizes of the devices, the larger the problem and associated probability of failure [66]. Additionally, the newer problem of cascaded SEUs occurring in a section of a device, or an MBU occurring, has been seen to markedly increase over the past couple of generations of devices. The specifics of SEEs and MBUs, challenges of radiation-induced failures, and the associated mitigation techniques were previously discussed in Chapter II.

For every generation of FPGA developed—the time it takes to fully space qualify an FPGA is seen to increase markedly. The physical differences between commercial and

[6] XC2064® is a registered trademark of Xilinx, Inc.

[7] Virtex® is a registered trademark of Xilinx, Inc.

space-grade devices, and various radiation effect challenges, are presented in Chapters III and IV. Even though the Xilinx Virtex 7 series has been released for some time, only the Virtex 5QV series has been developed as a rad-hard variant of its commercial counterpart [67]. This is a lag time on the range of three to five years from release of their commercial counterparts. Obviously, a new method of fault tolerance is desired—to allow commercial or military grade devices to produce similar performance as a fully rad-hard, flight tested FPGA, in an identical environment. This change could reduce the need for the product lines of rad-hard devices from various manufacturing companies (Xilinx, Actel, Altera); however, it would readily open up their more recent and capable products to the space environment where they are currently not targeted.

3. Types of Modern FPGA Technology

The method of FPGA construction has evolved in recent years to one of three primary forms of design and implementation. Depending on the method of operation, these various FPGA technologies all possess their own set of strengths and weaknesses. In order to properly compare the three technologies: antifuse, SRAM-based, and flash-based FPGAs—the relevant design features and differences are presented.

a. Antifuse-based FPGAs

Antifuse technology is a type of permanent configuration that allows a connection to be made or broken once, upon initial programming. The antifuse FPGAs, while configurable, are not re-configurable types of devices. The permanent connections involve the creation or destruction between two wires. Antifuse-based FPGAs use a small piece of dielectric, usually smaller than 1 μm^2, as an open switch between the two lines. Layers of amorphous Silicon in the vias provide isolation between metal layers. Where a connection between the two metal lines is desired, a programming pulse is used to short out the dielectric. When this configuration voltage is applied, the amorphous silicon changes to a low impedance state, creating a metal-to-metal interconnect [68].

There are some benefits to this type of technology since it does not need to be configured each time power is applied to the device. It is this live-at-power-up capability that makes antifuse FPGAs a true single-chip solution for circuit design,

similar to ASICs. In addition, antifuse FPGAs do not suffer from radiation-induced configuration faults as their SRAM-based counterparts do [69].

b. SRAM-based FPGA Implementations

Reconfigurable FPGAs store their configuration in some form of memory. This memory is often static random-access memory (SRAM) in its implementation. In SRAM-based FPGAs, the configuration is downloaded at power-on with the contents controlling the setting of various switches. These switches in turn determine which metal lines are connected within the FPGA interconnects and logic blocks [20]. There are three primary disadvantages of SRAM-based FPGAs—one, that the configuration is volatile and must be re-loaded at each subsequent power-on; two, the use of active transistors for switches slightly increases the resistance of such connections; and three, SRAM technology is more susceptible to SEUs due to its inherent structure and design [20]. Vendors of SRAM-based FPGAs typically offer higher logic densities when compared to antifuse based solutions with nearly the same performance and speed levels [69]. This makes them an attractive option for space-based implementations.

The design decisions related to the use of SRAM-based FPGAs include the selection of secondary components to support boot configuration and the manner in which to configure within the time allocated for system startup. When a SRAM-based FPGA experiences a brownout or power-glitch, dedicated circuitry such as voltage supervisors and CPLDs must be in place to capture these events and force a reset and reconfiguration of the FPGA [69]. All of these design factors lead to a greater system cost, higher associated mass, increased circuit complexity, and higher power consumption. Since the sequencer is a resource intended for the space environment, weight and power consumption are important issues for consideration.

Previous thesis work done at NPS regarding CFTP utilized Xilinx Virtex I and II FPGAs for the experimental componentry. These types of devices are SRAM-based technology and required additional components for support such as synchronous dynamic random-access memory (SDRAM), programmable read-only memory (PROM), and electrically erasable PROM (EEPROM), as well as the discrete devices to support

those, including resistors, capacitors, voltage regulators, and an oscillator [9]. The EEPROM and PROM provide the configuration storage for the two FPGAs.

c. Flash-based FPGAs

Existing as a hybrid of antifuse and SRAM-based FPGA technologies, flash-based FPGAs offer many of the benefits of the two previously discussed types. These FPGAs utilize flash cells to store configuration information. Positive or negative charge stored on floating-gate transistors is used to hold pass transistors in either "on" or "off" states. This either opens or closes connections between routing tracks and logic resources [70]. This is similar to the method utilized for storage in typical flash-based memory, as illustrated in Figure 16. The flash cells are nonvolatile due to the storage of charge within the floating-gates, even when power is removed from the circuit.

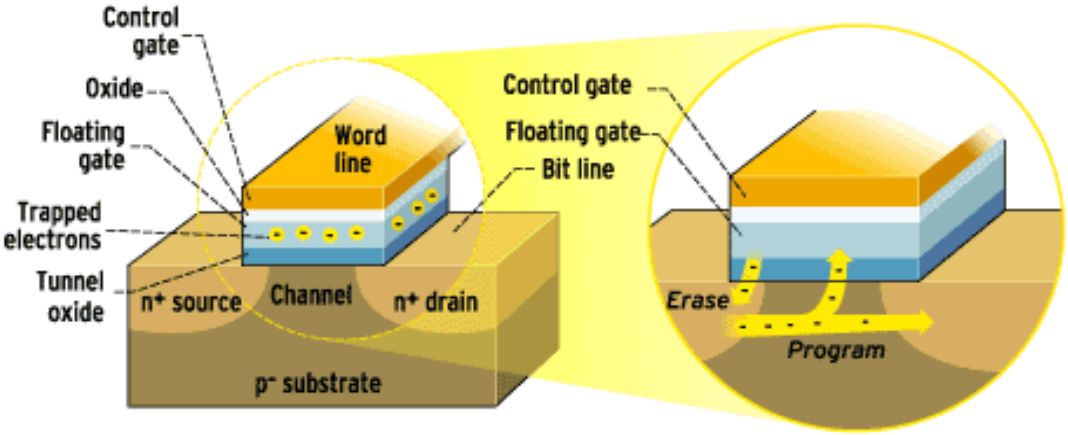


Figure 16. Internal flash transistor configuration with charge build-up in floating gate allowing for reconfiguration and nonvolatile storage (From [71]).

The flash-based technology allows the logic blocks, I/O block, and reconnect matrix to emulate reconfiguration capabilities of SRAM-based FPGAs while retaining the nonvolatile aspects of antifuse FPGAs. This allows the flash-based FPGA to have an operating mode almost instantaneously at power-up with no boot sequence or secondary components necessary to support. The inherent reduced SEU vulnerability of flash memory also makes them more desirable than SRAM for configuration memory.

4. Design Concerns

There are some concerns that arise due to the differences in FPGA manufacturing. The primary concern for the sequencer is the FPGA's radiation tolerance. Though not necessarily required for the sequencer application, a secondary interest is the ability of the general payload processor application to be re-programmable any time prior to launch. Reprogrammability demands that one of the two types of FPGAs incorporating re-writable characteristics in their logic blocks and interconnect matrix be utilized. Reconfiguration capability prior to launch for the general purpose payload processor design rules out using antifuse technology as the design would be permanent after the first configuration event.

In comparing the SRAM to flash-based FPGAs, the difference must be carefully considered. Since the majority of FPGA use at the Naval Postgraduate School has been with SRAM-based products, there is a strong desire to remain within those product lines to benefit from the experience of past designs. The use of SRAM technology, however, makes the overall design more complex in schematic layout, microcontroller interfaces, and power provisioning. This is related to the requirement to restore the configuration of the SRAM-based FPGA at initial power-on from an external memory device. This increases the time from power-on until operation and the need to protect secondary components from the effects of the space environment. Radiation is a challenge no matter how many components are utilized, however, increasing the secondary required devices and interfaces only add to these difficulties. Inherently, SRAM-based FPGAs are more vulnerable to SEUs than flash-based products.

On the basis of simplicity, with an eye toward speed and reliability, the initial viewpoint of the author is to take the flash-based FPGA route. This will yield a simple, reconfigurable circuit while providing the added speed benefit of operation at power-on. For the sequencer, this should eliminate any added "wait" time that would be necessary to account for before the deployment sequence proceeds. Due to the harsh environment and desire for very precise timing constraints, a sequencer that is ready to start deployments immediately upon receiving 'Arm' and 'Deploy' signals is of most interest. The final decision as to the type of FPGA technology to utilize is presented in the next

section of this chapter. The differences between the Actel and Xilinx FPGA technologies are more fully discussed in Appendix A.

B. FINAL SELECTION OF ACTEL PROASIC3 SERIES FPGA

1. Required FPGA Characteristics

There are several important characteristics that an FPGA must possess for applications related to the space environment. When choosing the appropriate device, several factors must be taken into consideration: non-recoverable hard errors such as SELs and SEBs, TID tolerance, choice of technology utilized, computing power concerns, and the cost of the device. Within these, the tolerance of the FPGA chosen to SEL and SEB effects, along with its accumulated TID tolerance, are among the most important factors for consideration in the choice of a device. Since non-recoverable hard errors may lead to a complete electronic failure of the FPGA, these must be protected against at all cost. The gain in survivability with regards to these effects is often worth the higher cost rad-hard technology when utilized for mission critical applications.

Following this, the computing power necessary for the proper operation of the logic necessary for a specific FPGA application is of immediate concern. If the FPGA is sufficiently protected from non-recoverable errors but unable to perform its desired function due to speed concerns, it will be of little use once on-orbit. Closely related to the speed of the device is the reconfigurable technology selected for the application. SRAM-based FPGAs tend to have inherent speed advantages over their flash-based counterparts; however, many of the speed increases come at the added cost of reduced reliability for radiation tolerance. Additionally, the added circuitry and support components required for the SRAM-based designs greatly impact the overall cost of the platform as a whole. In many cases, cost may become a large driver on the selection of the technology as certain payload requirements call for multiple FPGAs to be utilized, along with any space-grade support circuitry. This may greatly limit the options available to the designer on smaller scale programs such as this sequencer project.

2. Overall Sequencer and Processor Characteristics

When focusing on the specific sequencer application of this research toward a general-purpose payload processor, there are considerations that factor into selection of a particular FPGA. These include the resistance of the FPGA to recoverable, soft errors such as SEUs and SETs and the selection of a device with the lowest SWAP (size, weight, area, and power) requirements; the choice of a particular vendor and product line must take all these factors into account.

In general, flash-based FPGAs are more resilient to SEU and SET effects than SRAM-based devices. Also, due to the lack of a requirement for secondary support circuitry, flash-bashed FPGAs again save SWAP when taken in comparison to SRAM-based devices. The flash-based FPGAs are live-at-power-on, take almost no power during idle periods due to lack of a need to refresh the devices, and typically are smaller in SWAP requirements. This has the added effect of producing less of a thermal load, with less heat being produced by flash-based FPGAs. All of these factors typically lead to an overall smaller cost for flash-based FPGAs than SRAM-based FPGAs. The advantages of flash-based FPGAs over the SRAM-based products greatly sway the decision for these reasons.

3. Comparisons of Xilinx and Actel FPGAs

While the Virtex-5 series may provide more overall logic area for expansion, slightly increased TID radiation performance, and additional DSP functions not present in the ProASIC3 devices—for a sequencer designed to deploy CubeSats reliably, overall device utilization should not be the predominate factor. Even given the added logic overhead of TMR or quadded logic implementation, the ProASIC3 FPGA should be fully capable of configuration with the targeted design. A brief comparison of both the Virtex and ProASIC3 series FPGAs is provided in Table 1. More information concerning specifics of each device type is found in Appendix A.

With the Xilinx FPGAs utilizing SRAM-based technology versus the Actel flash-based FPGAs, the prominent difference between these two implementations exists at the hardware level. This equates to the need for secondary memory components required to

support the FPGA both during power-on configuration and subsequent operations. While the number of user I/Os is considerably less for the Actel ProASIC3 series, the provided 270 maximum is more than enough for the sequencer application within the general-purpose payload processor design.

Table 1. Overall hardware comparison between Actel ProASIC3 and Xilinx Virtex-5 series FPGAs illustrating strengths and weaknesses.

	FPGA Device Series	
	Actel RT ProASIC3	**Xilinx Virtex-5QV**
Device Type	Flash-based	SRAM-based
Quantity of Logic Cells	13,824 VersaTiles	20,480 Slices
Number of User I/Os	270	836
Number of Clocks	6	18
Clock Speed Frequency (MHz)	350 @ 1.5 VDC / 250 @ 1.2 VDC	450
SEL Immune (MeV-cm2/mg)	68	> 100
TID Immune	55 Krad	1 Mrad
External Memory Required	No	Yes
Number of Package Pins	484	1,738
Price (US Dollars)	$63.87	$4,352.40

If the goal of the general purpose payload processor were limited to DSP functions, or faster operation, the Xilinx Virtex-5 series would be a better solution. Since only comparing the number of logic gates is not meaningful when comparing different vendors, it is more useful to compare the baseline logic cell architectures. The equivalency for Xilinx logic slices to Actel VersaTiles is listed in Table 2 [72]. From this data, one may conclude that one Virtex-5 slice is equivalent to approximately eight Actel VersaTiles. Though the Virtex-5 slices have more logic capability than the Actel VersaTiles, there should be sufficient overhead in the Actel FPGAs for design of fault-tolerant logic for a sequencer design. Testing of different fault-tolerant schemes for FPGA logic utilization factors is conducted in Chapter V.

Table 2. Approximate relationship between different vendors FPGA logic blocks (After [72]).

Logic Block	Virtex-4 Slice Equiv.
Xilinx Virtex-4 Slice (Ref.)	1
Xilinx Virtex-5 Slice	2
Actel VersaTile	0.25

The volatile nature of SRAM-based FPGAs and the overall requirements for secondary componentry result in the choice of Actel flash-based technology over the Xilinx product. Even considering the reduced schematic capabilities of adopting Actel SoC over Xilinx ISE[8], the associated hardware benefits inherent to the ProASIC3 series are worth the effort. The difficulty in learning a new software package for logic design is more than compensated by the strengths of the single-chip Actel solution. This Actel ProASIC3 series is the clear choice for simplicity, due to the reduced number of secondary components, and the potential for reduced cost.

Given the need for a low-power, low-cost general purpose fault-tolerant processor design, the Actel ProASIC3 FPGA is the suitable choice for the development of hardware in this thesis. This differs from earlier work done on CFTP by limiting the amount of secondary component design necessary for completion of a PCB. The overall technology behind the FPGA has changed from SRAM to flash-based in this implementation. As previously discussed in Section A.3 of this chapter, this flash-based technology also has the added benefit of increased SEU resistance in the configuration memory. Since one of the goals of this sequencer design is simplicity, the ability for the sequencer to function immediately upon power application of the ProASIC3 fits better with the desired sequencer operation. This should allow for an ‘Arm’ signal to be passed to the sequencer by the power initiation, with the ‘Deploy’ signal coming from the internal sequencer logic. Further detailed comparisons between the Xilinx and Actel products which result in the selection of the Actel ProASIC3 FPGA may be found in Appendix A.

[8] ISE® is a registered trademark of Xilinx, Inc.

Initial work will be done to develop a 'ProASIC3 Test Board' using the Actel ProASIC3 nano series of FPGAs in Chapter IV. This will allow for design software familiarity prior to development of the flight prototype board. Altium Designer will be used for the schematic and PCB design of the ProASIC3 Test Board and SAD Version 3 – Prototype Flight Board. All prototype flight hardware schematic and PCB design work will be accomplished using the Actel ProASIC3 military-grade component (A3PE600L-FG484M) in Chapter IV. Drop-in compatibility with the lower-cost, commercial-grade FPGA (A3PE600-FG484) should allow for several prototypes to be manufactured. By staggering the design, one is able to quickly develop and manufacture a test board to become familiar with the Microsemi product lines, datasheet representations, and skill-set necessary to develop the more complete design.

C. SELECTION OF DEVELOPMENT BOARDS FOR LOGIC DESIGN

In addition to developing an in-house test board, various vendor manufactured development boards were purchased in tandem with this research. This allowed for a known working configuration to proceed with logic development parallel to hardware design and manufacture. Since past familiarity was with Xilinx FPGAs and software, a decision was made to purchase a Virtex-5 development board in addition to a ProASIC3 variant. If unexpected results with logic area utilization were to occur, the Virtex-5 series would provide a backup choice for work to continue. For quick logic development and testing, a Spartan-3E series development board was utilized. This development board was USB powered, allowing for portability and rapid testing. Finally, a ProASIC3 development board was purchased, with the same series of FPGA as the final BGA design—to allow for complete logic design testing. The comparison between the various boards is provided in Table 3.

Table 3. Comparison of FPGA development boards used in logic design and testing.

	Board #1	Board #2	Board #3
FPGA Device	Spartan-3E 100	Virtex-5 LX50T	ProASIC3L 1000
FPGA Part No.	XC3S100E	XC5VLX50T	M1A3P1000L
System Gates	100,000	Not Specified	1,000,000
Logic Cells	2,160	50,000	24,576
Total CLBs	240	3,600	N/A
Total Slices	960	7,200	N/A
VersaTiles	N/A	N/A	24,576
Distrib. RAM bits	15,360	491,520	147,456
Block RAM bits	73,728	2,211,840	147,456
Number of Clocks	2	6	6
User I/Os	66	480	300
Differential I/O Pairs	30	240	74
Max. Clock Speed	100 MHz	550 MHz	350 MHz
FPGA Size	16 x 16 mm	35 x 35 mm	23 x 23 mm
Number of Pins	100	1136	484

D. CHAPTER SUMMARY

In Chapter III, the Actel ProASIC3 FPGAs were seen as the best solution to the sequencer application. This is a result of the increased resilience to soft-error effects, reduced secondary hardware required for operation, ease of design, live at power-up operation, and pin-for-pin compatibility between versions. Antifuse FPGAs were discarded as a design choice due to the non-availability of reconfiguration. SRAM and flash-based FPGAs were contrasted with the differences in design considerations realized. Conclusively, flash-based FPGAs seem the most promising choice due to their configuration memory's reduced SEU vulnerability, reduced system complexity, and live-at-power-on capability. The ProASIC3 Test Board will be developed in Chapter IV; follow-on development of the SAD Version 3 – Flight Prototype Board will also occur in Chapter IV.

IV. HARDWARE DEVELOPMENT – PROASIC3 TEST BOARDS AND PRELIMINARY FLIGHT PROTOTYPE

A. PROASIC3 TEST BOARD

Due to the need to obtain test hardware as rapidly as possible, a decision was made to build a set of ProASIC3 test boards prior to the development of a more challenging flight hardware prototype. This would serve two purposes—one, that a ProASIC3 Test Board (PA3TB) would quickly allow for rapid prototyping of logic designs; two, that familiarity would be gained with the hardware development package, Altium Designer 10. The ProASIC3 nano FPGA selected allowed for familiarity with the Actel Libero[9] SoC software environment while retaining the ability to manufacture the test board in the Small Satellite Laboratory. The goal of the PA3TB design was to minimize the secondary components required and to realize the most basic design that could accomplish the requirements of the sequencer application.

Altium Designer provides a unified product development environment including: front-end design and capture, physical PCB design, FPGA hardware design, FPGA system implementation and debugging, embedded software development, mixed-signal circuit simulation, signal integrity analysis, and PCB layout [73]. The standard development environment can be seen in Figure 17, where schematic, two-dimensional (2-D) PCB layout, and 3-D board placement views are visible on the separate monitors.

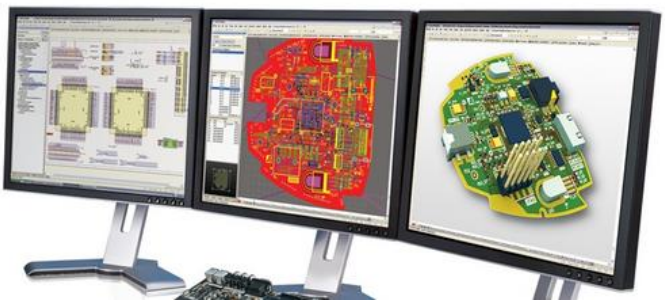


Figure 17. Altium Designer interface with separate schematic, 2-D PCB layout, and 3-D component placement views visible on separate monitors (From [73]).

[9] Libero® is a registered trademark of the Microsemi Corporation.

An added benefit to obtaining the preliminary hardware was experience gained in the manufacture and testing of the design. This allowed for in-house soldering skill to be gained with the smaller surface-mount components as well as the baseline FPGA device. A complete testing of the JTAG interface to be utilized on the flight prototype board was also possible to ensure all design difficulties were worked out prior to scaling up the design.

1. Derived Requirements

There were several derived requirements determined for the ProASIC3 Test Board. The particular requirements utilized for design are as follows:

1. Use of ProASIC3 FPGA with flash-based architecture;
2. Core voltage of 1.2 V_{DC} to reduce power consumption;
3. Specification of I/O bank voltages at core voltage or higher rating;
4. External JTAG (IEEE 1532) interface, compatible with standard hardware programmers;
5. Simulation of CubeSat P-POD deployment via indicators;
6. Expandability, in terms of additional I/O availability and extra interfaces such as USB;
7. Implementing the design in the smallest form-factor available, which implies utilizing surface-mount components.

2. Selection of Board Size and Layer Count

Since the actual ProASIC3 nano FPGA has a relatively small footprint, the PCB size was chosen to minimize the size of the board—allowing associated secondary components the minimum amount of room for layout. This decision allowed for effective resource usage when placing desired components. The overall dimensions of the board chosen are 1.87 inches × 1.87 inches (3.497 in^2). This allows for center placement of the FPGA upon the PCB with secondary components surrounding.

Since the SAD Version 3 Flight Prototype Board (SADv3) will use a BGA with hundreds of pins, the routing difficulties inherent to that technology mandate the use of a multi-layer PCB. To prepare for the evolution to that hardware, it was decided to manufacture the PA3TB using a four-layer design. Although the number of signal and power lines was such that a two-layer board could have been used, the experience gained from the four-layer PCB design will be valuable in the layout of the six to eight-layer SADv3. As is standard, the internal layers consisted of the ground (GND) and positive voltage (V_{CC}) place. By placing V_{CC} and GND copper polygon-filled areas internal to the PCB, the signal lines for data were visible on the top and bottom planes of the PCB. This allows for more flexibility in the case where signal lines are misrouted due to design-error or manufacturing defects. Additionally, there are capacitive and signal-noise issues that are resolved when designed in this fashion. The overall layer structure of the PA3TB is illustrated in Figure 18. Through-hole vias were used throughout the design for coupling between layers.

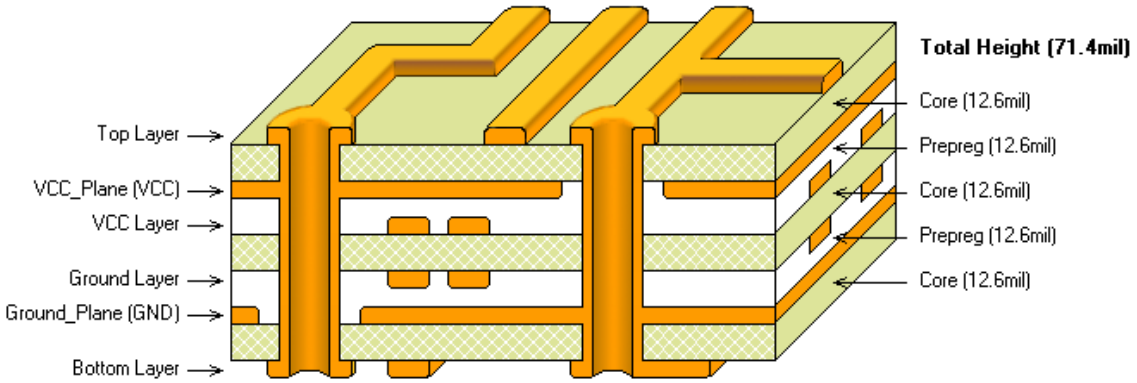


Figure 18. Four-layer stack of the ProASIC3 Test Board with internal power planes and external signal layers.

3. Selection of Components

In Chapter III it was decided that the ProASIC3 architecture was the best approach; the decision was made to utilize a comparable FPGA technology without the overhead of BGA manufacturing for test board development. Actel's ProASIC3 nano has the advantages of being a low-cost, single-chip solution utilizing flash-based technology

coupled with a 100-pin Very Thin Quad Flat Package (VQFP). The VQ100 package of the chosen A3PN250 FPGA, which allows for manufacture utilizing a microscope soldering station in the Small Satellite Laboratory, is illustrated in Figure 19. The particular characteristics of the chosen ProASIC3 FPGA are specified in Table 4. Specific schematic decisions, PCB layout and component placement choices, and the physical construction and testing of the PA3TB are fully described in Appendix B.

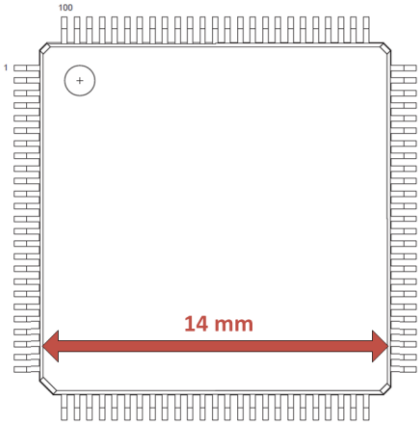


Figure 19. ProASIC3 nano VQ100 package of VQFP-type (From [74]).

Table 4. Characteristics of ProASIC3 nano FPGA (A3PN250) utilized in development of ProASIC3 Test Board (After [74]).

	PA3 Test Board
FPGA Device	ProASIC3 nano
FPGA Part No.	A3PN250
System Gates	250,000
Logic Cells	2,048
VersaTiles	6,144
Distrib. RAM bits	36,864
Block RAM bits	36,864
Number of Clocks	8
User I/Os	68
Differential I/O Pairs	N/A
Max. Clock Speed	350 MHz
FPGA Size	14 x 14 mm
Number of Pins	100

The full 2-D PCB layout can be seen in Figure 20. Individual electrical and mechanical layouts for the various layers are presented individually in Appendix B.

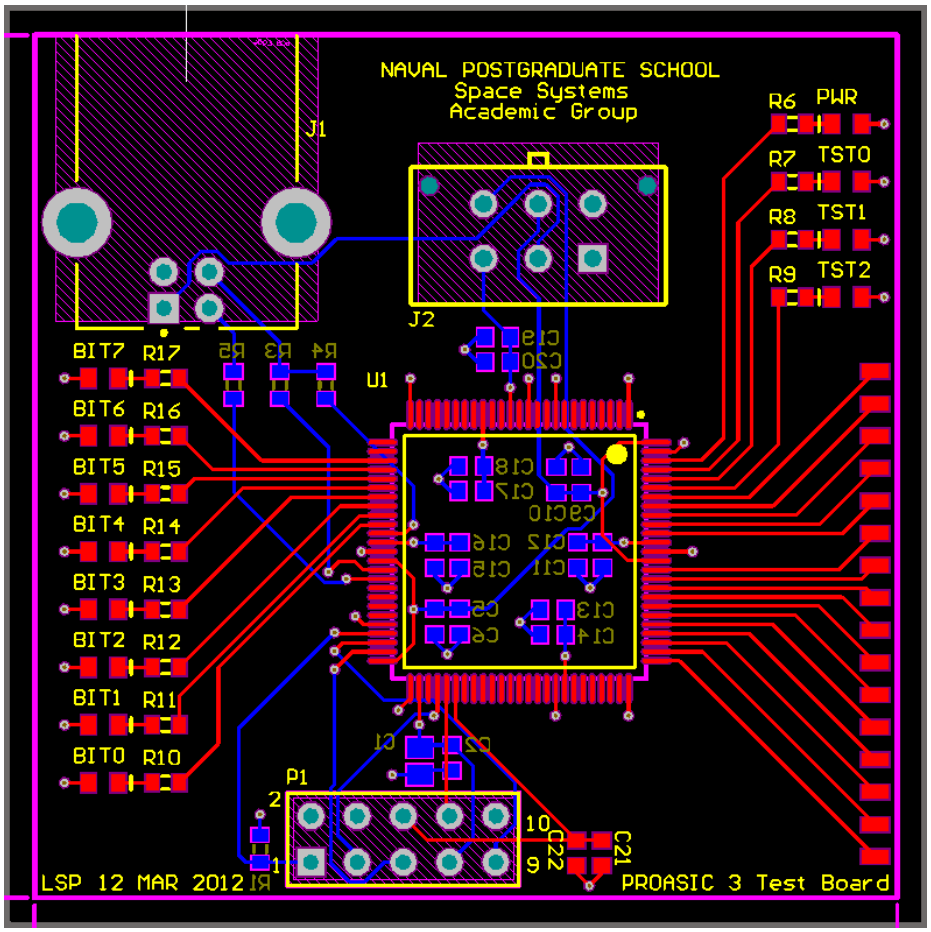


Figure 20. ProASIC3 Test Board 2-D PCB layout with all component footprints, signal lines, and vias visible.

After the 2-D layout was concluded, the various 3-D Standard for the Exchange of Product Data (STEP) models were imported into the design the check for fit. These components were viewed in a 3-D representation of the integrated PCB design, allowing for any fit issues to be quickly resolved, as illustrated in Figure 21. There were a few cases where vias or traces were moved to better accommodate the physical representation of the PCB. This is an important prototyping tool and saves a great deal of work prior to receiving the initial run of PCBs. The entire layout process took place over the course of approximately six weeks.

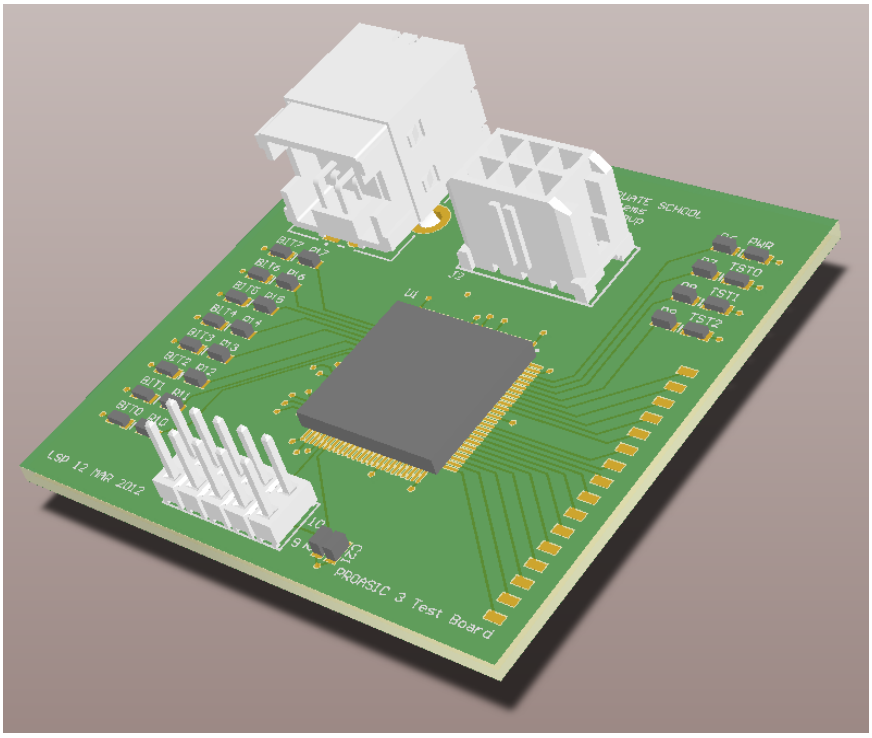


Figure 21. ProASIC3 Test Board 3-D isometric view with component models indicating approximate sizing and spacing of integrated board.

Further 3-D views of the top and bottom face of the PCB are available in Appendix B.

B. SAD VERSION 3 – FLIGHT PROTOTYPE BOARD

While work was proceeding with the fault-tolerant logic design, described in Chapter V, modification of the FPGA-based sequencer hardware design occurred in parallel. Concurrent design of the SAD Version 3 – Flight Prototype Board was undertaken to allow sufficient time for logic development and testing, with the anticipated long lead-time in PCB layout for a BGA FPGA. The means of implementing the more extensive BGA form-factor FPGA into a prototype, flight-hardware design is presented in Appendix C. The design of this board is not meant to be all-inclusive of the final general purpose processor design, as some necessary hardware revisions will understandably occur; the overall context of the schematic design along with supporting circuitry is expected to remain relatively constant. There is a desire for a design which can be easily expanded to incorporate future feature requests as necessary for future

missions. Though the sequencer is the first practical application of the general purpose payload processor, future iterations of the design may require extended logic and memory usage. As such, the developed BGA design incorporates several features such as additional SRAM and flash memory, along with general purpose and differentially paired I/Os to support future requests. The board developed is termed SADv3 for the purposes of the sequencer, however, it may be renamed in the future to indicate its general purpose feature set.

As in the previous section, all work done on the design of the SADv3 was accomplished with Altium Designer. A great deal more schematic development was required in the implementation of this more extensive design. Additionally, the added work in PCB layout, when transitioning from a six to eight-layer PCB, was seen to increase almost exponentially. As such, the final design presented in the course of this research will require multi-person design panel review before manufacture—there are simply too many connections and possible failure points to risk production without extensive oversight on the development of the board. To initiate production prior to complete design review would invite an expensive redesign should an error be present.

1. Discussion of Specific FPGA

In the transition to a larger FPGA fabric to incorporate additional logic and I/Os, the need for switching from a VQFP to BGA became readily apparent. In the development of the PA3TB, the largest resource-area FPGA was chosen in the design of that board. Although the quantity of gates in that design are more than suitable for the instantiation of the sequencer logic, any additional features would quickly expand past the resources of the device, especially once TMR is implemented. As discussed in Chapter III, all prototype flight hardware schematic and PCB design work will be accomplished using the Actel ProASIC3EL military-grade component (A3PE600L-FG484M). Drop-in compatibility with the lower-cost, commercial-grade FPGA (A3PE600-FG484) should allow for several prototypes to be manufactured. The FG484 package of the chosen A3PE600L FPGA is illustrated in Figure 22. The particular characteristics of the chosen ProASIC3EL FPGA are specified in Table 5. Again, should

there be a need for the fully rad-hard FPGA in future missions, the RT ProASIC3 (RT3PE600L-CG484B) is compatible with the defense-grade ProASIC3EL (A3PE600L-FG484M).

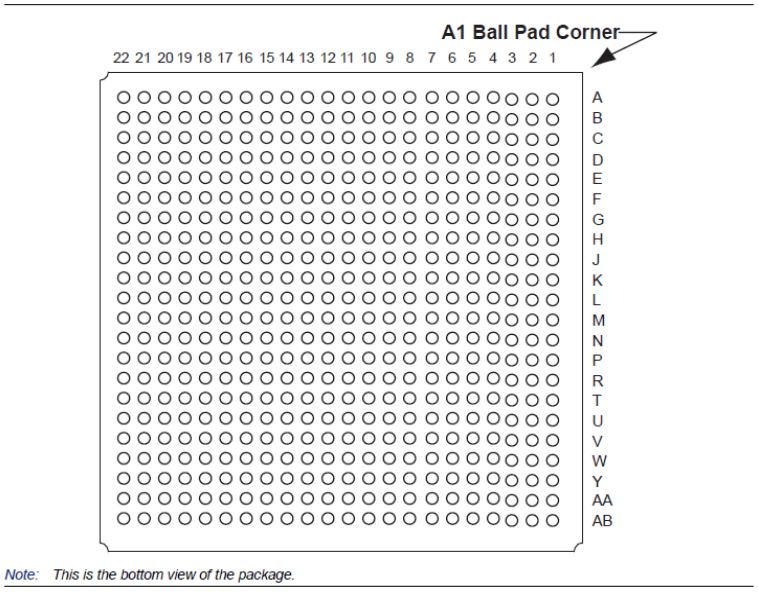


Figure 22. FG484 BGA package of the Actel ProASIC3L A3PE600L FPGA (From [75]).

Table 5. Characteristics of ProASIC3L BGA (A3PE600L) utilized in development of SAD Version 3 – Flight Prototype Board (After [75]).

	Flight Prototype
FPGA Device	ProASIC3EL
FPGA Part No.	A3PE600L
System Gates	600,000
Logic Cells	4,608
VersaTiles	13,824
Distrib. RAM bits	110,592
Block RAM bits	110,592
Number of Clocks	8
User I/Os	270
Differential I/O Pairs	135
Max. Clock Speed	350 MHz
FPGA Size	23 x 23 mm
Number of Pins	484

In using the BGA FG484 form-factor, the complexity of the design is increased markedly with a corresponding increase in requirements for PCB design. Due to the BGA package, soldering of the PCB is impossible utilizing equipment in the lab; manufacture of the board and placement of surface-mount components for reflow soldering will take place with an outside PCB production company. The design of the PCB necessarily takes into account the required sizing and layer count to properly locate and support the BGA FPGA and all supporting components.

The initial layout of the board shape with FPGA, headers, and prototyping area visible is illustrated in Figure 23. The ribbon-cable connector for attachment to the relay board can be seen toward the top edge of the PCB. As specified previously, the sequencer board is approximately half the width of the relay board (taking up area from the bottom to middle bolt-holes). This design allows for additional area in future revisions if the footprint is modified to be identical to the relay board.

Work continues on the appropriate component placement and wiring of traces. Placement of the power de-coupling capacitors is somewhat problematic with relation to the FPGA footprint; careful selection of appropriate capacitors to place closest to the footprint is imperative for ease of trace laydown. After wiring, extensive design reviews must be accomplished to verify not only the schematic but also the layout of the PCB in 2-D signal trace layers and 3-D component placement and sizing.

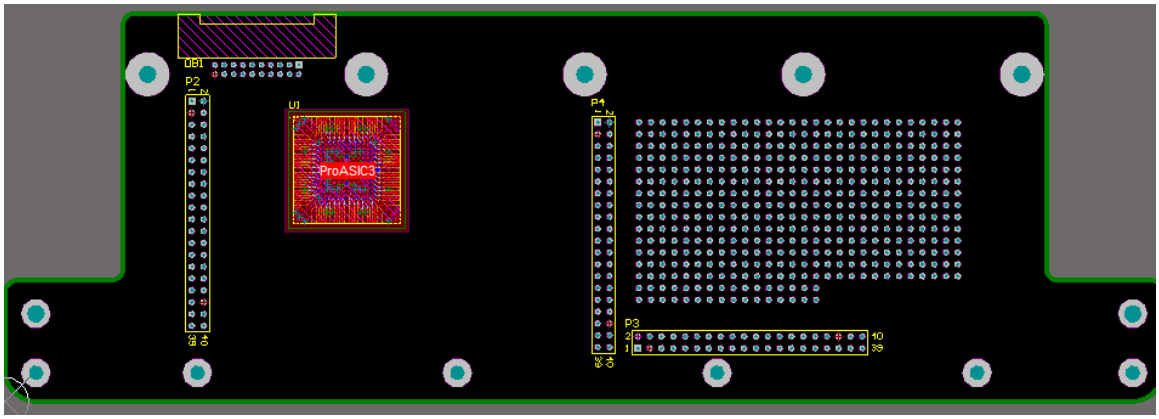


Figure 23. Initial 2-D PCB layout of FPGA, three 40-pin I/O headers, prototyping area, and ribbon-cable connector for relay board attachment.

Recent representations of individual electrical and mechanical layouts for the various layers are presented individually in Appendix C. Further 3-D views of the top and bottom face of the PCB are also available in Appendix C.

2. Additional Modifications Considered

Since the board developed is the first initial prototype of the eventual flight sequencer, there are some features included for bench testing which can be removed in the final representation of the design. For this design a +5 V_{DC} connector was utilized to provide V_{IN} to the LDO voltage regulators for conversion to the appropriate voltages. Since the relay board will be receiving a +28 V_{DC} supply from the host spacecraft bus, there will be a way to incorporate this conversion on the relay or sequencer board itself. As such, the +5 V_{DC} connector can be removed from the final flight design with the +5V net replacing the V_{IN} net. The SRAM and flash-memory can be left off the footprint since they are not required in the sequencer design and can be installed as necessary for future missions.

Additional modifications to the design include possible future incorporation of two-way telemetry concerning board and CubeSat status back to the host spacecraft, a video camera feed of CubeSat deployments with data saved in memory, smart charging and maintenance of CubeSats within NPSCuL prior to launch, and experimental payloads for testing of logic in a general payload processor design. One of the desired short-term features is the inclusion of an Ethernet port and standard for testing in radiation environments. Data provided over this port could be used in data logging the radiation upsets and recoveries during testing events.

C. CHAPTER SUMMARY

The reasons for the development of the ProASIC3 Test Board prior to design of the SAD Version 3 – Flight Prototype Board were discussed in this chapter. Development of the PA3TB was accomplished using Altium Designer 10 for the schematic and layout portions of board design. The design was implemented with a set of derived requirements to enable the rapid production of a hardware product; the PA3TB serves as a model for evolution to a more complex PCB. Minimization of PCB area and secondary components

were important in troubleshooting the device when problems were discovered. Other design decisions are presented concerning the manner in which they drove PCB component selection.

In addition, the design modifications for the SAD Version 3 – Flight Prototype Board were also discussed in this chapter. Again, development of the SADv3 was accomplished using Altium Designer for the schematic and layout portions of board design. The work in this chapter on the SAD Version 3 – Flight Prototype Board continues with all of the circuitry having been schematically represented. Component placement and trace layout for the support devices on the board is still on-going. Specific considerations for future board development and expansion were provided.

THIS PAGE INTENTIONALLY LEFT BLANK

V. COMPARISON OF FAULT-TOLERANCE METHODS FOR USE IN SEQUENCER LOGIC DEVELOPMENT

A. TMR VERSUS OTHER FAULT TOLERANT APPROACHES

For this portion of the thesis, the Xilinx Spartan-3E development board was used to simulate logic designs. The specifics of the FPGA utilized are listed in Chapter IV, Table 2. This device allowed for the utilization of the Xilinx ISE 14.3 WebPACK toolset so that all logic constructs could be created schematically and in VHDL or Verilog. A 16×16-bit array multiplier was constructed to test for the proper operation of the device using different redundancy techniques. All designs were carried through the synthesis, map, and place and route phases to ensure accurate timing information was obtained in the simulation. Additionally, the Xilinx techniques were manipulated to program Microsemi ProASIC3 devices. These altered designs will be incorporated into the final sequencer logic using Microsemi Libero SoC 10.1. In Chapter III, a decision was made to use the Actel products due to hardware concerns of the SRAM-based Xilinx technology versus the flash-based FPGA technology.

1. Maintaining Logic Structures within HDL Languages (Verilog / VHDL and Schematically)

To test for proper operation and characteristics of the Xilinx ISE 14.3 software, it was first necessary to create a test circuit which could be successfully duplicated and verified. This was accomplished via a combination of synthesis and simulation with follow-on hardware testing. The design chosen for the test consisted of a simple seven-segment LED counter. This design is a state-machine counter with sixteen states (0 through F) with the added functionality of push button and switch status. The counter design is implemented in VHDL with the overall clocking setup and top-level design applied in a modular fashion. As illustrated in Figure 24, the design was implemented on a Digilent, Inc. BASYS2 development board. This board incorporates the Xilinx Spartan-3E 100. Code for this state-machine counter is in Appendix D.

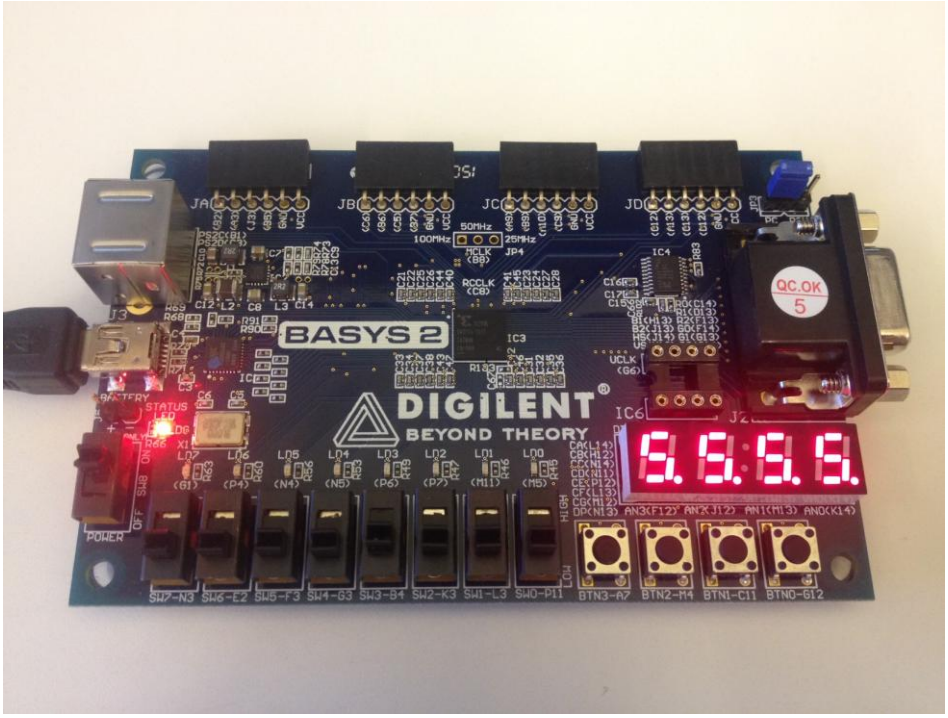


Figure 24. Digilent BASYS2 development board counting with proper state logic.

a. Baseline Design – Seven-Segment LED Counter Example

The resource utilization for the baseline LED counter design was found to use 19 of 960 available slices (2%) with a corresponding 1% utilization of FFs and 1% use of LUTs. As depicted in Figure 25, the overall gate layout of the design in a technology schematic is visible for the unaltered design. This provided a good baseline for comparison once modification to the design evolved.

Figure 25. Default gate layout for baseline seven-segment LED counter.

b. Triplicated Design – Seven-Segment LED Counter Example

Upon initial synthesis, there were some issues with ISE overly trimming the redundant logic from the design. This manifested itself in the removal of two of the three triplicated counter circuits and the voter circuit being marked as unnecessary in some cases. To alleviate these issues, 'Resource Sharing' in HDL Options and 'Equivalent Register Removal' in Xilinx Specific Options must be disabled in the Synthesis – XST 'Process Properties.' This alone was insufficient to prevent logic trimming of desired redundant circuits, so the following lines were added to the VHDL code:

```
attribute equivalent_register_removal: string;
attribute equivalent_register_removal of signal : signal is "no";
```

The preceding lines must be added to the architecture structures within the logic that is being trimmed after the signal names are defined.

For the operational test, the design was altered such that the top-level depiction occurs schematically, as seen in Figure 26. This was done by instantiating the seven-segment LED counter into its own schematic symbol. All signal lines were placed manually in the same fashion that they occurred in their VHDL depiction. The counter was triplicated, and a voter circuit was added at the output to test for proper operation and recombination of the signals.

This produced the proper, anticipated results as illustrated in Figure 27. The logic of the unaltered design can clearly be seen as triplicated in the technology schematic view of the gate structure. The resource utilization for the triplicated design was found to use 61 of 960 available slices (~6%) with a corresponding 4% utilization of FFs and 5% use of LUTs.

In the FPGA layout view, Figure 28, one of the three counters is highlighted in red with the triplicated counters occurring in the upper portion of the graphic. Since this occurs after the place and route phase, one can confidently say that the logic trimming of the redundant circuits has been prevented. This is an important step before continuing with the implementation of different fault-tolerant techniques.

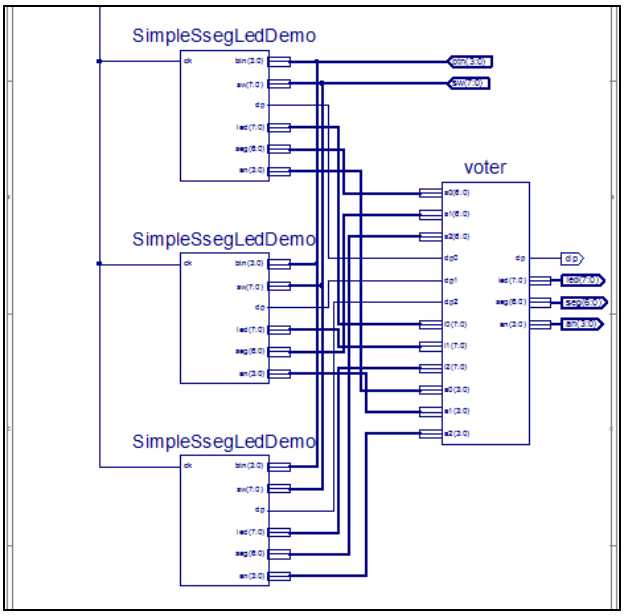


Figure 26. Triplicated seven-segment LED counter example with voting logic.

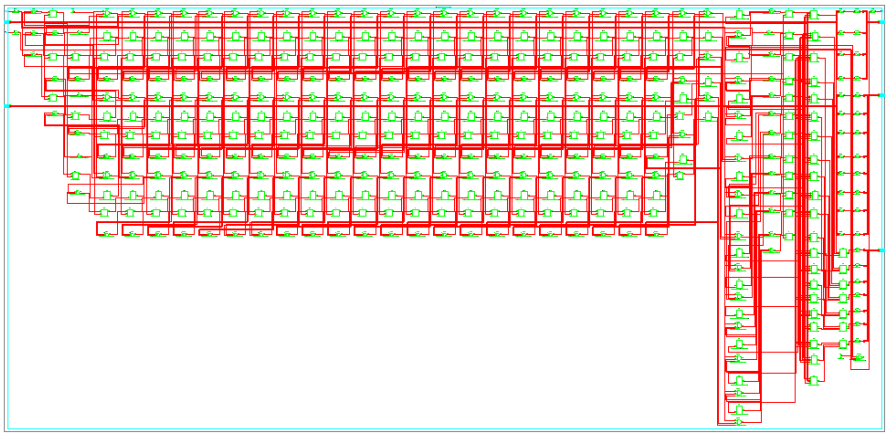


Figure 27. Triplicated gate layout with voting logic - seven-segment LED counter.

When testing onboard the physical device, the functionality of the development board was seen to be identical in the triplicated logic case to the non-fault-tolerant design. The FPGA continued to count in the proper state sequence with all push button and switch controls working as before. Finally, the design was altered to enable improper operation of one of the counters by reordering the states within that counter. This resulted in the mismatch of the output vectors between that counter and the other two, properly functioning counters. The voter was seen to work properly, as no discrepancies were noted in the operation of the development board FPGA.

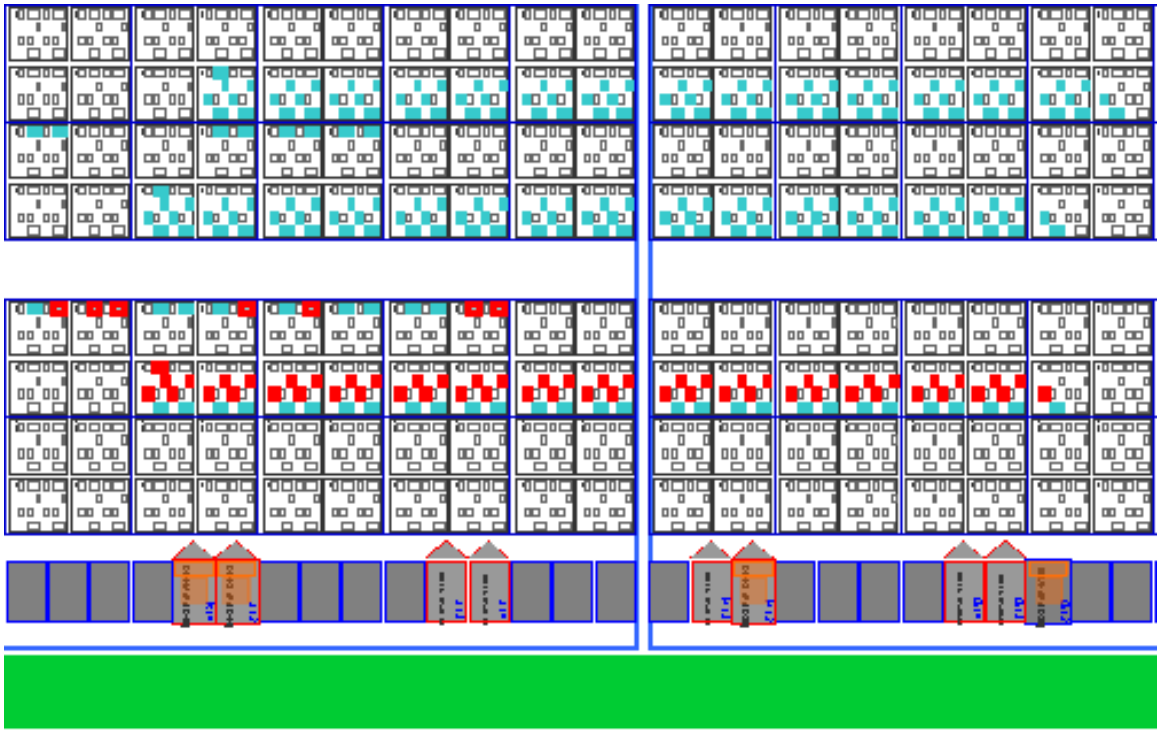


Figure 28. FPGA layout view illustrating placement of three counter structures.

2. Implementation of Various Redundant Techniques in Combinational Logic Designs

a. Baseline Design – 16×16-bit Array Multiplier

The design chosen for further testing of fault-tolerance is a 16×16-bit array multiplier which has been implemented in a tree structure consisting of 4×4-bit array multiplier subsections. The output of the array multiplier is a 32-bit unsigned integer value which correctly represents the product of the two input vectors. The 4×4-bit array multipliers are carry-save-adder versions with 7-bit sum and 5-bit carry output vectors. They are combined in the following fashion to generate proper multiplication:

$$
\begin{aligned}
P = A \times B &= \left(\sum_{k=0}^{3} a_k 4^k\right) \times \left(\sum_{l=0}^{3} b_l 4^l\right) = \sum_{k=0}^{3}\left[a_k 4^k \left(\sum_{l=0}^{3} b_l 4^l\right)\right] \\
&= 4^6\left(a_3 \times b_3\right) + 4^5\left[\left(a_3 \times b_2\right) + \left(a_2 \times b_2\right)\right] + 4^4\left[\left(a_3 \times b_1\right) + \left(a_2 \times b_2\right) + \left(a_1 \times b_3\right)\right] \\
&+ 4^3\left[\left(a_3 \times b_0\right) + \left(a_2 \times b_1\right) + \left(a_1 \times b_2\right) + \left(a_0 \times b_3\right)\right] \\
&+ 4^2\left[\left(a_2 \times b_0\right) + \left(a_1 \times b_1\right) + \left(a_0 \times b_2\right)\right] + 4^1\left[\left(a_1 \times b_0\right) + \left(a_0 \times b_1\right)\right] + 4^0\left(a_0 \times b_0\right)
\end{aligned}
\tag{1}
$$

As illustrated in Figure 29, the 4×4-bit array multipliers are internally wired with a combination of 4-bit partial product generators and full adders configured to give the proper sum and carry output vectors. These adders were initially configured in the logic structure seen in Figure 30. Each of these full adder blocks were replaced in subsequent iterations to yield proper comparison with this baseline design.

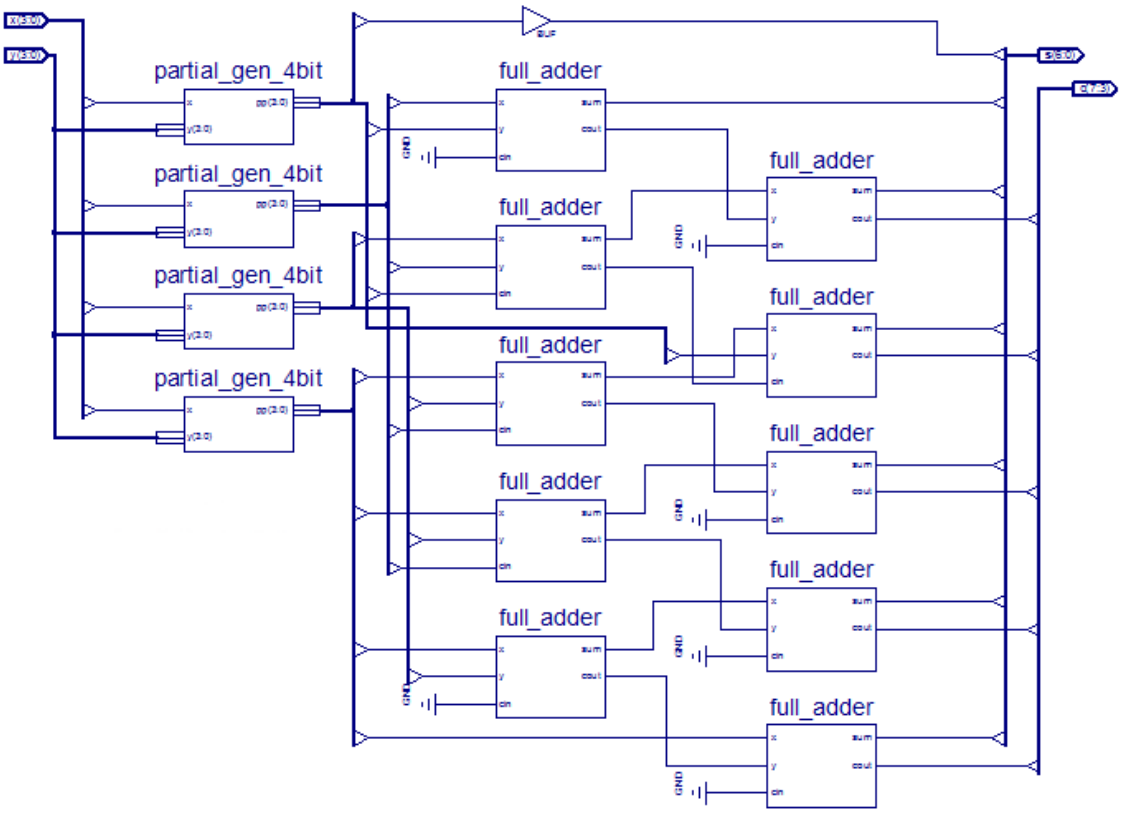


Figure 29. 4×4-bit array multiplier internal structure using partial product generators and full adders.

The 4×4-bit multiplier modules are combined with carry-save adders and ripple-carry adders to form the 16×16-bit multiplier in the top-level structure of the design. Specifics of the code and the overall top-level block diagram of the default 16×16-bit array multiplier structure are presented in Appendix D. This design is modified

slightly in the specific fault-tolerant schemes tested throughout the remainder of this chapter.

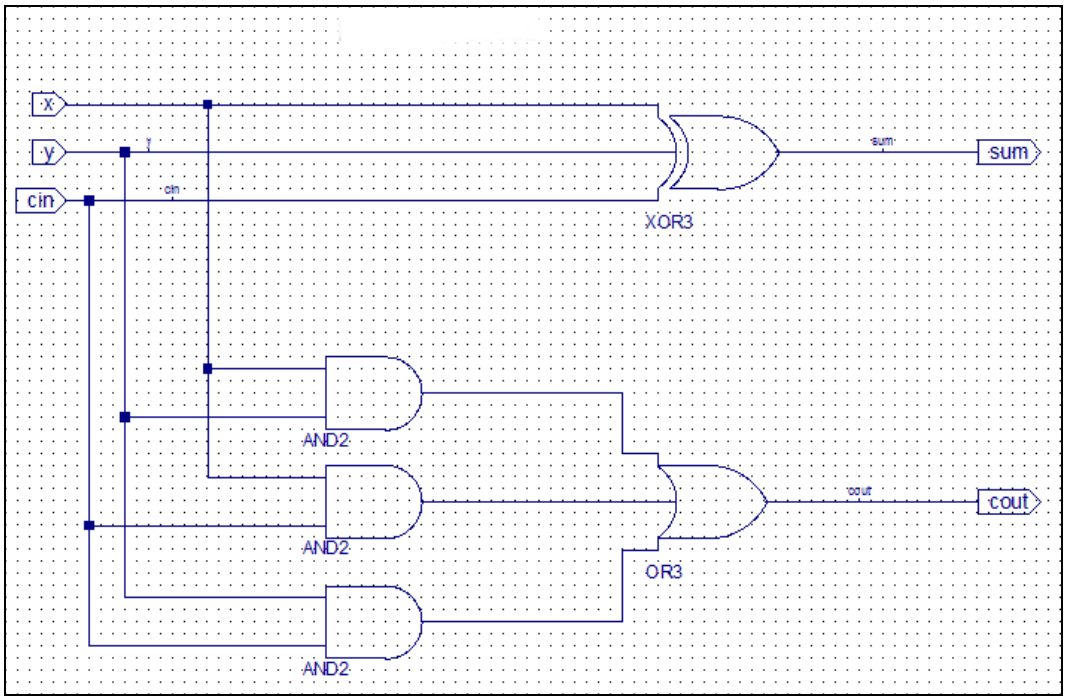


Figure 30. Full adder gate structure for the 16×16-bit array multiplier, baseline design.

The resource utilization of the FPGA after layout is seen in Figure 31. It is important to note that the layout of the FPGA can be manually edited at this stage.

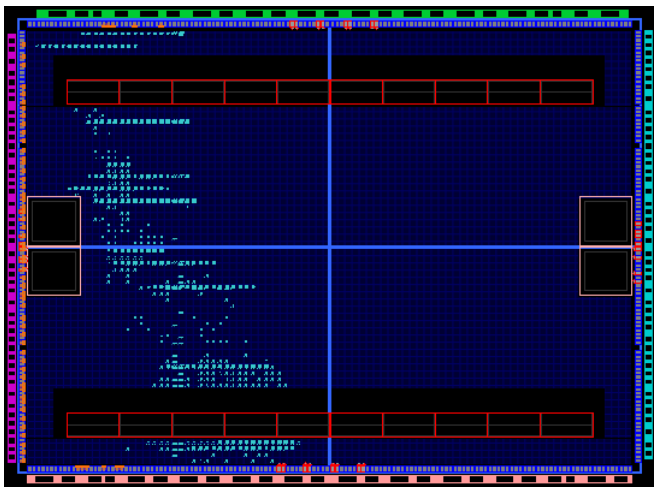


Figure 31. FPGA physical layout of default 16×16-bit array multiplier logic.

b. Triple Modular Redundancy (TMR)

The next iteration of design involved the implementation of the 16×16-bit array multiplier in TMR format. As illustrated in Figure 32, the design was modified in a similar fashion as that accomplished in the previous seven-segment LED counter example, with all Verilog code instantiated as schematic symbols. These were wired together with a voter on the output for a majority voting scheme of the output vectors.

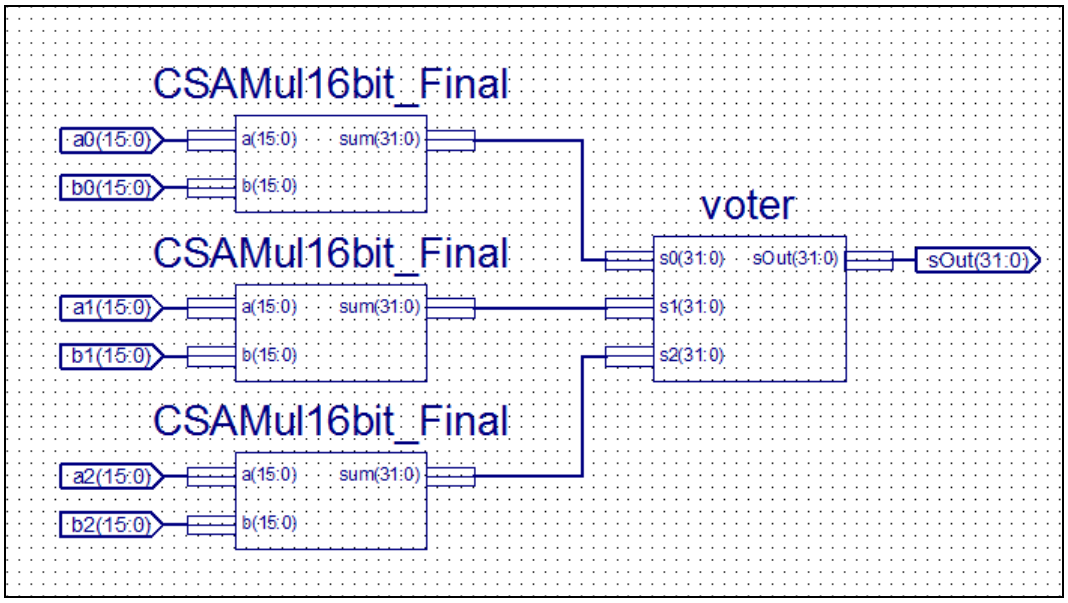


Figure 32. 16×16-bit array multiplier implemented in TMR with voting logic.

Again, with Verilog, some modification to the code is necessary to prevent trimming of logic that is desired for fault-tolerant purposes. In the case of Verilog, the code addition necessary is:

```
(* equivalent_register_removal = "NO" *)
```

This line must be placed immediately before the module declaration line in the Verilog file of each part of the design one does not want trimmed. In doing so, the FPGA layout of the device was seen to vary greatly from the unaltered case, as seen in Figure 33.

The input vectors for this design were left as three separate pairs, which allowed a differing input product to be injected into one of the array multiplier blocks.

This allowed for quick testing of the voting logic and overall multiplication when analyzed through simulation. In all cases, the proper simulation results were obtained and those of the mismatched multiplication discarded by the voting scheme. This initial simulation run was important in later fault-tolerance schemes to determine proper operation of the overall circuit.

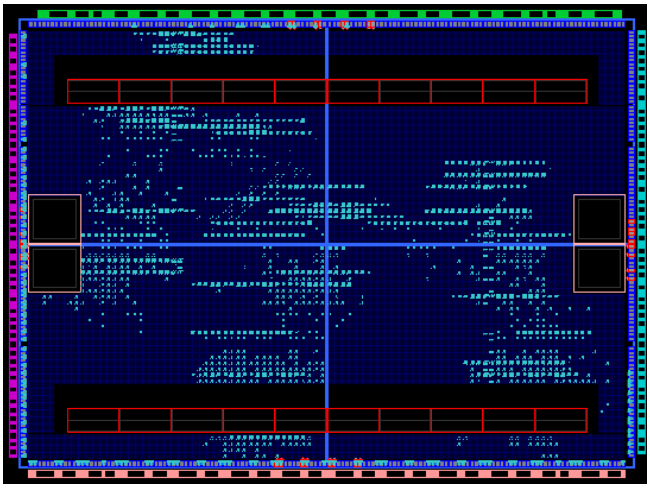


Figure 33. FPGA physical layout of 16×16-bit array multiplier using TMR logic.

c. Quadded Logic

To implement the quadded logic form of the 4×4-bit array multiplier, a NAND implementation of a half adder was constructed, as seen in Figure 34. This provided a good baseline for comparison and simulation to the quadded logic implementation; the half adder circuits were completely tested and simulated prior to being implemented in full adder combinations. To properly construct the OR gate for recombination of the carry output lines, a quadded NAND gate was also designed, as depicted in Figure 35.

Once the NAND configuration of the half adder and quadded format were determined, modifications to the half adder to generate a quadded logic implementation were made. This can be seen in Figure 36. Again, this half adder was fully tested and compared to the non-quadded logic format before insertion into full adder structures. Once the quadded logic full adder was complete, the structure was substituted in for the

full adder blocks in the overall 4×4-bit array multiplier schematic. The follow-on recombination into the appropriate 16×16-bit array multiplier structure then took place.

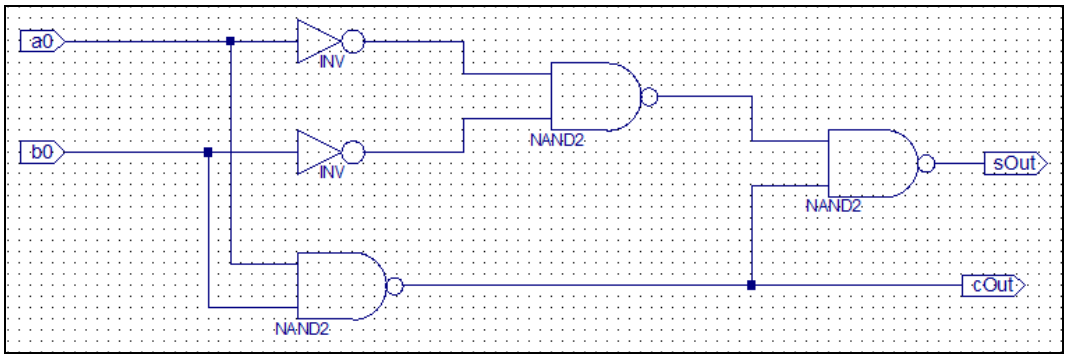


Figure 34. Baseline NAND implementation of half adder.

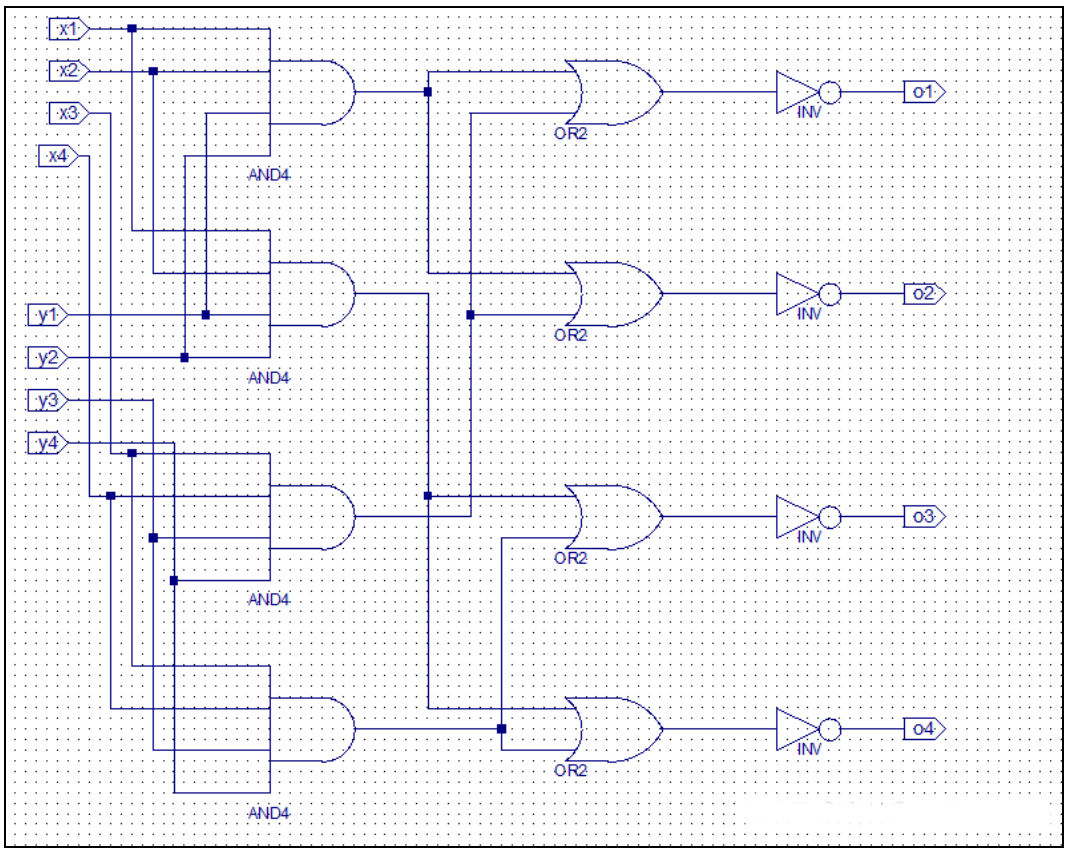


Figure 35. Quadded logic version of NAND gate.

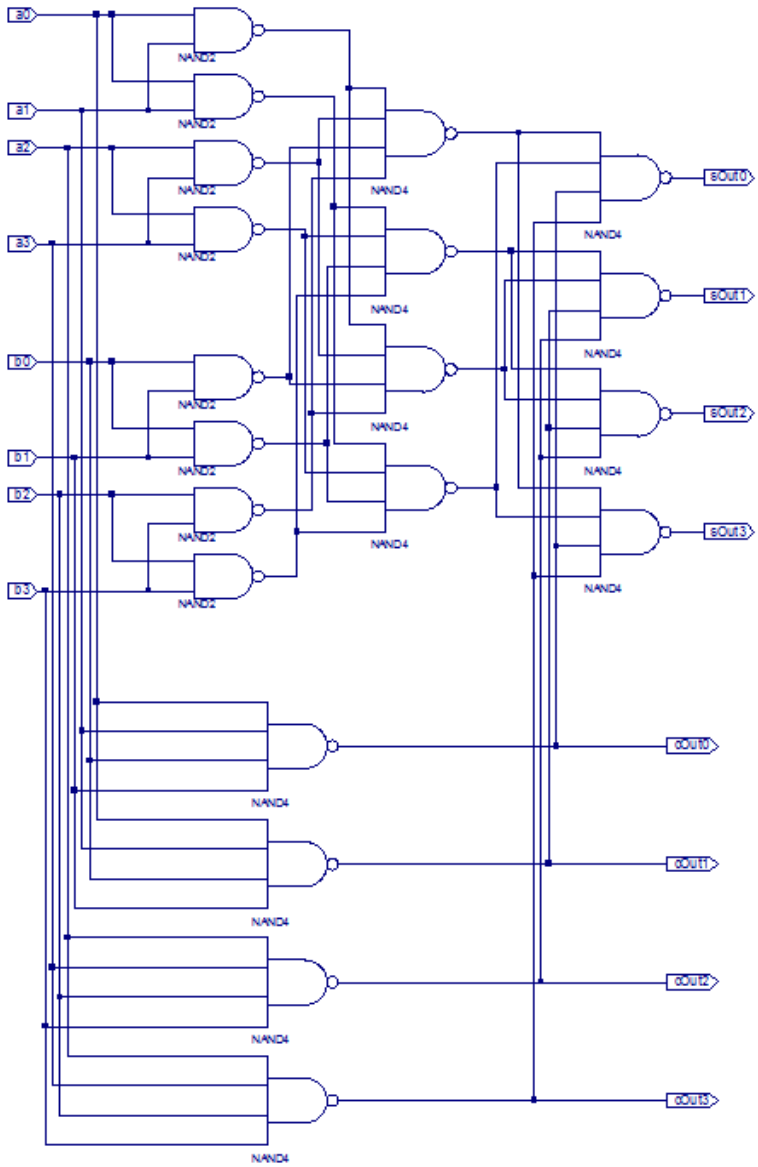


Figure 36. Quadded logic version of half adder.

As in the previous test cases, full simulation of the design was accomplished to account for proper outputs from the multiplier circuit. In all cases, the output product vectors were seen to match the baseline design. Further testing to alter the internal structure of the quadded logic half adders took place (i.e., tying certain gates to high or low logic values) to simulate internal errors. The quadded logic was seen to overcome these errors in all single and certain multiple upset cases. The outputs from the overall quadded logic circuit were identical to the non-error case.

It was particularly interesting to note the large amount of FPGA resources utilized in the layout view, as illustrated in Figure 37. This overhead is completely due to the quadded logic implementation of the full adders and should be approximately four times the resources as in the baseline 16×16-bit array multiplier.

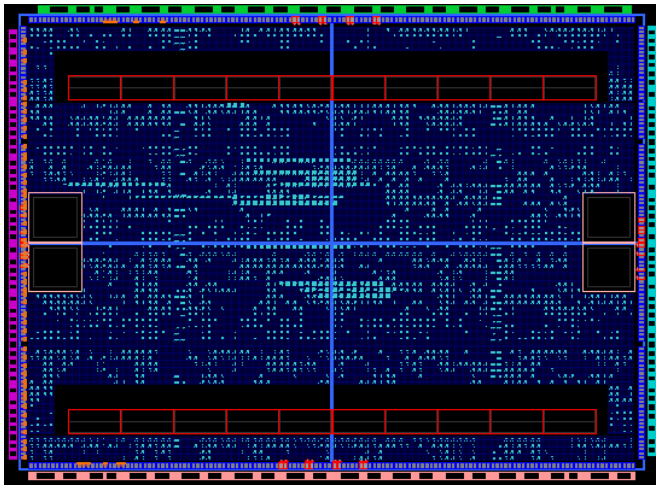


Figure 37. FPGA physical layout of 16×16-bit array multiplier using quadded logic.

d. Triplicated Interwoven Redundancy (TIR)

In the construction of the triplicated interwoven redundancy (TIR) case, the half adders within the 4×4-bit array multiplier tree were replaced with the TIR half adder design, as depicted in Figure 38. These half adders were combined into a full adder configuration with the signal lines recombined with a voter on the output. This allowed for a quick insertion into the 4×4-bit array multiplier tree, as previously indicated. The 4×4-bit array multipliers were again recombined into the overall 16×16-bit array multiplier tree structure to obtain the desired circuit.

Simulation was accomplished to verify the design against the unaltered multiplier configuration; simulation results were identical in both cases. The resource utilization of TIR in FPGA physical layout is illustrated in Figure 39.

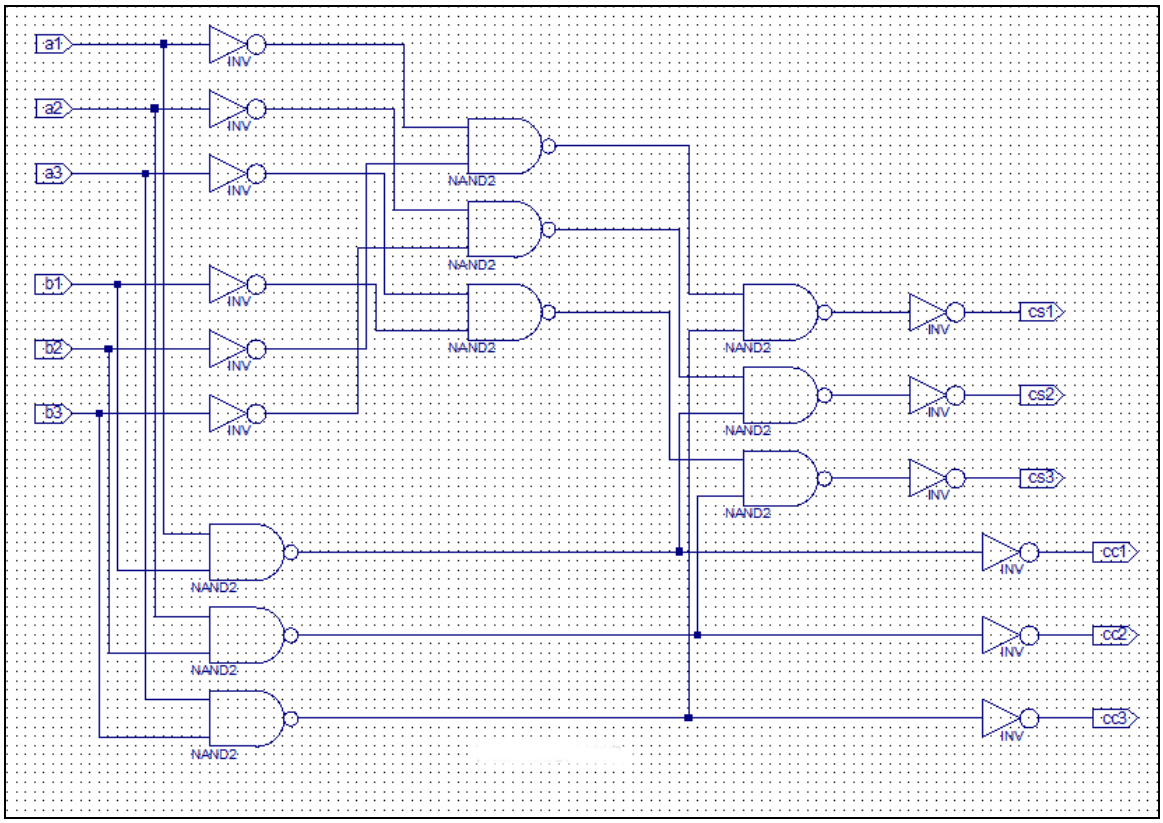


Figure 38. TIR logic version of half adder.

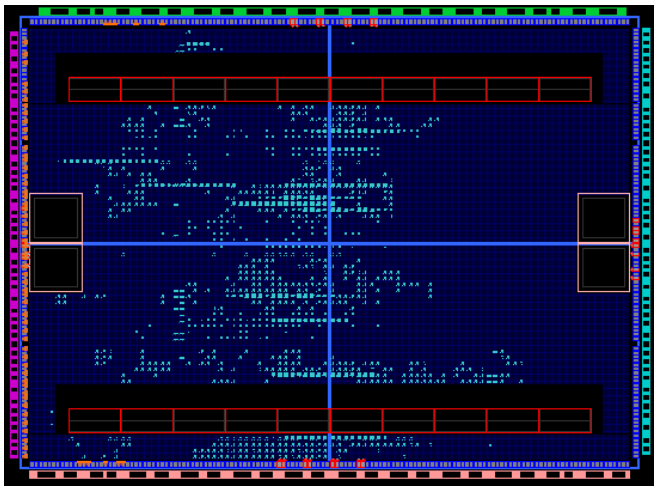


Figure 39. FPGA physical layout of 16×16-bit array multiplier using TIR logic.

e. Stacked Techniques – Quad-TMR & TIR-TMR

Since there was additional logic area on the FPGA for synthesis, a decision was made to attempt stacked fault-tolerant techniques to determine the limits of logic capacity and speed. This is similar to the manner in which encryption algorithms are stacked upon one another to increase the security of the encrypted electronic data. By stacking the fault-tolerant techniques, one should produce a more stable circuit in the event of multiple bit upsets or cascaded faults.

In the first case, the attempt to place the quadded logic implementation of the 16×16-bit array multiplier into TMR layout failed due to the physical constraints of the target FPGA. The overall resource utilization was such that 'OVERMAPPED' errors occurred in the map / place and route phases of the implementation. This was due to the volume limitations of the selected Spartan 3E FPGA device. The simulation of the device could still be accomplished and the results matched those of previous runs, indicating that both the multipliers and voter scheme were functioning properly.

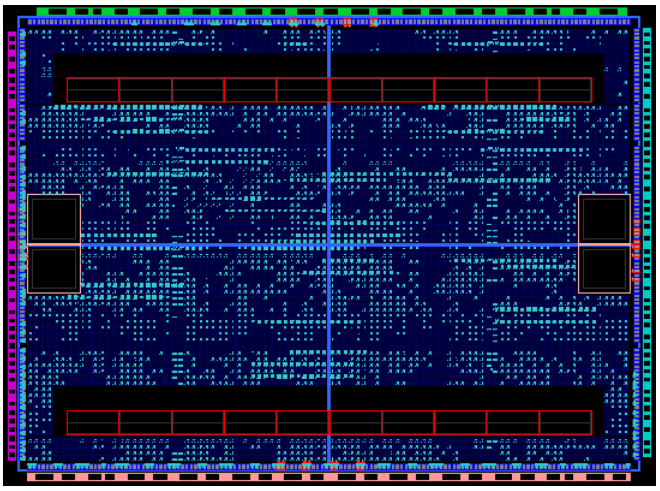


Figure 40. FPGA physical layout of 16×16-bit array multiplier using stacked TIR-TMR logic.

Finally, the TIR implementation of the multiplier was stacked upon the TMR layout for the final synthesis and implementation run. The FPGA physical layout

for this combinational technique is illustrated in Figure 40. Again, the simulation runs of the design indicated all circuits were functioning properly and as anticipated.

When utilizing these stacked fault-tolerant methods, it is difficult to predict the amount of logic resources that will fit in an FPGA's logic matrix. This is mostly due to the manner in which the software toolsets attempt to auto-optimize logic by combining and trimming for either speed or resource area utilization. Should one desire to target a design with these techniques, care must be taken to account for sufficient logic resources once the stacked fault-tolerant methods are employed.

3. Data Results and Analysis

The results of all six methods of fault-tolerance implementation were compared to obtain both FPGA resource utilization and timing information. The post-place and route (P&R) timing analysis consisted of a worst-case path measurement. The resources looked at consisted of the 4-input LUTs, number of occupied slices, number of logic LUTs, number of route-thru LUTs, and the average fan-out of the circuits. The results of both of these comparisons are presented in Tables 6 and 7, respectively.

Table 6. FPGA worst-case path timing for various fault-tolerant designs.

	Worst-Case Path [ns]
Default	38.54
TIR	42.15
TMR	38.71
Quadded	53.37
TIR-TMR	45.52
Quad-TMR	Fail to Map / P&R

Table 7. FPGA resource utilization for various fault-tolerant design methods.

	4-Input LUTs	Occupied Slices	# of Logic LUTs	# of Route-Thru LUTs	Avg. Fanout
Default	762 of 9,312 (8%)	389 of 4,656 (8%)	643	119	2.15
TIR	1,836 of 9,312 (19%)	949 of 4,656 (20%)	1,717	119	2.94
TMR	2,318 of 9,312 (24%)	1,196 of 4,656 (25%)	1,961	357	2.14
Quadded	4,445 of 9,312 (47%)	2,614 of 4,656 (56%)	4,325	120	3.29
TIR-TMR	5,540 of 9,312 (59%)	2,828 of 4,656 (60%)	5,183	357	2.93
Quad-TMR	13,361 of 9,312 (143%)	6,688 of 4,656 (143%)	13,001	360	3.28

These results are also presented graphically in Figures 41 and 42. The failed case for timing in the Quad-TMR scenarios was due to the aforementioned failure to map / place and route the design into the Spartan-3E FPGA. A retargeting of the design to a Xilinx Virtex 4/5 series was met with success; however, the resource utilization for this type of device was still high.

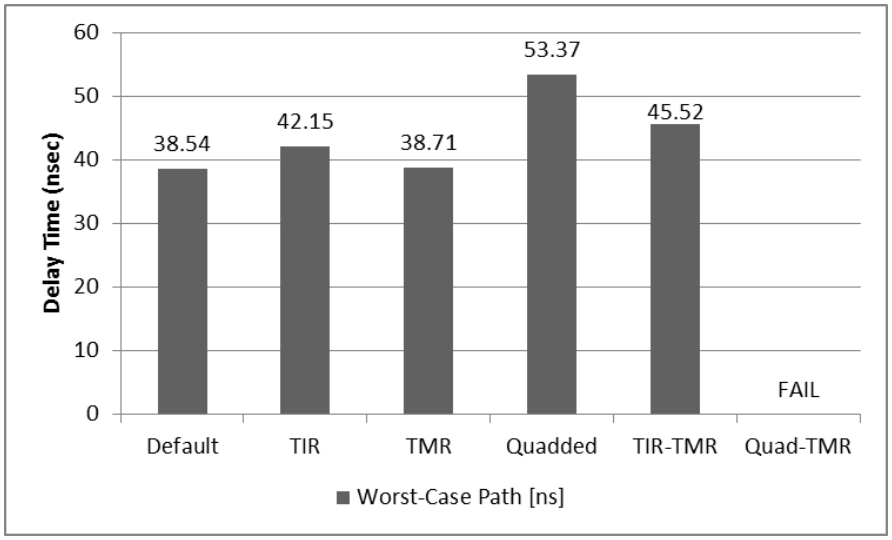


Figure 41. Timing analysis for worst-case paths in various fault-tolerant implementations.

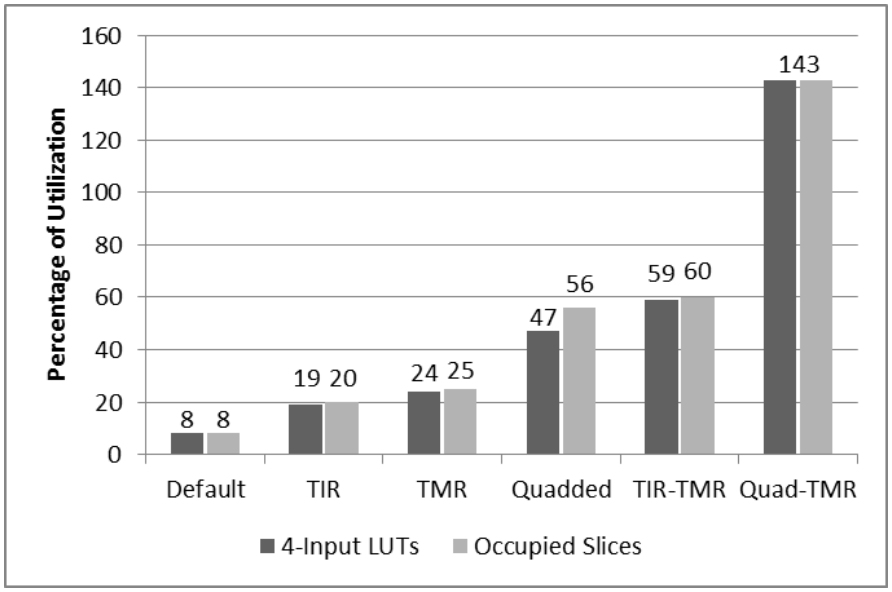


Figure 42. FPGA LUT and slice resource utilization in fault-tolerant implementations.

4. Discussion of Results

In looking closer at the timing analysis, it was interesting to note the variance in the TIR versus TMR design. Though both are similar (TIR being a version of modified TMR), the default and worst-case path timings were less in the TMR case. This extra 3.5 to 16.1 ns difference may not impact the design of a simple 16×16-bit array multiplier but could have greater implications in a more complex circuit or sequential logic. Likewise, the quadded logic and stacked fault-tolerant techniques were seen to have an even greater increase in timing delays. Again, these would have to be weighed carefully, dependent upon timing criticality and SEU tolerance benefit of the various techniques.

Similarly, the results of the FPGA resource utilization indicate that the TIR case, while having the increased timing requirements, uses less logic than the TMR case. The TMR case is fairly close to the TIR case—and has the added benefit of timing being fairly close to the baseline 16×16-bit array multiplier circuit. In the quadded logic case, the FPGA utilization was actually seen to consume more than the anticipated four times the amount of resources. This is likely due to the types of gates and LUTs necessary to synthesize this type of design. This is important to note, in that the extra work of wiring the quadded logic implementation of gates may not lead to the anticipated logic utilization. This, in combination with the markedly increased timing path delays, would preclude any but the simplest logic designs. One suspects that there would be added difficulties and overhead when incorporating quadded logic into sequential machines.

If the FPGA device family has sufficient resources, a better approach may be to choose the TIR-TMR stacked fault-tolerant technique. Both the path timing delay and resource utilization are similar to the quadded logic implementation. Any fault-tolerant benefits of stacked techniques would have to be compared to existing, baseline techniques both through fault injection simulations and in the presence of a radiation environment. Synthesis and implementation of the design through the map / place and route phases would be necessary before conclusively knowing the required FPGA resources; however, the construction of this design is greatly simplified when compared to quadded logic. Since the stacked techniques can be implemented without much

difficulty, future testing for possible benefits of stacked fault-tolerant schemes may be accomplished in a cyclotron.

Due to the path timing and FPGA resource utilization results, the best approach appears to be the complete TMR implementation of a design. This has almost exactly three times the required logic use plus the addition of the voting scheme. The timing being so similar to the non-modified case should yield excellent results in both combinational and sequential circuits. Future research should look more into the varying TMR techniques such as distributed versus global approaches. The different methods of clocking should produce varied results when used with sequentially clocked circuits. These methods should be combined with partial reconfiguration techniques, if necessary, to yield an optimal design for a particular scenario.

Additional work needs to be done in the realm of QFDR as it compares to quadded logic. Past research has shown some benefit to the application of this technique both with regards to timing and resource utilization. An open-source toolset known as RapidSmith exists to assist with this implementation. This should be a useful toolset—however, knowledge of Eclipse Java IDE and the interfaces between it and Xilinx ISE are necessary.

B. SELECTION OF A FAULT-TOLERANT TECHNIQUE

As a result of this study, the design of the sequencer being developed for NPSCuL will utilize Actel ProASIC3 flash-based FPGA logic in a TMR configuration. The sequential finite-state machines to control the timing and execution of deployments should benefit most from the decrease in path timing results and reduced logic utilization relative to other fault-tolerant techniques. Future work will analyze probabilities of failures, both in simulation and physically, in a radiation-exposed environment. Special attention needs to be given to the overall reliability measurements of the various fault-tolerant techniques, with particular emphasis placed on TMR versus quadded logic and the benefits of stacked combinational techniques.

One benefit to selection of the TMR technique in obtaining practical designs is the amount of commercial software available for implementation of these methods in

logic. While testing the TMR instantiations of redundant logic on combinatorial circuits, all designs were manually wired together in schematic form to achieve the results mentioned. One would desire to have these methods automatically inserted into a baseline design for quick implementation and comparison to manually developed methods. As such, a quick comparison of commercial products to manual TMR development is appropriate.

1. Xilinx TMRTool and Mentor Graphics Precision Hi-Rel

Two of the commercially available, fault-tolerant synthesis products available for the design of TMR logic are Xilinx TMRTool and Mentor Graphics Precision[10] Hi-Rel. The purpose of both these products is similar; an increase in designer productivity by reducing errors and speeding the time for TMR implementation. The main difference between the two tools is the family of products the software is intended for. TMRTool is only targeted toward Xilinx FPGAs; although, it works with all entry methods including schematics and HDLs [49]. Precision Hi-Rel, however, is a multi-vendor, multi-mode product [50]. While TMRTool is listed as an ITAR-controlled product [49], Precision Hi-Rel does not appear to have similar restrictions. Since the design of sequencer logic is taking place upon Actel ProASIC3 hardware, Precision Hi-Rel is the most suited application for further use.

Precision Hi-Rel offers fault recovery and fault-tolerant FSM encoding through a variety of features. This FPGA synthesis product was developed under NASA guidance. Precision Hi-Rel provides an enhanced form of safe FSM that offers full SEU detection and recovery [50]. The generated FSM, when it detects an invalid transition, must still go through a recovery process including an operational reset that may consume one or more clock cycles. Thus, Precision Hi-Rel also includes fault-tolerant FSM implementations which can absorb an SEU and continue operation without interruption [50]. Precision Hi-Rel also allows for selection of LTMR, DTMR, and GTMR methods and handles all embedded resources.

[10] Precision® is a registered trademark of Mentor Graphics.

The Precision Hi-Rel synthesis tool is not included as part of Microsemi's Libero SoC package. Instead, the default synthesis tool provided with Libero SoC is Synplify Pro. Synplify Pro does not have any of the added TMR capabilities of Precision Hi-Rel. Application of Precision Hi-Rel to the sequencer project will require a separate software license, as an academic or trial version is not available for educational institutions. The author recommends that future design work on the sequencer incorporate the use of this software package, as it would allow for testing and comparison with the manually inserted TMR methods. The ability to quickly switch to LTMR, DTMR, and GTMR modes in logic would be greatly beneficial when testing the final design in a radiation environment. Additionally, it will allow for ease of migration of more complex logic designs to TMR instantiations once expanded features are added to the sequencer.

2. Manual Insertion of TMR with Schematic and HDL Techniques

Due to the lack of available commercial software during the performance of this research, all designs were implemented manually using the LTMR methodology. In this method, all logic elements were triplicated with error correction performed by one voter on the output path. Certain features of DTMR were carried forward into the sequential logic development, such as feedback path voting being triplicated and tied to all triplicated structures. The clock input to the sequential logic and single voter output can still be seen as limiting factors in the application of this technique to the final sequencer logic design. It would be greatly beneficial to obtain the benefits of full DTMR with the triplicated clock signals of GTMR in the final representation of this sequencer logic.

C. APPLICATION OF TMR TO SEQUENTIAL LOGIC DESIGNS

After deciding upon TMR as the appropriate fault tolerance method, based on the experimental results in the combinational logic testing earlier in this chapter, the TMR technique was implemented in sequential logic. Many of the means of constructing TMR sequential logic were very similar to that utilized previously. All logic elements within the FSM design were triplicated with a voting scheme on the output. There was one important difference due to the addition of logic to the feedback paths within the FSMs. The FSM chosen for triplication was a clocked synchronous state machine of Moore

design, similar to that found in [76]. This allows for a predictable state transition with plenty of points to test circuit output or allow for fault injection in testing. The schematic representation of the FSM under test is illustrated in Figure 43.

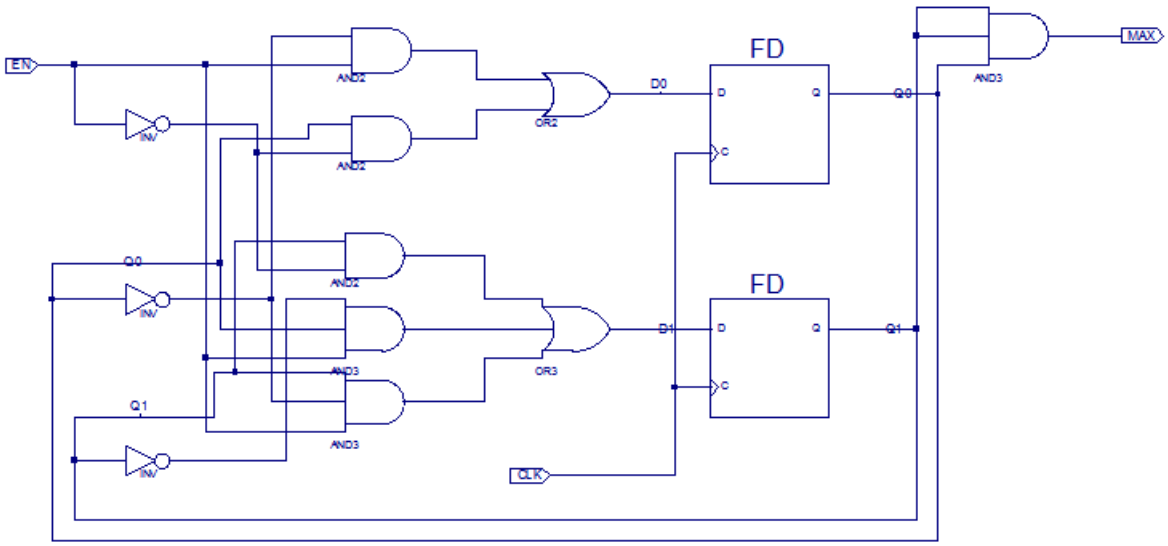


Figure 43. Moore design, clocked synchronous state machine utilizing positive-edge triggered D flip-flops (After [76]).

The resultant output of the post-place and routed design was run through a simulation utilizing a testbench file. The code for the testbench can be found in Appendix D. The resultant timing diagram is illustrated in Figure 44. The top lines of the timing diagram corresponds to the output of the FSM (MAX), with the clock signal (CLK) and enable (EN) signal lines following. The next four signal lines of the timing diagram (highlighted in blue) are the inputs and outputs of each of the two D-FFs – D0, D1, Q0, and Q1. The output of the circuit is seen to go high whenever Q0 and Q1 are high, corresponding to D0 and D1 falling on the positive-edge of the clock signal. The fault-free case of this timing diagram output appears as anticipated with the state transitions expected.

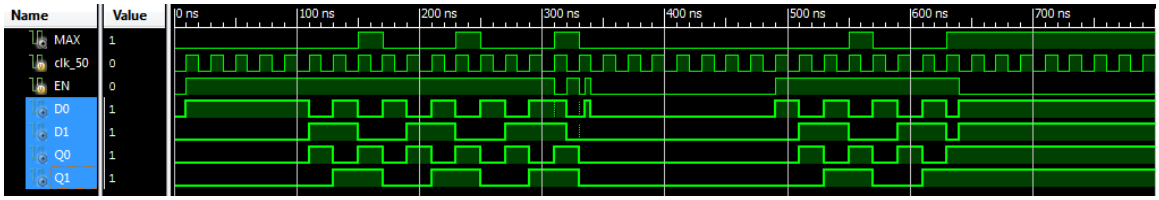


Figure 44. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (fault-free case).

After sequential behavior for the fault-free case of the FSM was obtained through simulation, a decision was made to implement fault cases for comparison in the non-TMR circuit.

1. Manual Insertion of Faults

For the initial testing on logic timing results due to error cases, there were three trials done to determine effects of error: floating-input, 'stuck-at-0,' and 'stuck-at-1.' These were inserted into the circuit on the second input of the second AND gate tied to the top OR gate. This would lead to an error in D0, dependent upon the combination of fault with the other passed logic values.

a. Floating-Input Case

For the floating-input case to the AND gate, the second input line was disconnected with no attachments, as illustrated in Figure 45. Breaks in the circuit similar to this case commonly result in non-valid logic values in the output represented by 'x' and a red floating line in timing diagrams.

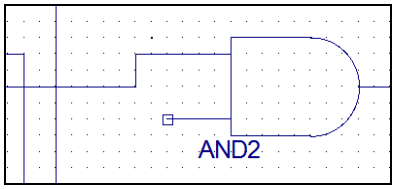


Figure 45. Floating-input case of AND gate second input disconnected, simulating break in circuit or disruption in input.

The resultant timing diagram for the floating-input case is seen in Figure 46. As anticipated, the logic values for both the overall circuit output and the intermediate D-FF inputs and outputs are inconclusive logic values. This floating break in the circuit leads to complete break-down of the overall FSM with inconclusive data returned. A disabling error such as this in the non-TMR case could produce erroneous or no output.

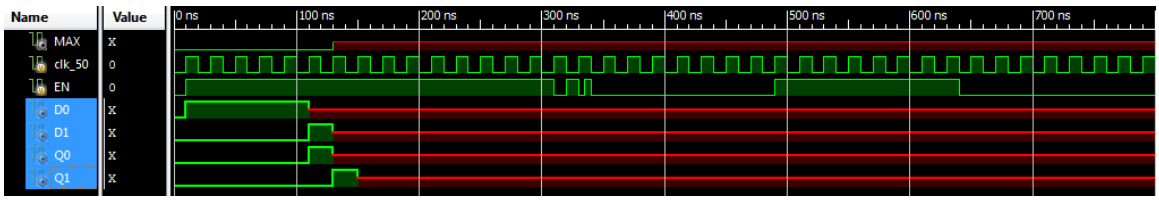


Figure 46. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (floating-input case).

b. 'Stuck-at-0' Logic Fault

For the constant logic '0' or 'stuck-at-0' case to the AND gate, the second input line was pulled down with a biasing resistor tied to ground, as illustrated in Figure 47. This simulates a constant value of binary zero to the second input of the AND gate, leading to incorrect output to the OR gate in certain cases.

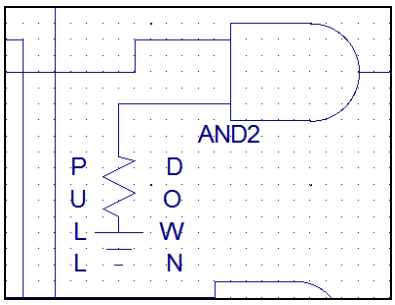


Figure 47. Second input of AND gate tied to logic '0', simulating constant logic zero or short in circuit to ground plane.

The resultant timing diagram for the 'stuck-at-0' case is seen in Figure 48. There are some noticeable differences in this timing diagram in comparison to the fault-free case in Figure 44. While the output (MAX) appears to go high in the expected manner, a closer look at D0 and Q0 reveal the error injection as they do not always go

high where anticipated. As such, the output does not remain high over the last 150 ns as in the fault-free case. An error such as this in the circuit, where a line is shorted to ground by an SEL, could lead to erroneous output timing for the sequencer. This could lead to individual CubeSats being deployed at the inappropriate time or missed deployments altogether.

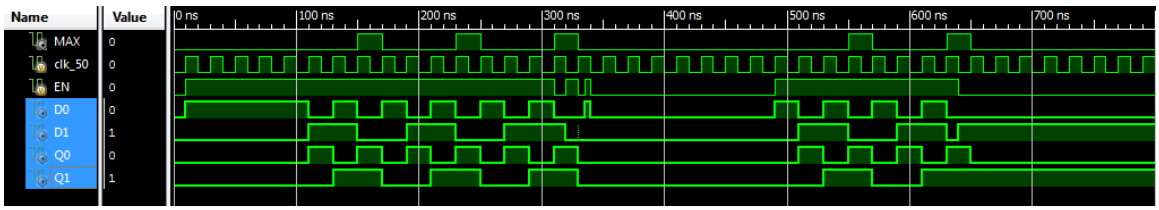


Figure 48. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (constant logic ‘0’ case).

c. ’Stuck-at-1’ Logic Fault

For the constant logic ‘1’ or stuck-at-1 case to the AND gate, the second input line was pulled up with a biasing resistor tied to V_{CC}, as illustrated in Figure 49. This simulates a constant value of binary one to the second input of the AND gate, leading to incorrect output to the OR gate in certain cases.

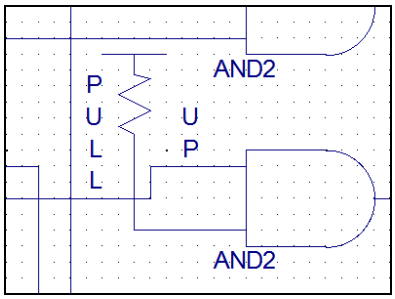


Figure 49. Second input of AND gate tied to logic ‘1’, simulating constant logic one or short in circuit to positive voltage plane.

The resultant timing diagram for the ’stuck-at-1’ case is seen in Figure 50. Again, there are some noticeable differences in this timing diagram in comparison to the fault-free case in Figure 44. The output (MAX) appears to go high in many unexpected

places as the incorrect value is passed forward by the AND gate through to D0 and Q0. D0 and Q0 get stuck at a logic high value which creates oscillations in the output state of the circuit. These outputs eventually affect D1 and Q1, as Q0 is tied to the second OR gate tree via a feedback path. This will most assuredly lead to individual CubeSats being deployed at the inappropriate time and a complete loss of sequencer function. An SEL of a signal line creating this type of condition is a catastrophic event.

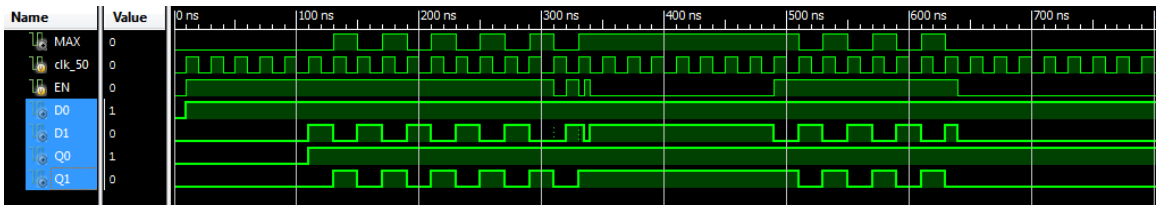


Figure 50. Timing diagram for the Moore-type FSM developed with output, clock, enable and D-FF input/output signals visible (constant logic ‘1’ case).

2. Modification of FSM Design to TMR Representation

Once the methods of fault injection were developed and fully tested for the non-TMR FSM implementation, work was done to develop the design to a TMR instantiation. This was done by voting on the output path, similar to the combinational logic methods utilized earlier. In addition, voters were added to the feedback paths to each FSM in the triplicated logic. This negated any errors generated in propagating back to the next state. The triplicated version of the Moore-type FSM is illustrated in Figure 51.

Initial testing of the TMR FSM yielded results identical to the non-TMR version of the circuit, as seen in the timing diagram of Figure 52. This verified the proper operation of the modified circuit with exact timing sequences matching the non-TMR case. To properly test for the fault cases, scenarios were run using each of the three manual, fault-injection test cases as previously specified. For the constant logic ‘0’ and constant logic ‘1’ cases, the output was seen to correct itself regardless of the intermediate states of D0, Q0, D1, and Q1. The more interesting of these cases, constant logic ‘1’, which yielded so many timing errors in the non-TMR instantiation, is illustrated in Figure 52.

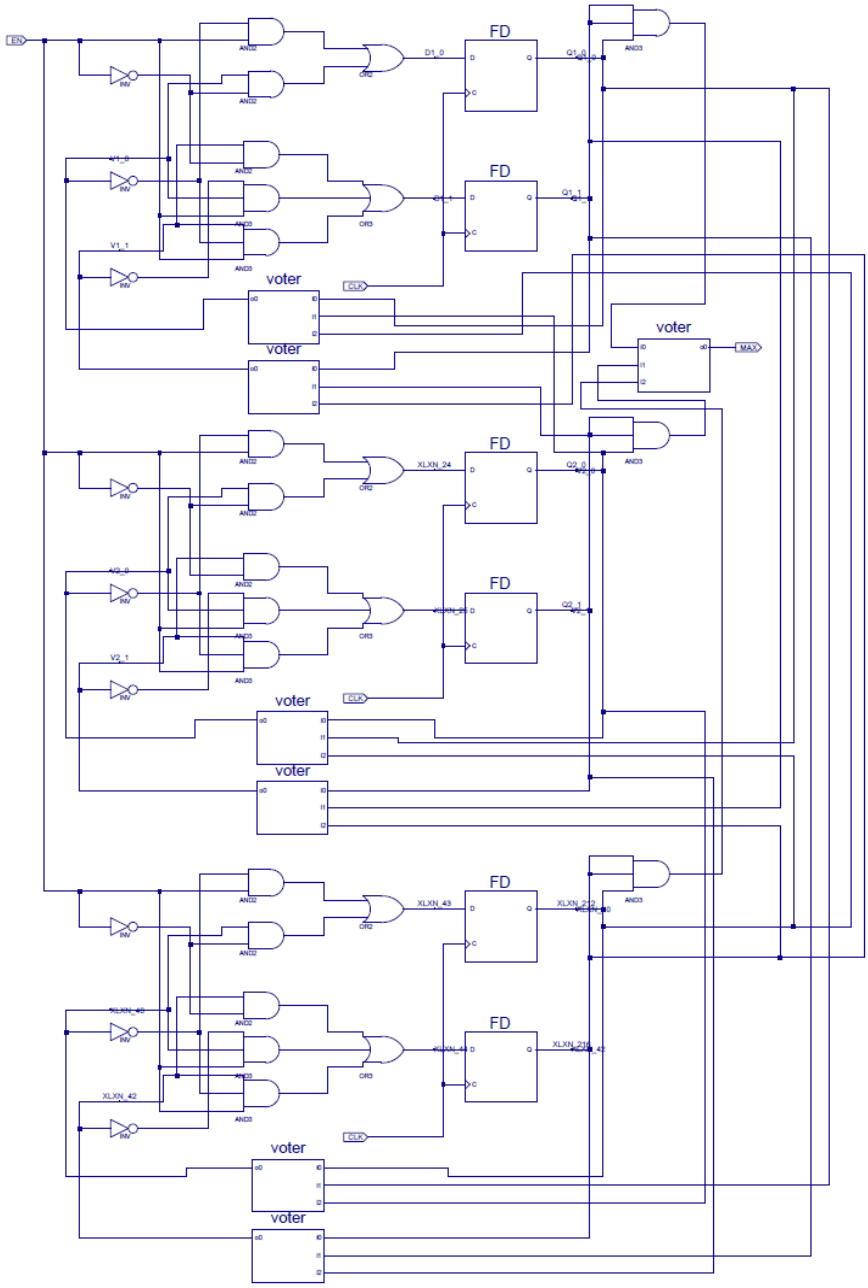


Figure 51. TMR representation of the Moore-type FSM with voting logic.

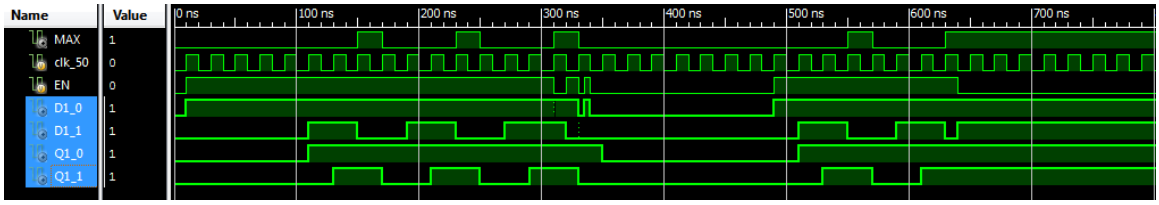


Figure 52. Timing diagram for the Moore-type, TMR version of the FSM developed showing output signal correction regardless of errors (constant logic ‘1’ case).

Equally as interesting was the manner in which the TMR version of the FSM corrected for the floating-input case. While the floating input lines may still be seen in the D0 and Q0 input and output of the first D-FF, the signal is corrected by the time it reaches the output, MAX. Additionally, the voting logic is working on the D0 and Q0 signals to restore proper operation when possible. The amount of time left in the non-valid logic states was minimized. This is illustrated in Figure 53.

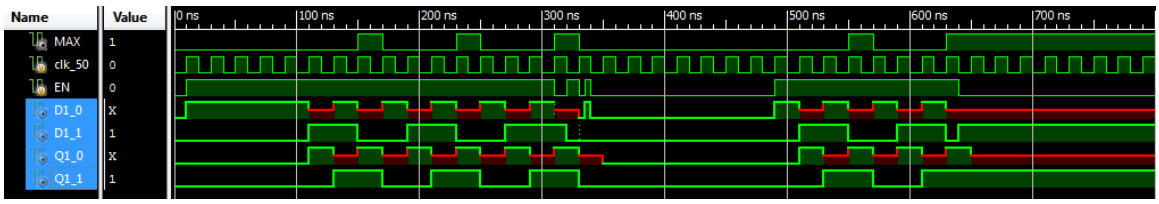


Figure 53. Timing diagram for the Moore-type, TMR version of the FSM developed showing output signal correction regardless of errors (floating-input case).

3. Timing and Resource Comparison for Sequential Circuits

Another analysis of the timing and resource utilization between the two designs was made prior to proceeding to the next phase of logic design. Once verification of the TMR technique was made for a general Moore-type FSM, one desires to obtain accurate timing information to further validate the approach. In the case of the non-TMR representation of the FSM design, an average delay of 4.9 ns across three levels of logic was measured. This corresponds to a clock frequency of up to 204 MHz. Worst case delay across four levels of logic was found to be 5.5 ns due to the input and output buffer delays on the FPGA. After the place and route phase, net skew was measured as 0.0 ns with a maximum delay of 0.2 ns. Setup time was 1.9 ns and hold time 1.3 ns; thus, considering the sequencer will be under-clocked to conserve power at a frequency of approximately 50 MHz, any necessary timing constraints are met.

For the TMR version of the FSM, the same measurements were conducted after the synthesis and place and route phases of design. An average delay of 6.5 ns across four levels of logic was measured. This corresponds to a clock frequency of up to 154 MHz. Worst case delay across four levels of logic was found to be 6.8 ns, again due to input and output buffer delays. After place and route, net skew was measured at 0.0 ns with a

maximum delay of 0.2 ns. Setup time was 3.1 ns and hold time 2.0 ns; again, any timing constraints necessary for the design are met.

The data above indicated that the added delay of the voter circuits introduced in the TMR version of the FSM is responsible for the increased timing results and subsequent reduction in clock frequency. This may be somewhat further affected by the delays of the output buffers, as they do not appear in the feedback paths of the TMR version. Most interesting was the increase in setup and hold times between the non-TMR and TMR versions of the FSM. While the increase of approximately 0.5 to 1 ns should not affect the operation of the simple sequencer application considered in this research, more complex functions such as DSP algorithms or more complex processing schemes may have to take similar timing increases into account. Complete verification of the data timing sequences would need to be carefully analyzed in either of these two scenarios.

A difference in resource utilization between the two designs was as anticipated. The non-TMR, baseline version of the FSM design utilized two of 9,312 FFs, three of 9,312 LUTs, and two of 4,656 slices with an average fan-out of 2.00. The TMR version used six of 9,312 FFs, eleven of 9,312 LUTs, and six of 4,656 slices with an average fan-out of 2.50. In both these baseline and TMR instantiations, the resource utilization for FFs and LUTs was well under 1% of the total available quantities. These resource utilization values should maintain an approximately three factor increase of TMR over the baseline design, as the complexity of the logic design increases. The three-fold increase of the TMR design in resource utilization from a baseline version will be readily apparent in percentages the larger a logic circuit becomes.

The timing and resource utilization comparisons between the non-TMR and TMR forms of sequential logic revealed that either design may be used for the sequencer design. The additional setup and hold time, slower clock speed available, and slightly increased path delay will have little to no effect on the sequencer operation. The additional fault-tolerant features of TMR make it the best solution for the sequencer project while requiring a factor of three to four times the resource utilization for the design.

D. PRELIMINARY SEQUENCER LOGIC DESIGN

After testing the validity of the TMR approach in sequential logic, design work proceeded with specific development of logic for use in the sequencer application. Transition of design work was carried over from the Xilinx test cases to Actel specific designs using Libero SoC. The Actel ProASIC3 development board was used for the first sequencer logic testing, as seen in Figure 54.

1. Conversion of Xilinx HDL to Actel HDL

One of the first challenges in carrying schematically-designed circuits from Xilinx ISE to Actel SoC is the manner in which logic primitives are specified in the two products when HDL is generated. This requires manual editing when porting the HDL specifying a Xilinx schematic design over to the Actel SoC development suite. For example, in Xilinx ISE the instantiation of a two-input NAND gate is as follows:

```
NAND2 XLXI_4 (.I0(XLXN_4),
              .I1(XLXN_3),
              .O(XLXN_5));
```

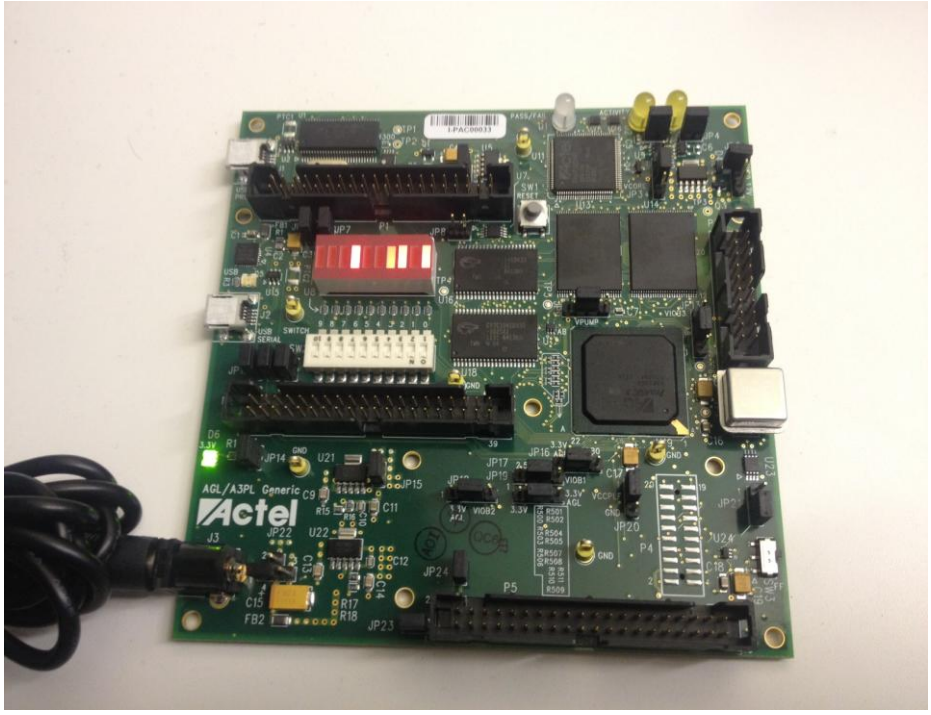


Figure 54. Actel ProASIC3 development board showing sequencer LED testing with TMR logic implemented.

To properly port this design over to an Actel equivalent representation of the two-input NAND gate, the signal inputs and output must be renamed to the appropriate variable term. In this instance, .I0 becomes .A, .I1 is renamed .B, and .O translates to .Y for the following representation:

```
NAND2 XLXI_4 (.A(XLXN_4),
              .B(XLXN_3),
              .Y(XLXN_5));
```

While the instance can be renamed from XLXI_4 to something else, leaving this device name helps with the realization that the code is ported from Xilinx to Actel representation. Comparing the two source code equivalents aids in troubleshooting the automatically generated HDL should errors occur. This may not seem like a great deal of work until one considers the amount of logic typically involved in complex designs. Since the design of the sequencer is implemented on the ProASIC3 family, knowledge of Verilog is critical to the success of the project; once the initial work is done schematically, time must be taken to understand any generated HDL code prior to porting the design to Actel SoC.

2. Synplify Pro Representations of FSM Logic

Once logic for an FSM is implemented in Actel SoC, there are a variety of synthesis tools available in Synplify Pro to aid in the realization of circuits. Instantiated logic can be viewed as a register-transfer level (RTL) or technology schematic. These views within Synplify Pro are illustrated in Figures 55 and 56 for a simple FSM.

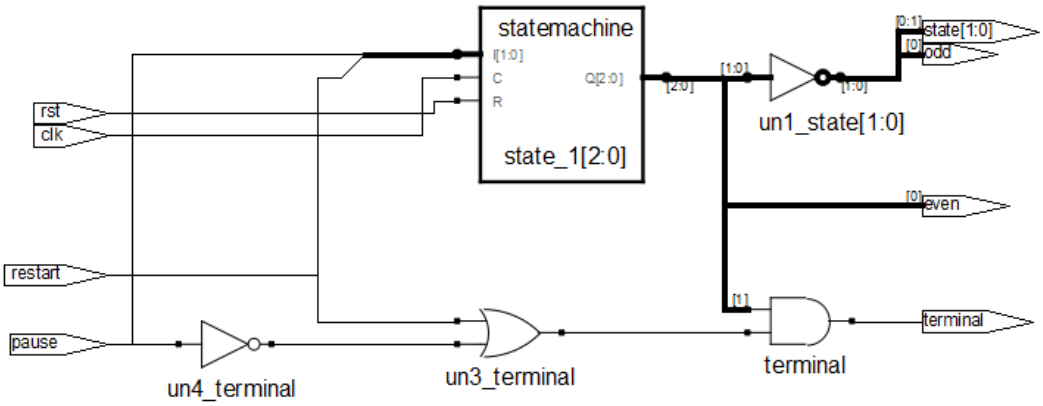


Figure 55. Synplify Pro RTL view of instantiated FSM logic.

The program also features an FSM explorer which may be utilized in the generation of state tables and state diagrams for any generated design. The state table and diagram for the FSM implemented is seen in Table 8 and Figure 57. All code for this basic, non-TMR FSM can be found in Appendix D.

The appeal of these tools is immediately apparent as they are beneficial in troubleshooting and verifying reference design material with generated output. During design of the TMR version of FSM logic, cross-referencing between the two state machines is possible in utilizing the features of Synplify Pro. Should the switch to Precision Hi-Rel occur in future testing, one may be able to check the output of that synthesis tool (various TMR representations of a base logic design) within Synplify Pro to verify expected output and state transitions.

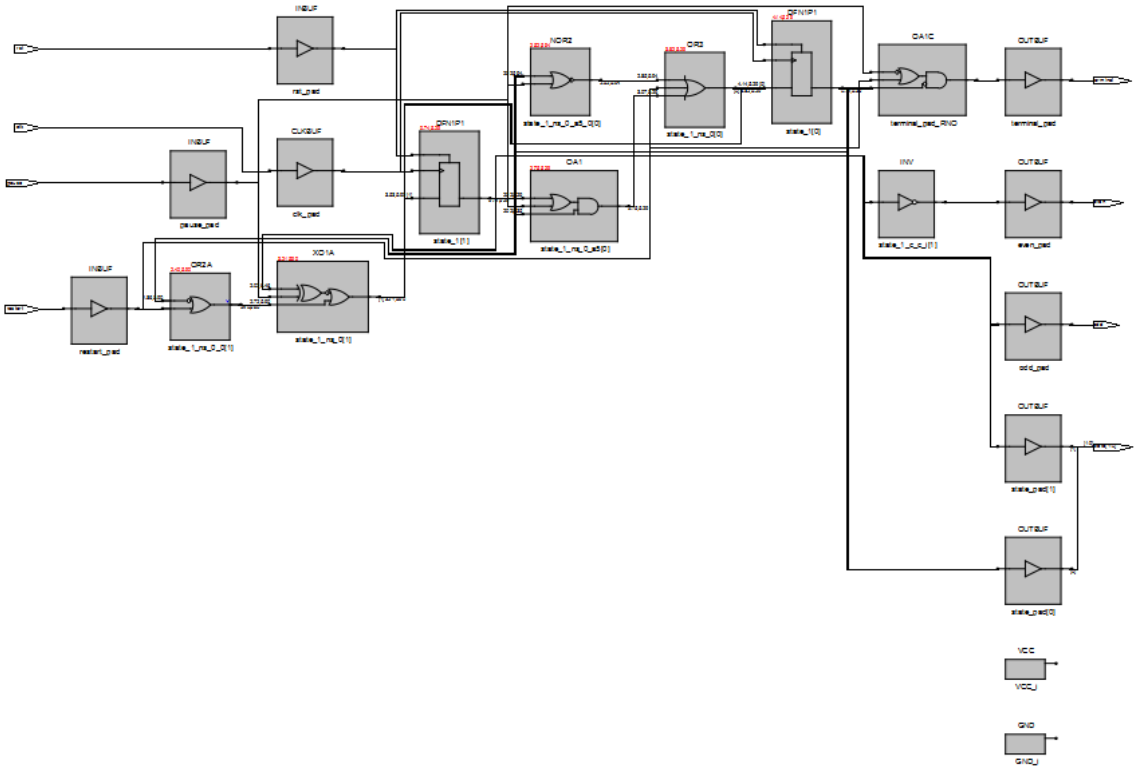


Figure 56. Synplify Pro technology view of instantiated FSM logic.

Table 8. Synplify Pro generated state table for FSM logic.

```
Number of states: 3
Number of inputs: 2
Inputs:
   0: restart
   1: pause
Clock: clk
Reset: rst to state S2
Transitions: (input, start state, destination state)
             1-     S0            S2
             00     S0            S1
             01     S0            S0
             1-     S1            S2
             -0     S1            S2
             01     S1            S1
             1-     S2            S2
             -1     S2            S2
             00     S2            S0
```

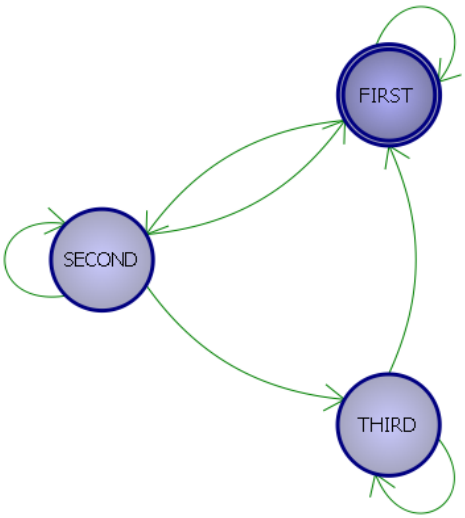


Figure 57. Synplify Pro generated state diagram for FSM logic.

3. Designation of Timing Constraints within Logic

In defining the timing variables for use in deployment, there are two possibilities for storage—saving the timing constants as parameters in Verilog and using the FlashROM feature of the ProASIC3 FPGA to store constant values.

a. Parameter Constants in Verilog HDL

In Verilog, parameters are constant values and do not incorporate any other data types such as nets or registers. In defining these parameters during the initial portion of Verilog code, one is able to rapidly change the values and operation of a module prior to re-synthesis and configuration of a target FPGA device. These parameter values are not modifiable at runtime and can act as the timing constants necessary to control the deployment sequence. The following code illustrates the use of parameter values within a Verilog module:

```
// Module – Test Module

module test_Mod (Clk, D, Q);
      parameter   width = 2,
                  delay = 10;
      input [width - 1 : 0] D;
      input Clk;
      output [width : 0] Q;
      assign #delay Q = D;
endmodule
```

In this manner, all eight of the timing constants defining the deployment sequence for the various P-POD deployments may be specified. Once the ‘Arm’ power signal activates the sequencer and the ‘Deploy’ command is internally generated, the sequencer FSM will execute the deployment sequence using the constant variables defined at the start of the Verilog declarations. To modify the timing sequence, only a quick re-declaration of parameters must occur prior to re-synthesis and implementation. The generated bitstream file for FPGA reconfiguration takes only minutes to produce. All interfacing and reprogramming may be accomplished through the JTAG port at any time prior to launch, when physical access to the sequencer is available.

b. Constant Declarations in ProASIC3 FlashROM

The ProASIC3/E devices have a dedicated nonvolatile Flash ROM (FROM) memory of 1,024 bits for storage of constant values or serial number identification. This FROM can be read, modified, and written using the JTAG interface; however, it can be read but not modified from the FPGA core. The FROM is logically organized as eight pages of 16 bytes and physically organized 8×128 bits [77]. Actel

ProASIC3/E devices are the only FPGA to support this feature as only flash-based technology supports reconfigurable, nonvolatile memory.

The advantage of this FlashROM is immediately seen in any application utilizing the FROM. Since the FPGA can be configured using HDL code with a portion referring to the constants stored in FROM, any changes via JTAG interface to the constant values do not require complete reconfiguration of the FPGA. The main portion of the synthesized and implemented FPGA configuration can be left untouched with only a simple flash re-programming of the FROM. This reduces the time required to redefine variables for sequencing from minutes to seconds.

Designation of FlashROM contents from within Actel SoC is a simple process of adding the appropriate module ('FlashROM_cmp') to a project requiring its use. The IP core is then instantiated as follows to call the FlashROM variables from the HDL code:

```
----------- FlashROM ip core instantiation -----------
-------------------------------------------------------
--FlashROM_cmp_inst: This FlashROM block is configured
--to generate different time intervals between launch
--select signals
-------------------------------------------------------

FlashROM_cmp_inst: FlashROM_cmp port map
      (
         CLK  => from_en_sig,
         ADDR => addr_sig,
         DOUT => dout_sig
      ) ;
```

The above example is modeled after an eight chip power sequencer reference design provided by Actel [78]. The decimal values stored in the first eight bytes of page zero in the FlashROM are illustrated in Figure 58. The design of this reference power sequencer is very easily modified to fit the requirements of the CubeSat launching application. All that remains is the modification to TMR implementation of logic using manual or commercially available means along with radiation testing. Preliminary code for the CubeSat sequencing application is provided in Appendix D.

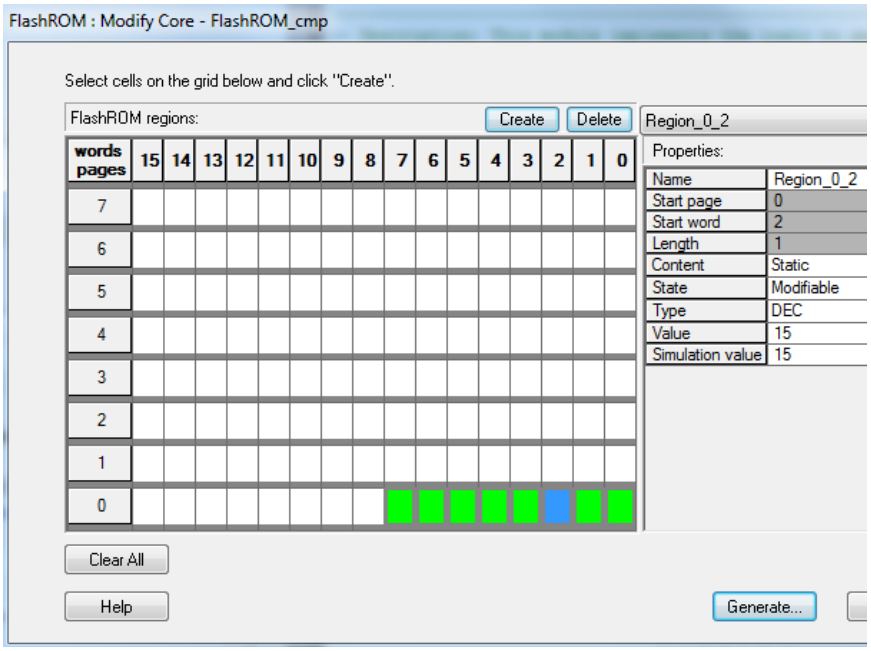


Figure 58. FlashROM storage of eight bytes defining sequencing timing; the third value (0x2) is highlighted containing decimal value '15.'

E. CHAPTER SUMMARY

The methods of implementing fault-tolerant techniques within logic leading to final selection of a particular method were presented in Chapter V. All combinational and sequential circuits were compared for accuracy in output, FPGA resource utilization, timing characteristics, and reliability for correcting errors. TMR was seen as the best solution for both combinational and sequential circuits due to small overhead requirements and small increases in logic delay. Discussion of manual and automatic fault injection techniques was provided.

Further research was conducted toward developing the sequencer logic for use in the developed ProASIC3 design. Work was carried over from Xilinx ISE to Actel SoC with porting methods discussed. An overview of Actel SoC requirements for logic development was provided with suggestions for future code design. Differences in storage methods for sequencer timing values were discussed, including storage of variables within HDL code or in the internal FlashROM of the FPGA. The use of a power sequencer reference design for the ProASIC3 family, coupled with TMR logic, is seen as the way forward for the sequencer application.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

A. CONCLUSIONS

During the development of a general payload processor, there were several items noted which vary from work done in the past. The research conducted to discern the appropriate type of processor suitable to the sequencer project led to a complete review of all available FPGA technology on the market. One of the most important differences between the development of the sequencer and work done previously on CFTP concerns the type of device utilized. While CFTP focused on SRAM-based FPGAs, with all of the inherent requirements for supporting secondary componentry, the fault-tolerant FPGA-based sequencer project utilizes Actel ProASIC3 flash-type FPGAs for embedded processing. This has several benefits including a large reduction in the number of secondary components required to operate the device, a low-power operating mode that reduces power consumption by up to 40%, live-at-power-on processing capability, and the inherent radiation tolerant properties of flash-type technology [79]. This type of FPGA has not been previously utilized in space-systems projects at NPS, and future research should incorporate this design. While the Xilinx Virtex-5 series has certain clock frequency and quantity of logic resources advantages, the Actel ProASIC3 series is suitable in all but the most demanding DSP applications. The ability to have one flash-based FPGA take the place of up to a dozen components on a SRAM-based platform is of great benefit in the radiation environment of space. When only one device is involved, protection of the design from radiation induced faults becomes a more manageable challenge.

In researching the various means of fault tolerance available, one is faced with the task of choosing between several competing schemes—each of which promises to accomplish the same end-state of reducing radiation-induced errors in logic. While there are certain benefits to the quadded logic approach, such as the elimination of the voter circuitry, the combined effects of greatly increased resource utilization coupled with a significant timing delay reduce the suitability of this approach. Additionally, the time-

intensive manual wiring of logic that accompanies the quadded logic approach is disproportionate to the returns in error masking. Similarly, TIR offers many of the same benefits of TMR in a slightly more resource efficient design; however, the lack of certainty in error correction coupled with delay path differences in the circuit negatively affects its usefulness. TMR offers the best combination of logic utilization and timing path delay for any of the circuits investigated.

The development of the PA3TB was instrumental in developing skills necessary for effective use of Altium Designer. Altium was found to be a highly capable platform for development of schematic and PCB layout. The ability to easily try out circuit implementations while linking 2-D footprint and 3-D models to their associated components is of great interest to any circuit designer. Many errors were avoided by careful planning of the schematic before layout and placement fitting of components. There were excellent results obtained in the development of the first PA3TB design; it served as a familiarization with not only the design software toolset but also the manufacturing and integration of multi-layer PCBs and SMT components. There was a general realization of what features would work or be too difficult to manufacture on the more complex SADv3 design after gaining hands-on experience with the integration of the PA3TB.

Design work and implementation of the SADv3 is seen as the most time-intensive portion of work occurring in this research. The complexities involved when moving from a four to eight-layer PCB design with a 484-pin BGA FPGA are easily the most daunting task accomplished. Overall design complexities involved with supporting the FPGA require pre-planning in the number and type of supporting circuitry such as capacitor and resistor banks. The automatic trace routing choices made by Altium Designer are not always in the best interest of the PCB layout. The traces and escape patterns provided by the auto-router may be permissible once extensive work is done to manage the settings and parameters of the layout options. The addition of the SRAM and flash-based memory components to the design is done in the best interest of future operations and mission sets. Although not necessary for the sequencer application, any additional data logging or data processing application will find these devices useful.

In both combinational and sequential circuits, the logic outputs of the TMR instantiated circuit under test were identical to the non-triplicated version. Additionally, the timing path differences between the two versions of circuitry were negligible. In all but the most complex circuits, this should remain the case. The ability to quickly represent HDL code within a schematic or Verilog representation, triplicate the instantiation, and combine with voting logic is a simple means of providing logic redundancy to any design. Sequential circuits take a bit more work, as the feedback paths providing next-state logic must also be properly protected to prevent error propagation. In all cases of TMR testing upon sequential logic, the designed circuit was able to correct any introduced fault prior to affecting the output. This is a clear benefit for the sequencer application, the greatest advantage of which is guaranteed timing and CubeSat deployments during the periods specified.

The ability to manually inject faults for testing, while applicable to the work in this thesis, is a time-intensive proposition. To test all possible fault points within a circuit, a more automated version of the fault injection technique must be developed. Additionally, the manner of TMR development and insertion has great implications concerning time available to pursue other logic research, development, and testing. With the availability of commercial software products, a great deal of time and effort can be saved by automatically configuring different forms of TMR for logic implementation using software packages such as Precision Hi-Rel. Use of this commercial synthesis software will allow for comparisons between the manual TMR methods developed and automated TMR generation. This program should be licensed to allow for continued testing of the various subsets of TMR implementations.

Any transitions from Xilinx to Actel Verilog HDL code require reformatting all instantiated logic components into the proper syntax for use by the Actel synthesis tools. This process is lengthy and tedious with regards to the appropriate syntax for transition from one software toolset to the other. Since Precision Hi-Rel can be targeted toward any vendor’s devices generically, it should also aid in automating syntax exchange to some degree. Even if the logic has been designed schematically in Xilinx, the Precision Hi-Rel package will be able to TMR the design and adapt it for Actel FPGA usage.

The development of HDL sequencer logic is an important step toward the realization of the integrated payload design. Through careful application of various techniques, including Verilog variable declarations and ProASIC3 FlashROM utilization, there are a variety of options available for logic implementation on the sequencer. The ability to quickly realize an FSM design using Verilog case and state assignments saves a great deal of time and effort. This is in contrast to attempt to reproduce a complex FSM design schematically with logic gate equivalency.

The design work in developing a general-purpose payload processor has many real world applications. Specifically, the work contained in this thesis sequencer design is applicable to a real-world, CubeSat deployment application for potential use on future missions. In developing a novel approach to processing and timing management operations, a low-cost solution to a difficult problem is realized. There is still research to be conducted and testing necessary to validate the design prior to manufacture and launch. This additional research should allow for several subsequent theses to be prepared on the topic. The general-purpose payload processor is flexible enough both in configurability and logic capacity to target new implementations. The TMR techniques and logic implementations for the sequencer are only a single example of an application.

B. FOLLOW-ON RESEARCH AND TESTING

There are several avenues available for development of the general payload processor design. One of the first tasks to accomplish is final component placement and subsequent trace verification of the SADv3 by a design review committee. This team of engineers must work to ensure that the design of the eight-layer PCB and all associated components are verified prior to board manufacture. As there are a large number of signal lines on the design schematic, care must be taken to verify all traces to ensure proper endpoint termination and routing. The choice of power supply may also be combined with current means of power conversion to obtain the required +5 V_{DC} voltage for down-conversion to other voltages. VPT Incorporated manufactures DC-DC converters for space applications which are rated to 30 krad TID and SEE characterized to 44 MeV-cm^2/mg with up to 85 MeV-cm^2/mg parts available [80]. They also manufacture an

associated electromagnetic interference (EMI) filter for use with the DC-DC converters [81]. These will work well in converting the +28 V_{DC} power supplied by the host spacecraft into a more manageable +5 V_{DC} for further regulation on the PCB. Their use has already been space-qualified in several previous mission hardware designs.

Another desired feature for inclusion on the SADv3 design is the incorporation of Ethernet for data logging during radiation testing. Preliminary research into the incorporation of Ethernet standards, in combination with ProASIC3 FPGAs, indicates that there is a Microsemi available 10/100 Ethernet Media Access Control (MAC) core to connect to local area networks (LANs) [82]. Work will need to be done to interface the appropriate pins of the FPGA and determine logic area utilization and suitability for triplication. Actel provided 10/100 MAC core specifications for the ProASIC3 series that specify an estimated 44% device utilization with a corresponding 49 MHz clock performance [82]. This utilization factor precludes TMR instantiation of this portion of the FPGA logic on the ProASIC3 series of devices.

The addition of SRAM and flash-based memory to the SADv3 design opens up new possibilities in both processing and video feed features. Implementation of the FPGA into a general payload processor arrangement for other applications can utilize such features. There are a multitude of cores available for processing available directly from the vendor [83]. While much of this IP may cost extra to license, there are open-source cores available which may serve an equally useful purpose [84]. The available open-source cores include communications controllers, processors, systems controllers, and video controllers. Research into these IP options should be conducted to find suitable items for possible inclusion into the SADv3 design. There are display reference designs available including video and still-shot camera design and programming files [85]. It should be noted that the large relative device utilization of such proprietary cores may make it impossible to TMR such features on the FPGA while leaving sufficient logic resources to TMR the sequencer portion of the design. Additional difficulties are incurred in trying to TMR proprietary IP with regard to timing constraints and power requirements. Some testing of this may be possible with the Precision Hi-Rel software

packages, although, there is still a logic resource utilization challenge as seen in the 10/100 MAC core specified previously.

The majority of remaining work involves radiation testing of a flight-ready SADv3 board with various fault-tolerant schemes. This is necessary to determine the best combination of FPGA hardware and configuration logic. The various forms of radiation introduction to the PCB need to be investigated, including FPGA dose rates and the possibility of whole board testing. Radiation levels need to be sufficient to push the FPGA to specified limits and beyond to determine suitability for the space environment. For the nuclear source chosen for testing, coordination between NPS and the cyclotron provider needs to be managed up-front to avoid testing errors in setup.

In conclusion, there are a number of applications for which the ProASIC3 general purpose payload processor is suitable. The sequencer work contained in this thesis is one application in which a significant gain in the state of current CubeSat deployment technology may be realized. The ability to deploy CubeSats using TMR logic should be quite cost effective when compared to custom, rad-hard components, possibly with no loss of fault tolerance. TMR is useful for protecting both combinational and sequential circuits, especially given the available fault-tolerant software toolsets. Future designs using Actel flash-based FPGA technology may have wide applicability due to its configurability and the flexibility with which the I/O pairs can be assigned. The compactness of the ProASIC3 circuit design, when compared to the more traditional Xilinx Virtex series FPGAs, is readily apparent. It is anticipated that this design, or some evolution of it, will be quickly developed into a flight-ready product.

APPENDIX A. FPGA HARDWARE PRODUCTS AND SOFTWARE DEVELOPMENT TOOL DIFFERENCES

A.1 COMPARISON OF XILINX AND ACTEL PRODUCTS

The goal of this appendix is to look at all of the considered hardware vendors to determine the optimal solution to the choice of FPGA for use in the CubeSat deployment sequencer. A variety of features are compared, including software packages and hardware feature sets. In past chapters, the choice of the FSM type, FPGA types considered, radiation challenges faced, and fault-tolerant logic implementations were discussed. The choice of a SRAM-based or flash-based FPGA using a Moore machine FSM was settled upon as the desired configuration for the sequencer. This appendix looks to further that choice to a specific product, having considered the software suites available and the particular hardware configurations which may fit a chosen fault-tolerant scheme.

In both cases, Xilinx or Actel products, design choices are limited to those devices with a full upgrade path to rad-hard technology. The devices compared are those that would have pin-to-pin compatibility between commercial and military-grade products with their rad-hard counterparts. Both vendors' software products are compared in terms of feature sets, usability, and ease of hardware interface. In all cases, attempts are made to test various logic designs upon the hardware compared using vendor provided development boards. These experimentation results were discussed previously in Chapter VII.

1. Software Feature Set

For the purposes of this research, two of the most prominent vendors of SRAM-based and flash-based FPGAs were compared in hardware and software features. Xilinx Corporation is known for its Spartan[11] and Virtex series of devices - with legacy Virtex I FPGAs being flown on CFTP during MidSTAR-1 and the upcoming NPSAT1 satellite missions. Xilinx devices are SRAM-based with all of the previously mentioned design challenges. Xilinx devices are best programmed with the Xilinx Integrated Software

[11] Spartan® is a registered trademark of Xilinx, Inc.

Environment (ISE[12]) development package [86]. Meanwhile, Actel (now a division of Microsemi) Corporation has long focused on antifuse technology but has recently made strides in configuring their ProASIC[13]3/L flash-based FPGAs for the space environment. Similar to ISE, Microsemi provides a software environment known as Libero[14] System on Chip (SoC) targeted to its various FPGA product lines [87].

a. Xilinx ISE 14.3

ISE WebPACK[15] design software is the name of the free Xilinx PLD development suite made for Linux, Windows[16] XP, and Windows 7. ISE WebPACK is a downloadable solution for FPGA and CPLD design offering HDL synthesis and simulation, implementation, device fitting, and Joint Test Action Group (JTAG) programming. It consists of the software development kit (SDK), MicroBlaze[17] controller system, project navigator, CORE Generator[14], PlanAhead[14], ChipScope[14] Pro, Xilinx Power Analyzer (XPA), ISE Simulator (ISim), and Xilinx Synthesis Tool (XST). The software package is upgradable to more extended editions including the Logic, Embedded, and DSP versions of the package [86]. The use of Xilinx ISE has a long history at NPS, an earlier version being the toolset utilized for the development of CFTP. Xilinx ISE, having used Mentor's ModelSim[18] in the past, now defaults to its proprietary ISim simulator. The primary interface of Xilinx ISE can be seen in Figure 59.

---

[12] ISE® is a registered trademark of Xilinx, Inc.

[13] ProASIC® is a registered trademark of the Microsemi Corporation.

[14] Libero® is a registered trademark of the Microsemi Corporation.

[15] WebPACK™ is a trademark of Xilinx, Inc.

[16] Windows® is a registered trademark of Microsoft Corporation.

[17] MicroBlaze™, CORE Generator™, PlanAhead™, and ChipScope™ are trademarks of Xilinx, Inc.

[18] ModelSim® is a registered trademark of the Mentor Graphics Corporation.

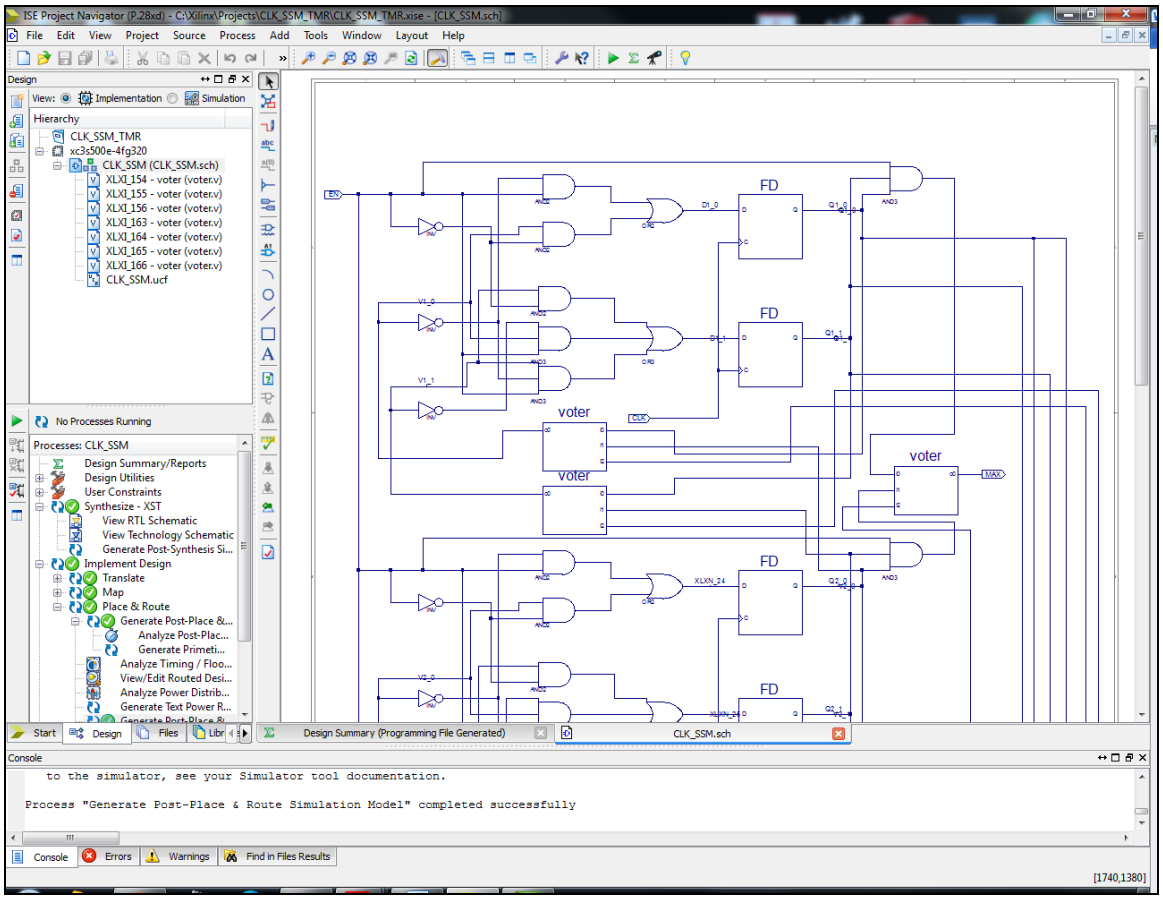


Figure 59. Interface of Xilinx ISE 14.3 showing schematic layout and design flow.

For the initial tasks of developing simple combinational and sequential circuits, an important difference between Xilinx ISE and Libero SoC was readily apparent. The Xilinx ISE package was found to have a much more comprehensive and accessible schematic editor, making construction of logic circuits using baseline gate instantiations a great deal easier. The Libero SoC package was found to be greatly limited in this aspect, with an overall system level schematic design being available yet difficult to use. One must suspect that the limited adoption of Actel FPGA products (in comparison to the widely utilized Xilinx series) may be due in some part to the limitations of this feature.

Additionally, the overall design flow of the Xilinx ISE product seemed to be more flexible and easy to grasp than that of the Actel SoC product. Xilinx ISE is useful in that it generates an overall 'Design Summary' at every step of the process and is

the default view upon starting a project or continuing work, as illustrated in Figure 60. This lets one view, in a stoplight and metric fashion, the remaining tasks and the overall logic utilization throughout every step of the process. The output of each process can be exported in a text logbook fashion for future reference and design history.

RAM Project Status (11/14/2012 - 09:46:30)			
Project File:	VHDL_RAM.xise	**Parser Errors:**	No Errors
Module Name:	RAM	**Implementation State:**	Programming File Generated
Target Device:	xc3s500e-4fg320	**•Errors:**	No Errors
Product Version:	ISE 14.2	**•Warnings:**	929 Warnings (672 new)
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	All Constraints Met
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Latches	2,048	9,312	21%	
Number of 4 input LUTs	1,351	9,312	14%	
Number of occupied Slices	1,715	4,656	36%	
Number of Slices containing only related logic	1,715	1,715	100%	
Number of Slices containing unrelated logic	0	1,715	0%	
Total Number of 4 input LUTs	1,351	9,312	14%	
Number of bonded IOBs	25	232	10%	
Average Fanout of Non-Clock Nets	3.35			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Figure 60. ISE 'Design Summary' window listing FPGA utilization and design flow.

The options within the toolset allow for easy customization of the programming options for FPGA synthesis. Logic designs can be tailored for speed, area optimization (size), or further customized to fit individual needs. The ability to rapidly reconfigure a target design to other FPGA target devices is as simple as changing a configuration option and was used extensively in the testing in Chapter VII. This allowed for selection of Spartan versus Virtex products in the development of stacked methods of fault tolerance to test sizing constraints.

Furthermore, the transition from schematic diagrams to generate HDL was seamless, as Xilinx ISE allows for HDL generation to Verilog or VHDL. In all cases, the

generated code was easy to follow due to the naming scheme and instantiations matching the schematic labeling. Actual synthesis of a design was very informative in the case of errors generated by XST. XST provided enough context help to locate the line of code in error or the schematic discrepancy. HDL files and schematic files can be checked for concurrency with standards prior to implementation.

b. Actel Libero SoC 10.1

Libero SoC is a comprehensive design software toolset for use with Microsemi's SmartFusion[19], IGLOO[16], ProASIC3 and Fusion[16] FPGA families. This software suite allows for entire design flow from design entry, synthesis and simulation, through place-and-route, timing and power analysis, with enhanced integration of the embedded design flow [87]. Libero SoC consists of the product design flow and management system including a 'Core Catalog' with commonly configurable cores and the SmartDesign[20] creation tool for building high level modular systems and subsystems. Additional software tools provided include Synplify Pro[21] AE, Synphony Model[22] Compiler AE, ModelSim AE, physical design layout, SmartTime[23], and SmartPower[20]. The primary interface of Libero SoC can be seen in Figure 61.

In contrast with Xilinx ISE, the Actel Libero SoC package seems more tailored to system level design, especially once the baseline HDL is written for the FPGA logic. While product literature indicates that SmartDesign is tailored to assemble and connection Microsemi intellectual property (IP), user-generated IP, and custom HDL modules [88], the development of initial HDL models (especially schematically) is troublesome within Libero SoC. Since Xilinx ISE is able to create schematic symbols for synthesized Verilog and VHDL, the product is more easily suited to wiring those instantiated symbols graphically.

---

[19] SmartFusion®, IGLOO®, and Fusion® are registered trademarks of the Microsemi Corporation.

[20] SmartDesign™ is a trademark of Synopsys, Inc.

[21] Synplify Pro® is a registered trademark of Synopsys, Inc.

[22] Synphony Model™ is a trademark of Synopsys, Inc.

[23] SmartTime™ and SmartPower™ are trademarks of the Microsemi Corporation.

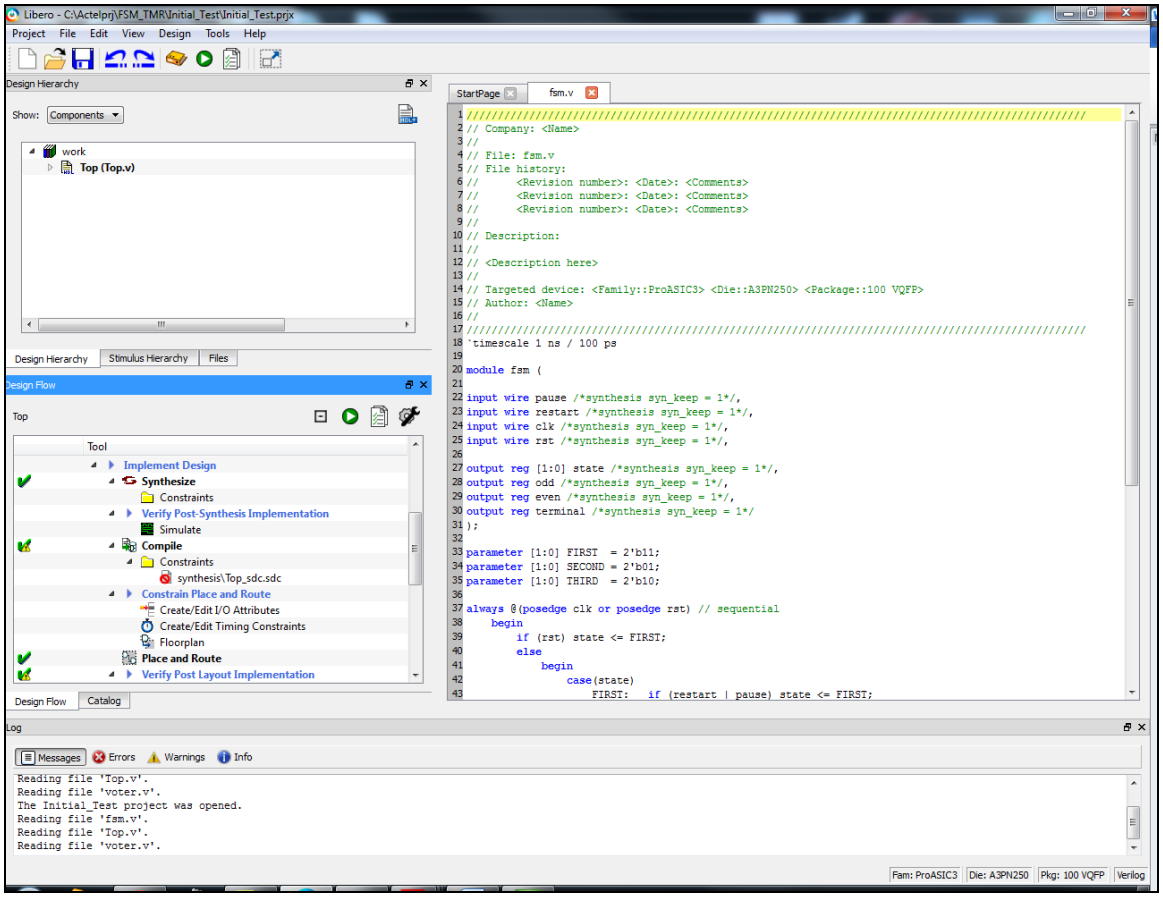


Figure 61. Interface of Libero SoC 10.1 showing Verilog HDL and design flow.

Both Xilinx and Actel have strong chip planning toolsets when looking at device pin-outs and logic floor plans. Much of the chosen product depends upon comfort with simulation software available for the two suites. Actel SoC defaults to Synplify Pro as its simulator of choice. Any reasonable experience with either of these two simulators could lead to a tendency to choose one software package over the other. The author has found, in extensive testing, that the feature sets of both ISim and Synplify are quite similar with logic timing diagrams generated being almost identical. Users of either simulation product should not have any difficulty switching from one to the other. Overall choice of software package comes down to comfort level of HDL versus schematic design and choice of FPGA. Both are fully capable in their synthesis, simulation results, and final output of bitstream files for FPGA configuration.

c. Results of Software Comparison

Overall, Xilinx ISE appears to be a more robust and feature-filled software suite. The ease of transition from HDL instantiation, to simulation, to synthesis and implementation is seamless and requires only a few hours of experimentation to grasp the basics. In contrast, Libero SoC seems very difficult to grasp, especially considering there are logic flow difference between versions 9.1 and 10.1 of the package. It appears as if Actel is trying to utilize a more contextual menu interface for design flow (similar to Xilinx) in its newer product, however, the graphical flow diagrams of the 9.1 software suite made a great deal more sense. On the basis of software alone, Xilinx ISE is a clear winner in feature set, configuration options, and the usability of the software package.

2. Hardware Feature Set

Significant differences in the Xilinx and Actel series of FPGAs are apparent in the choice of hardware for design. The SRAM-based implementations of Xilinx compared to the flash-based FPGAs of Actel are significantly different in their utilized technology. While the initial research in Chapter II led to the conclusion that flash-based FPGAs are superior in their simplicity, there is much left to be considered including their suitability in space-applications. For the sequencer developed for CubeSat deployment applications, one desires a re-programmable, low-power device of the most simplistic nature available. SRAM-based FPGAs, due to their requirement for external PROM type devices for configuration and microcontrollers to enable the power-on configuration, are significantly more challenging in the secondary component requirements.

Experience with Xilinx FPGAs shows that at least 90% of SEEs cause configuration faults rather than logic faults. Configuration faults can cause persistent errors in computation. The error-correcting technique used in the logic must detect as well as correct any errors in the computation. Detection of a repeated error in the computation then should signal the need for reconfiguration. Until reconfiguration of that portion containing the configuration error, the error-correction capability maintains correctness of the result [89].

A comparison between the radiation tolerant properties of both devices is necessary for down-selection to a specific product line. While many of the overall features, include logic real-estate, configuration methodology, and use in the final circuit are similar, clearly the differences in radiation tolerant properties and secondary component requirements have the most impact upon selection. The two product lines considered for use in the research are the Xilinx Virtex and Actel ProASIC3 series of FPGA devices. Experiments conducted on logic development in Chapter VII were targeted to both vendors' products, allowing for the earlier software feature comparison and subsequent testing of development boards.

a. Xilinx Virtex Series FPGAs

The Xilinx SRAM-based Virtex series of FPGAs are consistently utilized devices with long heritage in space applications. Their product familiarity, coupled with many years of on-orbit lifetime utilization, has led to a product line extending almost a decade. The space-rated rad-hard versions of the Virtex-I/II gave rise to the Virtex-4QV that has, in turn, led to the development and fielding of the radiation hardened Virtex-5QV. One must suspect that their more recent product lines, including the Virtex-7 series FPGAs, are undergoing modification and certification for use in the space environment.

For the rad-hard Virtex-5QV device, the FPGA consists of 131,072 logic cells, 87,920 internal fabric FFs, 87,920 real 6-input LUTs utilizing greater than five million LUT bits, 450 MHz system performance, up to 18 total clock generators, 836 user I/Os, and 20,480 configuration logic blocks (CLBs). Each CLB slice contains four LUTs and four FFs (an increase from earlier generation devices). There are also 320 DSP slices consisting of a 25×18-bit multiplier, an adder, and an accumulator [61]. The internal structure of a Virtex-5 CLB slice is illustrated in Figure 62.

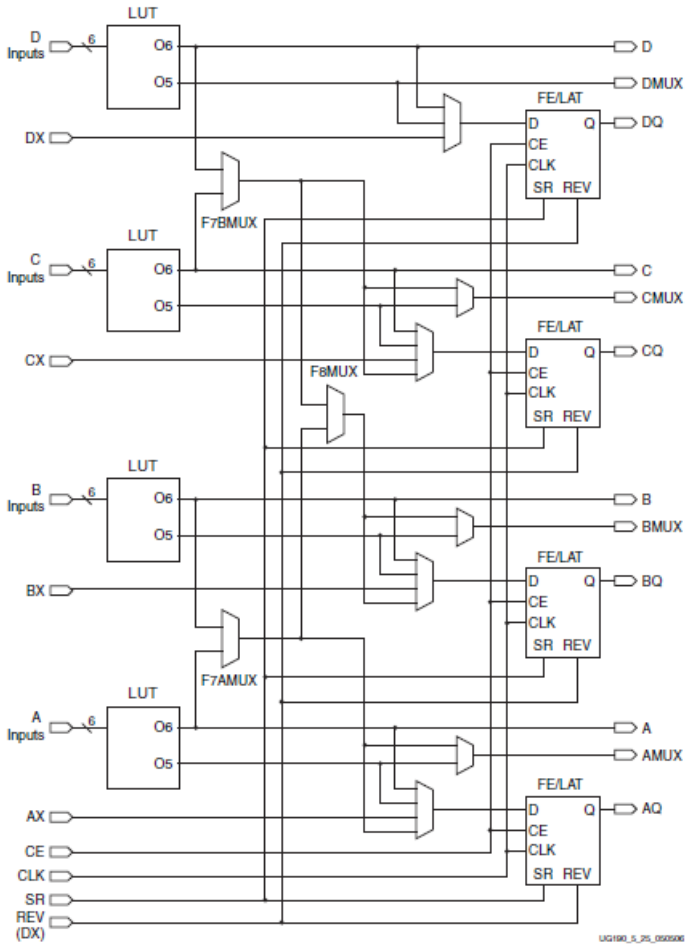


Figure 62. Each Virtex-5 CLB slice contains four LUTs and four FFs (From [79]).

The Virtex-5QV series advertises guaranteed SEL immunity to LET greater than 100 MeV-cm^2/mg, 1 Mrad TID, SEU hardened configuration memory cells and control logic with 3.8×10^{-10} errors/bit/day, and SEU and SET hardened CLB flip flops and internal data paths [61]. The SRAM cells used for the configuration memory of the Xilinx FPGAs are larger and more robust than the SRAM cells used for general-purpose memory, which are optimized for speed and cost [25]. These configuration memory cells are optimized for SEU resistance by a combination of their feature size and manufacturing technique.

The Virtex-5QV is listed as compatible with the commercial and defense-grade Virtex 5 FPGAs, allowing for low-cost, rapid prototyping and easy design migration to flight hardware. For the purposes of a component comparison, the Virtex-

5QV (XQR5VFX130) rad-hard version is compatible with the commercial-grade Virtex-5 (XC5VFX130T) [90]. The embedded microprocessor of the Virtex-5QV is disabled when compared to the commercial versions. Also, there are some pin differences, as the rad-hard version contains 1,752 pins, while the commercial version utilizes 1,738 pins [61], [90]. There are compatibility guides provided by Xilinx for the correct pin conversions from one design type to the other. The lowest cost for a commercial-grade Virtex-5 is $4,352.40 [91].

b. Actel ProASIC3 Series FPGAs

A more recent development in FPGA manufacturing stems from the use of flash-based FPGA technology to provide operational capability at power initiation and reprogrammable aspects, as previously discussed. The use of flash technology is a great enabler for the reduction in secondary components required for operation. Additionally, one may benefit from the inherent radiation tolerant properties of flash-memory when compared to SRAM-based FPGA configurations. In this sequencer research, the RTAX[24] and RTSX antifuse products from Actel were discounted for lack of reconfiguration techniques previously discussed. Therefore, only RT ProASIC3 devices, with pin-compatibility to the ProASIC3L defense-grade of FPGAs, were considered for use in the project. With lack of a reconfiguration requirement, however, RTSX-SU devices, which implement TMR in hardware, could be ideal [92].

In the RT ProASIC3 device, the FPGA consists of 600,000 logic gates including: 13,824 VersaTiles[25] (D-FFs), 350 MHz system performance at 1.5 V_{DC} or 250 MHz at 1.2 V_{DC}, six clock-conditioning circuits (CCCs) with integrated phase-locked loops (PLLs), 270 user I/Os, and 135 differential I/O pairs. Each VersaTile can be configured as a three-input logic function, a D-flip-flop (with or without enable), or a latch by programming the appropriate flash-switch interconnections. There is also 108 kbit of RAM and 1 kbit of non-volatile ROM provided [19]. The internal structure of a ProASIC3 VersaTile is illustrated in Figure 63.

---

[24] RTAX™ is a trademark of the Microsemi Corporation.

[25] VersaTile™ is a trademark of the Microsemi Corporation.

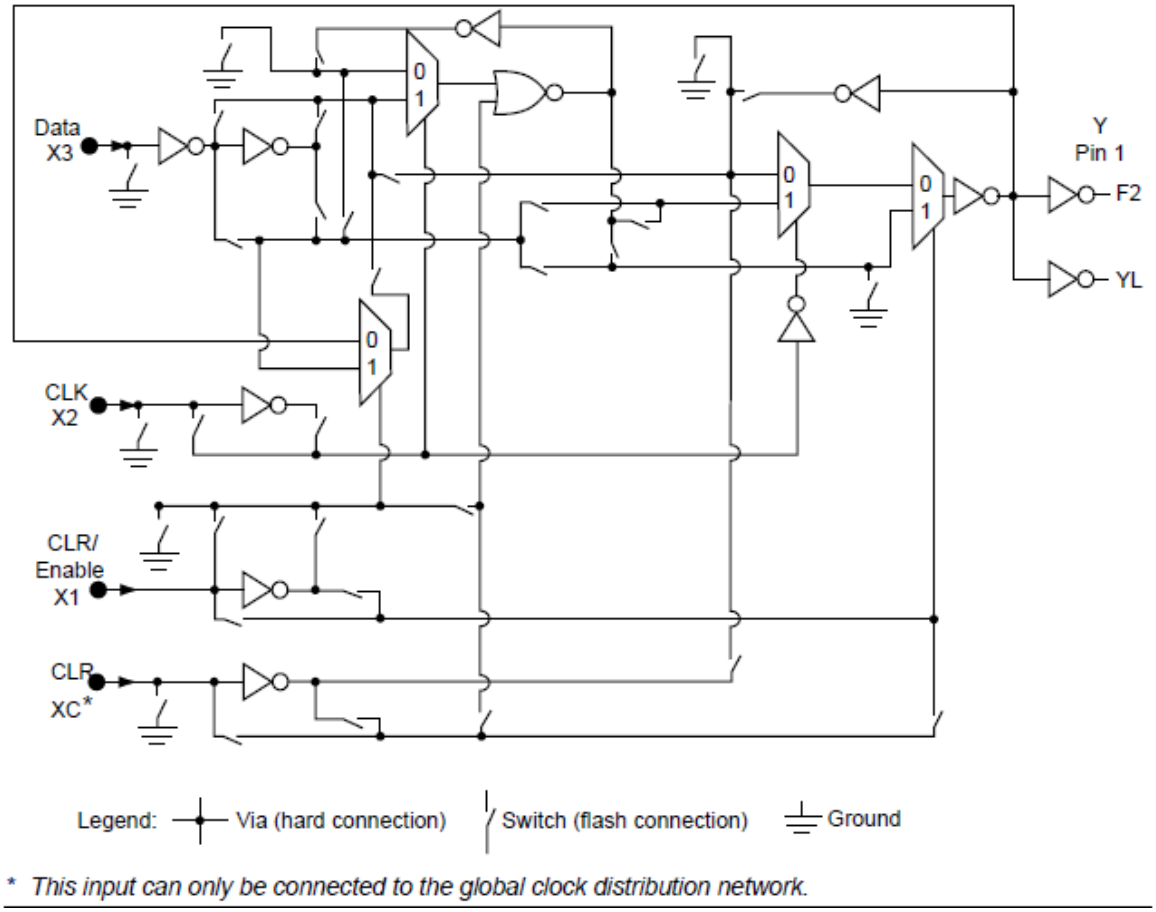


Figure 63. Configuration of low-power ProASIC3 device core VersaTile (From [93]).

The RT ProASIC3 series advertises SEL immunity to a projected LET threshold of 68 MeV-cm^2/mg, D-FFs and internal SRAM SEU limits of 63.5 MeV protons, and heavy ion transients on the global clock networks and I/O banks greater than 70 MeV-cm^2/mg [43]. SEE mitigation strategies mentioned include TMR of the clock network, I/O banks and D-FFs. This may not be the most efficient manner to mitigate SEEs in the combinatorial logic and embedded SRAM memory; strategies for these logic types are explained in more detail in Microsemi's radiation reports [29].

RT ProASIC3 devices have been tested for TID effects under x-ray and gamma ray environments [94]. There is an observable effect of increase in propagation delay through pass transistors as TID increases. This is due to floating gates being less able to hold interconnection pass transistors in their 'on' state. With a dose rate of 5

krad/min, an increase in propagation delay of 10% has been observed at a gamma ray TID of 25 to 30 krad. The results of this analysis are indicated in Figure 64. At 1 rad/min, which is more representative of the space environment of LEO, the 10% propagation delay occurs at a TID level of approximately 40 krad. A 15% propagation delay was seen at a TID level up to 55 krad [83]. Microsemi is working to alter Libero SoC to include propagation delay increase for designers wishing to incorporate these delays into simulation and static timing analysis prior to deployment [83].

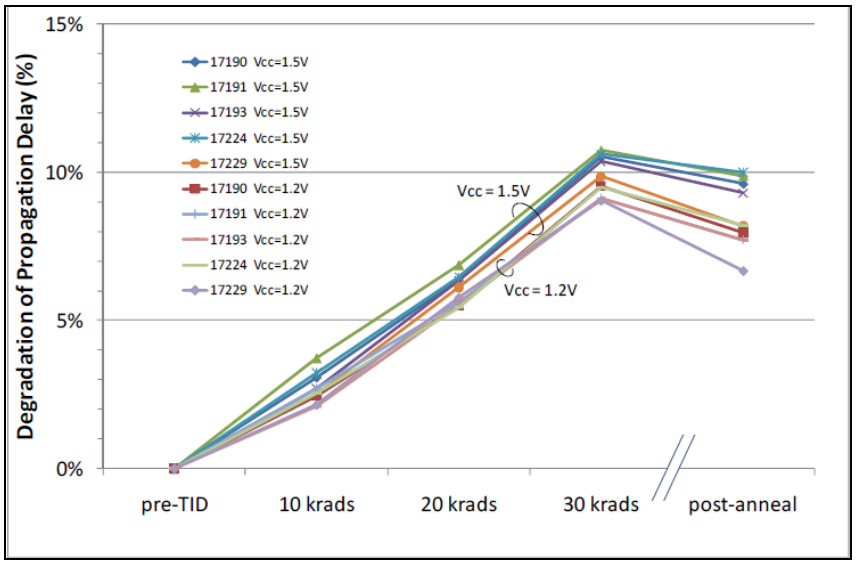


Figure 64. Degradation of propagation delay versus TID for RT ProASIC3 (From [83]).

The RT ProASIC3 is listed as pin-to-pin and timing compatible with the defense-grade ProASIC3EL FPGAs. For the purposes of a component comparison, the RT ProASIC3 (RT3PE600L-CG484B) rad-hard version is compatible with the defense-grade ProASIC3EL (A3PE600L-FG484M). The use of the M version (military temperature range) ensures that timing can be verified in hardware across the full temperature range from -55 C to 125 C [83]. The lack of pin differences between the rad-hard and military-grade components is of great benefit in comparison to the Xilinx Virtex-5QV. This should allow for one standard design in which schematics can be developed and not modified if utilizing either the rad-hard or military-grade FPGA. Since the overall goal of this thesis is to select lower-cost componentry, with the possibility for

future upgrade dependent on mission, the Actel series devices are more design friendly in this regard. The individual cost for both these devices require product quotes; a logic compatible device in the 484 pin-package can be purchased for as little as $63.87—this device (A3PE600-FG484) should be sufficient for initial prototype testing [95]. Depending on the results of future radiation testing, this prototype FPGA may be reliable enough with redundant logic to utilize on final flight hardware.

3. Associated Hardware Requirements

There are many associated benefits with the ProASIC3 and ProASIC3L FPGAs related to supporting hardware. These devices can be powered from a single 1.2 V_{DC} or 1.5 V_{DC} supply for core voltage and I/Os. There is no requirement for a 2.5 V_{DC} or 3.3 V_{DC} supply for power-up; although, these are added as necessary to supply I/O bank voltages. Running the entire system on a single supply saves costs and area associated with voltage regulators [90].

Since there is no boot PROM or flash microcontroller required to load the FPGA at system power-up, the ProASIC3 line is a single-chip solution [90]. Since CPLDs are not required, ProASIC3 FPGAs are live at power-up. Again, this saves on board layout area and total system cost. Additionally, no power-up monitor chip is required; unlike SRAM configuration bits, flash configuration switches do not brownout [90]. Clock management is also simplified by live at power-up CCCs and PLLs, allowing for the removal of additional clock distribution chips often used to boot SRAM FPGAs or microcontrollers during system start-up [90].

A.2 APPENDIX SUMMARY

The comparison of Xilinx versus Actel software and hardware products was discussed in Appendix A. Xilinx ISE 14.3 was compared to Actel SoC 10.1 with ISE being realized as the more complete design suite due to its' integrated schematic editor. Both software suites were seen to have capable synthesis and simulation toolsets.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. PROASIC3 TEST BOARD – REFERENCE, SCHEMATICS, AND PCB LAYOUT DIAGRAMS

The design files for the ProASIC3 Test Board are divided into four sections: the bill of materials, schematics for electrical connections, the 2-D PCB layout documents, and 3-D component placement views. All of the associated documents were constructed using Altium Designer 10 during the development portion of this research.

B.1 PROASIC3 TEST BOARD DEVELOPMENT METHODOLOGY

1. Detailed Discussion of ProASIC3 Test Board Design Decisions

Since one of the objectives of this test board was to minimize the amount of secondary components placed upon the PCB, the power regulation circuitry for the PA3TB was not implemented on the PCB. A wiring harness configuration was determined to be the proper method for providing power to the board. A bench-top Agilent E3632A DC power supply was utilized for the power supplied to the FPGA core with an Agilent E3631A triple output DC power supply used to supply secondary voltages. By using external power supply and conditioning, any design errors could be localized to the FPGA design rather than the power circuitry. This greatly aids in trouble shooting, should the PA3TB not operate as anticipated upon integration.

For the power supply harness, a six wire configuration was chosen as it would supply the required voltages and ground for proper operation of the FPGA in various manners. As seen in Figure 65, pin 1 is tied to ground, pin 2 is USB voltage (optional), pin 3 is V_{CCPLF} (tied to 1.5 V_{DC} or GND dependent upon analog PLL usage) [96], pin 4 is V_{CC} (1.2 V_{DC}), and pins 4 and 5 provide I/O bank voltages (1.2, 2.5, or 3.3 V_{DC}). This allows for maximum flexibility in testing of the PCB, as one desires to enable certain features or drive I/O lines at different voltages depending on connections. A Molex 43045-0614 2×3 Micro-Fit 3.0[26] connector was chosen as the six wire header [97].

[26] Micro-Fit 3.0™ is a trademark of Molex.

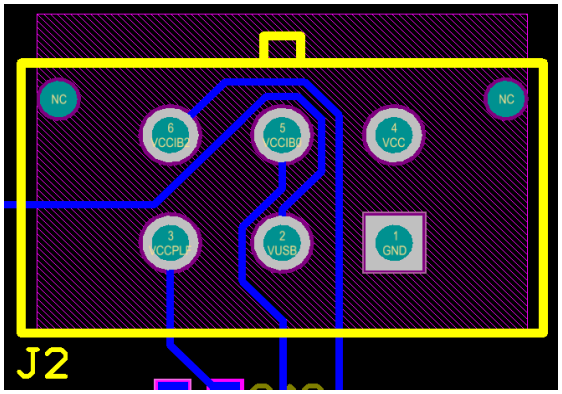


Figure 65. Footprint of Molex Micro-Fit 3.0 6-pin header for power supply to FPGA core and I/O bank voltages.

Similarly, the JTAG header was chosen for ease of interface to the Microsemi hardware programming device, a FlashPro4[27]. The FlashPro4 provides in-system programming for all series of Actel FPGAs including the ProASIC3 (nano and rad-hard variants). The FlashPro4 supports IEEE 1149 and IEEE 1532 standards for programming [98]. This allows for easy programming when coupling the Libero SoC IDE and FlashPro software. The JTAG interface is a 10-pin header in a 5×2 configuration. The footprint of this connector as seen on the PCB layout as illustrated in Figure 66. The straight pin header used on the PCB board for JTAG interface is manufactured by TE Connectivity (P/N 5103308-1) [99]. This part is the replacement for the discontinued components specified in the FlashPro4 user's guide [100].

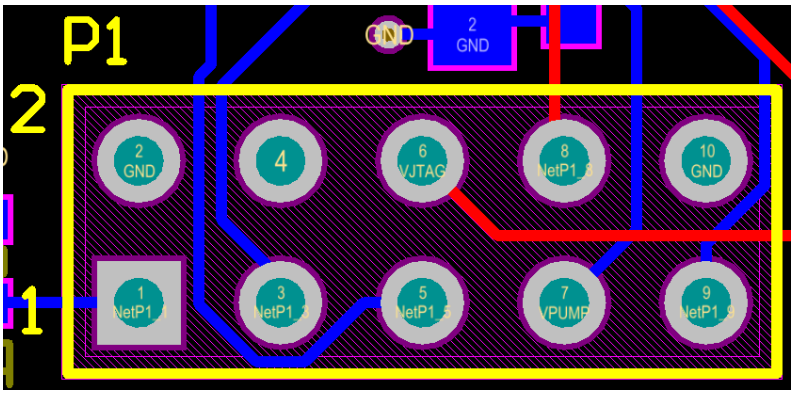


Figure 66. Footprint of FlashPro4 JTAG 10-pin header for programmer interface.

[27] FlashPro is a trademark of the Microsemi Corporation.

To allow for future debugging and interfacing with secondary data logging equipment, a USB connector was added to the PA3TB design. This interface was useful in adding experience in interfacing secondary standards with the FPGA I/O banks. A standard USB 1.1 Series-B receptacle was added. A Tyco Electronics receptacle assembly was chosen as the device for implementation (P/N 292304-1) [101].

The emulation of CubeSat P-POD deployment is simulated by the inclusion of eight 'bit' light-emitting diodes (LEDs) which light to indicate activity on that particular I/O pin of the FPGA. These were labeled BIT0 through BIT7. The specific LEDs chosen were green, surface-mount of standard brightness in the 0603 form-factor (P/N QTLP601C-4) [102]. These were coupled with current-limiting resistors also in the 0603 form-factor. By utilizing these LEDs, we can realize a visual representation of sequencer timing. Four additional LEDs were added on a different bank for testing and indication purposes, labeled PWR and TST0 through TST2.

Since the LEDs and resistors chosen were of 0603 type, all power-decoupling capacitors required for the FPGA power inputs were also selected as 0603 form-factor. This minimizes board area necessary for wiring while still allowing for soldering under the microscope. These small LEDs, chip resistors and capacitors can be manipulated with tweezers while soldering with a fine-tip so long as magnification is available. In addition, the VQ100 package of the ProASIC3 nano is able to be soldered under the microscope.

The remaining requirement of expandability is met by adding as many copper pads to the PCB as will fit. This will be discussed during the PCB layout section of this appendix. The full 'Bill of Materials' for the ProASIC3 Test Board is provided in Appendix A.

2. Discussion of Schematic

For the development of PA3TB, the schematic portion of Altium Designer was utilized to develop the circuit design. The ProASIC3 nano was laid out first, with the banks not utilized in construction ('Bank 1' and 'Bank 3') placed with non-connected I/Os. These are additional connections that can be implemented in future designs with this device. Concerning 'Bank 0', very few of the available I/O resources were utilized;

I/O_00 through I/O_03 were tied to the PWR and TST0 through TST3 indicators. I/O_04 through I/O_19 were seen as points to implement future electrical connections. These I/Os are tied to copper pad areas on the PCB for expandability. 'Bank 2' I/O_38 through I/O_40 were utilized for the USB connection with resistive values allocated as per the datasheets [101]. The BIT0 through BIT7 LEDs were also implemented on this bank via I/O_41 through I/O_48; resistor values of 390 Ω were utilized as per specifications for LED protection. A close-up schematic view of the USB and LED connections are illustrated in Figure 67.

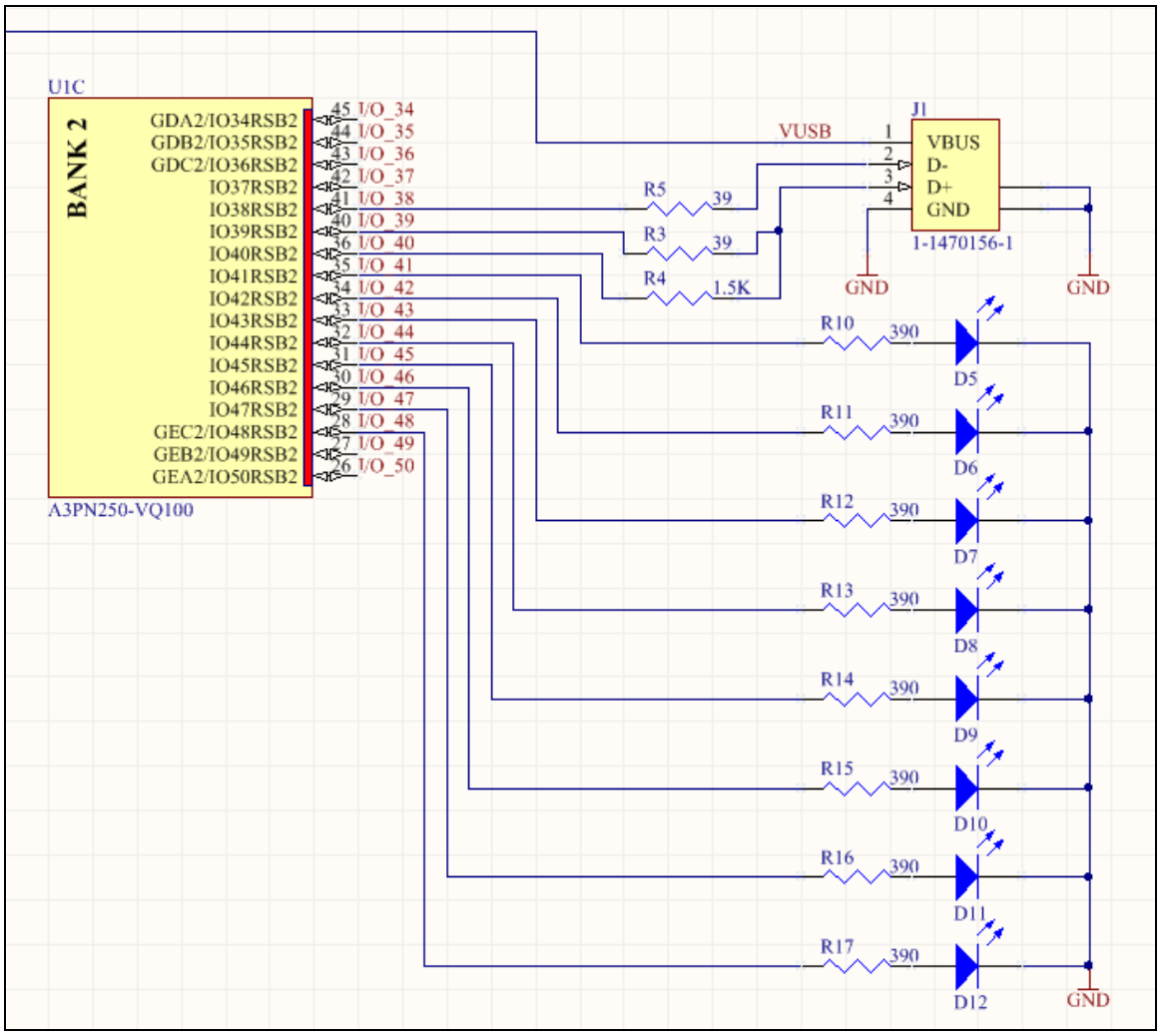


Figure 67. Schematic view of PA3TB 'Bank 2' connections to USB connector and BIT0 through BIT7 LEDs.

All power de-coupling capacitors for the FPGA were tied to the appropriate V_{CC} inputs for proper operation. Again, this was done in accordance with the ProASIC3 datasheet to reduce noise from the power supply to the FPGA [74]. The 6-pin 2×3 Micro-Fit 3.0 connector was placed on the schematic and wired to the appropriate power and ground lines on the FPGA and USB header. In placing the JTAG connector on the schematic, a 5×2 header was utilized for the electrical connections as seen in Figure 68. Proper biasing of the V_{PUMP} connection was done in accordance with the datasheet [74].

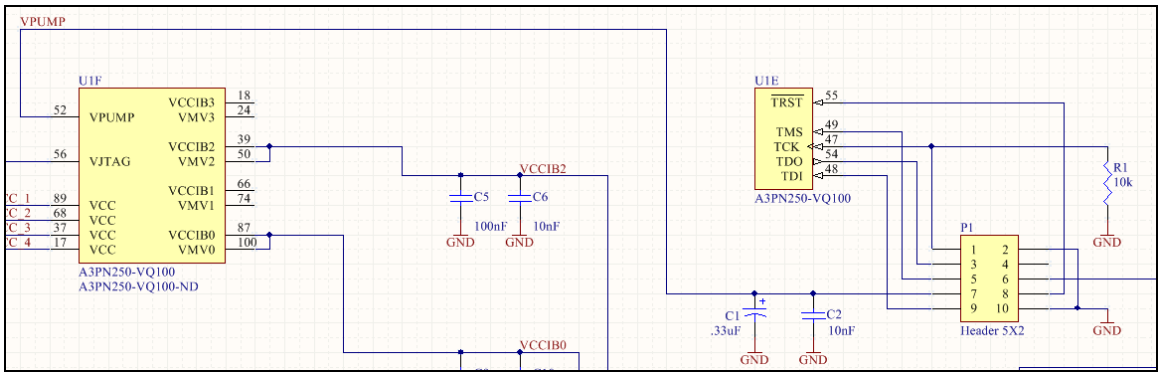


Figure 68. Schematic wiring diagram of JTAG interface header to ProASIC3 nano connection points on PA3TB.

The complete ProASIC3 Test Board schematic is presented in Appendix A.

3. Discussion of PCB Layout and Component Placement

After schematic design of the ProASIC3 Test Board was complete, development moved into the PCB layout and board management phase. The PCB was setup with the previously specified dimensions and layer configuration and all components from the schematic phase brought into the layout view for placement. Initial work concentrated on placing the ProASIC3 nano FPGA toward the center of the board design to obtain area utilization data. The FPGA was seen to require approximately 12% to 15% of the available board area on the top face of the PCB (including fan-out pad fingers). Work was then done to place the de-coupling capacitors as close to the FPGA as possible, in line with the manufacturer recommendations. Thus, the capacitors were placed within the footprint of the FPGA on the bottom face of the PCB. They are staggered in a

configuration which allows them all to fit and utilize vias for connection from top to bottom signal layer planes to their associated V_{CC} and GND connections.

Placement of the USB, power header, and JTAG header occurred next as these were the next largest components on the PCB. The headers and connector were also placed on the top face of the PCB to allow for the bottom face of the board to rest flat upon a surface. When associated with their required resistors and capacitors, the combination of USB, headers, and FPGA utilize approximately 50% of the available PCB area. This is before any wiring was accomplished to tie electrical connections together. The USB and power header were placed near the top of the PCB; since the power harness extends quite a distance from the PCB to the associated bench power supply, this reduces wiring clutter with the JTAG interface. The JTAG interface was placed at the bottom of the PCB, allowing for electrical connection to the FlashPro4 programmer via ribbon cable.

Layout of the PWR, TST0 – TST2, and BIT0 – BIT7 LEDs with their associated resistors was then undertaken. These components were seen to fill another 25% of the PCB top layer area. All of the BIT indicators were placed on the left side of the PCB so that any prototyped sequencer timings could be quickly visualized on LED indication. The power indicator and TST LEDs were placed at the top-right of the PCB to keep them separated from the BIT indicators. This also has the benefit of tying these lines to their 'Bank 0' electrical connections on the FPGA without any complicated routing requirements.

The sixteen copper pad areas were placed in the middle and lower-right section of the PCB. These are used in the connection of the PCB to any external equipment desired; the first iteration of the sequencer relay board can be tested with these I/Os for prototyping of the sequencer logic. This allows for a quick breadboard type setup to be accomplished while work is completed on the SADv3 design.

All of the remaining board area was used for the connection of signal lines between the FPGA, secondary components, and required resistors and capacitors. The large diameter vias for power and ground supply were tied to the internal V_{CC} and GND

planes, which connected the internal power planes to their associated nets. With this method, extraneous wiring from component ground points to localized vias is minimized.

Most of the top face traces are associated with the LED indicators or copper pad fingers. The bottom face traces are associated with the various capacitor and resistor banks. Wiring was accomplished with 8.0 mil traces. Care was taken to avoid any right angle routed traces. All PCB footprints and component identifications were placed on top and bottom silkscreen layers, with board identification information filling the remaining space.

4. Physical Construction and Testing

After completion of PCB layout, the design was exported in mechanical and electrical Gerber Computer-Aided Manufacturing (CAM) files for fabrication. These files were verified by automated design checking tools by Advanced Circuits [103]. This allowed for any mechanical or electrical deficiencies to be detected prior to manufacturing; several issues were reported as potential errors and these were corrected with corrected CAM files re-exported. Since the initial PA3TB run consisted of only three boards, these were sent to a local PCB manufacturer for production [104]. This was a lower-cost option for small production runs of non-flight, non-ITAR (International Traffic in Arms Regulations) hardware restricted PCBs. The turn-around time for the manufacturing of the PCBs was approximately two weeks. One of the manufactured PCBs prior to integration appears in Figure 69.

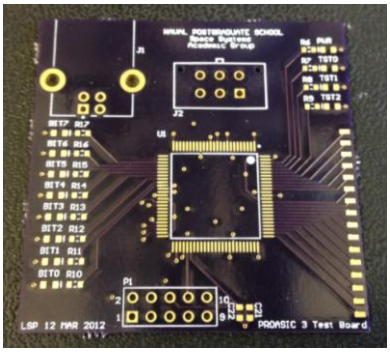


Figure 69. Manufactured four-layer PCB with gold surface pads and through-hole vias.

One the PCBs were received, work began on the integration of surface-mount components onto the PCB. This was accomplished with a PACE PRC 200 Process Control System station equipped with a microscope and video camera output, as seen in Figure 70. The soldering iron was equipped with the smallest tip available, a 1/64 in. conical sharp tip set at an 800 F operating temperature.

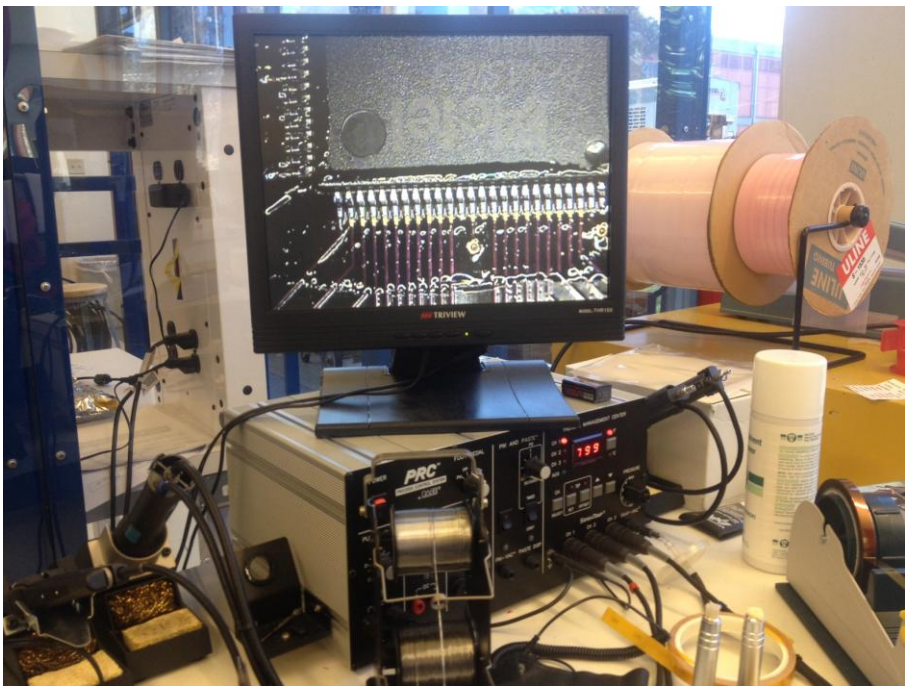


Figure 70. External camera view of ProASIC3 nano soldering to PCB board as seen through the microscope.

All soldering was done under the microscope with the finest diameter solder in the laboratory, Kester "44" 0.38 mm lead-tin (Sn 63% / Pb 37%) rosin-core. This fine diameter solder was very well suited to the 0603 surface-mount component size and VQ100 FPGA package pins. A Luxo 150-watt fiber optic illuminator provided extra lighting under magnification for ease of assemble. Thorough wetting with a Kester #186 rosin flux, type RMA Flux-Pen[28] was necessary to prevent oxidation of surfaces and ease of solder flow. The assembly of all three PCBs, as seen in Figure 71, took approximately five days.

[28] Flux-Pen™ is a trademark of Kester.

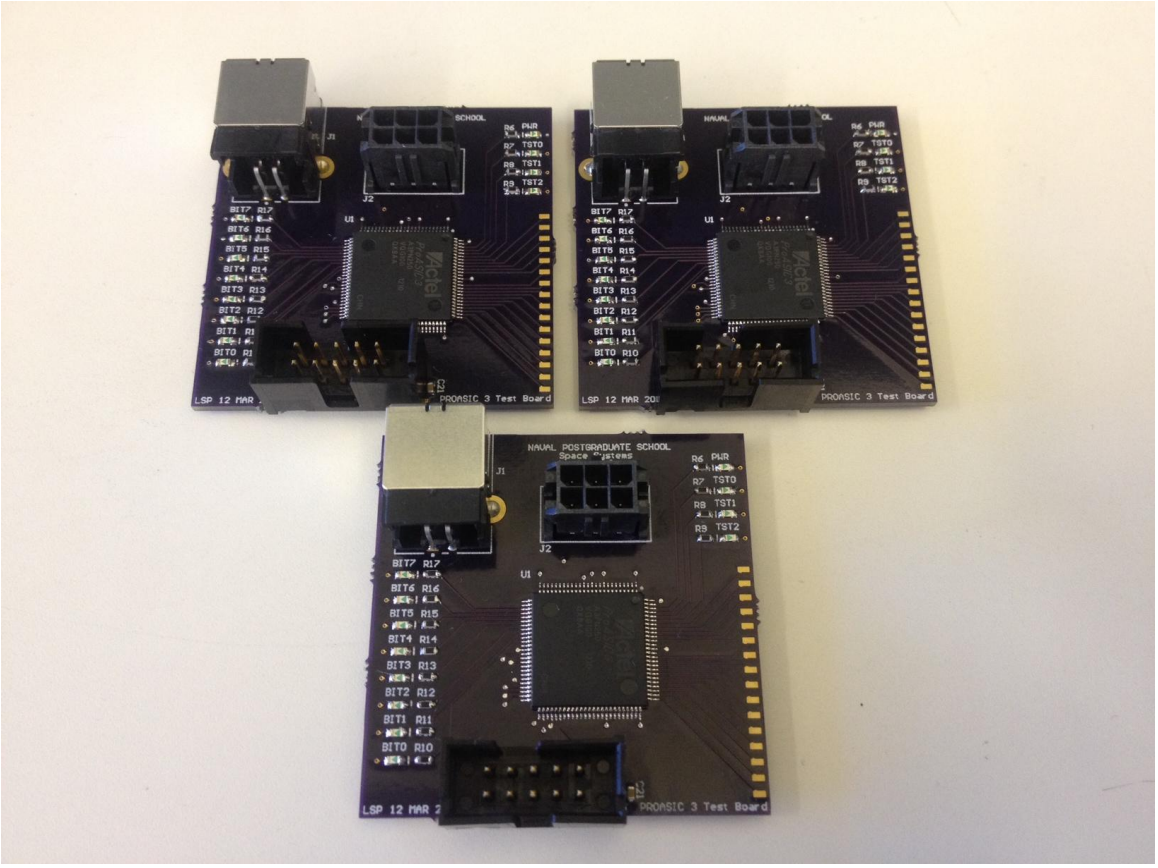


Figure 71. All three ProASIC3 Test Boards soldered and integrated—ready for testing.

After assembly of the PCBs, electrical testing of the PA3TB proceeded with the test bench setup illustrated in Figure 72. The ProASIC3 Test Board was connected to the bench-top power supplies with the FlashPro4 programmer attached to the JTAG interface. Initially, there were issues with the configuration of the FPGA, as the device would pass the JTAG scan chain with a returned identification (ID) code; however, erasing and configuration of the FPGA failed. The problem was determined to be in the power wiring to the PCB. Once a new power harness was constructed, using the appropriate end connector and clamped pin sets, the problem with erasing and configuration of the FPGA was resolved. All device tests passed and programming of several test bitstream files succeeded.

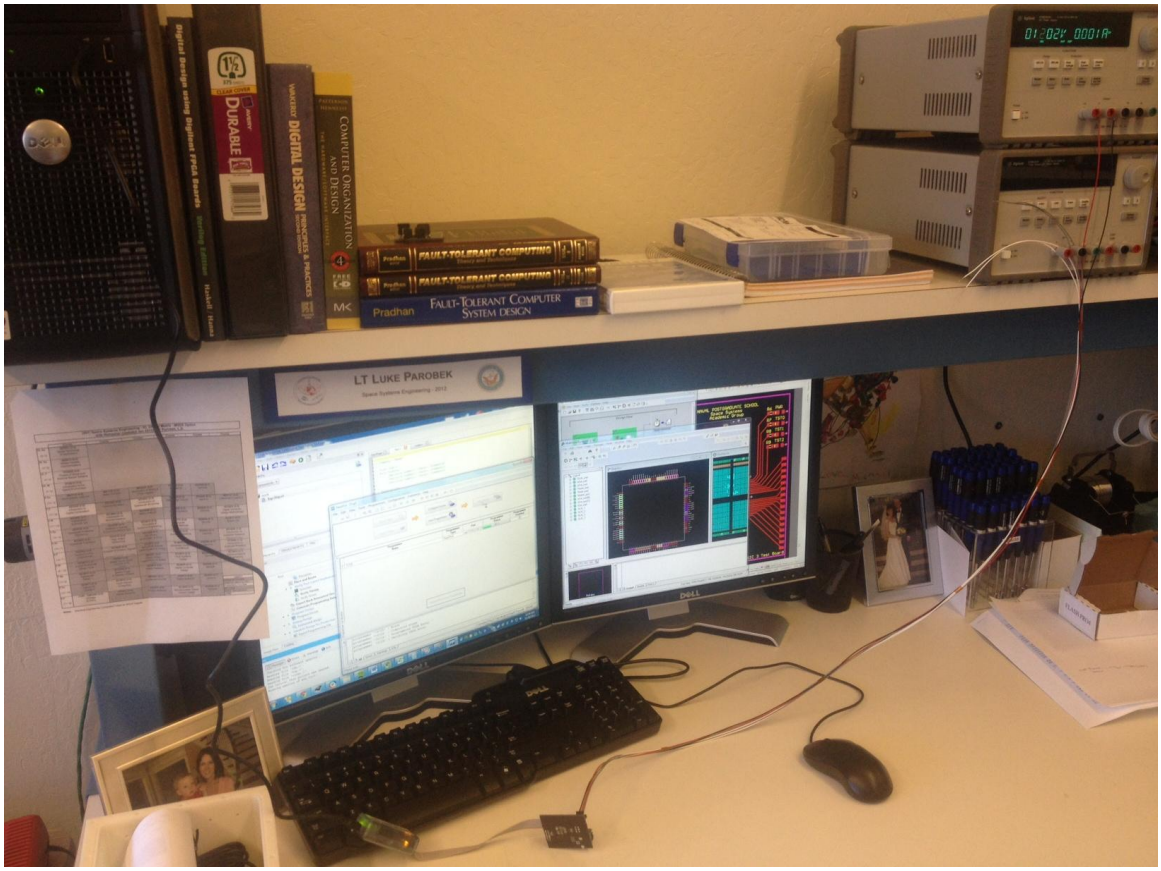


Figure 72. Test bench setup of ProASIC3 Test Board wired to power supply; FlashPro4 interface via USB to desktop computer station and programmer interface.

Initial testing of LEDs for verification through Verilog coding was successful, all initial pin-out specifications and PCB layout graphics for the trial design are located in later sections of this appendix. For the Verilog test code for LED testing, see Appendix D.

B.2 BILL OF MATERIALS

The following bill of materials specifies the required components necessary to construct the PA3TB in the same configuration specified schematically and in PCB layout. Values for the associated components are located on the schematics.

Bill of Materials

Actel A3PN250-VQ100 FPGA Breakout Board

Source Data From: ProASIC3_Nano.PrjPcb
Project: ProASIC3_Nano.PrjPcb
Variant: None

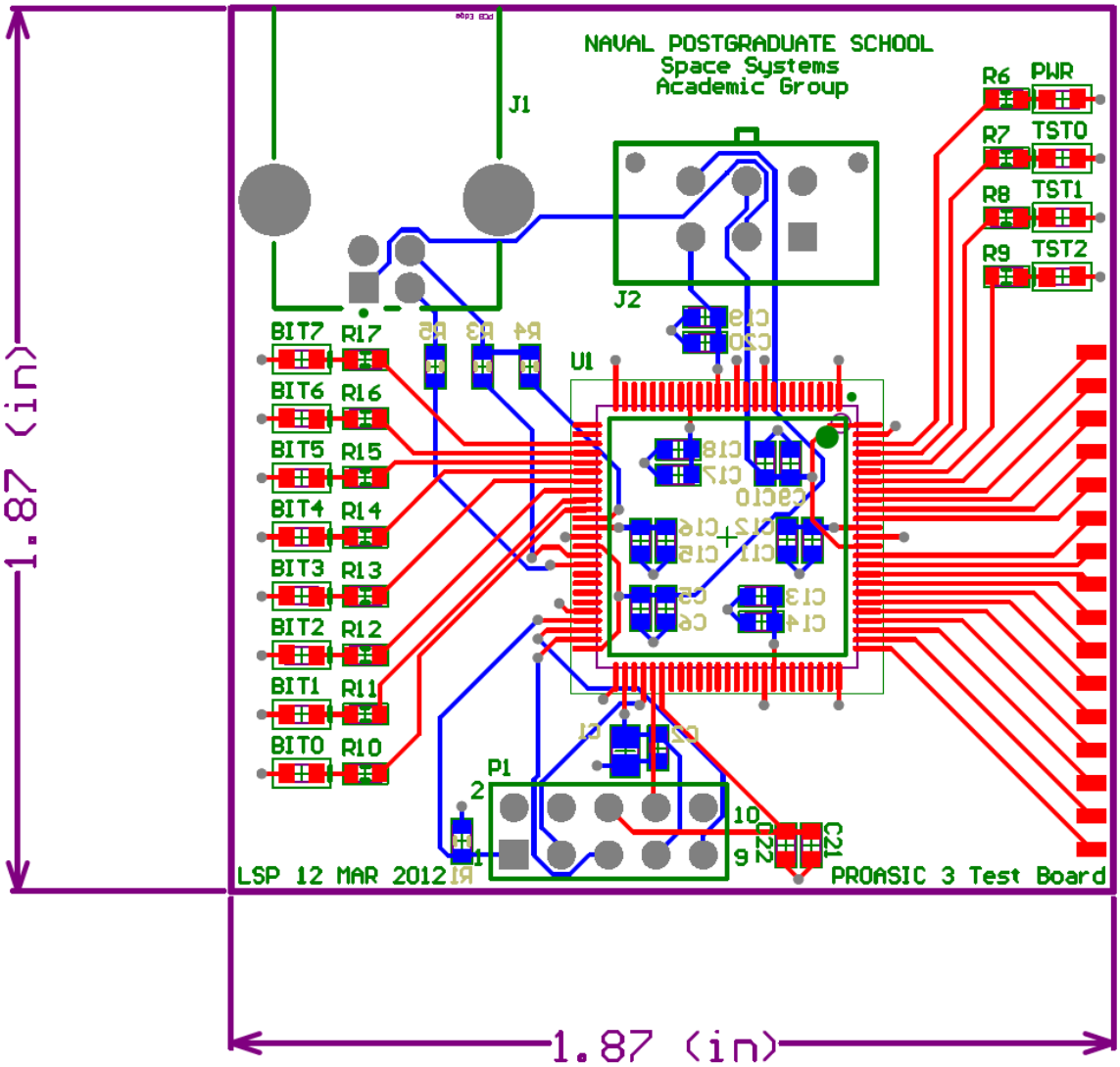
Creation Date: 11/17/2012 5:53:41 PM
Print Date: 41230 41230.74565

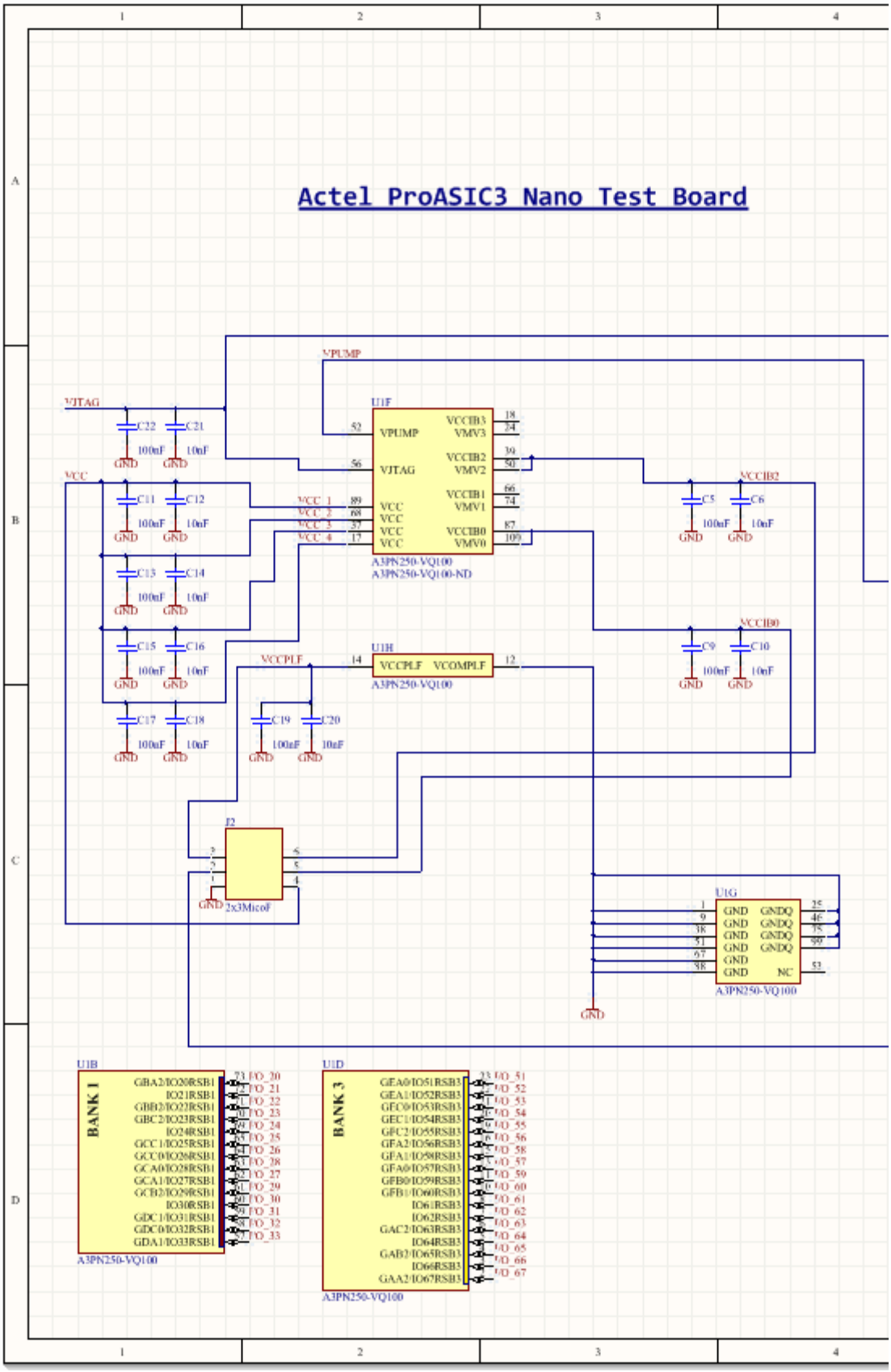
Footprint	Comment	LibRef	Designator	Description	Quantity
C0805	Cap Pol3	Cap Pol3	C1	Polarized Capacitor (Surface Mount)	1
1608[0603]	Cap Semi	Cap Semi	C2, C5, C6, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22	Capacitor (Semiconductor SIM Model)	17
QTLP601C-4	QTLP601C-4	QTLP601C-4	D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12		12
1-1470156	1-1470156-1	1-1470156-1	J1	USB 1.1, Right Angle, Thru-Hole, B Type, Receptacle, 4 Position, White	1
43045-0614 SAD SJ	2x3MicoF	2x3MicoF	J2	P-Pod	1
HDR2X5	Header 5X2	Header 5X2	P1	Header, 5-Pin, Dual row	1
J1-0603	Res3	Res3	R1, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17	Resistor	16
VQ100_L	A3PN250-VQ100	CMP-0140-00120-1	U1	ProASIC3 nano Flash FPGA, 68 User IOs, 250K System Gates, Standard Speed, 100-Pin VQFP, Commerical Grade	1
					50

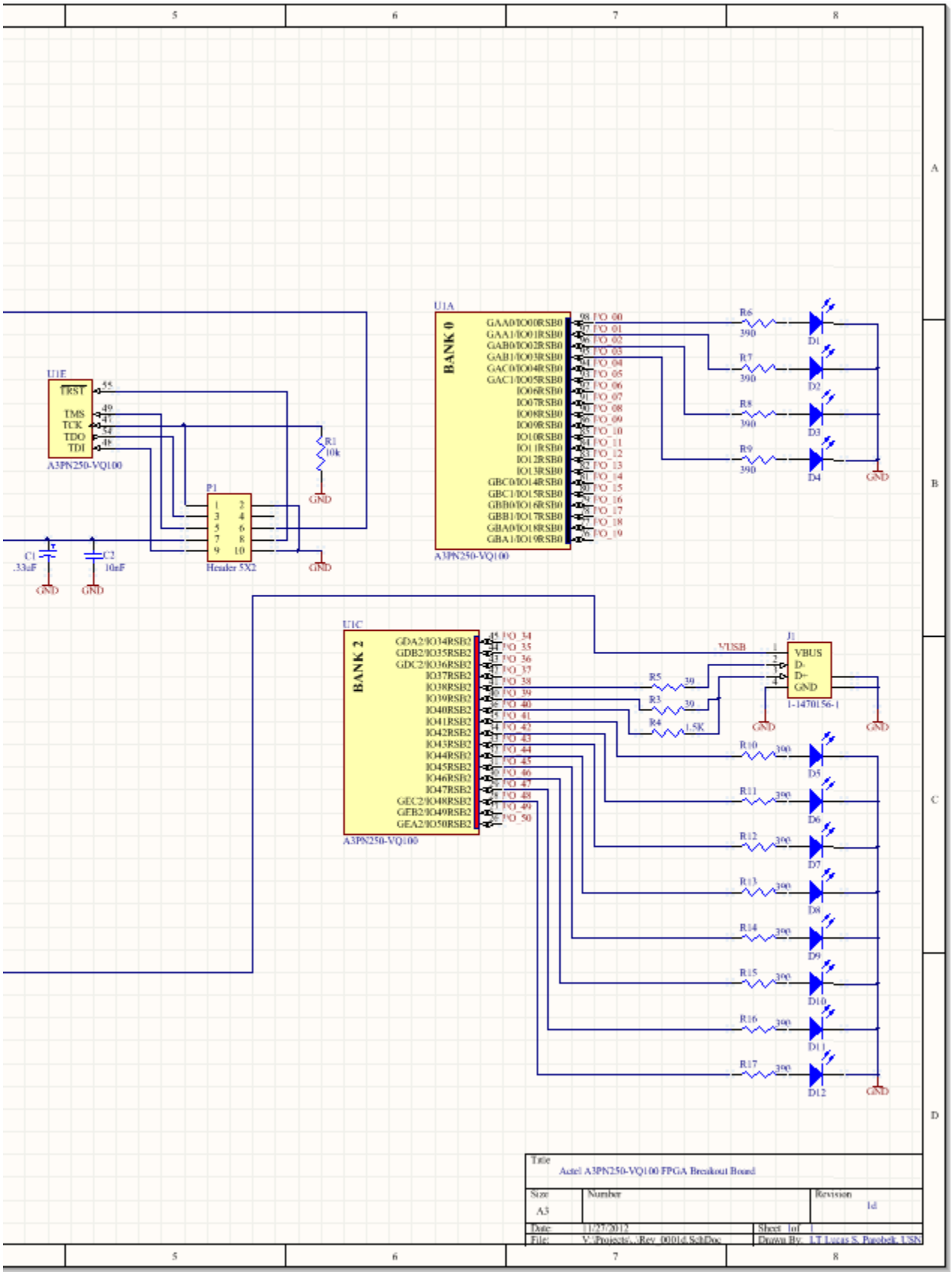
Approved	Notes	

B.3 MECHANICAL DIMENSIONS AND ELECTRICAL SCHEMATICS

The following diagrams illustrate the mechanical sizing constraints and electrical schematics for the PA3TB. The schematic is broken up into left and right-halves to fit the size constraints of the page. The schematics are available electronically for import into Altium, for reproduction or modification of the design as necessary.

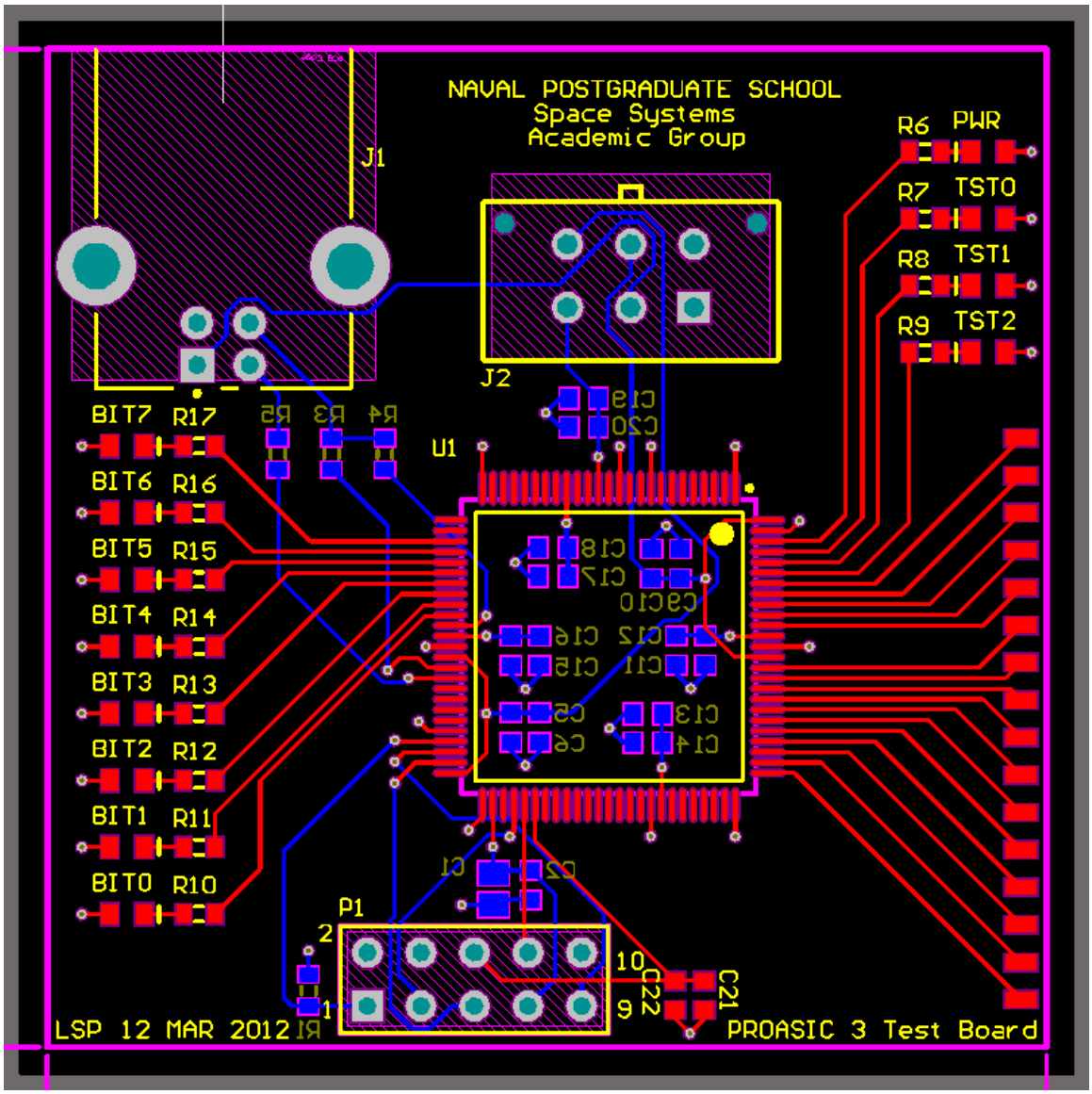


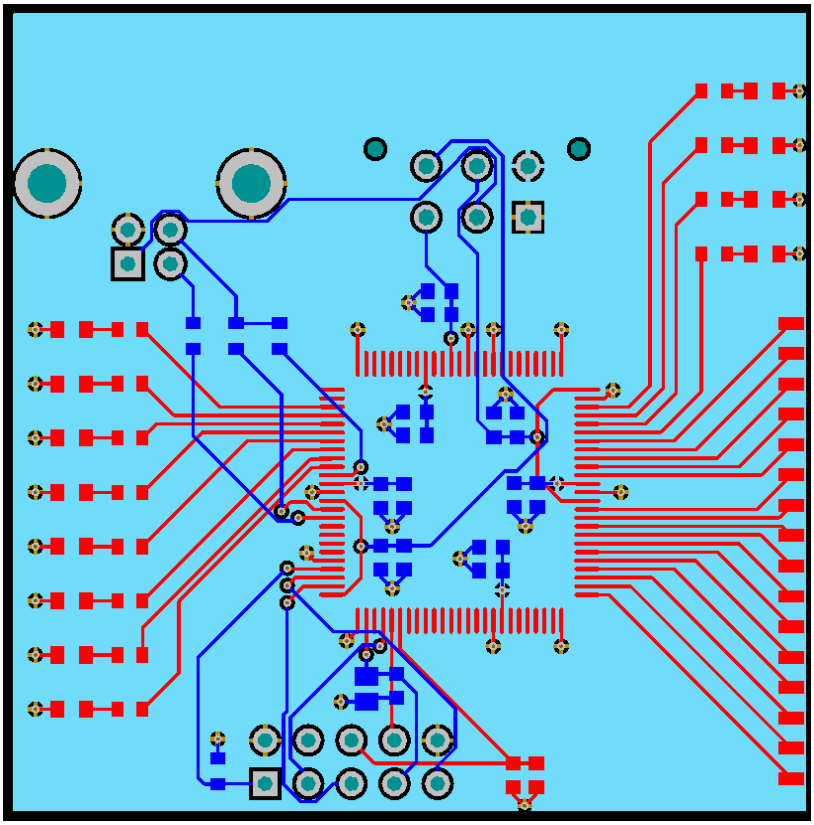

Actel ProASIC3 Nano Test Board
VPUMP
VJTAG
VCC
U1F
VPUMP
VJTAG
VCCIB3
VMV3
VCCIB2
VMV2
VCCIB1
VMV1
VCCIB0
VMV0
A3PN250-VQ100
A3PN250-VQ100-ND
VCCIB2
VCCIB0
VCCPLF
U1H
VCCPLF
VCOMPLF
A3PN250-VQ100
J2
2x3MicroF
U1G
GND
GNDQ
NC
A3PN250-VQ100
U1B
BANK 1
A3PN250-VQ100
U1D
BANK 3
A3PN250-VQ100

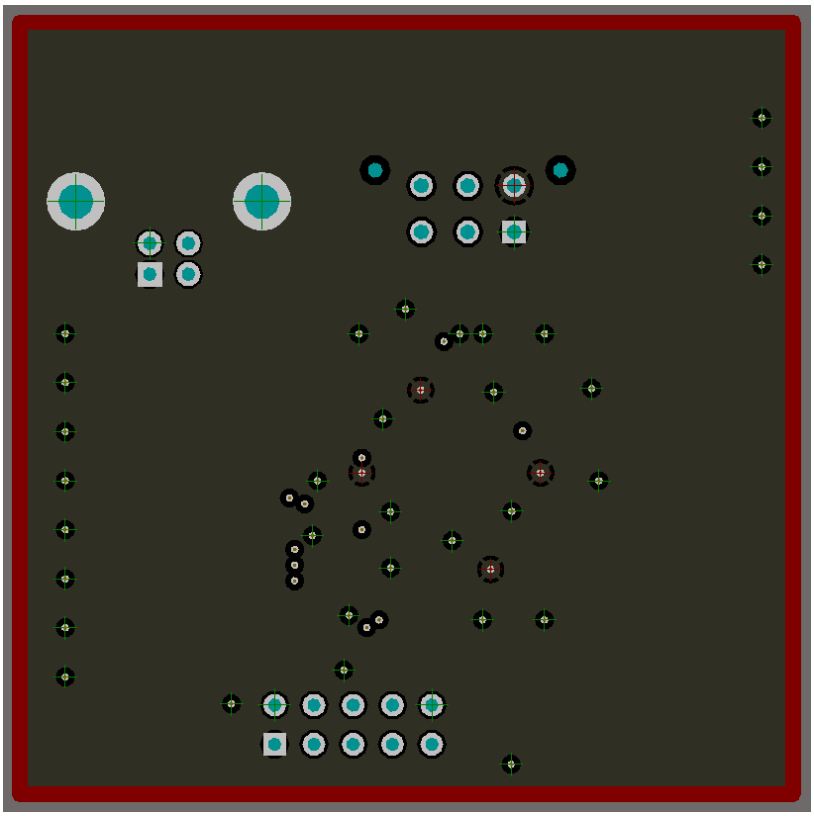

U1E
TRST
TMS
TCK
TDO
TDI
A3PN250-VQ100
P1
Header 5X2
R1
10k
C1
.33uF
C2
10uF
GND
U1A
BANK 0
A3PN250-VQ100
R6
R7
R8
R9
390
D1
D2
D3
D4
U1C
BANK 2
A3PN250-VQ100
VUSB
J1
VBUS
D-
D+
GND
1-1470156-1
R5
R3
R4
39
1.5K
R10
R11
R12
R13
R14
R15
R16
R17
D5
D6
D7
D8
D9
D10
D11
D12
Title
Actel A3PN250-VQ100 FPGA Breakout Board
Size
A3
Number
Revision
1d
Date
11/27/2012
Sheet 1 of 1
File
V:\Projects\...\Rev_0001d.SchDoc
Drawn By: LT Lucas S. Puzobek, USN

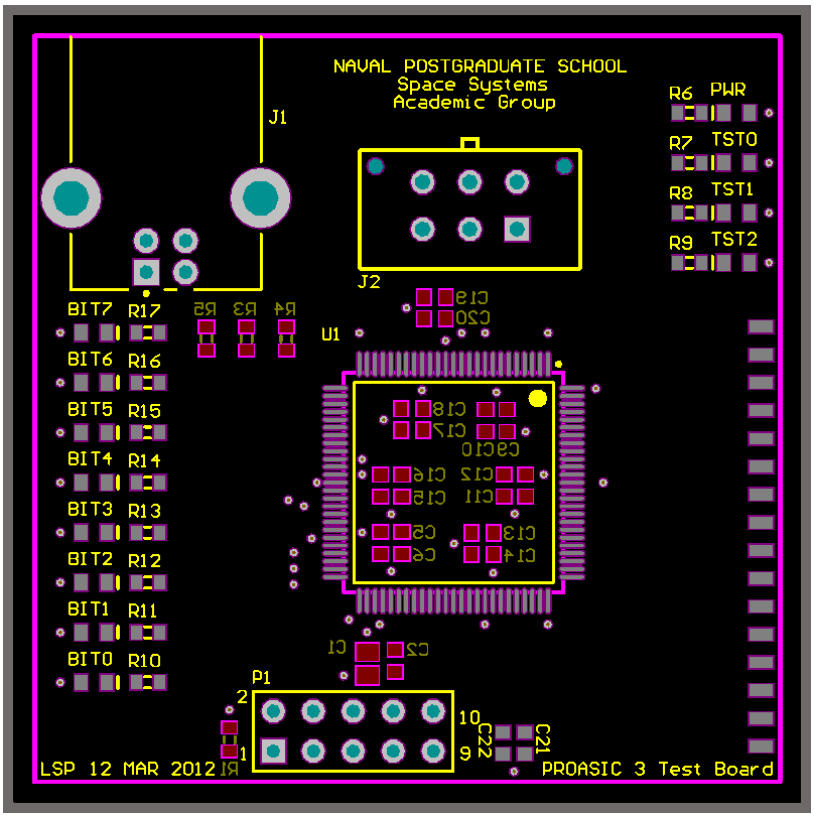
B.4 2-D PCB LAYER DIAGRAMS

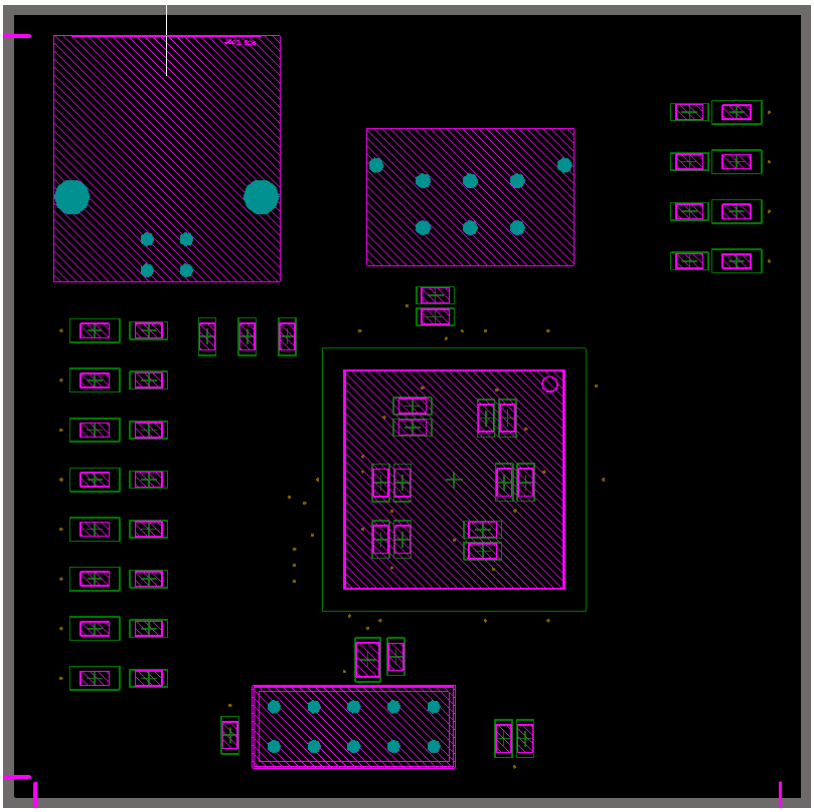
The following PCB layer diagrams illustrate the layout decisions in the PA3TB placement of components. The various diagrams included are the overall layer set, signal layer, plane layer, non-signal layer, and mechanical layer. These layouts are available electronically for import into Altium, for reproduction or modification of the design as necessary.





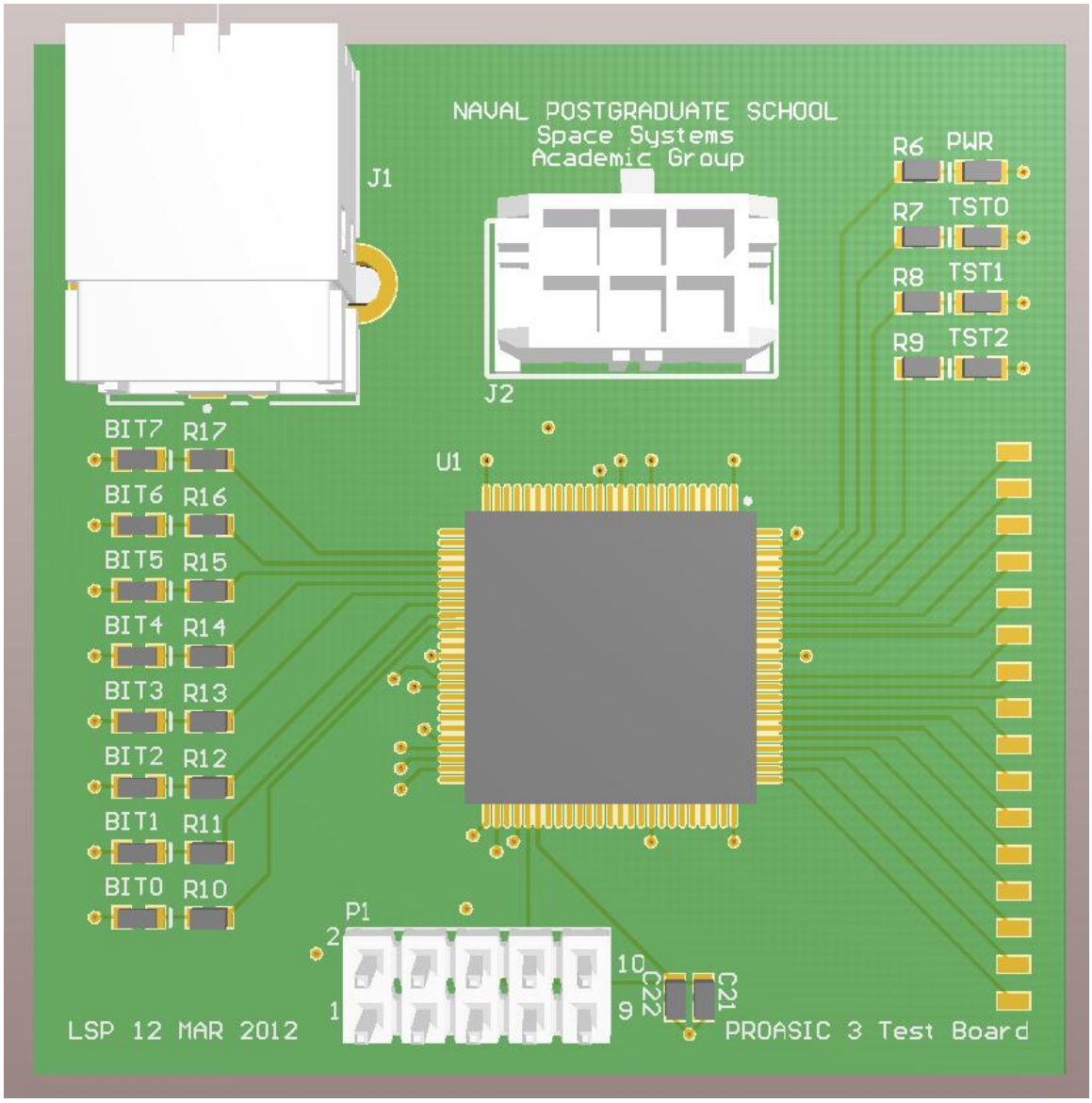


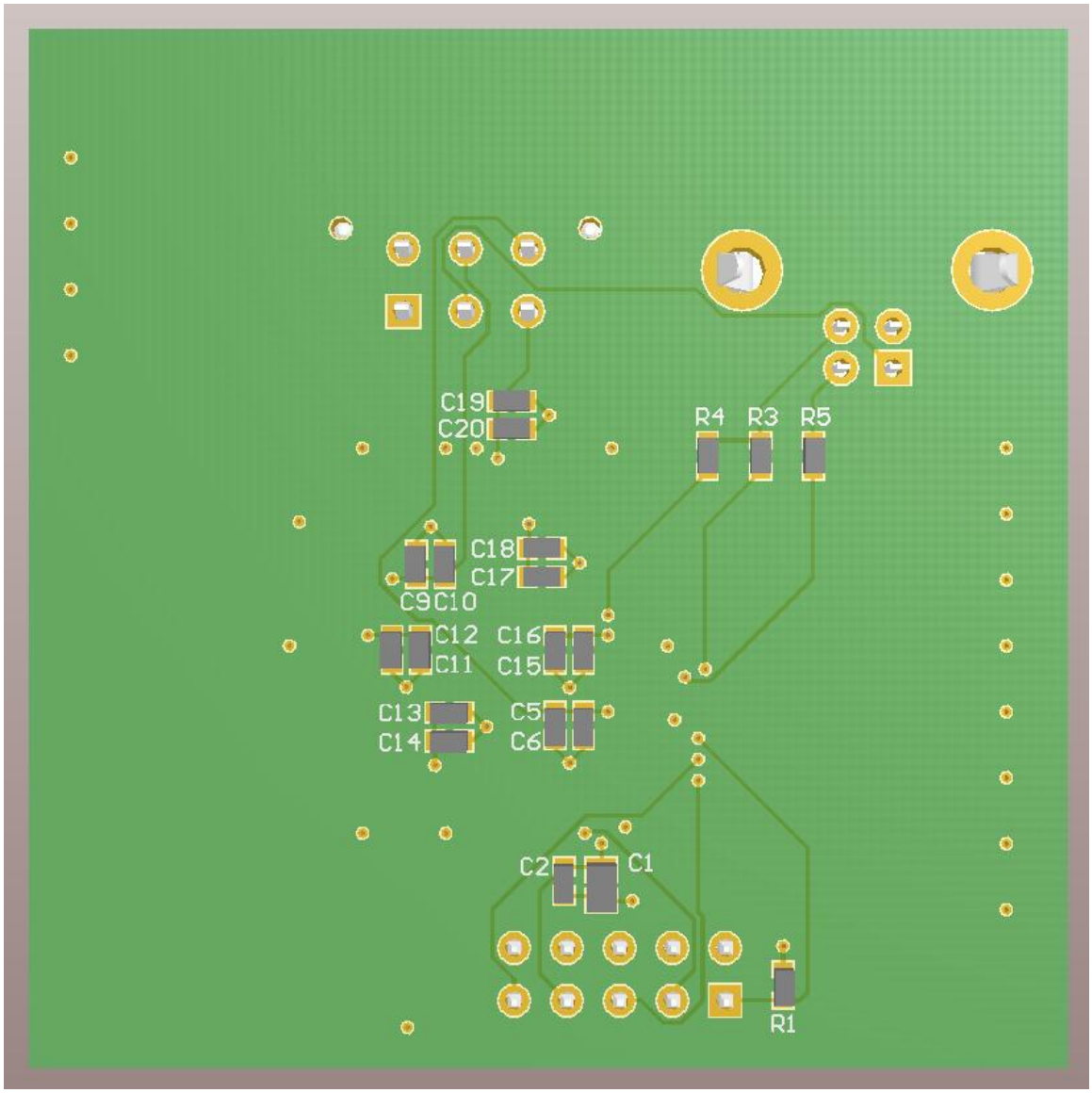

NAVAL POSTGRADUATE SCHOOL
Space Systems
Academic Group
J1
J2
R6 PWR
R7 TST0
R8 TST1
R9 TST2
BIT7 R17
BIT6 R16
BIT5 R15
BIT4 R14
BIT3 R13
BIT2 R12
BIT1 R11
BIT0 R10
U1
P1
C21
C22
LSP 12 MAR 2012
PROASIC 3 Test Board



B.5 3-D PCB COMPONENT PLACEMENT

Top and bottom face 3-D views of the PA3TB are provided to depict component fit and placement upon the PCB. These layouts are available electronically for import into Altium, for reproduction or modification of the design as necessary.





B.6 APPENDIX SUMMARY

In Appendix B, the development of the design was presented from schematic to PCB layout with reasons presented for component selection and placement. A complete description of the board assembly process was provided, with difficulties in manufacture specified. An overview of the PCB testing process for programming was outlined with the initial power supply harness issue resolution. The work in this appendix led to the development of the SADv3 presented in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SAD VERSION 3 – FLIGHT PROTOTYPE BOARD – REFERENCE, SCHEMATICS, AND PCB LAYOUT DIAGRAMS

The design files for the SAD Version 3 – Flight Prototype Board are divided into four sections: the bill of materials, schematics for electrical connections, the 2-D PCB layout documents, and 3-D component placement views. All of the associated documents were constructed using Altium Designer 10 during the development portion of this research.

C.1 SAD VERSION 3 – FLIGHT PROTOTYPE BOARD DEVELOPMENT METHODOLOGY

1. Selection of Board Size and Layer Count

The transition of sequencer from the PA3TB to SADv3 necessitates an increased board size and layer count to support the BGA FPGA and fit within the confines of the SAD enclosure. The scope for interface within the enclosure incorporates the sequencer as a motherboard. The relay firing circuitry is attached as a daughter-board via ribbon cable. The overall dimensions of the SADv3 board utilized are 9.81 in. × 3.28 in. (32.177 in^2). These dimensions, however, do not take into account the reduced area due to cut-outs for possible inclusion of battery packs, bolt holes, and rounded corners to fit within the enclosure. The size and shape of the board were chosen to conform to the already specified size of the relay board. The relay board will fit underneath the sequencer board, attached with the same bolt-hole pattern and ribbon cable for signal transmission. This board is in the process of being developed with primary work on it accomplished by another student [105]. The current PCB design for the relay board is shown in Figure 73. Manufacturing and testing of this relay board had not occurred at the time of publication of this thesis.

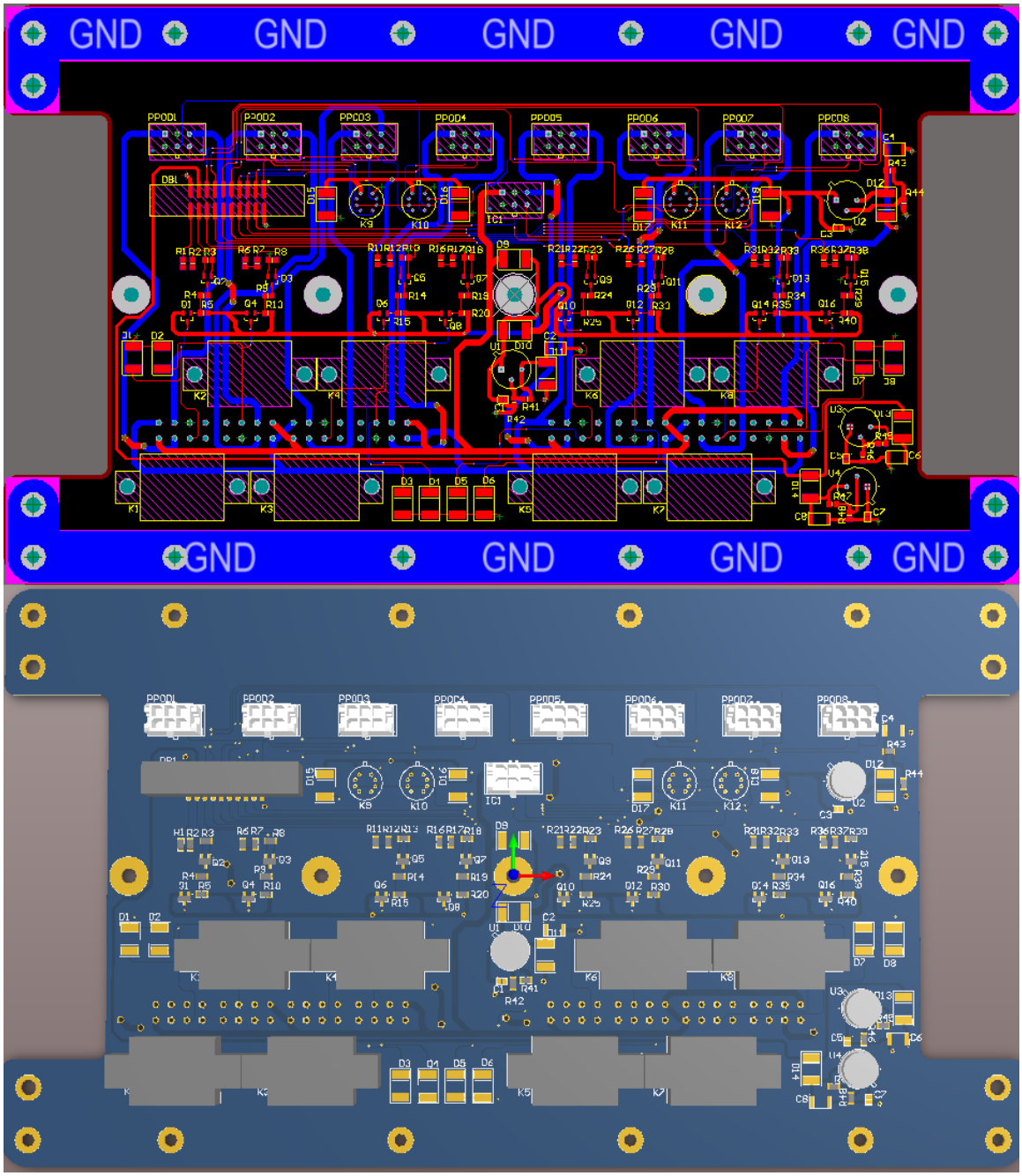


Figure 73. Current 2-D layout and 3-D component placement of SADv3 Relay daughter-board with size and shape for proper enclosure fit (From [98]).

Since the SADv3 has an increased board size, there were additional areas for expansion upon the PCB itself. These were targeted for the incorporation of additional SRAM and flash-memory devices and a breadboard prototyping area. Additionally,

headers were desired for ease of connection to the various general and differential pair I/Os. These were accommodated by three 40-pin headers (20×2).

To properly develop the SADv3 layer design, the move to a six-layer PCB was accomplished. Attempting to fan-out the BGA FPGA was possible at this layer count; however, there were insufficient power planes internal to the PCB to fit the overall voltage scheme. Due to the requirements for a 3.3, 2.5, and 1.2 V_{DC} power supply, plus the addition of a ground plane, the decision was made to transition to an eight-layer design. This allows for the incorporation of the ground and power planes internal to the PCB with the outer four planes consisting of signal layers. The power planes and ground plane were structured in a manner to limit the capacitance between layers or any inductive effects that may impact signal lines. The overall layer structure of the SADv3 is illustrated in Figure 74. Through-hole vias were used throughout the design for coupling between layers.

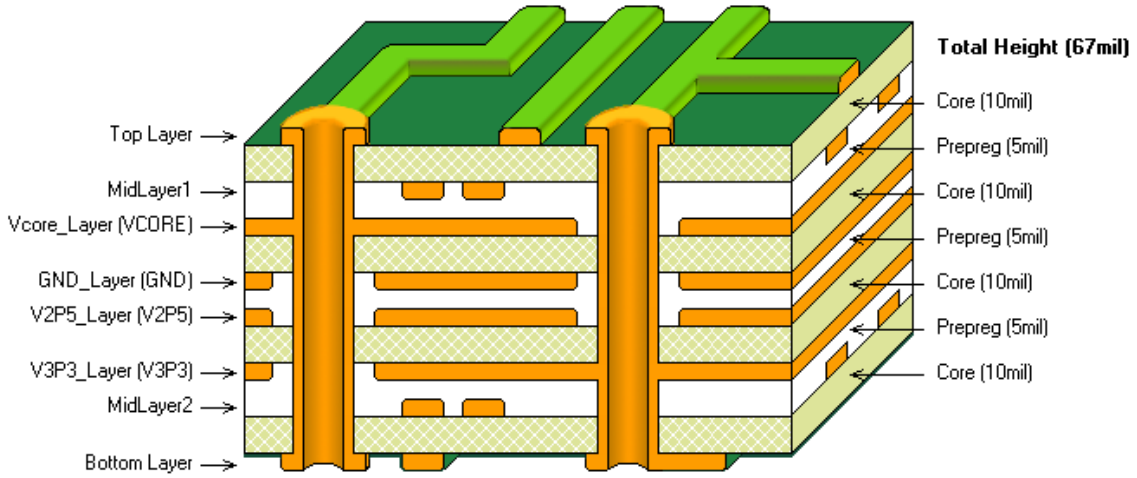


Figure 74. Eight-layer stack of the SAD Version 3 – Flight Prototype Board with internal power planes and external signal layers.

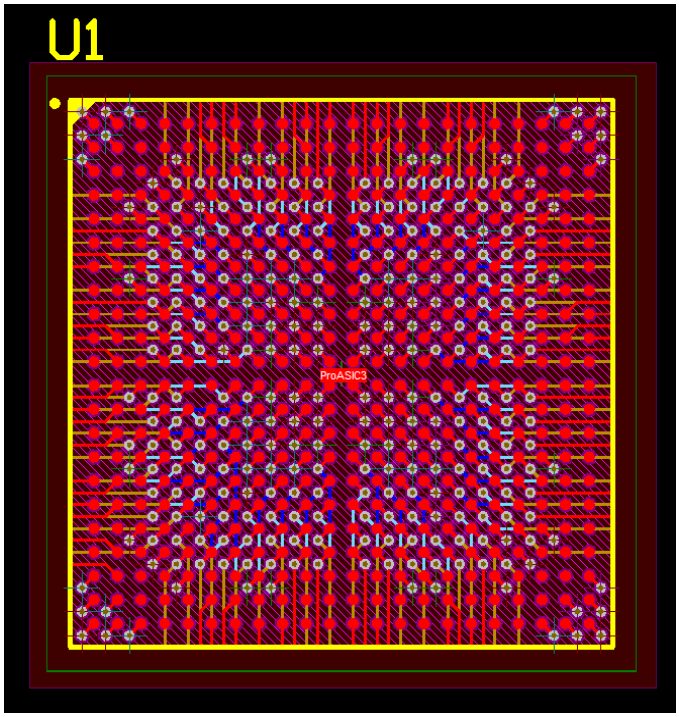


Figure 75. Final fanned-out configuration of the ProASIC3EL FPGA with multiple signal layers and all vias visible.

The final fan-out configuration of the ProASIC3EL FG484 BGA FPGA is shown in Figure 75. This configuration was chosen as the optimum solution once the additional power planes were added to the footprint of the FPGA design. This allowed for through-hole vias connecting to the necessary pins of the device while still allowing for enough room to route signal lines. This was not possible with the previous six-layer PCB design as there were certain pins trapped within the footprint. In addition, the revised eight-layer PCB allowed for the proper fan-out configuration of a crossed middle-channel open beneath the footprint of the FPGA, as illustrated in Figure 76. This allows enough room for various power decoupling capacitors to be placed immediately below and near the BGA footprint vias on the bottom plane of the board. To properly fit these surface-mount technology (SMT) capacitors within the footprint of the center-channel left, sizing of the capacitors must be equal to or less than 0603. Fortunately, many of the secondary required components to support the FPGA could be maintained at the 0603 footprint; although, some of the smaller resistors chosen necessitated 0402 sizing. Past experience shows manually soldering at less than 0603 size to be problematic.

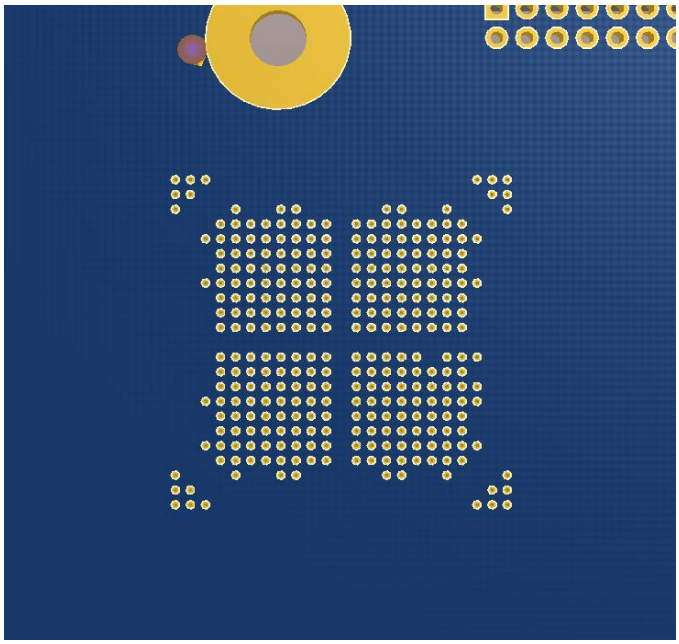


Figure 76. Bottom plane pattern of vias within FPGA footprint on eight-layer PCB with room for center-channel power decoupling capacitors.

2. Selection of Components

In contrast to the off-board power supply scheme of the PA3TB, the SADv3 requires all associated power supply hardware to be co-located upon the PCB for proper operation. Due to the increased requirement for multiple voltage levels, the supporting circuitry designed for power distribution included 3.3, 2.5, and 1.2 V_{DC} levels. Texas Instruments low-dropout (LDO) low-power linear voltage regulators (P/N TPS79601) were chosen as the means of reducing the +5 V_{DC} input voltage to the required levels. These devices feature high power supply rejection ratio (PSRR), ultralow-noise, fast start-up, and excellent line and load transient responses [106].

For future expansion purposes, the inclusion of SRAM and flash-based memory was necessary to allow for more data handling capability by the FPGA. Choice of the memory devices used was done in concert with the reference design for the ProASIC3 development board. This was done to allow for design modifications where necessary while allowing for similar pin specification and design constrain files to be utilized. In the case of the sequencer, these devices will not be necessary for inclusion on the PCB, and the footprints can be left open if not included. For the CMOS Static RAM, 4-Mbit

(256k words × 16-bit) devices were chosen from Cypress Semiconductor (P/N CY7C1041DV33) [106]. The logic block diagram of these SRAM chips can be seen in Figure 77 with proper address and data lines specified.

The embedded flash-memory used for this design consisted of two Intel / Numonyx™ 0.13 μm NOR-based, 64-Mbit density devices (P/N JS28F640J3D-75). Specified features include reliable, low-voltage capability (3 V_{DC} read, program, and erase) with high-speed, low-power operation [107]. These feature 75 ns initial access speed and operation at the 3.3 V_{DC} voltage level. The 64-Mbit capacity (8 Mbyte) is configured as sixty-four 128-kbyte erase blocks. Security features are also present on the device that prevent altering of code through different code protection schemes that can be implemented based on user requirements [107]. The logic block diagram of these flash-memory chips can be seen in Figure 78. All lines indicated are utilized as per specification in the manufacture's datasheet [107].

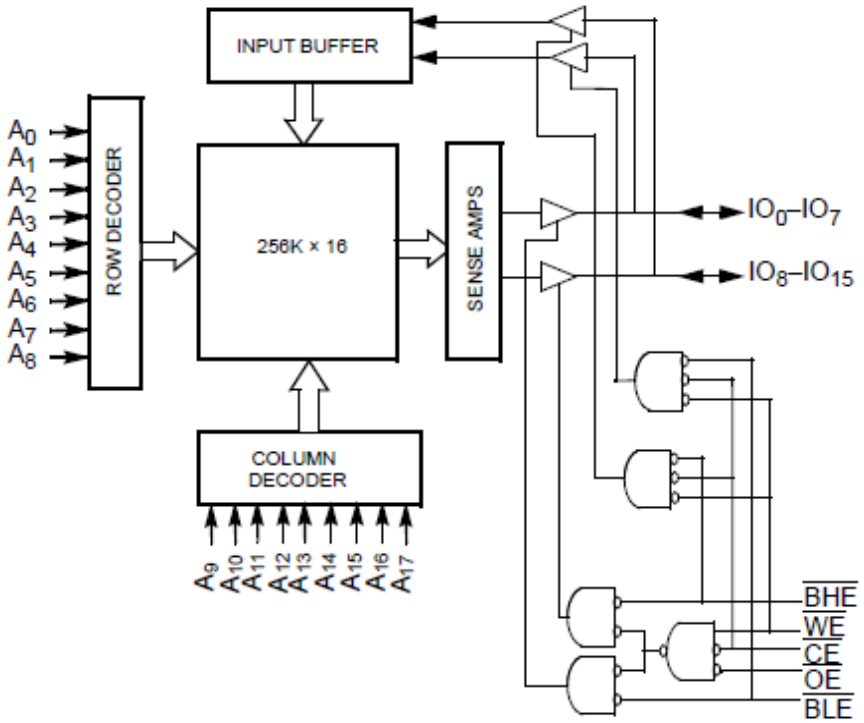


Figure 77. Logic block diagram of Cypress Semiconductor 4-Mbit (256k words × 16-bit) SRAM devices illustrating address and data line use (From [99]).

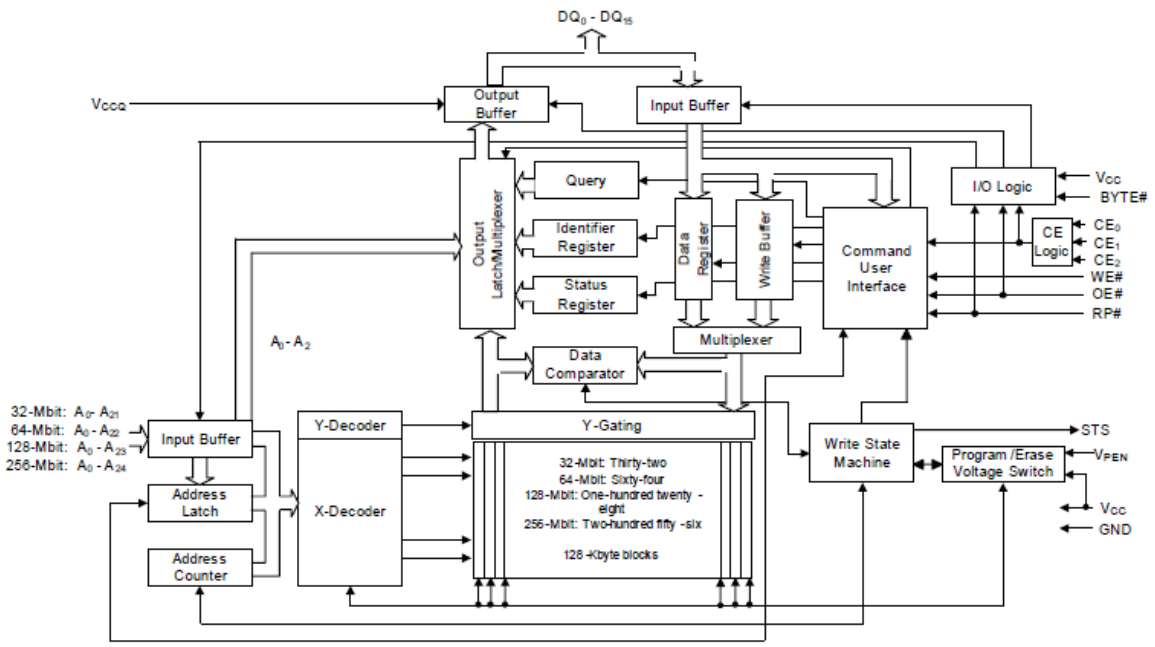


Figure 78. Logic block diagram of Numonyx™ 64-Mbit embedded flash-memory devices illustrating address, data, and signal line use (From [107]).

The incorporation of a crystal clock oscillator was necessary to allow for future timing accuracy as a secondary clock signal source for the FPGA. Similar to the reference design, an Abracon 48.0-MHz through-hole oscillator was chosen for inclusion (P/N ACHL 48.0MHz-EK). This design features a compatible 3.3 VDC supply voltage, CMOS and transistor-transistor logic (TTL) compatibility, and tight frequency stability on the order of ± 30 parts per million (ppm) [108].

The remainder of components chosen was done to maximize the design similarities between the PA3TB and the SADv3. The JTAG interface design was reutilized with capacitor differences utilized where necessary. Additionally, the test LEDs of the PA3TB were included for troubleshooting purposes. An ‘always-on’ power LED was substituted for the PWR LED of the PA3TB; this change was made for easy of seeing the board power application, as a commanded signal should not be necessary for board power indication.

The full ‘Bill of Materials’ for the SAD Version 3 – Flight Prototype Board is provided in Appendix B.

3. Discussion of Schematic

The first component placed on the schematic was the ProASIC3EL BGA FPGA, with instantiation provided via the Altium Hobart vault. This online design vault specifies many of the FPGA devices made by vendors and greatly simplifies the inclusion into a design. The schematic layout of the FPGA consisted of seven banks, four of which were the associated 'Bank 0 through 4' as provided in datasheet specifications. The remaining banks were either non-connections or power and ground supply for core and bank voltages. A great deal of time was spent specifying the signal lines in a similar format to the ProASIC3 development board reference design.

The setup of the TPS79601 power supply circuitry was accomplished as specified in the datasheet, as illustrated in Figure 79. Within that diagram input voltage (V_{IN}) and output voltage (V_{OUT}) points lie at the left and right of the diagram respectively. Additionally, the unregulated input to the device (IN), enabling pin to turn on the regulator (EN), regulator ground (GND), output of the regulator (OUT), and feedback input voltage for adjustment (FB) terminals are all visible [109]. Specification of the desired output voltage utilized the following:

$$V_{OUT} = V_{REF} \times \left(1 + {R_1}/{R_2}\right) \; where \; V_{OUT} = 1.2246 \; V_{DC} \tag{2}$$

and

$$R_1 = \left(\frac{V_{OUT}}{V_{REF}} - 1\right) \times R_2 . \tag{3}$$

As per Equation (2), R_1 and R_2 must be chosen for approximately 40 µA of divider current [109]. The recommended design procedure is to choose R_2 = 30.1 kΩ to set the divider current at 40 µA, set C_1 = 15 pF for stability, and then calculate R_1 using Equation (3). The output capacitor was changed from the 1 µF value to 2.2 µF, as per datasheet specifications [109]. There were three difference circuits schematically wired to meet the positive voltage requirements of the FPGA and associated circuitry. All three voltages were placed into nets for inclusion in the internal power planes of the PCB. This

will allow for local power to be provided rather than tracing out long power lines to the FPGA and secondary components.

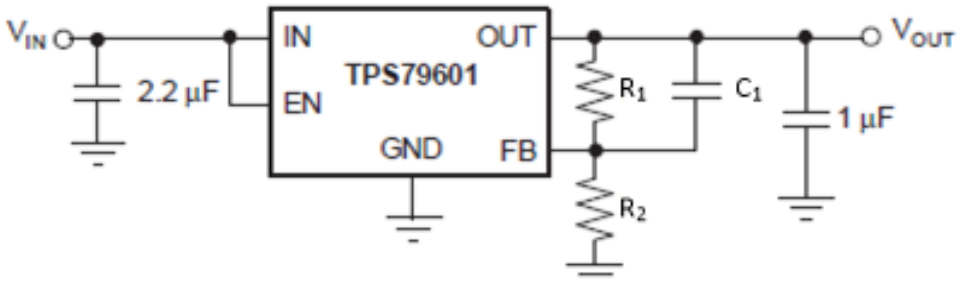


Figure 79. TPS79601 adjustable LDO regulator programming schematic for proper specification of R_1 and R_2 values (From [109]).

After development of the additional power circuitry necessary to power the board concluded, work was done to add the additional SRAM and flash-based memory to the design. The devices were wired with address and data lines as specified in datasheets [106], [107]. The schematic layouts of the SRAM and flash-memory devices are seen in Figure 80. The various data and address lines are wired to the appropriate pins of the FPGA with data control lines included as applicable. Both types of memory devices are wired to the 3.3 V_{DC} net, which allows for via coupling the associated voltage plane. A more complete representation of these layouts with their associated power decoupling capacitors may be found in Appendix B.

The general purpose and differential pair I/Os were next placed on the schematic with the various data lines being coupled to the 40-pin headers as appropriate. The general purpose I/Os utilize the 3.3 V_{DC} voltage net, while the low-voltage differential signaling (LVDS) signal lines utilize the 2.5 V_{DC} net. The schematic representation of a subset of these connectors is illustrated in Figure 81.

All of the remaining power de-coupling capacitors, JTAG circuitry, resistor banks, and clock oscillation circuitry were wired as in the PA3TB case or as specified by datasheets. The complete SAD Version 3 – Flight Prototype Board schematic is presented in Appendix B for further reference if modification to the design is desired.

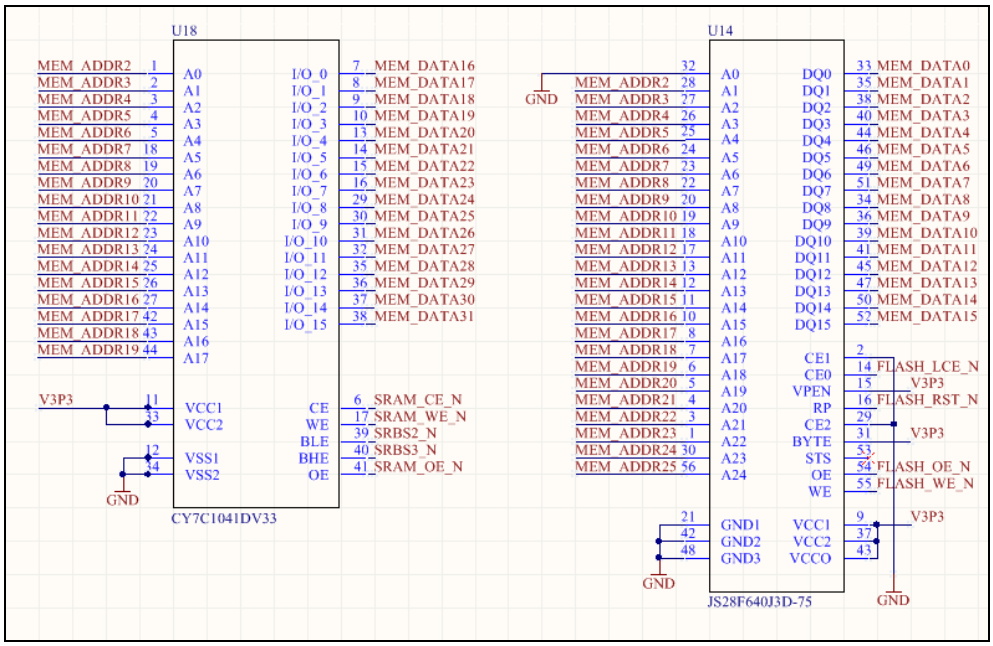


Figure 80. Schematic wiring of SRAM and flash-memory devices with address, control, and data lines as specified by design documents.

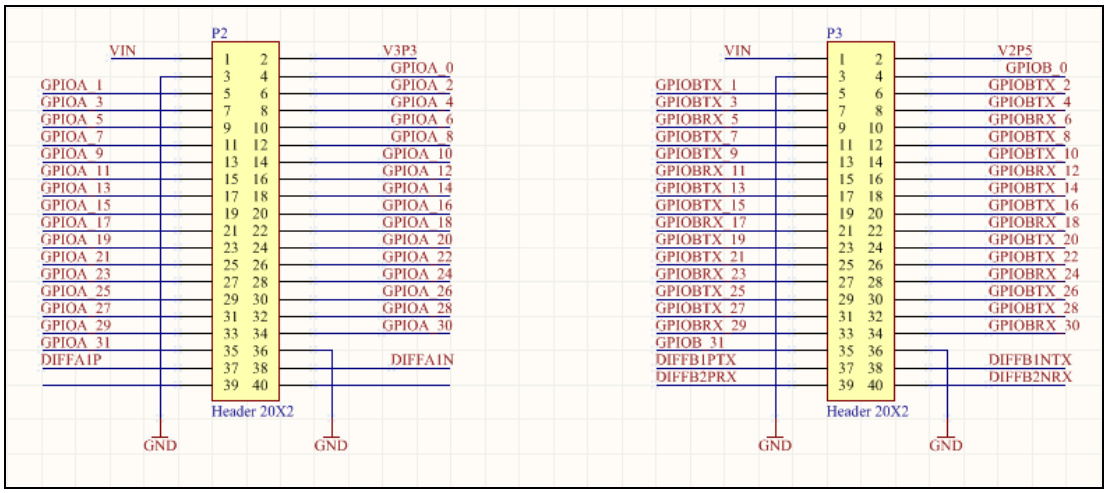


Figure 81. 40-pin general digital and LVDS headers as placed on the schematic.

4. Discussion of PCB Layout and Component Placement

After placement of the ProASIC3EL BGA FPGA, 40-pin I/O headers, breadboard prototyping area, and relay board connector as previously discussed in this chapter, the placement of the SRAM and flash-memory was accomplished. These were placed as close to the FPGA as practical to limit signal line losses between the components. The

orientation of these devices was done to best position the connectors of the memory chips to their respective pins on the FPGA. Fan-out of the memory devices was done to better route signal traces upon the four signal layers of the PCB. The flash-memory devices are located to the bottom of the FPGA when viewing the top plane, while the SRAM chips are located to the right of the FPGA footprint. The flash-memory devices come in 56-lead thin small-outline packages (TSOPs); the SRAM devices are 44-pin TSOPs (Z44-II). These devices required custom footprints and pin-outs to be built in the schematic and footprint editors in Altium. The location of these chips relative to the FPGA is seen in Figure 82.

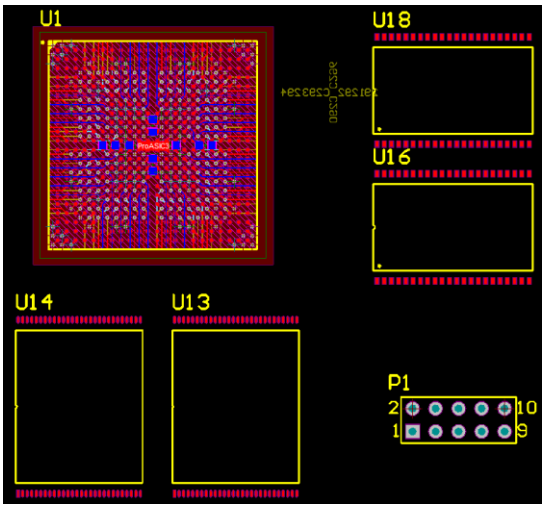


Figure 82. Location of SRAM and flash-based memory chips relative to BGA FPGA device; JTAG header visible in lower-right corner.

All of the required power supply decoupling capacitors, LDO voltage regulators, resistor banks, JTAG circuitry, crystal clock oscillator, and any additional components were then added from the schematic to their associated PCB footprint. The LDO voltage regulators used the SOT223-6 footprint, which required additional custom footprint and pin-out specification in Altium. Placement of these individual components is a tedious process, requiring many weeks of locating components to their best electrical and physical separations. Wiring of these components will require at least one to two months of concerted work to associate all traces amongst the four signal layers.

Initial test cases were run to test various Altium settings on the auto-routing of the design. In all cases, the auto-router is limited in the choices it makes regarding traces. Though there are a myriad of settings available (including trace widths, bend radius, via allocations, and placement strategies), the output of the auto-router is not optimum in the decisions it makes regarding traces. There are many instances where shorter traces could be shortened or run across less signal layers. Portions of the initial auto-route 2-D and 3-D test cases of the PCB are illustrated in Figures 83 and 84. Certain features of the auto-router, especially fan-out, will be utilized in the final PCB board design; however, the majority of signal lines must be manually routed.

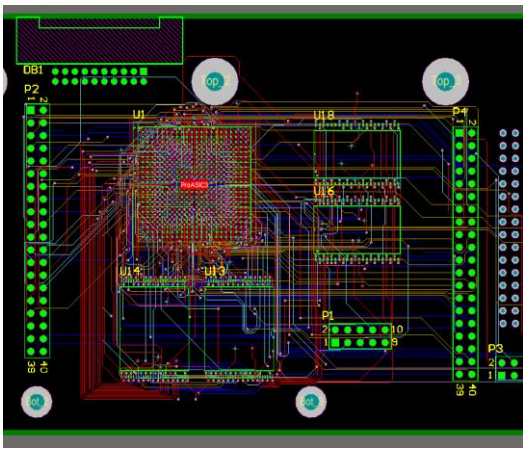


Figure 83. Auto-route testing of SRAM and flash-memory connections to ProASIC3EL FPGA across four-layer signal planes.

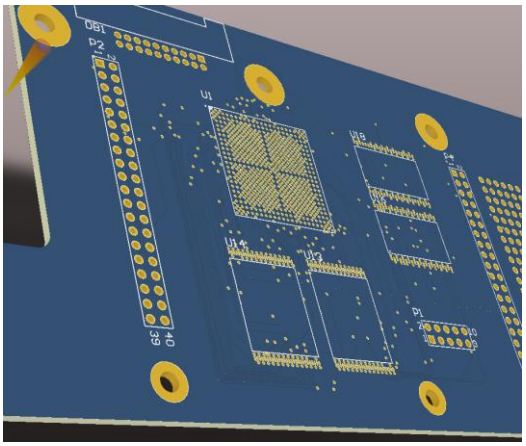


Figure 84. Initial 3-D footprint placement of FPGA and memory devices in isometric view of PCB top plane.

C.2 BILL OF MATERIALS

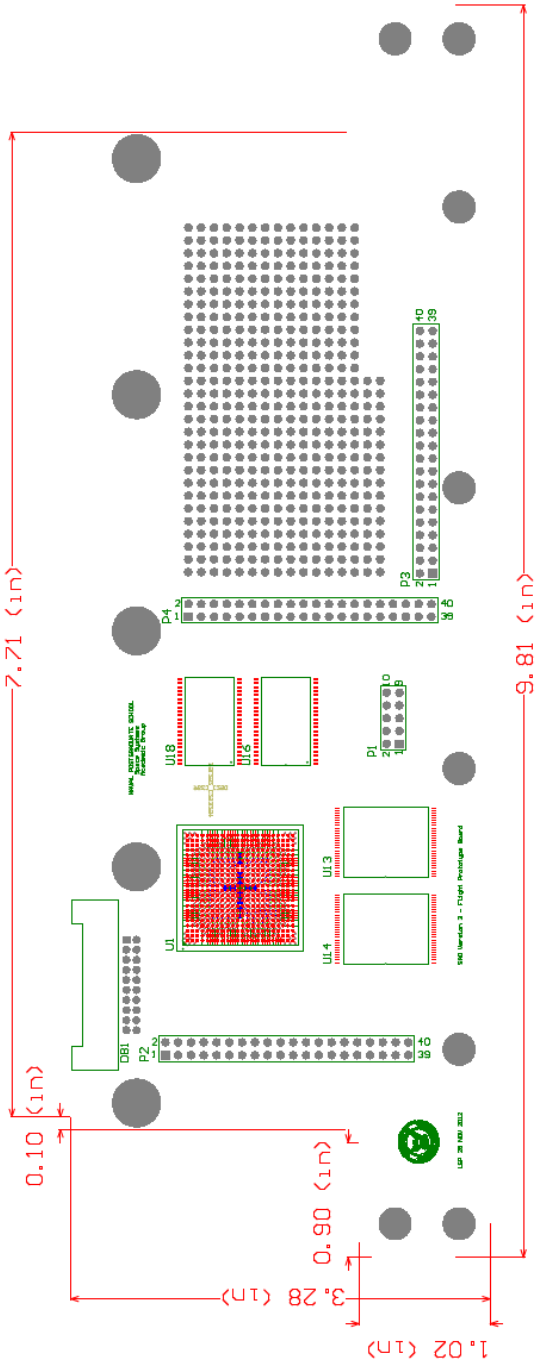
The following bill of materials specifies the required components necessary to construct the SADv3 in the same configuration specified schematically and in PCB layout. Values for the associated components are located on the schematics.

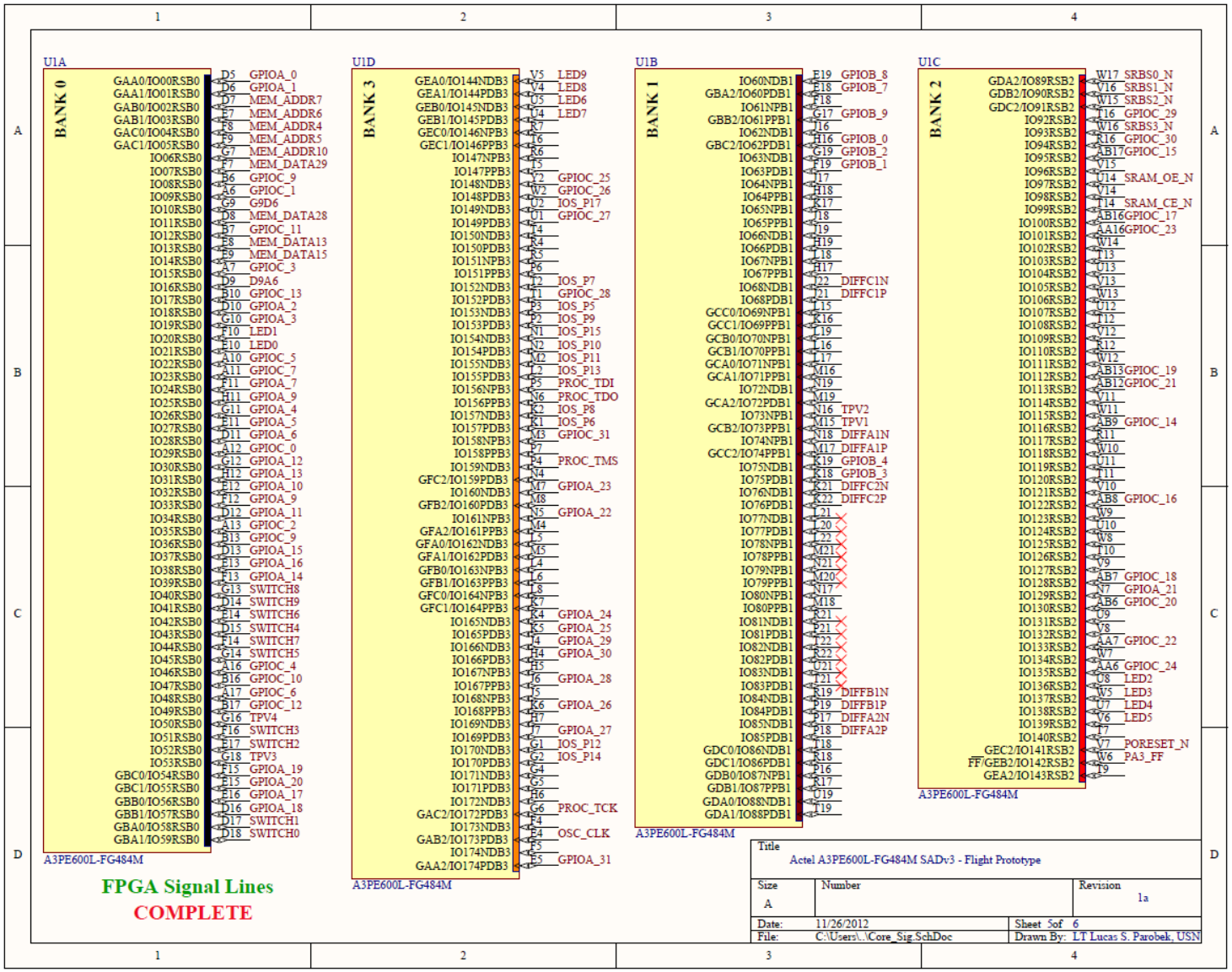
Comment	Description	Designator	Footprint	LibRef	Quantity
Cap Pol3	Polarized Capacitor (Surface Mount)	C1, C9, C11, C13, C14, C17, C227, C230, C243, C246	C0805	Cap Pol3	10
Cap Semi	Capacitor (Semiconductor SIM Model)	C2, C8, C10, C12, C16, C21, C22, C218, C219, C235, C236, C237, C238, C239, C240, C242, C245, C247, C248, C249, C250, C251, C252, C253, C254, C255, C256, C257, C258, C259, C260, C261, C264, C265, C266, C267, C268, C270, C271, C272, C273, C274, C275, C276, C277, C278, C279, C280, C281, C282, C286, C288, C289, C290, C291, C292, C293, C294, C295, C296, C297, C298, C299, C300, C302, C303, C304, C305, C306, C307, C309, C311, C312, C313, C314, C315, C317, C321, C322, C325	1608[0603]	Cap Semi	80
[TANT_107A]	Polarized Capacitor (Surface Mount)	C15	C0805	Cap Pol3	1
LM4041CEM3X-1.2	Precision Micropower Shunt Voltage Reference, 3-pin SOT-23	D4	MF03A_M	CMP-0074-00738-1	1
LTST-C150GKT	Typical RED, GREEN, YELLOW, AMBER GaAs LED	D6	3.2X1.6X1.1	LED2	1
Header 10X2A	Header, 10-Pin, Dual row	DB1	EHT-D	Header 10X2A	1
FERRITE_BEAD	SMD EMI Suppression Ferrite Bead WE-CBF, Z = 100 Ohm	FB2	SMD-0805	CMP-0220-00039-1	1

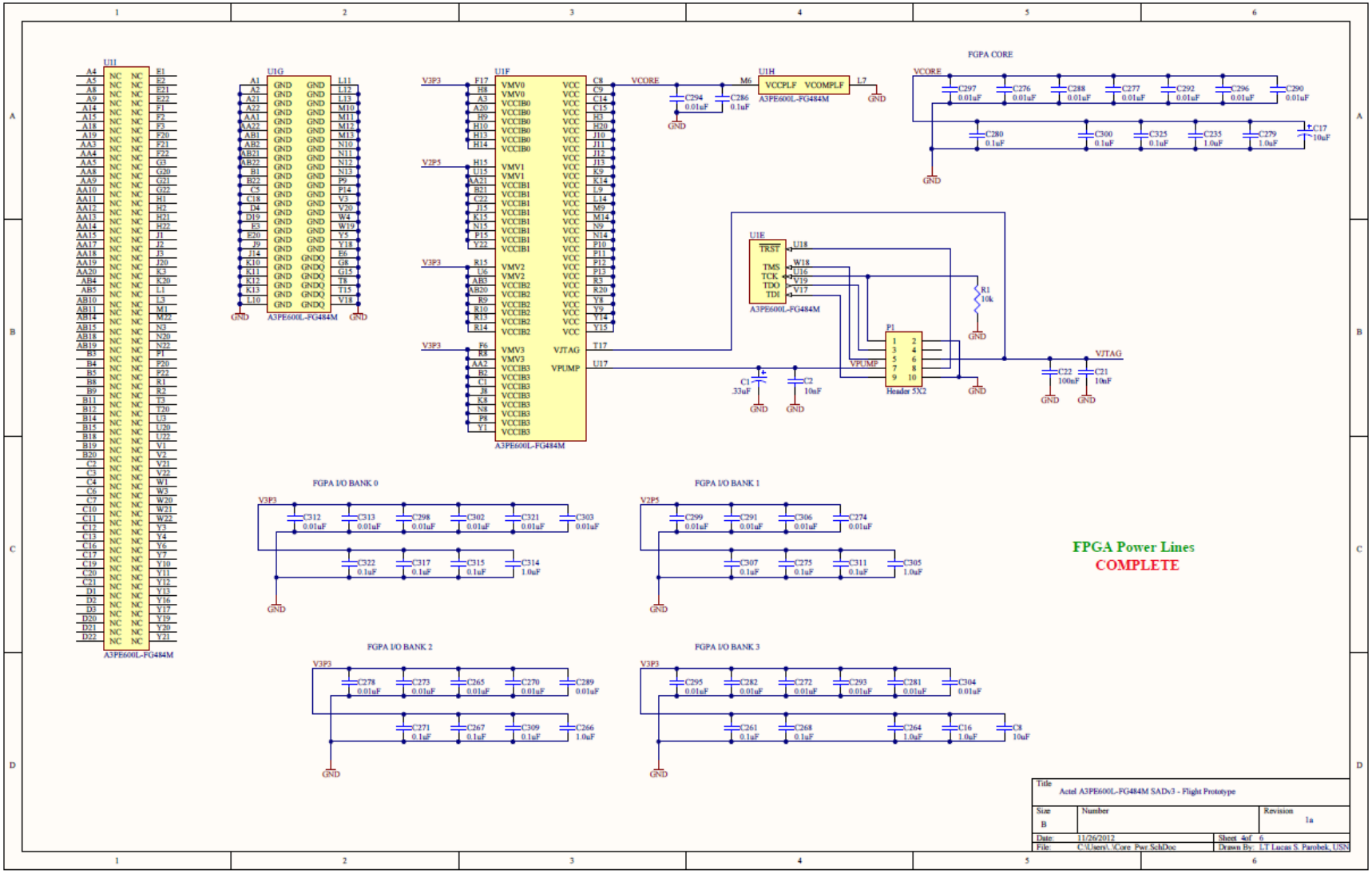
5VDC	Low Voltage Power Supply Connector	J3	KLD-0202	PWR2.5	1
Jumper	Jumper Wire	JP5	RAD-0.2	Jumper	1
Header 5X2	Header, 5-Pin, Dual row	P1	HDR2X5	Header 5X2	1
Header 20X2	Header, 20-Pin, Dual row	P2, P3, P4	HDR2X20	Header 20X2	3
FDV301N	Digital FET, N-Channel	Q3	SOT23_N	FDV301N	1
Res3	Resistor	R1, R223, R245, R246	J1-0603	Res3	4
1%	Resistor	R14, R15, R16, R17, R18, R216, R217, R220, R221, R222, R235, R236, R237, R238, R239, R240	J1-0603	Res3	16
1%	Resistor	R500, R501, R502, R503, R504, R505, R506, R507, R508, R509, R510, R511, R512, R513, R514, R515, R516, R517, R518, R519, R520, R521, R522, R523, R524, R525, R526, R527, R528, R529, R530, R531, R532, R533, R534, R535	0402	Res3	36
A3PE600L-FG484M	ProASIC3L Low-Power Flash Family FPGA, 270 User IOs, 600K System Gates, 108 Kbits RAM, 1 Kbit FlashROM, 6PLLs, 484-Ball FBGA, Military Grade	U1	FG484	CMP-0141-00090-1	1
TPS79601	Low-Dropout Linear Regulators (SOT223-6)	U12, U21, U22	SOT223-6	TPS79601	3
JS28F640J3D-75	Numonyx Embedded Flash Memory	U13, U14	56-Lead TSOP	JS28F640J3D-75	2
CY7C1041DV33	4-Mbit (256K x 16) Static R	U16, U18	TSOP Z44-II	CY7C1041DV33	2

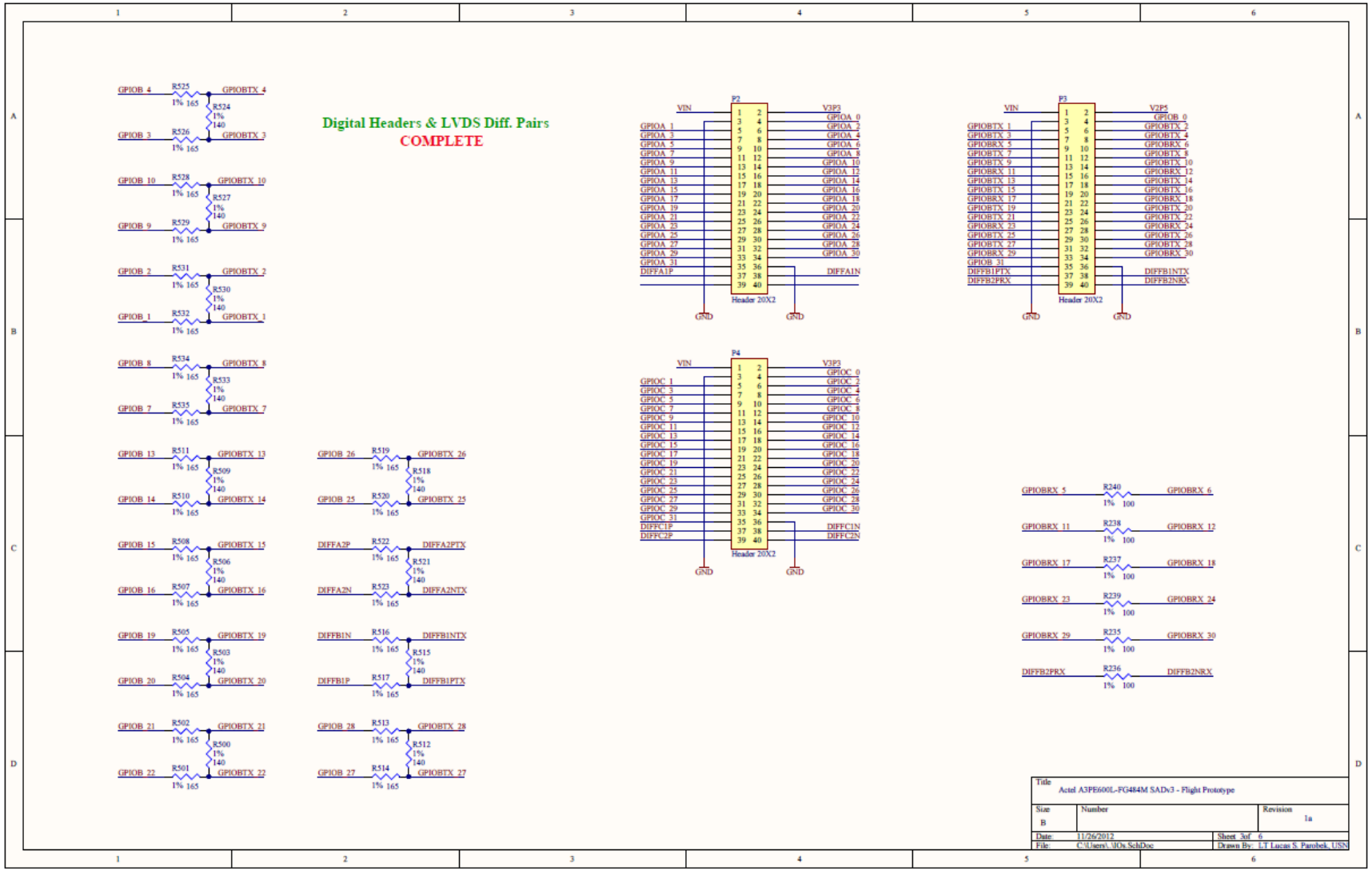
C.3 MECHANICAL DIMENSIONS AND ELECTRICAL SCHEMATICS

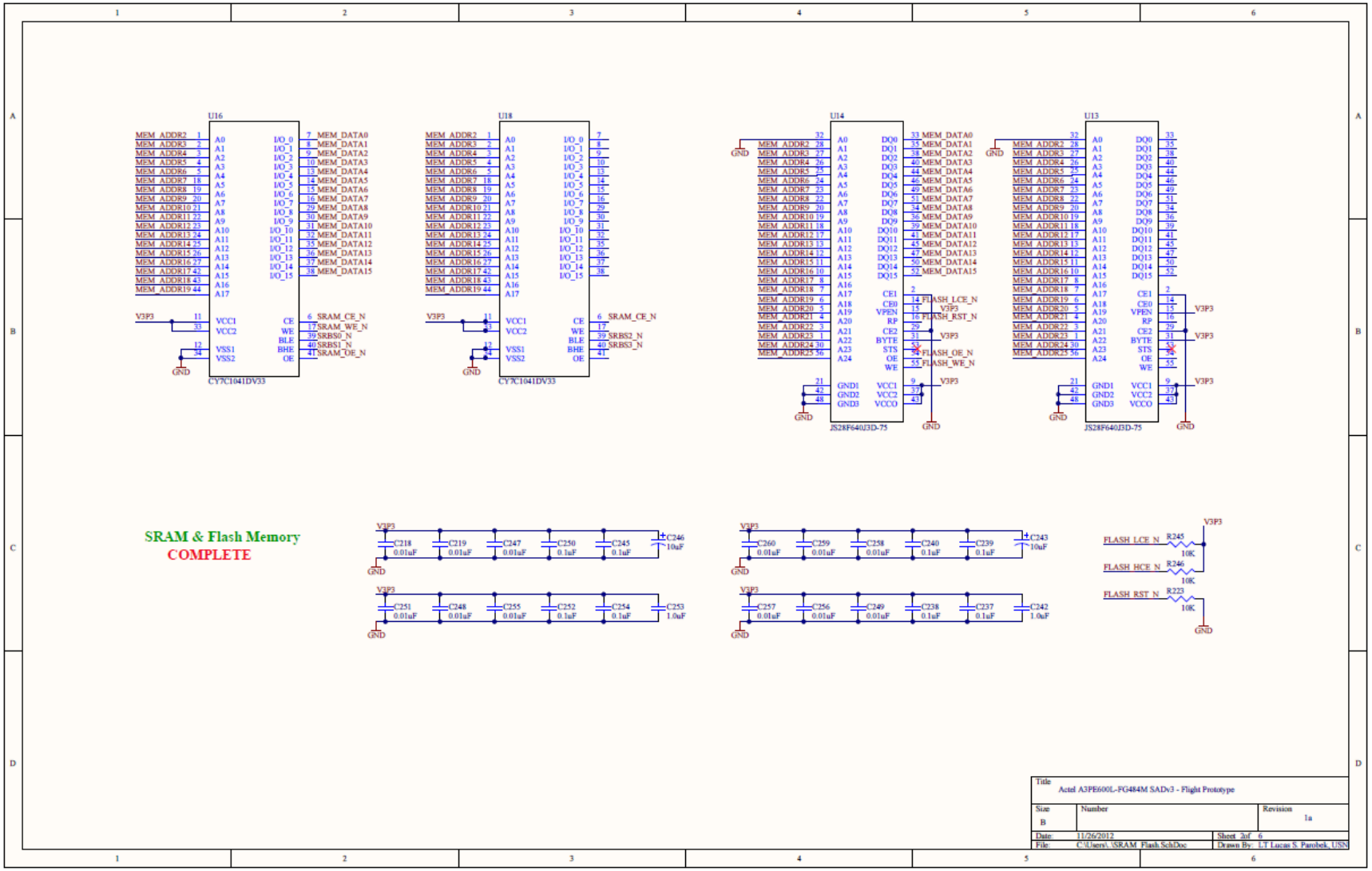
The following diagrams illustrate the mechanical sizing constraints and electrical schematics for the SADv3. The schematic is broken up into multiple pages for ease of viewing in this thesis format. An overall schematic is available electronically for import into Altium, for reproduction or modification of the design as necessary.

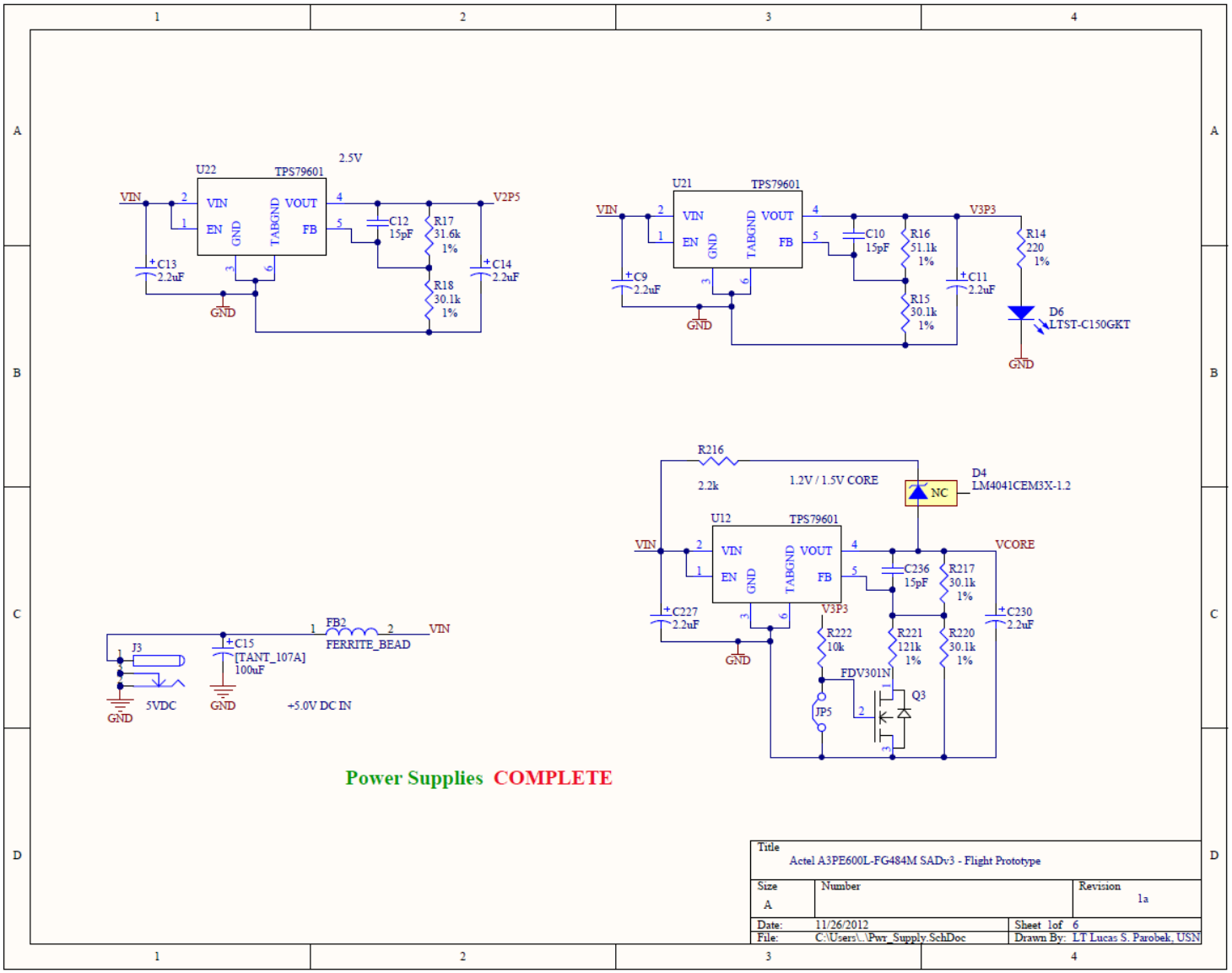



U1A
BANK 0
GAA0/IO00RSB0 D5 GPIOA_0
GAA1/IO01RSB0 D6 GPIOA_1
GAB0/IO02RSB0 D7 MEM_ADDR7
GAB1/IO03RSB0 E7 MEM_ADDR6
GAC0/IO04RSB0 F8 MEM_ADDR4
GAC1/IO05RSB0 F9 MEM_ADDR5
IO06RSB0 G7 MEM_ADDR10
IO07RSB0 F7 MEM_DATA29
IO08RSB0 B6 GPIOC_9
IO09RSB0 A6 GPIOC_1
IO10RSB0 G9 G9D6
IO11RSB0 D8 MEM_DATA28
IO12RSB0 B7 GPIOC_11
IO13RSB0 E8 MEM_DATA13
IO14RSB0 E9 MEM_DATA15
IO15RSB0 A7 GPIOC_3
IO16RSB0 D9 D9A6
IO17RSB0 B10 GPIOC_13
IO18RSB0 D10 GPIOA_2
IO19RSB0 G10 GPIOA_3
IO20RSB0 E10 LED1
IO21RSB0 E10 LED0
IO22RSB0 A10 GPIOC_5
IO23RSB0 A11 GPIOC_7
IO24RSB0 F11 GPIOA_7
IO25RSB0 H11 GPIOA_9
IO26RSB0 G11 GPIOA_4
IO27RSB0 E11 GPIOA_5
IO28RSB0 D11 GPIOA_6
IO29RSB0 A12 GPIOC_0
IO30RSB0 G12 GPIOA_12
IO31RSB0 H12 GPIOA_13
IO32RSB0 E12 GPIOA_10
IO33RSB0 F12 GPIOA_9
IO34RSB0 D12 GPIOA_11
IO35RSB0 A13 GPIOC_2
IO36RSB0 B13 GPIOC_9
IO37RSB0 D13 GPIOA_15
IO38RSB0 E13 GPIOA_16
IO39RSB0 F13 GPIOA_14
IO40RSB0 G13 SWITCH8
IO41RSB0 D14 SWITCH9
IO42RSB0 E14 SWITCH6
IO43RSB0 D15 SWITCH4
IO44RSB0 F14 SWITCH7
IO45RSB0 G14 SWITCH5
IO46RSB0 A16 GPIOC_4
IO47RSB0 B16 GPIOC_10
IO48RSB0 A17 GPIOC_6
IO49RSB0 B17 GPIOC_12
IO50RSB0 G16 TPV4
IO51RSB0 F16 SWITCH3
IO52RSB0 E17 SWITCH2
IO53RSB0 G18 TPV3
GBC0/IO54RSB0 F15 GPIOA_19
GBC1/IO55RSB0 E15 GPIOA_20
GBB0/IO56RSB0 E16 GPIOA_17
GBB1/IO57RSB0 D16 GPIOA_18
GBA0/IO58RSB0 D17 SWITCH1
GBA1/IO59RSB0 D18 SWITCH0
A3PE600L-FG484M
U1D
BANK 3
GEA0/IO144NDB3 V5 LED9
GEA1/IO144PDB3 V4 LED8
GEB0/IO145NDB3 U5 LED6
GEB1/IO145PDB3 U4 LED7
GEC0/IO146NPB3 R7
GEC1/IO146PPB3 T6
IO147NPB3 R6
IO147PPB3 T5
IO148NDB3 Y2 GPIOC_25
IO148PDB3 W2 GPIOC_26
IO149NDB3 U2 IOS_P17
IO149PDB3 U1 GPIOC_27
IO150NDB3 T4
IO150PDB3 R4
IO151NPB3 R5
IO151PPB3 P6
IO152NDB3 T2 IOS_P7
IO152PDB3 T1 GPIOC_28
IO153NDB3 P3 IOS_P5
IO153PDB3 P2 IOS_P9
IO154NDB3 N1 IOS_P15
IO154PDB3 N2 IOS_P10
IO155NDB3 M2 IOS_P11
IO155PDB3 L2 IOS_P13
IO156NPB3 P5 PROC_TDI
IO156PPB3 N6 PROC_TDO
IO157NDB3 K2 IOS_P8
IO157PDB3 K1 IOS_P6
IO158NPB3 M3 GPIOC_31
IO158PPB3 P7
IO159NDB3 P4 PROC_TMS
GFC2/IO159PDB3 N4
IO160NDB3 M7 GPIOA_23
GFB2/IO160PDB3 M8
IO161NPB3 N5 GPIOA_22
GFA2/IO161PPB3 M4
GFA0/IO162NDB3 L5
GFA1/IO162PDB3 M5
GFB0/IO163NPB3 L4
GFB1/IO163PPB3 L6
GFC0/IO164NPB3 L8
GFC1/IO164PPB3 K7
IO165NDB3 K4 GPIOA_24
IO165PDB3 K5 GPIOA_25
IO166NDB3 J4 GPIOA_29
IO166PDB3 H4 GPIOA_30
IO167NPB3 H5
IO167PPB3 J6 GPIOA_28
IO168NPB3 J5
IO168PPB3 K6 GPIOA_26
IO169NDB3 H7
IO169PDB3 J7 GPIOA_27
IO170NDB3 G1 IOS_P12
IO170PDB3 G2 IOS_P14
IO171NDB3 G4
IO171PDB3 G5
IO172NDB3 H6
GAC2/IO172PDB3 G6 PROC_TCK
IO173NDB3 F4
GAB2/IO173PDB3 E4 OSC_CLK
IO174NDB3 F5
GAA2/IO174PDB3 E5 GPIOA_31
A3PE600L-FG484M
U1B
BANK 1
IO60NDB1 E19 GPIOB_8
GBA2/IO60PDB1 E18 GPIOB_7
IO61NPB1 F18
GBB2/IO61PPB1 G17 GPIOB_9
IO62NDB1 H16
GBC2/IO62PDB1 H16 GPIOB_0
IO63NDB1 G19 GPIOB_2
IO63PDB1 F19 GPIOB_1
IO64NPB1 J17
IO64PPB1 H18
IO65NPB1 K17
IO65PPB1 J18
IO66NDB1 J19
IO66PDB1 H19
IO67NPB1 L18
IO67PPB1 H17
IO68NDB1 L22 DIFFC1N
IO68PDB1 L21 DIFFC1P
GCC0/IO69NPB1 L15
GCC1/IO69PPB1 K16
GCB0/IO70NPB1 L19
GCB1/IO70PPB1 L16
GCA0/IO71NPB1 L17
GCA1/IO71PPB1 M16
IO72NDB1 N19
GCA2/IO72PDB1 M19
IO73NPB1 N16 TPV2
GCB2/IO73PPB1 M15 TPV1
IO74NPB1 N18 DIFFA1N
GCC2/IO74PPB1 M17 DIFFA1P
IO75NDB1 K19 GPIOB_4
IO75PDB1 K18 GPIOB_3
IO76NDB1 K21 DIFFC2N
IO76PDB1 K22 DIFFC2P
IO77NDB1 L21
IO77PDB1 L20
IO78NPB1 L22
IO78PPB1 M21
IO79NPB1 N21
IO79PPB1 M20
IO80NPB1 N17
IO80PPB1 M18
IO81NDB1 P21
IO81PDB1 P21
IO82NDB1 T22
IO82PDB1 R22
IO83NDB1 U21
IO83PDB1 T21
IO84NDB1 R19 DIFFB1N
IO84PDB1 P19 DIFFB1P
IO85NDB1 P17 DIFFA2N
IO85PDB1 P18 DIFFA2P
GDC0/IO86NDB1 T18
GDC1/IO86PDB1 R18
GDB0/IO87NPB1 P16
GDB1/IO87PPB1 R17
GDA0/IO88NDB1 U19
GDA1/IO88PDB1 T19
A3PE600L-FG484M
U1C
BANK 2
GDA2/IO89RSB2 W17 SRBS0_N
GDB2/IO90RSB2 V16 SRBS1_N
GDC2/IO91RSB2 W15 SRBS2_N
IO92RSB2 T16 GPIOC_29
IO93RSB2 W16 SRBS3_N
IO94RSB2 R16 GPIOC_30
IO95RSB2 AB17 GPIOC_15
IO96RSB2 V15
IO97RSB2 U14 SRAM_OE_N
IO98RSB2 V14
IO99RSB2 T14 SRAM_CE_N
IO100RSB2 AB16 GPIOC_17
IO101RSB2 AA16 GPIOC_23
IO102RSB2 W14
IO103RSB2 T13
IO104RSB2 U13
IO105RSB2 V13
IO106RSB2 W13
IO107RSB2 U12
IO108RSB2 T12
IO109RSB2 V12
IO110RSB2 R12
IO111RSB2 W12
IO112RSB2 AB13 GPIOC_19
IO113RSB2 AB12 GPIOC_21
IO114RSB2 V11
IO115RSB2 W11
IO116RSB2 AB9 GPIOC_14
IO117RSB2 R11
IO118RSB2 W10
IO119RSB2 U11
IO120RSB2 T11
IO121RSB2 V10
IO122RSB2 AB8 GPIOC_16
IO123RSB2 W9
IO124RSB2 U10
IO125RSB2 W8
IO126RSB2 T10
IO127RSB2 V9
IO128RSB2 AB7 GPIOC_18
IO129RSB2 Y7 GPIOA_21
IO130RSB2 AB6 GPIOC_20
IO131RSB2 U9
IO132RSB2 V8
IO133RSB2 AA7 GPIOC_22
IO134RSB2 W7
IO135RSB2 AA6 GPIOC_24
IO136RSB2 U8 LED2
IO137RSB2 V5 LED3
IO138RSB2 U7 LED4
IO139RSB2 V6 LED5
IO140RSB2 T7
GEC2/IO141RSB2 V7 PORESET_N
FF/GEB2/IO142RSB2 V6 PA3_FF
GEA2/IO143RSB2 T9
A3PE600L-FG484M
FPGA Signal Lines
COMPLETE
Title
Actel A3PE600L-FG484M SADv3 - Flight Prototype
Size A
Number
Revision 1a
Date: 11/26/2012
Sheet 5of 6
File: C:\Users\..\Core_Sig.SchDoc
Drawn By: LT Lucas S. Parobek, USN


FGPA CORE
FGPA I/O BANK 0
FGPA I/O BANK 1
FGPA I/O BANK 2
FGPA I/O BANK 3
FPGA Power Lines
COMPLETE
A3PE600L-FG484M
Header 5X2
VCORE
VJTAG
VPUMP
V3P3
V2P5
GND
Title
Actel A3PE600L-FG484M SADv3 - Flight Prototype
Size
B
Number
Revision
1a
Date: 11/26/2012
Sheet 4of 6
File: C:\Users\.\Core Pwr.SchDoc
Drawn By: LT Lucas S. Parobek, USN

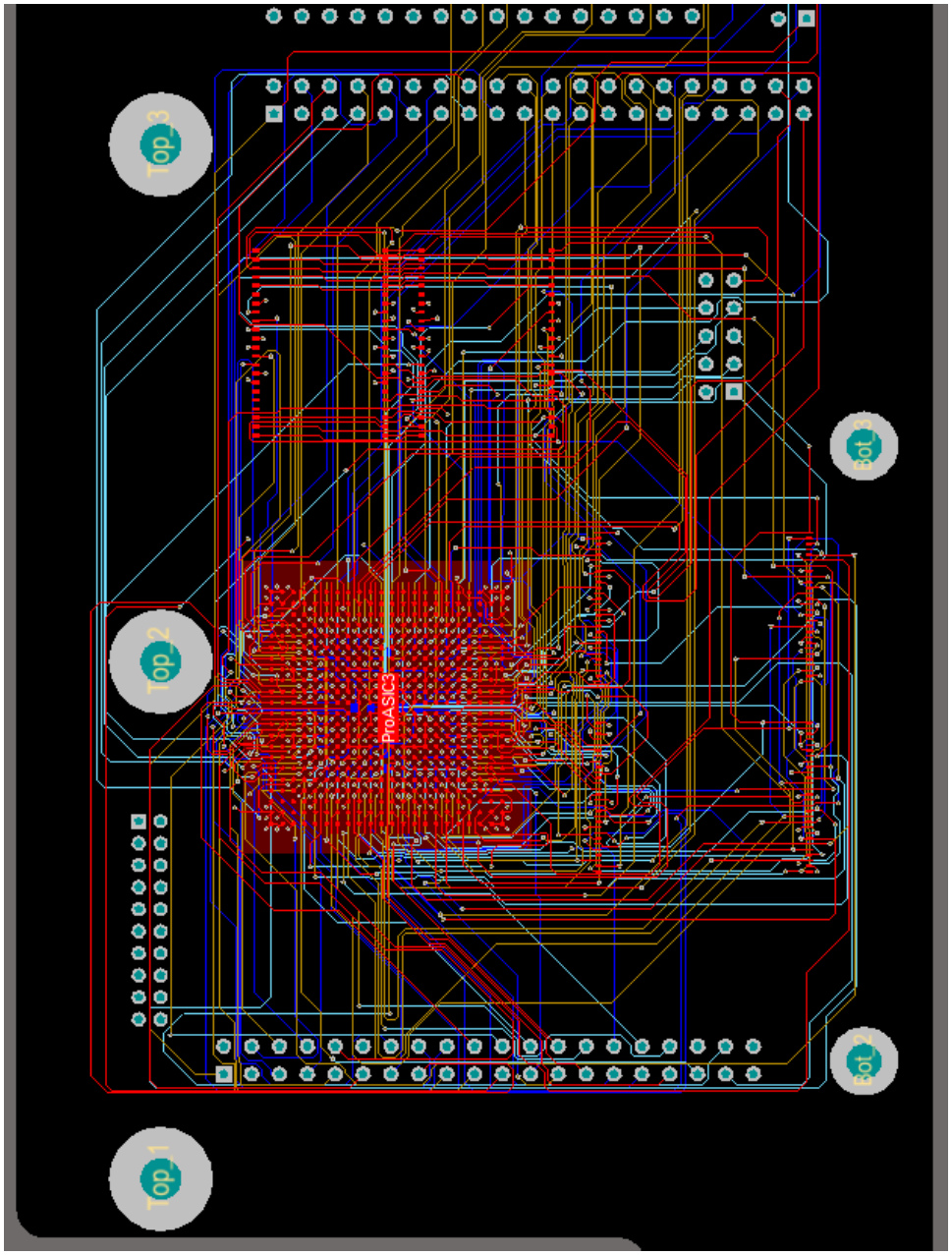

Digital Headers & LVDS Diff. Pairs
COMPLETE
GPIOB_4 R525 1% 165 GPIOBTX_4
R524 1% 140
GPIOB_3 R526 1% 165 GPIOBTX_3
GPIOB_10 R528 1% 165 GPIOBTX_10
R527 1% 140
GPIOB_9 R529 1% 165 GPIOBTX_9
GPIOB_2 R531 1% 165 GPIOBTX_2
R530 1% 140
GPIOB_1 R532 1% 165 GPIOBTX_1
GPIOB_8 R534 1% 165 GPIOBTX_8
R533 1% 140
GPIOB_7 R535 1% 165 GPIOBTX_7
GPIOB_13 R511 1% 165 GPIOBTX_13
R509 1% 140
GPIOB_14 R510 1% 165 GPIOBTX_14
GPIOB_15 R508 1% 165 GPIOBTX_15
R506 1% 140
GPIOB_16 R507 1% 165 GPIOBTX_16
GPIOB_19 R505 1% 165 GPIOBTX_19
R503 1% 140
GPIOB_20 R504 1% 165 GPIOBTX_20
GPIOB_21 R502 1% 165 GPIOBTX_21
R500 1% 140
GPIOB_22 R501 1% 165 GPIOBTX_22
GPIOB_26 R519 1% 165 GPIOBTX_26
R518 1% 140
GPIOB_25 R520 1% 165 GPIOBTX_25
DIFFA2P R522 1% 165 DIFFA2PTX
R521 1% 140
DIFFA2N R523 1% 165 DIFFA2NTX
DIFFB1N R516 1% 165 DIFFB1NTX
R515 1% 140
DIFFB1P R517 1% 165 DIFFB1PTX
GPIOB_28 R513 1% 165 GPIOBTX_28
R512 1% 140
GPIOB_27 R514 1% 165 GPIOBTX_27
P2
VIN
V3P3
GPIOA_1 GPIOA_3 GPIOA_5 GPIOA_7 GPIOA_9 GPIOA_11 GPIOA_13 GPIOA_15 GPIOA_17 GPIOA_19 GPIOA_21 GPIOA_23 GPIOA_25 GPIOA_27 GPIOA_29 GPIOA_31 DIFFA1P
GPIOA_0 GPIOA_2 GPIOA_4 GPIOA_6 GPIOA_8 GPIOA_10 GPIOA_12 GPIOA_14 GPIOA_16 GPIOA_18 GPIOA_20 GPIOA_22 GPIOA_24 GPIOA_26 GPIOA_28 GPIOA_30 DIFFA1N
Header 20X2
GND
P3
VIN
V2P5
GPIOBTX_1 GPIOBTX_3 GPIOBTX_5 GPIOBTX_7 GPIOBTX_9 GPIOBRX_11 GPIOBTX_13 GPIOBTX_15 GPIOBRX_17 GPIOBTX_19 GPIOBTX_21 GPIOBRX_23 GPIOBTX_25 GPIOBTX_27 GPIOBRX_29 GPIOB_31 DIFFB1PTX DIFFB2PRX
GPIOB_0 GPIOBTX_2 GPIOBTX_4 GPIOBRX_6 GPIOBTX_8 GPIOBTX_10 GPIOBRX_12 GPIOBTX_14 GPIOBTX_16 GPIOBRX_18 GPIOBTX_20 GPIOBTX_22 GPIOBRX_24 GPIOBTX_26 GPIOBTX_28 GPIOBRX_30 DIFFB1NTX DIFFB2NRX
Header 20X2
P4
VIN
V3P3
GPIOC_1 GPIOC_3 GPIOC_5 GPIOC_7 GPIOC_9 GPIOC_11 GPIOC_13 GPIOC_15 GPIOC_17 GPIOC_19 GPIOC_21 GPIOC_23 GPIOC_25 GPIOC_27 GPIOC_29 GPIOC_31 DIFFC1P DIFFC2P
GPIOC_0 GPIOC_2 GPIOC_4 GPIOC_6 GPIOC_8 GPIOC_10 GPIOC_12 GPIOC_14 GPIOC_16 GPIOC_18 GPIOC_20 GPIOC_22 GPIOC_24 GPIOC_26 GPIOC_28 GPIOC_30 DIFFC1N DIFFC2N
Header 20X2
GPIOBRX_5 R240 1% 100 GPIOBRX_6
GPIOBRX_11 R238 1% 100 GPIOBRX_12
GPIOBRX_17 R237 1% 100 GPIOBRX_18
GPIOBRX_23 R239 1% 100 GPIOBRX_24
GPIOBRX_29 R235 1% 100 GPIOBRX_30
DIFFB2PRX R236 1% 100 DIFFB2NRX
Title
Actel A3PE600L-FG484M SADv3 - Flight Prototype
Size B
Number
Revision 1a
Date: 11/26/2012
Sheet 3of 6
File: C:\Users\.\IOs.SchDoc
Drawn By: LT Lucas S. Parobek, USN

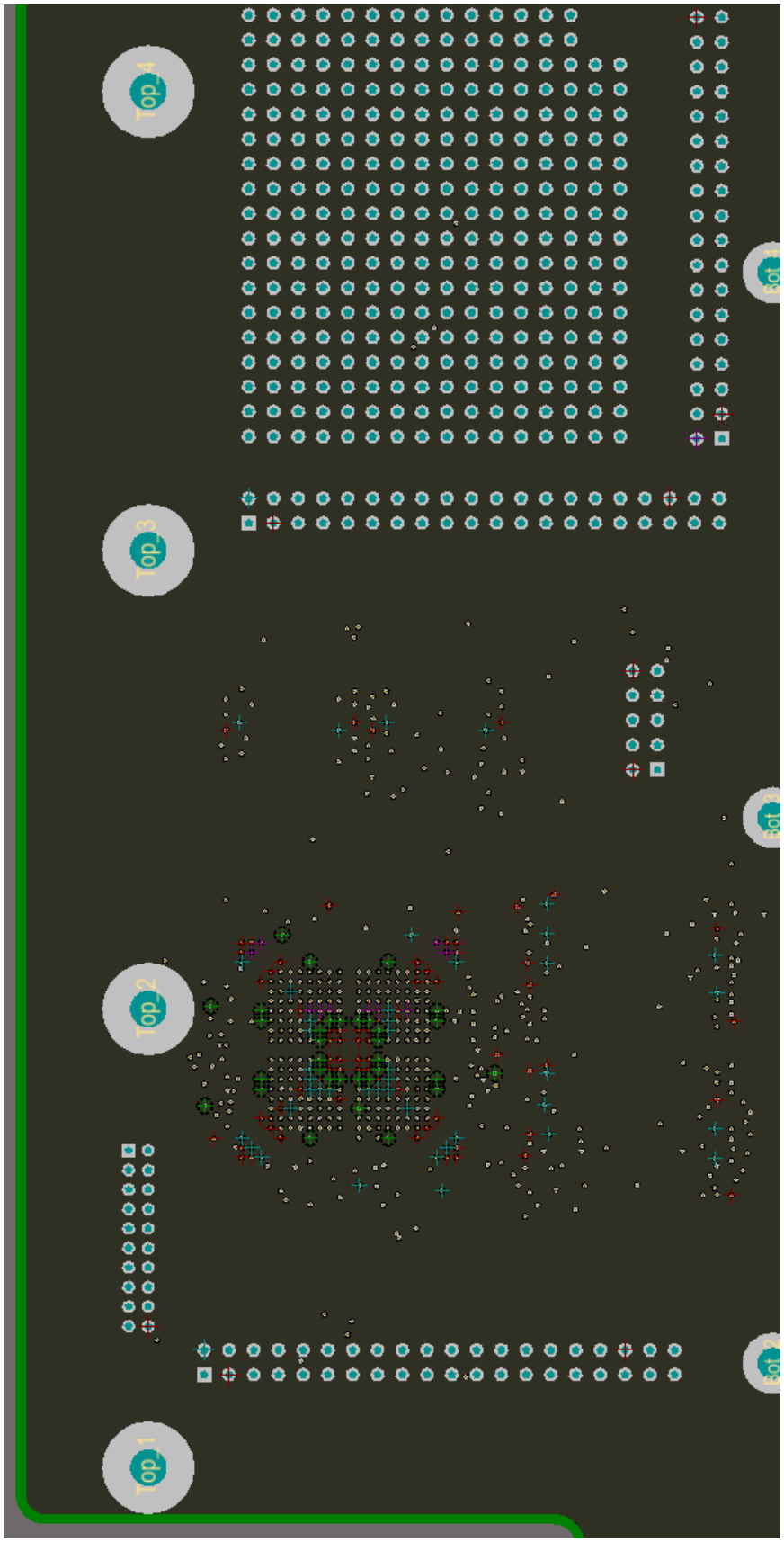

SRAM & Flash Memory
COMPLETE
U16
CY7C1041DV33
U18
CY7C1041DV33
U14
JS28F640J3D-75
U13
JS28F640J3D-75
V3P3
GND
MEM_ADDR2
MEM_ADDR3
MEM_ADDR4
MEM_ADDR5
MEM_ADDR6
MEM_ADDR7
MEM_ADDR8
MEM_ADDR9
MEM_ADDR10
MEM_ADDR11
MEM_ADDR12
MEM_ADDR13
MEM_ADDR14
MEM_ADDR15
MEM_ADDR16
MEM_ADDR17
MEM_ADDR18
MEM_ADDR19
MEM_ADDR20
MEM_ADDR21
MEM_ADDR22
MEM_ADDR23
MEM_ADDR24
MEM_ADDR25
MEM_DATA0
MEM_DATA1
MEM_DATA2
MEM_DATA3
MEM_DATA4
MEM_DATA5
MEM_DATA6
MEM_DATA7
MEM_DATA8
MEM_DATA9
MEM_DATA10
MEM_DATA11
MEM_DATA12
MEM_DATA13
MEM_DATA14
MEM_DATA15
SRAM_CE_N
SRAM_WE_N
SRBS0_N
SRBS1_N
SRAM_OE_N
SRBS2_N
SRBS3_N
FLASH_LCE_N
FLASH_RST_N
FLASH_OE_N
FLASH_WE_N
FLASH_HCE_N
C218 0.01uF
C219 0.01uF
C247 0.01uF
C250 0.1uF
C245 0.1uF
C246 10uF
C251 0.01uF
C248 0.01uF
C255 0.01uF
C252 0.1uF
C254 0.1uF
C253 1.0uF
C260 0.01uF
C259 0.01uF
C258 0.01uF
C240 0.1uF
C239 0.1uF
C243 10uF
C257 0.01uF
C256 0.01uF
C249 0.01uF
C238 0.1uF
C237 0.1uF
C242 1.0uF
R245 10K
R246 10K
R223 10K
Title
Actel A3PE600L-FG484M SADv3 - Flight Prototype
Size
B
Number
Revision
1a
Date: 11/26/2012
Sheet 2of 6
File: C:\Users\ \SRAM_Flash.SchDoc
Drawn By: LT Lucas S. Parobek, USN

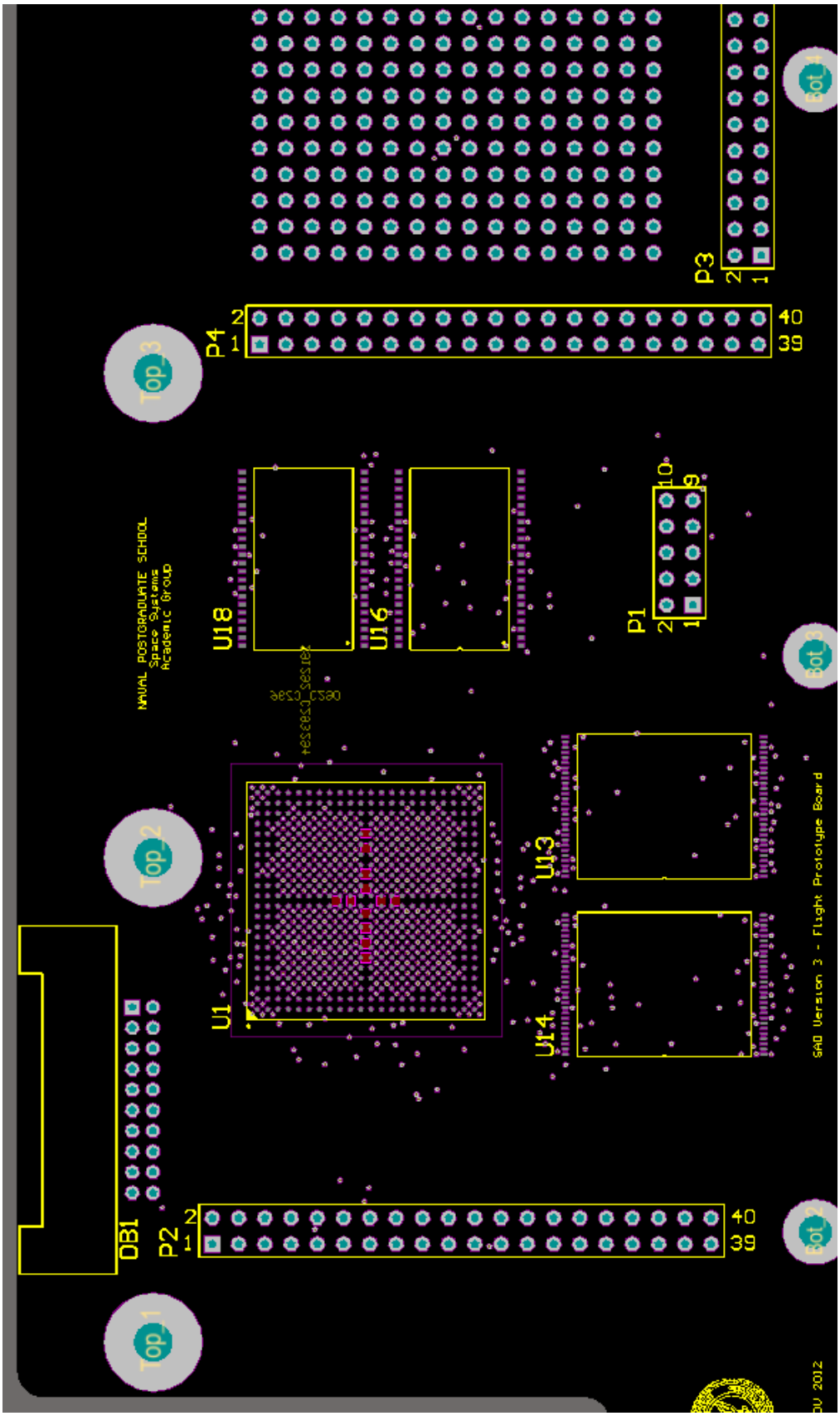

U22
TPS79601
2.5V
VIN
VIN
EN
GND
TABGND
VOUT
FB
C13
2.2uF
C12
15pF
R17
31.6k
1%
R18
30.1k
1%
C14
2.2uF
V2P5
GND
U21
TPS79601
VIN
VIN
EN
GND
TABGND
VOUT
FB
C9
2.2uF
C10
15pF
R16
51.1k
1%
R15
30.1k
1%
C11
2.2uF
V3P3
R14
220
1%
D6
LTST-C150GKT
GND
R216
2.2k
1.2V / 1.5V CORE
NC
D4
LM4041CEM3X-1.2
U12
TPS79601
VIN
VIN
EN
GND
TABGND
VOUT
FB
C227
2.2uF
C236
15pF
R217
30.1k
1%
VCORE
C230
2.2uF
V3P3
R222
10k
R221
121k
1%
R220
30.1k
1%
FDV301N
Q3
JP5
GND
FB2
FERRITE_BEAD
VIN
J3
C15
[TANT_107A]
100uF
5VDC
GND
GND
+5.0V DC IN
Power Supplies COMPLETE
Title
Actel A3PE600L-FG484M SADv3 - Flight Prototype
Size
A
Number
Revision
1a
Date: 11/26/2012
Sheet 1 of 6
File: C:\Users\..\Pwr_Supply.SchDoc
Drawn By: LT Lucas S. Parobek, USN

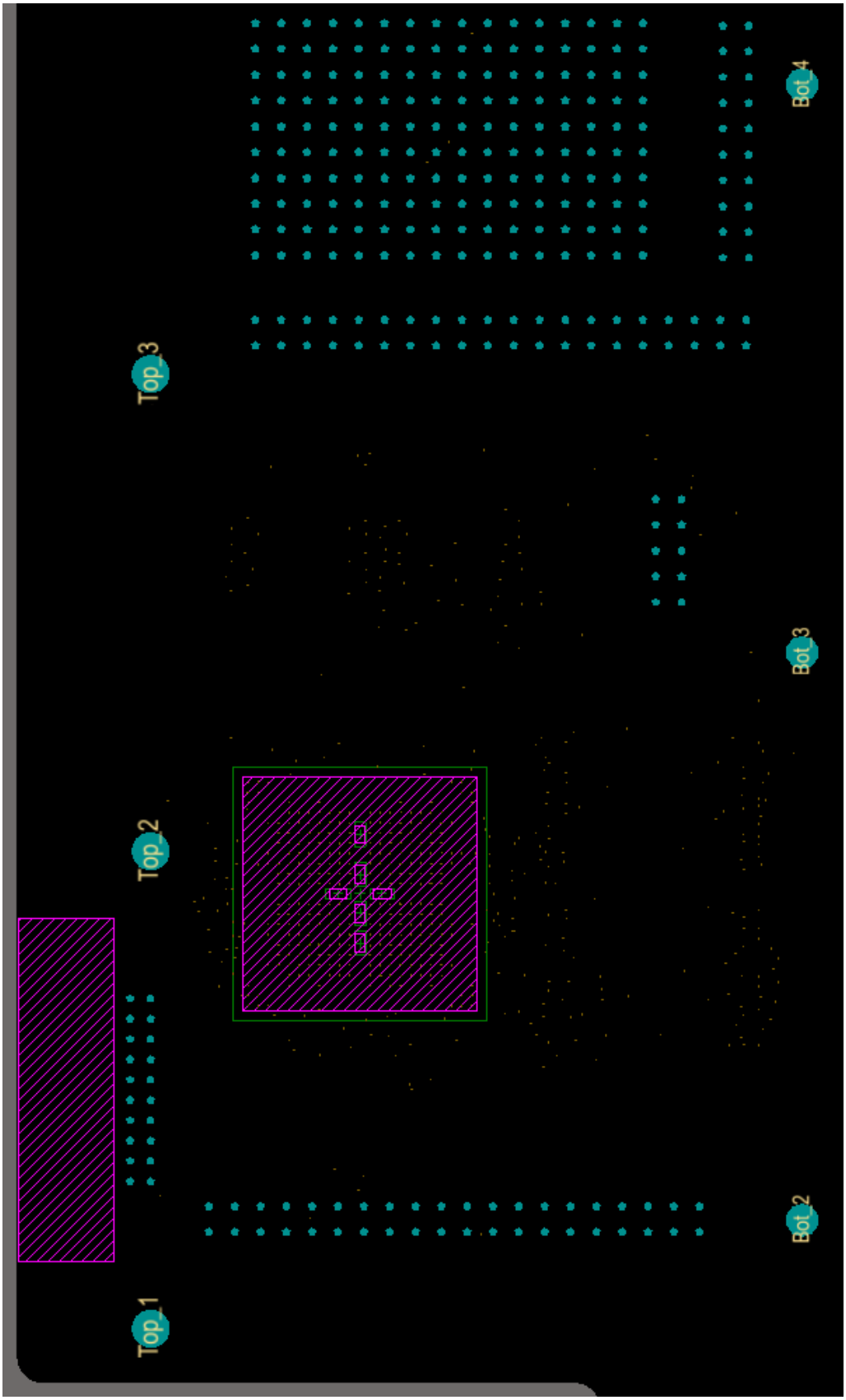
C.4 2-D PCB LAYER DIAGRAMS

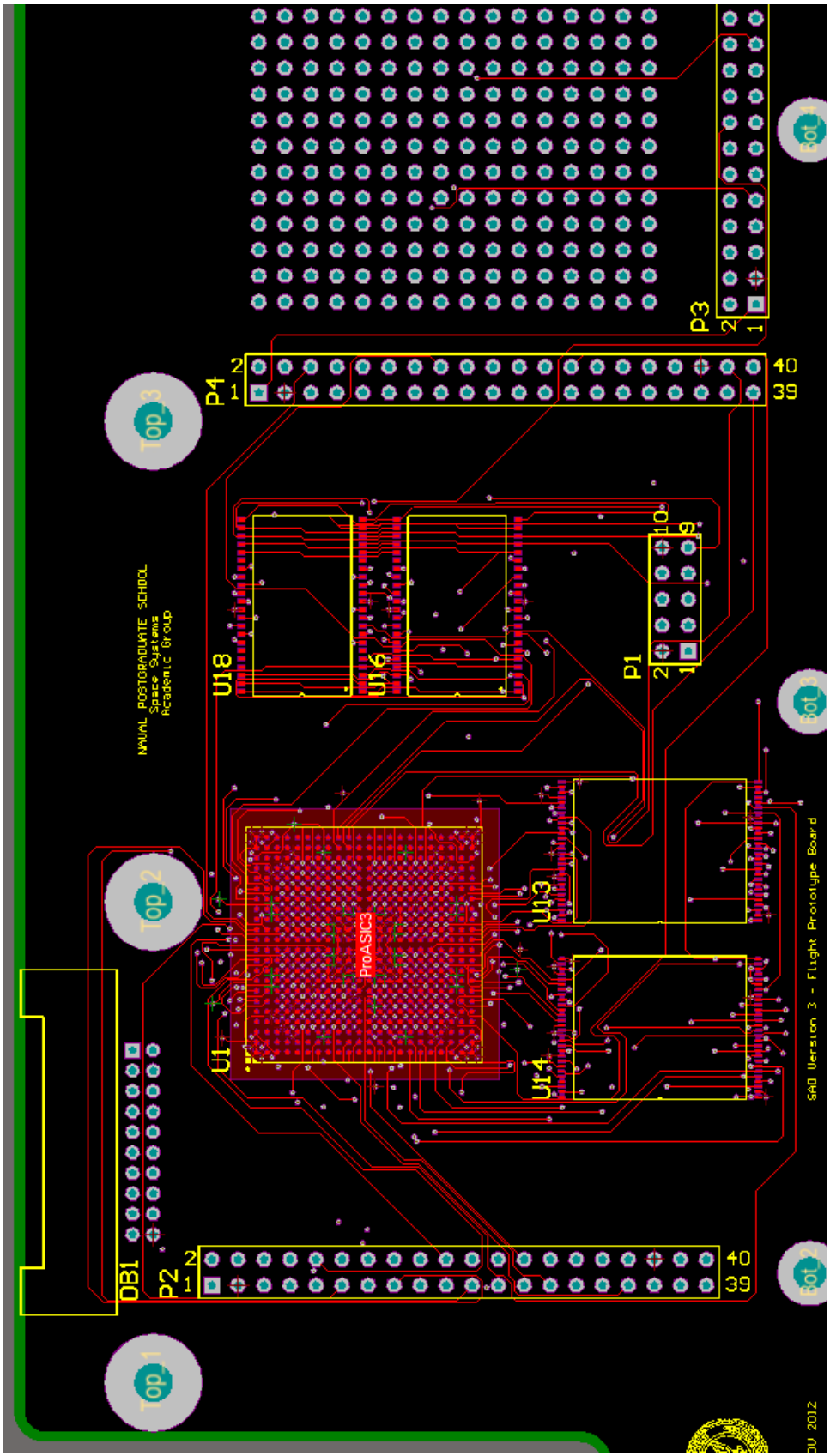
The following PCB layer diagrams illustrate the layout decisions in the PA3TB placement of components. The various diagrams included are combined signal layers, plane layer, non-signal layer, mechanical layer, top plane layers, and bottom plane layers. These layouts are available electronically for import into Altium, for reproduction or modification of the design as necessary.

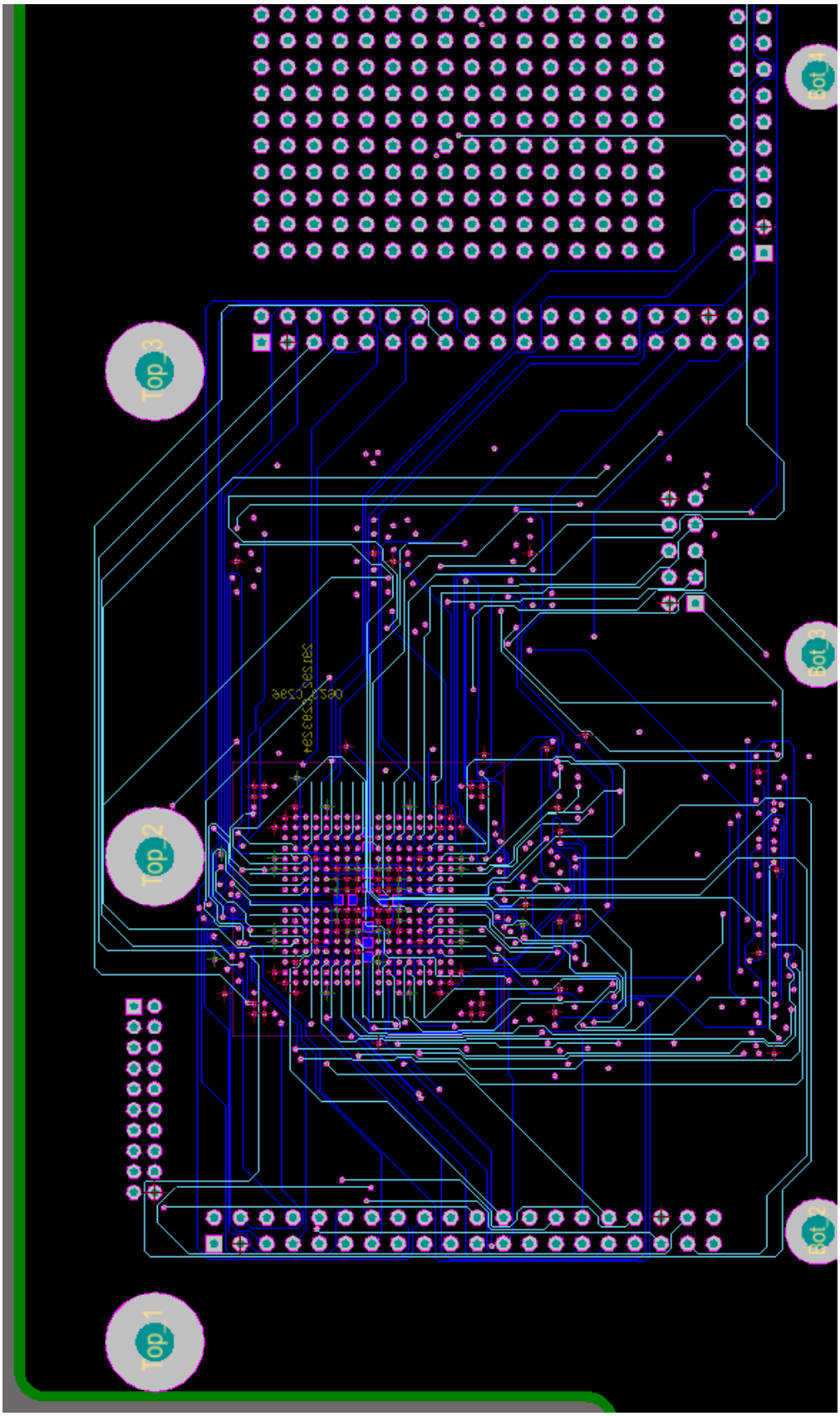



Top_1
Top_2
Top_3
Top_4
Bot
Bot_3
Bot_4


NAVAL POSTGRADUATE SCHOOL
Space Systems
Academic Group
U18
U16
U1
U13
U14
P1
P2
P3
P4
DB1
Top 1
Top 2
Top 3
Bot 1
Bot 2
Bot 3
GAD Version 3 - Flight Prototype Board

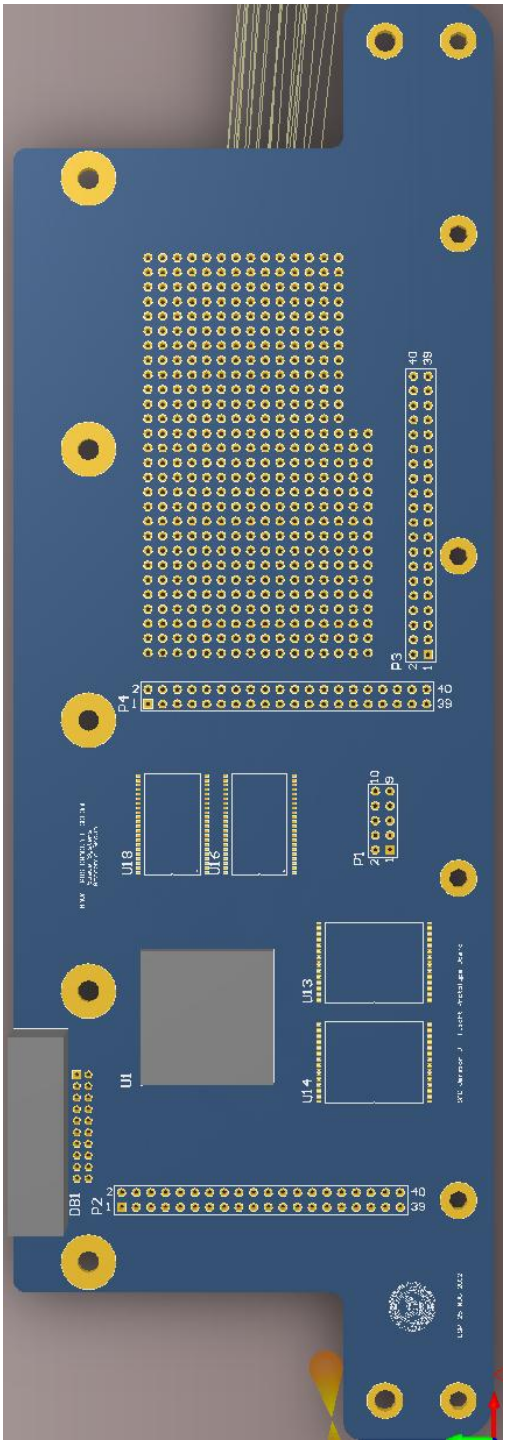

Top_1
Top_2
Top_3
Bot_2
Bot_3
Bot_4

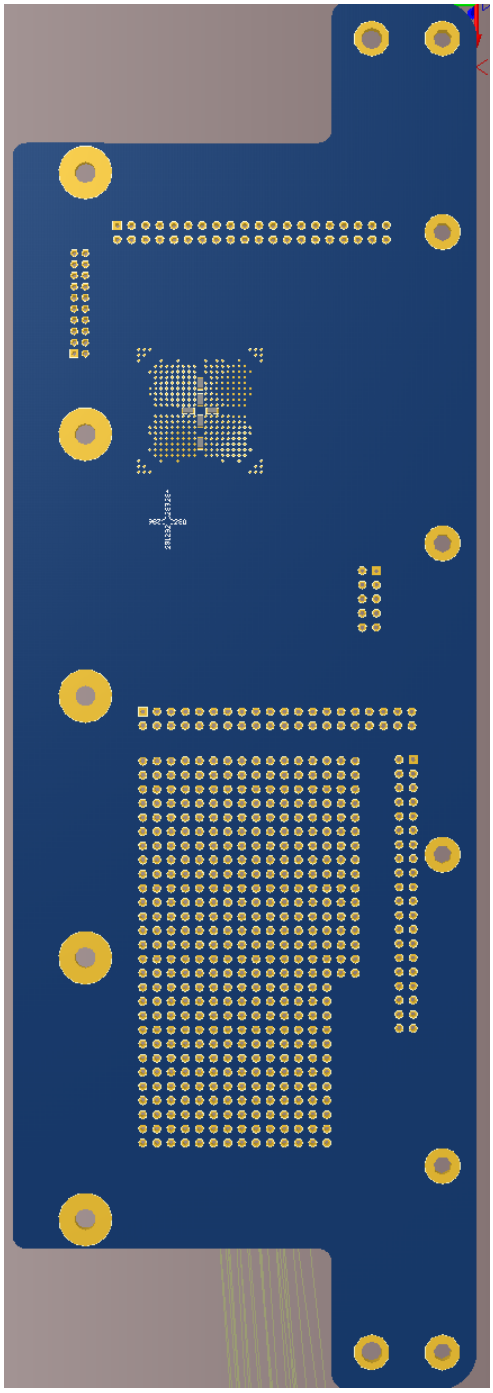

NAVAL POSTGRADUATE SCHOOL
Space Systems
Academic Group
U18
U16
P1
P4
P3
U1
ProASIC3
U13
U14
DB1
P2
SAO Version 3 - Flight Prototype Board
Top 1
Top 2
Top 3
Bot 1
Bot 2
Bot 3


Top_3
Top_2
Top_1
Bot_3
Bot_2
Bot_1

C.5 3-D PCB COMPONENT PLACEMENT

Top and bottom face 3-D views of the SADv3 are provided to depict component fit and placement upon the PCB. These layouts are available electronically for import into Altium for reproduction or modification of the design as necessary.





C.6 APPENDIX SUMMARY

In Appendix C, design decisions were presented concerning the manner in which they drove PCB component selection. Development of the design was presented from schematic to PCB layout with reasons presented for component selection and placement. Difficulties in schematic design with associated PCB layout complexities were presented.

APPENDIX D. SEQUENCER HDL PRESENTATION

The source code to generate the various bitstream files for testing of FSM and TMR logic was developed utilizing Verilog and VHDL. The code was synthesized using Xilinx ISE 14.3 and Libero SoC 10.1 and was executed on a Windows 7 PC with 8 GB of RAM and an Intel® i7™ 2600K CPU operating at 4.2 GHz. The code is designed for implementation onto the various Xilinx and Actel development kits and can be easily ported to the final SAD Version 3 - Flight Prototype Board.

All source code was formatted for presentation using Notepad++.

D.1.1 SEVEN-SEGMENT LED COUNTER (BASYS2USERDEMO.VHD)

The following code is a modified form of the top-level module for instantiation of a seven-segment LED counter with the clocking required:

```
--------------------------------------------------------------------------------
-- Company: Digilent RO
-- Engineer: Mircea Dabacan
-- Modified by: LT Lucas S. Parobek
--
-- Create Date:  23:51:25 03/22/2009
-- Modified on:  19:41:00 10/15/2012
-- Design Name:  Basys 2 User Demo
-- Module Name:  Basys2UserDemo - Structural
-- Project Name: Basys 2
-- Target Devices: Spartan 3E 100 (250)
-- Tool versions: ISE 10.1.03 / ISE 14.3
-- Description:  Overall top-level module for seven-segment LED counter.
--               Modification to occur to transfer to schematic representation.
--
-- The file contains the structural description of the Basys2 User Demo.
-- It combines the components:
-- - ckMux - to select between mclk and uclk
-- - SimpleSsegLedDemo - to test buttons, switches, LEDs and 7- segment disp.
--
-- Dependencies: ckMux.vhd, SimpleSsegLedDemo.vhd
--
-- Revision:
-- Revision 1.00 - File Modified for Use in TMR Design
-- Revision 0.01 - File Created
-- Additional Comments: All signal lines reproduced in schematic depiction...
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Basys2UserDemo is
```

```
   port (
          mclk     : in    std_logic;
          uclk     : in    std_logic;
          btn      : in    std_logic_vector (3 downto 0);
          sw       : in    std_logic_vector (7 downto 0);
          led      : out   std_logic_vector (7 downto 0);
          seg      : out   std_logic_vector (6 downto 0);
          an       : out   std_logic_vector (3 downto 0);
          dp       : out   std_logic
        );

end Basys2UserDemo;

architecture Structural of Basys2UserDemo is

   signal ck50MHz  : std_logic;

   component SimpleSsegLedDemo
      port ( ck  : in    std_logic;
             btn : in    std_logic_vector (3 downto 0);
             sw  : in    std_logic_vector (7 downto 0);
             led : out   std_logic_vector (7 downto 0);
             seg : out   std_logic_vector (6 downto 0);
             an  : out   std_logic_vector (3 downto 0);
             dp  : out   std_logic
            );
   end component;

   component ckMux
      port ( ck0   : in    std_logic;
             ck1   : in    std_logic;
             sel   : in    std_logic;
             ckOut : out   std_logic
            );
   end component;

begin

   SimpleSsegLedDemoInst : SimpleSsegLedDemo
      port map (
                ck=>mclk,
                btn(3 downto 0)=>btn(3 downto 0),
                sw(7 downto 0)=>sw(7 downto 0),
                led(7 downto 0)=>led(7 downto 0),
                seg(6 downto 0)=>seg(6 downto 0),
                an(3 downto 0)=>an(3 downto 0),
                dp=>dp
                );

   ckMuxInst : ckMux
      port map (
                ck0=>mclk,
                ck1=>uclk,
                sel=>sw(7),
                ckOut=>ck50MHz
                );

end Structural;
```

D.1.2 SEVEN-SEGMENT LED COUNTER (SIMPLESSEGLEDDEMO.VHD)

The following code is a the seven-segment LED module for instantiation of a seven-segment LED counter and corresponding switch and button inputs:

```
--------------------------------------------------------------------------------
-- Company: Digilent RO
-- Engineer: Mircea Dabacan
-- Modified by: LT Lucas S. Parobek
--
-- Create Date:  19:04:55 03/22/2009
-- Modified on:  19:45:00 10/15/2012
-- Design Name:  Basys 2 User Demo
-- Module Name:  SimpleSsegLedDemo - Behavioral
-- Project Name: Basys 2
-- Target Devices: Spartan 3E 100 (250)
-- Tool versions:  ISE 10.1.03 / ISE 14.3
-- Description:    Test buttons, switches, LEDs and seven segment disp.
--
-- This is the source file for the Simple Demo for Basys 2,
-- provided by the Digilent Reference Component Library.

-- The project demonstrates the behavior of:
--  - seven segment display: all digits count synchronously from 0 to F
--    hexadecimal. All decimal points are turned ON.
--  - buttons: pressing a button turns OFF the corresponding seven
--    segment display digit
--  - Switches and LEDs: switches control LEDs state
--
-- Dependencies: Basys2UserDemo.vhd (parent)
--
-- Revision:
-- Revision 1.00 - File Modified for Use in TMR Design
-- Revision 0.01 - File Created 20/03/2009(MirceaD)
-- Additional Comments: Modified for Use in TMR Design (+ Synth. Flags)
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity SimpleSsegLedDemo is

  Port (ck:  in  std_logic;
        btn: in  std_logic_vector(3 downto 0);
        sw:  in  std_logic_vector(7 downto 0);
        led: out std_logic_vector(7 downto 0);
        seg: out std_logic_vector(6 downto 0);
        dp:  out std_logic;
        an:  out std_logic_vector(3 downto 0)
          );

end SimpleSsegLedDemo;

architecture Behavioral of SimpleSsegLedDemo is

  signal cntDiv: std_logic_vector(28 downto 0); -- general clock div/cnt
  alias cntDisp: std_logic_vector(3 downto 0) is cntDiv(28 downto 25);
```

```
  -- four bits of the main counter

-- Synthesis Flags to Prevent Logic Trimming (LSP - 10/15/2012)
attribute equivalent_register_removal: string;
attribute equivalent_register_removal of cntDiv : signal is "no";

begin

  led <= sw;  -- switches control LEDs

  ckDivider: process(ck)
  begin
    if ck'event and ck='1' then
      cntDiv <= cntDiv + '1';
    end if;
  end process;

  --HEX-to-seven-segment decoder
--   HEX:   in    STD_LOGIC_VECTOR (3 downto 0);
--   LED:   out   STD_LOGIC_VECTOR (6 downto 0);
--
-- segment encoinputg
--      0
--     ---
--  5 |   | 1
--     ---   <- 6
--  4 |   | 2
--     ---
--      3

    with cntDisp SELect
   seg<= "1111001" when "0001",   --1
         "0100100" when "0010",   --2
         "0110000" when "0011",   --3
         "0011001" when "0100",   --4
         "0010010" when "0101",   --5
         "0000010" when "0110",   --6
         "1111000" when "0111",   --7
         "0000000" when "1000",   --8
         "0010000" when "1001",   --9
         "0001000" when "1010",   --A
         "0000011" when "1011",   --b
         "1000110" when "1100",   --C
         "0100001" when "1101",   --d
         "0000110" when "1110",   --E
         "0001110" when "1111",   --F
         "1000000" when others;   --0

  an <= btn;  -- released buttons turn corresponding digits ON
  dp <= '0';  -- all decimal point ON

end Behavioral;
```

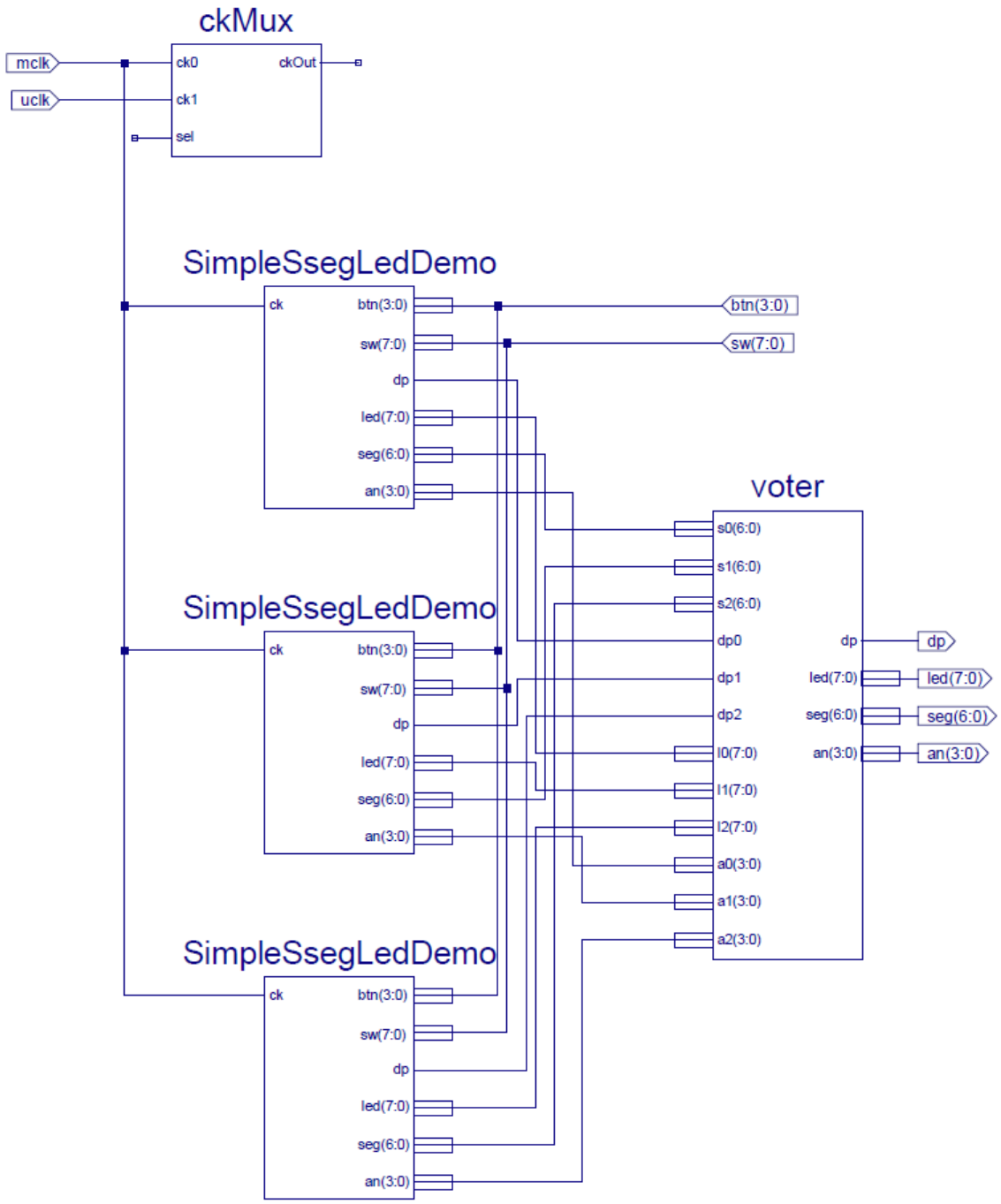
D.1.3 SEVEN-SEGMENT LED COUNTER (VOTER.V)

The following code is a the voter module for instantiation of a seven-segment LED counter in TMR and corresponding switch and button inputs:

```
////////////////////////////////////////////////////////////////////////////////
// Company: Naval Postgraduate School
// Engineer: LT Lucas S. Parobek
//
// Create Date:    19:48:32 10/23/2012
// Design Name:    Seven-Segment LED Counter - TMR Version
// Module Name:    voter.v
// Project Name:   Basys 2
// Target Devices: Spartan 3E 100 (250)
// Tool versions:  ISE 14.3
// Description:    Voter module to allow for TMR representation of counter.
//
// Dependencies:   Basys2UserDemo.vhd (parent)
//
// Revision:
// Revision 1.00 - File Modified for Use in TMR Design
// Additional Comments: All switch and button lines voted on to allow for
//                      continued operation as per non-TMR design…
////////////////////////////////////////////////////////////////////////////////
`timescale 1ns / 1ps
module voter(dp0,dp1,dp2,
                 l0,l1,l2,
                 s0,s1,s2,
                 a0,a1,a2,
                 dp,led,seg,an);

     input dp0;
     input dp1;
     input dp2;

     input [7:0] l0;
     input [7:0] l1;
     input [7:0] l2;

     input [6:0] s0;
     input [6:0] s1;
     input [6:0] s2;

     input [3:0] a0;
     input [3:0] a1;
     input [3:0] a2;

    output dp;
    output [7:0] led;
    output [6:0] seg;
    output [3:0] an;

assign dp = (dp0 & dp1) | (dp0 & dp2) | (dp1 & dp2);
assign led = (l0 & l1) | (l0 & l2) | (l1 & l2);
assign seg = (s0 & s1) | (s0 & s2) | (s1 & s2);
assign an = (a0 & a1) | (a0 & a2) | (a1 & a2);

endmodule
```

D.1.4 SEVEN-SEGMENT LED COUNTER (TMR SCHEMATIC)

The following schematic implements the seven-segment LED counter in TMR:



D.2 LAUNCH SEQUENCE GENERATOR (LAUNCH_SEQUENCER.VHD)

The following VHDL code provides the preliminary basis of the launch sequencer logic timing utilizing the FROM portion of the ProASIC3 to store timing constants:

```
-------------------------------------------------------------------------------
--                      Actel Corporation                                    --
--                            www.actel.com                                  --
-------------------------------------------------------------------------------
--        VHDL Design :  Launch Sequence Generator                           --
--         Design     :  ACTEL Launch Sequence using FlashROM IP             --
--         File name  :  launch_sequencer.vhd                                --
-------------------------------------------------------------------------------
-- Description: This module implements the logic to generate a sequence of    --
-- P-POD select signals in user defined time interval. Actels FlashROM design --
-- is configured and generated with different contents to meet this time      --
-- difference.                                                                --
--                                                                            --
-------------------------------------------------------------------------------
-- Modification History                                                       --
--                                                                            --
-- Initial revision                                                           --
--                                                                            --
-- Revision 1.0                                                               --
-------------------------------------------------------------------------------
--IMPORTANT-READ THESE TERMS CAREFULLY BEFORE UTILIZING THE INFORMATION       --
--CONTAINED IN THIS DESIGN.                                                   --
--                                                                            --
--NO SUPPORT/NO WARRANTY.                                                     --
--THE DESIGN FILES AND DOCUMENTATION ARE PROVIDED WITHOUT SUPPORT OR WARRANTY --
--OF ANY KIND.                                                                --
--ACTEL IS NOT OBLIGATED TO PROVIDE UPDATES, BUG FIXES, OR TECHNICAL SUPPORT  --
--AND DISCLAIMS ALL WARRANTIES INCLUDING, WITHOUT LIMITATION, THE WARRANTIES  --
--OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND WARRANTIES OF     --
--NON-INFRINGEMENT OF THE RIGHTS OF THIRD PARTIES (INCLUDING, WITHOUT         --
--LIMITATION, RIGHTS UNDER PATENT, COPYRIGHT, TRADE SECRET, OR OTHER          --
--INTELLECTUAL PROPERTY RIGHTS).  RECIPIENT ACCEPTS THE DESIGN FILES AND      --
--DOCUMENTATION IN "AS-IS" CONDITION.                                         --
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--                              L I B R A R Y                                --
-------------------------------------------------------------------------------
library IEEE;

use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_arith.all;
use IEEE.STD_LOGIC_unsigned.all;

library proasic3;
use proasic3.all;

-------------------------------------------------------------------------------
--                              E N T I T Y                                  --
-------------------------------------------------------------------------------
entity launch_sequencer is
port(
        --Input Clock
        clk        : in std_logic ;
        --Active low POWER ON RESET
        poreset_n     : in std_logic ;
        --First P-POD Select signal
        cs1        : out std_logic;
        --Second P-POD Select signal
        cs2        : out std_logic;
        --Third P-POD Select signal
```

```
        cs3        : out std_logic;
        --Fourth P-POD Select signal
        cs4        : out std_logic;
        --Fifth P-POD Select signal
        cs5        : out std_logic;
        --Sixth P-POD Select signal
        cs6        : out std_logic;
        --Seventh P-POD Select signal
        cs7        : out std_logic;
        --Eighth P-POD Select signal
        cs8        : out std_logic;
        --Reference signal for all P-POD select signal
        reference_out        : out std_logic

   );
end launch_sequencer;

-------------------------------------------------------------------------------
--                        A R C H I T E C T U R E                            --
-------------------------------------------------------------------------------
architecture launch_sequencer_arch of launch_sequencer is

-- FlashROM ip core
component FlashROM_cmp
    port
    (
     CLK  : in std_logic;
     ADDR : in std_logic_vector(6 downto 0);
     DOUT : out std_logic_vector(7 downto 0)
    ) ;
end component;

signal addr_sig : std_logic_vector(6 downto 0);
signal dout_sig : std_logic_vector(7 downto 0);

signal data1_sig : std_logic_vector(7 downto 0);
signal data2_sig : std_logic_vector(7 downto 0);
signal data3_sig : std_logic_vector(7 downto 0);
signal data4_sig : std_logic_vector(7 downto 0);
signal data5_sig : std_logic_vector(7 downto 0);
signal data6_sig : std_logic_vector(7 downto 0);
signal data7_sig : std_logic_vector(7 downto 0);
signal data8_sig : std_logic_vector(7 downto 0);

signal count1_sig : std_logic_vector(7 downto 0);
signal count2_sig : std_logic_vector(7 downto 0);
signal count3_sig : std_logic_vector(7 downto 0);
signal count4_sig : std_logic_vector(7 downto 0);
signal count5_sig : std_logic_vector(7 downto 0);
signal count6_sig : std_logic_vector(7 downto 0);
signal count7_sig : std_logic_vector(7 downto 0);
signal count8_sig : std_logic_vector(7 downto 0);

signal nrst_sig : std_logic ;

signal addr_en : std_logic;
signal addr_fprom_s :  std_logic_vector(6 downto 0);

component pll_4_40
    port(POWERDOWN, CLKA : in std_logic;  LOCK, GLA, GLB : out
        std_logic) ;
end component;

signal lock_sig : std_logic;
signal div_clk : std_logic ;

signal clk_divider : std_logic_vector(9 downto 0);
signal div_clkms : std_logic ;
```

```
signal div_clkms_sig : std_logic ;
signal load_en_sig : std_logic ;

COMPONENT CLKINT
  port(
                A          : in     STD_ULOGIC;
               Y                  : out     STD_ULOGIC);
END COMPONENT;

component addr_counter

    port( Aclr   : in    std_logic;
          Clock  : in    std_logic;
          Q      : out   std_logic_vector(6 downto 0);
          Enable : in    std_logic
        );

end component;

component data_counter

    port( Aclr   : in    std_logic;
          Clock  : in    std_logic;
          Q      : out   std_logic_vector(7 downto 0);
          Enable : in    std_logic
        );

end component;

signal clk_div_sig : std_logic;
signal clk_div2_sig : std_logic;
signal clk_div3_sig : std_logic;
signal clk_sig     : std_logic ;

signal one_sig : std_logic;
signal two_sig : std_logic;
signal three_sig : std_logic;
signal four_sig : std_logic;
signal five_sig : std_logic;
signal six_sig : std_logic;
signal seven_sig : std_logic;
signal eight_sig : std_logic;

type from_read is (idle_st, en_st, delay1_st, add_en_st) ;
signal pr_st, nx_st : from_read;

signal from_en_sig : std_logic;
signal add_en_sig : std_logic;

signal clk_from_sig : std_logic;
signal clk_40 : std_logic;
signal en_sig : std_logic;

signal den1_sig: std_logic;
signal cs1_sig : std_logic;
signal den2_sig : std_logic;
signal cs2_sig : std_logic;
signal den3_sig : std_logic;
signal cs3_sig : std_logic;
signal den4_sig : std_logic;
signal cs4_sig : std_logic;
signal den5_sig : std_logic;
signal cs5_sig : std_logic;
signal den6_sig : std_logic;
signal cs6_sig : std_logic;
signal den7_sig : std_logic;
signal cs7_sig : std_logic;
signal den8_sig : std_logic;
signal cs8_sig : std_logic;
```

```
begin

----------- Active low reset on board ---------------
nrst_sig <= poreset_n and lock_sig;


----------- Time Interval Generation ----------------
-----------------------------------------------------
--pll 4 40 inst: This PLL block generates 4Mhz and 40MHz clocks
--from external 48Mhz clock
-----------------------------------------------------
pll_4_40_inst: pll_4_40
      port map
      (
         POWERDOWN => '1',
         CLKA      => clk,
         LOCK  => lock_sig,
         GLA   => clk_div_sig,
         GLB   => clk_40
      ) ;

Process(clk_40, nrst_sig)
begin
     if nrst_sig = '0' then
       clk_from_sig <= '0' ;
     elsif clk_40'event and clk_40 = '1' then
       clk_from_sig <= not clk_from_sig ;
     end if ;
end process;

Process(clk_div_sig, nrst_sig, load_en_sig)
begin
     if nrst_sig = '0' then
       clk_div2_sig <= '0' ;
     elsif clk_div_sig'event and clk_div_sig = '1' then
      if load_en_sig = '1' then
       clk_div2_sig <= not clk_div2_sig ;
      end if;
     end if ;
end process;

Process(clk_div2_sig, nrst_sig)
begin
     if nrst_sig = '0' then
       clk_div3_sig <= '0' ;
     elsif clk_div2_sig'event and clk_div2_sig = '1' then
       clk_div3_sig <= not clk_div3_sig ;
     end if ;
end process;

--1Mhz clock
div_clk <= clk_div3_sig ;

Process(div_clk, nrst_sig, clk_divider)
begin
     if nrst_sig = '0' then
       clk_divider <= (others => '0') ;
       div_clkms_sig   <= '0' ;
     elsif div_clk'event and div_clk = '1' then
      if clk_divider = 0 then
           clk_divider <= "1111101000" ;
           div_clkms_sig   <= '1' ;
      else
           clk_divider <= clk_divider - '1' ;
           div_clkms_sig   <= '0' ;
      end if ;
     end if ;
end process ;
```

```
CLKINT_inst: CLKINT
    port map(
                A => div_clkms_sig,
                Y   => div_clkms
            );

----------- FlashROM ip core instantiation -----------
-----------------------------------------------------
--FlashROM_cmp_inst: This FlashROM block is configured
--to generate different time interval between P-PODs
--select signals
-----------------------------------------------------
FlashROM_cmp_inst: FlashROM_cmp port map
      (
         CLK  => from_en_sig,
         ADDR => addr_sig,
         DOUT => dout_sig
      ) ;

----------- Address Generation -----------------------

----------state machine for generating clock enable for FlashROM and address enable for
addr counter-------
process(clk_from_sig, nrst_sig)
begin
        if nrst_sig = '0' then
          pr_st <= idle_st ;
        elsif clk_from_sig'event and clk_from_sig = '1' then
          pr_st <= nx_st ;
        end if;
end process ;


process(pr_st)
begin
      case (pr_st) is

         when idle_st   =>  nx_st <=  en_st;
         when en_st     =>  nx_st <=  delay1_st;
         when delay1_st =>  nx_st <=  add_en_st;
         when add_en_st =>  nx_st <=  idle_st;

         when others =>  nx_st <=  idle_st;
      end case ;
end process ;

process(pr_st)
begin
      case (pr_st) is

         when idle_st   =>  from_en_sig <= '0' ; add_en_sig <= '0' ;
         when en_st     =>  from_en_sig <= '1' ; add_en_sig <= '0' ;
         when delay1_st =>  from_en_sig <= '0' ; add_en_sig <= '0' ;
         when add_en_st =>  from_en_sig <= '0' ; add_en_sig <= '1' ;

         when others    =>  from_en_sig <= '0' ; add_en_sig <= '0' ;

      end case ;
end process ;

en_sig <= add_en_sig and (not (load_en_sig)) ;

addr_counter_inst: addr_counter

    port map( Aclr   => nrst_sig,
              Clock  => clk_from_sig,
              Q      => addr_fprom_s(6 downto 0),
```

```
              Enable => en_sig
            );

ROMaddr_proc: Process(clk_from_sig, nrst_sig, addr_fprom_s, addr_en, add_en_sig)
begin
     if (nrst_sig = '0') then
       load_en_sig <= '0' ;
     elsif clk_from_sig'event and clk_from_sig = '1' then
        if addr_fprom_s = "0001001" then
           load_en_sig <= '1' ;
        end if;
     end if ;
end process ROMaddr_proc;

addr_sig <= addr_fprom_s(6 downto 0) ;

-----------Address Decoder ---------------
one_sig  <=  (not (addr_fprom_s(0)) ) and (not (addr_fprom_s(1)) ) and (not
(addr_fprom_s(2)) ) and (not (addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not
(addr_fprom_s(5)) ) and (not (addr_fprom_s(6)) ) ;

two_sig  <=  addr_fprom_s(0) and (not (addr_fprom_s(1)) ) and (not (addr_fprom_s(2)) )
and (not (addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not (addr_fprom_s(5)) )
and (not (addr_fprom_s(6)) ) ;

three_sig  <=  (not (addr_fprom_s(0)) ) and addr_fprom_s(1) and (not (addr_fprom_s(2)) )
and (not (addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not (addr_fprom_s(5)) )
and (not (addr_fprom_s(6)) ) ;

four_sig  <=  addr_fprom_s(0) and addr_fprom_s(1) and (not (addr_fprom_s(2)) ) and (not
(addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not (addr_fprom_s(5)) ) and (not
(addr_fprom_s(6)) ) ;

five_sig  <=  (not (addr_fprom_s(0)) ) and (not (addr_fprom_s(1)) ) and addr_fprom_s(2)
and (not (addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not (addr_fprom_s(5)) )
and (not (addr_fprom_s(6)) ) ;

six_sig  <=  addr_fprom_s(0) and (not (addr_fprom_s(1)) ) and addr_fprom_s(2) and (not
(addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not (addr_fprom_s(5)) ) and (not
(addr_fprom_s(6)) ) ;

seven_sig  <=  (not (addr_fprom_s(0)) ) and addr_fprom_s(1) and addr_fprom_s(2) and (not
(addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not (addr_fprom_s(5)) ) and (not
(addr_fprom_s(6)) ) ;

eight_sig  <= (not (from_en_sig)) and addr_fprom_s(0) and addr_fprom_s(1) and
addr_fprom_s(2) and (not (addr_fprom_s(3)) ) and (not (addr_fprom_s(4)) ) and (not
(addr_fprom_s(5)) ) and (not (addr_fprom_s(6)) ) ;

----------FROM content registers ---------------
Process(clk_from_sig, nrst_sig, add_en_sig, one_sig)
begin
     if nrst_sig = '0' then
       data1_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and one_sig = '1') then
           data1_sig <= dout_sig ;
      end if ;
     end if ;
end process;

Process(clk_from_sig, nrst_sig, add_en_sig, two_sig)
begin
     if nrst_sig = '0' then
       data2_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and two_sig = '1') then
           data2_sig <= dout_sig ;
      end if ;
```

```
     end if ;
end process;

Process(clk_from_sig, nrst_sig, add_en_sig, three_sig)
begin
     if nrst_sig = '0' then
       data3_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and three_sig = '1') then
           data3_sig <= dout_sig ;
      end if ;
     end if ;
end process;

Process(clk_from_sig, nrst_sig, add_en_sig, four_sig)
begin
     if nrst_sig = '0' then
       data4_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and four_sig = '1') then
           data4_sig <= dout_sig ;
      end if ;
     end if ;
end process;

Process(clk_from_sig, nrst_sig, add_en_sig, five_sig)
begin
     if nrst_sig = '0' then
       data5_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and five_sig = '1') then
           data5_sig <= dout_sig ;
      end if ;
     end if ;
end process;

Process(clk_from_sig, nrst_sig, add_en_sig, six_sig)
begin
     if nrst_sig = '0' then
       data6_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and six_sig = '1') then
           data6_sig <= dout_sig ;
      end if ;
     end if ;
end process;

Process(clk_from_sig, nrst_sig, add_en_sig, seven_sig)
begin
     if nrst_sig = '0' then
       data7_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and seven_sig = '1') then
           data7_sig <= dout_sig ;
      end if ;
     end if ;
end process;

Process(clk_from_sig, nrst_sig, add_en_sig, eight_sig)
begin
     if nrst_sig = '0' then
       data8_sig <= "11111111" ;
     elsif clk_from_sig'event and clk_from_sig = '0' then
      if (add_en_sig = '1' and eight_sig = '1') then
           data8_sig <= dout_sig ;
      end if ;
     end if ;
end process;
```

```
----------- CS Generation ---------------------------
den1_sig <= load_en_sig and ( not(cs1_sig) ) ;

data_counter1_inst: data_counter

    port map ( Aclr   => nrst_sig,
               Clock  => div_clk,
               Q      => count1_sig,
               Enable => den1_sig
             );

Process(div_clk, nrst_sig,  count1_sig)
begin
     if (nrst_sig = '0' ) then
       cs1_sig <= '0' ;
     elsif div_clk'event and div_clk = '1' then
      if count1_sig = data1_sig then
           cs1_sig <= '1' ;
      end if ;
     end if ;
end process;

den2_sig <= load_en_sig and ( not(cs2_sig) ) ;

data_counter2_inst: data_counter

    port map ( Aclr   => nrst_sig,
               Clock  => div_clk,
               Q      => count2_sig,
               Enable => den2_sig
             );

Process(div_clk, nrst_sig,  count2_sig)
begin
     if (nrst_sig = '0' ) then
       cs2_sig <= '0' ;
     elsif div_clk'event and div_clk = '1' then
      if count2_sig = data2_sig then
           cs2_sig <= '1' ;
      end if ;
     end if ;
end process;

den3_sig <= load_en_sig and ( not(cs3_sig) ) ;

data_counter3_inst: data_counter

    port map ( Aclr   => nrst_sig,
               Clock  => div_clk,
               Q      => count3_sig,
               Enable => den3_sig
             );

Process(div_clk, nrst_sig,  count3_sig)
begin
     if (nrst_sig = '0' ) then
       cs3_sig <= '0' ;
     elsif div_clk'event and div_clk = '1' then
      if count3_sig = data3_sig then
           cs3_sig <= '1' ;
      end if ;
     end if ;
end process;

den4_sig <= load_en_sig and ( not(cs4_sig) ) ;

data_counter4_inst: data_counter

    port map ( Aclr   => nrst_sig,
```

```
                Clock  => div_clk,
                Q      => count4_sig,
                Enable => den4_sig
              );

Process(div_clk, nrst_sig,  count4_sig)
begin
     if (nrst_sig = '0' ) then
       cs4_sig <= '0' ;
     elsif div_clk'event and div_clk = '1' then
      if count4_sig = data4_sig then
           cs4_sig <= '1' ;
      end if ;
     end if ;
end process;

den5_sig <= load_en_sig and ( not(cs5_sig) ) ;

data_counter5_inst: data_counter

    port map ( Aclr   => nrst_sig,
                Clock  => div_clkms,
                Q      => count5_sig,
                Enable => den5_sig
              );

Process(div_clkms, nrst_sig,  count5_sig )
begin
     if (nrst_sig = '0' ) then
       cs5_sig <= '0' ;
     elsif div_clkms'event and div_clkms = '1' then
      if count5_sig = data5_sig then
           cs5_sig <= '1' ;
      end if ;
     end if ;
end process;

den6_sig <= load_en_sig and ( not(cs6_sig) ) ;

data_counter6_inst: data_counter

    port map ( Aclr   => nrst_sig,
                Clock  => div_clkms,
                Q      => count6_sig,
                Enable => den6_sig
              );

Process(div_clkms, nrst_sig,  count6_sig)
begin
     if (nrst_sig = '0' ) then
       cs6_sig <= '0' ;
     elsif div_clkms'event and div_clkms = '1' then
      if count6_sig = data6_sig then
           cs6_sig <= '1' ;
      end if ;
     end if ;
end process;

den7_sig <= load_en_sig and ( not(cs7_sig) ) ;

data_counter7_inst: data_counter

    port map ( Aclr   => nrst_sig,
                Clock  => div_clkms,
                Q      => count7_sig,
                Enable => den7_sig
              );

Process(div_clkms, nrst_sig,  count7_sig)
```

```
begin
     if (nrst_sig = '0' ) then
       cs7_sig <= '0' ;
     elsif div_clkms'event and div_clkms = '1' then
      if count7_sig = data7_sig then
           cs7_sig <= '1' ;
      end if ;
     end if ;
end process;

den8_sig <= load_en_sig and ( not(cs8_sig) ) ;

data_counter8_inst: data_counter

    port map ( Aclr   => nrst_sig,
               Clock  => div_clkms,
               Q      => count8_sig,
               Enable => den8_sig
             );

Process(div_clkms, nrst_sig,  count8_sig)
begin
     if (nrst_sig = '0' ) then
       cs8_sig <= '0' ;
     elsif div_clkms'event and div_clkms = '1' then
      if count8_sig = data8_sig then
           cs8_sig <= '1' ;
      end if ;
     end if ;
end process;

-------------------------------------------------------------------------------
-- launch sequencer output signals                                           --
-------------------------------------------------------------------------------
Process(div_clk, nrst_sig)
begin
     if nrst_sig = '0' then
       reference_out   <= '0' ;
     elsif div_clk'event and div_clk = '1' then
           reference_out   <= '1' ;
     end if ;
end process ;

cs1 <= cs1_sig;
cs2 <= cs2_sig;
cs3 <= cs3_sig;
cs4 <= cs4_sig;
cs5 <= cs5_sig;
cs6 <= cs6_sig;
cs7 <= cs7_sig;
cs8 <= cs8_sig;

end launch_sequencer_arch ;
-------------------------------------------------------------------------------
--                                END OF FILE                                --
-------------------------------------------------------------------------------
```

LIST OF REFERENCES

[1] H. Helvajian, P. D. Fuqua, W. W. Hansen, and S. Janson, "Laser microprocessing for nanosatellite microthruster applications," *RIKEN Review*, no. 32, pp. 57–63, Jan. 2001.

[2] SPACE.com Staff, "Secret U.S. spy satellite launches into space after 6-week delay," *Space.com*, 13 September 2012. [Online] Available: http://www.space.com/17592-secret-spy-satellite-rocket-launch-nrol-36.html. [Accessed: 2 Nov. 2012]

[3] A. Chin, R. Coelho, R. Nugent, R. Munakata, and J. Puig-Suari, "CubeSat: the pico-satellite standard for research and education," presented at the AIAA Space Conference and Exposition, San Diego, CA, 2008.

[4] Planetary Systems Corp., "User's manual for Mark II Lightband," 2000785 datasheet, 9 Jul. 2012.

[5] Kaushish, V., "NPSCuL user's guide," Technical Report, Naval Postgraduate School, Monterey, CA, Aug. 2011.

[6] M. Niknahad, O. Sander, and J. Becker, "Fine grain fault tolerance - a key to high reliability for FPGAs in space," in *Proceedings, 2012 IEEE Aerospace Conference*, pp. 1–10, Big Sky, MT, 2012.

[7] M. Niknahad, O. Sander, and J. Becker, "QFDR-an integration of quadded logic for modern FPGAs to tolerate high radiation effect rates," presented at the 12th European Conference on Radiation and its Effects on Components and Systems (RADECS), Sevilla, Spain, 19–23 Sept. 2011.

[8] S. A. Johnson, "Implementation of a configurable fault tolerant processor (CFTP), M.S. thesis, Naval Postgraduate School, Monterey, CA, Mar. 2003.

[9] D. A. Ebert, "Design and development of a configurable fault-tolerant processor (CFTP) for space applications," M.S. thesis, Naval Postgraduate School, Monterey, CA, Jun. 2003.

[10] R. Yuan, "Triple modular redundancy (TMR) in a configurable fault-tolerant processor (CFTP) for space applications," M.S. thesis, Naval Postgraduate School, Monterey, CA, Dec. 2003.

[11] J. C. Coudeyras, "Radiation testing of the configurable fault tolerant processor (CFTP) for space-based applications," M.S. thesis, Naval Postgraduate School, Monterey, CA, Dec. 2005.

[12] P. J. Majewicz, "Implementation of a configurable fault tolerant processor (CFTP) using internal triple modular redundancy (TMR)," M.S. thesis, Naval Postgraduate School, Monterey, CA, Dec. 2005.

[13] G. W. Caldwell, "Implementation of configurable fault tolerant processor (CFTP) experiments," M.S. thesis, Naval Postgraduate School, Monterey, CA, Dec. 2006.

[14] G. M. Perez Casanova, "Implementation of a fault tolerant control unit within an FPGA for space applications," M.S. thesis, Naval Postgraduate School, Monterey, CA, Dec. 2006.

[15] D. Dwiggins, Jr., "Fault tolerant microcontroller for the configurable fault tolerant processor," M.S. thesis, Naval Postgraduate School, Monterey, CA, Sept. 2008.

[16] D. Sakoda, J. A. Horning, and S. D. Moseley, "Naval Postgraduate School NPSAT1 small satellite," presented at the ESA Small Satellite Systems and Services Symposium, Sardinia, Italy, 25–29 Sept. 2006.

[17] Texas Instruments, "Precision timers," NA555 datasheet, Sept. 1973 [Revised Jun. 2010].

[18] ARM Holdings, "Real-Time Operating Systems (RTOS)," *ARM.com*, 2012 [Online]. Available: http://www.arm.com/community/software-enablement/rtos-real-time-operating-system.php. [Accessed: 4 Nov. 2012]

[19] Microsemi Corp., "Radiation-tolerant ProASIC3 low power spaceflight flash FPGAs with Flash*Freeze technology," datasheet, Nov. 2009 [Revised Sept. 2012].

[20] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design – The Hardware / Software Interface*, 4th ed. Waltham, MA: Morgan Kaufmann, 2012.

[21] Actel Corp., "Single-event effects in FPGAs," Tech Report 55700011–2, Sunnyvale, CA, Oct. 2007.

[22] J. McHale, "Radiation-hardened electronic designers face increasing difficulty with shrinking chip geometries," in *Military & Aerospace Electronics*, 17 May 2011. [Online] Available: http://www.militaryaerospace.com/articles/2011/05/radiation-hardened-electronics.html. [Accessed: 9 Nov. 2012]

[23] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, Apr. 1962.

[24] F. H. Hardie and R. J. Suhocki, “Design and use of fault simulation for Saturn computer design,” *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 4, pp. 412–429, Aug. 1967.

[25] Xilinx Inc., “Considerations surrounding single-event effects in FPGAs, ASICs, and processors,” WP402 white paper, Sept. 2011.

[26] E. J. Daly, A. Hilgers, G. Drolshagen, and H. D. R. Evans, “Space environment analysis: Experience and trends,” in *Proceedings, ESA Symposium on Environment Modeling for Space-based Applications*, 18 Sept. 2006. [Online]Available: http://esamultimedia.esa.int/conferences /96a09/Abstracts/abstract45/. [Accessed: 11 Nov. 2012]

[27] A. D. Tipton, J. A. Pellish, R. A. Reed, R. D. Schrimpf, et al., “Multiple-bit upset in 130 nm CMOS technology,” in *IEEE Transactions on Nuclear Science*, vol. 53, pp. 3259–3264, 2006.

[28] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong, “Characterization of multi-bit soft error events in advanced SRAMs,” in *Proceedings, IEEE International Electronic Device Meeting*, pp. 519–522, Dec. 2003.

[29] S. Rezgui, J. J. Wang, E. C. Tung, B. Cronquist, and J. McCollum, “New methodologies for SET characterization and mitigation in flash-based FPGAs,” in *IEEE Transactions on Nuclear Science*, vol. 54, pp. 2512–2524, 2007.

[30] S. Buchner and D. McMorrow, “Single-event transients in bipolar linear integrated circuits,” in *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3079–3102, Dec. 2006.

[31] J. R. Schwank, M. R. Shaneyfelt, J. Baggio, P. E. Dodd, et al., “Effects of particle energy on proton-induced single-event Latchup,” in *IEEE Transactions on Nuclear Science*, Vvol. 52, no. 6, pp. 2622–2629, Dec. 2005.

[32] H. J. Barnaby, “Total-ionizing-dose effects in modern CMOS technologies,” in *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3103–3121, Dec. 2006.

[33] T. R. Oldham and F. B. McLean, “Total ionizing dose effects in MOS oxides and devices,” in *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, Jun. 2003.

[34] J. Tarrillo, J. R. Azambuja, F. L. Kastensmidt, E. C. Pereria Fonseca, et al., “Analyzing the effects of TID in an embedded system running in a flash-based FPGA,” in *IEEE Transactions on Nuclear Science*, vol. 58, no. 6, Dec. 2011.

[35] J. S. George, R. Koga, and M. A. McMahan, “Neutron soft errors in Xilinx FPGAs at Lawrence Berkeley National Laboratory,” presented at IEEE Radiation Effects Data Workshop, Tucson, AZ, 14–18 Jul. 2008.

[36] Microsemi Corp., "FPGA reliability and the sunspot cycle," Tech. Report 55900103–1, Aliso Viejo, CA, Sept. 2011.

[37] A. S. Keys, J. H. Adams, R. C. Darty, M. C. Patrick, et al., "Radiation Hardened Electronics for Space Environments (RHESE) project overview," presented at International Planetary Probes Workshop, Atlanta, GA, 23–27 Jun. 2008.

[38] K. C. Fox, "The electrical atmosphere: plasma is next NASA science target," *NASA.gov*, 18 Jul. 2012. [Online] Available: http://www.nasa.gov/mission_pages/rbsp/news/electric-atmosphere.html. [Accessed: 12 Nov. 2012]

[39] E. G. Stassinopoulos and J. P. Raymond, "The space radiation environment for electronics," in *Proceedings of the IEEE*, vol. 76, no. 11, pp. 1423–1442, Nov. 1988.

[40] D. K. Pradhan, *Fault-Tolerant Computer System Design*. Upper Saddle River, N.J.: Prentice Hall PTR, 1996.

[41] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, "FPGA partial reconfiguration via configuration scrubbing," in *International Conference on Field Programmable Logic and Applications*, pp. 99–104, 2009.

[42] C. A. Hulme, H. H. Loomis, A. A. Ross, and R. Yuan, "Configurable fault-tolerant processor (CFTP) for spacecraft onboard processing," in *Proceedings, 2004 Aerospace Conference*, vol. 4, pp. 2269–2276, Big Sky, MT, 6–13 Mar. 2004.

[43] M. Niknahad, O. Sander, and J. Becker, "Fine grain fault tolerance – A key to high reliability for FPGAs in space," in *Proceedings, 2012 IEEE Aerospace Conference*, pp. 1–10, Big Sky, MT, 2012.

[44] R. D. Do, "New tool for FPGA designers mitigates soft errors within synthesis," *DSP-FPGA.com*, 6 Dec. 2011. [Online] Available: http://dsp-fpga.com/articles/new-errors-within-synthesis/. [Accessed: 13 Nov. 2012]

[45] K. S. Morgan, D. L. McMurtrey, B. H. Pratt, and M. J. Wirthlin, "A comparison of TMR with alternative fault-tolerant design techniques for FPGAs," in *IEEE Transactions on Nuclear Science*, vol. 54, pp. 2065–2072, 2007.

[46] J. G. Tryon, "Quadded logic," in *Redundancy Techniques for Computing Systems*, Wilcox and Mann, eds., pp. 205–228, Washington, D.C., 1962.

[47] J. Han, J. Gao, P. Jonker, Y. Qi, and J. A. B. Fortes, "Toward hardware-redundant, fault-tolerant logic for nanoelectronics," *IEEE Design & Test of Computers*, vol. 22, pp. 328–339, 2005.

[48] M. Niknahad, O. Sander, and J. Becker, "QFDR – An integration of quadded logic for modern FPGAs to tolerate high radiation effect rates," presented at the 12th European Conference on Radiation and its Effects on Components and Systems (RADECS), 19–23 Sept. 2011.

[49] Xilinx Inc., "Xilinx TMRTool Product Brief," *Xilinx.com*, 2012 [Online].Available: http://www.xilinx.com/ise/optional_prod/tmrtool.htm. [Accessed: 12 Nov. 2012]

[50] Mentor Graphics Corp., "Precision Hi-Rel," 1028010 datasheet, May 2011.

[51] Xilinx Inc., "Field Programmable Gate Array (FPGA)," *Xilinx.com*, 2012 [Online]. Available: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm. [Accessed: 2 Nov. 2012]

[52] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 203–215, 2007.

[53] G. Leopold, "Updated: How Mars rover got its 'dream mode'," *EETimes.com*, 15 Aug. 2012. [Online] Available: http://www.eetimes.com/electronics-news/4394104/How-Mars-rover-got-its--dream-mode-. [Accessed: 4 Nov. 2012]

[54] National Instruments, "FPGA Fundamentals," *NI.com*, 3 May 2012. [Online]Available: http://www.ni.com/white-paper/6983/en. [Accessed: 6 Nov. 2012]

[55] Altera Corp., "Logic array blocks and adaptive logic modules in Stratix IV devices," SIV51002–3.1 datasheet, May 2008 [Revised Feb. 2011].

[56] C. Pfeil, "Fanout patterns, parts 1–4," in *Printed Circuit Design and Fabrication*, 1 May 2008. [Online] Available: http://pcdandf.com/cms/magazine/95/4632. [Accessed: 6 Nov. 2012]

[57] Altera Corp., "FPGA run-time reconfiguration: two approaches," Tech. Report WP-01055–1.0, San Jose, CA, Mar. 2008.

[58] E. L. Horta, J. W. Lockwood, D. E. Taylor, and D. Parlour, "Dynamic hardware plugins in an FPGA with partial run-time reconfiguration," in *Proceedings, 39th Design Automation Conference*, pp. 343–348, 2002.

[59] E. J. McDonald, "Runtime FPGA partial reconfiguration," presented at the 2008 IEEE Aerospace Conference, pp. 1–7, 2008.

[60] Computer History Museum, "1978 – PAL User-Programmable Logic Devices Introduced," *ComputerHistory.org*, 2007. [Online] Available: http://www.computerhistory.org/semiconductor/timeline/1978-PAL.html. [Accessed: 6 Nov. 2012]

[61] D. W. Page and L. R. Peterson, "Re-programmable PLA," U.S. Patent 4,508,977, 2 Apr. 1985.

[62] D. W. Page, "Dynamic data re-programmable PLA," U.S. Patent 4,524,430, 18 Jun. 1985.

[63] R. H. Freeman, "Configurable electrical circuit having configurable logic elements and configurable interconnects," U.S. Patent 4,870,302, 26 Sept. 1989.

[64] P. Alfke, I. Bolsens, B. Carter, M. Santarini, and S. Trimberger, "It's an FPGA!," in *IEEE Solid-State Circuits Magazine*, vol. 3, no. 4, pp. 15–20, 2011.

[65] Xilinx Inc., "7 series FPGAs overview," DS180 datasheet, Jun. 2010 [Revised Oct. 2012].

[66] J. C. Pomager, "Parametric reliability of space-based field programmable gate arrays," M.S. thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, Mar. 2007.

[67] Xilinx Inc., "Radiation-hardened, space-grade Virtex-5QV family overview," DS192 datasheet, Jul. 2010 [Revised Mar. 2012].

[68] Actel Corp., "Design security in nonvolatile flash and antifuse FPGAs," Tech. Report 5172163–1, Sunnyvale, CA, Aug. 2002.

[69] K. O'Neill, "Antifuse FPGA technology: best option for satellite applications," in *COTS Journal*, Dec. 2003. [Online] Available: http://www.cotsjournalonline.com/articles/print_article/100087. [Accessed: 7 Nov. 2012]

[70] Microsemi Corp., "RT ProASIC3 FPGAs," *Actel.com*, 2012. [Online]Available: http://www.actel.com/products/milaero/rtpa3/default.aspx. [Accessed: 7 Nov. 2012]

[71] L. Geppert, "The new indelible memories," in *IEEE Spectrum*, Mar. 2003. [Online] Available: http://spectrum.ieee.org/semiconductors/memory/the-new-indelible-memories/2. [Accessed: 7 Nov. 2012]

[72] 1-Core Technologies, "FPGA Logic Cells Comparison," *1-Core.com*, 2009 [Online]. Available: http://www.1-core.com/library/digital/fpga-logic-cells/. [Accessed: 16 Nov. 2012]

[73] Altium Limited, "Wiki – The Altium Designer Environment," *Altium.com*, Jun. 2012 [Online] . Available: http://wiki.altium.com/display/ADOH/The+Altium+Designer+Environment. [Accessed: 17 Nov. 2012]

[74] Microsemi Corp., "ProASIC3 nano flash FPGAs," 51700111-10 datasheet, Oct. 2008 [Revised Sept. 2012].

[75] Microsemi Corp., "ProASIC3L low power flash FPGAs with Flash*Freeze technology," 51700100-12 datasheet, Feb. 2008 [Revised Sept. 2012].

[76] J. F. Wakerly, *Digital Design – Principles and Practices, 2nd ed.*, Englewood Cliffs, N.J.: Prentice Hall, 1994.

[77] Actel Corp., "ProASIC3/E FlashROM (FROM)," 51900053-2 application note, Jan. 2005 [Revised Jun. 2005].

[78] Actel Corp., "Using IGLOO and ProASIC3 FPGAs as a system power sequencer design example," AC268 application note, Apr. 2009.

[79] Microsemi Corp., "ProASIC3 Series Overview," *Actel.com*, 2012 [Online]. Available: http://www.actel.com/products/pa3series/default.aspx. [Accessed: 16 Nov. 2012]

[80] VPT Inc., "SVSA2800D series – High reliability hybrid radiation tolerant DC-DC converters," SVSA2800D-2.0 datasheet, 2012.

[81] VPT Inc., "DVMA28 series – High reliability hybrid EMI filters," DVMA28-2.0 datasheet, 2012.

[82] Microsemi Corp., "Core 10100 v4.0 handbook," 50200077-6 datasheet, Feb. 2009.

[83] Microsemi Corp., "IP Cores," *Actel.com*, 2012. [Online] Available: http://www.actel.com/products/ip/default.aspx. [Accessed: 24 Nov. 2012]

[84] OpenCores, "Core Projects," *OpenCores.org*, 2012. [Online] Available: http://opencores.org/projects. [Accessed: 24 Nov. 2012]

[85] Microsemi Corp., "Display Reference Designs," *Actel.com*, 2012 [Online]. Available: http://www.actel.com/products/solutions/display/refdesign.aspx. [Accessed: 24 Nov. 2012]

[86] Xilinx Inc., "ISE WebPACK Design Software," *Xilinx.com*, 2012 [Online]. Available: http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm. [Accessed: 13 Nov. 2012]

[87] Microsemi Corp., "Microsemi FPGA and SoC Development Software," *Actel.com*, 2012 [Online]. Available: http://www.actel.com/products /software/libero/default.aspx. [Accessed: 13 Nov. 2012]

[88] Actel Corp., "SmartDesign v8.4 user's guide," datasheet, Jul. 2008.

[89] M. Surratt, H. H. Loomis, A. A. Ross, and R. Duren, "Challenges of remote FPGA configuration for space applications," in *Proceedings, 2005 IEEE Aerospace Conference*, Big Sky, MT, 2005.

[90] Xilinx Inc., "Virtex-5 FPGA user guide," UG190 datasheet, Apr. 2006 [Revised Mar. 2012].

[91] Digi-Key Corp., "XC5VFX130T-1FFG1738I," *DigiKey.com*, 2012 [Online]. Available: http://www.digikey.com/product-detail/en/XC5VFX130T-1FFG1738I/XC5VFX130T-1FFG1738I-ND/1957110. [Accessed: 16 Nov. 2012]

[92] Microsemi Corp., "RTSX-SU radiation-tolerant FPGAs (UMC)," 51700053-9 datasheet, Mar. 2006 [Revised Mar. 2012].

[93] Microsemi Corp., "ProASIC3L FPGA fabric user's guide," 502000261-4 datasheet, Jul. 2010 [Revised Sept. 2012].

[94] Microsemi Corp., "Total Ionizing Dose test report," Tech. Report 11T-RT3PE600L-CG484-QJA2H, Irvine, CA, Apr. 2011.

[95] Digi-Key Corp., "A3PE600-FG484," *DigiKey.com*, 2012. [Online]Available: http://www.digikey.com/product-detail/en/A3PE600-FG484/A3PE600-FG484-ND/2858762. [Accessed: 16 Nov. 2012]

[96] Microsemi Corp., "What is the nominal voltage for V_{CCPLF} in A3P030?," *Actel.com*, Jan. 2012. [Online] Available: http://www.actel.com/kb/article.aspx?id=FQ1206. [Accessed: 17 Nov. 2012]

[97] Molex, "Micro-Fit 3.0 connectors," 43045-0614 datasheet, Jun. 2012.

[98] Microsemi Corp., "FlashPro Programming System," *Actel.com*, 2012 [Online]. Available: http://www.actel.com/products/hardware /program_debug/flashpro/default.aspx. [Accessed: 17 Nov. 2012]

[99] Digi-Key Corp., "A33159-ND / 5103308-1," *Digi-Key.com*, 2012 [Online]. Available: http://www.digikey.com/product-search/en?KeyWords=A33159-ND. [Accessed: 17 Nov. 2012]

[100] Microsemi Corp., "FlashPro for software v10.1 user's guide," 5-02-9138-19 datasheet, Aug. 2012.

[101] Tyco Electronics, “Connector system for Universal Serial Bus (USB),” 7-1773442-0 datasheet, Mar. 2006.

[102] Fairchild Semiconductor, “Surface mount LED lamp – Standard bright 0603 (0.6 mm Height),” 300087 datasheet, Feb. 2003.

[103] Advanced Circuits, “FreeDFM File Check Service,” *My4PCB.com*, 2012 [Online]. Available: https://www.my4pcb.com/net35/FreeDFMNet/FreeDFMHome.aspx. [Accessed: 18 Nov. 2012]

[104] OSH Park, “Community PCB Order,” *OSHPark.com*, 2012. [Online]Available: http://oshpark.com/. [Accessed: 18 Nov. 2012]

[105] Billy Beecher, “Sequencer relay board logic, schematic, and PCB design,” unpublished lab notes and Altium design, 2012.

[106] Cypress Semiconductor Corp., “CY7C1041DV33 4-Mbit (256 K x 16) Static RAM,” 38-05473 datasheet, Jul. 2008 [Revised Jun. 2011].

[107] Numonyx, “Embedded flash memory (J3 v D) – 64-Mbit Monolithic,” 316577-06 datasheet, Dec. 2007.

[108] Abracon, “Half size DIP low voltage 3.3V HCMOS/TTL compatible crystal clock oscillator,” ACHL datasheet, Dec. 2010.

[109] Texas Instruments, “Ultralow-noise, high PSRR, fast, RF, 1A low-dropout linear regulators,” SLVS351N datasheet, Sept. 2002 [Revised Jan. 2011].

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor R. Clark Robertson
   Chair, Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Rudolph Panholzer
   Chair, Space Systems Academic Group
   Naval Postgraduate School
   Monterey, California

5. Professor Herschel H. Loomis, Jr.
   Naval Postgraduate School
   Monterey, California

6. Professor James H. Newman
   Naval Postgraduate School
   Monterey, California

7. LT Lucas S. Parobek, USN
   National Reconnaissance Office
   Chantilly, Virginia

8. CAPT Bruce Dickey, USN
   National Reconnaissance Office
   Chantilly, Virginia

9. CAPT Paul Stewart, USN
   U.S. Naval Research Laboratory
   Washington, DC

10. CDR Michael Porter, USN
    National Reconnaissance Office
    Chantilly, Virginia

11. Lt. Col. Steve Lach
National Reconnaissance Office
Chantilly, Virginia

12. Professor John W. Peeples
The Citadel
Charleston, South Carolina

13. Professor Robert J. Barsanti, Jr.
The Citadel
Charleston, South Carolina

14. Professor Mark H. McKinney
The Citadel
Charleston, South Carolina