IMPLEMENTING A GROUND PENETRATING RADAR USER INTERFACE IN SYSTEM-ON-CHIP TECHNOLOGY

E.F. Bauermeister

Dissertation presented for the degree of Master of Science in Applied Science

In the Department of Electrical Engineering

University of Cape Town

August 2005

Acknowledgements

I would like to take this opportunity to thank my supervisors, Prof Mike Inggs (UCT) and Dr Alan Langman (OpenFuel (Pty) Ltd), for their willingness to undertake this design project as well as their guidance, support and encouragement. Without their help I could not have completed this design project. I would also like to thank OpenFuel (Pty) Ltd for putting their resources at my disposal.

Declaration

I hereby:

(a) **grant the University free license to reproduce the above dissertation in whole or in part, for the purpose of research;**

(b) **declare that:**

(i) **the above dissertation is my own unaided work, both in conception and execution, and that apart from the normal guidance of my supervisor, I have received no assistance apart from that stated below;**

(ii) **except as stated below, neither the substance or any part of the dissertation has been submitted in the past, or is being, or is to be submitted for a degree in the University or any other University.**

(iii) **I am now presenting the dissertation for examination for the Degree of MSc(ApplSc).**

signature removed

28/8/2005

E.F. Bauermeister **Date**

Synopsis

Ground penetrating radar technology is used to provide a fast and accurate method for target location compare to other geophysical sensing techniques. An existing ground penetrating radar system developed by OpenFuel (Pty) Ltd is used in the detection and avoidance of obstacles for a sub-surface horizontal directional drill mechanism. This ground penetrating radar system could be implemented as a portable surface-based version of the system for geophysical applications. A factor limiting its implementation is a personal or a laptop computer required to execute the human-machine interface software package for the radar system. Thus, there exists a need to produce a radar user interface to replace the computer required by the current ground penetrating radar system, while maintaining the original functionality of the radar system.

The purpose of this design project was to develop a user interface for a ground penetrating radar system in hardware. The radar user interface had to allow for the autonomous operation of the ground penetrating radar system and the human-machine interface application software.

The objectives of the design project were to initially review existing ground penetrating radar systems, generate a system specification for the radar user interface and then identify a suitable implementation technology for the radar user interface. From this we would then proceed to design the hardware for the radar user interface and finally verify, test and integrate the radar user interface hardware.

The review of existing ground penetrating radar technology showed that integrating the human-machine interface and the radar electronics was the optimum design concept to implement. We generated the system specification for the radar user interface with this design concept and the current ground penetrating radar electronics system specification in mind. A survey of the current processor technology in the semiconductor industry showed us that system-on-chip processor technology would be the ideal solution for the radar user interface as it provided low-power consumption, increased operating speed, reduced size and complexity, lower manufacturing costs and increased system reliability of the end-user product. From a survey of the available system-on-chip processors we

selected the AMD Alchemy Au1100 MIPS32-Based processor as it had the lowest power consumption versus relative speed of all processors reviewed.

A concept design for the radar user interface was done based on the system specification and the system-on-chip processor selected. The hardware design of the radar user interface was done by initially selecting the required components, capturing the schematic diagrams based on the concept design and the components selected and then the printed circuit boards were designed from these schematic diagrams.

The required software and firmware to test the radar user interface was designed and implemented. The hardware for the radar user interface was verified and tested and then integrated with the software and firmware developed for testing. The test software and firmware were then verified and tested and found to be operating as required. The radar user interface was then tested with the test software and firmware and compared with the user requirements and system specification to verify that the objectives of the design project had been reached.

In conclusion, the purpose and objectives of the design project had been satisfied as a radar user interface for the ground penetrating radar system had been developed successfully.

Table of Contents

Glossary of Terms and Abbreviations xiii

List of Figures xvii

List of Tables xix

Chapter 1

Introduction 1

1.1 Purpose and Scope of the Project 1

1.2 Project Layout 2

1.3 Conclusions 4

Chapter 2

Project Background 5

2.1 Existing Ground Penetrating Radar System Architecture 5

2.1.1 System Overview 5

2.1.2 Human-Machine Interface 7

2.1.3 Communications Physical Layer 7

2.1.4 Communications and Radar Controller Module 7

2.1.5 Sampler and Controller Module 8

2.1.6 Transmit Synthesizer Module 8

2.1.7 Receive Demodulator 8

2.1.8 Intermediate Frequency Cancellation Module 8

2.1.9 Antenna Controller 9

2.1.10 Radar Module Link Interface 9

2.2 Commercial-of-the-Shelf (COTS) GPR Systems 9

2.3 Conclusions 12

Chapter 3

User Requirements and System Specification 13

3.1 Functional User Requirements 13

3.1.1 Electrical 13

3.1.2 Mechanical 14

3.1.3 Configuration and Programming 14

3.1.4 Communication 14

3.1.5 Functional 15

3.1.6 Environmental 15

3.2 Hardware User Requirements 15

3.3 Software and Firmware User Requirements 16

3.4 System Specification 16

3.4.1 Electrical 17

3.4.2 Mechanical 17

3.4.3 Configuration and Programming 17

3.4.4 Radar Module Link 17

3.4.5 HMI Serial Link 17

3.4.6 Debugging Serial Link 18

3.4.7 USB Interface 18

3.4.8 Ethernet Interface 18

3.4.9 Functional 18

3.4.10 Environmental 18

3.5 Conclusions 19

Chapter 4

Initial Analysis and Concept Design 21

4.1 Initial Analysis 21

4.1.1 Industry and Technology Overview 21

4.1.2 The SoC Revolution 22

4.1.3 SoC Processor Selection 22

4.2 Concept Design 26
4.2.1 Radar User Interface Module 26
4.2.2 Radar User Interface Motherboard 27
4.2.3 Radar User Interface Mechanical Concept Design 30
4.3 Conclusions 30

Chapter 5

Hardware Implementation 33
5.1 RUI Module Component Selection 33
5.1.1 SoC Processor 33
5.1.2 Non-Volatile Program Memory 33
5.1.3 SDRAM Volatile Program Memory 34
5.2 RUI Motherboard Component Selection 34
5.2.1 Field-Programmable Gate Array (FPGA) 34
5.2.2 10/100Mbps Ethernet Interface 35
5.2.3 RS-422 HMI Interface 35
5.2.4 RS-232 Debug Interface 35
5.2.5 Real Time Clock and Power-On Reset Circuit 36
5.2.6 Power Supply Units 36
5.3 RUI Module Schematic Design 39
5.3.1 SoC Processor Design 39
5.3.2 Flash Non-Volatile Program Memory Design 41
5.3.3 SDRAM Volatile Program Memory Design 42
5.3.4 RUI Motherboard Interface Design 43
5.4 RUI Motherboard Schematic Design 43
5.4.1 Field-Programmable Gate Array (FPGA) Design 44
5.4.2 10/100Mbps Ethernet Interface Design 46
5.4.3 RS-422 HMI Interface Design 46
5.4.4 RS-232 Debug Interface Design 47
5.4.5 Real Time Clock and Power-On Reset Circuit Design 47

5.4.6 Power Supply Units Design 47
5.4.7 USB Interface Design 52
5.4.8 LCD Interface Design 53
5.4.9 Radar Module Link Interface Design 53
5.4.10 Non-Volatile Data Memory Design 55
5.4.11 RUI Module Interface Design 55
5.5 RUI Module Printed Circuit Board Design 56
5.6 RUI Motherboard Printed Circuit Board Design 58
5.7 Conclusions 62

Chapter 6

Software and Firmware Implementation 67
6.1 FPGA Hardware Configuration Firmware 67
6.2 FPGA RML Interface Firmware 74
6.3 Bootloader Software 77
6.4 Radar Application Software 77
6.5 Conclusions 78

Chapter 7

Verification, Integration and Testing 79
7.1 Verification, Integration and Testing Procedure 79
7.2 Verification, Integration and Testing Procedure Results 80
7.2.1 General Hardware Verification and Testing 80
7.2.2 Power Supply Verification and Testing 80
7.2.3 Integration of the RUI Module with the RUI Motherboard ... 85
7.2.4 FPGA Verification and Testing 86
7.2.5 SoC Processor Verification and Testing 87
7.2.6 RUI Bootloader Verification and Testing 87
7.2.7 10/100Mbps Ethernet Interface Testing 88
7.2.8 RS-422 HMI Interface Testing 88

7.2.9 RS-232 Debug Interface Testing 89
7.2.10 Real Time Clock and Power-On Reset Circuit Testing 89
7.2.11 USB Interface Testing 89
7.2.12 LCD Interface Testing 89
7.2.13 Radar Module Link (RML) Interface Testing 89
7.2.14 Non-Volatile Data Memory Interface Testing 90
7.3 Verification, Integration and Testing Procedure Summary 90
7.4 Conclusions 92

Chapter 8

Conclusions and Further Studies 93
8.1 Conclusions 93
8.2 Further Studies 94

References 99

Appendix A

Schematic Diagrams and Parts Lists 103
A.1 RUI Module 103
A.2 RUI Motherboard 113
A.3 Miscellaneous Schematic Diagrams 130

Appendix B

Printed Circuit Board Layouts 133
B.1 RUI Module 133
B.2 RUI Motherboard 144

Appendix C

Software and Firmware Source Code Listings and Information 155
C.1 FPGA Hardware Configuration Firmware Listing 155

C.2 Bootloader Download Procedure 157
C.2.1 Hardware and Software Requirements 157
C.2.2 File Location Setup Procedure 158
C.2.3 RUI Hardware Setup Procedure 158
C.2.4 Macraigor Systems LLC OCD Commander Software Configuration Procedure 158
C.2.5 MicroMonitor Bootloader SDRAM Download 159
C.2.6 Terminal Emulator Software Configuration Procedure 161
C.2.7 MicroMonitor Bootloader FLASH Memory Programming . 161
C.3 Bootloader Configuration Macro 164

Appendix D

Building a MIPS32 Target GNU Cross-Platform Development Toolchain under Linux 167
D.1 List of Resources 167
D.2 Required Source Components 167
D.3 Toolchain Building Procedure 168

Glossary of Terms and Abbreviations

ANT	**A**ntenna Controller
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
BGA	**B**all **G**rid **A**rray
CTL	**C**ommunications and Radar **C**ontroller
COTS	**C**ommercial-**o**f-**t**he-**S**helf
CPLD	**C**omplex **P**rogrammable **L**ogic **D**evice - It is a programmable logic device that supports the implementation of moderately large digital logic circuits (< 20 000 equivalent gates) using AND or OR planes.
CPU	**C**entral **P**rocessing **U**nit
DC	**D**irect **C**urrent
DCM	**D**igital **C**lock **M**anager
DIP	**D**ual **I**nline **P**ackage
EMC	**E**lectro**m**agnetic **C**ompatibility
EMI	**E**lectro**m**agnetic **I**nterference
ESD	**E**lectro**s**tatic **D**ischarge
ESR	**E**ffective **S**eries **R**esistance
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray - It is a programmable logic device that supports the implementation of relatively large digital logic circuits (> 20 000 equivalent gates) using logic blocks.
FSF	**F**ree **S**oftware **F**oundation
FTP	**F**ile **T**ransfer **P**rotocol
GPR	**G**round **P**enetrating **R**adar - It is radar technology that is adapted for used in subsurface remote sensing and imaging applications. The radar can either be located on the surface of the ground or below the surface, i.e. subsurface.
GPS	**G**lobal **P**ositioning **S**ystem
HMI	**H**uman-**M**achine **I**nterface - It is the device through which a human

	operator interacts with a machine. A machine is defined to be any electronic, electrical or electromechanical system.
IEEE	Institute of Electrical and Electronic Engineers
I^2C	Inter-Integrated Circuit
IFC	Intermediate Frequency Cancellation Module
ISP	In-System Programming
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LVDS	Low Voltage Differential Signalling
MAC	Media Access Controller
Mbps	Megabits per second
MIPS	Microcomputer without Interlocking Pipeline Stages - A RISC-based CPU design that was developed in the mid-80s.
MMU	Memory Management Unit
PCB	Printed Circuit Board
PCMCIA	Personal Computer Memory Card International Association
PLL	Phase-Locked Loop
Radar	Radio Detection and Ranging - It is a system for detecting the existence, location and displacement of objects through the transmission and reception of radio waves.
RFD	Receive Demodulator
RISC	Reduced Instruction Set Computer
RML	Radar Module Link
RMS	Root-Mean-Square
RTC	Real-Time Clock
RUI	Radar User Interface
SD	Secure Digital
SDRAM	Synchronous Dynamic Random Access Memory
SLIP	Serial Line Internet Protocol

SMC	Sampler and Controller Module
SoC	System-on-Chip - It is a large scale electronic system, generally implemented in multiple separate semiconductor wafers, that is integrated into a single semiconductor wafer and produced as a single integrated circuit.
SRAM	Static Random Access Memory
TXS	Transmit Synthesizer Module
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
VHDL	Very High Speed Integrated Circuit (VHSIC) Hardware Description Language - IEEE standardised language, standardised under IEEE 1164 (1993), used for describing the structure and modelling the behaviour of complex digital circuits.

List of Figures

Figure 1 OpenFuel (Pty) Ltd Ground Penetrating Radar System Block Diagram ... 6
Figure 2 GSSI TerraSIRch CIR-3000 Ground Penetrating Radar System 11
Figure 3 RUI Module Concept Design 27
Figure 4 RUI Motherboard and Display Concept Design 28
Figure 5 Radar User Interface Mechanical Concept Design 30
Figure 6 RUI Module Design Block Diagram 40
Figure 7 RUI Motherboard Design Block Diagram 45
Figure 8 RUI Module Top and Bottom Unpopulated PCB View 59
Figure 9 RUI Module Top and Bottom Populated PCB View 60
Figure 10 RUI Motherboard Top and Bottom Unpopulated PCB View 63
Figure 11 RUI Motherboard Top and Bottom Populated PCB View 64
Figure 12 FPGA Hardware Configuration Firmware Block Diagram - Part 1 of 2 . 68
Figure 13 FPGA Hardware Configuration Firmware Block Diagram - Part 2 of 2 . 69
Figure 14 Integrated RUI Module and Motherboard 86
Figure 15 RUI Radar Electronics Backplane Concept Design 95
Figure 16 RUI Module Schematic Diagram - Sheet 1 of 8 104
Figure 17 RUI Module Schematic Diagram - Sheet 2 of 8 105
Figure 18 RUI Module Schematic Diagram - Sheet 3 of 8 106
Figure 19 RUI Module Schematic Diagram - Sheet 4 of 8 107
Figure 20 RUI Module Schematic Diagram - Sheet 5 of 8 108
Figure 21 RUI Module Schematic Diagram - Sheet 6 of 8 109
Figure 22 RUI Module Schematic Diagram - Sheet 7 of 8 110
Figure 23 RUI Module Schematic Diagram - Sheet 8 of 8 111
Figure 24 RUI Motherboard Schematic Diagram - Sheet 1 of 12 114
Figure 25 RUI Motherboard Schematic Diagram - Sheet 2 of 12 115
Figure 26 RUI Motherboard Schematic Diagram - Sheet 3 of 12 116
Figure 27 RUI Motherboard Schematic Diagram - Sheet 4 of 12 117

Figure 28 RUI Motherboard Schematic Diagram - Sheet 5 of 12 118
Figure 29 RUI Motherboard Schematic Diagram - Sheet 6 of 12 119
Figure 30 RUI Motherboard Schematic Diagram - Sheet 7 of 12 120
Figure 31 RUI Motherboard Schematic Diagram - Sheet 8 of 12 121
Figure 32 RUI Motherboard Schematic Diagram - Sheet 9 of 12 122
Figure 33 RUI Motherboard Schematic Diagram - Sheet 10 of 12 123
Figure 34 RUI Motherboard Schematic Diagram - Sheet 11 of 12 124
Figure 35 RUI Motherboard Schematic Diagram - Sheet 12 of 12 125
Figure 36 RUI RS-232 Cable 131
Figure 37 RUI Module PCB Layout - Top Overlay Layer 134
Figure 38 RUI Module PCB Layout - Top Signal Layer 135
Figure 39 RUI Module PCB Layout - Ground Plane 136
Figure 40 RUI Module PCB Layout - Internal Signal Layer 1 137
Figure 41 RUI Module PCB Layout - +1.22V Supply Plane 138
Figure 42 RUI Module PCB Layout - +2.5V Supply Plane 139
Figure 43 RUI Module PCB Layout - Internal Signal Layer 2 140
Figure 44 RUI Module PCB Layout - +3.3V Supply Plane 141
Figure 45 RUI Module PCB Layout - Bottom Signal Layer 142
Figure 46 RUI Module PCB Layout - Bottom Overlay Layer 143
Figure 47 RUI Motherboard PCB Layout - Top Overlay Layer 145
Figure 48 RUI Motherboard PCB Layout - Top Signal Layer 146
Figure 49 RUI Motherboard PCB Layout - Ground Plane 147
Figure 50 RUI Motherboard PCB Layout - Internal Signal Layer 1 148
Figure 51 RUI Motherboard PCB Layout - +1.22V/+1.5V Supply Plane 149
Figure 52 RUI Motherboard PCB Layout - +2.5V/+5V Supply Plane 150
Figure 53 RUI Motherboard PCB Layout - Internal Signal Layer 2 151
Figure 54 RUI Motherboard PCB Layout - +3.3V/V_{IN} Supply Plane 152
Figure 55 RUI Motherboard PCB Layout - Bottom Signal Layer 153
Figure 56 RUI Motherboard PCB Layout - Bottom Overlay Layer 154

List of Tables

Table 1 SoC Processor Architecture Selection Criteria 23
Table 2 SoC Processor Power Consumption Comparison 25
Table 3 Calculated Maximum Power Supply Current Consumption 37
Table 4 Linear vs. Switching Regulator Comparison 38
Table 5 Calculated Resistor Values for +1.22V and +1.5V Power Supplies 48
Table 6 Calculated Power Supply Maximum Output Current vs. Input Voltage ... 49
Table 7 Calculated Power Supply Catch Diode Current vs. Input Voltage 50
Table 8 Calculated Peak Power Supply Inductor Current vs. Input Voltage 51
Table 9 Calculated Power Supply Output Ripple Current vs. Input Voltage 52
Table 10 FPGA Hardware Configuration Firmware Signal Definitions 72
Table 11 FPGA RML Interface Firmware Signal Definitions 74
Table 12 Power Supply Full Load Capacity Test Parameters 81
Table 13 Measured RUI Power Supply Output and Ripple Voltages 82
Table 14 RUI Power Supply Ripple Factors 83
Table 15 RUI Power Supply Line Regulation 84
Table 16 RUI Power Supply Load Regulation 85
Table 17 Verification, Integration and Testing Procedure Summary 90
Table 18 RUI Module Parts List 112
Table 19 RUI Motherboard Parts List 126
Table 20 RUI RS-232 Cable Parts List 132
Table 21 Macraigor Systems LLC OCD Commander Software Configuration ... 159
Table 22 Terminal Emulator Software Configuration Parameters 161

Chapter 1

Introduction

Ground penetrating radar (GPR) technology provides a fast and accurate method for target location compare to other geophysical sensing techniques. The rapid acquisition of data is an essential feature of ground penetrating radar technology, since target information must be acquired and evaluated rapidly. OpenFuel (Pty) Ltd has developed a ground penetrating radar system used in the detection and avoidance of obstacles for a sub-surface horizontal directional drill mechanism. In theory, the same basic radar system could be implemented in a portable surface-based version for geophysical applications. The only factor that prevents the realisation of this implementation is the requirement for a personal or a laptop computer to execute the human-machine interface (HMI) software package for the radar system.

To this end, there exists a need to produce a radar user interface (RUI) to replace the computer required to execute the radar human-machine interface software package, which will allow the current ground penetrating radar system to be industrialised and made portable for a surface-based version of the radar system, while maintaining the original functionality of the radar system.

1.1 Purpose and Scope of the Project

The purpose of this design project is the development of a user interface for a ground penetrating radar system in hardware with suitable software and firmware. The radar user interface must allow for the autonomous operation of the ground penetrating radar system and the human-machine interface software package. The scope of the design project can be defined as follows:

- the review of the existing ground penetrating radar system,
- the review of the existing commercially available ground penetrating radar systems,
- the generation of the system specification for the radar user interface,

- the identification of a suitable technology for the implementation of the radar user interface hardware,
- the design and implementation of the radar user interface hardware,
- the design and implementation of the basic radar user interface software and firmware and
- the integration and testing of the radar user interface hardware, software and firmware.

1.2 Project Layout

The design project is divided into seven primary sections that are reflected in this thesis as chapters and a secondary section containing appendices. Each chapter deals with a specific section pertaining to the overall scope of the project. The following section briefly describes each chapter and appendix in order to give the reader an overview of the project layout.

- **Chapter 2: Project Background** describes the background to the design project as far as previous work on which the design project is based goes. This section also describes the basic layout and operation of the current ground penetrating radar system with which the radar user interface must be integrated. This information is necessary for the generation of the user requirements and system specification. A study of commercial ground penetrating radar products is also presented and discussed. The information gained from this study was also used in the generation of the system specification.
- **Chapter 3: User Requirements and System Specification** deals with the requirements of the project as defined by the user and the subsequent system specification that was generated from the user requirements. An accurate system specification is vital in order to ensure that the user requirements are met and the design project is successful. The system specification is used in the initial analysis and concept design phase.
- **Chapter 4: Initial Analysis and Concept Design** discusses various technology

related issues that had to be investigated before the hardware, software and firmware implementation phases of the project could be undertaken. This section looks specifically at System-on-Chip (SoC) technology for implementation in the radar user interface. This section also describes the concept design of the radar user interface that was produced based on the initial analysis of the technology related issues and the system specification. This concept design shows the radar user interface hardware implemented as two separate modules to allow for future expansion and upgrade of the radar user interface.

- **Chapter 5: Hardware Implementation** describes the design and implementation of the hardware required for the radar user interface based on the initial analysis and concept design that was done. The selection of specific hardware devices is discussed and a detailed schematic hardware design is implemented. This schematic design is taken to the next step of implementing it on a printed circuit board. Issues regarding high speed printed circuit board design is also presented and discussed.
- **Chapter 6: Software and Firmware Implementation** describes the design and implementation of the software and firmware required for the radar user interface based on the hardware design and implementation. Specifically the firmware necessary for linking the hardware signals in the design and the firmware for the radar module link (RML) is discussed. The bootloader software used for the radar user interface and its use is also presented and discussed.
- **Chapter 7: Verification, Integration and Testing** describe the integration and testing of the radar user interface hardware, software and firmware. The procedure for the verification and integration of the hardware design is presented followed by the firmware. Finally the bootloader software and some of the hardware of the RUI are tested.
- **Chapter 8: Conclusions and Further Studies** discuss the significance and end result of this design project. It also discusses further work that can be done based on this design project. Specifically the expansion and enhancement of the hardware implementation of this design project are discussed.

- **Appendix A: Schematic Diagrams and Parts Lists** contains all the schematic diagrams and parts lists of the design project for reference purposes.
- **Appendix B: Printed Circuit Board Layouts** contains all the printed circuit board layouts of the design project for reference purposes.
- **Appendix C: Software and Firmware Source Code Listings and Information** contains the most important parts of the source code, both software and firmware, of the design project for reference purposes as well as information pertinent to the use of the software and firmware.
- **Appendix D: Building a GNU Cross-Platform Development Toolchain for a MIPS Target under Linux** describes the procedure that was developed during this design project for building the required GNU cross-platform development toolchain for the SoC processor under Linux.

1.3 Conclusions

In this chapter we had a look at a basic introduction to the design project. We examined the primary purpose for undertaking the design project as well as the scope of the work that was done. We also had a look at the layout of the thesis which serves as a macroscopic overview of the design project as well as being useful for indexing specific information or concepts.

Chapter 2

Project Background

This chapter provides background to the design project as far as previous work on which the design project is based goes. The basic layout and operation of the current ground penetrating radar system with which the radar user interface must be integrated are discussed. Commercial ground penetrating radar products were studied and the results are also discussed.

2.1 Existing Ground Penetrating Radar System Architecture

2.1.1 System Overview

A ground penetrating radar system was developed by OpenFuel (Pty) Ltd for the use in the detection and avoidance of obstacles for a sub-surface horizontal directional drill mechanism. The basic architectural layout of the radar system is shown in the block diagram in **Figure 1**. It is outside of the scope of this document to provide detailed information to the specific application of ground penetrating radar technology as was used in that project, but a brief overview of the operation of this specific system will be given as it has direct bearing on the design of the radar user interface.

The radar system consists of a human-machine interface (HMI) consisting of a personal or laptop computer running the radar application software, which is written in Java to make it portable and hardware independent. The HMI interfaces to the radar electronics through the communications physical layer. The radar electronics is modular in design, but is integrated in a single unit which is located on the sub-surface drill mechanism. The sub-surface radar electronics consists of the communications and radar controller (CTL) module, sampler and controller (SMC) module, transmit synthesizer (TXS) module, receive demodulator (RFD), intermediate frequency cancellation (IFC) module and antenna controller (ANT).

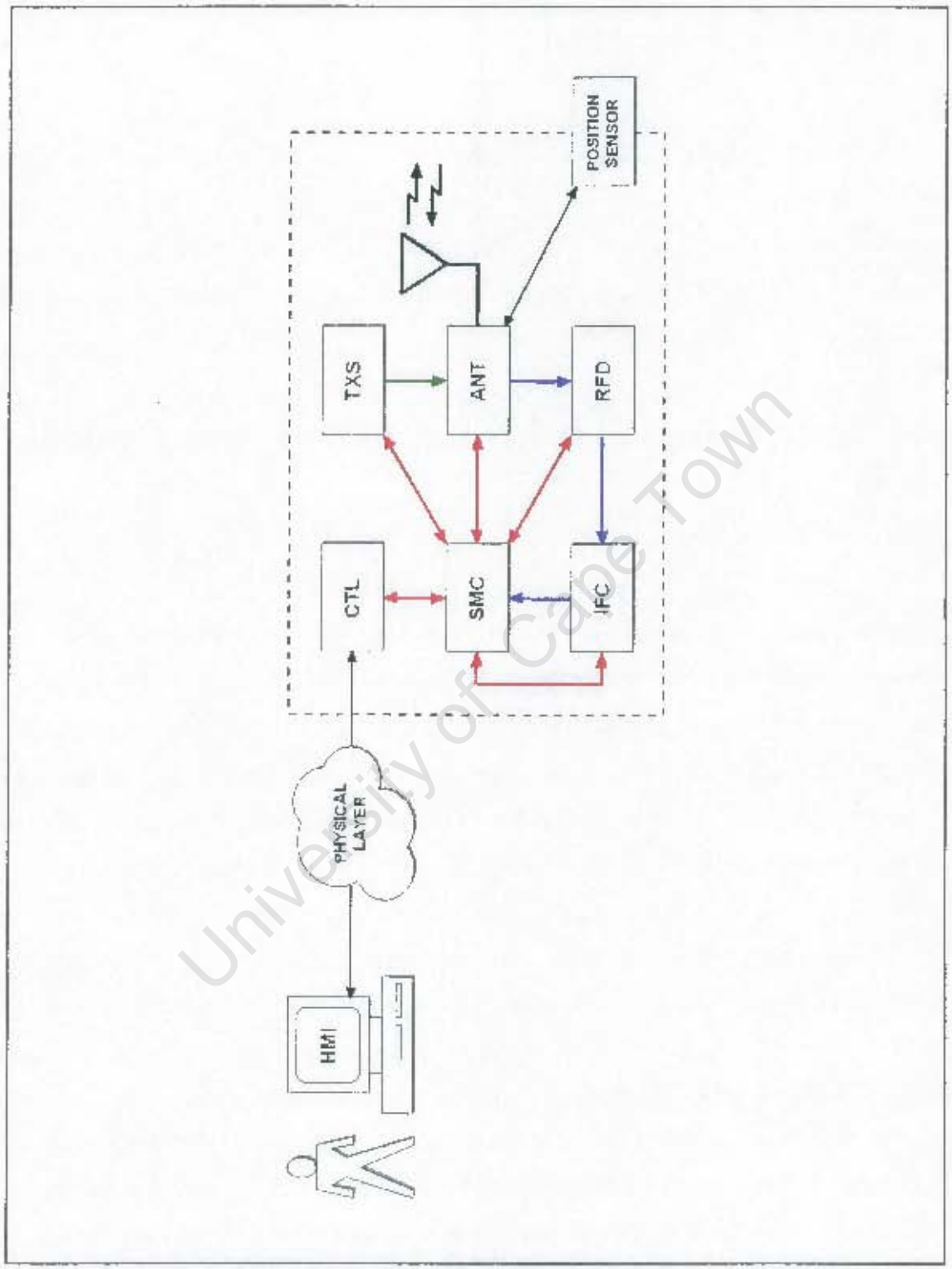

Figure 1 OpenFuel (Pty) Ltd Ground Penetrating Radar System Block Diagram

The radar electronics modules interface and communicate via a proprietary communications interface developed by OpenFuel (Pty) Ltd. The communication interface is known as the Radar Module Link (RML) interface and is based on low voltage differential signalling (LVDS). The RML interface is shown in red on the block diagram in **Figure 1**. The SMC module acts as the network master as the RML interface is based on a star network topology.

The radar signal transmission path is shown in green and the reception path is shown in blue on the block diagram. Both these radar signal paths connect to the ANT, which connect to the antenna. The antenna is a separate unit to the radar electronics; designed and manufactured by an independent company. Each part of the radar system outlined above will briefly be discussed to indicate its function in the system.

2.1.2 Human-Machine Interface

The HMI execute the radar application software and also acts as the data storage device for the capture radar data. The user interacts with and controls or configure the radar via the HMI. The HMI also display the data captured from the sub-surface radar electronics in various formats.

2.1.3 Communications Physical Layer

The communications physical layer refers to the communications interface between the HMI and the sub-surface radar electronics. The main implementation of the physical layer is full-duplex RS-422 over CAT5 UTP cable, but there is also a communication-over-power implementation. The implementation method is transparent to the radar system so it will not be discussed further.

2.1.4 Communications and Radar Controller Module

The CTL module implements the communications interface which allows the HMI to

communicate with the sub-surface radar electronics. The CTL also implements some basic control functions for the radar system, like controlling the power-on sequence and monitoring the power supply levels.

2.1.5 Sampler and Controller Module

The SMC module performs the sampling of the intermediate frequency signal from the IFC module. It also controls the rest of the radar electronics through the RML interface. The SMC also implements the sample averaging and calibration routines for the captured waveforms as well as monitoring the temperature of the sub-surface radar electronics.

2.1.6 Transmit Synthesizer Module

The TXS module generates the radar signal that is transmitted via the radar antenna. The TXS module is fully configurable from the HMI as far as the output power, frequency and gain of the transmitted radar signal go. It can also measure a number of parameters and pass the information on tho the HMI.

2.1.7 Receive Demodulator

The function of the RFD is to receive and demodulate the radar signal to the intermediate frequency and then pass it onto the IFC module. The RFD module is fully configurable from the HMI as far as the input power, frequency and gain of the signal go. As in the case of the TXS, it can also measure a number of parameters and pass the information on tho the HMI.

2.1.8 Intermediate Frequency Cancellation Module

The function of the IFC module is to cancel out any error signal that arises from power leakage between the transmit section and the receive sections of the radar electronics. This

is done by generating and adding an inverted noise signal to the received radar waveform before sending it onto the SMC module.

2.1.9 Antenna Controller

The ANT implements a variety of functions pertaining to the radar antenna subassembly. These include the control of the antenna calibration and matching circuitry. This circuitry is located on the antenna subassembly itself. The ANT also provides relative radar position information to the HMI through a position sensor located in the antenna subassembly. The control of the antenna subassembly is done via the HMI.

2.1.10 Radar Module Link Interface

The RML interface forms the backbone of the sub-surface radar electronics. The RML interface forms a virtual memory bus connecting all the modules in the sub-surface radar electronics and accessing them with unique addressing. The RML interface structure is based on a star network topology with the SMC acting as the network master. The RML interface will be discussed further when dealing with the software and firmware implementations for the radar user interface.

2.2 Commercial-of-the-Shelf (COTS) GPR Systems

A survey of the commercial ground penetrating radar system market yielded a number (<10) of commercial manufacturers of these systems. All, but one of these systems, are of a modular design with the user interface being a separate unit; usually an industrialised personal computer or laptop computer.

One system, produced by Geophysical Survey Systems, Inc. (GSSI), seems to be the only ground penetrating radar system to integrate the radar user interface with the radar electronics and not use a standard or industrialised personal or laptop computer in the process. They also produce a variety of portable handheld radar systems that integrate the

antenna subassembly into the radar system as well. This design is interesting as a similar implementation can be done with the OpenFuel (Pty) Ltd radar system and the radar user interface. **Figure 2**[1] shows two views of the GSSI TerraSIRch SIR-3000 GPR system without an antenna, which is a separate module.[1] The top image shows the display area and the user control buttons. The bottom image shows the rear view of the system with the antenna connection and other interfaces being clearly visible. The unit is contained in a water-resistant, shockproof enclosure.

The key system specifications of the system according to the literature supplied by GSSI are listed below.

- **Processor:** 32-bit Intel StrongArm™ RISC Processor @ 206 MHz
- **Display:** Enhanced 8.4" TFT, 800 x 600 resolution, 64K colours
- **Internal Memory:** 256 Megabit Flash memory card
- **Compact Flash Port:** memory up to 1 Gigabyte
- Ethernet interface
- RS-232 interface
- USB interface
- GPS interface
- **Operating Temperature:** -10°C to +40°C ambient
- **Dimensions:** 31.5 (L) x 22 (W) x 10.5 (H) cm
- **Weight:** 4.1 kg, including battery

This commercial ground penetrating radar system served as an excellent example of what issues had to be considered in the overall design of the radar user interface for this design project. These issues were defined to be:

- integrated vs. modular design,
- processor selection criteria,
- user interface design,
- data storage medium selection,

[1] Figure 2 is reproduced with permission from Geophysical Survey Systems, Inc. (GSSI), who is the copyright holder of the images.

Figure 2 GSSI TerraSIRch CIR-3000 Ground Penetrating Radar System

- interface options and
- added functionality.

These issues are discussed further in the chapter dealing with the initial analysis and concept design of the design project.

2.3 Conclusions

In this chapter we looked at the architecture of the existing ground penetrating radar system to which the radar user interface must be integrated. It is important to understand the basic hardware structure of the ground penetrating radar system in order to advance to the next phase which is the generation of the user requirements and system specification.

We also looked at commercial ground penetrating radar systems in order to understand implementation strategies and design concepts that can be integrated into the system specification. We concluded that a design approach of integrating the HMI and the radar electronics was the optimum route to follow.

Chapter 3

User Requirements and System Specification

The first step in the design process is to generate the user requirement for the radar user interface. The user requirements will be used in the second step of the design process to generate the system specification for the radar user interface. This design flow is necessary to ensure that the final product meets the initial user requirements. This chapter deals with these two aspects of the design project.

3.1 Functional User Requirements

This section details the functional user requirements of the radar user interface (RUI) based on the project description and background.

3.1.1 Electrical

- The RML high speed serial interface must comply with the TIA/EIA-644 (LVDS) specification.
- The RUI to external HMI serial interface must comply with the TIA/EIA-422 specification.
- The debugging serial interface must comply with the TIA/EIA-232-F specification.
- The keyboard and mouse interfaces must comply with the USB 1.1 specification.
- The Ethernet interface must comply with the IEEE 802.3 specification.
- The RUI will operate from a +7V to +12V external input voltage and internally generate any of the supply voltages required by the RUI.
- All digital logic components on the motherboard and RUI module must be compatible with +3.3V systems.
- All efforts must be made to reduce EMI generation and ESD susceptibility.
- The RUI must be designed to be as low power as possible.

3.1.2 Mechanical

- The width and length dimension of the RUI must be kept to a minimum.
- The RUI must be implemented on a PCB not exceeding 8 layers.

3.1.3 Configuration and Programming

- The FPGA must be configurable via a JTAG interface.
- A FLASH-based configuration device must be provided for the FPGA to run stand-alone.
- The FPGA configuration device must be configurable via a JTAG interface.
- The microprocessor must be programmable via a JTAG interface.

3.1.4 Communication

- The RUI interface to the radar electronics must by via an RS-422 communications channel.
- The RUI must also be able to interface to the radar via an LVDS RML link.
- The RUI debugging serial interface must be via an RS-232 (no handshaking) communication channel.
- The RUI must be able to act as the network master of the RML embedded network.
- The communications interfaces must be capable of handling all data and control information required to operate the radar in real-time at the maximum sampling rate of the radar (50 profiles of radar data per second).
- The LVDS high speed serial link must be capable of data rates up to 500Mbits/s.
- The RS-422 serial link must be capable of data rates up to 10Mbits/s.
- The RS-232 serial link must be capable of data rates up to 120kbits/s.
- The USB interface must support data rates compatible with the USB 1.1 specification.
- The Ethernet interface must support data rates of both 10Mbps and 100Mbps.

3.1.5 Functional

- The RUI shall provide a suitable local HMI in the form of an LCD and USB peripherals.
- The RUI shall provide sufficient processing power to operate a local HMI for the control of the radar.

3.1.6 Environmental

- The RUI shall operate from -20˚C to +80˚C.
- The RUI shall survive storage from -20˚C to +80˚C.
- All components used in the RUI shall be of industrial temperature range.

3.2 Hardware User Requirements

Based on the functional user requirements, the following hardware user requirements were drawn.

- A microprocessor capable of handling the HMI and control requirements of the radar
- Non-volatile memory for the microprocessor program storage
- High density non-volatile memory for data storage
- High density volatile memory for program storage
- An FPGA to implement the RML, RS-422 and RS-232 communication interfaces
- Stand-alone FPGA configuration
- JTAG FPGA, configuration device and microprocessor configuration and programming
- On-board power supplies
- RML high speed serial interface
- RS-422 transceiver
- RS-232 transceiver

- USB interfaces for a keyboard and mouse
- LCD for local HMI display
- Ethernet interface
- Industrial temperature range components
- Real time clock for time stamping
- Low power design

3.3 Software and Firmware User Requirements

Based on the functional user requirements, the following software and firmware user requirements were drawn.

- RML must be realisable in standard FPGA technology.
- The FPGA firmware control logic will be designed and implemented in VHDL.
- The system must contain RML master and slave interfaces and be compliant with the RML specification.
- The firmware must be configurable and modular.
- The firmware should allow components to be reused for other modules in the radar and other projects.
- The firmware must be flexible and configurable at a high level.
- All parameters and control interfaces of the firmware must be readable, modifiable and controllable.
- The firmware must provide an interface to these parameters and control interfaces via RML.
- The RUI must interface to the radar and be compliant with the current HMI specification.

3.4 System Specification

The following system specification was drawn up based on the user requirements from the previous section.

3.4.1 Electrical

- RUI external supply voltage ..... +7V to +12V
- RUI internal power supply voltages and maximum supply currents
  - +1.22V @ 1A
  - +1.5V @ 1A
  - +2.5V @ 1A
  - +3.3V @ 1A
  - +5V @ 1A

3.4.2 Mechanical

- RUI size must be kept as small as possible

3.4.3 Configuration and Programming

- Interface type ..... JTAG

3.4.4 Radar Module Link

- Type ..... LVDS
- Data rate ..... < 500Mbit/s
- Cable length ..... < 10m
- Protocol ..... RML

3.4.5 HMI Serial Link

- Type RS-422
- Data rate ..... < 10Mbit/s
- Cable length ..... < 100m

- Protocol ..... SLIP

3.4.6 Debugging Serial Link

- Type RS-232
- Data rate ..... < 120kbit/s
- Cable length ..... < 3m
- Protocol RS232

3.4.7 USB Interface

- Type ..... USB
- Protocol ..... USB 1.1
- Cable length ..... As per specification
- Protocol ..... USB

3.4.8 Ethernet Interface

- Type ..... Ethernet
- Data rate ..... 10 & 100Mbps
- Cable length ..... As per specification
- Protocol ..... TCP/IP

3.4.9 Functional

- Provide local HMI for radar control

3.4.10 Environmental

- Operational temperature ..... -20˚C to +80˚C

- Storage temperature ..... -20°C to +80°C

3.5 Conclusions

In this chapter we have generated the user requirements and system specification for the design project. As we now have the system specification for the radar user interface, we can proceed to the next phase which is the analysis of the system specification and the concept design of the system that can be done based on the analysis.

Chapter 4

Initial Analysis and Concept Design

From the system specification we could now generate a concept design for the radar user interface. The concept design did not include any hardware specific designs point, but it was rather a visualisation of the system specification. Before we did that though, it was necessary to do an analysis of the technology available in which to implement the radar user interface in order to ensure the correct hardware selection for the hardware implementation phase of the design project.

4.1 Initial Analysis

4.1.1 Industry and Technology Overview

In the last few years with the increased pressure from the mobile computer and wireless communication applications market, the emphasis on packaging and integration has increased. Furthermore, the semiconductor industry in general is driven by five major factors.[2] These factors, in no specific order of importance, are:

- optimizing system performance,
- speed and complexity,
- design to market,
- product size constraints and
- power density.

The optimization of system performance and power density, coupled with the increase of speed and complexity, coupled with the external pressure from the mentioned market sector has led to the development of single-chip integration design strategies rather than that of stand-alone or application specific integrated circuit (ASIC) design strategies. The advantage of this approach has also led to a reduction in the time it takes to design and produce a new product, due to the high level of integration of technology.

4.1.2 The SoC Revolution

The implementation of an entire electronic system on a single chip through the integration of multiple components is known as system-on-chip (SoC) technology. The primary advantages of using SoC technology over the usual integration of components on conventional printed circuit boards (PCB) are:

- increased operating speeds,
- lower power consumption,
- reduced size and complexity of the end-user product,
- low manufacturing costs and
- increased system reliability.

These advantages enable end-user electronics designers and manufacturers to leverage the technology and improve the size, cost and performance of end-user products. Another, even more important advantage, is the opening of new application areas where a PCB implementation would have been unusable or impossible. Examples of these applications would include any product from the wireless communication applications' market.

The development of SoC semiconductor components has led to a revolution in the semiconductor industry. There has been a steady increase in the revenue generated by SoC technology. In 1997 the SoC share in the semiconductor market was 5.2% and it has been increasing to 11.8% in 2002.[3] There has also been a steady increase in the technology, i.e., logic, SRAM, Flash etc., that has been implemented in SoC technology.

The obvious advantages of SoC technology, coupled with the user requirements for this design project has led us to pursue SoC technology for the implementation of the radar user interface.

4.1.3 SoC Processor Selection

There exists a vast number of SoC processors in the current market.[4] These processors can broadly be classified according to the architecture they implement, as shown below.

- ARM

- MIPS
- PowerPC
- x86
- 68000
- Other (any not listed before)

It was necessary to lay down some key criteria to aid in the selection of the SoC processor for the radar user interface. These key criteria were defined to be:

- SoC processor architecture,
- power consumption vs. relative speed,
- peripheral integration and
- design resources and support.

The available processor architectures were evaluated first. The results of the evaluation process are shown in **Table 1**. The evaluation was done by evaluating the SoC processors that were commercially available at that stage.

Table 1 SoC Processor Architecture Selection Criteria

SoC Processor Architecture	Power Consumption vs. Relative Speed	Peripheral Integration	Design Resources and Support
ARM	Good	High	High
MIPS	Good	High	High
PowerPC	Moderate	Moderate	Moderate
x86	Moderate	Moderate	High
68000	See Text*	See Text*	High
Other	See Text**	See Text**	Low

* Processors that fell in the *'68000'* category was not considered as these processors generally did not have a memory management unit (MMU) and thus could not provide the processing power needed to execute the radar application software. Similarly, processors

in the *'ARM'* category that implemented the ARM7 architecture, generally did not have an MMU and were thus also excluded from the survey. ** Processors that fell in the *'Other'* category were not considered either, because there tended not to be a lot of design resources and support available for them due to their limited and application specific use. Their availability in limited quantities for development purposes was also a concern.

Looking at the other criteria we can see the ARM and MIPS architectures being generally better than either PowerPC or x86. The reason for this is due to the use of PowerPC and x86 architectures in more processor intensive areas such as embedded personal computers and the ASIC market. ARM and MIPS architectures are used more in PDA and cellular applications where low power consumption is the major design factor. These reasons caused us to only focused on ARM and MIPS architecture-based SoC processors that were specifically intended for low power consumption applications.

Taking the next step, we looked at the criteria of power consumption vs. relative speed and peripheral integration. The criteria of design resources and the level of support were deemed to be equal at this stage. Peripheral integration of the SoC processors was measured against the radar user interface system specification to ensure only processors that could meet the requirements were evaluated. The following peripherals had to be present on the SoC processor to qualify it for consideration.

- Integrated LCD Controller
- Static Bus Interface
- Separate SDRAM Controller
- Dual USB Interface
- Dual UART Interfaces
- Ethernet Interface
- Bulk Storage Interface, like Compact Flash or PCMCIA

Table 2 shows the ARM and MIPS architecture SoC processors that were evaluated based on these criteria.

It was difficult to compare the different processors with the information supplied by the manufacturers, as they implement different architectures, peripherals and operate at different clock speeds. Nevertheless, we had to assume that all the characteristics were

more or less equal in order to proceed. From the table it is clear that the AMD Alchemy Au1100 MIPS32 series of processors had the best power consumption vs. relative speed characteristic. We decided to use this range of SoC processors as it also had added power reduction features that we will discuss in the chapter on the hardware implementation of the radar user interface.

Table 2 SoC Processor Power Consumption Comparison

Processor	SoC Processor Architecture	Power Consumption vs. Relative Speed
AMD Alchemy Au1000 266MHz	MIPS32	<300mW @ 266MHz
AMD Alchemy Au1000 400MHz	MIPS32	500mW @ 400MHz
AMD Alchemy Au1000 500MHz	MIPS32	900mW @ 500MHz
AMD Alchemy Au1100 333MHz	MIPS32	<200mW @ 333MHz
AMD Alchemy Au1100 400MHz	MIPS32	250mW @ 400MHz
AMD Alchemy Au1100 500MHz	MIPS32	400mW @ 500MHz
Atmel AT91RM9200	ARM920T	120mW @ 180MHz (abs. min. estimate)
Cirrus Maverick EP9301	ARM920T	100 - 675mW @ 166MHz
Cirrus Maverick EP9302	ARM920T	<500mW @ 200MHz
Cirrus Maverick EP9312	ARM920T	45 - 750mW @ 200MHz
Cirrus Maverick EP9315	ARM920T	100 - 750mW @ 200MHz
Freescale i.MX1 MC9328MX1	ARM920T	800mW @ 200MHz

Processor	SoC Processor Architecture	Power Consumption vs. Relative Speed
Intel XScale PXA255 200MHz	ARM5TE	178mW @ 200MHz
Intel XScale PXA255 300MHz	ARM5TE	283mW @ 300MHz
Intel XScale PXA255 400MHz	ARM5TE	411mW @ 400MHz
NEC VR4121 131MHz	MIPS64	300mW @ 131MHz
NEC VR4121 168MHz	MIPS64	385mW @ 168MHz
NEC VR4122	MIPS64	270mW @ 180MHz
NEC VR4131	MIPS III & MIPS16	270mW @ 180MHz
NEC VR4181A	MIPS III & MIPS16	<500mW @ 131MHz
Samsung S3C2440	ARM920T	368mW @ 400MHz
Sharp LH7A404	ARM922T	265mW @ 200MHz (abs. min. estimate)

4.2 Concept Design

The next task was to generate a concept design for the radar user interface. It was decided to divide the radar user interface design into two parts, namely the RUI module and the RUI motherboard. This was done in order to allow for the upgrade of the processor section of the radar user interface, i.e. RUI module, without having to re-manufacture or redesign the whole radar user interface.

4.2.1 Radar User Interface Module

The concept design for the RUI module is shown in **Figure 3**. It consists of the AMD Au1100 Alchemy MIPS32 SoC processor, the non-volatile Flash boot memory and the

volatile synchronous dynamic random access memory (SDRAM). It also has the JTAG programming interface connector for programming the non-volatile Flash boot memory and the SoC processor. All the other signals from the processor are taken to a connector that will allow the RUI module to interface with the RUI motherboard.

4.2.2 Radar User Interface Motherboard

The concept design for the RUI motherboard is shown in **Figure 4**. It consists of a field -programmable gate array (FPGA), configuration device for the FPGA and interfaces for the peripherals. It also has the JTAG programming interface for the FPGA, which is only used for debugging the FPGA in this application, as well as the programming interface for the FPGA configuration device. The FPGA was required to implement the radar module

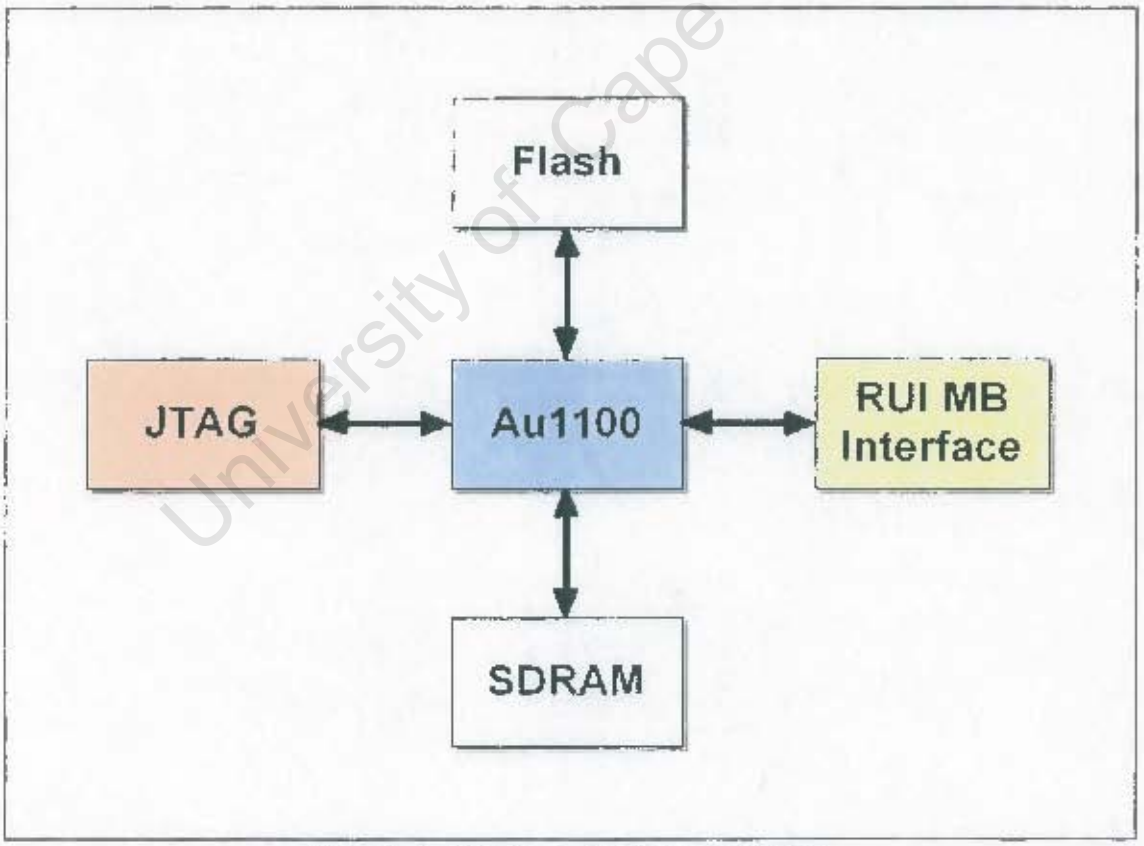

Figure 3 RUI Module Concept Design

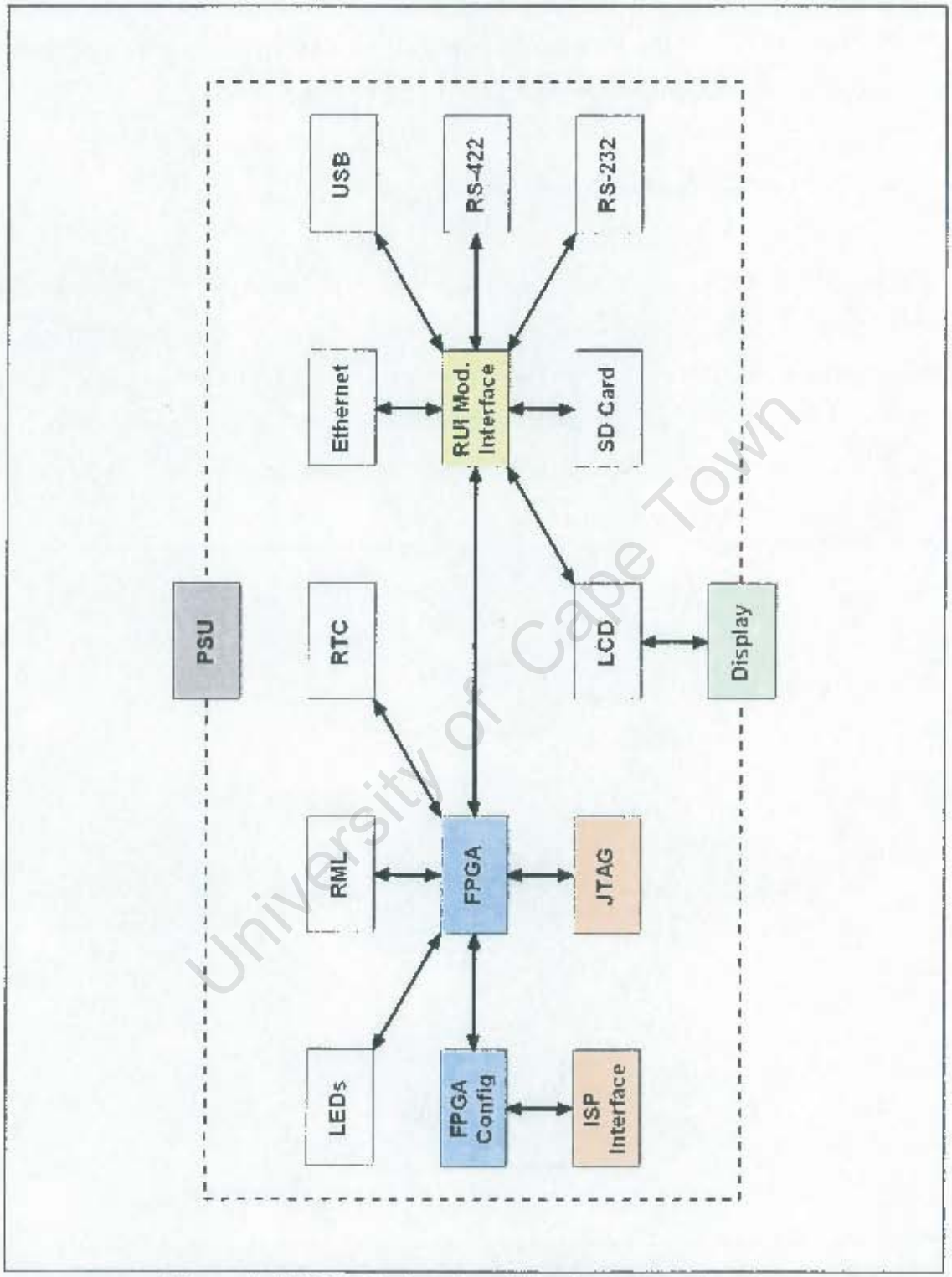

Figure 4 RUI Motherboard and Display Concept Design

link (RML) interface as well as providing logic interfacing between some of the components of the RUI motherboard. These interfacing functions are usually implemented in a complex programmable logic device (CPLD) or discrete logic components, but to reduce components and thus power consumption the FPGA was used for that function as well. The RML interface was required in the design to allow the radar user interface to connect with the radar electronics in the current ground penetrating radar system design. It was also decided to include an RS-422 interface in the design to allow the radar user interface to seamlessly integrate with the current ground penetrating radar system implementation that has been described previously.

For the peripherals, an Ethernet, two USB interfaces and an RS-232 interface was implemented. A real-time clock (RTC) was also added to the design to allow for the accurate time-keeping and time-stamping of radar data. Some light emitting diodes (LEDs) were added to the design to aid in debugging the hardware.

For the storage of radar data, two secure digital (SD) interfaces were provided that allow SD memory cards to be used. These interfaces also have other implementations like global positioning system (GPS) and wireless Ethernet interfaces. This functionality can thus be added to the radar user interface in the future without redesigning and re-manufacturing the RUI motherboard.

It was decided to use a commercial colour liquid crystal display (LCD) with an integrated touch screen to enable optional user interaction with the radar user interface through the touch screen rather than the keyboard or a mouse. A commercial display unit from Cogent Computer Systems, Inc. with an integrated touch screen controller and power supplies were selected in order not to bind the radar user interface hardware design to any specific LCD design.

The power supplies for the radar user interface would be located on the RUI motherboard. It would provide power to the RUI module and RUI motherboard. The LCD module would also receive its main power from the RUI motherboard.

The RUI motherboard would have connectors that would allow it to interface with the RUI module, LCD module, the Ethernet and USB peripherals as well as the RML, RS-232 and RS-422 communications interfaces.

4.2.3 Radar User Interface Mechanical Concept Design

The mechanical concept design of the radar user interface is shown in **Figure 5**. The LCD module fits on top of the RUI motherboard that also has the RUI module on top of it. The radar electronics sits on a separate carrier backplane that interfaces to the bottom of the RUI motherboard. It is not in the scope of this design project to produce the carrier backplane for the radar electronics. An implementation of this carrier backplane already exists in a different form in the current ground penetrating radar data system.

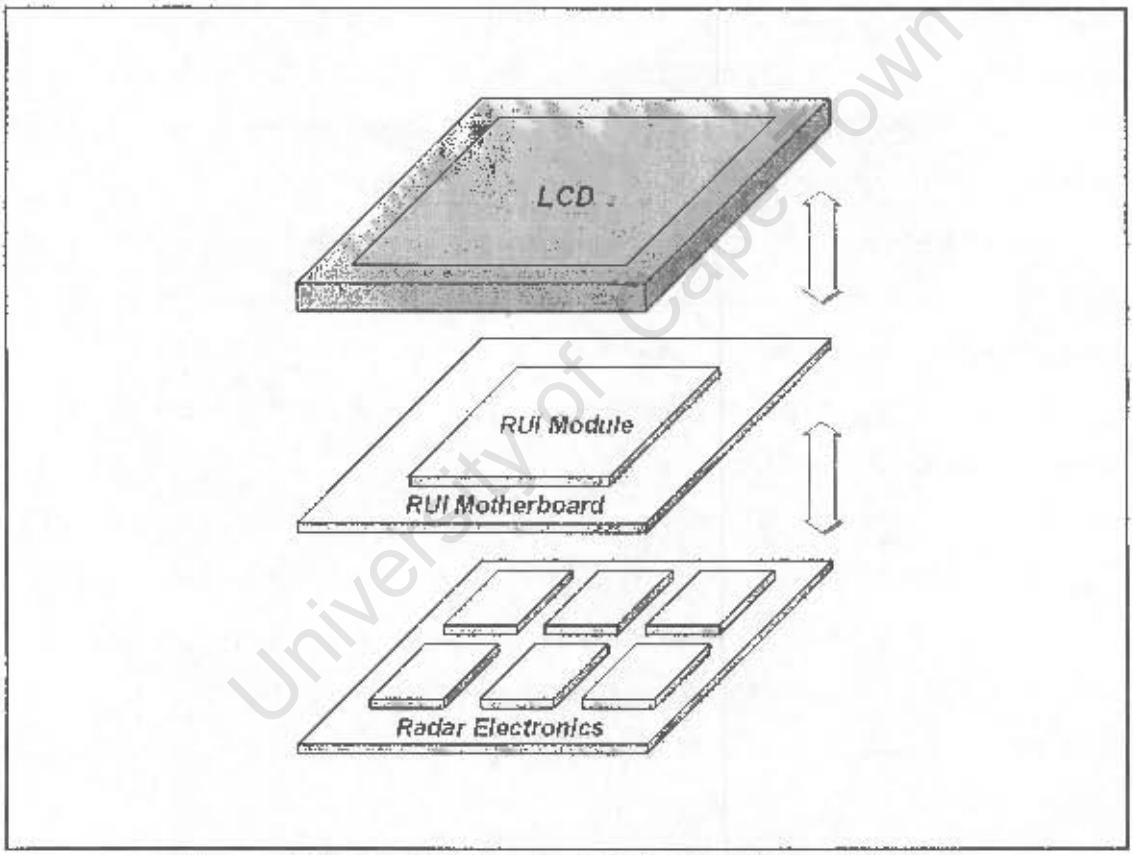

Figure 5 Radar User Interface Mechanical Concept Design

4.3 Conclusions

It this chapter we firstly had a look at the current semiconductor industry as far as

processor technology implementation goes. We concluded that SoC processor technology would be the ideal selection for the radar user interface. Following on from this, we selected a suitable SoC processor based mainly on the system specification and other important criteria like processor architecture, power consumption vs. relative speed and peripheral integration. The SoC processor selected was the AMD Alchemy Au1100 MIPS32-based processor as it had the lowest power consumption vs. relative speed of all the devices that had been evaluated.

Next, we did a concept design for the radar user interface based on the SoC processor selected and the system specification. It was decided to split the hardware implementation into two sections to allow for the reuse of the hardware platform. We will use this concept design to produce the hardware implementation of the radar user interface in the next chapter.

Chapter 5

Hardware Implementation

The concept design of the radar user interface that was generated in the previous chapter allowed us to implement the hardware for the radar user interface. We started by selecting the required components to optimise the design for low power consumption, followed by doing the schematic designs of the RUI module and RUI motherboard and finally implementing the schematic designs in printed circuit boards.

5.1 RUI Module Component Selection

5.1.1 SoC Processor

The microprocessor that was used in the RUI module was the AMD Alchemy Au1100-400MBC from AMD.[5] The processor is a complete MIPS architecture SoC processor based on the MIPS32 instruction set. It is specifically designed for maximum performance with low power consumption for mobile and handheld applications. The power dissipation for the device is less than 250mW for the 400MHz version, which is the speed version that we selected. The 500MHz version's power consumption was deemed too high (400mW) for a speed increase of a 100MHz over the 400MHz version. The device is highly integrated with on-chip SDRAM, SRAM/Flash EPROM memory controllers, an LCD controller, 10/100 Ethernet controller, USB 1.1 host and device controller, three UARTs, and general purpose inputs and outputs; up to 48 with 13 being dedicated.

5.1.2 Non-Volatile Program Memory

The RUI required non-volatile memory for program storage. The memory that was selected was an Intel Corporation StrataFlash® TE28F256J3C-125 memory, which is a

125ns (nanosecond) 256-Mbit (megabit) Flash memory.[6] The reason for selecting this specific Flash memory was Intel's use of two-bit-per-cell technology in the memory manufacturing. This means that the memory provided double the storage density when compared to standard memory devices in the same footprint without a twofold increase in power consumption, i.e. this memory consumes less power per Mbit when compared with one-bit-per-cell technology memory devices.

It was also decided to implement one of these memory devices to give the RUI module 32MB (megabyte) of non-volatile program memory storage. This was deemed enough memory to store the bootloader and the compressed program image.

5.1.3 SDRAM Volatile Program Memory

The RUI requires SDRAM for temporary program, data and variable storage. The Au1100 SoC processor has a separate SDRAM bus interface. This interface allows the SDRAM bus to operate at a lower voltage than the static memory bus of the processor. For this reason we selected a Micron Technology, Inc. MT48V16M16LFTG-10 Mobile SDRAM, which is 100MHz 256-Mbit SDRAM.[7] This memory is specifically design for low power consumption mobile device applications and operates from +2.5V compared to other devices that operate from +3.3V.

It was decided to implement the maximum amount of SDRAM that the Au1100 SoC processor was designed for to allow for the maximum amount of flexibility for the software implementation phase of the design project. This equated to two of these memory devices, giving a total of 64MB of SDRAM program memory.

5.2 RUI Motherboard Component Selection

5.2.1 Field-Programmable Gate Array (FPGA)

The RUI required an FPGA for the implementation of the RML interface and general logic interfacing. The FPGA that was selected for use in the RUI motherboard was a Xilinx

XC2V250-6FG256I Virtex™-II Platform FPGA.[8] This FPGA is a 250 000 gate device which has enough resources to implement the RML interface and general logic interfacing.

The FPGA needed a configuration device and an Atmel Corporation AT17LV002-10CI FPGA Configuration EEPROM Memory was selected.[9] This allows the FPGA to operate in a stand-alone application, i.e. the FPGA does not need to be manually reprogrammed every time the RUI is switched on and used.

5.2.2 10/100Mbps Ethernet Interface

The RUI had to have a 10/100Mbps Ethernet interface. The Au1100 SoC processor had an integrated Ethernet controller (MAC) and all it required was an Ethernet PHY (transceiver). The AMD Am79C874VI NetPHY™-ILP Low Power 10/100-TX/FX Ethernet Transceiver was selected.[10] This device was selected as it was designed for low power consumption applications and was recommended by AMD for use with the Au1100 SoC processor.

5.2.3 RS-422 HMI Interface

The RUI had to have an RS-422 interface to allow it to interface with the current ground penetrating radar system. It was decided to use the Sipex Corporation SP3485EN +3.3V Low Power Half-Duplex RS-485 Transceiver with 10Mbps Data Rate.[11] These devices were designed for low power applications and they typically consume only 1mA of supply current. These devices also have internal short-circuit protection. Although they are designed for RS-485 applications, they can be used for RS-422 applications as well. Two of these devices will be used to implement a transmit and a receive communication channel.

5.2.4 RS-232 Debug Interface

The RUI had to have an RS-232 interface for debugging and any general RS-232

communications functions. It was decided to use the Sipex Corporation SP3232EEY True +3.0V to +5.5V RS-232 Transceiver.[12] These devices were designed for low power applications and they typically consume only 0.3mA of supply current. These devices also have internal short-circuit protection.

5.2.5 Real Time Clock and Power-On Reset Circuit

The RUI had to have a real time clock (RTC) for accurate time-keeping and time-stamping of radar data. The Xicor, Inc. X1227S8I-2.7A Real Time Clock/Calendar/CPU Supervisor with EEPROM was selected.[13] The device was selected as it featured a RTC with clock/calendar, two polled alarms with integrated 512x8 EEPROM, oscillator compensation, CPU supervisor and a battery backup switch. The device is accessed via an I^2C interface and operates from an external 32.768kHz crystal.

5.2.6 Power Supply Units

The RUI would be supplied with +7V to +12V DC and it had to generate the required internal voltages; in this case +1.22V, +1.5V, +2.5V, +3.3V and +5V. The current requirements for the power supplies were calculated as best as possible. **Table 3** shows the results of the calculations. These values only reflect the current consumption of the major components that will be used in the RUI design. Generally, the rule of thumb for power supply current design is to design for 80% of the power supply capacity being the load current with a reserve of 20%. In this case we will design for 50% of the power supply current being the load current with a reserve of 50% giving a power supply current requirement of 1.5A. This was done as we did not have a very accurate power supply current consumption model. As an example, the current consumption of the FPGA would vary depending on the design that would be implemented in the device. It was also decided to use the same power supply device for all supplies in order to reduce the number of different components that will be required and so simplify the electronic component procurement process. It was decided to use the Linear Technology Corporation LT1767

Table 3 Calculated Maximum Power Supply Current Consumption					
Device	**+1.22V**	**+1.5V**	**+2.5V**	**+3.3V**	**+5V**
Am79C874 Ethernet PHY	-	-	-	130mA	-
AT17LV002 FPGA Configuration PROM	-	-	-	5mA	-
Au1100 400MHz SoC	287mA	-	86mA	17mA	-
Ethernet Connector	-	-	-	40mA	-
FPGA Oscillator	-	-	-	25mA	-
LCD Module (estimate)	-	-	-	-	500mA
MT48V16M16LFTG-10 SDRAM (x2)	-	-	330mA	-	-
SD Card (x2)	-	-	-	200mA	-
SP3232 RS-232 Transceiver	-	-	-	100mA	-
SP3485 RS-484 Transceiver (x2)		-	-	4mA	-
TE28F256J3C-125 Flash Memory	-		-	80mA	-
USB0 Device (limited)	-	-	-	-	100mA
USB1 Device (limited)	-	-	-	-	100mA
X1227 RTC	-	-	-	3mA	-
XC2V250 FPGA (minimum)	-	200mA	-	150mA	-
TOTAL	287mA	200mA	416mA	754mA	700mA

Monolithic 1.5A, 1.25MHz Step-Down Switching Regulators.[14] These regulators were

high-efficiency (90% maximum) switching regulators with a large input voltage range (+3V to +25V) for battery powered applications that were also available in fixed voltage versions and were selected for these reasons.

Table 4 shows a comparison between linear and switching regulators.[15] Although linear regulators have a clear advantage when it comes to reduced complexity, total cost and output ripple/noise when compared to switching regulators, switching regulators have the advantage when it comes to improved efficiency and reduced waste heat generated. Reduced size is also an advantage in high power type applications where a linear regulator would need a heatsink, while a switching regulator would not need a heatsink. Linear regulators only really have an advantage over switching regulators if the input and output voltage differential of the regulator is small. This is not the case in the RUI implementation where the input voltage can vary between +7V and +12V and the output voltages vary from +1.22V to +5V. Furthermore, a linear regulator can only step the input voltage down, so if the RUI is supplied with an input voltage below the minimum input voltage for any reason, the linear regulator would not operate correctly.

Table 4 Linear vs. Switching Regulator Comparison

Comparison Criteria	Linear Regulator	Switching Regulator
Function	Only Steps Down	Steps Down, Up or Inverts
Efficiency	Low/Medium	High
Waste Heat	High	Low
Complexity	Low	Medium/High
Size	Small/Medium @ Low Power	Larger @ Low Power compared to Linear Reg.
	Large @ High Power (Requires Heatsink)	Smaller @ High Power compared to Linear Reg.
Total Cost	Low	Medium/High
Ripple/Noise	Low	Medium/High

5.3 RUI Module Schematic Design

An overview of the design of the RUI module is shown in the block diagram in **Figure 6**. The schematic diagrams and parts list for the RUI module is included in **Appendix A.1** for reference purposes.

5.3.1 SoC Processor Design

The central component in the RUI module design is the AMD Au1100 SoC processor. It has an integrated SDRAM controller that can operate at a bus voltage of either +2.5V or +3.3V that is selectable by applying the required logic level (0 or 1) on the *VSEL* pin. In this design the SDRAM bus was configured to operate at +2.5V as the SDRAM used was of the +2.5V bus voltage type. For this reason the *VSEL* pin was tied to ground (logic 0). The SoC processor has a static bus controller that allows for the memory mapping of external memory devices. The SoC processor had to be configured to boot from either the SDRAM or static bus controllers by applying the required logic level (0 or 1) to the *ROMSEL* pin. In this design the SoC processor had to boot from the static bus controller. For this reason the *ROMSEL* pin was tied to ground (logic 0).

The SoC processor was designed to operate from a 12MHz crystal and has an internal phase-locked loop (PLL) to generate the required internal clock frequencies. The power supply connections for the PLL had to be decoupled with a special circuit consisting of a 22µF and a 10nF capacitor and a 100Ω resistor. There was also a 32.768kHz crystal connection that allowed for the implementation of an RTC in software in the processor. It was decided not to use this function of the processor as an external RTC was going to be implemented on the RUI motherboard. Nevertheless, the unused power supply connections for the 32.768kHz crystal connection also had to be decoupled with the same circuit as was used for the 12MHz crystal. The unused 32.768kHz crystal connections also had to be tied to +3.3V.

The SoC processor required a core voltage of +1.22V connected to the *VDDI* pins and an interface bus voltage of +3.3V connected to the *VDDX* pins. The internal SDRAM

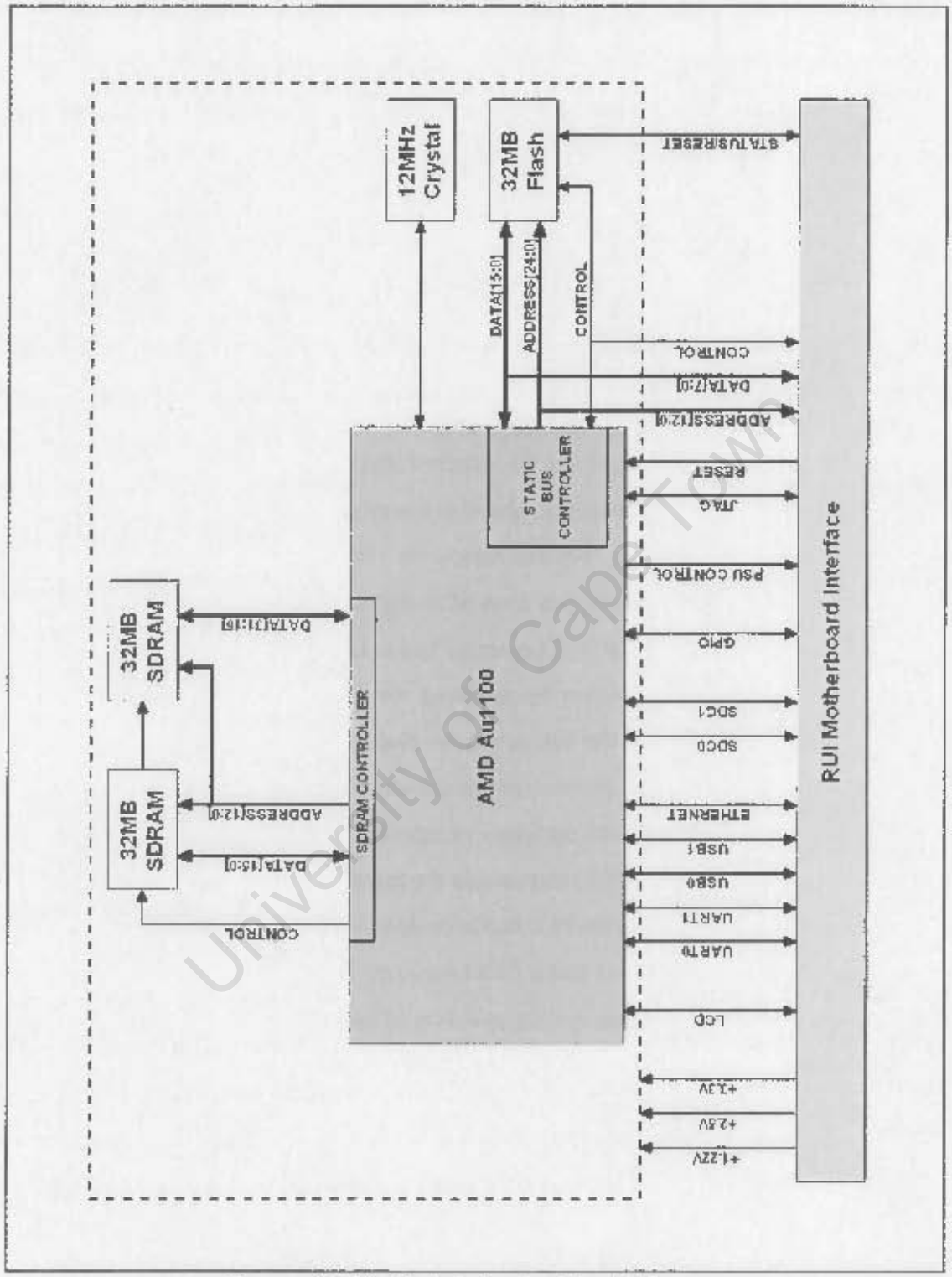

Figure 6 RUI Module Design Block Diagram

controller required a voltage of +2.5V connected to the *VDDY* pins as discussed earlier. The power supply pins on the processor was decoupled using mainly 100nF capacitors. A 6.8nF, 10nF and 15nF capacitor was also added to provide a broader spectrum of high-frequency noise filtering. A 10µF capacitor connected to each of the three supplies were also included in the design to filter out noise from the power supplies. The SoC processor had an output, called *PWR_EN*, that was used to enable the +1.22V power supply. This output goes to a logic high level (+3.3V) as soon as the *VDDXOK* input is asserted high. The *VDDXOK* input is used to signal to the processor that the +3.3V interface bus voltage supply (*VDDX* pins) is stable. The *VDDXOK* input was tied to the *RESETIN* input. The *RESETIN* input was connected to the FPGA on the RUI motherboard. This allowed for the control of the power-up sequence of the processor. The SoC processor provided a buffered output of the *RESETIN* input, called *RESETOUT*, that was not used.

The JTAG programming interface that allowed for the programming of the SoC processor and devices memory mapped to it had to be included in the design. It was decided to connect this interface through to the RUI motherboard and place the JTAG connector there to reduce the side profile of the RUI module. This would also reduce the overall side profile of the radar user interface.

5.3.2 Flash Non-Volatile Program Memory Design

As stated before, 32MB of Flash memory was added to the design to provide the non-volatile program memory. This memory would be used to store the bootloader and the compressed program image. The bus width of the boot memory had to be selected with the *ROMSIZE* input on the SoC processor by applying the required logic level (0 or 1) to the input. The boot size options were 16-bit or 32-bit data bus width memory devices. It was decided to include the option of booting from either size option to allow for the maximum flexibility in the design. Applying a logic 0 to the *ROMSIZE* input, boots from a 32-bit data bus memory device and applying a logic 1 to the *ROMSIZE* input, boots from a 16-bit data bus memory device. The Flash memory device was configured for a 16-bit data bus width by connecting the *BYTE* input to a logic high level. In this configuration the *A1*

address input becomes the least significant address bit. The *A0* address input was not used and thus connected to a logic high level. The *VPEN* input was also connected to a logic high level to allow for the device to be written to and erased.

The Flash memory was directly connected to the static bus controller with the *RAD[24:1]*, *RD[15:0]* and *ROE* signals. The static bus controller signals *RCS[3:0]*, *RBEN[1:0]*, *RWE* and *EWAIT* were connected to the FPGA on the RUI motherboard and the Flash memory control signals *RCS* and *RWE* were connected from the FPGA to the Flash memory. This allowed for the implementation of the Flash memory mapping and decoding logic circuitry in the FPGA, which allowed for a very flexible design. The static bus controller signals *RAD[12:0]* and *RD[7:0]* were also connected to the FPGA on the RUI motherboard to allow the SoC processor to access the RML interface implementation in the FPGA as a memory mapped device.

The Flash memory device had an input, called *RESET*, and an output, called *STATUS*, that was connected to the FPGA on the RUI motherboard as well. The *STATUS* output indicates if the Flash memory is busy with an internal operation. The *RESET* input resets the Flash memory and will be controlled during the power-up sequence by the FPGA.

The power supply pins of the Flash memory device were connected to +3.3V with some 100nF decoupling capacitors added to the design to provide noise filtering and power supply decoupling.

5.3.3 SDRAM Volatile Program Memory Design

Two 32MB SDRAM devices were added to the design to provide 64MB of volatile program memory. These devices were connected to the SDRAM controller of the SoC processor as described earlier. The SDRAM controller signals *SDA[12:0]*, *SDBA[1:0]*, *SDRAS*, *SDCAS*, *SDWE*, *SDCS0*, *SDCLK0* and *SDCKE* were connected to both SDRAM devices. The 32-bit data bus of the SDRAM controller was divided between the two SDRAM devices with the *SDD[15:0]* and *SDQM[1:0]* signals connecting to the least significant word SDRAM device and the *SDD[31:16]* and *SDQM[3:2]* signals connecting

to the most significant word SDRAM device.

The address configuration for the chip select signals of the SDRAM devices (*SDCS0*) can be configured in the SoC processor and this will be done by the bootloader software on start-up.

The power supply pins of the SDRAM devices were connected to +2.5V with some 100nF decoupling capacitors added to the design to provide noise filtering and power supply decoupling.

5.3.4 RUI Motherboard Interface Design

The RUI module interfaced to the RUI motherboard through four connectors. These connectors provided connection for the following devices and functions.

- 17 General Purpose Inputs/Outputs (1 Programmable Clock Output)
- Ethernet Controller
- Flash Memory Interface (*RCS*, *RWE*, *STATUS* and *RESET*)
- LCD Controller
- SD Card 0 & SD Card 1
- SoC Processor JTAG Interface
- SoC Processor Reset and Core Power Supply Enable
- Static Bus Controller (*RAD[12:0]*, *RD[7:0]*, *RCS[3:0]*, *RBEN[1:0]*, *RWE*, *ROE*, and *EWAIT* signals)
- UART0 & UART1
- USB0 & USB1
- +1.22V Power Supply
- +2.5V Power Supply
- +3.3V Power Supply

5.4 RUI Motherboard Schematic Design

An overview of the design of the RUI motherboard is shown in the block diagram in

Figure 7. The schematic diagrams and parts list for the RUI motherboard is included in **Appendix A.2** for reference purposes.

5.4.1 Field-Programmable Gate Array (FPGA) Design

The central component in the RUI motherboard design was the Xilinx XC2V250 Virtex-II FPGA. The FPGA was needed to implement the RML interface and general logic interfacing needed for the Flash interface and reset and power-up circuitry. The FPGA has a core (*VCCINT*) that operates from +1.5V, while the auxiliary circuitry (*VCCAUX*) operates from +3.3V. The inputs/outputs are arranged in eight banks that can each operate from a different voltage. Since all the external circuitry except the RML interface operated at +3.3V, the input/output banks (*VCCO-[3:0]* and *VCCO-[7:5]*) were supplied with +3.3V. The bank that connected to the RML interface (*VCCO-4*) was supplied with +2.5V. The power supply pins on the FPGA were decoupled using 100nF capacitors.

The FPGA had to have an external oscillator to provide the clock signal to drive the internal circuitry. A 32MHz oscillator was added to the design and connected to one of the digital clock managers (DCM) of the FPGA. The oscillator had to have an 1nF and 100nF decoupling capacitor for noise filtering.

The FPGA JTAG interface was also included in the design to allow for the debugging and programming of the FPGA during the firmware development process. The Atmel AT17LV002 configuration device and its in-system programming (ISP) connector was included in the design to allow the FPGA to operate in a stand-alone application. On power-up the firmware design is uploaded from the non-volatile configuration device to the FPGA. Some 100nF decoupling capacitors were added to the power supply connections of the configuration device.

It was decided to also include eight LEDs and some switches in the design to allow for the debugging of the hardware and firmware. These LEDs and switches were connected to the FPGA so that they could be accessed via the firmware design in the FPGA.

All the static bus controller signals *RAD[12:0]*, *RD[7:0]*, *RCS[3:0]*, *RBEN[1:0]*,

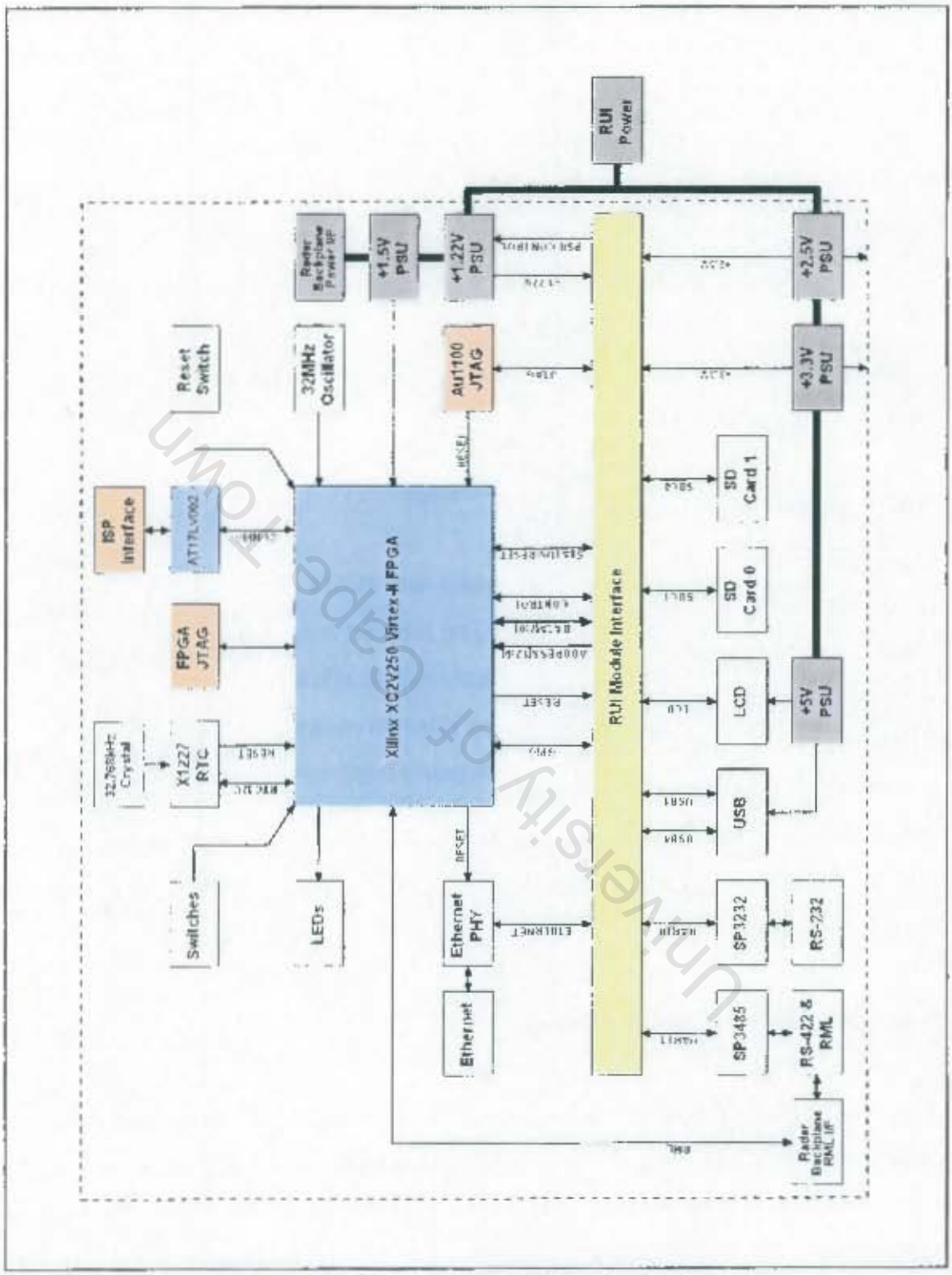

Figure 7 RUI Motherboard Design Block Diagram

RWE, *ROE*, and *EWAIT* were connected to the FPGA to allow the SoC processor on the RUI module to access the RML interface implementation in the FPGA. The 17 general purpose inputs/outputs from the RUI module were also connected to the FPGA to be implemented in firmware as required. The Flash memory control signals *STATUS*, *RESET*, *RCS* and *RWE* were connected to the FPGA as the memory control decoding would be implemented in firmware in the FPGA. The reset signals from the RTC, SoC processor and its JTAG connector and the Ethernet PHY transceiver were also connected to the FPGA as it forms part of the reset logic firmware design. Finally, the RTC I^2C signals were connected to the FPGA so that it could be routed to some of the general purpose inputs/outputs in firmware.

5.4.2 10/100Mbps Ethernet Interface Design

The Ethernet interface consisted of the AMD Am79C874 PHY transceiver connected to the Ethernet MAC on the SoC processor. The Ethernet transceiver also had to have an RJ-45 Ethernet connector and isolation transformer to allow it to connect to the external Ethernet network. It was decided to use an Ethernet connector with integrated magnetics and status LEDs to reduce the number of separate components used in the design and thus the complexity of the design. The actual detail of the Ethernet interface design will not be discussed here as the reference design in the datasheet of the Am79C874 PHY transceiver was implemented.

5.4.3 RS-422 HMI Interface Design

The RS-422 HMI design consisted of two Sipex SP3485 RS-485 transceivers that were implemented as a transmit and a receive communications channel. The SP3485 RS-485 transceivers were connected to UART1 of the SoC processor and allowed for the interfacing of the RUI to the radar electronics of the current ground penetrating radar system implementation. These devices had to have 120Ω termination resistors on each channel as well as 100nF decoupling capacitors for power supply decoupling and noise

filtering. The RS-422 HMI interface connects to a 9-Way D-Type plug that allows it to interface to the external radar electronics.

5.4.4 RS-232 Debug Interface Design

The RS-232 debug interface design consisted of the Sipex SP3232 RS-232 transceiver with its required capacitors and a 9-Way D-Type socket. The SP3232 RS-232 transceiver was connected to UART0 of the SoC processor and allows for debugging and any general RS-232 communications functions. The RS-232 debug interface only implemented the TX and RX signals of the RS-232 specification.

5.4.5 Real Time Clock and Power-On Reset Circuit Design

The Xicor X1227 real time clock (RTC) was included in the design to perform two functions. Firstly, it had to serve a time-keeping and time-stamping function. Secondly, it provided a power-on reset signal as well as a programmable watchdog timer reset signal. This reset signal was connected to the FPGA to allow for the implementation of this signal by the firmware in the FPGA. The RTC is accessed by the SoC processor via the I^2C interface, also connected to the FPGA.

The RTC required an external 32.768kHz crystal that provided the clock for the internal circuitry as well as requiring a backup battery and charging circuit. This would allow the RTC to maintain the correct time when the RUI was switched off.

5.4.6 Power Supply Units Design

The RUI motherboard and RUI module required five separate power supplies. The Linear Technology LT1767 switching regulators were used to implement the power supplies. For the +2.5V, +3.3V and +5V power supplies, fixed voltage versions of the LT1767 were used. For the +1.22V and +1.5V power supplies, a programmable output voltage version of the LT1767 was used. The output voltage of this device is programmed with an

external programming resistor which value is calculated with a fixed formula that is shown below.

$$R_{EXT} = \frac{10k \times (V_{OUT} - 1.2V)}{1.2V - (10k \times 0.25\mu A)} \quad \text{..... [16]}$$

The calculated values of the required resistors are shown in **Table 5**. As resistors are manufactured in standard resistance ranges, the closest available resistance value from the 1% tolerance range of resistors to the calculated resistance value had to be chosen. The percentage error between the required and calculated output voltages of the power supplies are also shown to ensure that the power supplies do not over- or under-supply the specific electronics circuits. From **Table 5** it can be seen that the selected resistor values produce output voltages with negligibly small error values.

The +1.22V power supply is controlled from the SoC processor via the shutdown pin on the LT1767 switching regulator. This allows for the core voltage of the SoC processor to only be applied once the other power supplies have stabilised.

Table 5 Calculated Resistor Values for +1.22V and +1.5V Power Supplies

Required Voltage	Calculated Resistor Value	Selected Resistor Value	Calculated Output Voltage	% Error
+1.22V	167.015Ω	169Ω	+1.220V	0%
+1.5V	2505.219Ω	2500Ω	+1.499V	0.067%

The power supplies are supplied with +7V to +12V from an external power supply or battery. The inputs to these power supplies have 10µF capacitors for noise filtering as well as multiple 10µF capacitors on the outputs of these supplies for power supply

decoupling and noise filtering. The voltage from the external power supply is also taken to a connector that would allow it to be fed through to the backplane for the radar electronics.

An important issue relating to the power supply design is to ensure that it can maintain the required output voltage at the maximum designed load current over the full supplied input voltage range. The maximum output current that the power supply design can supply in relation to the maximum load current of the regulator (I_P = 1.5A), the switching frequency of the regulator (f = 1.25MHz), the input voltage (V_{IN}), the required output voltage (V_{OUT}) and the output inductor value of the power supply design (L) is given by the formula shown below.

$$I_{OUT(MAX)} = I_P - \frac{(V_{OUT})(V_{IN} - V_{OUT})}{2(L)(f)(V_{IN})} \quad \text{..... [17]}$$

Table 6 shows the calculated values of the output current of the power supplies for an output inductor value of 10µH and a maximum continuous current of 1.7A. This specific inductor value was selected as it provides for higher output current values than lower inductor values. The trade-off is that inductors of larger inductance values are physically larger that inductors with smaller inductance values.

Table 6 Calculated Power Supply Maximum Output Current vs. Input Voltage

V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
+7V	1.460A	1.453A	1.436A	1.430A	1.443A
+12V	1.456A	1.448A	1.421A	1.404A	1.383A

From **Table 6** it can be seen that for the full input voltage range of +7V to +12V and the maximum designed output current of 1.5A the calculated output current for a 10µH output

inductor is in most cases very close to the 1.5A required and only varies with a maximum of 117mA under this value which is acceptable for this application as the power supplies were over designed by more than 100%.

Another important issue relating to the power supply design is to ensure that the catch diode used in the power supply design can handle the required current through it for the maximum designed load current. The average catch diode current for the power supply design in relation to the maximum load current (I_{OUT}= 1.5A), the input voltage (V_{IN}) and the required output voltage (V_{OUT}) is given by the formula shown below.

$$I_{D(AVG)} = \frac{I_{OUT} \times (V_{IN} - V_{OUT})}{V_{IN}} \quad \ldots\ldots\ldots\ldots\ldots [18]$$

Table 7 shows the calculated values of the output catch diode current of the power supplies for the required input voltage range.

Table 7 Calculated Power Supply Catch Diode Current vs. Input Voltage

V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
+7V	1.239A	1.179A	0.964A	0.793A	0.429A
+12V	1.348A	1.313A	1.188A	1.088A	0.875A

From **Table 7** we can see that the highest average current that will flow through the catch diode will be 1.348A, so we need to select a diode that is rated to handle this current.

Another important issue relating to the power supply design is to ensure that the inductor used in the power supply design can handle the required current through it for the maximum designed load current. The peak inductor current for the power supply design in relation to the maximum load current (I_{OUT}= 1.5A), the input voltage (V_{IN}), the required output voltage (V_{OUT}), the switching frequency of the regulator (f = 1.25MHz) and the

output inductor value of the power supply design (L) is given by the formula shown below.

$$I_{PEAK} = I_{OUT} + \frac{V_{OUT}(V_{IN} - V_{OUT})}{2(L)(f)(V_{IN})} \quad \text{.....} \quad |19|$$

Table 8 shows the calculated values of the output inductor current of the power supplies for the required input voltage range.

Table 8 Calculated Peak Power Supply Inductor Current vs. Input Voltage

V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
+7V	1.540A	1.547A	1.564A	1.570A	1.557A
+12V	1.544A	1.553A	1.579A	1.596A	1.617A

From **Table 8** we can see that the peak calculated output inductor current value is 1.617A, which means that the inductor we had previously selected would be adequate for this application as it has a peak output current rating of 1.7A.

A final important issue relating to the power supply design is to ensure that the output ripple current value is not excessive as this would negatively influence the operation of the digital circuits in the design. To this end we must calculate the output ripple current value for the power supplies for the input voltage range and then ensure that the correct type of capacitor was selected for power supply decoupling and noise filtering as recommended by the applications information in the datasheet of the LT1767. The important factor in selecting the type of power supply decoupling and noise filtering capacitor is the effective series resistance (ESR) of the capacitor which relates to the amount of ripple current it can handle and not the actual capacitance value of the capacitor. The RMS output ripple current for the power supply design in relation to the input voltage (V_{IN}), the required output voltage (V_{OUT}), the switching frequency of the regulator (f =

1.25MHz) and the output inductor value of the power supply design ($L = 10\mu H$) is given by the formula shown below.

$$I_{RIPPLE(RMS)} = \frac{0.29(V_{OUT})(V_{IN} - V_{OUT})}{(L)(f)(V_{IN})} \quad \ldots\ldots\ldots [20]$$

Table 9 shows the calculated values of the output catch diode current of the power supplies for the required input voltage range.

Table 9 Calculated Power Supply Output Ripple Current vs. Input Voltage

V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
+7V	23.4mA	27.3mA	37.3mA	40.5mA	33.1mA
+12V	25.4mA	30.5mA	45.9mA	55.5mA	67.7mA

From **Table 9** we can see that the highest calculated ripple current value is 68mA which means that a power supply decoupling and noise filtering capacitor with a maximum ESR rating of anything from 0.1Ω to 0.9Ω can be selected, according to the applications information in the LT1767 datasheet.

5.4.7 USB Interface Design

The USB interface design consisted of a dual USB Type-A connector for the two USB interfaces USB0 and USB1. This type of connector was used as it had a smaller footprint that two separate single USB Type-A connectors. The resistors on the USB signals were required as stated in the USB specification. The power for the USB interface was going to be supplied from the +5V power supply on the RUI motherboard. Some 100nF

decoupling capacitors were added to the design for power supply decoupling and noise filtering.

5.4.8 LCD Interface Design

The LCD interface design consisted of a connector compatible with the commercial colour LCD that was initially going to be used in the RUI. The power for the LCD module was going to be supplied from the +5V power supply on the RUI motherboard.

5.4.9 Radar Module Link Interface Design

The RML interface will be implemented in firmware in the FPGA and is thus connected to the FPGA. The RML interface signals also connect to the same 9-Way D-Type plug as the RS-422 HMI interface uses. This will allow the RUI to interface directly to an RML channel in the radar electronics of the current ground penetrating radar system implementation.

The RML interface is physically implemented with low voltage differential signalling (LVDS). LVDS is a data communication system that uses a very low voltage swing of about 350mV, differentially over two PCB signal traces or over a balanced cable. This characteristic of LVDS allows for high speed, low power, low cost and most importantly low noise communication. To maintain this low noise characteristic of LVDS some important design rules need to be followed. These rules are briefly outlined below.

- Use a solid ground plane beneath the LVDS signals to establish a controlled impedance for the transmission line interconnects.
- Isolate LVDS signals from all other signals, especially high speed digital signals with fast edge rates, to minimize crosstalk that may couple onto the LVDS lines.
- Minimize the length between the LVDS drivers and receivers and the LVDS connectors as mush as possible.
- Provide enough power supply decoupling and noise filtering for LVDS devices in the form of bypass capacitors.

- Power and ground connections of LVDS devices should be low impedance, i.e. use wide PCB traces.
- Minimize PCB ground return paths as mush as possible to minimize the loop for the image currents to return.
- Use two or more vias to connect the power and ground connections from bypass capacitors to minimize the effects of inductance.
- LVDS signal traces should be coupled close together and designed for 100Ω differential impedance.

As LVDS signal traces must be implemented in controlled impedance PCB traces, this means using either *Microstrip*, *Stripline* or *Broadside Stripline* signal traces. We decided to use *Microstrip* signal traces as we would probably only have one ground plane in the PCB. The formula for calculating the LVDS *Microstrip* differential impedance (Z_{DIFF}) in relation to the trace separation distance (S), trace width (W), trace thickness (t), trace and ground plane separation distance (h) and PCB material dielectric constant (ε_r) is shown below.

$$Z_{DIFF} \approx 2 \times Z_O \left(1 - 0.48e^{-0.96\frac{S}{h}}\right)$$

$$where$$

$$Z_O = \frac{60}{\sqrt{0.475\varepsilon_r + 0.67}} \ln\left(\frac{4h}{0.67(0.8W + t)}\right) \quad \ldots\ldots [21]$$

The best approach to using this formula is to use the minimum trace separation (S) that is possible in the PCB manufacturing process, fixed values for the trace thickness (t), trace and ground plane separation distance (h) and PCB material dielectric constant (ε_r) and then to manipulate the trace width (W) to obtain the required nominal differential impedance of

100Ω or a value in the range of 90Ω to 130Ω as stated in the LVDS specification.

A final design point on LVDS is to terminate the differential signal with a termination resistor that best matches the differential impedance of your transmission line. This value should be between 90Ω and 130Ω for point-to-point connections as stated before from the LVDS specification. Furthermore, fail-safe pull-up and pull-down resistors should be place on the differential signal pairs to force it into a known logic state in the event of a failure.

5.4.10 Non-Volatile Data Memory Design

The RUI required non-volatile memory for radar data storage. The SoC processor had two dedicated Secure Digital (SD) controllers as well as a PCMCIA or Compact Flash (CF) interface for two devices that were shared on the static bus and controlled by the static bus controller and some dedicated signals. The SD controllers allowed for the direct connection of two SD memory cards or SD slot devices. It was decided to rather use the two SD slot interfaces instead of the PCMCIA or CF interfaces in order to reduce the amount of traffic on the static bus, which would mainly be used for accessing the RML interface after the boot process. Currently, the memory cards come in different storage capacities up to 512MB in size which would provide more that enough storage space for the acquired radar data.

5.4.11 RUI Module Interface Design

The RUI motherboard interfaced to the RUI module through four connectors. These connectors have been described before, but are included again in this section on the RUI motherboard for completeness. These connectors provided connection for the following devices and functions.

- 17 General Purpose Inputs/Outputs (1 Programmable Clock Output)
- Ethernet Controller
- Flash Memory Interface (*RCS*, *RWE*, *STATUS* and *RESET*)

- LCD Controller
- SD Card 0 & SD Card 1
- SoC Processor JTAG Interface
- SoC Processor Reset and Core Power Supply Enable
- Static Bus Controller (*RAD[12:0]*, *RD[7:0]*, *RCS[3:0]*, *RBEN[1:0]*, *RWE*, *ROE*, and *EWAIT signals)*
- UART0 & UART1
- USB0 & USB1
- +1.22V Power Supply
- +2.5V Power Supply
- +3.3V Power Supply

5.5 RUI Module Printed Circuit Board Design

As the RUI module would plug directly onto the RUI motherboard we first did the PCB layout for the RUI module as its size would indirectly dictate the size of the RUI motherboard. The printed circuit board layout for the RUI module was done once the schematic diagrams were completed and verified for correctness. The PCB layout for the RUI module is included in **Appendix B.1** for reference purposes.

One of the user requirements of the design project was to keep the size of the PCB to a minimum and not to exceed eight layers in the PCB design. It was decided to use the maximum number of PCB layers as was allowed for in the user requirements as it would allow for a continuous ground plane and supply planes. A continuous ground plane is important as a slot cut into the ground plane will add inductance to signal traces running perpendicular to the slot in the ground plane and increase the inductance of the signal traces, increase crosstalk between them and slow down the rising edges of the signal traces.[22] Continuous power supply planes are also important as segmented power supply planes can introduce large amounts of self-inductance and mutual inductance in the various sections of the same power supply plane if it is not designed correctly.[23] For these reasons the eight PCB layers were functionally divided as shown below.

- Top Signal Layer
- Ground Plane
- Internal Signal Layer 1
- +1.22V Supply Plane
- +2.5V Supply Plane
- Internal Signal Layer 2
- +3.3V Supply Plane
- Bottom Signal Plane

As the power supply planes were solid, the edges of the power supply planes were masked back by a few millimeters to minimize PCB resonance effects as stated by the 20H rule.[24] The rule states that the edges of the power supply planes need to be pulled back by a factor of up to 20 times the plane-to-plane separation distance as this reduces the amount of radiation along the edges of the PCB being reflected back into the PCB and causing resonance within the PCB. This resonance can adversely influence the signal quality of high-speed digital signal lines. The trade-off with the 20H rule is that more energy will be radiated away from the PCB and will thus negatively influence EMC compliance, but then again you can't have everything in life.

As far as possible, the smallest surface mount components practically usable were used in the design. Availability of components was also a limiting factor in the package selection of the components. Four mounting holes were added to the design to allow the RUI module to be secured to the RUI motherboard although the four connectors along the edges of the RUI module will secure it firmly enough. The final size of the RUI module was 60mm by 80mm.

Once the PCB layout was completed, it was verified for correctness and then sent to be manufactured and the components placed. The PCB was manufactured from FR-4 based dielectric material as all the signals in the RUI module design were digital and it has been proven that it is the most suitable material for this specific type of application.[25] The manufacturing specification of the RUI module PCB was defined as follows with the layers placed as specified before.

- 1 ounce Copper Foil (Top Layer)

- 0.063mm x 3 Prepreg
- 1 ounce Copper 0.2mm Thin Laminate (Power Plane 1/Signal Layer 1)
- 0.063mm x 3 Prepreg
- 1 ounce Copper 0.2mm Thin Laminate (Power Plane 2/Power Plane 3)
- 0.063mm x 3 Prepreg
- 1 ounce Copper 0.2mm Thin Laminate (Signal Layer 2/Power Plane 4)
- 0.063mm x 3 Prepreg
- 1 ounce Copper Foil (Bottom Layer)

This manufacturing specification gave a nominal PCB thickness of 1.6mm. The copper layer thickness was specified at 1 ounce of copper per square foot of PCB to give sufficient current carrying capacity to the PCB.

Figure 8 shows the top and bottom views of the manufactured RUI module PCB. **Figure 9** shows the top and bottom views of the populated RUI module PCB with the most important features clearly visible.

5.6 RUI Motherboard Printed Circuit Board Design

The printed circuit board layout for the RUI motherboard was done once the schematic diagrams were completed and verified for correctness and the RUI module PCB layout was completed. The printed circuit board layout for the RUI motherboard is included in **Appendix B.2** for reference purposes.

As stated before, one of the user requirements of the design project was to keep the size of the PCB to a minimum and not to exceed eight layers in the PCB design. It was decided once again to use the maximum number of PCB layers as was allowed for in the user requirements as it would allow for a continuous ground plane and supply planes with the minimum amount of split supply planes in them. The reasons that this is required have been stated in the previous section. For these reasons the eight PCB layers were functionally divided as shown below.

- Top Signal Layer
- Ground Plane

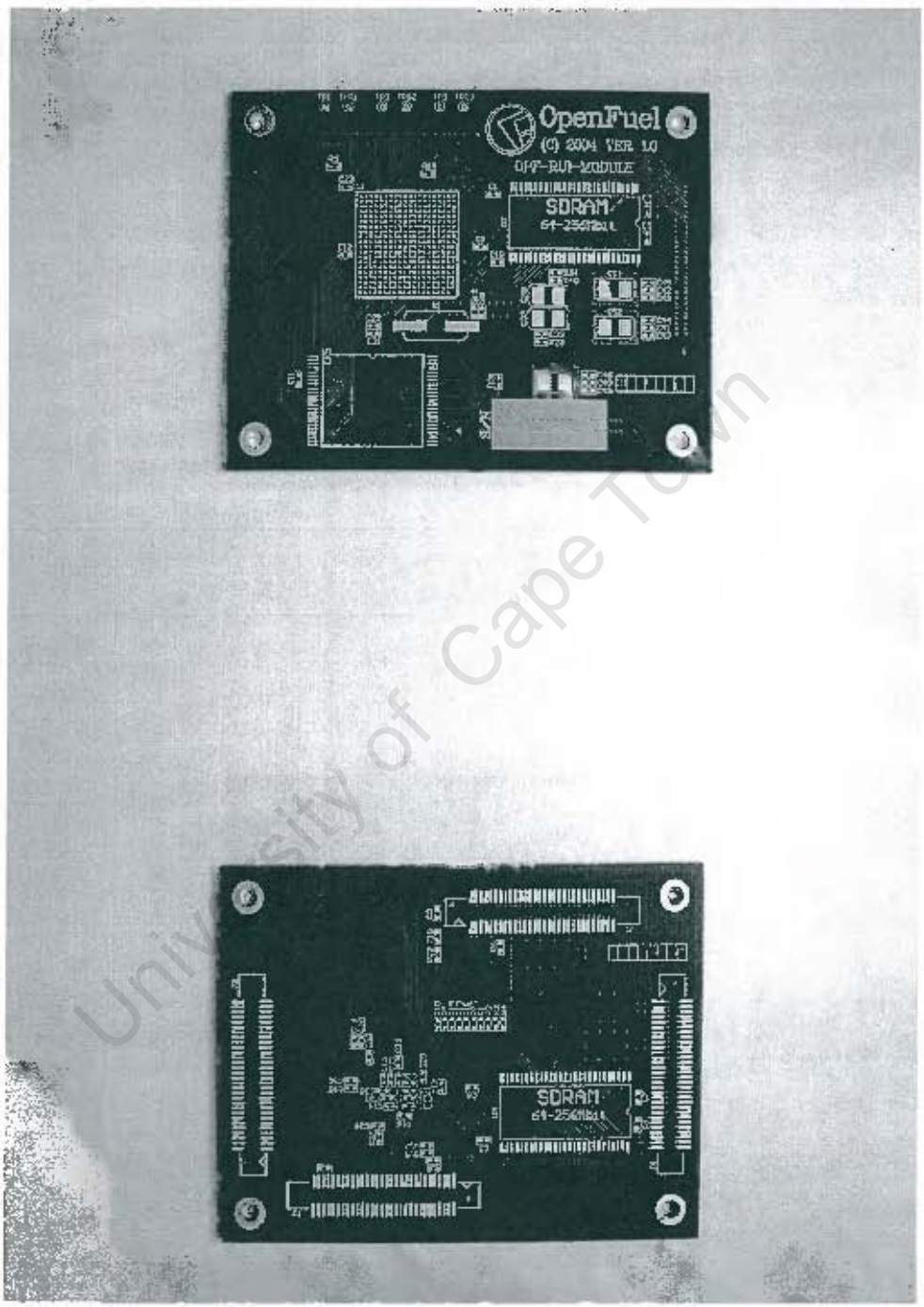

Figure 8 RUI Module Top and Bottom Unpopulated PCB View

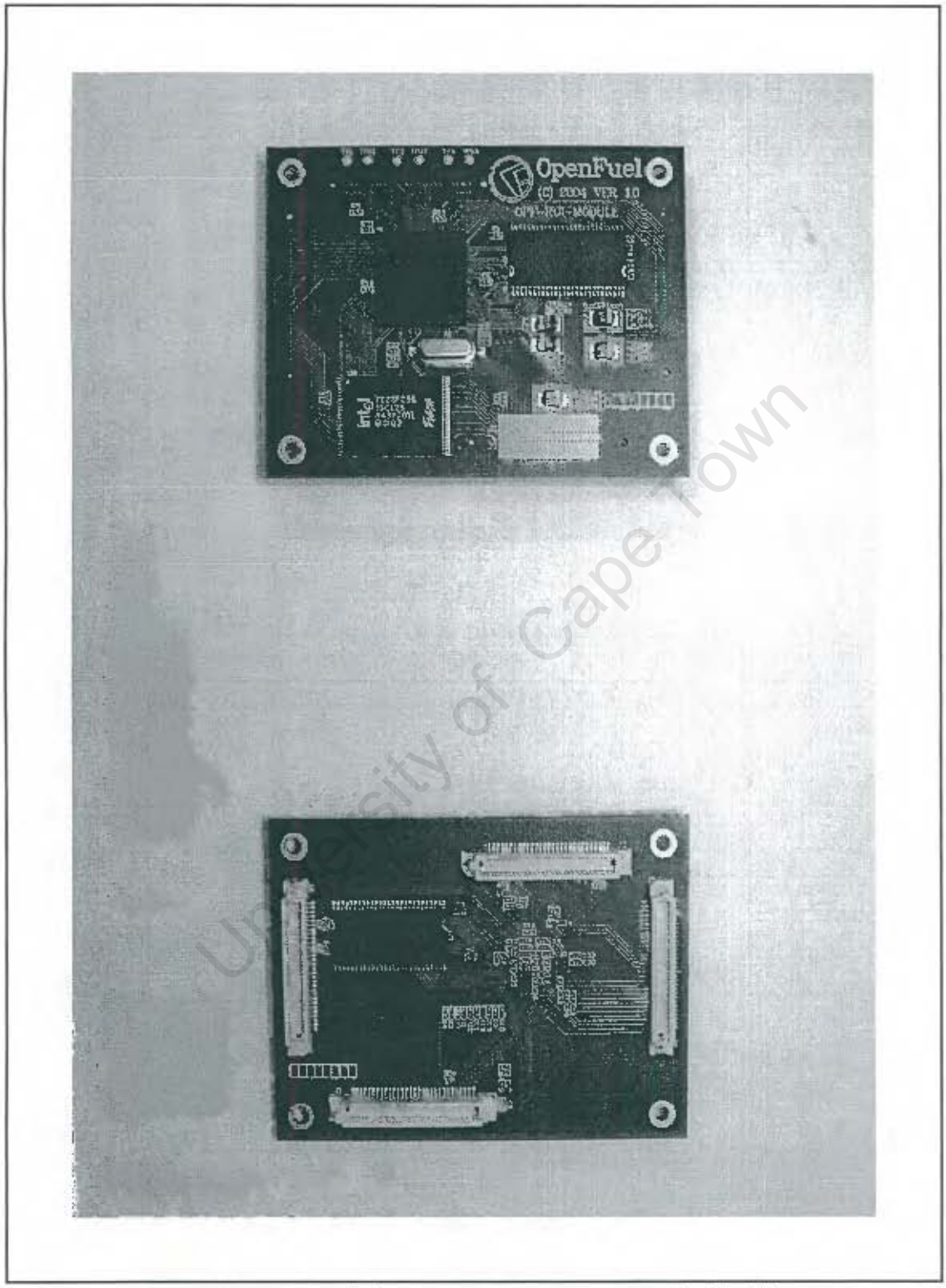

Figure 9 RUI Module Top and Bottom Populated PCB View

- Internal Signal Layer 1
- +1.22V/−1.5V Supply Plane
- +2.5V/+5V Supply Plane
- Internal Signal Layer 2
- +3.3V/V_{IN} Supply Plane
- Bottom Signal Plane

As the power supply planes were solid, the edges of the power supply planes were masked back by a few millimeters to minimize PCB resonance effects as stated by the 20H rule and as was done with the RUI module PCB layout.

As far as possible, the smallest surface mount components practically usable were used in the design as was done in the RUI module PCB layout. Availability of components were once again also a limiting factor in the package selection of the components. The size of the RUI motherboard PCB was made the same as the dimensions of the LCD and the four mounting holes of the LCD were aligned to those on the RUI motherboard. This was done to conform to the mechanical concept design that was shown previously. The four mounting holes for the RUI module were also placed and aligned to the RUI module PCB layout. The connectors for the USB, RS-422, RS-232, Ethernet and RML interfaces and the power connector were placed at the top of the PCB to allow for easy access and connection to external equipment. The two SD card slots were both placed on the right-hand side of the PCB to allow it to be easily accessed from the right-hand side, while the RUI is being held on the left-hand side.

The area under the Ethernet connector, which contained integrated magnetics, had to be cleared on all power supply and ground planes as these magnetics created a zone of high-frequency noise that could couple onto any planes in close proximity to it.[26] A guard trace was place around and between the RML interface signals to reduce crosstalk between these signals and other signals on the same PCB layer. The final size of the RUI motherboard was 120mm by 160mm.

Once the PCB layout was completed, it was verified for correctness and then sent to be manufactured and the components placed. The PCB was also manufactured from FR-4 based dielectric material as all the signals in the RUI motherboard design were digital and

it has been proven that it is the most suitable material for this specific type of application, as discussed previously. The same manufacturing specification as was used for the RUI module PCB was used for the RUI motherboard PCB. This manufacturing specification was defined as follows with the layers placed as specified before.

- 1 ounce Copper Foil (Top Layer)
- 0.063mm x 3 Prepreg
- 1 ounce Copper 0.2mm Thin Laminate (Power Plane 1/Signal Layer 1)
- 0.063mm x 3 Prepreg
- 1 ounce Copper 0.2mm Thin Laminate (Power Plane 2/Power Plane 3)
- 0.063mm x 3 Prepreg
- 1 ounce Copper 0.2mm Thin Laminate (Signal Layer 2/Power Plane 4)
- 0.063mm x 3 Prepreg
- 1 ounce Copper Foil (Bottom Layer)

This manufacturing specification also gave a nominal PCB thickness of 1.6mm. The copper layer thickness was also specified at 1 ounce of copper per square foot of PCB to give sufficient current carrying capacity to the PCB.

Figure 10 shows the top and bottom views of the manufactured RUI motherboard PCB. **Figure 11** shows the top and bottom views of the populated RUI motherboard PCB with the most important features clearly visible.

5.7 Conclusions

In this chapter we had a look at the hardware implementation of the RUI. We started by selecting the components for the RUI module and the RUI motherboard. We then took the concept design of each section and implemented it in a detailed hardware design. Some minor changes were made from the original concept designs. The detailed hardware designs were captured in schematic diagrams and then converted into PCBs while observing the constraints imposed by the user requirements. These PCBs were manufactured and the components placed. We can thus conclude that the hardware we had implemented has met the criteria of the system specification that was generated before.

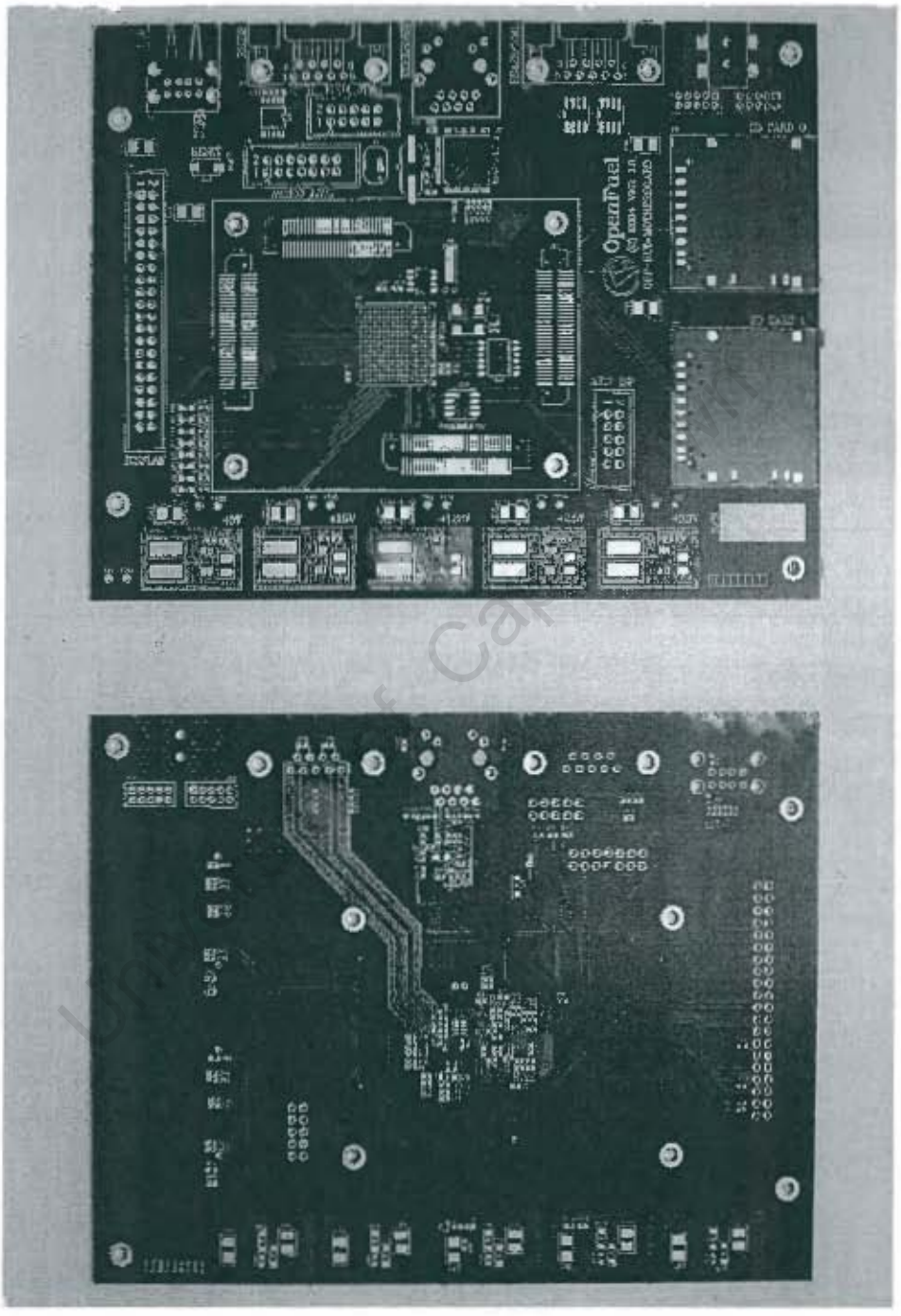

Figure 16 RUI Motherboard Top and Bottom Unpopulated PCB View

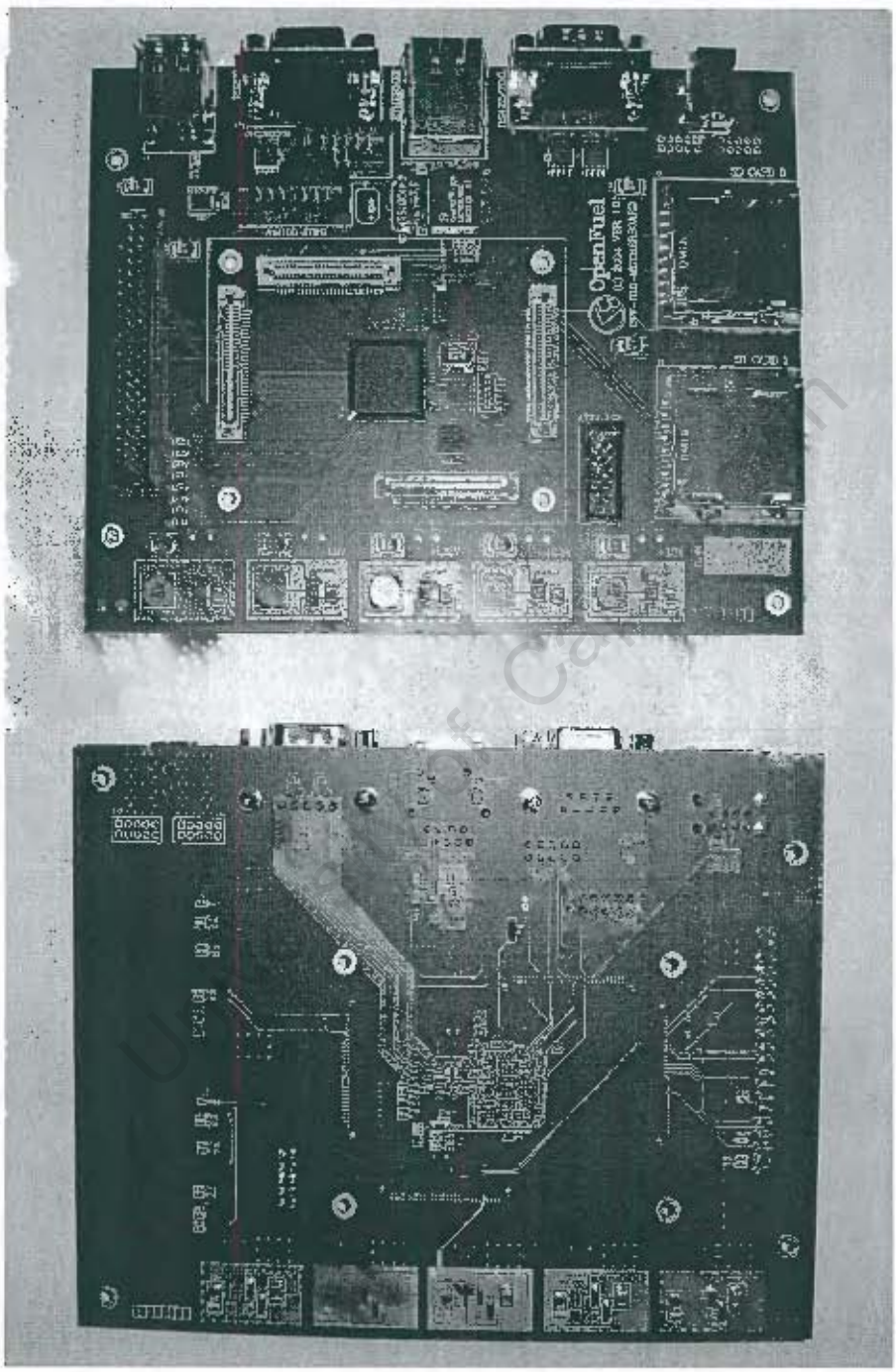

Figure 11 RUI Motherboard Top and Bottom Populated PCB View

In the next chapter we will look at the software and firmware implementation of the design project in detail, including the FPGA hardware configuration firmware, the RML interface FPGA firmware, the software bootloader and the radar application software.

Chapter 6

Software and Firmware Implementation

The hardware implementation of the RUI that was completed in the previous chapter, now allowed us to shift our focus to the software and firmware implementation for the RUI. We started with the FPGA hardware configuration firmware that linked the hardware signals in the RUI design. Next we discuss the RML interface FPGA firmware that has already been developed and used in the current ground penetrating radar system and adapted for use in the RUI. Finally, we looked at the bootloader that was used in the RUI and examined the status of the radar application software.

6.1 FPGA Hardware Configuration Firmware

The function of the FPGA hardware configuration firmware is to route the required hardware signals for the SoC processor, Flash non-volatile program memory, RTC, Ethernet transceiver and master reset switch. **Figure 12** and **Figure 13** shows the functional block diagrams of the FPGA hardware configuration firmware. The source code listing for the FPGA hardware configuration firmware is included in **Appendix C.1** for reference purposes. The firmware was implemented in VHDL.

From **Figure 12** and **Figure 13** we see that the signal from the master reset switch (*nRESET*), the RTC reset signal (*RTC:nRESET*) and the reset signal from the SoC processor JTAG connector (*JTAG:nRESET*) are connected to a 3-input logic AND gate and provides four output reset signals as well as the *Master Reset* signal, which is internally used in the FPGA. These signals go to the Ethernet interface (*ETH:nRESET*), SoC processor (*AUP:nRESET*), Flash non-volatile program memory (*FLASH:nRESET*) and one of the general purpose debug LEDs (*LED0*). The RTC reset signal (*RTC:nRESET*) and the reset signal from the SoC processor JTAG connector (*JTAG:nRESET*) are enabled by signals from the DIP switch (*SWITCH4* & *SWITCH5* respectively) through two OR gates. This function allows the user to enable or disable these two reset sources as required. The

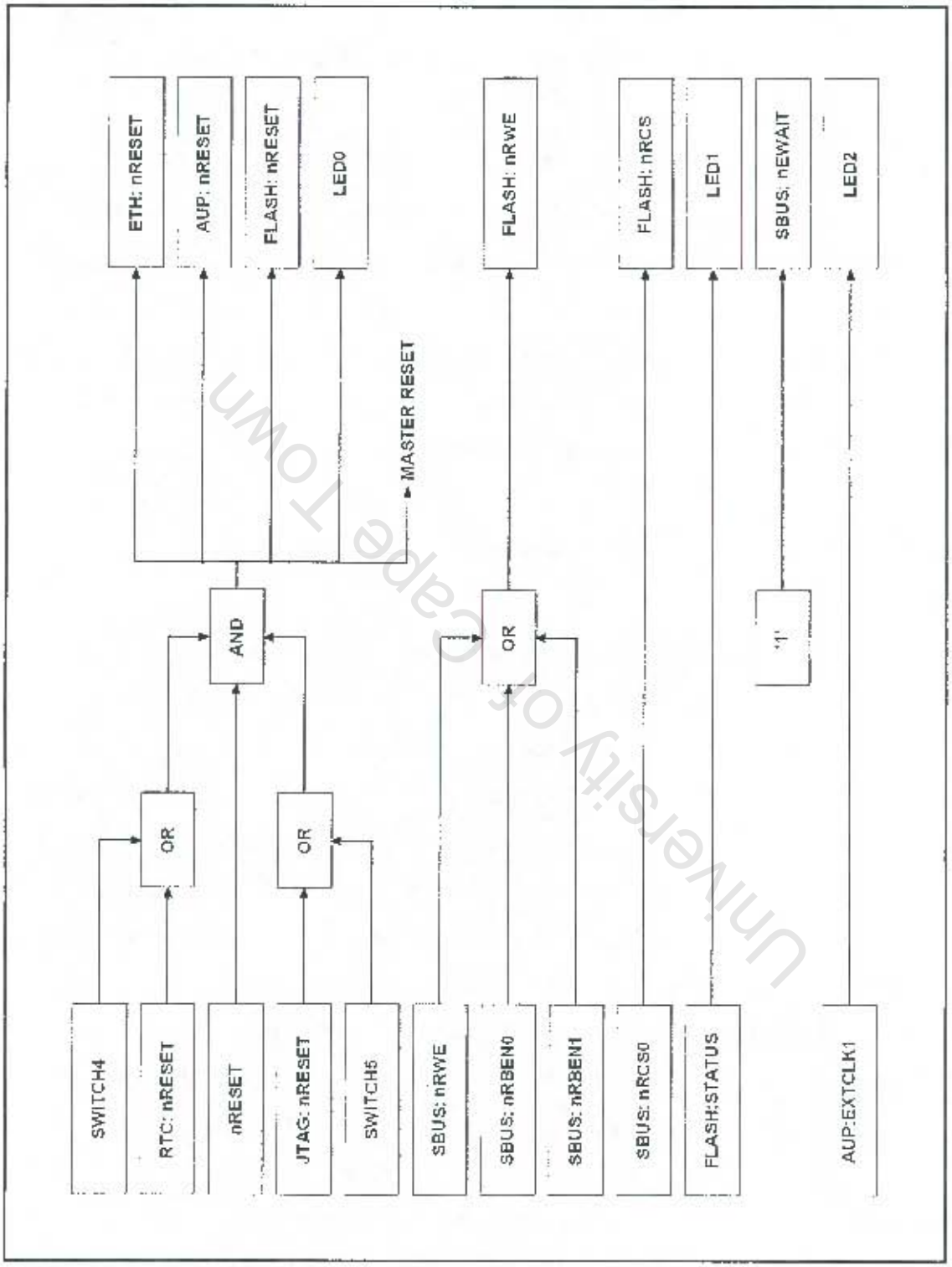

Figure 12 FPGA Hardware Configuration Firmware Block Diagram - Part 1 of 2

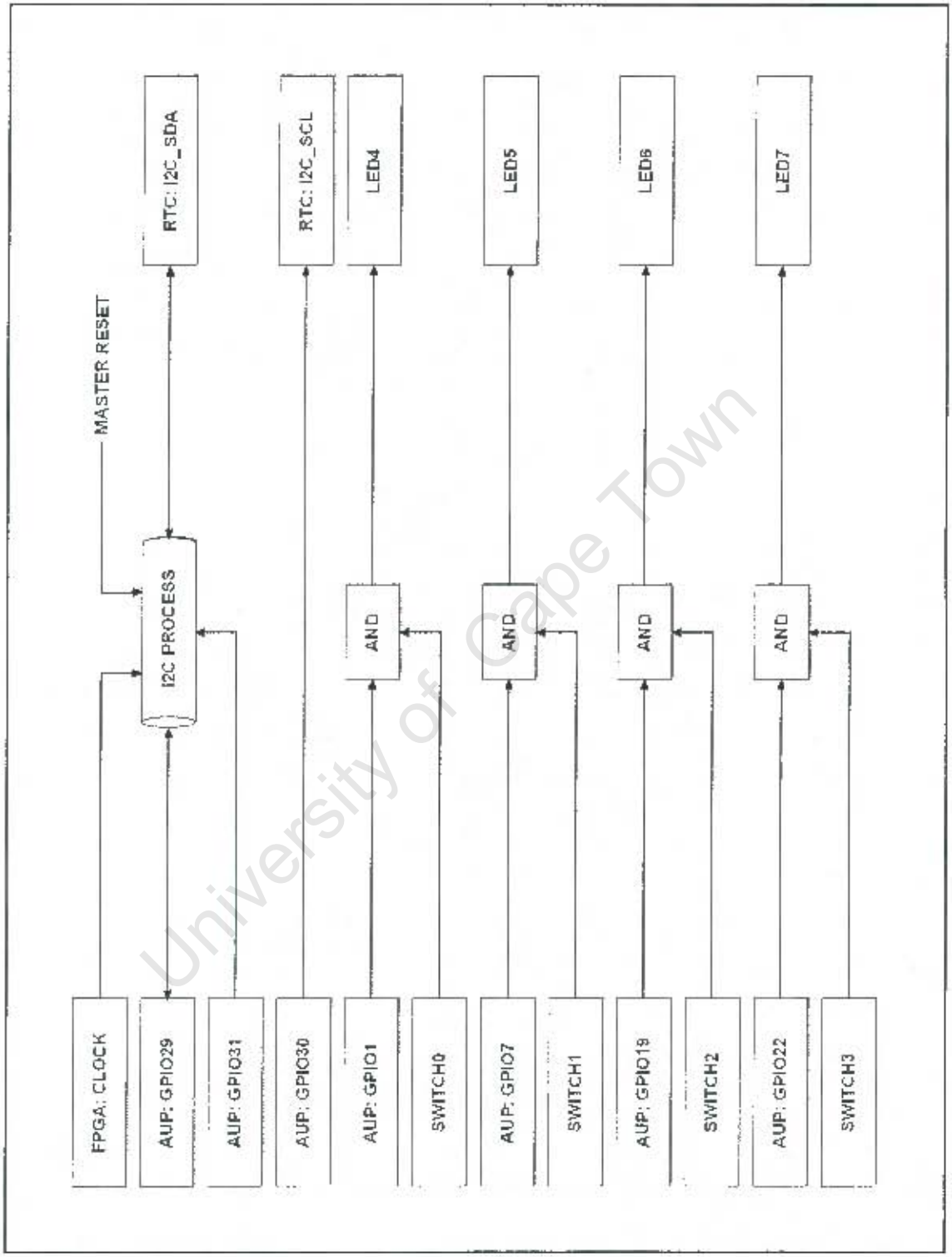

Figure 13 FPGA Hardware Configuration Firmware Block Diagram - Part 2 of 2

three input signals to the AND gate are all active low and thus the signal from the AND gate will be logic low if any of the input signals are logic low. This will ensure that the SoC processor, the Flash non-volatile program memory and the Ethernet interface will be reset if either the master reset switch is pressed, the RTC forces the reset signal low or the reset signal from the SoC processor JTAG connector is forced low. The LED is just a visual indication of the state of the reset signal. The RTC can force the reset signal low if the watchdog timer is enabled and overflows or when the power-on reset circuit is activated. The power-on reset signal will keep the system in reset until the power supply levels have reached a stable state or if the power supply level drops below a certain value it will force the reset signal low.

The next group of signals deals with more of the Flash non-volatile program memory control signals. The write enable signal (*SBUS:nRWE*), byte enable signal for data bits 0 to 7 (*SBUS:nRBEN0*) and byte enable signal for data bits 8 to 15 (*SBUS:nRBEN1*) of the static bus controller of the SoC processor are connected to a 3-input logic OR gate and provides the write enable signal to the Flash non-volatile program memory (*FLASH:nRWE*). The three input signals to the OR gate are all active low and thus the signal from the OR gate will be logic low only if all of the input signals are logic low. The function of this circuit is to enable the write function of the Flash non-volatile program memory only when it is required. The first chip select signal (*SBUS:nRCS0*) of the static bus controller is connected to the chip select signal (*FLASH:nRCS*) of the Flash non-volatile program memory. This chip select signal is active low, i.e. the Flash non-volatile program memory will only be enabled in the chip select signal is driven logic low by the SoC processor static bus controller. The function of this signal is to memory map the Flash non-volatile program memory to the memory address space of the SoC processor static bus controller. The status signal from the Flash non-volatile program memory (*FLASH:STATUS*) is connected to one of the general purpose debug LEDs (*LED1*). The user can configure, via the SoC processor application software, the function of the status signal to indicate what status that the status signal should indicate. The status signal could indicate the status of the Flash non-volatile program memory as far as programming, erasing and internal functions are concerned, i.e. the LED will turn on or flash briefly when

any of these functions are performed or remain turned on while these functions are being performed.

The static bus controller wait access signal (*SBUS:nEWAIT*) is not used by the RUI, but was included in the design for future applications. The function of the wait access signal is to stretch the bus access time when it is forced logic low. The signal is connected to a logic high to prevent it from floating and generating wait states in the static bus controller of the SoC processor.

One of the external clock signals from the SoC processor (*AUP: EXTCLK1*) is connected to one of the general purpose debug LEDs (*LED2*). This allows for the external clock signal generator in the SoC processor to be programmed to generate a slow clock signal (1Hz for example) that can be used as a visual heartbeat indicator for the SoC processor.

The RTC I^2C clock signal (*RTC:I2C_SCL*) and bi-directional data signal (*RTC:I2C_SDA*) are connected to two general purpose input/output signals (*AUP:GPIO29* & *AUP:GPIO30* respectively) on the SoC processor. The function of these signals is to provide access to the RTC and its internal functions. The direction of the bi-directional I^2C data signal is controlled via a process that is triggered by an input signal (*AUP:GPIO31*) from the SoC processor. This process is also triggered by the FPGA clock signal (*FPGA:CLOCK*) to make it synchronous as well as the *Master Reset* signal, which resets the state of the process. The SoC processor application software will implement the I^2C driver and necessary RTC access software.

Four of the debug LEDs (*LED4, LED5, LED6* & *LED7*) are connected to four general purpose input/output signals (*AUP:GPIO1, AUP:GPIO7, AUP:GPIO19* & *AUP:GPIO22* respectively) on the SoC processor. The function of these LEDs is to aid in the development and debugging of hardware and software on the RUI. These LEDs are also individually controlled via signals from the DIP switch (*SWITCH0, SWITCH1, SWITCH2* & *SWITCH3* respectively) through four AND gates. This function allows the user to enable or disable the LEDs as required.

The seven unused general purpose input/output signals (*AUP:GPIO14, AUP:GPIO15, AUP:GPIO201, AUP:GPIO202, AUP:GPIO203, AUP:GPIO204,*

AUP:GPIO206, AUP:GPIO208 & AUP:GPIO214 respectively) from the SoC processor are set as outputs and driven high. These signals may be used for future applications.

All the signals used in the FPGA hardware configuration firmware and its respective pin numbers on the FPGA are shown in **Table 10** below.

Table 10 FPGA Hardware Configuration Firmware Signal Definitions

Signal	Type	Description	Pin Number
AUP:EXTCLK1	Input	Au1100 External Clock (Heartbeat)	N8
AUP:GPIO1	Input	Au1100 Debug Input 0	P1
AUP:GPIO14	Input	Not Used	B12
AUP:GPIO15	Input	Not Used	B13
AUP:GPIO7	Input	Au1100 Debug Input 1	L1
AUP:GPIO19	Input	Au1100 Debug Input 2	E14
AUP:GPIO22	Input	Au1100 Debug Input 3	D14
AUP:GPIO29	Bidirectional	RTC I^2C Bus Bidirectional Data Signal to Au1100	A12
AUP:GPIO30	Input	RTC I^2C Bus Clock Signal from Au1100	E15
AUP:GPIO31	Input	RTC I^2C Bus Data Direction Control	C16
AUP:GPIO201	Input	Not Used	B5
AUP:GPIO202	Input	Not Used	A5
AUP:GPIO203	Input	Not Used	B4
AUP:GPIO204	Input	Not Used	L2
AUP:GPIO206	Input	Not Used	C4
AUP:GPIO208	Input	Not Used	D16
AUP:GPIO214	Input	Not Used	E16

Signal	Type	Description	Pin Number
AUP:nRESET	Output	Au1100 Reset	M2
ETH:nRESET	Output	Ethernet Interface Reset	H16
FLASH:nRESET	Output	Flash Memory Reset	D8
FLASH:nRCS	Output	Flash Memory Chip Select	C8
FLASH:nRWE	Output	Flash Memory Write Enable	N3
FLASH:STATUS	Input	Flash Memory Configurable Status Input	M3
FPGA:CLOCK	Input	32MHz FPGA Clock	R8
JTAG:nRESET	Input	Au1100 JTAG External Reset Input	A11
LED0	Output	Reset Debug LED	G1
LED1	Output	Flash Status Debug LED	F2
LED2	Output	Au1100 External Clock (Heartbeat) LED	F1
LED4	Output	Debug LED 0	E1
LED5	Output	Debug LED 1	D2
LED6	Output	Debug LED 2	D1
LED7	Output	Debug LED 3	C1
nRESET	Input	Master Reset Switch	C12
RTC:I2C_SCL	Output	RTC I^2C Bus Clock	G15
RTC:I2C_SDA	Bidirectional	RTC I^2C Bus Data	G16
RTC:nRESET	Input	RTC Reset Input	H15
SBUS:nEWAIT	Output	Au1100 Static Bus Controller Bus Wait Access	N1
SBUS:nRBEN0	Input	Au1100 Static Bus Controller Enable for Data Bits RD[7:0]	A9

Signal	Type	Description	Pin Number
SBUS:nRBEN1	Input	Au1100 Static Bus Controller Enable for Data Bits RD[15:8]	D9
SBUS:nRCS0	Input	Au1100 Static Bus Controller Chip Select 0	D10
SBUS:nRWE	Input	Au1100 Static Bus Controller Write Enable	E10
SWITCH0	Input	Debug LED 0 Enable	R5
SWITCH1	Input	Debug LED 1 Enable	T5
SWITCH2	Input	Debug LED 2 Enable	R6
SWITCH3	Input	Debug LED 3 Enable	T6
SWITCH4	Input	Enable RTC Reset	R7
SWITCH5	Input	Enable JTAG reset	T7

6.2 FPGA RML Interface Firmware

The RML interface firmware has already been developed for the current ground penetrating radar system by OpenFuel (Pty) Ltd.[27] It was thus only necessary to port the RML interface firmware to the RUI by changing the interface signal pin number assignments. Since the RML interface is a proprietary developed interface and a patent is still pending on its design [28], we will only present the signals that are used in interfacing the SoC processor with the RML interface firmware through the FPGA.

All the signals used in the RML interface firmware and its respective pin numbers on the FPGA are shown in **Table 11** below.

Table 11 FPGA RML Interface Firmware Signal Definitions

Signal	Type	Description	Pin Number
FPGA:CLOCK	Input	32MHz FPGA Clock	R8

Signal	Type	Description	Pin Number
LVDS:CLK_N	Output	RML Interface Clock Inverting	R9
LVDS:CLK_P	Output	RML Interface Clock Non-inverting	T9
LCDS:DATA_N	Bidirectional	RML Interface Data Inverting	R12
LVDS:DATA_P	Bidirectional	RML Interface Non-inverting	T12
LED3	Output	RML Link Active LED	E2
SBUS:nRBEN0	Input	Au1100 Static Bus Controller Enable for Data Bits RD[7:0]	A9
SBUS:RAD0	Input	Au1100 Static Bus Controller Address Bus Bit 0	C9
SBUS:RAD1	Input	Au1100 Static Bus Controller Address Bus Bit 1	A7
SBUS:RAD2	Input	Au1100 Static Bus Controller Address Bus Bit 2	A8
SBUS:RAD3	Input	Au1100 Static Bus Controller Address Bus Bit 3	A6
SBUS:RAD4	Input	Au1100 Static Bus Controller Address Bus Bit 4	B7
SBUS:RAD5	Input	Au1100 Static Bus Controller Address Bus Bit 5	B8
SBUS:RAD6	Input	Au1100 Static Bus Controller Address Bus Bit 6	D5
SBUS:RAD7	Input	Au1100 Static Bus Controller Address Bus Bit 7	D6
SBUS:RAD8	Input	Au1100 Static Bus Controller Address Bus Bit 8	B9

Signal	Type	Description	Pin Number
SBUS:RAD9	Input	Au1100 Static Bus Controller Address Bus Bit 9	C5
SBUS:RAD10	Input	Au1100 Static Bus Controller Address Bus Bit 10	D7
SBUS:RAD11	Input	Au1100 Static Bus Controller Address Bus Bit 11	C7
SBUS:RAD12	Input	Au1100 Static Bus Controller Address Bus Bit 12	B6
SBUS:nRCS1	Input	Au1100 Static Bus Controller Chip Select 1	A10
SBUS:RD0	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 0	J4
SBUS:RD1	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 1	K4
SBUS:RD2	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 2	K3
SBUS:RD3	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 3	K1
SBUS:RD4	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 4	K2
SBUS:RD5	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 5	L4
SBUS:RD6	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 6	L3
SBUS:RD7	Bidirectional	Au1100 Static Bus Controller Data Bus Bit 7	M1
SBUS:nROE	Input	Au1100 Static Bus Controller Output Enable	N2

Signal	Type	Description	Pin Number
SBUS:nRWE	Input	Au1100 Static Bus Controller Write Enable	E10

6.3 Bootloader Software

The bootloader that was used for this project was MicroMonitor.[29] The bootloader provides an environment onto which a developer can develop and deploy a variety of different types of applications. The bootloader can be ported to a variety of different hardware targets, because it is not inhibited by the use of an overlaying operating system or file system.

For this reason, the bootloader will only be used for initially testing the operation of the SoC processor. The complexity of using the USB interface, LCD interface, 10/100Mbps Ethernet interface and RML interface, will have to be handled by an operating system. Similarly, the storage of the radar data on the SD memory cards via the SD interfaces will have to be handled by the operating system's file system. As MicroMonitor does not allow for the use of a suitable overlaying operating system, a different bootloader will have to be used once a suitable operating system has been selected and ported to the RUI hardware platform.

The MicroMonitor bootloader was ported and compiled with a MIPS32 target GNU cross-platform toolchain that was built for the RUI hardware under Linux. The whole process that was followed to build the toolchain is documented in **Appendix D**. The detail of porting the bootloader software to a specific target platform is well documented in the reference.

6.4 Radar Application Software

The original radar application software for the ground penetrating radar system is written in Java and is thus easily portable between hardware and software platforms. It is outside

of the scope of this design project to deal with the radar application software or the software platform on which it would be used. The radar application software will be ported and a software platform developed for the RUI at a later stage as a separate project.

6.5 Conclusions

In this chapter we had a look at the software and firmware implementation of the RUI. We started with the design and implementation of the FPGA hardware configuration firmware that linked the hardware signals in the RUI design. The implementation of the RML interface FPGA firmware that has already been developed and used in the current ground penetrating radar system and adapted for use in the RUI was briefly discussed. We briefly introduced and examined the operation of the RUI bootloader and briefly discussed the status of the radar application software. In the next chapter we will look at the verification, testing and integration of the RUI hardware, firmware and software.

Chapter 7

Verification, Integration and Testing

Once the hardware implementation and the software and firmware implementation phase of the design project was completed we proceeded to the next phase of the design project. This was the verification, integration and testing phase of the design project. In this phase, we first verified and tested the hardware components and then integrated the hardware with the software and firmware. We concluded this phase by verifying the operation of the software and firmware with the hardware.

7.1 Verification, Integration and Testing Procedure

The following steps were followed in the verification, integration and testing procedure of the RUI hardware, software and firmware.

- General Hardware Verification and Testing of the RUI Module
- General Hardware Verification and Testing of the RUI Motherboard
- General Power Supply Verification and Testing of the RUI Motherboard
- Integration of the RUI Module with the RUI Motherboard
- Power Supply Verification and Testing with the RUI Module and RUI Motherboard (Minimum/Maximum External Supply Voltage Range and Full Load Testing)
- FPGA Verification and Testing
- RUI Hardware Configuration Firmware Verification and Testing
- SoC Processor Verification and Testing
- RUI Bootloader Verification and Testing
- 10/100Mbps Ethernet Interface Testing
- RS-422 HMI Interface Testing
- RS-232 Debug Interface Testing
- Real Time Clock and Power-On Reset Circuit Testing
- USB Interface Testing

- LCD Interface Testing
- Radar Module Link (RML) Interface Testing
- Non-Volatile Data Memory Interface Testing

7.2 Verification, Integration and Testing Procedure Results

7.2.1 General Hardware Verification and Testing

We started by doing a general hardware verification by taking the blank RUI module and RUI motherboard PCBs and measuring the resistance between the ground plane and the power supply planes on the PCBs in order to ensure that there was no short-circuit condition between them. Although a flying probe test was conducted by the PCB manufacturer after the PCBs were manufactured to verify the manufacturing process, errors could still have occurred in the design process. No short-circuit conditions were found to exist between the ground plane and the power supply planes.

7.2.2 Power Supply Verification and Testing

The external power supply was connected to the RUI motherboard and +7V applied. The current draw by the RUI motherboard was monitored to ensure there was no excessive current being drawn that would indicate a possible problem with the hardware. The output voltages of the power supply units on the RUI motherboard were measured at the test points and found to be within normal operating parameters. The peak-to-peak ripple voltages for the power supply units very also measured and noted.

Next, the RUI module was plugged into the RUI motherboard and the same procedure as before was followed. The output and ripple voltages of the power supplies were measured on the RUI module test points and noted and found to be within normal operating parameters.

Finally, the full load capacity of the power supplies was measured by connecting high-power resistors to the test points and drawing approximately 20% less than the

maximum designed load current of the power supplies. That equates to approximately 1.2A for each of the power supplies. This figure is less than the maximum output current capacity of the power supplies (ie 1.5A), but still far above the projected maximum output current required by the RUI, as previously calculated in **Table 3 in Chapter 5**. The output and ripple voltages were measured and noted and found to be within normal operating parameters. These measurements were done with only the RUI motherboard with nothing else connected to it. **Table 12** shows the test resistance values and the resultant test currents used to perform the full load capacity test of the power supplies.

Table 12 Power Supply Full Load Capacity Test Parameters

PSU Voltage	Required Resistance @ 1.2A	Test Resistance	Test Current	Resistor Power Dissipation
+1.22V	1.017Ω	1Ω	1.220A	1.488W
+1.5V	1.25Ω	1.2Ω	1.250A	1.875W
+2.5V	2.083Ω	2.1Ω	1.190A	2.976W
+3.3V	2.75Ω	2.77Ω	1.191A	3.931W
+5V	4.167Ω	4.2Ω	1.190A	5.952W

The whole process as outlined above was repeated with +12V applied from the external power supply. This was done as the system specification defined the external supply voltage to be +7V to −12V. **Table 13** shows the results of all the measurements that were taken and noted as described previously.

We can see that the output voltages of the power supplies under all external input voltage and load conditions remain relatively stable. The ripple voltages of the power supplies do increase with an increase in load current as expected, but the levels remain at acceptable levels.

We further also monitored the power supplies for excessive heat generation over a period of time, but it also remained at acceptable levels.

Table 13 Measured RUI Power Supply Output and Ripple Voltages						
	V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
RUI Motherboard Test Point Voltage	+7V	+1.226V	+1.507V	+2.514V	+3.313V	+5.02V
	+12V	+1.228V	+1.507V	+2.514V	+3.313V	+5.02V
RUI Motherboard Ripple Voltage	+7V	+15mV	+15mV	+17.5mV	+17.5mV	+15mV
	+12V	+22.5mV	+20mV	+20mV	+17.5mV	+17.5mV
RUI Module/ Motherboard Test Point Voltage	+7V	+1.226V	+1.507V	+2.513V	+3.312V	+5.02V
	+12V	+1.226V	+1.507V	+2.514V	+3.313V	+5.02V
RUI Module/ Motherboard Ripple Voltage	+7V	+32.5mV	+32.5mV	+22.5mV	+32.5mV	+22.5mV
	+12V	+32.5mV	+32.5mV	+25mV	+37.5mV	+17.5mV
Complete RUI Test Point Voltage	+7V	+1.226V	+1.507V	+2.513V	+3.312V	+5.02V
	+12V	+1.226V	+1.507V	+2.513V	+3.313V	+5.02V
Complete RUI Ripple Voltage	+7V	+37.5mV	+42.5mV	+27.5mV	+40mV	+50mV
	+12V	+37.5mV	+47.5mV	+30mV	+42.5mV	+45mV

	V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
Full Load Test Point Voltage	+7V	+1.2V	+1.48V	+2.47V	+3.28V	+4.98V
	+12V	+1.22V	+1.49V	+2.49V	+3.29V	+5.01V
Full Load Ripple Voltage	+7V	+47.5mV	+37.5mV	+67.5mV	+47.5mV	+141mV
	+12V	+47.5mV	+40mV	+65mV	+45mV	+141mV

The power supply data needed to be reduced further to express it in standardised terms. The effectiveness of the power supply filtering is indicated by the ripple factor of the power supply. The ripple factor (in %) in terms of the RMS ripple voltage (V_r) and the power supply output voltage (V_{dc}) is given by the formula shown below.

$$r = \frac{V_r}{V_{dc}} * 100\% \quad \ldots\ldots\ldots [30]$$

Table 14 RUI Power Supply Ripple Factors

V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
+7V	2.8%	1.79%	1.93%	1.02%	2%
+12V	2.75%	1.9%	1.85%	0.97%	1.99%

Table 14 shows the calculated ripple factors for the power supplies over the external supply voltage range of +7V to +12V as defined by the system specification. The ripple

factors are for the full load condition as this represents the worst case scenario for the design of the power supplies. The RMS ripple voltage values are taken as 0.707 times that of the peak-to-peak ripple voltage values.

The effectiveness of the power supply to produce a constant output voltage with a change in the input voltage to the power supply, under a no load condition, is defined as the line regulation of the power supply. The line regulation (in %) in terms of the change in output voltage (ΔV_{OUT}), output voltage (V_{OUT}) and change in input voltage (ΔV_{IN}) is given by the formula shown below.

$$Line_{REG} = \frac{(\Delta V_{OUT} / V_{OUT})}{\Delta V_{IN}} * 100\% \quad \ldots\ldots\ldots [31]$$

Table 15 RUI Power Supply Line Regulation					
V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
+7V to +12V	0.02%	0%	0%	0%	0%

Table 15 shows the calculated line regulation values for the power supplies over the external supply voltage range of +7V to +12V as defined by the system specification.

The effectiveness of the power supply to produce a constant output voltage across a varying load with a resultant change in the current through the load, is defined as the load regulation of the power supply. The load regulation (in %) in terms of the no load output voltage (V_{NL}) and full load output voltage (V_{FL}) is given by the formula shown below.

Table 16 shows the calculated load regulation values for the power supplies over the external supply voltage range of +7V to +12V as defined by the system specification and the full load condition as stated before.

$$Load_{REG} = \frac{(V_{NL} - V_{FL})}{V_{FL}} * 100\% \quad \text{..... [32]}$$

Table 16 RUI Power Supply Load Regulation					
V_{IN}	+1.22V	+1.5V	+2.5V	+3.3V	+5V
+7V	2.17%	1.82%	1.78%	1%	0.8%
+12V	0.66%	1.14%	0.96%	0.7%	0.2%

If we examine the data in **Table 14**, **Table 15**, and **Table 16** we see that the performance of the power supplies is very good, even taking into consideration that the power supplies are switching regulators, which are more efficient than a linear regulator, but does not have good performance figures when comparing ripple factors. The +1.22V power supply has the worst performance figures compared to the other power supplies and this is probably due to the fact that the power supply is operating very close to the minimum output voltage, being +1.2V, of that specific switching regulator. Nevertheless, these figures are for a load condition in excess of what the system will normally operate at and is thus not a major concern.

7.2.3 Integration of the RUI Module with the RUI Motherboard

During the power supply testing phase of the verification, integration and testing phase of the design project, the RUI module was integrated with the RUI motherboard.

Figure 14 shows the top view of the integrated RUI module and RUI motherboard after the general hardware and power supply verification and testing was done, to illustrate how the two parts of the RUI integrate.

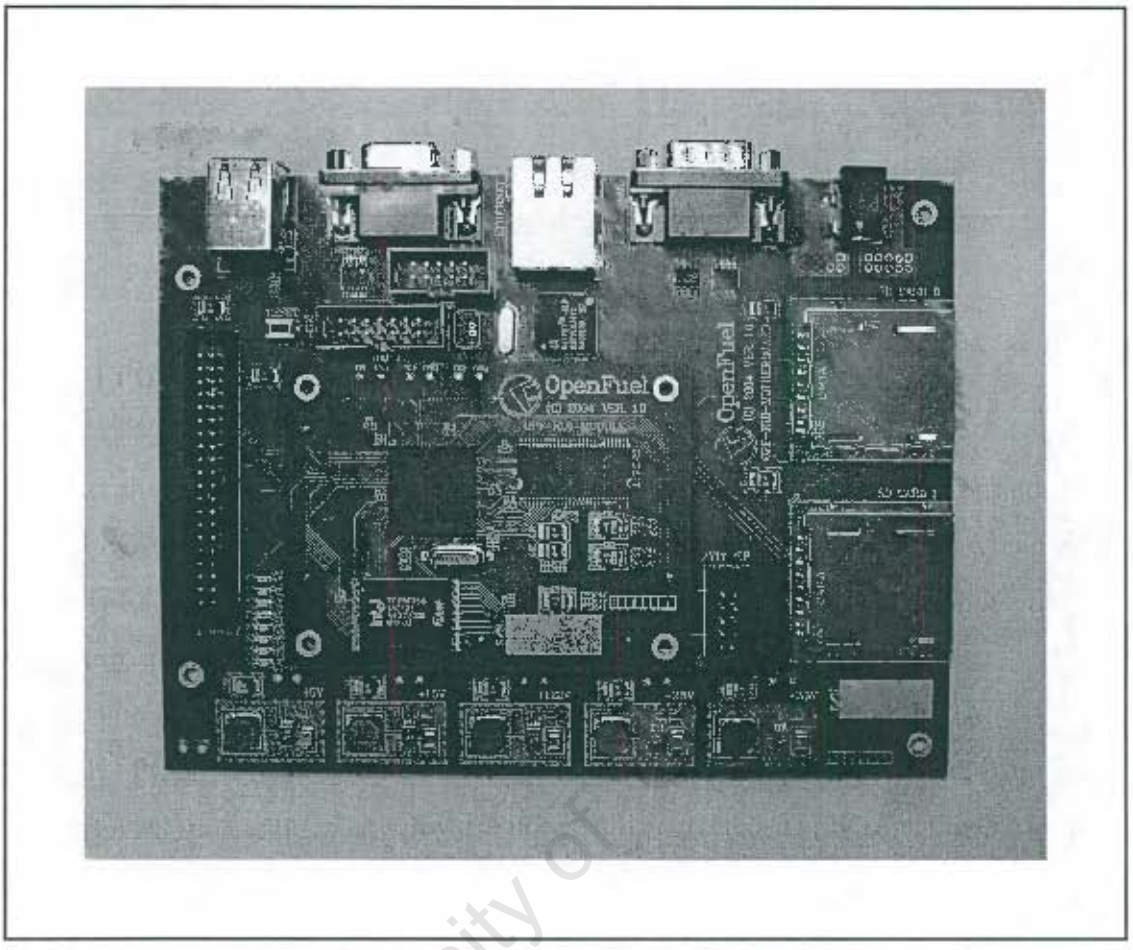

Figure 14 Integrated RUI Module and Motherboard

7.2.4 FPGA Verification and Testing

The next phase of the RUI hardware verification and testing was the verification and testing of the FPGA. The FPGA was packaged in an 1.0mm pitch BGA package, which meant that it was a highly technical process to place it on the PCB. For this reason we first had to perform an initial test to verify whether the FPGA was place correctly.

The FPGA verification and testing were done by scanning the FPGA JTAG port using an FPGA JTAG programmer and the iMPACT programming software from Xilinx (http://www.xilinx.com). The software successfully detected the FPGA in the JTAG chain, which meant that the FPGA was receiving the correct supply voltages and could now be

configured.

The FPGA configuration PROM was configured with the FPGA hardware configuration firmware, as discussed in **Chapter 6**, using an ATDH2225 in-system programming cable and AT17 CPS software from Atmel (http://www.atmel.com).

The test LEDs on the RUI motherboard were enabled with the DIP switch and the LEDs switched on as expected. This was an indication that the FPGA had been successfully configured. At this stage we concluded that the FPGA was operating correctly.

7.2.5 SoC Processor Verification and Testing

The next phase of the RUI hardware verification and testing was the verification and testing of the Au1100 SoC processor. The Au1100 SoC processor was packaged in a 0.8mm pitch BGA package, which meant that it was a highly technical process to place it on the PCB. For this reason we first had to perform an initial test to verify whether the Au1100 SoC processor was place correctly, before we could attempt to download any software to the system for execution.

The verification and testing were done by scanning the Au1100 SoC processor JTAG port with a Wiggler programmer and the OCDemon Flash Memory Programmer software from Macraigor Systems LLC (http://www.macraigor.com). The software successfully detected the Au1100 SoC processor and Flash memory and it was possible to read the contents of the internal registers in the Au1100 SoC processor and read and write values to the Flash memory. This indicated that the Au1100 SoC processor and Flash memory was operating correctly.

7.2.6 RUI Bootloader Verification and Testing

The next phase of the RUI hardware verification and testing was the integration and then verification and testing of the bootloader with the RUI hardware. The following procedure was followed to integrated the bootloader with the RUI hardware.

- Download the bootloader to the SDRAM on the RUI module.
- Run the bootloader form the SDRAM on the RUI module across the RS-232 debug port on the Au1100 SoC processor.
- Download the Flash image of the bootloader to the RUI hardware across the RS-232 debug port on the Au1100 Soc processor.
- Program the Flash image of the bootloader into the Flash memory.

This procedure was done by using the Au1100 SoC processor JTAG port with a Wiggler programmer and the OCD Commander software from Macraigor Systems LLC (http://www.macraigor.com).

The procedure discussed in the previous paragraph is technically more complex than alluded to in the discussion. For this reason a detailed explanation of how to integrated the bootloader with the RUI hardware is included in **Appendix C.2**. In **Appendix C.3** the macro that was used to set up the Au1100 SoC processor, as required by the procedure outlined in **Appendix C.2**, is shown.

7.2.7 10/100Mbps Ethernet Interface Testing

The 10/100Mbps Ethernet interface was not tested past testing the digital control lines, as the software effort required to implement the required device driver, TCP/IP stack and port a suitable operating system to the RUI hardware platform was outside of the scope of this design project.

7.2.8 RS-422 HMI Interface Testing

The basic hardware functionality of the RS-422 HMI interface was tested and found to be operating correctly. The RS-422 HMI interface can be used to communicate with the current ground penetrating radar system, but it requires that the radar application software and a suitable operating system be ported to the RUI hardware platform, which is outside of the scope of this design project.

7.2.9 RS-232 Debug Interface Testing

The RS-232 debug interface was tested with the bootloader, as the bootloader operated via this interface, and found to be operating correctly.

7.2.10 Real Time Clock and Power-On Reset Circuit Testing

The basic functionality of the Real Time Clock (RTC) was tested. The RTC also provide a power-on reset signal, 250ms after the supply voltage reaches a predetermined value, to the rest of the RUI hardware via the FPGA. This power-on reset circuit was also tested and found to be functioning correctly.

7.2.11 USB Interface Testing

The integrated USB interface of the Au1100 SoC processor was not tested, as the software effort required to implement the required USB device driver and port a suitable operating system to the RUI hardware platform was outside of the scope of this design project.

7.2.12 LCD Interface Testing

The integrated LCD interface of the Au1100 SoC processor was not tested past testing the digital control lines, as the software effort required to implement the required LCD device driver and port a suitable operating system to the RUI hardware platform was outside of the scope of this design project.

7.2.13 Radar Module Link (RML) Interface Testing

The radar module link (RML) interface was not tested past testing the physical interface lines, as it required the radar application software and a suitable operating system to be ported to the RUI hardware platform. The RML firmware has been used before in the

current ground penetrating radar system, so its operation has been verified.

7.2.14 Non-Volatile Data Memory Interface Testing

The basic hardware functionality of the two non-volatile data memory interfaces was tested. The software effort required to implement the required file system and port a suitable operating system to the RUI hardware platform was outside of the scope of this design project.

7.3 Verification, Integration and Testing Procedure Summary

The results of the verification, integration and testing procedure are summarised in **Table 17** below.

Table 17 Verification, Integration and Testing Procedure Summary

Procedural Step	Test Status	Result/Reason
General Hardware Verification and Testing of the RUI Module	Yes	Pass
General Hardware Verification and Testing of the RUI Motherboard	Yes	Pass
General Power Supply Verification and Testing of the RUI Motherboard	Yes	Pass
Integration of the RUI Module with the RUI Motherboard	Yes	Pass
Power Supply Verification and Testing with the RUI Module and RUI Motherboard (Minimum/Maximum External Supply Voltage Range and Full Load Testing)	Yes	Pass
FPGA Verification and Testing	Yes	Pass

Procedural Step	Test Status	Result/Reason
RUI Hardware Configuration Firmware Verification and Testing	Yes	Pass
SoC Processor Verification and Testing	Yes	Pass
RUI Bootloader Verification and Testing	Yes	Pass
10/100Mbps Ethernet Interface Testing	Partial	Requires Device Driver, TCP/IP Stack & OS
RS-422 HMI Interface Testing	Yes	Pass
RS-232 Debug Interface Testing	Yes	Pass
Real Time Clock and Power-On Reset Circuit Testing	Yes	Pass
USB Interface Testing	No	Requires Device Driver & OS
LCD Interface Testing	Partial	Requires Device Driver & OS
Radar Module Link (RML) Interface Testing	Partial	Requires Radar Application Software
Non-Volatile Data Memory Interface Testing	Yes	Pass/Requires Device Driver & OS for File System

7.4 Conclusions

In this chapter we had a look at the verification, testing and integration of the hardware, software and firmware of the RUI. We started with the integration and testing of the hardware, followed by the integration and testing of the firmware. We concluded this phase of the design project with the integration of the software with the hardware and firmware. In the next chapter we will look at what conclusions can be drawn form this design project. We will also be looking at further studies that can be undertaken based on the work completed and the knowledge gained from this design project.

Chapter 8

Conclusions and Further Studies

With the objectives of the design project successfully achieved, we will conclude by discussing what conclusions can be drawn form this design project. We will also present and discuss further studies that can be undertaken based on the work completed and the knowledge gained from this design project.

8.1 Conclusions

The successful achievement of the objectives of this design project allows us to draw some conclusions based on what we have experience and the knowledge we have gained in achieving these objectives. We will also show that the original aim of the design project has been met. The conclusions we have drawn and a brief discussion of each conclusion will follow next.

- In reviewing the existing ground penetrating radar system, we concluded that its modular design is optimal for implementation in a surface-based ground penetrating radar system.
- In reviewing the existing commercially available ground penetrating radar systems, we concluded that the integration of the HMI and the radar electronics is not the most common design implementation, but from an engineering point of view it is the most optimal.
- In generating the system specification for the radar user interface, we concluded the detail that needed to be implemented in the design of the RUI.
- In the identification of a suitable technology for the implementation of the radar user interface hardware, we concluded that SoC processor technology was the most suitable technology to use.
- In the design and implementation phase of the radar user interface hardware, we concluded that we had met the system specification for the hardware that was

generated before by selecting the correct components and paying a great deal of attention to the detail of the design and the design practices used.

- In the design and implementation phase of the radar user interface firmware, we concluded that we had met the system specification for the hardware that was generated before by correctly designing the configuration firmware that linked the hardware signals in the RUI design. We further concluded that the RML interface firmware that was only briefly presented, due to a pending patent application, only needed to be ported to the RUI as it already had been designed and used before in the current ground penetrating radar system.
- In the design and implementation phase of the radar user interface software, we concluded that a less complex and easily portable bootloader was most suitable for the initial testing of the basic functionality of the RUI hardware. We further concluded that the radar application software had to be ported and a software platform developed for the RUI as a separate project at a later stage, as it was outside of the scope of this design project to do so.
- In the verification, integration and testing phase of the radar user interface hardware, software and firmware, we concluded that the basic operation of the RUI had been verified. We further concluded that a suitable bootloader and operating system had to be ported to the RUI to allow for the development of the required drivers for the RUI hardware that would be needed by the radar application software.

From these conclusions we can conclude that the design project has been successful and all the objectives of the design project have been achieved.

8.2 Further Studies

As the design project has been concluded successfully we can finally look at further studies that can be undertaken based on the work completed and the knowledge gained from this design project.

In order to produce a commercially viable surface-based ground penetrating radar

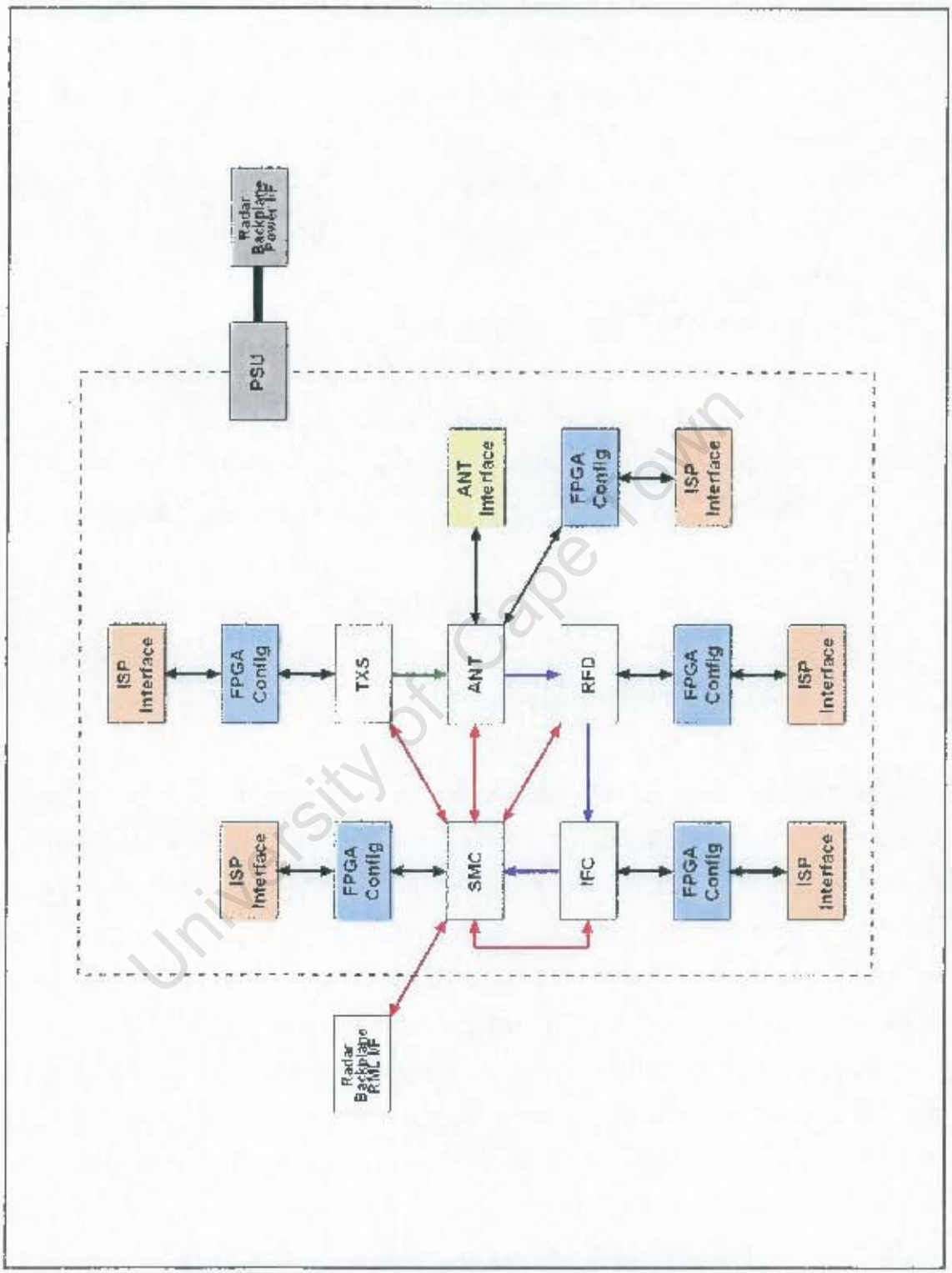

Figure 15 RUI Radar Electronics Backplane Concept Design

system, two phases of the design process still needs to be completed. The first phase would involve the implementation of the ground penetrating radar system application software on the RUI. This requires a number of steps to be completed as outlined below.

- Identify a suitable operating system for use with the Au1100 SoC processor; most likely this would be Linux.
- Identify a suitable bootloader that is compatible with both the Au1100 SoC processor and the operating system; most likely this would be the U-Boot bootloader.
- Port the selected bootloader to the RUI hardware platform.
- Port the selected operating system to the RUI hardware platform.
- Develop the hardware drivers for the RUI hardware platform that is required by the current ground penetrating radar system application software.
- Port the current ground penetrating radar system application software to the RUI hardware platform.
- Verify and test the operation of the ground penetrating radar system application software.

The second phase of the design process would involve the hardware design and implementation of the RUI radar electronics backplane. **Figure 15** shows a possible concept design of the RUI radar electronics backplane that could be implemented in practice. It is based on the current ground penetrating radar electronics that had been described before. The RUI would interface to the RUI radar electronics backplane through the RML interface to the SMC module. In this specific implementation the CTL module would not be used and its functionality would be implemented in the ground penetrating radar system application software. The antenna and the RF part of the ANT module would be integrated into an external unit. This antenna assembly will interface to and be controlled by the RUI through the ANT interface via the ANT module and an RML interface. Power would also be supplied to the antenna assembly through the ANT interface. The power supplies for the RUI radar electronics backplane would also be implemented on the RUI radar electronics backplane.

This design process only presents one possible concept design for the RUI radar

electronics backplane. Its function is to illustrate what could possibly be achieved in the near future based on the design of the current ground penetrating radar system and current RUI design. As technology is advancing at an ever increasing pace, one has to look at other possible implementations and innovations in the ground penetrating radar field when considering further studies based on this design project.

References

[1] Geophysical Survey Systems, Inc.: "TerraSIRch SIR-3000", Product Brochure on Website, http://www.geophysical.com, July 2004.

[2] Collander, P. & Laukkala, P.: "System in Package versus System on Chip", Nokia Networks & Nokia Research Centre, Espoo & Helsinki, Finland, 2000.

[3] Linden, G. & Somaya, D.: "System-on-a-Chip Integration in the Semiconductor Industry: Industry Structure and Firm Strategies", Department of Economics & R.H. Smith School of Business, University of California at Berkeley & University of Maryland, Berkeley & College Park, USA, p. 44, 2000.

[4] LinuxDevices.com: "Embedded Processor and System-on-Chip Quick Reference Guide", Website, http://www.linuxdevices.com/articles/AT4313418436.html, July 2004.

[5] AMD: "AMD Alchemy™ Solutions Au1100™ Processor Data Book", Datasheet, Publication ID: 30362B, http://www.amd.com, June 2003.

[6] Intel Corporation: "Intel StrataFlash® Memory (J3)", Datasheet, Order Number: 290667-017, http://www.intel.com, January 2004.

[7] Micron Technology, Inc.: "256Mb: x16 Mobile SDRAM - 4 Meg x 16 x 4 Banks", Datasheet, http://www.micron.com, April 2003.

[8] Xilinx: "Virtex™-II Platform FPGAs", Datasheet, DS031, http://www.xilinx.com, September 2002.

[9] Atmel Corporation: "FPGA Configuration EEPROM Memory - AT17LV",

Datasheet, http://www.atmel.com, June 2003.

[10] AMD: "Am79C874 NetPHY™-1LP Low Power 10/100-TX/FX Ethernet Transceiver", Datasheet, Publication #: 22235, http://www.amd.com, April 2001.

[11] Sipex Corporation: "SP3481/SP3485 +3.3V Low Power Half-Duplex RS-485 Transceivers with 10Mbps Data Rate", Datasheet, http://www.sipex.com, October 2002.

[12] Sipex Corporation: "SP3222E/3232E True +3.0V to +5.5V RS-232 Transceivers", Datasheet, http://www.sipex.com, November 2002.

[13] Xicor, Inc.: "X1227 Real Time Clock/Calendar/CPU Supervisor with EEPROM", Datasheet, Revision 1.1.18, http://www.xicor.com, August 2002.

[14] Linear Technology Corporation: "Monolithic 1.5A, 1.25MHz Step-Down Switching Regulators", Datasheet, http://www.linear.com, March 2002.

[15] Maxim Integrated Products: "Linear Regulators in Portable Applications", Application Note, AN751, http://www.maxim-ic.com, p. 2, May 2001.

[16] Linear Technology Corporation: "Monolithic 1.5A, 1.25MHz Step-Down Switching Regulators", Datasheet, http://www.linear.com, p. 7, March 2002.

[17] Linear Technology Corporation: "Monolithic 1.5A, 1.25MHz Step-Down Switching Regulators", Datasheet, http://www.linear.com, p. 8, March 2002.

[18] Linear Technology Corporation: "Monolithic 1.5A, 1.25MHz Step-Down Switching Regulators", Datasheet, http://www.linear.com, p. 9, March 2002.

[19] Linear Technology Corporation: "Monolithic 1.5A, 1.25MHz Step-Down Switching Regulators", Datasheet, http://www.linear.com, p. 9, March 2002.

[20] Linear Technology Corporation: "Monolithic 1.5A, 1.25MHz Step-Down Switching Regulators", Datasheet, http://www.linear.com, p. 7, March 2002.

[21] National Semiconductor: "LVDS Owner's Manual: A General Design Guide for National's Low Voltage Differential Signalling (LVDS) and Bus LVDS Products", Technical Manual, 2nd Edition, Revision 2.0, http://www.national.com, pp. 25-26, 2000.

[22] Johnson, H. & Graham, M.: "High-Speed Digital Design: A Handbook of Black Magic", First Edition, Prentice-Hall PTR, New Jersey, USA, pp. 191-199, 1993.

[23] Johnson, H. & Graham, M.: "High-Speed Digital Design: A Handbook of Black Magic", First Edition, Prentice-Hall PTR, New Jersey, USA, pp. 199-201, 1993.

[24] Gisin, F. & Pantić-Tanner, Z.: "Minimizing EMI Caused by Radially Propagating Waves Inside High Speed Digital Logic PCBs", Facta Universitatis (NIŠ) Series: Electronics and Energetics, Vol. 14, No. 2, pp. 205-223, August 2001.

[25] Ritchey, L.W.: "A Survey and Tutorial of Dielectric Materials used in the Manufacture of Printed Circuit Boards", Speeding Edge, Circuitree Magazine, November 1999.

[26] Micrel, Inc.: "General PCB Design and Layout Guidelines - Micrel 10/100 Switches and PHYs", Application Note, AN-111, http://www.micrel.com, June 2004.

[27] van Schaik, C.F.: "GTI-OAR-SS-0017: Obstacle Avoidance Radar Module Link

Standard Specification", Standards Specification, Version 1.0, OpenFuel (Pty) Ltd, Cape Town, South Africa, January 2003.

[28] Wilson-Langman, A., et al.: "Obstacle Detection System for Underground Operations", Patent Application, United States Patent Application #: 20050061547, 24 March 2005.

[29] Sutter, E.: "Embedded Systems Firmware Demystified", First Edition, CMP Books, Kansas, USA, 2002.

[30] Floyd, T.L.: "Electronic Devices", Third Edition, Macmillan Publishing Company (Merrill), New York, USA, pp. 52-53, 1992.

[31] Floyd, T.L.: "Electronic Devices", Third Edition, Macmillan Publishing Company (Merrill), New York, USA, pp. 806-807, 1992.

[32] Floyd, T.L.: "Electronic Devices", Third Edition, Macmillan Publishing Company (Merrill), New York, USA, pp. 807-808, 1992.

Appendix A

Schematic Diagrams and Parts Lists

A.1 RUI Module

The schematic diagrams and parts list of the RUI Module is shown in this appendix.

- **Figure 16** RUI Module Schematic Diagram - Sheet 1 of 8
- **Figure 17** RUI Module Schematic Diagram - Sheet 2 of 8
- **Figure 18** RUI Module Schematic Diagram - Sheet 3 of 8
- **Figure 19** RUI Module Schematic Diagram - Sheet 4 of 8
- **Figure 20** RUI Module Schematic Diagram - Sheet 5 of 8
- **Figure 21** RUI Module Schematic Diagram - Sheet 6 of 8
- **Figure 22** RUI Module Schematic Diagram - Sheet 7 of 8
- **Figure 23** RUI Module Schematic Diagram - Sheet 8 of 8
- **Table 18** RUI Module Parts List

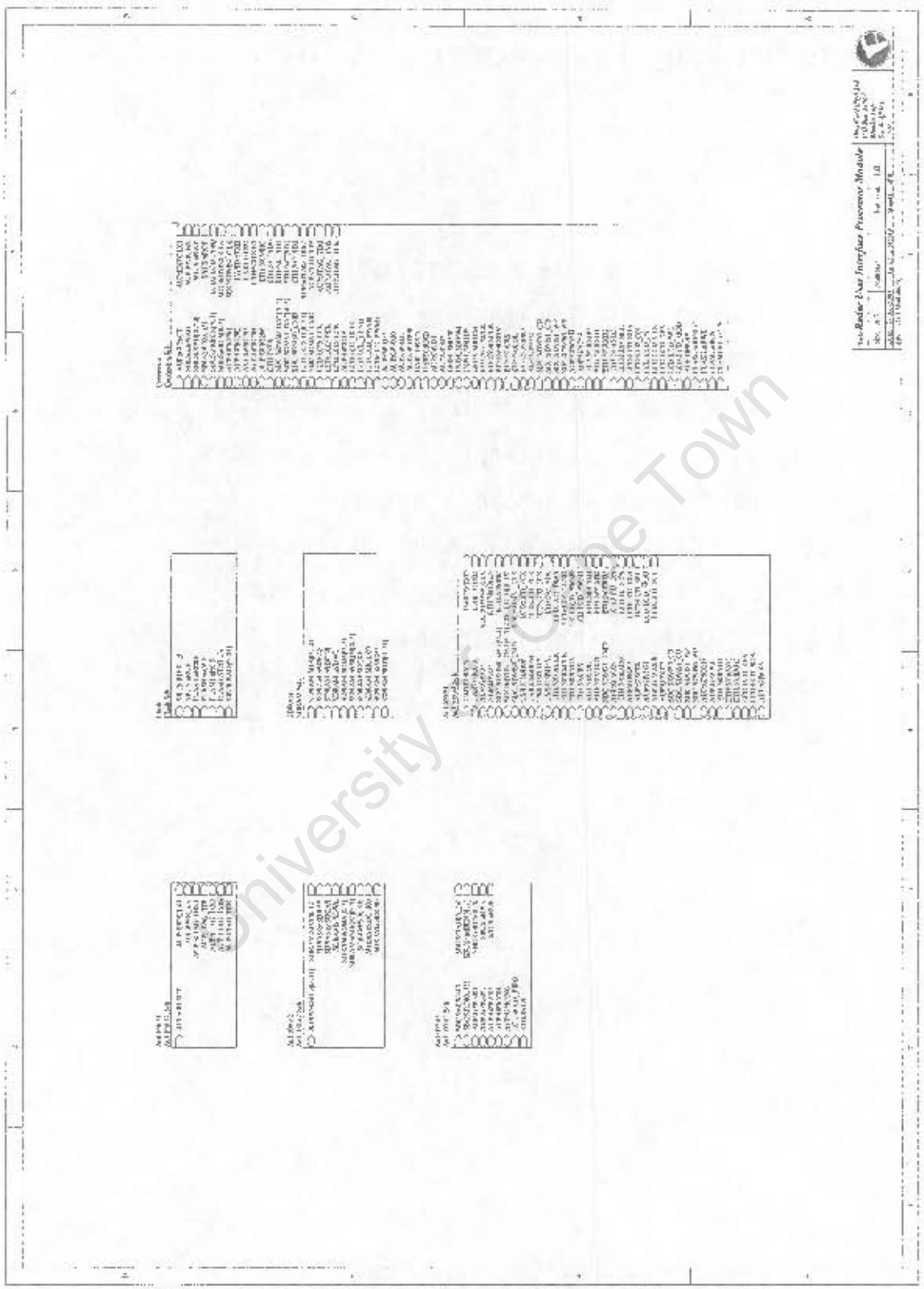

Figure 16 RUI Module Schematic Diagram - Sheet 1 of 8

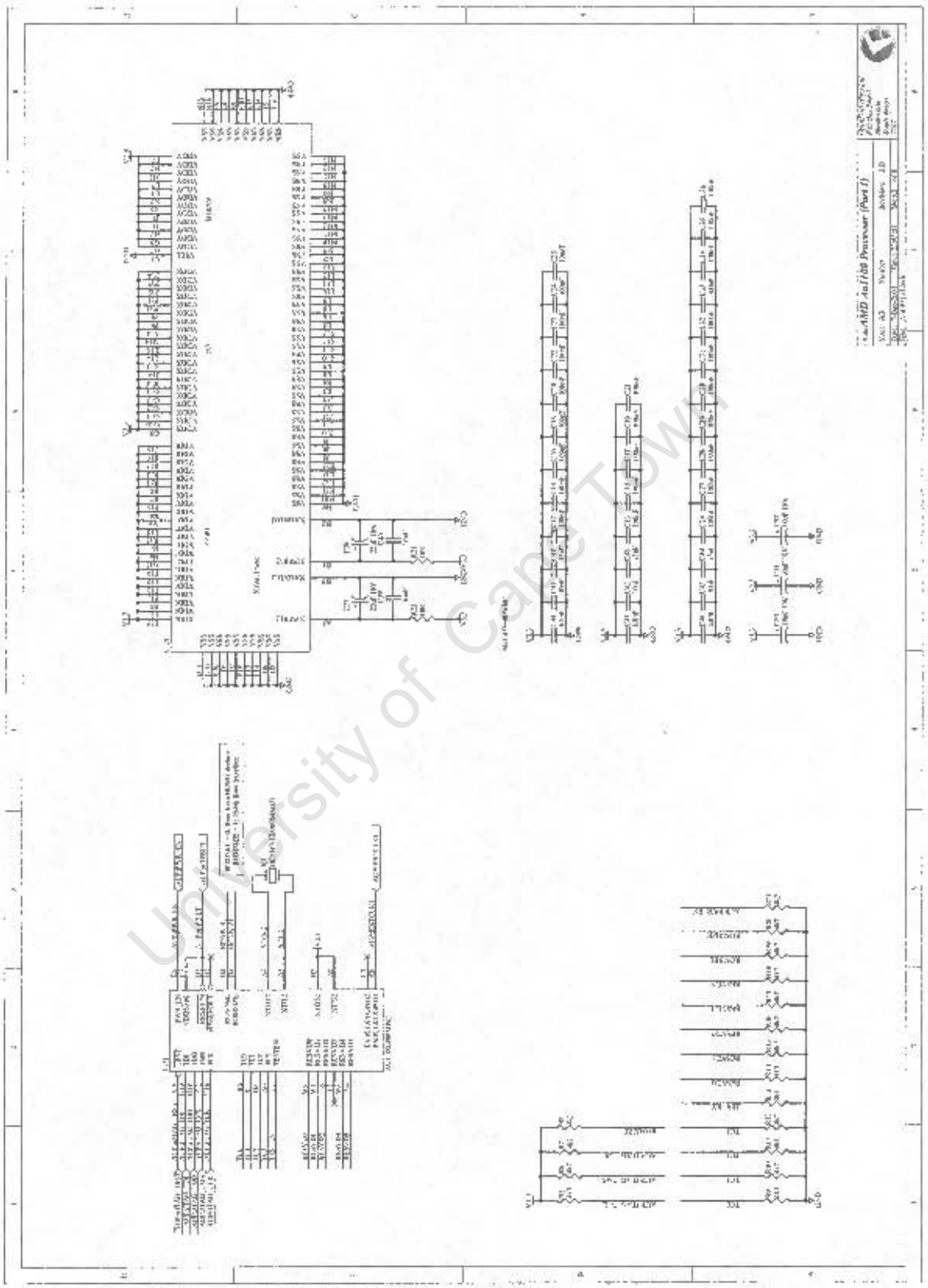

Figure 17 RUI Module Schematic Diagram - Sheet 2 of 8

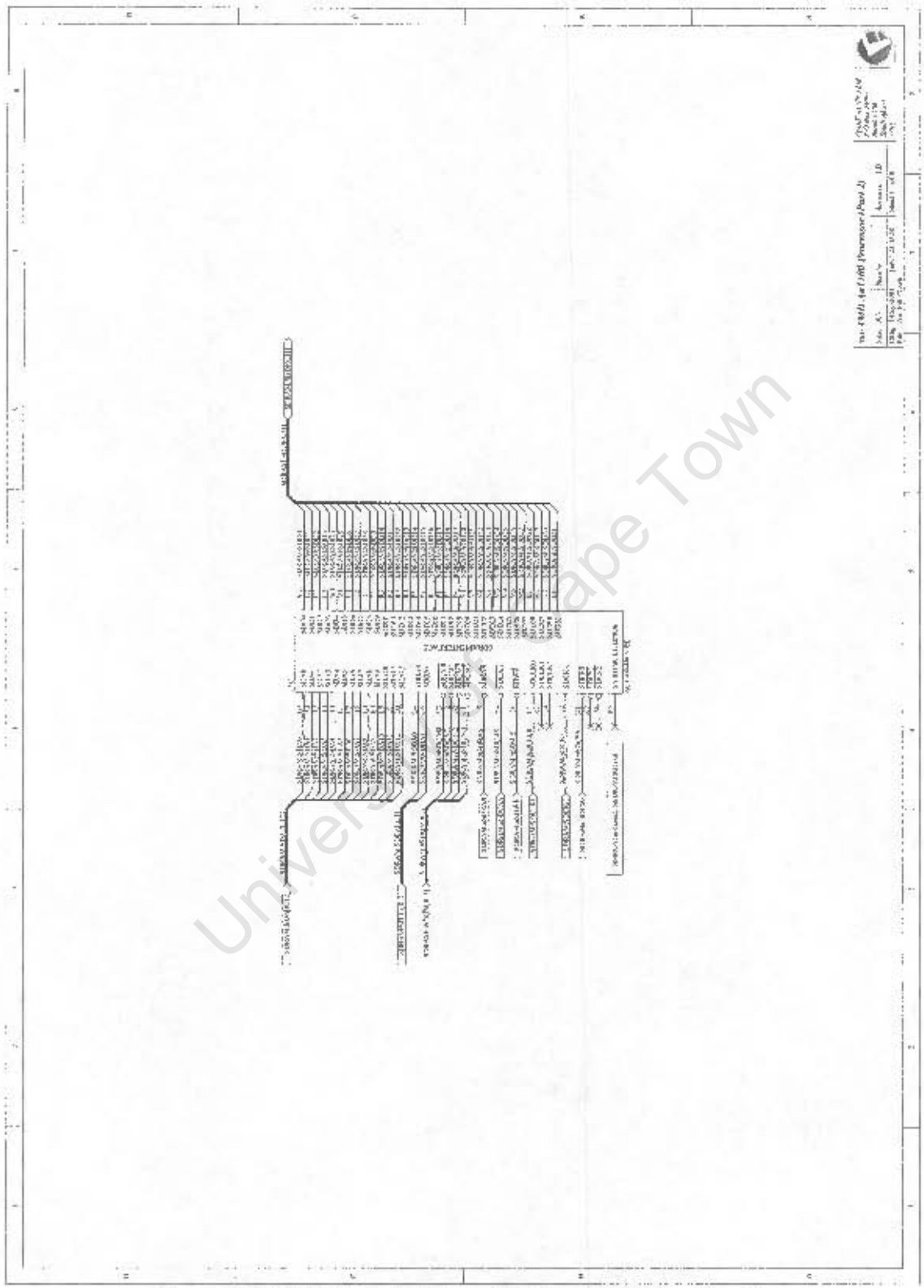

Figure 18 RUI Module Schematic Diagram - Sheet 3 of 8

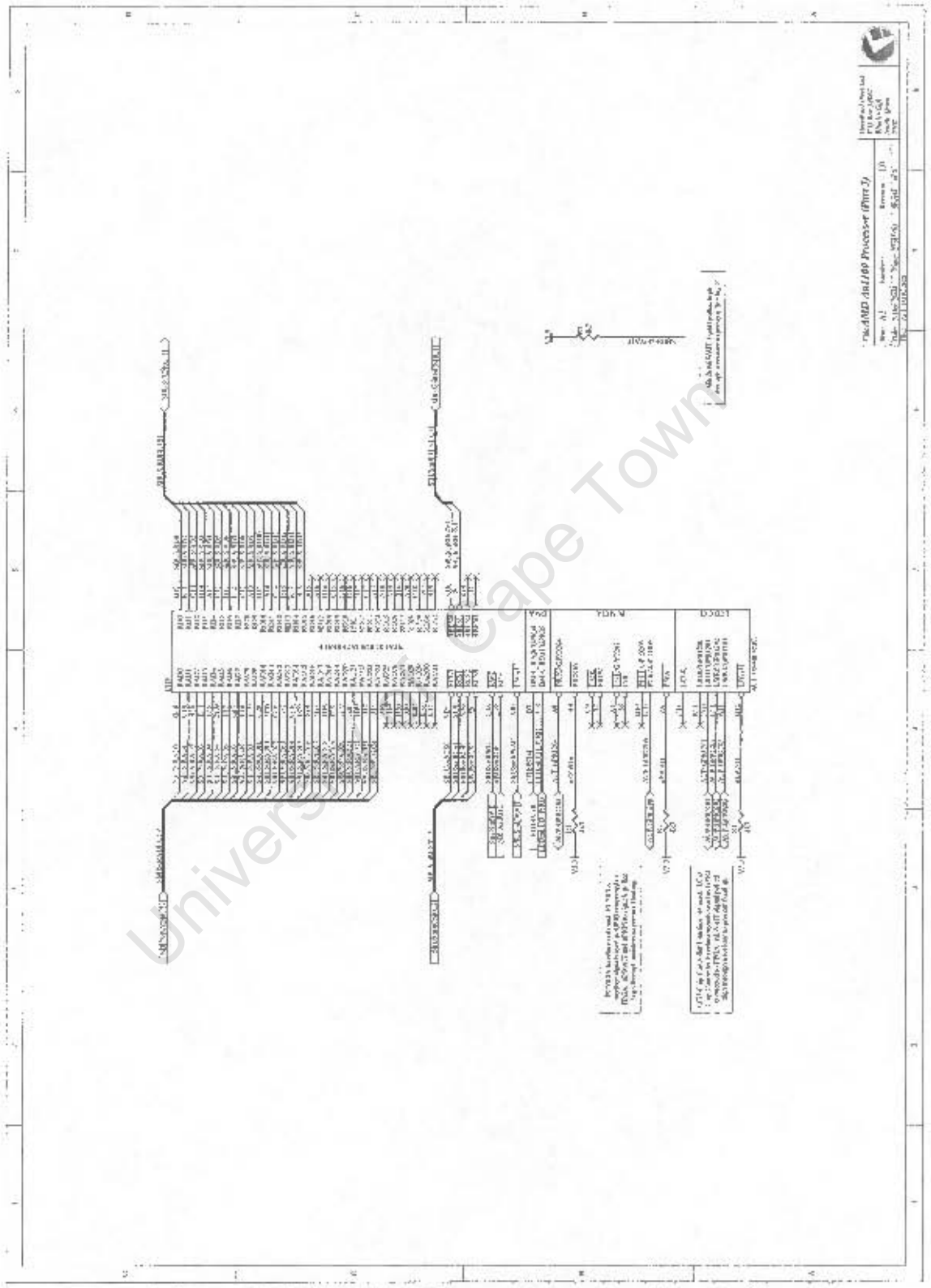

Figure 19 RUI Module Schematic Diagram - Sheet 4 of 8

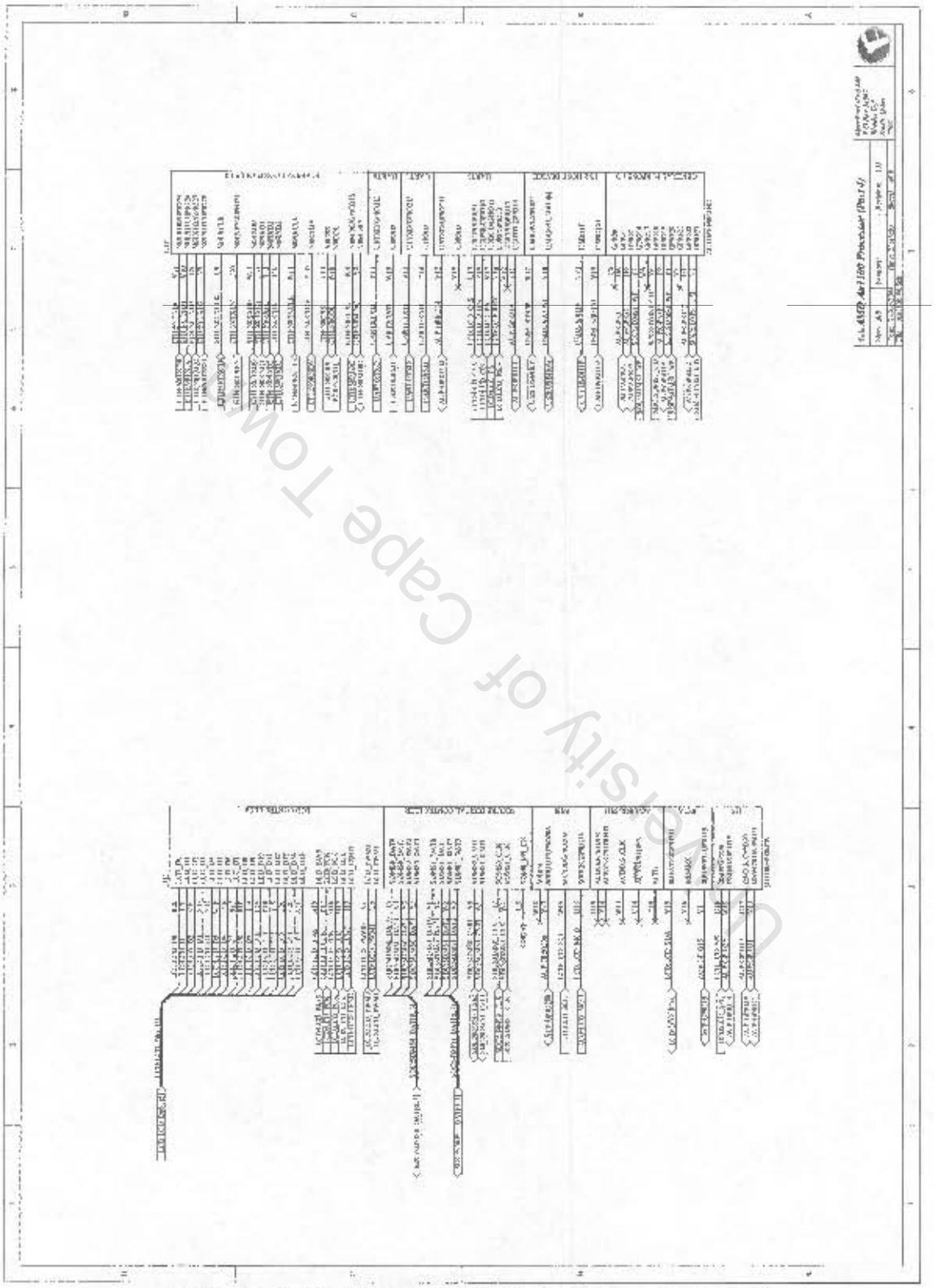

Figure 20 RUI Module Schematic Diagram - Sheet 5 of 8

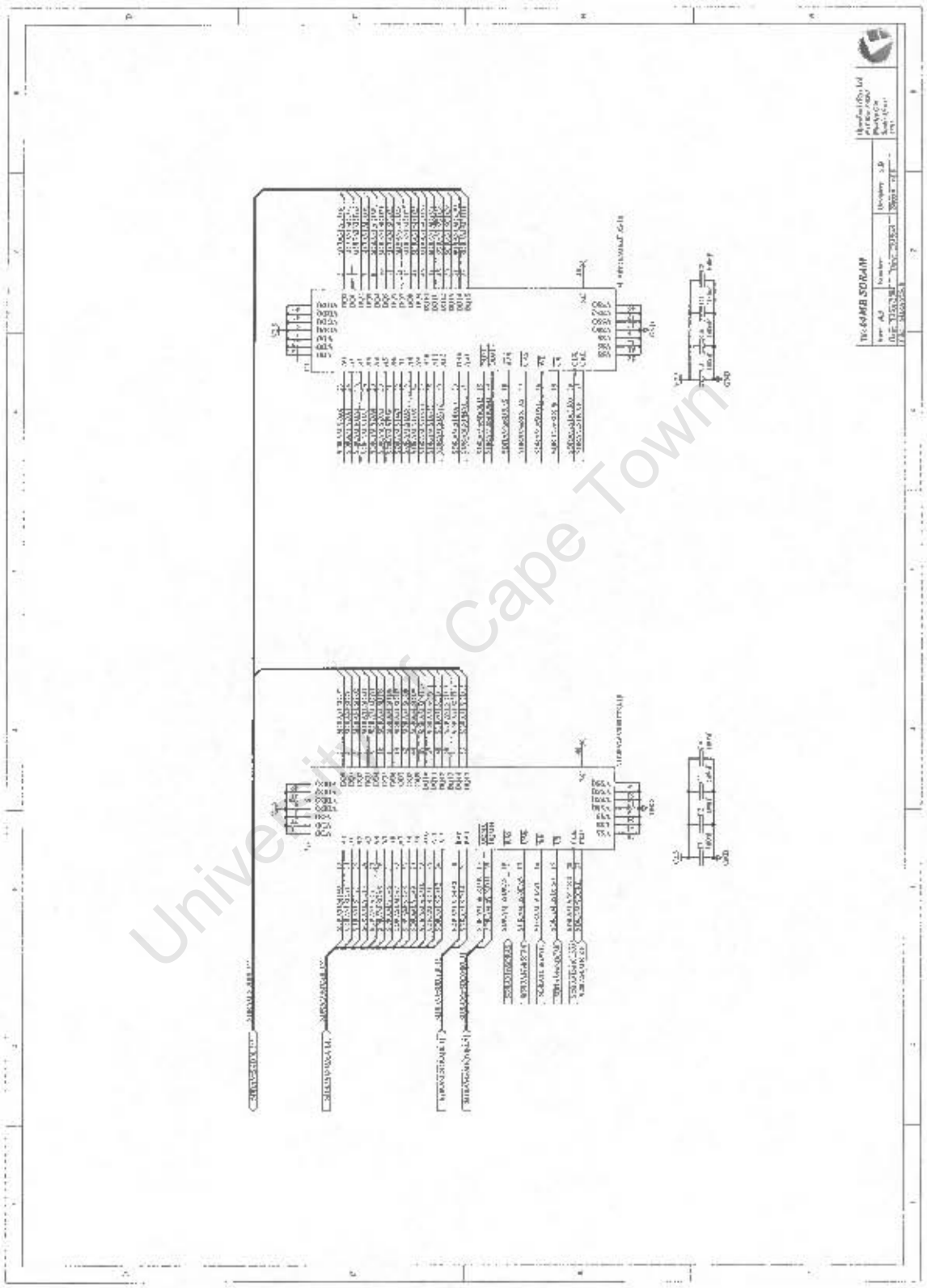

Figure 21 RUI Module Schematic Diagram - Sheet 6 of 8

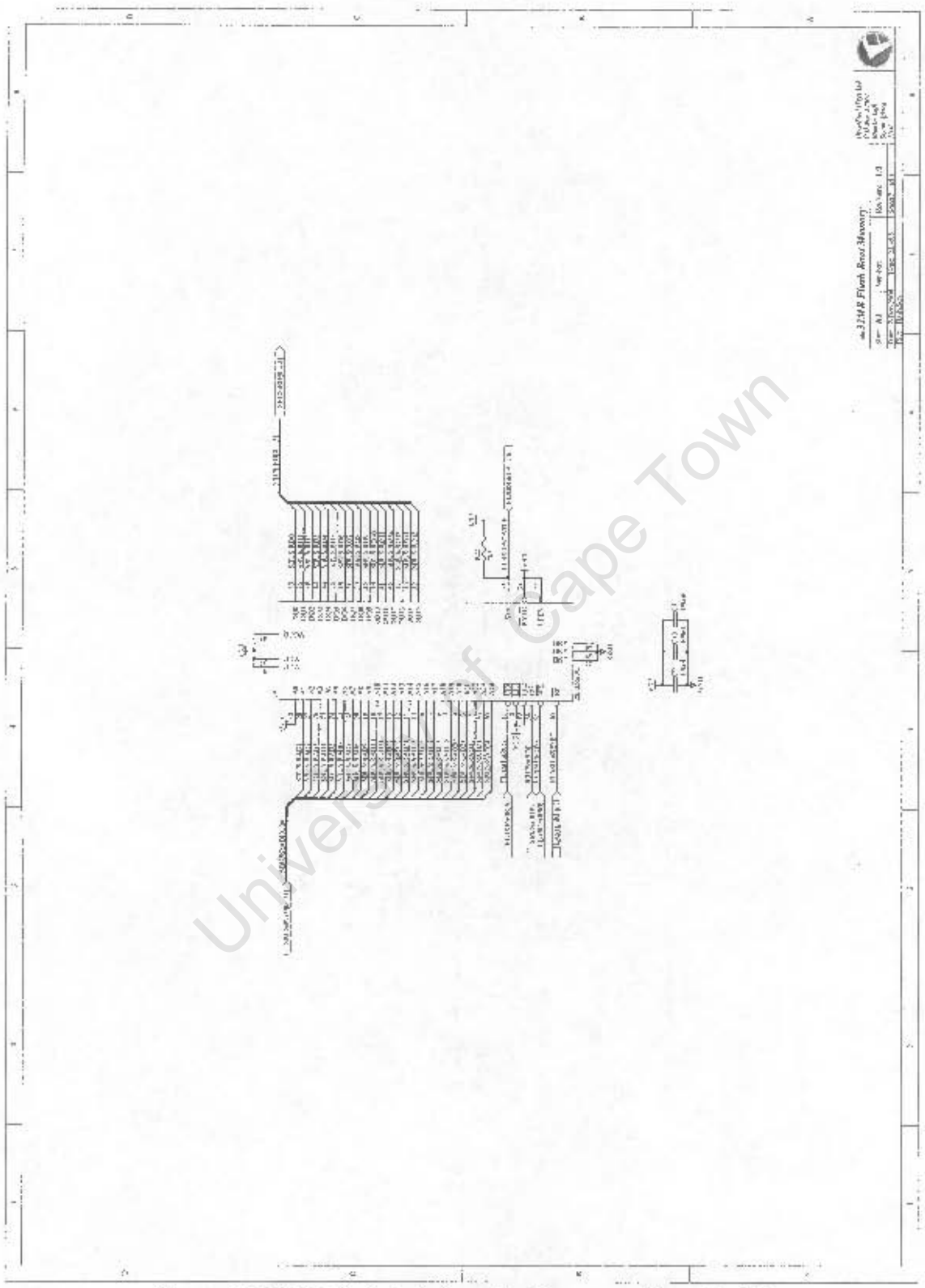

Figure 22 RUI Module Schematic Diagram - Sheet 7 of 8

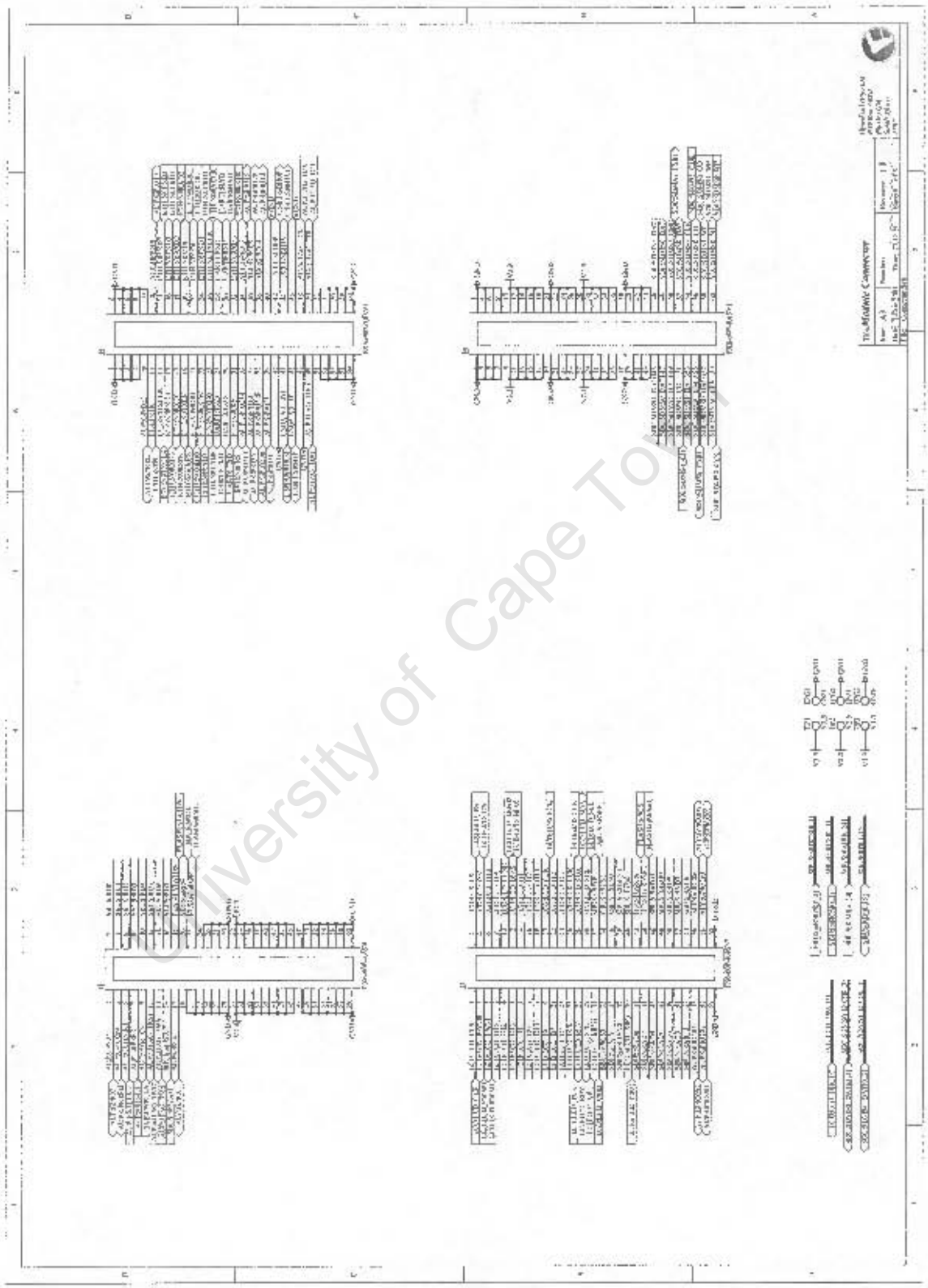

Figure 23 RUI Module Schematic Diagram - Sheet 8 of 8

Table 18 RUI Module Parts List			
Designator	**Value/Part Number**	**Footprint/ Package**	**Description**
C1 - C36	100nF	0402	Ceramic Capacitor, 10%, X7R, 10V
C37, C38	22µF	3528B	Tantalum Capacitor, 10%, X7R, 16V
C39 - C43	10nF	0402	Ceramic Capacitor, 10%, X7R, 10V
C44 - C46	6.8nF	0402	Ceramic Capacitor, 10%, X7R, 10V
C47 - C49	15nF	0402	Ceramic Capacitor, 10%, X7R, 10V
C50 - C52	10µF	3528B	Tantalum Capacitor, 10%, X7R, 16V
R1 - R21	4.7kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R22	2.5kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R23, R24	10Ω	0402	Carbon Film Resistor, 5%, 0.125W
J1 - J4	FX6-60P-0.8SV1	-	Hirose 60-Way SMD Plug
U1	AU1100-400MBC	LF-PBGA-399	AMD Alchemy MIPS32 SoC Processor
U2, U3	MT48V16M16LF TG-10	TSOP54	Micron 256-Mbit Mobile SDRAM
U4	TE28F256J3C-125	TSOP56	Intel 256-Mbit StrataFlash® Flash Memory (J3)
X1	HCM49-12.000MABJT	HCM49	12MHz Crystal

A.2 RUI Motherboard

The schematic diagrams and parts list of the RUI Motherboard is shown in this appendix.

- **Figure 24** RUI Motherboard Schematic Diagram - Sheet 1 of 12
- **Figure 25** RUI Motherboard Schematic Diagram - Sheet 2 of 12
- **Figure 26** RUI Motherboard Schematic Diagram - Sheet 3 of 12
- **Figure 27** RUI Motherboard Schematic Diagram - Sheet 4 of 12
- **Figure 28** RUI Motherboard Schematic Diagram - Sheet 5 of 12
- **Figure 29** RUI Motherboard Schematic Diagram - Sheet 6 of 12
- **Figure 30** RUI Motherboard Schematic Diagram - Sheet 7 of 12
- **Figure 31** RUI Motherboard Schematic Diagram - Sheet 8 of 12
- **Figure 32** RUI Motherboard Schematic Diagram - Sheet 9 of 12
- **Figure 33** RUI Motherboard Schematic Diagram - Sheet 10 of 12
- **Figure 34** RUI Motherboard Schematic Diagram - Sheet 11 of 12
- **Figure 35** RUI Motherboard Schematic Diagram - Sheet 12 of 12
- **Table 19** RUI Motherboard Parts List

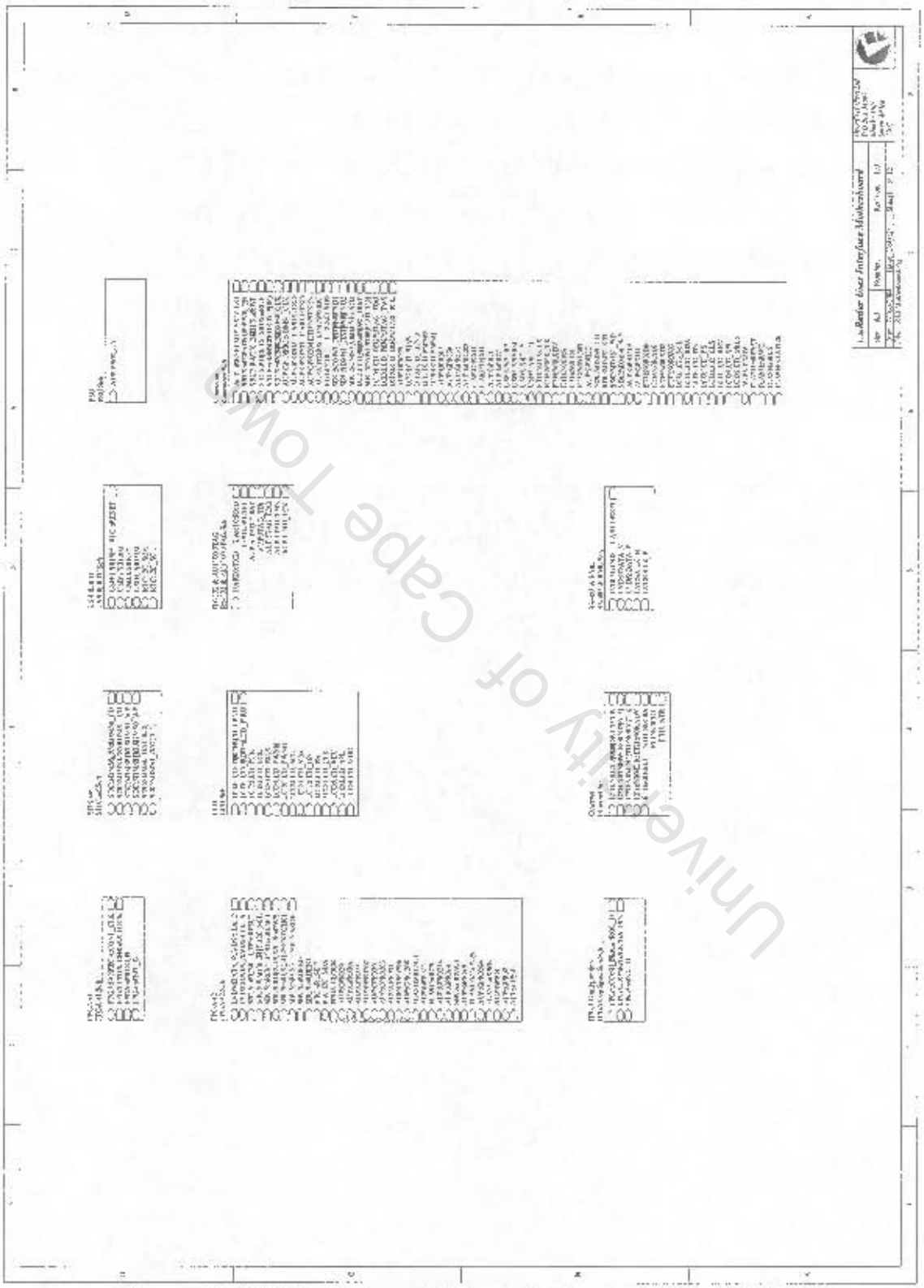

Figure 24 RUI Motherboard Schematic Diagram - Sheet 1 of 12

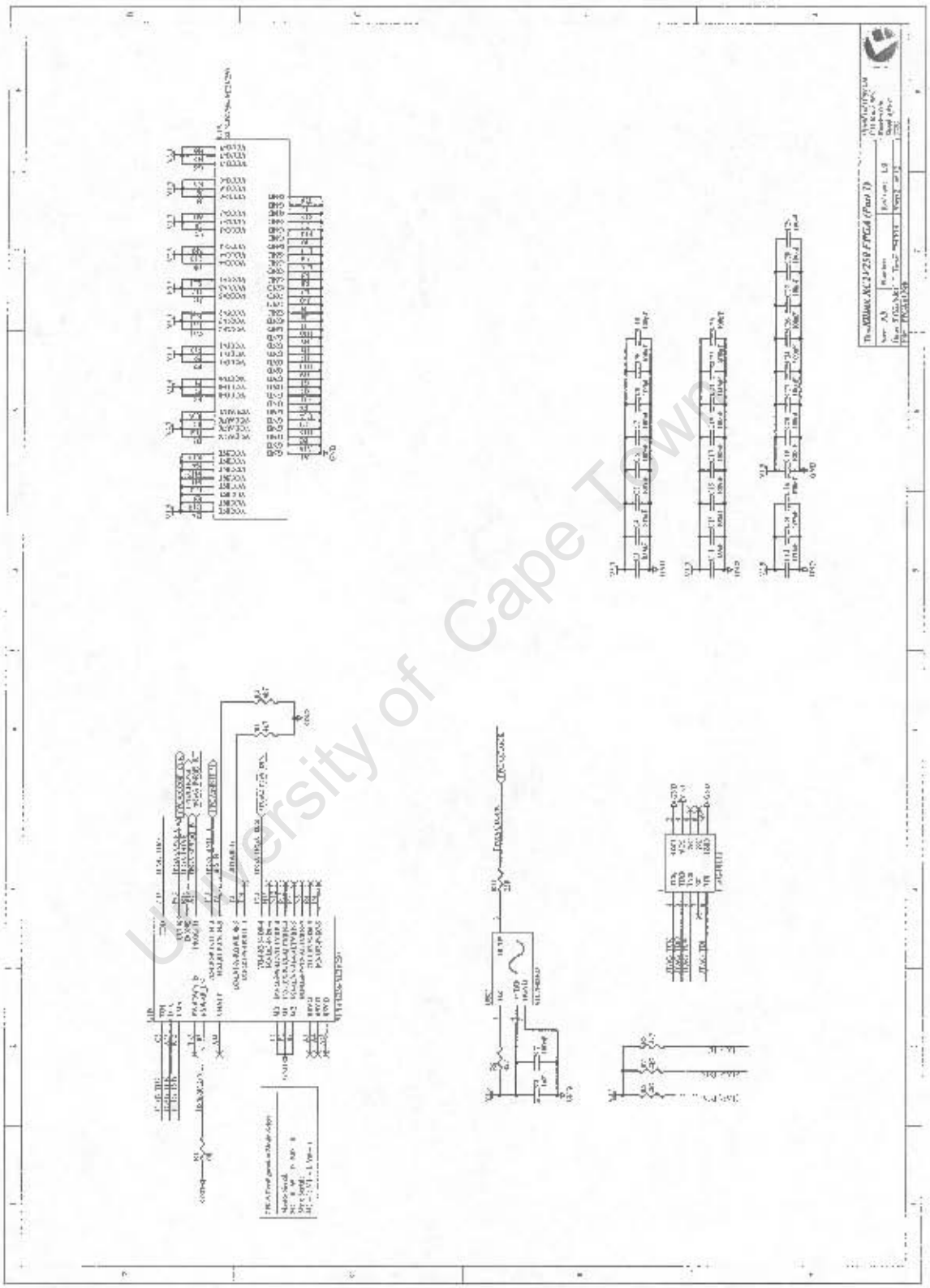

Figure 25 RUI Motherboard Schematic Diagram - Sheet 2 of 12

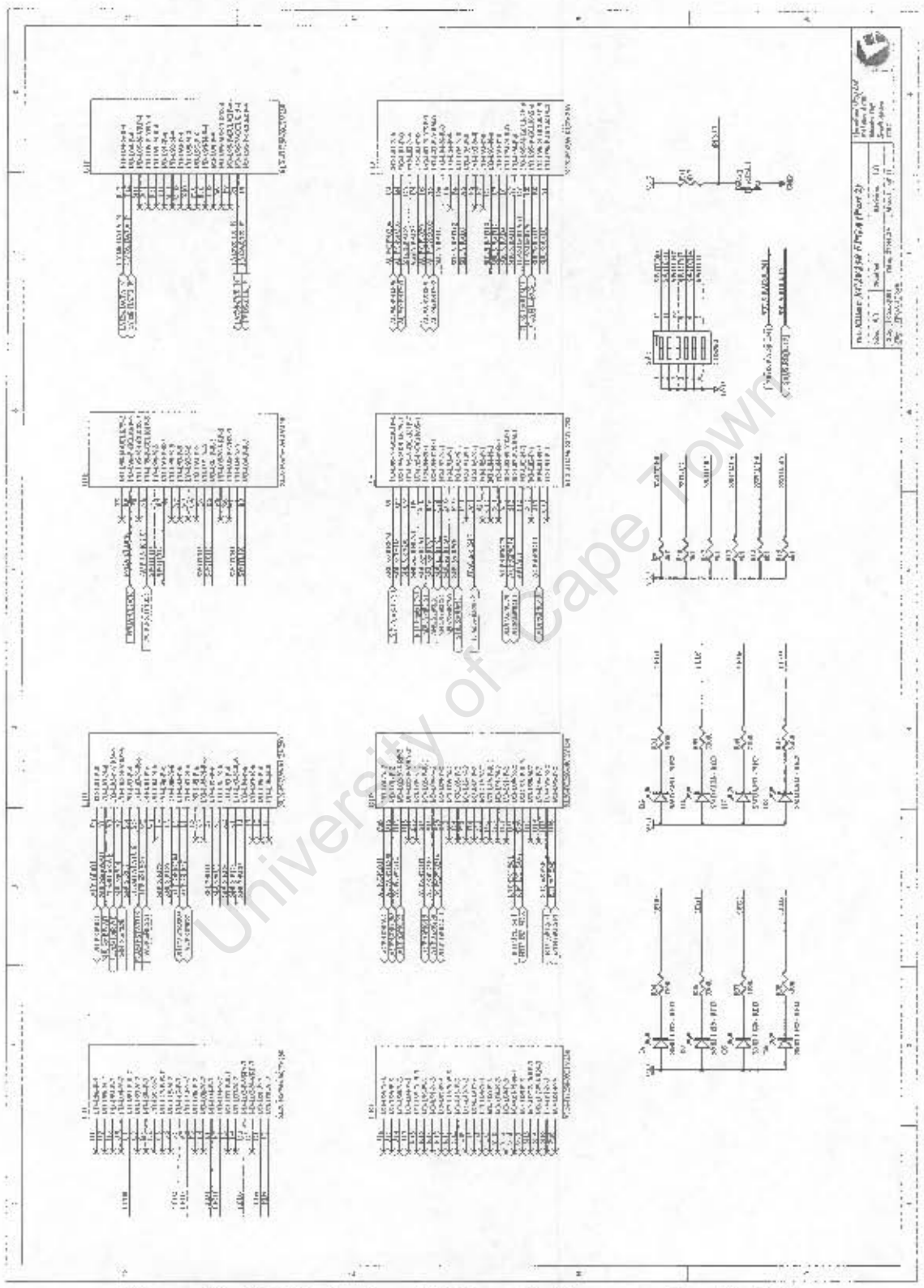

Figure 26 RUI Motherboard Schematic Diagram - Sheet 3 of 12

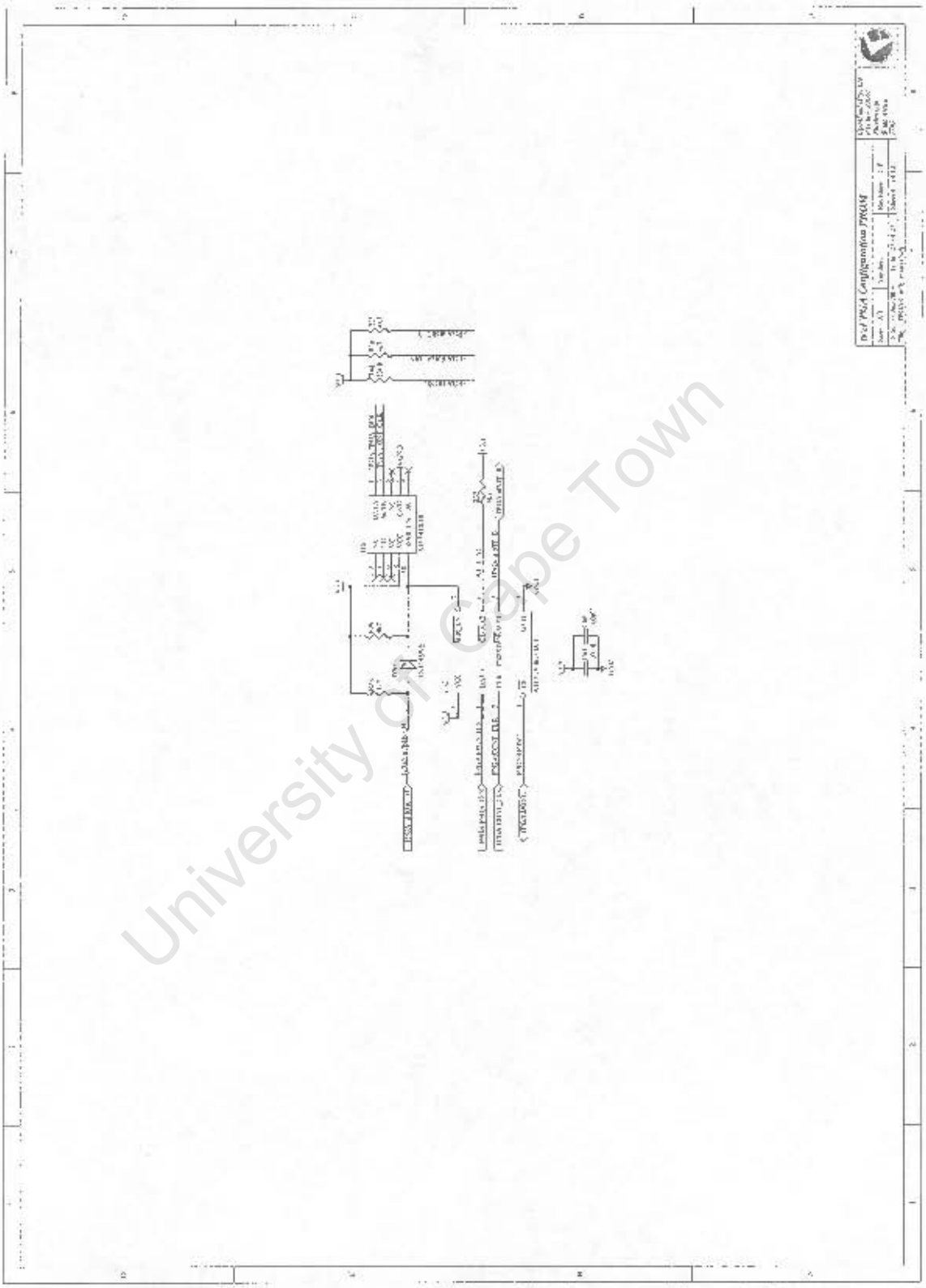

Figure 27 RUI Motherboard Schematic Diagram - Sheet 4 of 12

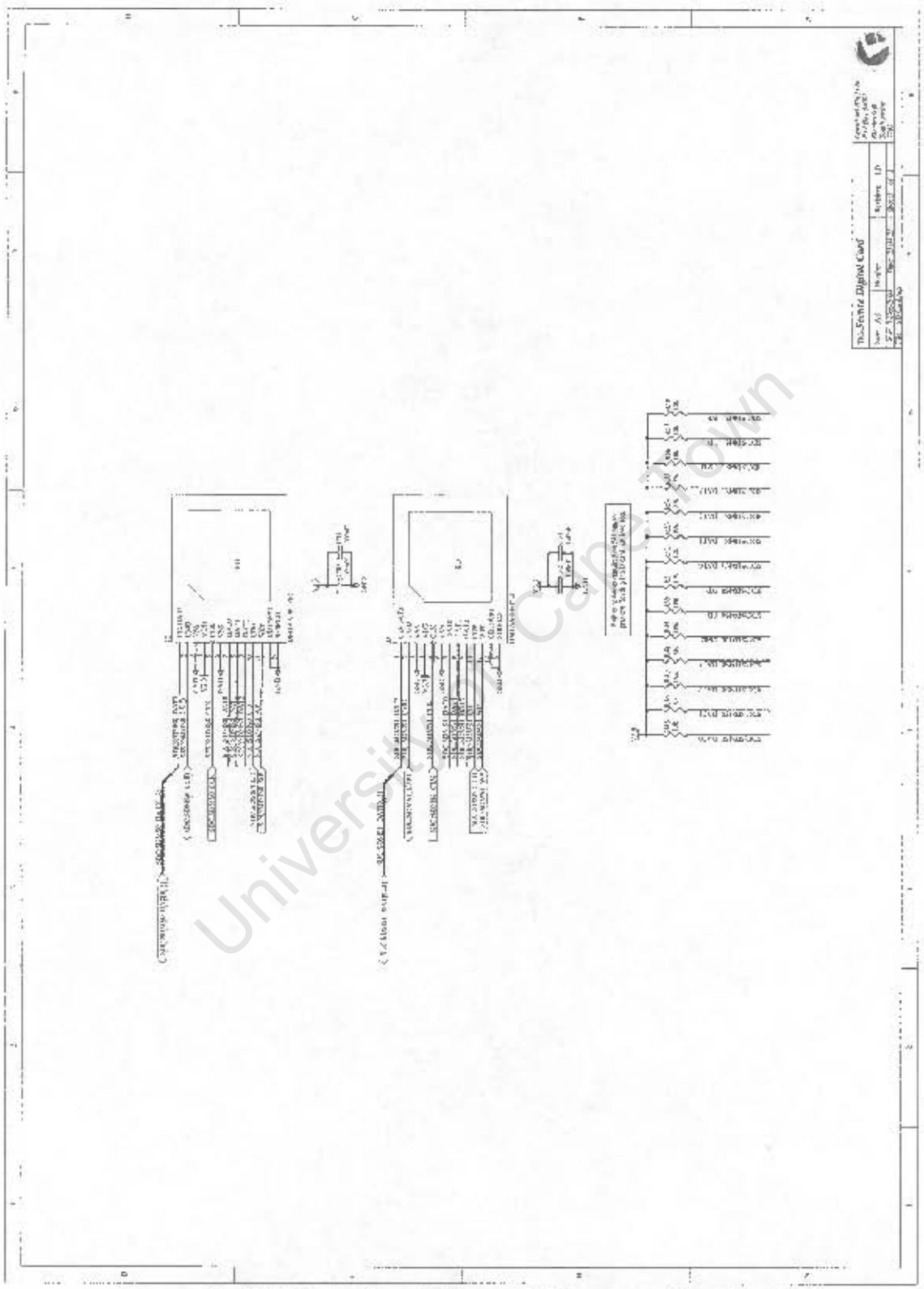

Figure 28 RUI Motherboard Schematic Diagram - Sheet 5 of 12

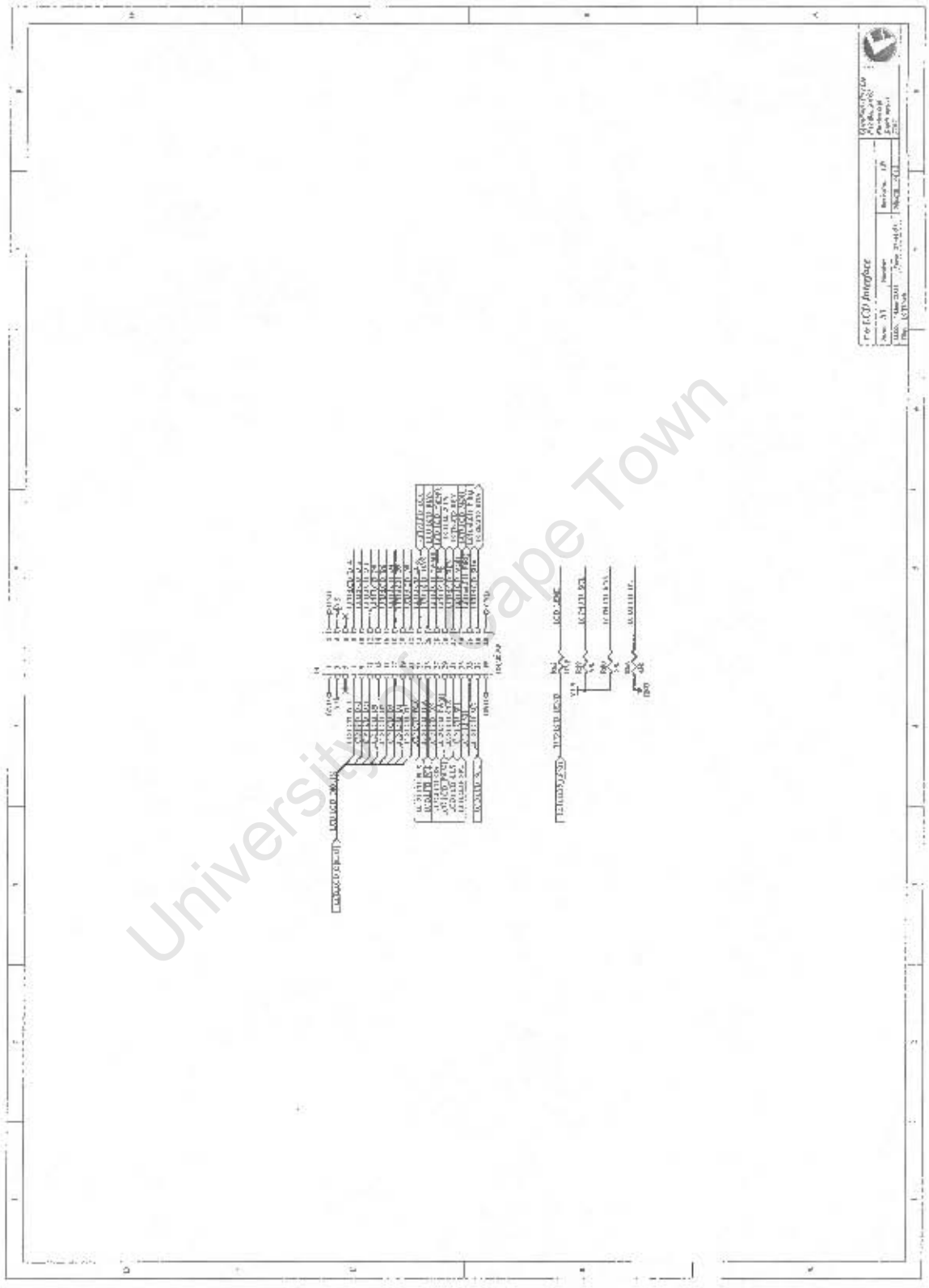

Figure 29 RUI Motherboard Schematic Diagram - Sheet 6 of 12

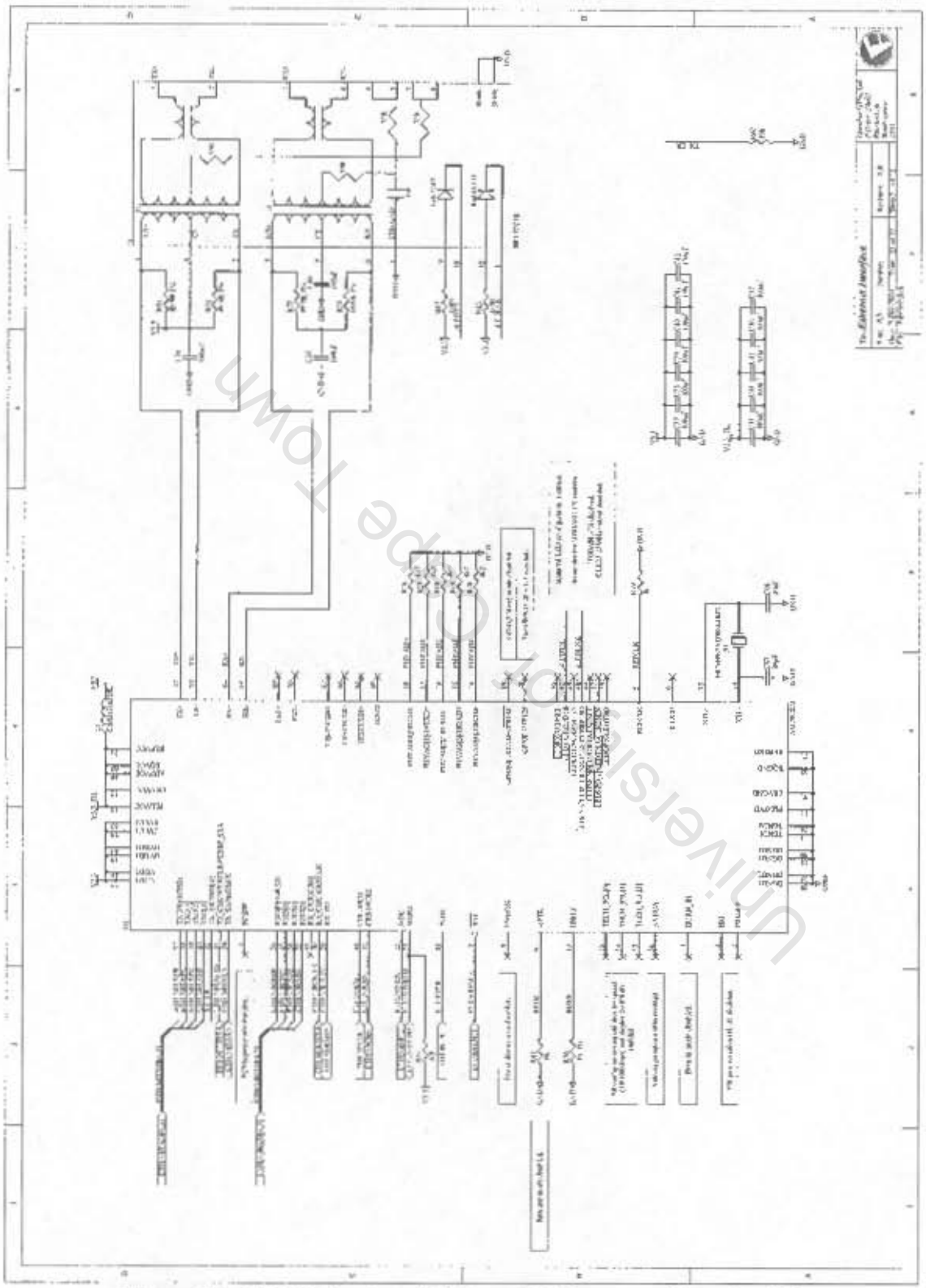

Figure 30 RUI Motherboard Schematic Diagram - Sheet 7 of 12

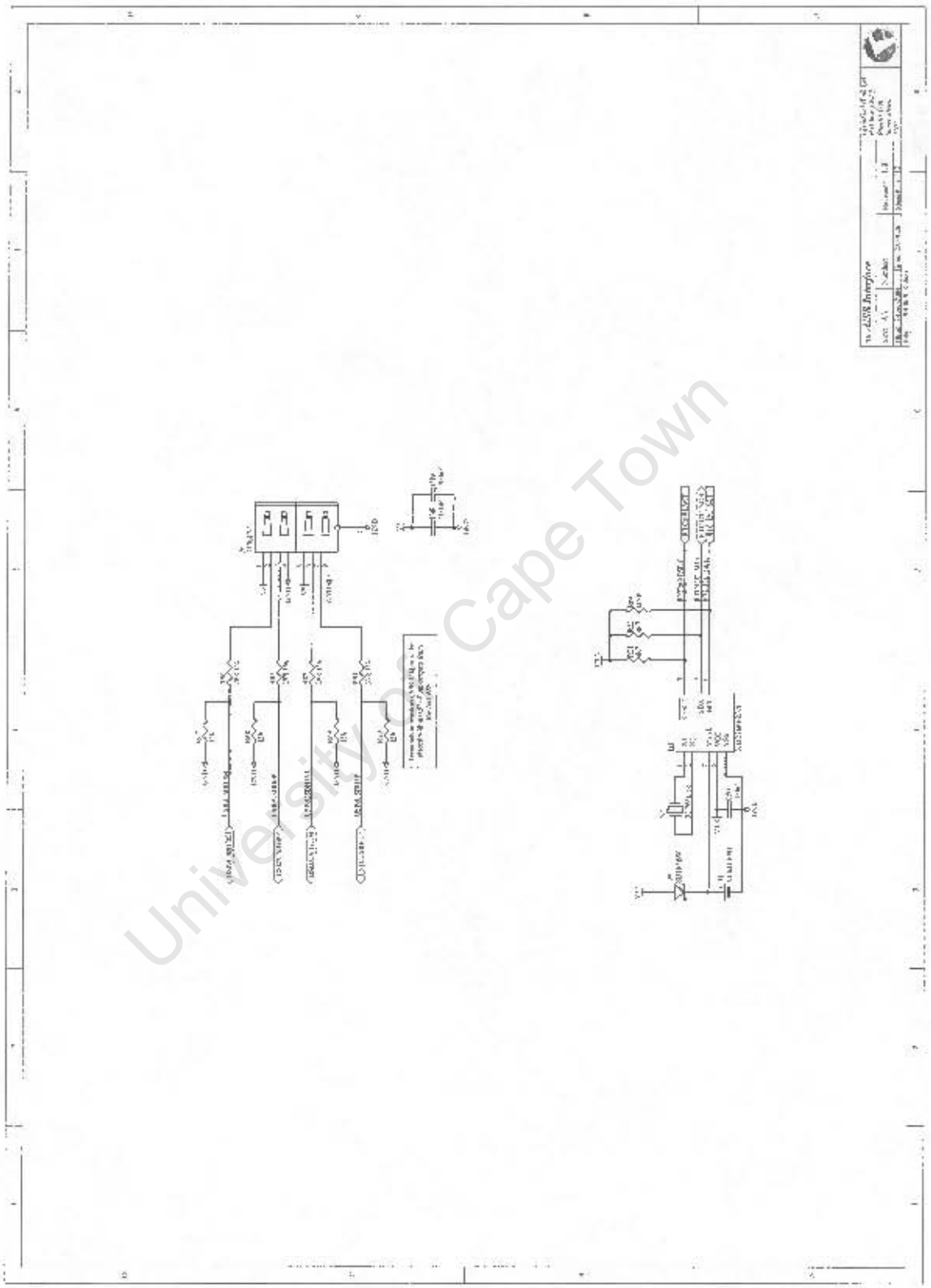

Figure 31 RUI Motherboard Schematic Diagram - Sheet 8 of 12

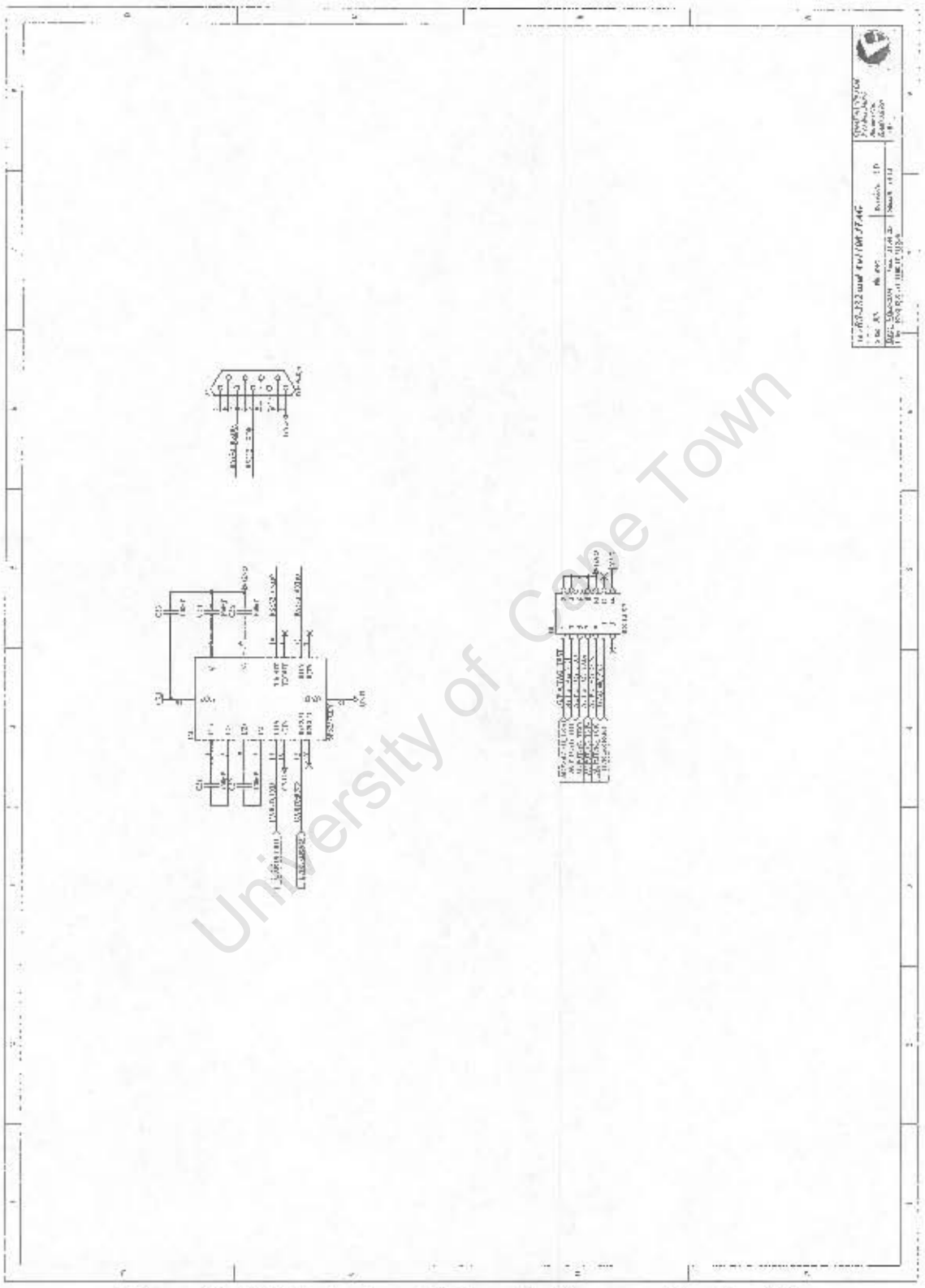

Figure 32 RUI Motherboard Schematic Diagram - Sheet 9 of 12

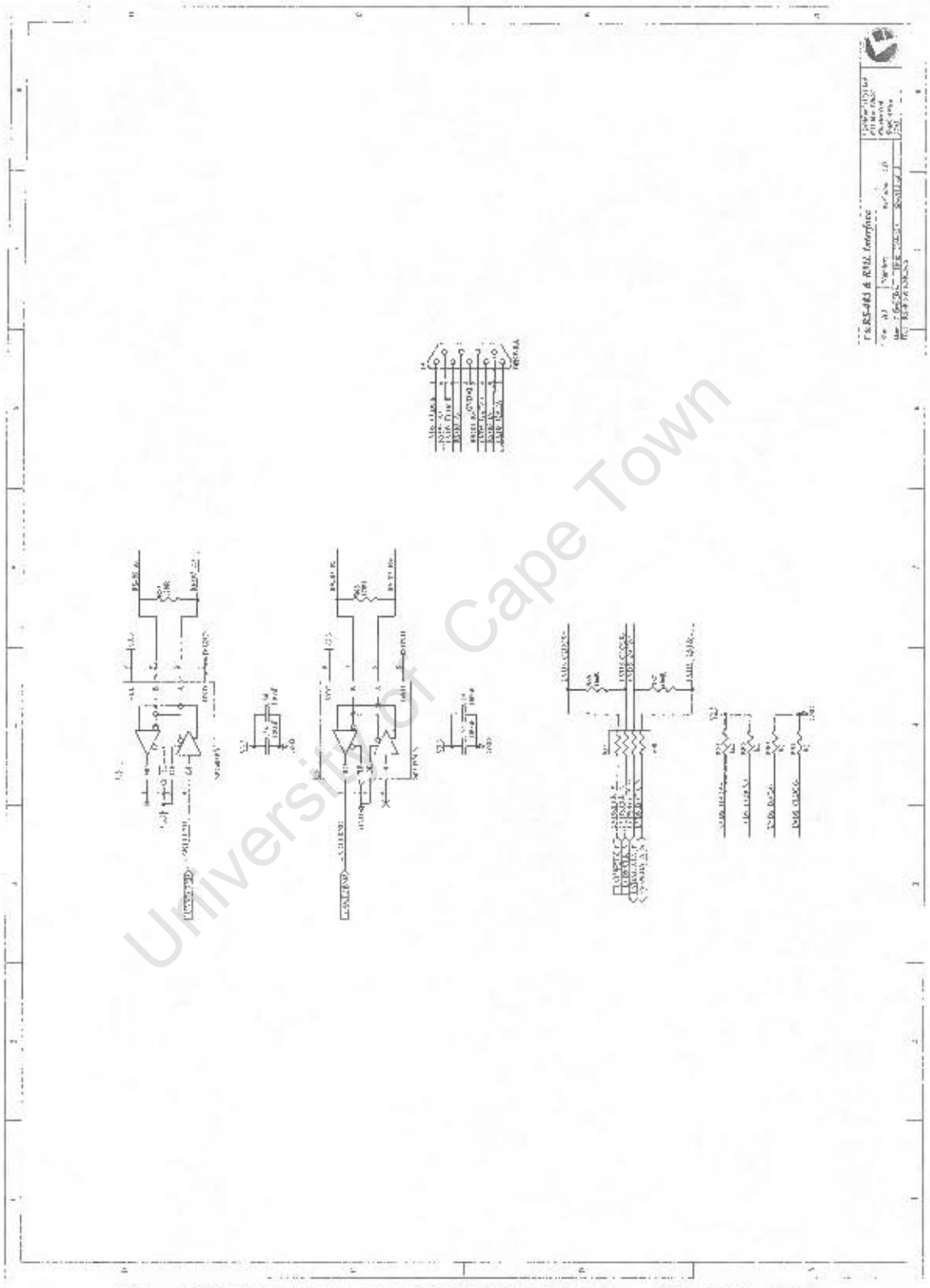

Figure 33 RUI Motherboard Schematic Diagram - Sheet 10 of 12

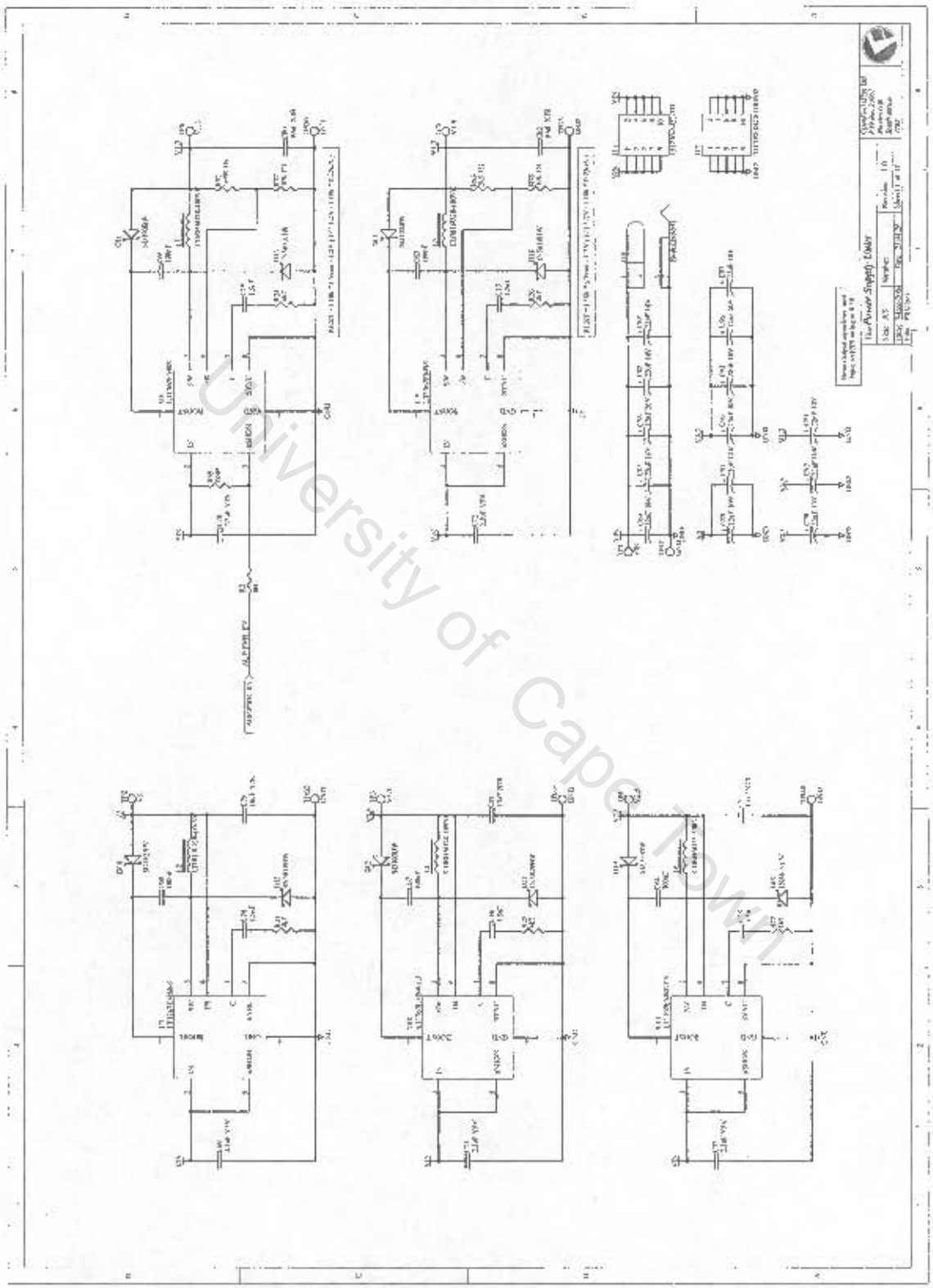

Figure 34 RUI Motherboard Schematic Diagram - Sheet 11 of 12

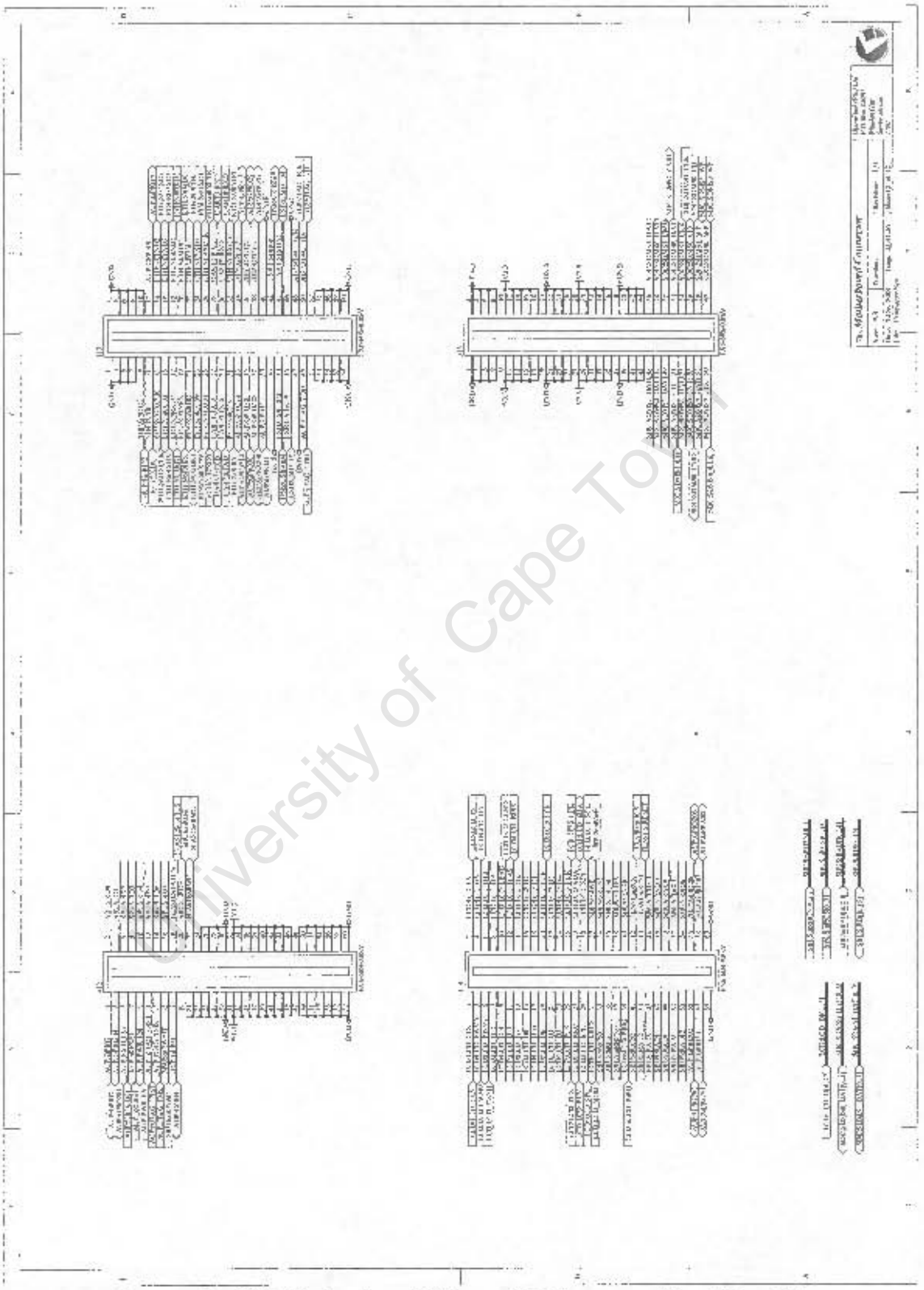

Figure 35 RUI Motherboard Schematic Diagram - Sheet 12 of 12

Table 19 RUI Motherboard Parts List			
Designator	**Value/Part Number**	**Footprint/ Package**	**Description**
B1	VL621-F9D	-	Panasonic Battery
C1	1nF	0402	Ceramic Capacitor, 10%, X7R, 10V
C2 - C66	100nF	0402	Ceramic Capacitor, 10%, X7R, 10V
C67, C68	30pF	0402	Ceramic Capacitor, 10%, X7R, 10V
C69 - C73	2.2µF	1210	Ceramic Capacitor, 10%, Y5V, 16V
C74 - C78	1.5nF	0402	Ceramic Capacitor, 10%, X7R, 10V
C79- C83	10µF	1210	Ceramic Capacitor, 10%, X7R, 16V
C84 - C97	22µF	3528B	Tantalum Capacitor, 10%, X7R, 16V
D1 - D8	L62401CT	0805	Clear Red Light Emitting Diode
D9 - D14	SD103BW	SOD-123	Schottky Diode, 30V, 400mW
D15 - D19	1N5819HW	SOD-123	Schottky Diode, 40V, 1A
D20	1N4148WS	SOD-323	Switch Diode, 75V, 200mW
J1, J16	IDC10	DIL10	10-Way IDC Header
J2, J3	DM1AA-SF-PEJ	-	Hirose SD Memory Card Connector, Standard Type
J4	IDC40	DIL40	40-Way IDC Header
J5	J0011D21B	-	Pulse RJ45 PulseJack Integrated Connector
J6	787617-1	-	Tyco Amp Shielded Stacked USB Type A Connector
J7	DB9S-RA	-	9-Way D-Type Socket, Right Angle, PCB Mount
J8	IDC14	DIL14	14-Way IDC Header

J9	DB9P-RA	-	9-Way D-Type Plug, Right Angle, PCB Mount
J10	PJ-002B-SMT	-	2.5mm SMD Power Jack
J11, J17	151210-8422-TB	DIL10	10-Way 2mm x 2mm Vertical Plug
J12 - J15	FX6-60S-0.8SV	-	Hirose 60-Way SMD Socket
L1	CIM21J102NE	0805	Samsung 1kΩ 200mA Chip Inductor, EMI Suppression
L2 - L6	CDRII6D28-100NC	-	Sumida 10µH 65mΩ 1.7A Power Inductor
OSC1	STCA-05433	-	32MHz Oscillator
R1, R2	0Ω	0402	Carbon Film Resistor, 5%, 0.125W
R3 - R32	4.7kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R33	22Ω	0402	Carbon Film Resistor, 5%, 0.125W
R34 - R44	330Ω	0402	Carbon Film Resistor, 5%, 0.125W
R45 - R62	10kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R63 - R65	DNP	0402	Carbon Film Resistor, 5%, 0.125W
R66 - R70	15kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R71 - R74	49.9Ω	0402	Carbon Film Resistor, 1%, 0.125W
R75	1.5kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R76 - R78	10kΩ	0402	Carbon Film Resistor, 1%, 0.125W
R79	1kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R80 - R83	20Ω	0402	Carbon Film Resistor, 1%, 0.125W
R84, R85	120Ω	0402	Carbon Film Resistor, 5%, 0.125W
R86, R87	100Ω	0402	Carbon Film Resistor, 5%, 0.125W
R88 - R91	1.2kΩ	0402	Carbon Film Resistor, 5%, 0.125W
R92	169Ω	0402	Carbon Film Resistor, 1%, 0.125W
R93	2.5kΩ	0402	Carbon Film Resistor, 1%, 0.125W
RP1	50Ω	RP4-0402	Resistor Pack, 1%, 0.125W

SW1	CHS06B	SOP12	6-Way SMD Slide Switch
SW2	FSMSM	-	Tactile Switch 6 x 3.5mm SPST
U1	XC2V250-6FG256I	BGA256	Xilinx 250k System Gate Virtex™-II Platform FPGA
U2	AM79C874VI	PQT80	AMD NetPHY™-1LP Low Power 10/100-TX/FX Ethernet Transceiver
U3	X1227S8I-2.7A	SOP8	Xicor Real Time Clock/Calendar/ CPU Supervisor with EEPROM
U4	MAX3232CUE	TSSOP16	Maxim +3.3V Multichannel RS-232 Transceiver
U5, U6	ST3485EBDR	SOIC8	ST Microelectronics RS-485/422 Transceiver
U7	LT1767EMS8-5	SOP8	Linear Technology +5V Monolithic 1.5A, 1.25MHz Step-Down Switching Regulator
U8, U9	LT1767EMS8	SOP8	Linear Technology Adjustable Monolithic 1.5A, 1.25MHz Step-Down Switching Regulator
U10	LT1767EMS8-3.3	SOP8	Linear Technology +3.3V Monolithic 1.5A, 1.25MHz Step-Down Switching Regulator
U11	LT1767EMS8-2.5	SOP8	Linear Technology +2.5V Monolithic 1.5A, 1.25MHz Step-Down Switching Regulator
U12	AT17LV002-10CI	LAP8	Atmel 2-Mbit FPGA Configuration EEPROM Memory
X1	HCM49-25.000MABJT	HCM49	25MHz Crystal

X2	STCA-05585	-	32.768kHz Crystal

A.3 Miscellaneous Schematic Diagrams

The miscellaneous schematic diagrams of the design project is shown in this appendix.

- **Figure 36** RUI RS-232 Cable
- **Table 20** RUI RS-232 Cable Parts List

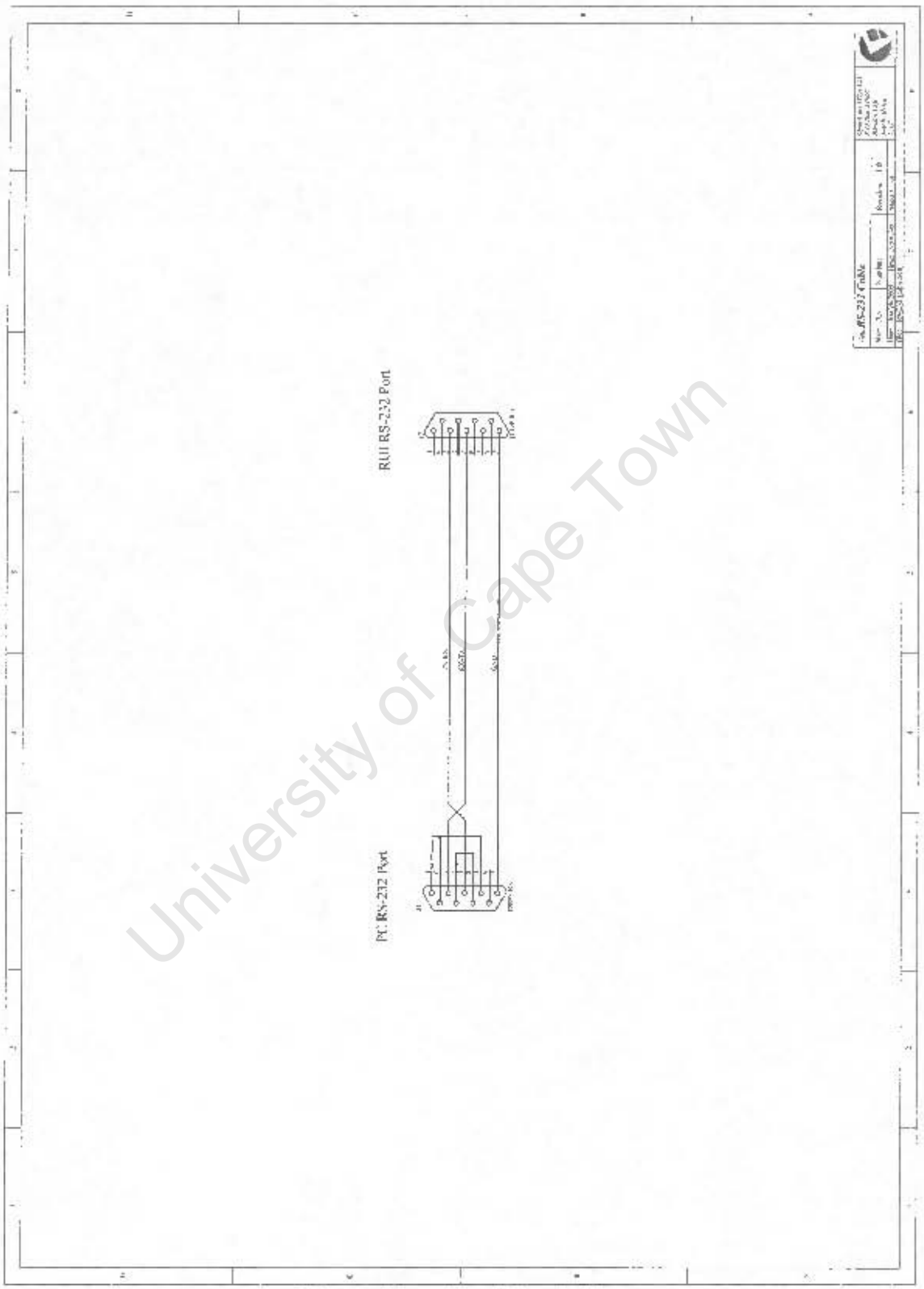

Figure 36 RUI RS-232 Cable

Table 20 RUI RS-232 Cable Parts List			
Designator	**Value/Part Number**	**Footprint/ Package**	**Description**
J1	DB9S-RA	-	9-Way D-Type Socket, Solder Type
J2	DB9P-RA	-	9-Way D-Type Plug, Solder Type

Appendix B

Printed Circuit Board Layouts

B.1 RUI Module

The printed circuit board layout of the RUI Module is shown in this appendix.

- **Figure 37** RUI Module PCB Layout - Top Overlay Layer
- **Figure 38** RUI Module PCB Layout - Top Signal Layer
- **Figure 39** RUI Module PCB Layout - Ground Plane
- **Figure 40** RUI Module PCB Layout - Internal Signal Layer 1
- **Figure 41** RUI Module PCB Layout - +1.22V Supply Plane
- **Figure 42** RUI Module PCB Layout - +2.5V Supply Plane
- **Figure 43** RUI Module PCB Layout - Internal Single Layer 2
- **Figure 44** RUI Module PCB Layout - +3.3V/V_{IN} Supply Plane
- **Figure 45** RUI Module PCB Layout - Bottom Signal Layer
- **Figure 46** RUI Module PCB Layout - Bottom Overlay Layer

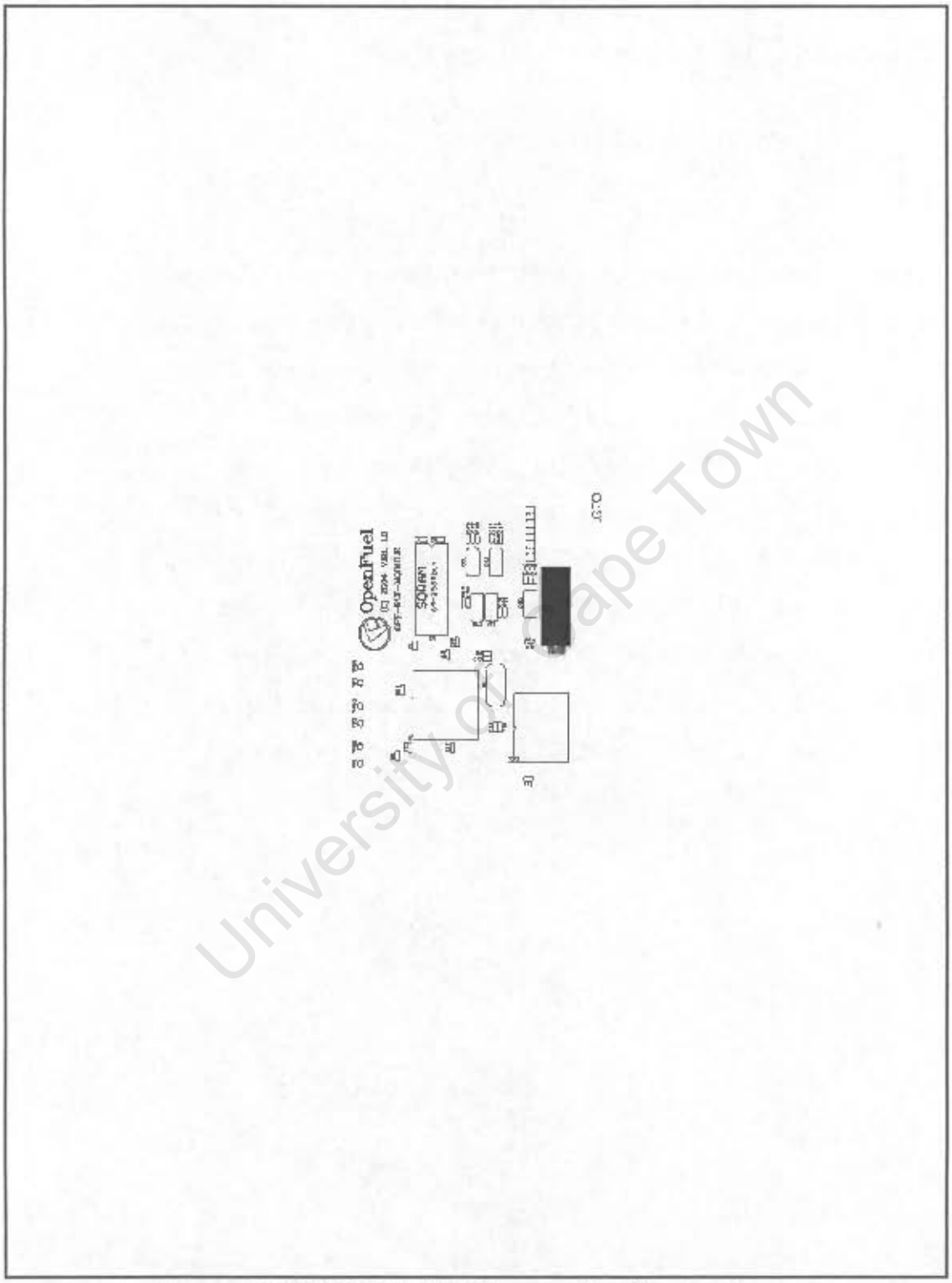

Figure 37 RUI Module PCB Layout - Top Overlay Layer

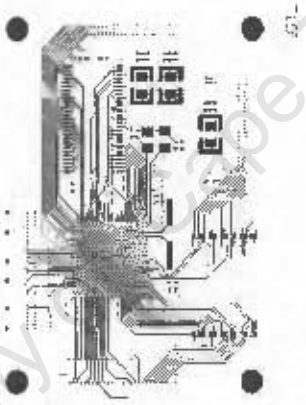

Figure 38 RUI Module PCB Layout - Top Signal Layer

Figure 39 RUI Module PCB Layout - Ground Plane

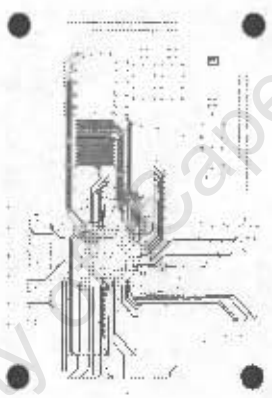

Figure 40 RUI Module PCB Layout - Internal Signal Layer 1

Figure 41 RUI Module PCB Layout - +1.22V Supply Plane

Figure 42 RUI Module PCB Layout - +2.5V Supply Plane

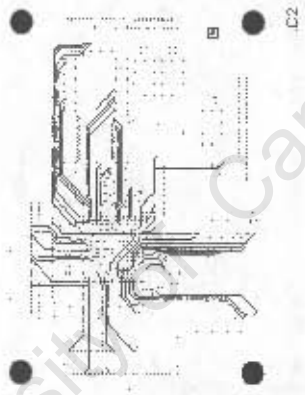

Figure 43 RUI Module PCB Layout - Internal Signal Layer 2

Figure 44 RUI Module PCB Layout - +3.3V Supply Plane

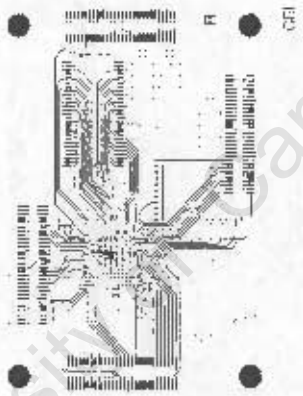

Figure 45 RUI Module PCB Layout - Bottom Signal Layer

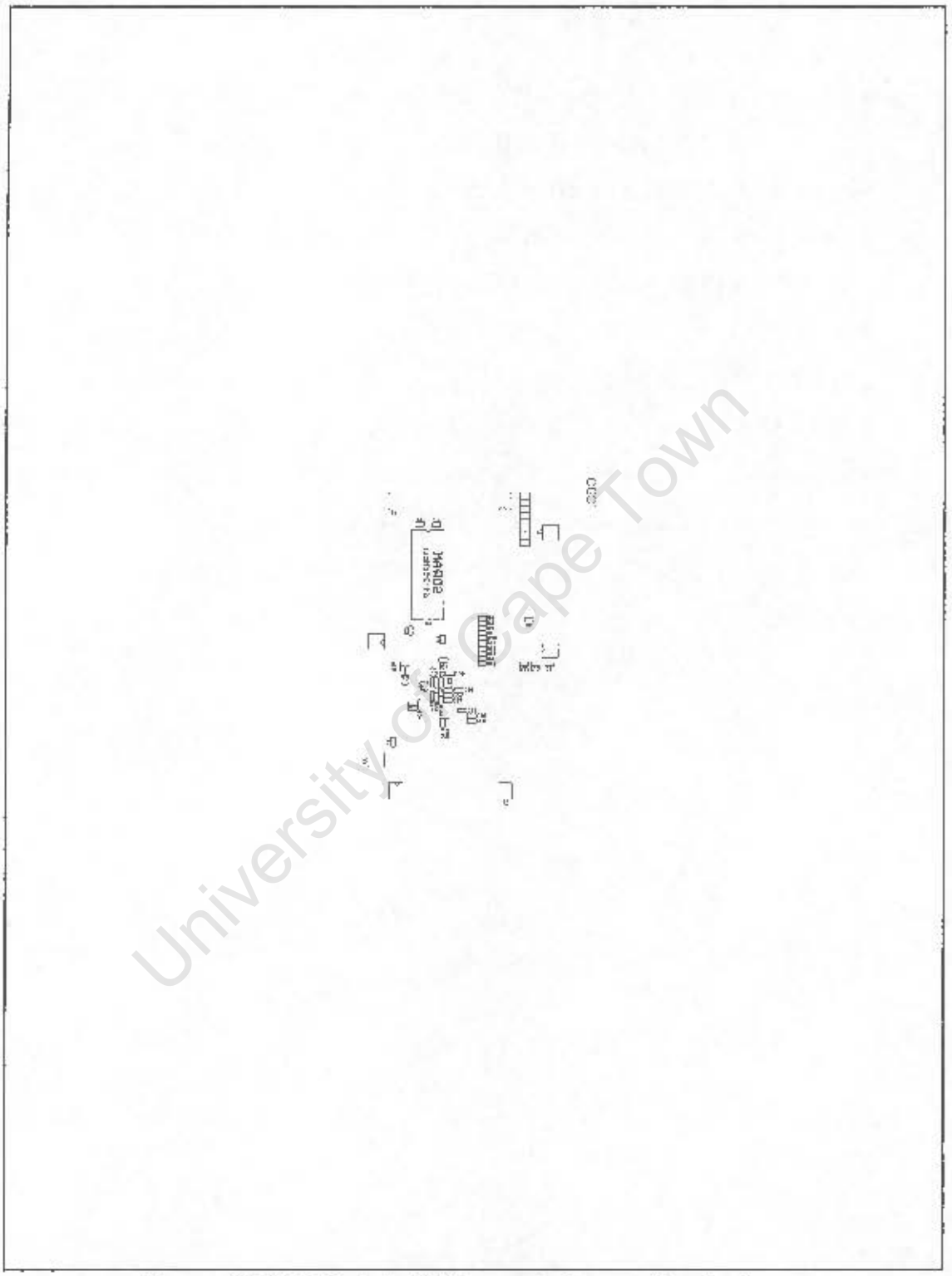

Figure 46 RUI Module PCB Layout - Bottom Overlay Layer

B.2 RUI Motherboard

The printed circuit board layout of the RUI Motherboard is shown in this appendix.

- **Figure 47** RUI Motherboard PCB Layout - Top Overlay Layer
- **Figure 48** RUI Motherboard PCB Layout - Top Signal Layer
- **Figure 49** RUI Motherboard PCB Layout - Ground Plane
- **Figure 50** RUI Motherboard PCB Layout - Internal Signal Layer 1
- **Figure 51** RUI Motherboard PCB Layout - +1.22V/+1.5V Supply Plane
- **Figure 52** RUI Motherboard PCB Layout - -2.5V/-5V Supply Plane
- **Figure 53** RUI Motherboard PCB Layout - Internal Single Layer 2
- **Figure 54** RUI Motherboard PCB Layout - +3.3V/V_{IN} Supply Plane
- **Figure 55** RUI Motherboard PCB Layout - Bottom Signal Layer
- **Figure 56** RUI Motherboard PCB Layout - Bottom Overlay Layer

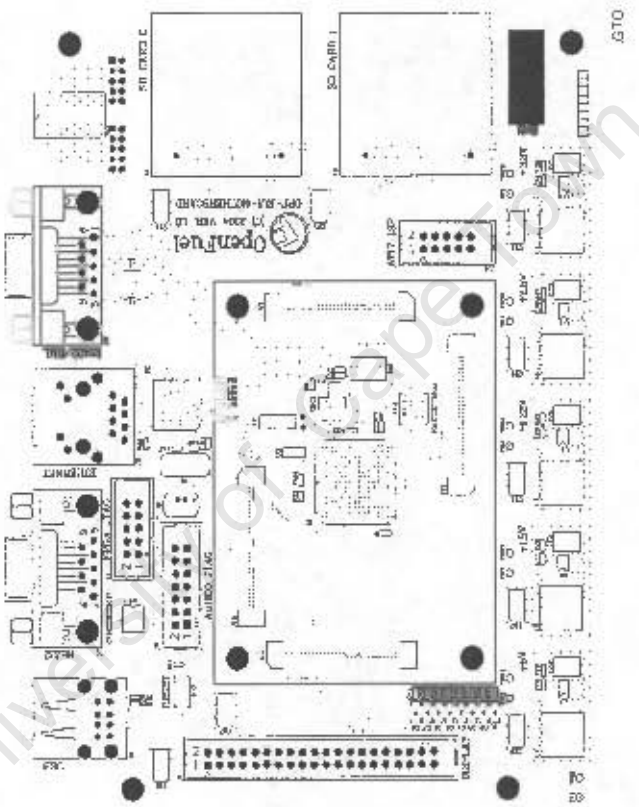

Figure 47 RUI Motherboard PCB Layout - Top Overlay Layer

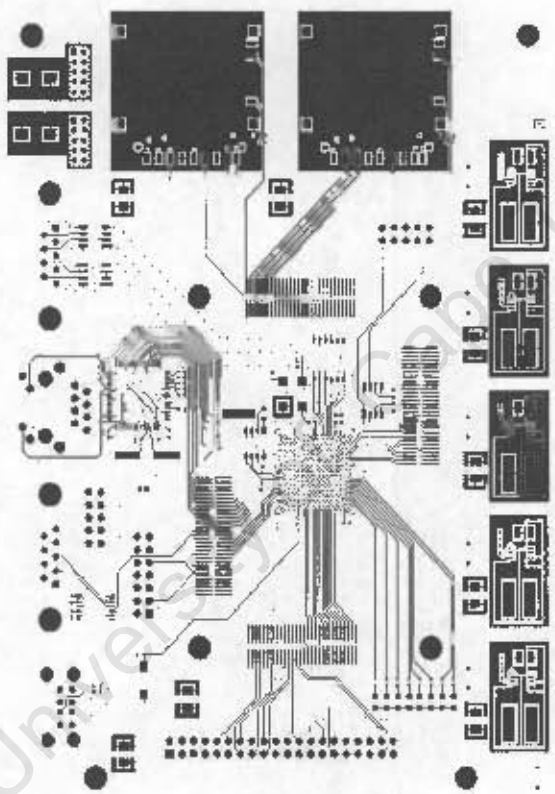

Figure 48 RUI Motherboard PCB Layout - Top Signal Layer

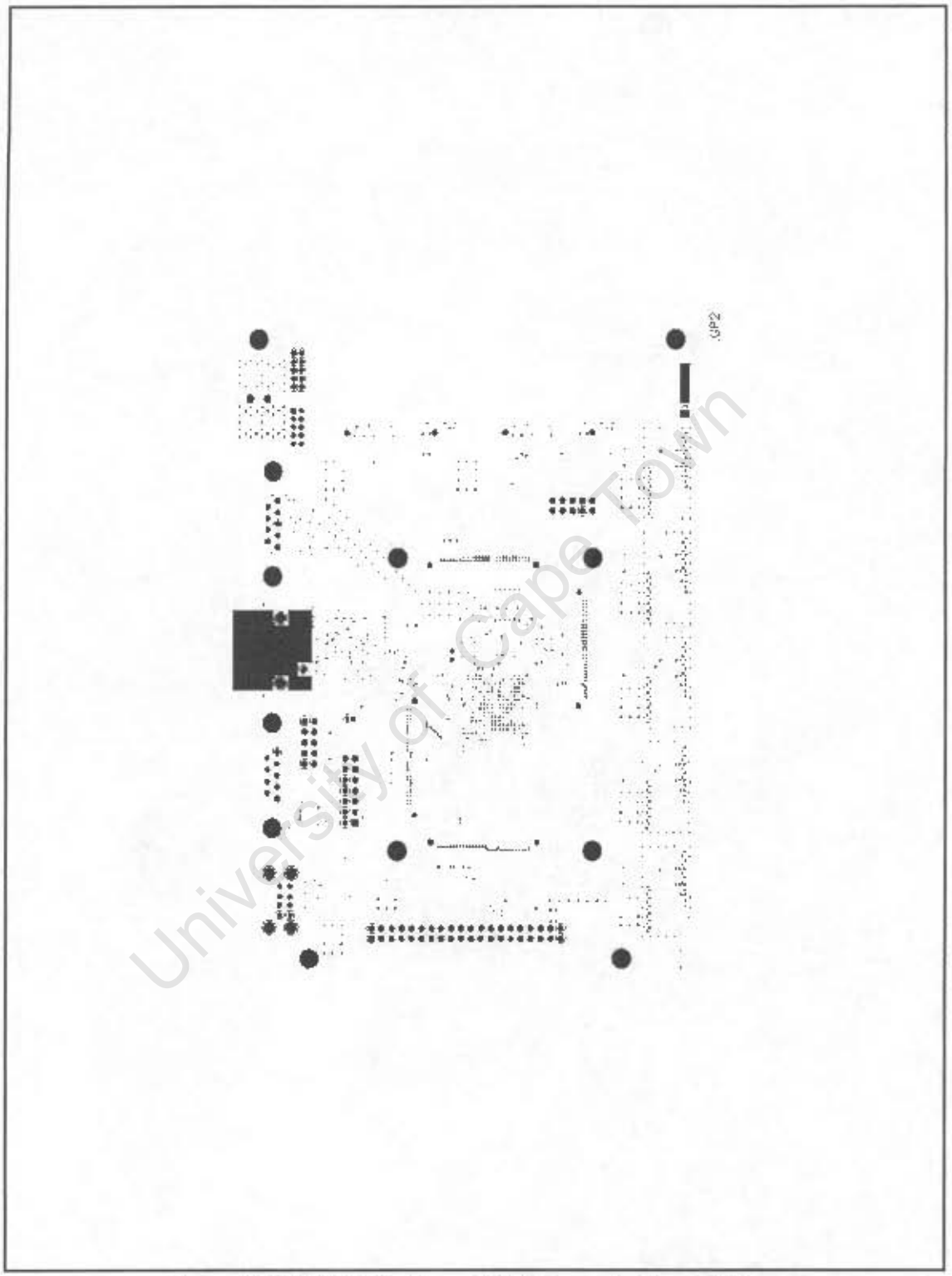

Figure 49 RUI Motherboard PCB Layout - Ground Plane

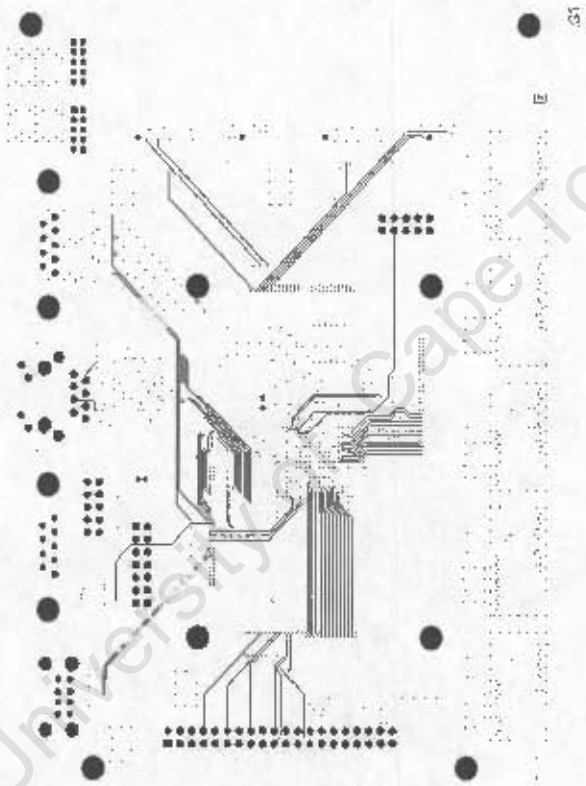

Figure 50 RUI Motherboard PCB Layout - Internal Signal Layer 1

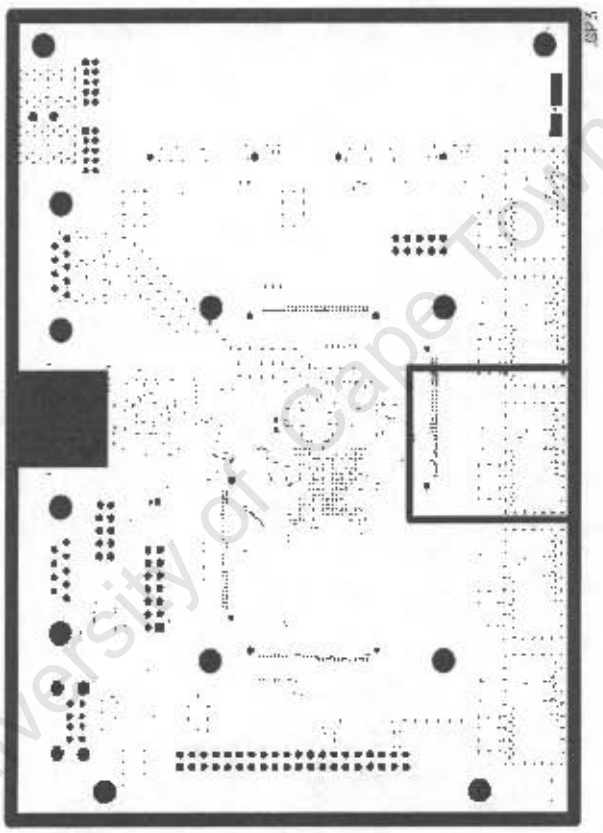

Figure 51 RUI Motherboard PCB Layout - +1.22V/+1.5V Supply Plane

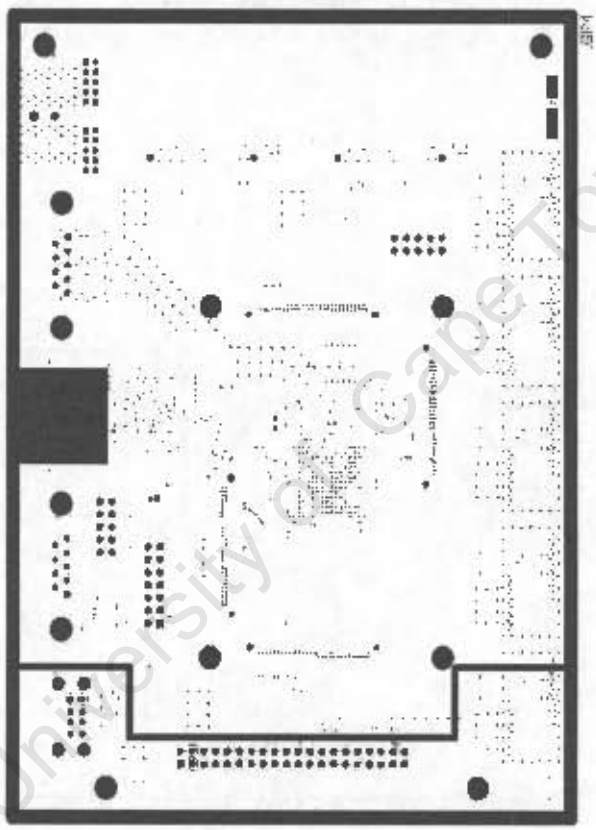

Figure 52 RUI Motherboard PCB Layout - +2.5V/+5V Supply Plane

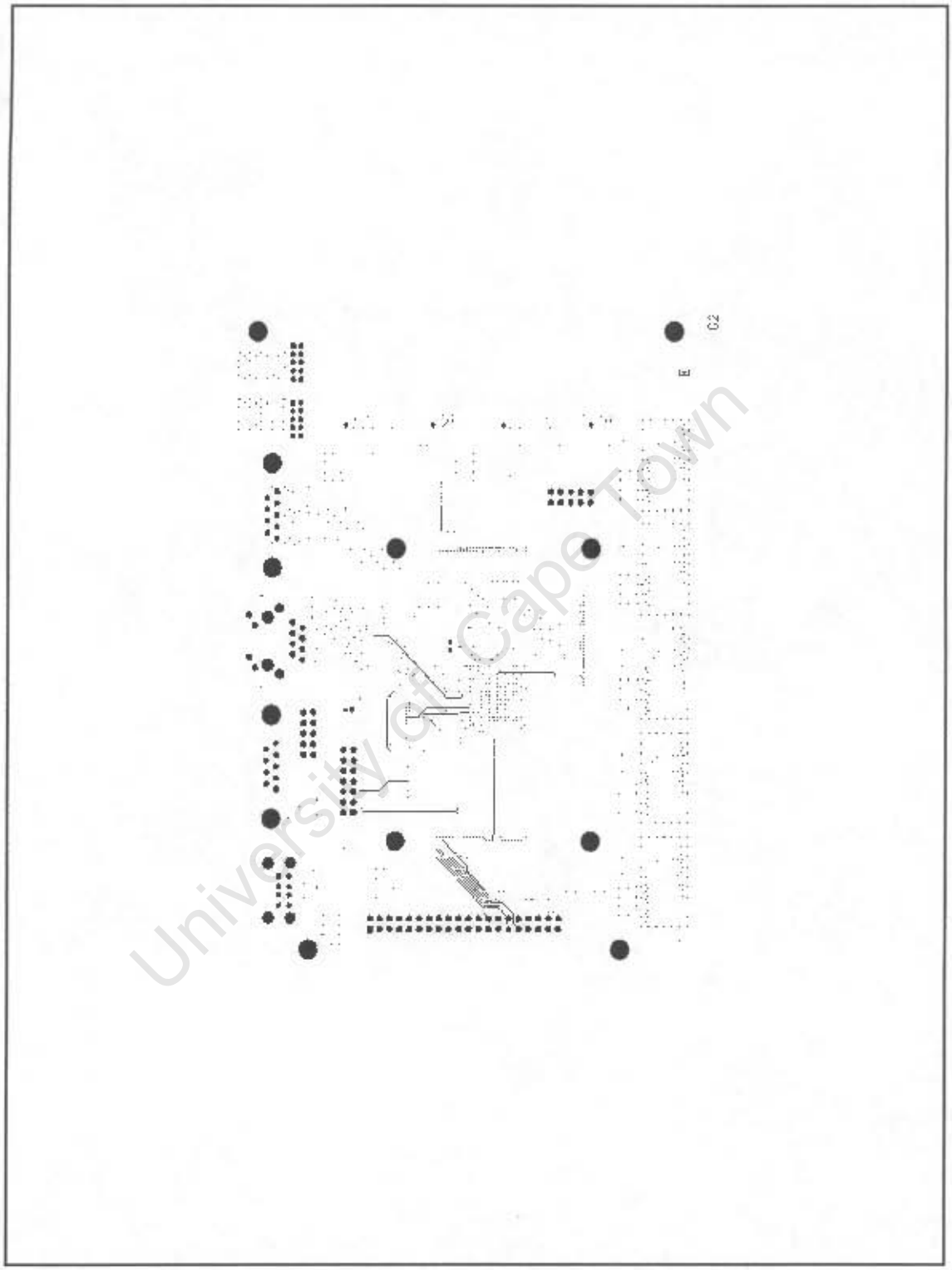

Figure 53 RUI Motherboard PCB Layout - Internal Signal Layer 2

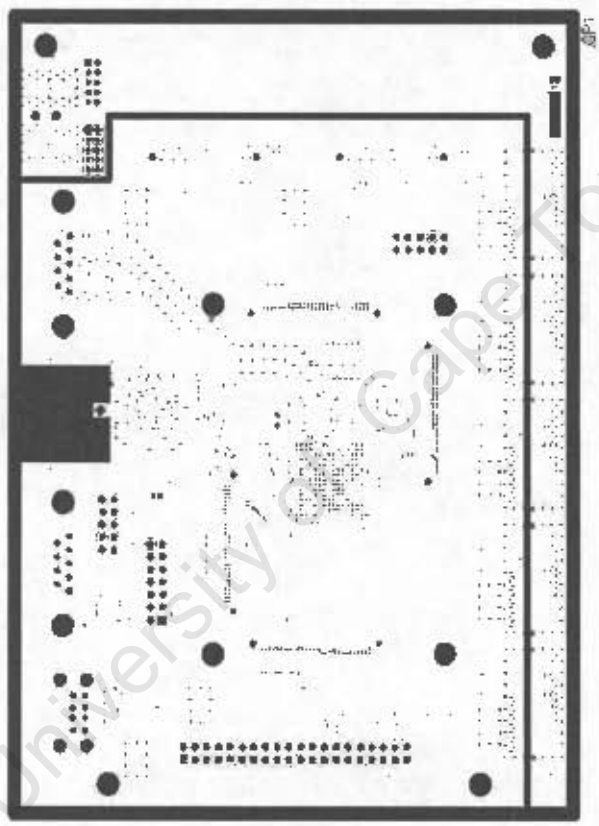

Figure 54 RUI Motherboard PCB Layout - +3.3V/V_{IN} Supply Plane

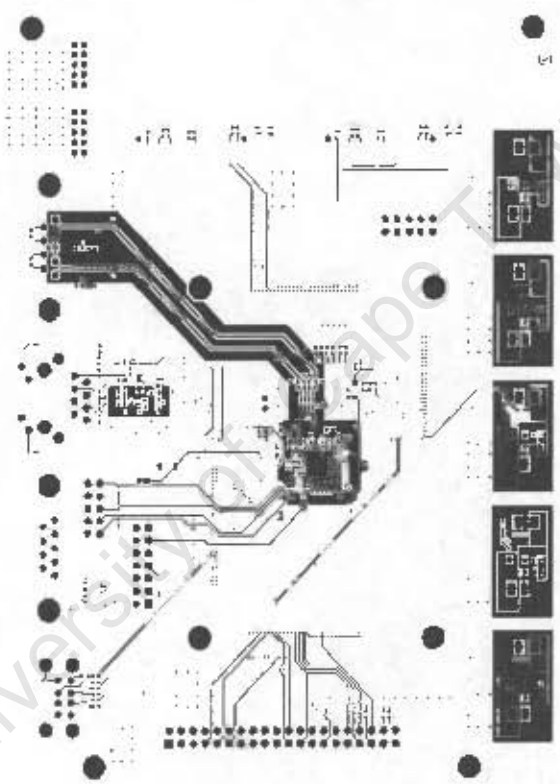

Figure 55 RUI Motherboard PCB Layout - Bottom Signal Layer

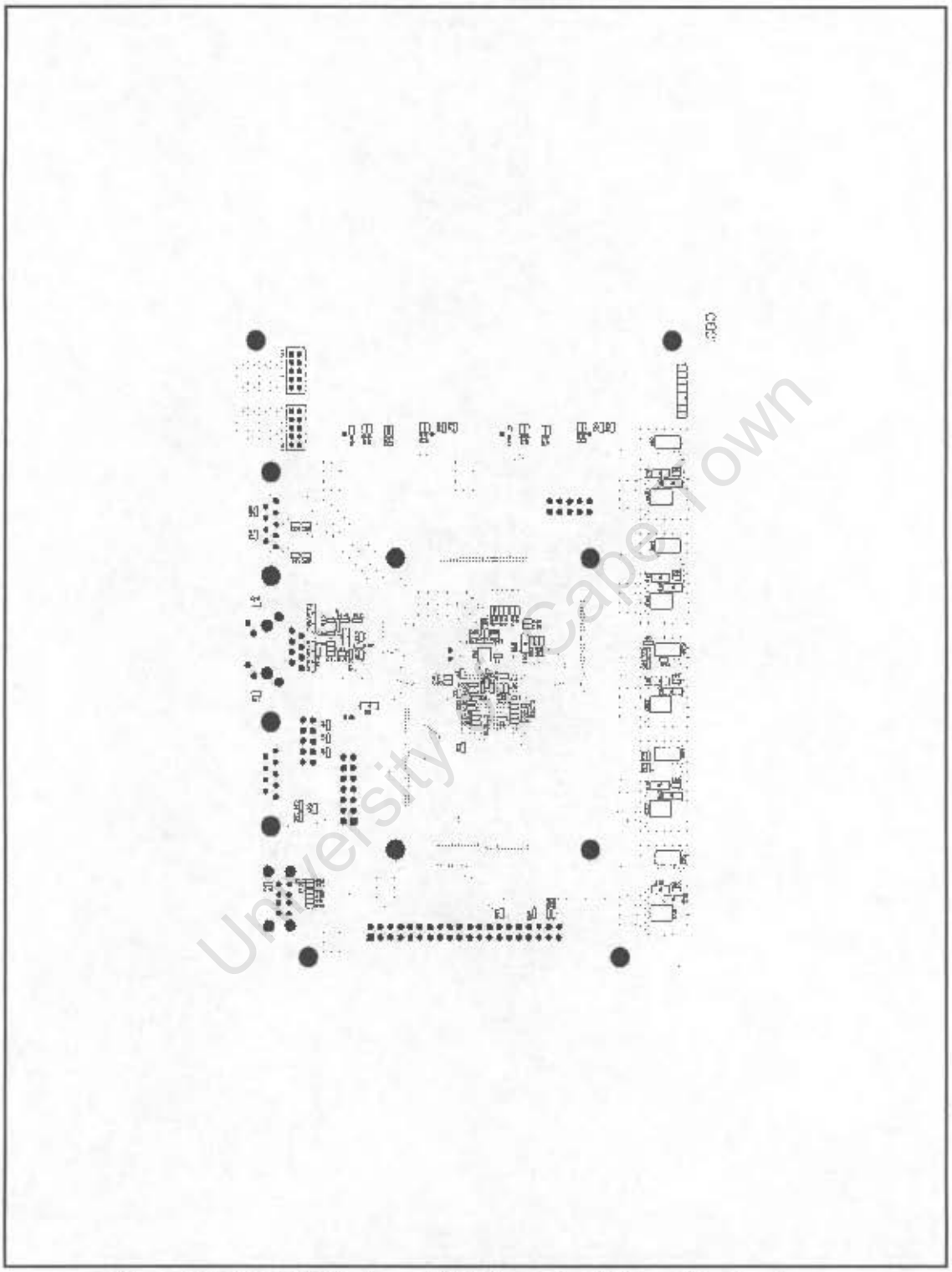

Figure 56 RUI Motherboard PCB Layout - Bottom Overlay Layer

Appendix C

Software and Firmware Source Code Listings and Information

C.1 FPGA Hardware Configuration Firmware Listing

```
library ieee;
use ieee.std_logic_1164.all;

entity RUI is
port(
        nReset, JTAG_nReset, RTC_nReset                          : in    std_logic;
        SBUS_nRBEN0, SBUS_nRBEN1, SBUS_nRCS0, SBUS_nRWE          : in    std_logic;
        FLASH_Status, FPGA_Clock                                 : in    std_logic;
        AUP_GPIO1, AUP_GPIO7, AUP_GPIO19, AUP_GPIO22             : in    std_logic;
        AUP_GPIO30, AUP_GPIO31, AUP_EXTCLK1                      : in    std_logic;
        Switch                                  : in    std_logic_vector(5 downto 0);
        AUP_nRESET, ETH_nRESET, FLASH_nReset                     : out   std_logic;
        FLASH_nRCS, FLASH_nRWE, RTC_I2C_SCL, SBUS_nEWait         : out   std_logic;
        AUP_GPIO14, AUP_GPIO15, AUP_GPIO201, AUP_GPIO202         : out   std_logic;
        AUP_GPIO203, AUP_GPIO204, AUP_GPIO206, AUP_GPIO208       : out   std_logic;
        AUP_GPIO214                                              : out   std_logic;
        LED                                     : out   std_logic_vector(7 downto 0);
        AUP_GPIO29, RTC_I2C_SDA                                  : inout std_logic
);
end RUI;

architecture Hardware_Config of RUI is

signal Master_nReset, RML_Link                                   : std_logic;

begin
```

```
Master_nReset <= nReset and (RTC_nReset or Switch(4)) and (JTAG_nReset or Switch(5));
AUP_nReset <= Master_nReset;
ETH_nReset <= Master_nReset;
FLASH_nReset <= Master_nReset;

LED(0) <= Master_nReset;
LED(1) <= FLASH_Status;
LED(2) <= AUP_EXTCLK1;
LED(3) <= RML_Link;              --RML Firmware uses LED3 for RML Link Indicator
RML_Link <= '1';                 --Force Link Indicator Off
LED(4) <= AUP_GPIO1 and Switch(0);
LED(5) <= AUP_GPIO7 and Switch(1);
LED(6) <= AUP_GPIO19 and Switch(2);
LED(7) <= AUP_GPIO22 and Switch(3);

SBUS_nEWait <= '1';

FLASH_nRWE <= SBUS_nRBEN0 or SBUS_nRBEN1 or SBUS_nRWE;
FLASH_nRCS <= SBUS_nRCS0;

AUP_GPIO14 <= '1';               --Unused I/O set as outputs and driven high
AUP_GPIO15 <= '1';
AUP_GPIO201 <= '1';
AUP_GPIO202 <= '1';
AUP_GPIO203 <= '1';
AUP_GPIO204 <= '1';
AUP_GPIO206 <= '1';
AUP_GPIO208 <= '1';
AUP_GPIO214 <= '1';

I2C : process (Master_nReset, FPGA_Clock)
        begin
                if Master_nReset = '1' then
                        RTC_I2C_SCL <= '1';
                        RTC_I2C_SDA <= 'Z';
                        AUP_GPIO29 <= 'Z';
                elsif rising_edge(FPGA_Clock) then
```

```
                    RTC_I2C_SCL <= '1';
                    RTC_I2C_SDA <= 'Z';
                    AUP_GPIO29 <= 'Z';
                    if AUP_GPIO31 = '0' then
                            RTC_I2C_SDA <= AUP_GPIO29;
                    else
                            AUP_GPIO29 <= RTC_I2C_SDA;
                    end if;
                    RTC_I2C_SCL <= AUP_GPIO30;
            end if;
    end process;

end Hardware_Config;
```

C.2 Bootloader Download Procedure

C.2.1 Hardware and Software Requirements

To install the MicroMonitor bootloader on the RUI hardware you will need the following hardware and software.

- Macraigor Systems LLC or compatible Wiggler JTAG programmer
- Macraigor Systems LLC OCD Commander software (Available for free download at http://www.macraigor.com)
- RUI hardware
- Serial cable as outlined in **Appendix A.3**
- RUI power adapter (+7V to +12V @ 1A)
- Terminal emulator software similar to Hyper Terminal that is supplied with Windows
- MicroMonitor bootloader SDRAM file for the RUI hardware in ELF format
- MicroMonitor bootloader Flash memory file for the RUI hardware in BIN format
- Macraigor OCD Commander macro file for configuring and downloading the MicroMonitor bootloader file to the RUI SDRAM (The contents of the file is

shown in **Appendix C.3**.)

C.2.2 File Location Setup Procedure

All the files required for installing the MicroMonitor bootloader on the RUI hardware should be placed in the following location on the PC:

C:/RUI

If the files are in any other location then the procedure outlined below will not work. Another location can be used, but the Macraigor OCD Commander macro file must then be edited to reflect this new location.

C.2.3 RUI Hardware Setup Procedure

- Connect the Macraigor Systems LLC or compatible Wiggler JTAG programmer between the Au1100 SoC processor JTAG connector (J8) on the RUI and the PC parallel port (generally LPT1).
- Note the placement of Pin 1 on the JTAG header (J8) and ensure that the Macraigor Systems LLC or compatible Wiggler JTAG programmer is connected correctly.
- Connect the serial cable between the RUI RS-232 connector (J7) and the PC serial port. Either COM1 or COM2 can be used as the serial port used will be defined in the terminal emulator software configuration in **Appendix C.2.5**.
- Connect the power adapter to the RUI hardware power connector (J10).
- Press the reset switch (SW2) on the RUI hardware.

C.2.4 Macraigor Systems LLC OCD Commander Software Configuration Procedure

- Start the Macraigor Systems LLC OCD Commander software.

- Select the configuration from the ***"Connection Dialog"*** window as shown in **Table 21.**

Table 21 Macraigor Systems LLC OCD Commander Software Configuration	
Target Processor	Alchemy - Au1100 (This configuration must be selected from two different drop-down menus.)
OCD Interface Device	Wiggler
Multiple	Deselected
Port Number	1 (This configuration is dependant on the parallel port the Macraigor Systems LLC or compatible Wiggler JTAG programmer is connected to.)
OCD Speed	1
Start with logging on	Deselected
Pop-up API errors	Selected

- Clip the ***"OK"*** button.
- The main Macraigor Systems LLC OCD Commander program screen should now be displayed.
- If at this point any error messages are display in the main program screen the Macraigor Systems LLC or compatible Wiggler JTAG programmer could not establish a connection with the RUI hardware. Repeat the procedures as described in **Appendix C.2.3** and **Appendix C.2.4.**

C.2.5 MicroMonitor Bootloader SDRAM Download

- Continue with the MicroMonitor bootloader SDRAM download while still in the OCD Commander program window.

- Click the ***"status"*** button (7th from the left above the output screen). The following message should be displayed in the output screen:

  >STATUS

  and on the next line

  Target RUNNING

  should be displayed.
- Click the ***"halt"*** button (4th from the left above the output screen). The following message should be displayed in the output screen:

  >Halt

- Click the ***"status"*** button (7th from the left above the output screen). The following message should be displayed in the output screen:

  >STATUS

  and on the next line

  In DEBUG Mode

  should be displayed.
- Select the ***"Commands"*** followed by the ***"Macro"*** option from the drop-down menus at the top of the screen.
- Go to the location of the Macraigor Systems LLC OCD Commander macro file and select the macro file. The location of the file was specified in **Appendix C.2.2.**
- At this point the macro will execute and the download will start.

- An indication that the download is taking place are sequential "." (period) characters being displayed on the last line of the output screen. If this does not happen then terminate the Macraigor Systems LLC OCD Commander software and repeat the procedures as described in **Appendix C.2.3** and **Appendix C.2.4**.
- At the end of the download procedure an error message should be displayed which can be ignored.
- Proceed to the next section without terminating the Macraigor Systems LLC OCD Commander software.

C.2.6 Terminal Emulator Software Configuration Procedure

- Start the terminal emulator software.
- Setup a serial connection with the parameters shown in **Table 22**.

Table 22 Terminal Emulator Software Configuration Parameters

Baud Rate	38400
Data Bits	8
Parity	None
Stop Bits	1
Handshaking	Hardware

- Define the serial port as the one the RUI hardware is connected to as described in **Appendix C.2.3**.
- Proceed to the next section without terminating the terminal emulator software.

C.2.7 MicroMonitor Bootloader FLASH Memory Programming

- Return to the Macraigor Systems LLC OCD Commander software.
- Click the ***"go"*** button (3rd from the left above the output screen). The following

message should be displayed in the output screen:

>Go

- Proceed to the next point without terminating the Macraigor Systems LLC OCD Commander software.
- Return to the terminal emulator software.
- Some text should be displayed in the terminal emulator window similar to the example below:

TFS Scanning //Flash_Block_A/...
TFS Scanning //Flash_Block_B/...
MICRO MONITOR
CPU: AMD Au1100
Platform: RUI
Built: Feb_09,2004 @ 12:08:18
Monitor RAM: 0xa002a000-0xa0047434
Application RAM Base: 0xa0300000
uMON>

- If this does not happen then either the serial connection is not correctly configured or the download process has failed. If this does not happen then terminate the Macraigor Systems LLC OCD Commander software and repeat the procedures as described in **Appendix C.2.3** and **Appendix C.2.4**.
- In the terminal emulator window type:

xmodem -B

(note that this is case sensitive) followed by the ***"Enter"*** key.

- At this point consecutive ***"§"*** characters should start appearing on the screen.

- Send the MicroMonitor bootloader file for the RUI FLASH memory to the RUI hardware using the terminal emulator software. Note that the protocol specified for this procedure must be ***"XMODEM"***.
- Wait until the download has completed.
- The following message or something similar should be displayed in terminal emulator window once the download has completed:

  Rcvd 1353 pkts (173184 bytes)
  Reprogramming boot @ 0xbfc00000 from 0xa0300000, 173184 bytes.
  OK?

- Press ***"Y"*** to program the bootloader into the RUI Flash memory.
- The Flash status debug LED (D2) on the RUI hardware should come on for a few seconds (on average 10 seconds) and then go off.
- This message should be followed by the bootloader rebooting and some text should be displayed in the terminal emulator window similar to the example below:

  TFS Scanning //Flash_Block_A/...
  TFS Scanning //Flash_Block_B/...
  MICRO MONITOR
  CPU: AMD Au1100
  Platform: RUI
  Built: Feb_09,2004 @ 12:08:18
  Monitor RAM: 0xa002a000-0xa0047434
  Application RAM Base: 0xa0300000
  uMON>

- At this point the MicroMonitor bootloader should have been successfully programmed into the RUI FLASH memory and should reside there even with the power removed.
- The test as to whether the MicroMonitor bootloader has been successfully

programmed into the RUI FLASH memory, involves cycling the power to the RUI hardware and seeing if the bootloader start-up sequence is displayed in the terminal emulator window followed by the *"uMON>"* prompt.

- If this happens then the MicroMonitor bootloader RUI Flash memory programming procedure had been successful.

C.3 Bootloader Configuration Macro

The Macraigor OCD Commander macro file is used for configuring and downloading the MicroMonitor bootloader file to the RUI SDRAM. This section lists the contents of the macro file. The last line of the macro file lists the location and filename of the MicroMonitor bootloader image to be downloaded into the SDRAM of the RUI with the Macraigor Wiggle JTAG programmer. This location and filename can be changed if required.

```
;Configure the CPU PLL to produce 392MHz CPU clock
word 0xB1900060 = 0x00000021
;Set the system bus divider to 3
word 0xB190003C = 0x00000001
;Configure the AUX PLL to produce 96MHz Auxiliary clock
word 0xB1900064 = 0x00000008
;Enable the 32kHz Oscillator
word 0xB1900014 = 0x00000100
;Configure the FLASH for 16-bit Big Endian operation
word 0xB4001000 = 0x00000243
word 0xB4001008 = 0x11F83FE0
;Configure the SDRAM Controller
word 0xB4000000 = 0x00552229
word 0xB400000C = 0x001003F8
word 0xB4000018 = 0x74000C30
word 0xB400001C = 0x00000000
```

```
word 0xB4000020 = 0x00000000
word 0xB4000020 = 0x00000000
word 0xB4000018 = 0x76000C30
word 0xB4000024 = 0x00000023
;Download bootloader to SDRAM
download C:\RUI\rui_ram
```

Appendix D

Building a MIPS32 Target GNU Cross-Platform Development Toolchain under Linux

D.1 List of Resources

The procedure that was developed during this design project for building the required GNU cross-platform development toolchain for the MIPS32 SoC processor under Linux was based on the information from the resources shown below.

- Yaghmour, K.: "Building Embedded Linux Systems", First Edition, O'Reilly & Associates Inc., Sebastopol, USA, pp. 107-155, 2003.
- LaRonde, B.D.: "Building a Modern MIPS Cross-Toolchain for Linux", Version 2.4, http://laronde.org/~brad/mips/mips-cross-toolchain/, 2001.
- A website of resources and information aimed at the Linux/MIPS development community: http://www.linux-mips.org
- The FTP site of the FSF and the location of the most recent components of the GNU toolchain: ftp://ftp.gnu.org/gnu/
- The FTP site of the Kernel.Org Organization, Inc. and the location of the Linux kernel archives: http://ftp.kernel.org/

D.2 Required Source Components

The components that were required in the building of the GNU cross-platform development toolchain for the MIPS32 SoC processor under Linux and its origins are shown below.

- binutils-2.13.90.0.10.tar.bz2 - Version from H.J. Lu with patch that had to be applied and available from http://ftp.kernel.org/pub/linux/devel/binutils/
- gcc-3.2-7.1.src.rpm - Patched version from H.J. Lu and only available in SRPM

format from ftp://ftp.linux-mips.org/pub/linux/mips/redhat/7.3/test/SRPMS/

- glibc-2.2.5.tar.gz - ftp://ftp.gnu.org/gnu/glibc/
- glibc-2.2.5-mips-build-gmon.diff - MIPS patch that had to be applied to glibc-2.5.5 and available from http://kegel.com/crosstool/current/patches/glibc-2.2.5/
- glibc-linuxthreads-2.2.5.tar.gz - ftp://ftp.gnu.org/gnu/glibc/
- linux-2.4.28.tar.gz - http://ftp.kernel.org/pub/linux/kernel/v2.4/

D.3 Toolchain Building Procedure

The GNU cross-platform development toolchain was built on an i686 host running Red Hat Linux 8.0 (Kernel 2.4.18) while being logged on to Linux as ***root*** (i.e. su). The first step was to make the project directory structure as shown below in the user's home directory. This project directory structure was as described by KarimYaghmour in *Building Embedded Linux Systems* (see Appendix D.1). With this project directory structure the toolchain was built in the ***~/mips/build-tools*** directory and installed in the ***~/mips/tools*** directory.

/ mips / bootldr

/ build-tools

/ build-binutils

/ build-boot-gcc

/ build-gcc

/ build-glibc

/ debug

/ doc

/ images

/ kernel

/ project

/ rootfs

/ sysapps

/ tmp

/ tools

At this point it is important to decide for what type of endian the toolchain must be built. MIPS processors can be configured for either big or little endian operation. If a big endian target is required then the term ***mips*** will be used as the top directory name and in all configuration commands. If a little endian target is required then the term ***mipsel*** will be used as the top directory name and in all configuration commands. These terms were used as they are recognized command line options for the GNU C-compiler. The previous project directory structure was thus obviously made for a big endian target. Any toolchain that will be built will also reflect the type of endian configuration in its name. A big endian tool in the toolchain will be named ***mips-linux-[toolname]*** and a little endian tool in the toolchain will be named ***mipsel-linux-[toolname]***.

The final part of the first step was to make a script file to set up the required environmental variables. This script file was as described by KarimYaghmour in *Building Embedded Linux Systems* (see Appendix D.1). The script file was placed in the user's home directory. The contents of the script file, called ***devmips***, is shown below.

export PROJECT=mips

export PRJROOT=/home/[username]/${PROJECT}

export TARGET=mips-linux

export PREFIX=${PRJROOT}/tools

export TARGET_PREFIX=${PREFIX}/${TARGET}

export PATH=${PREFIX}/bin:${PATH}

cd $PRJROOT

Take note that if a little endian target was required then ***mipsel-linux*** would have been used as the value for ***TARGET*** and the value of ***PROJECT***, the top directory name, would have been ***mipsel***. Also take note that ***[username]*** would be the name used when the user logged on to Linux. The script file has to be executed each time the user logs on to Linux and before the toolchain is used by typing the following on the command line in the user's home directory.

$. devmips

The second step was setting up the Linux kernel headers. This was done by copying the gzip-compressed kernel source tar file to the ***~/mips/kernel*** directory. It was extracted by typing the following on the command line.

$ tar xvzf linux-2.4.28.tar.gz

If the kernel source tar file was in a bzip2-compressed format then it would have been extracted by typing the following on the command line.

$ tar xvjf linux-2.4.28.tar.bz2

The kernel source file was extracted in a directory named ***~/mips/kernel/linux-2.4.28***. Next the kernel needed to be configured for the MIPS processor. This was done by changing to the ***~/mips/linux/linux-2.4.28*** directory and typing the following on the command line.

$ make ARCH=mips CROSS_COMPILE=mips-linux- menuconfig

Take note that if a little endian target was required then the ***ARCH*** option would have been ***mipsel*** and the ***CROSS_COMPILE*** option would have been ***mipsel-linux-***. This command displayed the menu for configuring the kernel. The most important configurations that had to be set was the processor and system type. The configuration was saved and the kernel configuration utility exited. The newly configured kernel headers had to be copied to the location required by the toolchain. This was done by typing the following commands on the command line.

$ mkdir -p ${TARGET_PREFIX}/include

$ cp -r include/linux/ ${TARGET_PREFIX}/include

$ cp -r include/asm-mips/ ${TARGET_PREFIX}/include/asm

$ cp -r include/asm-generic/ ${TARGET_PREFIX}/include

Take note that if a little endian target was required then the third command would have stayed exactly the same as the kernel configuration headers are the same for both big endian and little endian targets.

The third step was setting up the binary utilities. This was done by copying the bzip2-compressed binary utilities source tar file to the ***~/mips/build-tools*** directory. It was extracted by typing the following on the command line.

$ tar xvjf binutils-2.13.90.0.10.tar.bz2

The binary utilities source file was extracted in a directory named ***~/mips/build-tools/binutils-2.13.90.0.10***. Next a MIPS processor specific patch was applied. This was done by changing to the ***~/mips/build-tools/binutils-2.13.90.0.10/mips*** directory and typing the

following on the command line.

$ chmod 744 README

$ cd..

$ ./mips/README

The next procedure was to configure the binary utilities for cross-platform development. This was done by changing to the ***~/mips/buitd-tools/build-binutils*** directory and typing the following on the command line.

$ ../binutils-2.13.90.0.10/configure - -target=$TARGET - -prefix=${PREFIX}

Next we built the binary utilities by typing the following on the command line.

$ make

Finally the binary utilities were installed by typing the following on the command line.

$ make install

The fourth step was setting up the bootstrap compiler for compiling the C library in the next step. This was done by copying the bzip2-compressed GNU compiler source tar file to the ***~/mips/build-tools*** directory. This file was rebuilt from the original source RPM format file, named ***gcc-3.2-7.1.src.rpm***, by typing the following on the command line.

$ rpmbuild - -rebuild -v gcc-3.2-7.1.src.rpm

After the rebuild process, the bzip2-compressed GNU compiler source tar file will end up in the ***/usr/src/redhat/SOURCES*** directory from where it can be copied. The GNU compiler source tar file was extracted by typing the following on the command line.

$ tar xvjf gcc-3.2-20020903.tar.bz2

The GNU compiler source file was extracted in a directory named ***~/mips/build-tools/gcc-3.2-20020903***. The next procedure was to configure the bootstrap compiler for cross-platform development. This was done by changing to the ***~/mips/build-tools/build-boot-gcc*** directory and typing the following on the command line.

$ ../gcc-3.2-20020903/configure - -target=$TARGET - -prefix=${PREFIX}
- -without-headers - -with-newlib - -enable-languages=c - -disable-shared
- -disable-threads

Next we built the bootstrap compiler by typing the following on the command line.

$ make all-gcc

Finally the bootstrap compiler was installed by typing the following on the command line.

$ make install-gcc

The fifth step was setting up the C library. This was done by copying the gzip-compressed C library source tar files and the C library patch file to the ***~/mips/build-tools*** directory. The C library source tar files were extracted by typing the following on the

command line.

$ tar xvcf glibc-2.2.5.tar.gz

$ tar -xcvf glibc-linuxthreads-2.2.5.tar.gz - -directory=glibc-2.2.5

The C library source files were extracted in a directory named ***~/mips/build-tools/glibc-2.2.5***. The next procedure was to patch the C library source files. This was done by changing to the ***~/mips/build-tools/glibc-2.2.5*** directory and typing the following on the command line.

$ patch -p1 -i ../glibc-2.2.5-mips-build-gmon.diff

The next procedure was to configure the C library for cross-platform development. This was done by changing to the ***~/mips/build-tools/build-glibc*** directory and typing the following on the command line.

$ CFLAGS="-O2 -g -finline-limit=10000" CC=mips-linux-gcc
../glibc-2.2.5/configure - -host=$TARGET - -prefix="/usr" - -enable add-ons
- -with-headers=${TARGET_PREFIX}/include

Take note that if a little endian target was required then the ***CC*** option would have been ***mipsel-linux-gcc***. Next we compiled the C library by typing the following on the command line.

$ make

Finally the C library was installed by typing the following on the command line.

$ make install_root=${TARGET_PREFIX} prefix="" install

The final procedure in this step was to finalise the installation of the C library by modifying the link script located in the ***~/mips/tools/mips-linux/lib*** directory. Take note that if a little endian target was required then the location of the link script would have been located in the ***~/mipsel/tools/mipsel-linux/lib*** directory. This was done by first making a backup of the original link script as shown below.

$ cd ${TARGET_PREFIX}/lib

$ cp ./libc.so ./libc.so.orig

The original link script, named ***libc.so***, was modified by replacing the last line in the script file with ***GROUP (libc.so.6 libc_nonshared.a)***.

The sixth step was setting up the full compiler. This was done by changing to the ***~/mips/build-tools/build-gcc*** directory and typing the following on the command line.

$ ../gcc-3.2-20020903/configure - -target=$TARGET - -prefix=${PREFIX}
- -enable-languages=c,c++

Next we built the full compiler by typing the following on the command line.

$ make all

Finally the full compiler was installed by typing the following on the command line.

$ make install

The seventh and final step was to move the host binary utilities to a separate directory than that of the target binary utilities. This was done by typing the following on the command line.

$ cd ${PREFIX}/${TARGET}/bin

$ mv as ar gcc ld nm ranlib strip ${PREFIX}/lib/gcc-lib/mips-linux/3.2.1

Take note that if a little endian target was required then the command would have been ***$ mv as ar gcc ld nm ranlib strip ${PREFIX}/lib/gcc-lib/mipsel-linux/3.2.1***. Finally, symbolic links were made to the new location of the host binary utilities by typing the following script on the command line.

$ for file in as ar gcc ld nm ranlib strip

> do

> ln -s ${PREFIX}/lib/gcc-lib/mips-linux/3.2.1/$file .

> done

Take note that if a little endian target was required then the second line of the script would have been ***> ln -s ${PREFIX}/lib/gcc-lib/mipsel-linux/3.2.1/$file ..***

The full MIPS32 target GNU cross-platform development toolchain under Linux has now been built and can be used as required.