RECONFIGURABLE DATA ACQUISITION SYSTEM FOR WEATHER RADAR APPLICATIONS [1]

A Thesis Presented

by

RISHI KHASGIWALE

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

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

December 2004

Electrical and Computer Engineering

[1]This work is supported primarily by the Engineering Research Centers Program of the National Science Foundation under NSF award number 0313747. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

RECONFIGURABLE DATA ACQUISITION SYSTEM FOR WEATHER RADAR APPLICATIONS

A Thesis Presented

by

RISHI KHASGIWALE

ABSTRACT

RECONFIGURABLE DATA ACQUISITION SYSTEM FOR WEATHER RADAR APPLICATIONS

DECEMBER 2004

RISHI KHASGIWALE

B.E., MAULANA AZAD COLLEGE OF TECHNOLOGY, REGIONAL ENGINEERING COLLEGE (REC), BHOPAL, INDIA

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

Directed by: Professor Russell G. Tessier

Tornado, hazardous weather and flood detection radar systems demand high-throughput sustainable, high-speed data acquisition and processing systems. These systems need to be capable of implementing powerful signal processing algorithms in real-time on the raw data collected by these radars. Support for high-speed network facilities for real-time data dissemination to the end-user is expected for timely and accurate detection of imminent weather disasters. Designing a data acquisition and processing system to meet these requirements has been made feasible due to the recent developments in fields of networking and reconfigurable computing. This work describes a single circuit card, real-time acquisition and processing system with capabilities of real-time data distribution. Remote-host based control and remote-reconfigurability of the system further assist in meeting the guidelines of such applications. The goal of this research work is to provide the Distributed Collaborative Adaptive Sensing (DCAS) radars being developed by the Engineering Research Center (ERC) for Collaborative Adaptive Sensing of the Atmosphere (CASA) at the University of Massachusetts-Amherst with a powerful data acquisition and processing system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Data acquisition systems are an integral part of radar applications [26] [12] [22]. The raw analog data acquired by a radar system can be cast into meaningful, intelligible data packets only after it has been digitized and processed and can then be stored at suitable locations. All these processes should be fast enough to support high rate analog data inputs [13] and should be completed in real-time, especially for applications such as atmospheric sensing, involved in critical weather predictions such as tornado detection and warning [14]. The processed data requires bulk-storage devices for post-processing and analysis. In the event the storage resources are not accessible locally, a network interface is needed to access these remote bulk-storage devices. Further, these processes require various control signals and parameters, which necessitates an efficient, reliable communication system for the acquisition unit. All the above functions coupled together define the purpose and importance of a data acquisition system in a radar application.

In the past, the devices and components available for constructing a data acquisition system could not support high data rates, complex processing needs, large storage capacities and high-speed reliable network interfaces [7]. The problem was further aggravated due to the large physical dimensions and power consumption of the analog and digital components required. With technological advancements, high-speed devices with limited physical dimensions and excellent power performance are now available off-the-shelf. With the arrival of high speed Ethernet and Gigabit Ethernet interfaces, reliable, high-speed network support is also possible. All these improvements have made the development of a complete data acquisition system with high processing power and a low form-factor an achievable proposal.

The two most commonly used devices on a data acquisition system are Field Programmable Gate Arrays (FPGAs) or general purpose processors and microcontrollers [27]. A well-planned design with well-organized distribution of tasks between these devices paves the way for a state-of-the-art data acquisition architecture. Specialized hardware for high-speed data processing tasks and operations requiring fine-grained timing control can be effectively modeled by a combination of application specific hardware and a reconfigurable processor, which adds the dimension of reconfigurability to the architecture and allows for system flexibility. High-level functions that include administrating and synchronizing the working of all these devices can be achieved by the use of a microcontroller. Thus, an intelligent arrangement of application specific hardware and reconfigurable processors, administered by a microcontroller, as developed in this project, can create a highly effective and powerful data acquisition system.

For this thesis project, we have developed a data acquisition system to meet the requirements of an atmospheric monitoring radar application - the Distributed Collaborative Adaptive Sensing (DCAS) radars [13], being developed at the Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere (CASA) at the University of Massachusetts-Amherst. This radar system is being developed with the aim of timely prediction and detection of tornados, hazardous weather conditions and floods. The data acquisiton architecture has been implemented using off-the-shelf components such as ARM (Advanced RISC Machine) based Microcontroller and FPGAs. These two components comprise the main processing and controlling power of this architecture. A variety of communication interfaces have been incorporated in the design for the ease of controlling the system and routing and storing data at suitable locations. A careful distribution of processing requirements and communication interfaces between the microcontroller and the FPGAs has been designed. Radar data processing, network interface via a Gigabit Ethernet, low-level hard-disk control etc. have been assigned to the FPGAs. High-level operations like hard-disk storage handling, radar control network upload/download via Fast Ethernet, USB etc. has been confined to the ARM microcontroller.

Before fabrication, the functionality of the data acquisition system design was verified through simulations. To test the fabricated board, dummy FPGA algorithms and ARM diagnostic software were written to test various hardware interfaces on the board. The data acquisition board system and its various interfaces have been tested. This system now ready to be integrated with the DCAS radars and applied to a real-time environment.

This report is organized as follows. Chapter 2 describes previous work done in this field and the application of this design. The requirements and specifications set by the radar unit for this data acquisition system are listed in Chapter 3. Chapter 4 describes an architecture to meet these requirements. The configuration, functioning and operations performed by each section of this design and the system as a whole are discussed in detail in Chapter 5. The power filtering mechanism and the clock distribution scheme developed for this system are discussed in chapters 6 and 7 respectively. Chapter 8 pertains to the bootup strategy employed for different components included in this design. This will be followed with a discussion in Chapter 9, on the steps followed for designing and simulating the system. Chapter 10 lists the configuration strap settings needed to be performed on the hardware system. The testing and debugging strategy that was employed for the fabricated boards has been dealt with in chapter 11 followed by the results that were generated during this testing/debugging phase in chapter 12. This report concludes in Chapter 13, with a summary of the ideas and goals driving this project and a few suggestions on future tasks that can be implemented.

CHAPTER 2
BACKGROUND

2.1 Radar System Overview

The proposed data acquisition system will be integrated with the Distributed Collaborative Adaptive Sensing (DCAS) radars [13]. DCAS systems involve low-cost, small radars deployed on cell phone towers or other infrastructure and designed to observe and detect hazardous weather and floods. *Distributed* refers to the use of a dense network of radars capable of high spatial and temporal resolution. These systems will operate *collaboratively* within a dynamic information technology infrastructure, *adapting* to changing conditions in a manner that meets competing needs of end users; the government, private industry, and the public. Second generation DCAS systems will focus on volumetrically sensing wind and thermodynamic variables in a pre-storm environment, or clear air. A third instantiation of DCAS will involve mobile volumetric sensors that can communicate with ground-based sensors. Thus, studying and monitoring the lower troposphere, the part of the atmosphere that contains most natural and man made hazards, for accurate and timely tornado and flood detection and alarms is the goal of these radars.

For effective and timely weather hazard predictions, the data being collected by these radars need to be digitized, processed into packets of concise, succinct data and dispatched to the end user in real-time. These are the fundamental goals that drive the design of a data acquisition system for such radars.

The proposed system equipped with high-density, high-performance, fast, reconfigurable hardware (FPGAs) [3], an intelligent, embedded, feature rich, master-controller (Microcontroller) [9] and contemporary high-speed communication interfaces like Gigabit and 10/100 Mbps Ethernet, Universal Serial Bus (USB), Serial Ports and Integrated Drive Electronics (IDE), is ideally suited for the intended application. A high-level representation of a data acquisition system integrated in a radar unit is as shown in figure 2.1

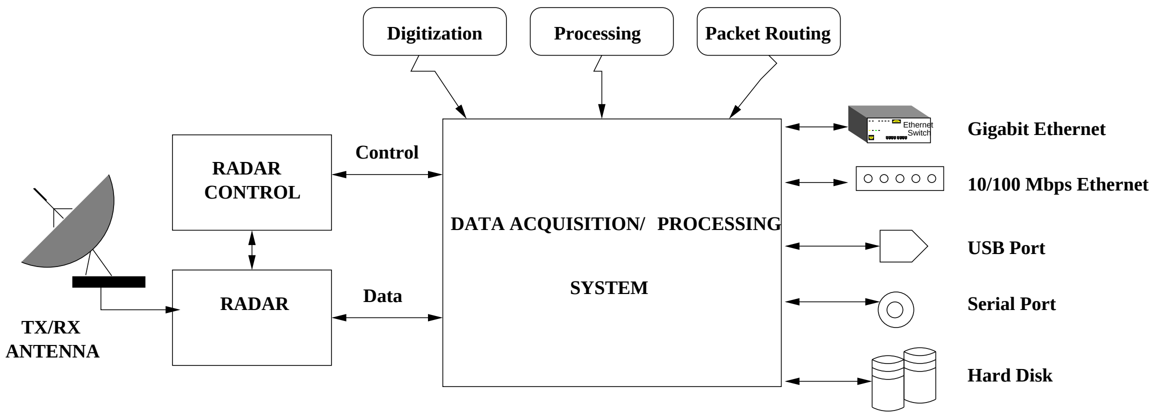


Figure 2.1. Data Acquisition board integrated with a radar unit - Block diagram.

2.2 Background work

Prior work on designing data acquisition systems for weather and ocean monitoring radar systems include [26] and commercial solutions like [15]. The system developed in [26] is equipped with reconfigurable hardware (FPGAs), processor based system control and has communication interfaces such as a 10/100 Mbps Ethernet, serial ports and IDE channels. The 12 bit 65 MSPS AtoD input channels implemented in this system do not meet the 14 bit 100 MSPS requirement [13] of this application. Further, processor speed (processor running at 50 MHz), memory and data output interfaces lack the ability to cope with the high data rates [13] and complex signal processing algorithms [17] required for implementation on bulk data inputs for this application. Commercial solutions like [15] have high-density, high-speed Xilinx Virtex-II FPGA modules. For converting analog inputs from a radar to the digital domain, this product needs to be integrated to daughter PCBs that add on to the cost and the complexity of the system. USB and parallel port interfaces are the only data output channels available. The data acquisiton architecture developed for this project costs about USD 6,240 per board. Thus, the commercial solutions are not only expensive (USD 15,500 for the basic system and an additional USD 1,100 for two A to D interface daughter boards) but also lack the superior performance and ease of control rendered by a microcontroller and the presence of high-speed communication interfaces, such as the Gigabit Ethernet interface, a necessity for the intended application.

CHAPTER 3
RADAR SYSTEM SPECIFICATIONS

3.1 Radar System

Data acquisition systems often interface with radar units to form a complete Ocean/Atmosphere monitoring radar system [26] [12] [16] [22]. Predicting tornados [13] is the main purpose of this application. For timely and effective tornado warnings the radar analog data needs to processed by a low latency (in the order of a millisecond) data acquisition system. Thus, real-time data processing is a must for the proposed acquisition system. Initial estimates state that the radar will generate data at a rate of about 100 Mbits per second [13]. For useful information to be extracted from the analog data, complex signal processing algorithms must be implemented to process the data [17]. The basic requirement is to develop a powerful, high-speed data acquisition system with abundant resources to support complex signal processing algorithms and provide standard, high-speed communication interfaces for processed data routing. From the user point of view, it should be a low-cost system made from off-the-shelf components with ease of control and programmability.

3.2 Radar Terminology Definitions

Before proceeding to the technical specifications and parameters set by the radar unit for this acquisition system, this section defines certain radar related technical terminologies.

1. Pulse Repetition Frequency (PRF): PRF is the frequency at which radar pulses are transmitted [26].

2. Range Gate: The transmitted pulses reflect back from the target and need to be sampled in the window between two transmitted pulses. The samples within the sampling window is called the Range Gates [26].

3. Range Resolution: Range resolution is defined as the distance between two consecutive radar samples [26].

3.3 Radar Specifications

Following are the specifications [13] set by the radar unit that have to be met by the proposed architecture.

1. Support real-time data acquisition, processing and storage.

2. Support data rates of about 100 Mbits per second, generated by the radar.

3. Support dual channel inputs to accept 10 MHz - 30 MHz IF signals.

4. Perform digital demodulation, filtering and decimation.

5. Perform digital coherent-on-receive magnetron phase correction.

6. Support pulse-pair and/or 128-point FFT algorithms, with a raw I and Q pass-through mode, for 300 - 600 range gate modes.

7. Support 100m and 200m range sampling modes.

8. Capable of averaging 1 - 10,000 pulses (non-coherent).

9. Support digital frequency drift compensation with accuracy of 2 kHz.

3.4 Data Acquisiton System Features

A list of features supported by this design with the objective of meeting the radar-defined specifications and requirements is given below. The block diagram of the proposed architecture is shown in figure 3.1.

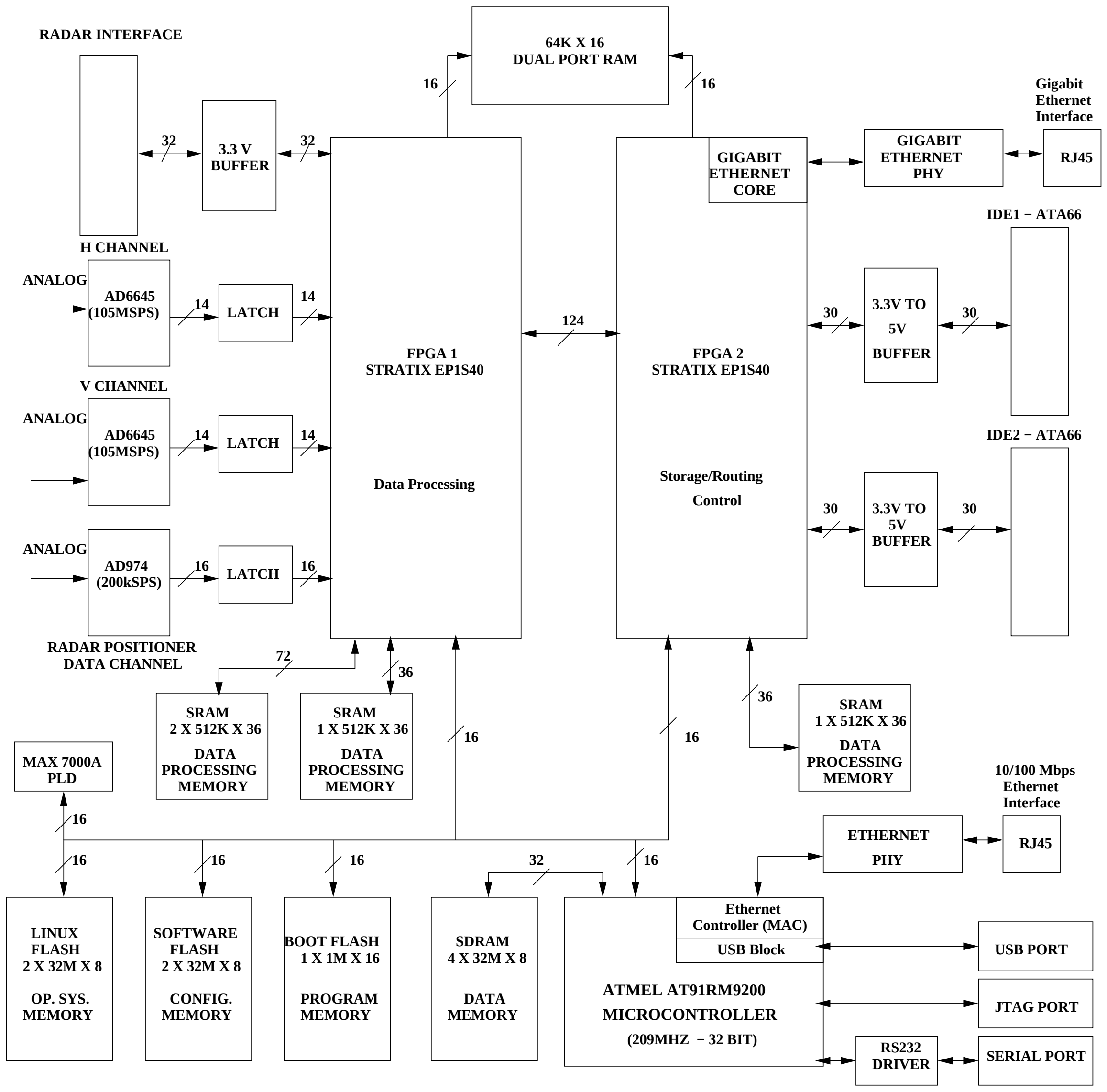

RADAR INTERFACE
3.3 V BUFFER
64K X 16 DUAL PORT RAM
GIGABIT ETHERNET CORE
GIGABIT ETHERNET PHY
RJ45
Gigabit Ethernet Interface
H CHANNEL
ANALOG
AD6645 (105MSPS)
LATCH
V CHANNEL
AD974 (200kSPS)
RADAR POSITIONER DATA CHANNEL
FPGA 1 STRATIX EP1S40
Data Processing
FPGA 2 STRATIX EP1S40
Storage/Routing Control
IDE1 − ATA66
IDE2 − ATA66
3.3V TO 5V BUFFER
SRAM 2 X 512K X 36 DATA PROCESSING MEMORY
SRAM 1 X 512K X 36 DATA PROCESSING MEMORY
MAX 7000A PLD
10/100 Mbps Ethernet Interface
ETHERNET PHY
LINUX FLASH 2 X 32M X 8 OP. SYS. MEMORY
SOFTWARE FLASH 2 X 32M X 8 CONFIG. MEMORY
BOOT FLASH 1 X 1M X 16 PROGRAM MEMORY
SDRAM 4 X 32M X 8 DATA MEMORY
Ethernet Controller (MAC)
USB Block
ATMEL AT91RM9200 MICROCONTROLLER (209MHZ − 32 BIT)
USB PORT
JTAG PORT
RS232 DRIVER
SERIAL PORT
32
14
16
124
30
72
36

Figure 3.1. Block Diagram of Data Acquisition System.

1. Two 14 bit 105 MSPS Analog to Digital (A to D) input data channels to support Horizontal (H) and Vertical (V) components of input data.

2. Additional 16 bit 200KSPS auxiliary A to D channel for radar positional data or other radar related input information.

3. Off-the-shelf, high speed, Altera Stratix FPGAs with abundant internal memory (41,250 Logic elements and 3,423,744 RAM bits), 14 DSP blocks and 112

9x9 bit multipliers [3] to support fast, powerful signal processing algorithms [17] such as Discrete Fourier Transform (DFT), Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters and Coordinate Rotation Digital Computer (CORDIC) for implementing coherence on receive algorithms. Reconfigurable hardware (FPGAs) imparts high performance and flexibility [27] to the system.

4. ARM embedded processor acting as the master control to the system adding to the ease of control and flexibility of the design.

5. On-board memory resources in the form of Flash chips, SDRAMs (Synchronous Dynamic Random Access Memory), SRAMs (Static Random Access Memory) and DPRAM (Dual Port Random Access Memory) to support complex signal processing algorithms on large data quantities.

6. Standard contemporary communication interfaces such as Gigabit Ethernet, 10/100 Mbps Ethernet, Universal Serial Bus (USB) port, Serial ports and IDE (Integrated Drive Electronics) channels.

7. An interface to the radar for control signals and sampling information.

8. Joint Test Action Group (JTAG) port and a dedicated Universal Asynchronous Serial Transmission (UART) debug channel to connect to a PC for testing and debugging the system.

9. Use of off-the-shelf, easily available, commonly used components.

In context to the specifications set-forth by the radar system in section 3.3, these resources shall meet the requirements as enumerated below.

1. Fast hardware components, intelligent control and abundance of memory resources will help support real-time data acquisition, processing and storage and

implement signal processing algorithms such as digital demodulation, filtering and decimation, digital coherent-on-receive magnetron phase correction and digital frequency drift compensation with ease.

2. Two A to D channels, each 14 bit, capable of supporting 105 MSPS, can support data-rates of upto (14 bit x 105 MSPS x 2 channels) 2.94 Gbps as compared to the 100 Mbps required by the radar system.

3. The A to D channels can support dual channel inputs to accept 5 MHz - 270 MHz IF signals [6], which exceeds the dual channel inputs to accept 10 MHz - 30 MHz IF signal requirement of the radar system.

4. This architecture can support modes of upto 1024 range gates with 256 point FFT (with 50% overlap) [17]. This exceeds the required 128-point FFT algorithms for 300 - 600 range gate modes expected by the radar system.

5. With an input bandwidth of 52.5 MHz, this system can support a minimum range of 3 meters (52.5 MHz = 19.04 ns. Total distance = 19.04 ns x speed of light = 5.71 meters. Thus, Range = 3 meters approximately). This exceeds the requirement of a minimum range of 100 meters set by the radar system.

6. This system is capable of averaging 1 to 2^{48} pulses, the upper limit set by a 48-bit accumulator utilized for the averaging. This is far better than the averaging of 1 - 10,000 pulses as expected by the radar unit.

Thus, we observe that this data acquisition system supersedes the requirements of the radar unit and is ideally suited for high data rate, real-time data acquisition and processing needs.

CHAPTER 4
DATA ACQUISITION SYSTEM ARCHITECTURE

The architecture of this board has been designed considering the requirements defined in Chapter 3.

The block diagram of the proposed architecture is re-drawn in figure 4.1. The two main processing modules of this architecture are the Stratix EP1S40 FPGAs [3], namely FPGA1 and FPGA2, which are being controlled and supervised by an Atmel AT91RM9200 [9] Microcontroller. FPGAs have been shown to be highly effective in performing time varying tasks such as data compression, digital signal processing and communication related operations [27]. For the proposed design, FPGA1 is the data processing center while FPGA2 is the final data product distribution center. The following discussion briefly explains the data and control flow through the proposed architecture.

4.1 Radar Interface and Analog Section

The data acquisition system has a radar interface that includes three analog data input channels, sampling clock input channel and a digital interface to feed sampling and control parameters to the system. The board consists of two 14-bit A to D channels (H and V), each of which can sample data at a maximum rate of 105 MSPS and form the two primary analog data input channels. The digital outputs of these A to D channels are latched before being fed to FPGA1 to reduce the critical path [23] and to compensate for board delays. A separate channel is utilized to provide the sampling clock for these A to D conversions. The design also has a 16-bit A to

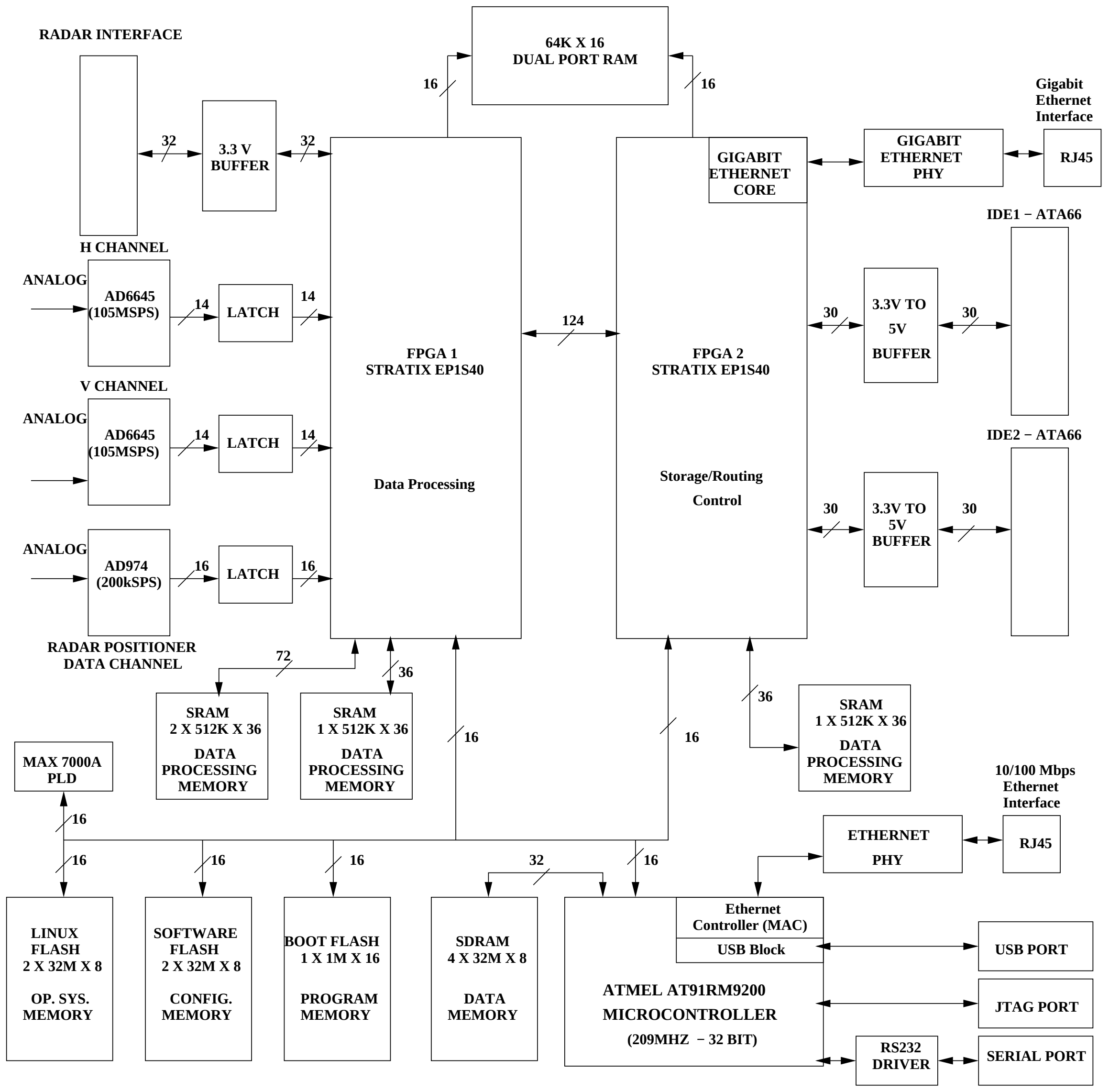


Figure 4.1. Block Diagram of Data Acquisition System.

D channel as the third analog input channel. This interface can sample data at a maximum rate of 200 KSPS. This is an auxilary analog channel that can be used to feed radar positioner data to FPGA1. This data can be used to control the positioning of the radar or can be used to add radar positioner information to data [13]. The H and V A to D channels and the sampling clock input channel form the all important, high-speed analog section of this architecture. Inorder to provide noise immunity to this section, the ground plane of the analog section has been isloated from the ground plane of the digital section. A differential transmitter-receiver combination between the A to D latched outputs and FPGA1 inputs provides a bridge between these two isolated sections of the design. Further, a 32-bit digital interface, a part of the radar interface, can be used to acquire data processing and control parameters from the radar and feed header data to FPGA1. This digital interface is isolated from FPGA1 using 3.3 V buffers.

4.2 FPGA1 and FPGA2

FPGA1, the data processing center of this design, receives sampled data over its differential interface to the A to D channels. Various signal processing algorithms, for radar data processing, can be built into FPGA1. A 72-bit high-bandwidth interface to SRAM memory modules assists FPGA1 in implementing high-speed, complex signal processing algorithms.

Both the FPGAs interface with a Dual Port RAM (DPRAM). The DPRAM acts as an intermediate memory module between them. FPGA1 stores a processed packet of data in the DPRAM while working on the next packet. FPGA2 can simultaneously read these processed packets out of the DPRAM for distributing over an output channel.

As mentioned above, FPGA2 is the data distribution control module. It selects one of the different available output channels and routes the processed data packets to the end user. It has a Gigabit Ethernet core [20] embedded in it along with two IDE hard disk controllers [26], in accordance to ATA5/ATA66 standards [29]. An external physical layer device is coupled between the embedded Gigabit Ethernet core and an external connector (RJ45) to form a Gigabit Ethernet interface. This interface, which can support a maximum data transfer rate of 1 Gbps [20], forms the processed-data-output channel for the proposed system. In the event of a network breakdown or

Gigabit Ethernet interface unavailability, the IDE interfaces can be used as a source for temporary data backup.

A 124 bit digital interface between the two FPGAs can be utilized to communicate data processing/routing related parameters or control signals between them. As in the case of FPGA1, FPGA2 also has access to external SRAM modules to facilitate large data storage capacities required especially during the execution of signal processing algorithms.

4.3 Microcontroller

The Atmel ARM based Microcontroller - AT91RM9200 [9] acts as the master control for the system. This chip, which can operate at a maximum speed of 209 MHz (200 MIPS at 180 MHz), is a high performance microcontroller solution for communication based, real-time embedded applications. The microcontroller controls all the operations being performed by the two FPGAs by programming their respective internal registers. It has a built in 10/100 Mbps Ethernet Media Access Controller (EMAC). A physical layer device (PHY) connected between the microcontroller and an external connector (RJ45) forms an Ethernet interface that can support a maximum data transfer rate of 100 Mbps. This interface can be utilized to download data for testing and debugging purposes and to feed control instructions to the system over the network. It also supports a variety of other standard communication interfaces such as a serial interface (via RS232 transceiver and external connector), a dedicated Universal Asynchronous Serial Transmission (UART) debug channel and a Universal Serial Bus (USB) interface, in addition to a standard JTAG (Joint Test Action Group) port. These interfaces can be used for a variety of communication, debug and emulation operations. The microcontroller has access to external FLASH and SDRAM (Synchronous Dynamic RAM) memory modules that act as program and data memories respectively. For the ease of controlling the microcontroller and supporting all available communication interfaces, an operating system [10] has been ported to the microcontroller.

An additional function of the microcontroller is to control the configuration process of the two FPGAs. This architecture supports two FPGA configuration schemes [4]. The FPGAs can either be configured through a dedicated JTAG interface or through an Altera Programmable Logic Device (PLD) [5]. Under the latter approach, a separate 'Software' FLASH is responsible for storing the FPGA configuration files. The PLD and the 'Software' FLASH are controlled by the microcontroller. Thus, by utilising the microcontroller based 10/100 Mbps Ethernet interface, an additional feature of network-based reconfiguration of the FPGAs can be attained.

CHAPTER 5

SYSTEM OPERATION

This chapter pertains to the different building blocks of this system, their operation and their functionality.

5.1 ARM Microcontroller

As mentioned in the earlier chapters, the microcontroller performs the function of supervising the operation of this system and is an important building block in this architecture. The internal architecture of the ARM microcontroller is shown in figure 5.1.

5.1.1 Introduction

This architecture uses AT91RM9200, an ARM based Microcontroller by Atmel [9]. This microcontroller is a complete system-on-chip built around the 32-bit ARM920T ARM processor that can operate at a maximum speed of 209 MHz (200 MIPS at 180 MHz) and has two separate 16-Kbyte each data and instruction caches. In addition to this processor, AT91RM9200 incorporates a rich set of system and application peripherals and standard interfaces in order to provide a single-chip solution for a wide range of compute-intensive applications that require maximum functionality at minimum power consumption at lowest cost. The AT91RM9200 incorporates a high-speed on-chip SRAM workspace, and a low-latency External Bus Interface (EBI) for seamless connection to whatever configuration of off-chip memories and memory-mapped peripherals is required by the application. In addition to the processor caches, the

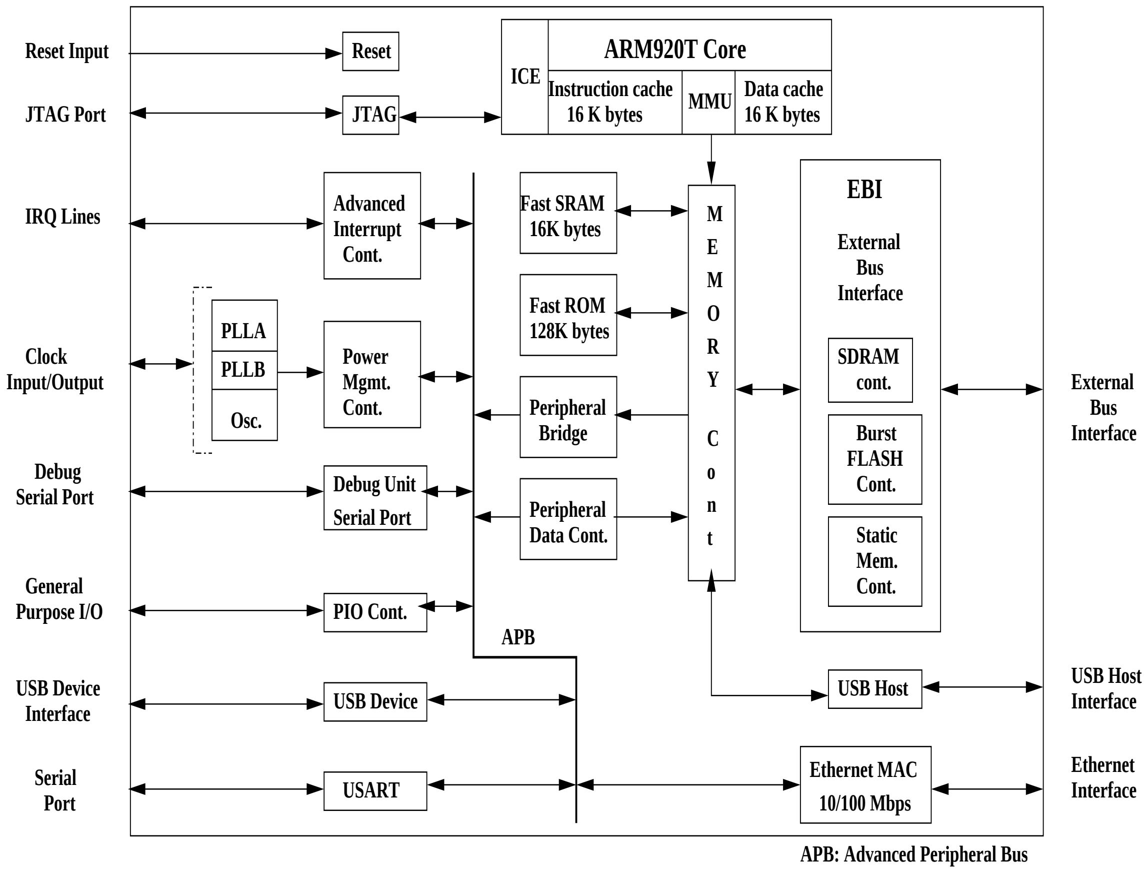


Figure 5.1. ARM Microcontroller Architecture.

chip has 16K Bytes of SRAM and 128K Bytes of ROM embedded into it. The EBI incorporates controllers for synchronous DRAM (SDRAM), burst Flash and static memories. The AT91RM9200 integrates a wide range of standard interfaces including USB 2.0 Full Speed (12 Mbits per second) Device, USB 2.0 Host (supporting both full as well as low speed protocols), Serial ports and Ethernet 10/100 Base-T Media Access Controller (MAC), which provides connection to an extensive range of external peripheral devices and a widely used networking layer. Further, debug features such as a JTAG-ICE interface and a dedicated UART debug channel enable the development and debug of all applications, especially those with real-time constraints. The following sub-sections discuss these modules and the working of the microcontroller.

5.1.2 Ethernet Media Access Controller (EMAC)

The AT91RM9200 EMAC is compatible with the IEEE 802.3 standard and can operate at 10 or 100 Mbits per second data throughput capability in half-duplex or full-duplex mode. An external physical layer device (PHY) coupled between this EMAC and an external connector (RJ45) forms a 10/100 Mbps Ethernet interface for the design. This EMAC can support a Media Independent Interface (MII) as well as a Reduced Media Independent Interface (RMII). In this design, the microcontroller operates in the MII mode to meet with the PHY specifications. The EMAC manages frame transmission and reception including collision detection, preamble generation and detection, CRC control and generation, transmitted frame padding, frame encapsulation and decapsulation and error detection. The RMII (MII) supplies the 50 MHz (25 MHz) transmission and reception clocks for 100Mbits per second operation to the PHY.

5.1.3 Universal Serial Bus (USB) Controller

The USB controller is compliant with the USB 2.0 specifications. The USB controller consists of a USB Host Port (UHP) (number of UHP ports is package specific, there is one UHP for the 208-lead PQFP package used for this design) and a USB Device Port (UDP) [9].

5.1.3.1 USB Host Port (UHP)

The USB Host Port interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as USB v2.0 Full-speed (12 Mbps) and Low-speed (1.5 Mbps) protocols. The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several high-speed half-duplex serial communication ports at a baud rate of 12 Mbit/s. Up to 127 USB devices (printer, camera, mouse, keyboard, disk, etc.) and the USB hub can be connected to the USB host in the USB tiered star topology [9]. Required drivers embedded into the operating system on the microcontroller help interface external USB devices to this system.

5.1.3.2 USB Device Port (UDP)

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) V2.0 full-speed (12 Mbps) device specification. It is designed to be associated with Atmels embedded USB transceiver and interfaced with the ARM9TDMI core. For this design, the signals of this port are available on an external header and can be utilized to generate a USB device interface daughter board for future applications.

Both the UHP as well as the UDP require a 48 MHz clock input, which is generated internally by the microcontroller.

5.1.4 Debug Unit (DBGU)

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of the microcontroller. The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. Moreover, the association with two peripheral data controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the ARM processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the ARM processor, making possible the handling of the DCC under interrupt control.

Finally, the Debug Unit features a Force Reset - 'NTRST' capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

5.1.5 Universal Synchronous Asynchronous Receiver Transceiver (USART)

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver timeout enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multi-drop communications are also supported through address bit handling in reception and transmission.

The USART is capable of managing several types of serial synchronous or asynchronous communications. Out of these different modes, the two main that have been implemented in this design are:

1. 5- to 9-bit full-duplex asynchronous serial communication:
   - MSB- or LSB-first.
   - 1, 1.5 or 2 stop bits.
   - Parity even, odd, marked, space or none.
   - By-8 or by-16 over-sampling receiver frequency.
   - Optional hardware handshaking.
   - Optional modem signals management.
   - Optional break management.
   - Optional multi-drop serial communication.
2. High-speed 5- to 9-bit full-duplex synchronous serial communication:
   - MSB- or LSB-first.

- 1 or 2 stop bits.
- Parity even, odd, marked, space or none.
- by 8 or by-16 over-sampling frequency.
- Optional Hardware handshaking.
- Optional Modem signals management.
- Optional Break management.
- Optional Multi-Drop serial communication.

5.1.6 Memory Map

The Memory Controller is responsible for decoding and mapping the on-chip address space to various embedded and external peripherals. The memory controller performs a first level of decoding, i.e., by the implementation of the Advanced System Bus (ASB) with additional features. This decodes the four highest bits of the 32-bit address bus and splits the 4G bytes of available address space into 11 separate areas.

1. One 256-Mbyte address space for internal memories.
2. Eight 256-Mbyte address spaces, each assigned to one of the eight chip select lines of the External Bus Interface (EBI), for interfacing with external chips.
3. One 256-Mbyte address space reserved for the embedded peripherals.
4. An undefined address space of 1536M bytes that returns an Abort if accessed, to assist in application debug.

Figure 5.2 shows the assignment of these memory areas.

5.1.6.1 Internal Memory Mapping

Within the Internal Memory address space, the address decoder of the Memory Controller decodes eight more address bits to allocate 1-Mbyte address spaces for

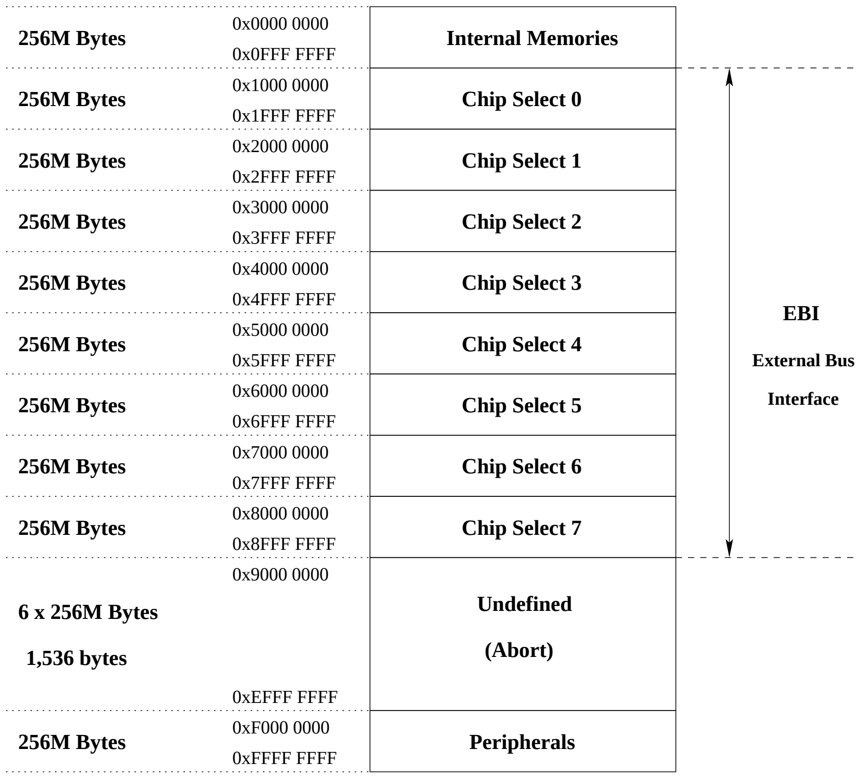


Figure 5.2. External Memory Areas.

the embedded memories, as show in figure 5.3. The allocated memories are accessed along the 1-Mbyte address space and are repeated n times within this address space, n equaling 1M byte divided by the size of the memory. When the address of the access is undefined within the internal memory area, i.e. over the address 0x0040 0000, the Address Decoder returns an Abort to the master.

At reset, depending on the state of the Boot Mode Select (BMS) input pin for the microcontroller, Area 0 in figure 5.3 points either to the on-chip ROM (BMS = 1) or the non-volatile external memory (BMS = 0) connected to chip select zero. After the Remap command, the internal SRAM at address 0x0020 0000 is mapped into Internal Memory Area 0.

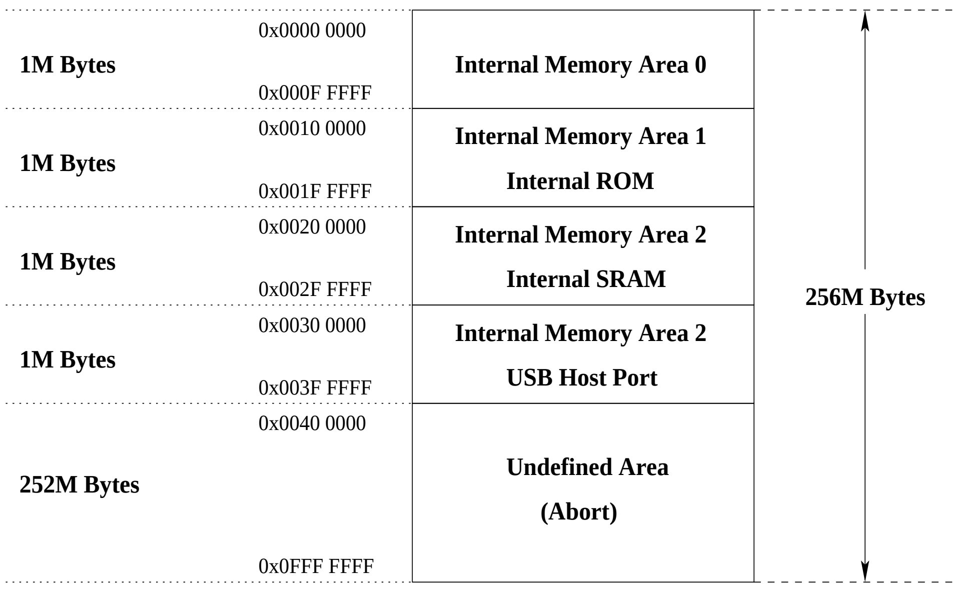


Figure 5.3. Internal Memory Map.

5.1.6.2 External Bus Interface (EBI)

The External Bus Interface (EBI) is designed to ensure successful data transfer between several external devices and the embedded Memory Controller. The Static Memory, SDRAM and Burst Flash Controllers are all featured external Memory Controllers on the EBI. These external Memory Controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash, SDRAM and Burst Flash. The 8 different banks with the associated chip select signals (NCS) are as follows:

1. Burst Flash Controller or Static Memory Controller on NCS0 (Chip Select Signal-0)
2. SDRAM Controller or Static Memory Controller on NCS1
3. Static Memory Controller on NCS3, Optional SmartMedia Support
4. Static Memory Controller on NCS4 - NCS6, Optional CompactFlash Support
5. Static Memory Controller on NCS7

Following list enumerates the components that have been connected to these chip-select signals on our developed architecture:

1. Boot FLASH on NCS0.
2. SDRAMs on NCS1 (also called as SDCS).
3. FLASH chips for storing operating system kernel (Linux FLASH) on NCS2.
4. FPGA1 on NCS3.
5. FPGA2 on NCS4.
6. FLASH chips for storing FPGA configuration files (Software FLASH) on NCS5.
7. A Programmable Logic Device (PLD) for FPGA configuration control on NCS6.
8. NCS7 is unused (routed to FPGA2 as a general purpose input/output pin).

5.1.6.3 Memory Allocation for Peripherals

The last block in the memory allocation in figure 5.2 is assigned to System and User Peripherals mapping. System peripherals include the Power Management Controller (responsible for generating the different clocks for the system), Memory Controller, Interrupt Controller etc. User peripherals include the USB host and device ports, Ethernet MAC etc.

5.1.7 Microcontroller Operation

In the proposed system, the microcontroller interfaces with the processing units - the two FPGAs and various communication ports. Some of the operations controlled include selecting and setting parameters for the signal-processing algorithms in FPGA1, adding header information and controlling data packet formation in it, initiating packet transfer from FPGA1's internal memory to the external DPRAM memory and further to FPGA2, and selecting the output path (Gigabit Ethernet port

Feature	EP1S40
LEs	41,250
M512 RAM blocks (32 x 18 bits)	384
M4K RAM blocks (128 x 36 bits)	183
M-RAM blocks (4K x 144 bits)	4
Total RAM bits	3,423,744
DSP blocks	14
Embedded Multipliers (9x9-bit)	112
PLLs	12
Maximum user I/O pins	822

Table 5.1. Altera Stratix EP1S40

or IDE interface) for the data in FPGA2. Further, its Ethernet, Serial and USB ports can be used for testing purposes as well as for downloading vital data that can be used by a remote user to control the functioning and operations of this system.

5.1.7.1 Operating System

Porting an operating system onto the microcontroller shall equip the user with the ease of programming and controlling the microcontroller and the data acquisition system. An AT91RM9200 dedicated, patched, ready-to-build Linux kernel 2.4.19, distributed by Atmel [10], has been ported to the microcontroller. The package provided by Atmel includes a set of pre-compiled tools that can be used to build both kernel and user applications. It also includes device drivers for the microcontroller communication interfaces.

5.2 Field Programmable Gate Arrays (FPGAs)

Integration of FPGAs endows the architecture with the added dimension of reconfigurability [27]. Two Altera Stratix - EP1S40 FPGAs have been employed as the processing powerhouses for the proposed system. Table 5.1 gives a brief insight into the resources that each of these chips offer to our system.

With the memory resources, DSP blocks, multipliers and high user I/O pin count these FPGAs are ideally suited for implementing signal processing algorithms and data packaging and routing applications.

5.2.1 FPGA1 Architecture

FPGA1 is the main processing unit of the system. A high-level view of its architecture is shown in figure 5.4. It accepts sampled data from the Analog to Digital converters (A to Ds), processes it and transfers the processed data to the external DPRAM. The control signals for sampling the data and processing it are obtained through the radar control interface. FPGA1 also acquires the header information that has to be added to every processed data packet, from either the radar control board or via its interface to the microcontroller. On-chip and system resources are utilized to implement data processing algorithms. FPGA1 implements the logic required to transfer these processed data packets and the header information to the external DPRAM. Data processing and transfer operations are sequenced and controlled by the microcontroller, by programming various FPGA1 registers. The microcontroller interface can also be used for downloading data for testing purposes via the microcontroller communication ports. A 124-bit digital interface between FPGA1 and FPGA2 can be utilized to transfer data or control parameters between them.

5.2.2 FPGA2 Architecture

FPGA2 has the responsibility of routing the processed data over a desired channel to the remote user, and thus implements two IDE Controllers and a Gigabit Ethernet core to provide these communication interfaces. It also has interfaces to the external DPRAM and the microcontroller. The architecture of FPGA2 and various interfaces and modules implemented are as shown in figure 5.5.

FPGA2 accepts processed data packets from the external DPRAM, places them in the requisite order and routes them to the end user by selecting an output channel

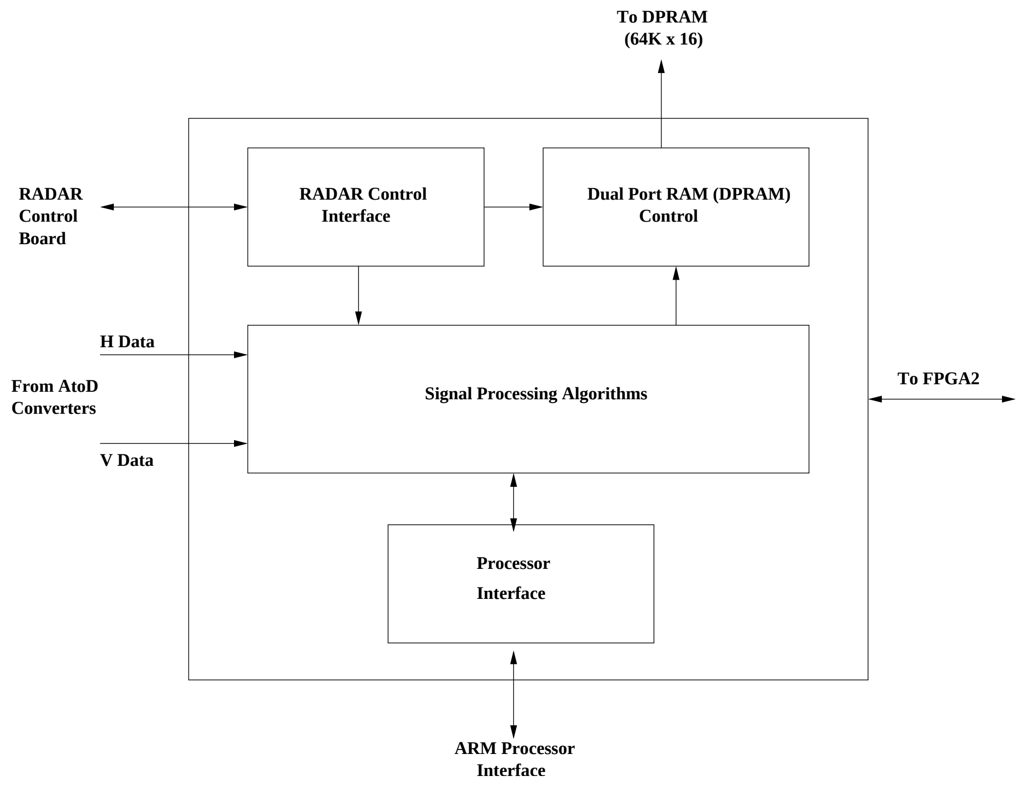


Figure 5.4. FPGA1 Architecture.

from among the two available. The Gigabit Ethernet channel can support a maximum data rate of up to 1 Gbps [20]. It can sustain the high output data rates (approximately 100 Mbits per second) [13] required by the system and forms the default communication channel for transferring processed data packets to the end user or storage device. The IDE channels offer a backup that can be utilized for temporary data storage in the advent of network problems and Gigabit Ethernet interface unavailability. The microcontroller, through its interface, has access to various status/control registers in FPGA2, and thus controls data acquisition and routing processes performed by it. This interface can also be used for downloading data for testing purposes via the microcontroller.

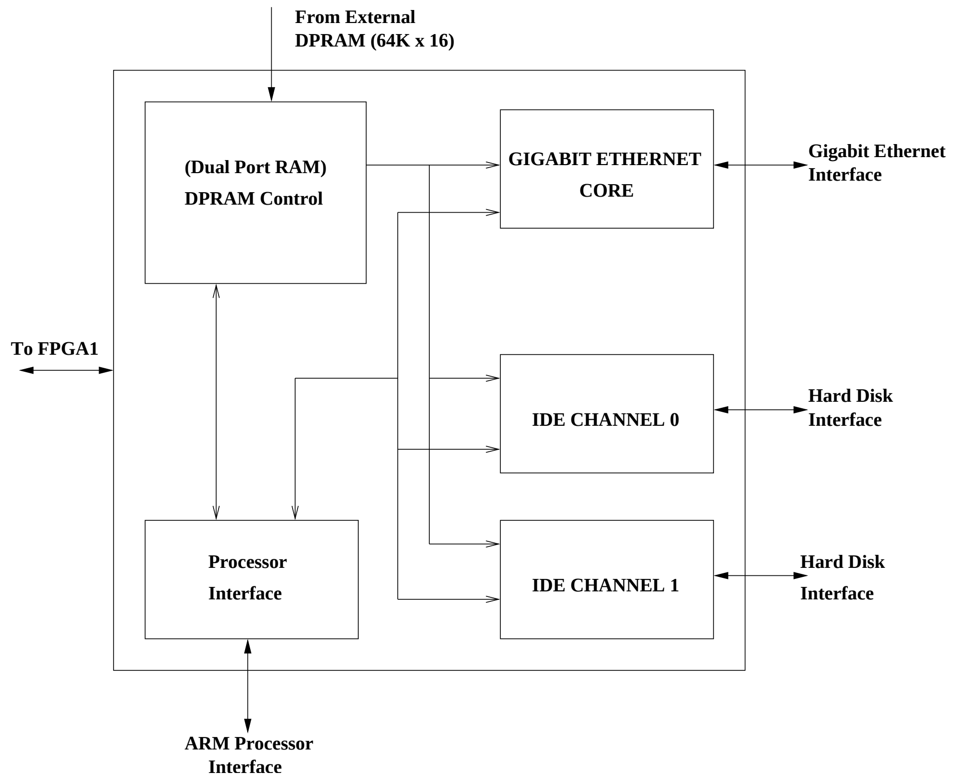


Figure 5.5. FPGA2 Architecture.

5.2.2.1 Gigabit Ethernet Core

FPGA2 implements the Mentor Graphics PE-GMAC0 Gigabit Ethernet MAC [20]. This core supports both the IEEE 802.3z Gigabit Ethernet specification and 802.3x Full Duplex Flow Control. It provides a 10-bit, 1 Gbps Gigabit Media Independent Interface (GMII), ideally suited for high-speed data routing requirements [13] of this application. An external physical layer device is coupled between the embedded Gigabit Ethernet core and an external connector (RJ45) to form the Gigabit Ethernet interface for this system.

5.2.2.2 IDE Controller

FPGA2 supports two IDE channels and implements an IDE controller developed by [26] for each channel. The IDE standard chosen is ATA5 [29], which can support two hard disks per channel and a maximum data rate of up to 66.67 MBps. Two such IDE-ATA5 channels provide an interface for four hard disk drives.

5.3 External On-board Memory Resources

The internal memory of the FPGAs and the microcontroller will not be sufficient to support the complex, high data rate processing and bulk storage requirements of this application [13], an operating system and a boot program for the microcontroller. So, the storage capacity of the system has been enhanced by providing external memory interfaces. Four SDRAM memory chips each of capacity 256 Megabits (32M x 8 bits), act as data memory for the microcontroller. The SDRAM memory space can be used if the internal memory is not enough for data storage. A Flash memory chip of capacity 16 Megabits (1M x 16 bits) functions as Program/Boot memory for the microcontroller. This is a non-volatile memory chip and can retain the boot-program when the board is powered down. These SDRAM and Flash components can be accessed only by the microcontroller. Since the storage space in the FPGAs is limited, external memory in the form of SRAMs and Synchronous Dual Port RAM (DPRAM) is provided to the FPGAs. FPGA1 has access to three SRAM chips each of capacity 18 Megabits (512K x 36 bits). FPGA2 has a single SRAM chip with a capacity of 18 Megabits (512K x 36 bits) for data storage. The DPRAM provided between FPGA's 1 and 2 has the advantage that FPGA1 can write a data word and FPGA2 can read it at the same time. The DPRAM has a storage capacity of 64K x 16 bits. To store FPGA configuration files, the board has two Flash chips named as 'Software' Flash with a capacity of 256 Megabits (32M x 8 bits) each. They can be accessed by the microcontroller via the Programmable Logic Device (PLD) to configure the FPGAs. The user can access these Flash chips via the microcontroller or through the JTAG interface to the PLD to store configuration files. The microcontroller also has access to two Flash chips named as 'Linux' Flash, with a capacity of 256 Megabits (32M x 8 bits) each, for storing the kernel and ramdisk of its operating system.

5.4 System Operation

This section summarizes the path followed by data through the system shown in figure 4.1.

Analog data is fed to the system by the radar through the H and V data channels. The Analog to Digital converters digitize this data and supply it to FPGA1. FPGA1 also receives sampling information and control parameters through its interface to the radar control board. In accordance to the instructions given by the microcontroller, FPGA1 processes this raw digitized data by implementing signal processing algorithms and forms processed data packets. These packets are transferred to FPGA2 via the external DPRAM, a process controlled again by the microcontroller. Depending on the path selected for the header bits, they are added to the data packets either in FPGA1 or FPGA2 to form the final data product. The microcontroller now instructs FPGA2 to route these data packets to the end-user via the Gigabit Ethernet interface or to the IDE hard disks for temporary backup in the event of a network breakdown.

The interfaces that the user can use to control the operation of this system are the radar interface to FPGA1 and the USB, 10/100 Mbps Ethernet and Serial port interfaces to the microcontroller.

CHAPTER 6

POWER SUPPLY FILTERING AND DISTRIBUTION MECHANISM

The data acquisition architecture consists of a very sensitive analog section. The A to D channels are highly susceptible to noise and interference especially from the power supply. As per the device specifications the Signal to Noise Ratio (SNR) of the A to Ds is about 75 dB [6]. The processing that would be done in FPGA1 helps gain about 17 dB. Therefore, the total SNR for the processed data is about 92 dB. The A to Ds can accept a maximum signal of +5dBm [6]. Thus, for a 1 MHz bandwidth signal, the noise floor is at -87 dBm (5 - 92). The A to Ds have a Power Supply Rejection Ratio (PSRR) of $\pm$1.0 mV [6] which is equivalent to 60 dBm. Thus, as shown in figure 6.1, noise levels up to -30 dBm (approximately) can be filtered out by this system. But any noise or interference greater than -30 dBm will be considered as a signal by these A to D converters. -30 dBm being equivalent to 0.001 milliwatts, demonstrates that our A to D channels are very sensitive to the input power supply noise. This emphasizes the need for clean power sources for the analog section of the design and hence and extensive power filtering scheme.

In order to achieve noise immunity and isolate the power supplies to different sections of this architecture, an elaborate power supply distribution and filtering mechanism [18] has been developed for this design. Further, external power-lugs included in this design, provide the added feature of powering-up the system using external power-supplies. This chapter discusses this power filtering and distribution mechanism.

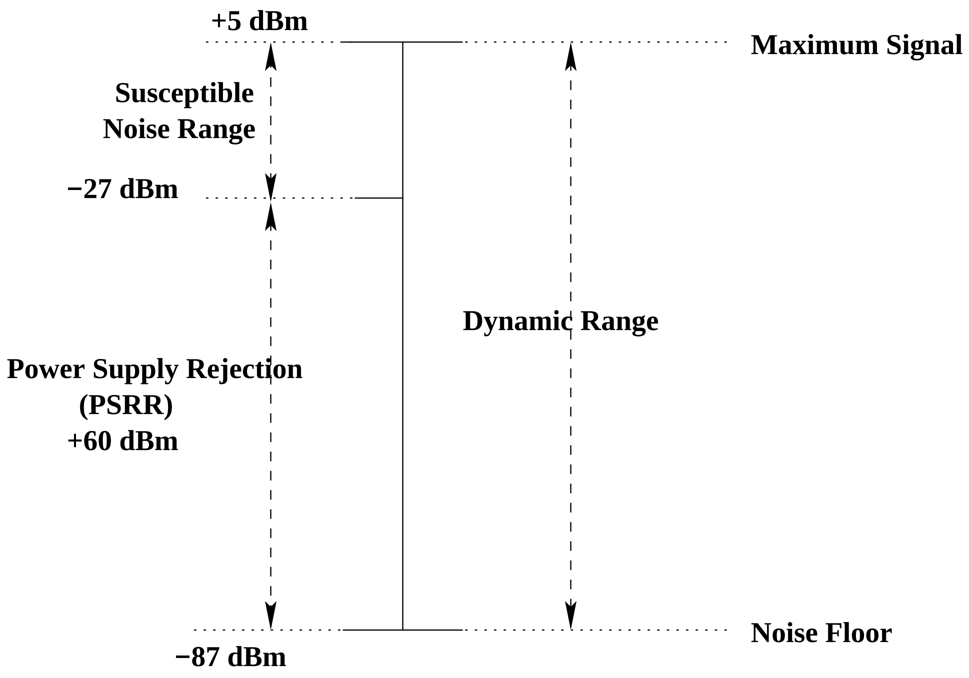


Figure 6.1. A to D Converter Specifications

An off-the-shelf ATX power supply unit powers an ATX connector on this architecture. Figure 6.2 depicts the voltages generated and the filtering implemented for the ATX power supply connector. From the available ATX voltages, we utilize the +12V (V+12), +5V (V+5) and the +3.3V (V+3_3) and the ATX connector ground (ATX_GND) for our design. When passed through the filters, they generate V12, V3_3, V5 and digital ground - GND respectively. More information can be gained and the actual filtering circuitry viewed from pages 3 and 11 of the schematics included in appendix A of this document.

These voltage sources are then utilized to generate the voltages required for different hardware modules included in the analog and digital sections of our architecture. The power supplies for the analog and digital sections, and for the two Analog to Digital conversion channel circuits within the analog section- are generated separately for isolation and noise immunity.

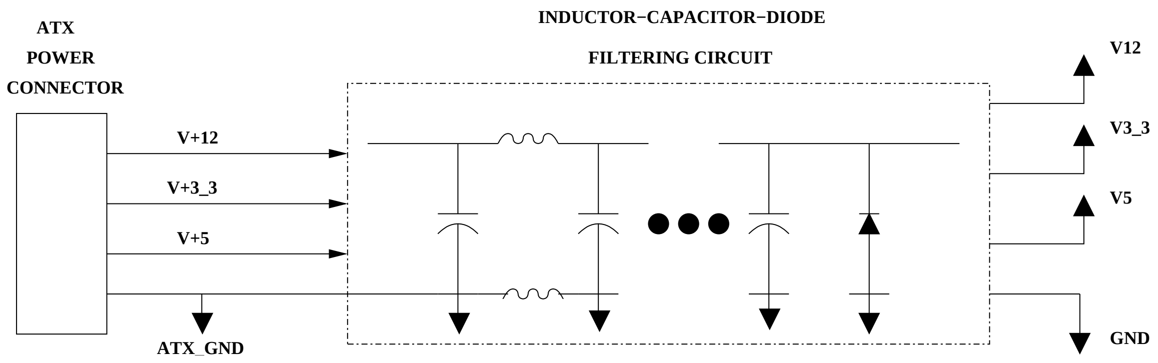


Figure 6.2. ATX Power filtering.

6.1 Power Supply filtering for Digital section

3.3V generated in figure 6.2 is required by the FPGAs, microcontroller and other hardware modules included in the digital section of the design. Figure 6.3 shows the filter circuitry to generate other voltages required by the modules included in the digital section. Two switching regulators are implemented to generate 1.5V (V1_5), 2.5V (V2_5) and 1.8V (V1_8) from V5 of figure 6.2. 1.5V is supplied to the FPGAs while the microcontroller requires a power supply of 1.8V in addition to the 3.3V.

Some sensitive components in the digital section such as the FPGA PLLs and the Gigabit Ethernet core transmit and receive interfaces, require a clean power supply. In order to maintain isolation between analog voltages, these analog voltages are independent to the analog voltage supply for the analog section of the design. As shown in figure 6.3, analog 2.5V (AV2_5) and analog 1.8V (AV1_8) required by the Gigabit Ethernet Physical layer device, and the analog 1.5V (VA1_5) required by the FPGA PLLs, are generated from digital 2.5V (V2_5), digital 1.8V (V1_8) and digital 1.5V (V1_5) respectively, utilizing separate filtering circuitries. Further, the radar positioner data input channel requires an analog 5V supply and an analog ground. Since, this is a slow (200 kSPS) analog to digital channel, the 5V analog (AV5_M) for this analog to digital converter is generated from the 5V digital (V5) supply employing a filter, and its analog ground is same as the digital ground (GND) used for all the modules included in the digital section of the design. More information on

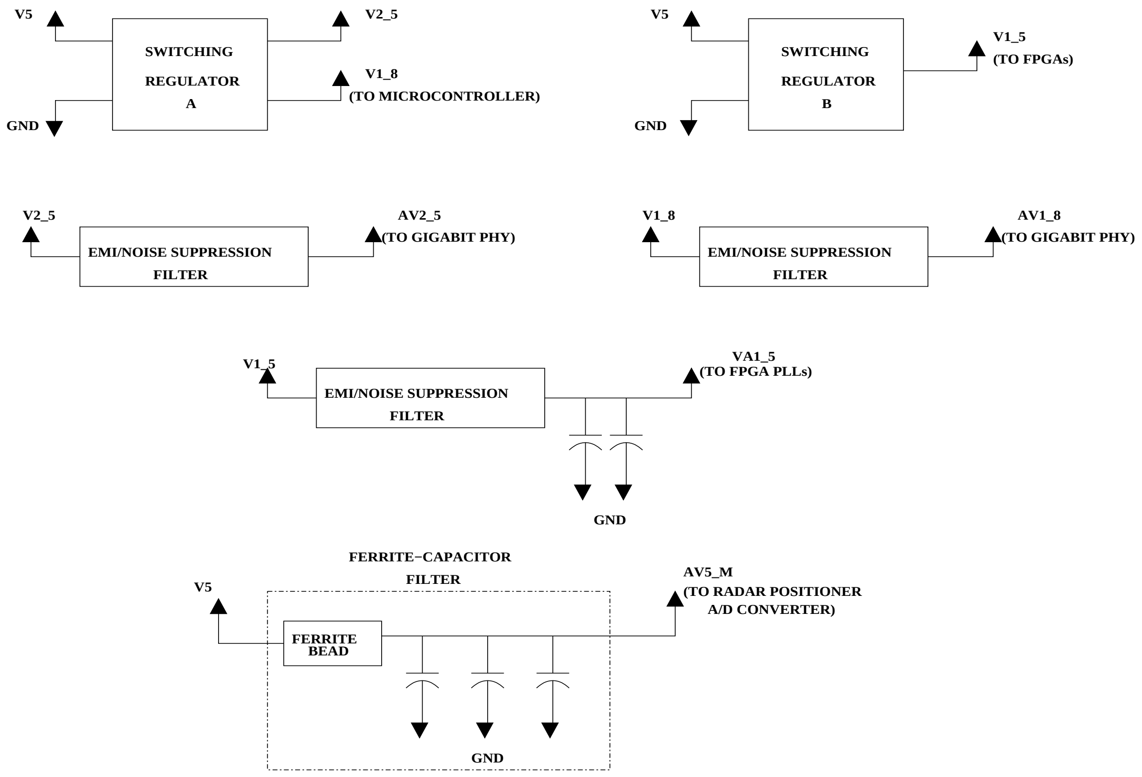


Figure 6.3. Power filtering for Digital section.

these issues can be obtained from page 11 and page 25 of the schematics included in appendix A.

6.2 Power Supply filtering for Analog section

As shown in figure 6.4, a step-down switching regulator is implemented to generated an intermediate 6.5V (V6_5) from the 12V (V12) shown in figure 6.2. This switching regulator has an INHIBIT input, which when pulled to ground, inhibits the output of this switching regulator and severs the power supply to the analog section of the design. This feature is helpful while testing/debugging the architecture. A filter is applied to this intermediate 6.5V and digital ground - GND, to generate three subsidiary 6V supplies namely F6VA, F6VB and F6VC and an intermediate ground FGND.

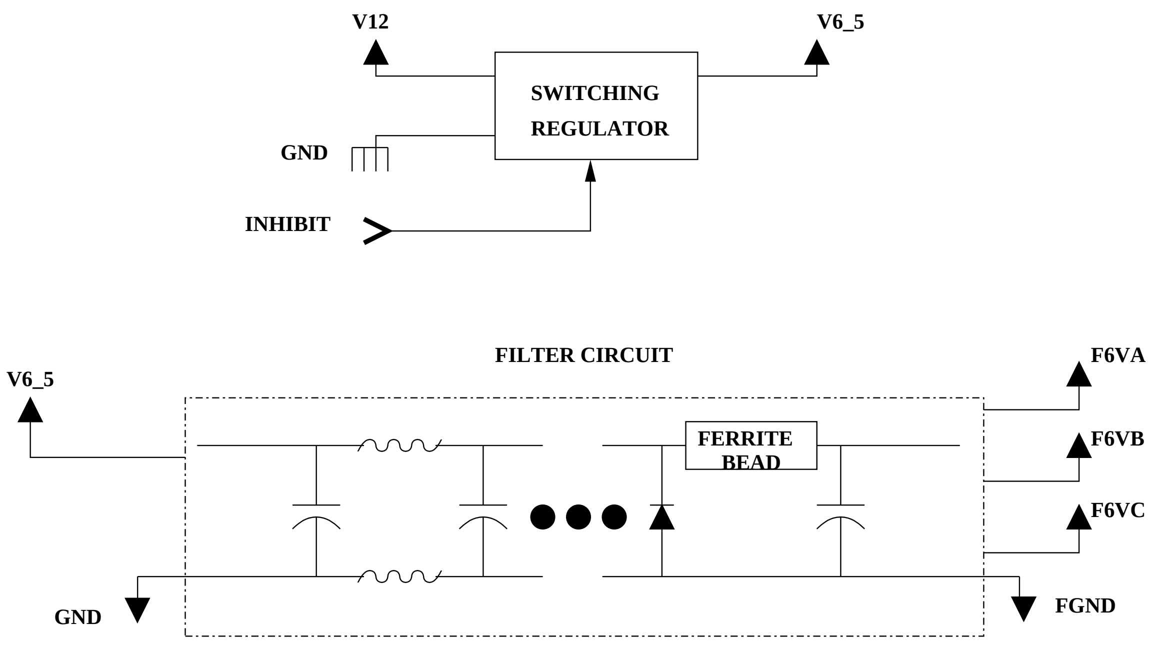


Figure 6.4. Power filtering for Analog section.

These three subsidiary analog 6V supplies when passed through a voltage regulator and filter circuitry generate voltages for three sections of the analog circuitry namely, analog 5V and analog 3.3V for each of the two analog to digital channels and a digital 3.3V for the digital components in the analog section of the design. These voltages shown in figure 6.5 have been listed below:

- analog 5V → AV5H and analog 3.3V → V3_31H for the H-channel.
- analog 5V → AV5V and analog 3.3V → V3_31V for the V-channel.
- digital 3.3V → V3_32 for the digital components of the analog section of this design.

Further, analog ground AGND1 is generated from intermediate ground FGND by implementing an inductor-ferrite bead filter circuitry as shown in this figure.

Isolating the voltage supply to the analog section from the digital section and further separating the power supplies to the H and V channels of the analog circuitry

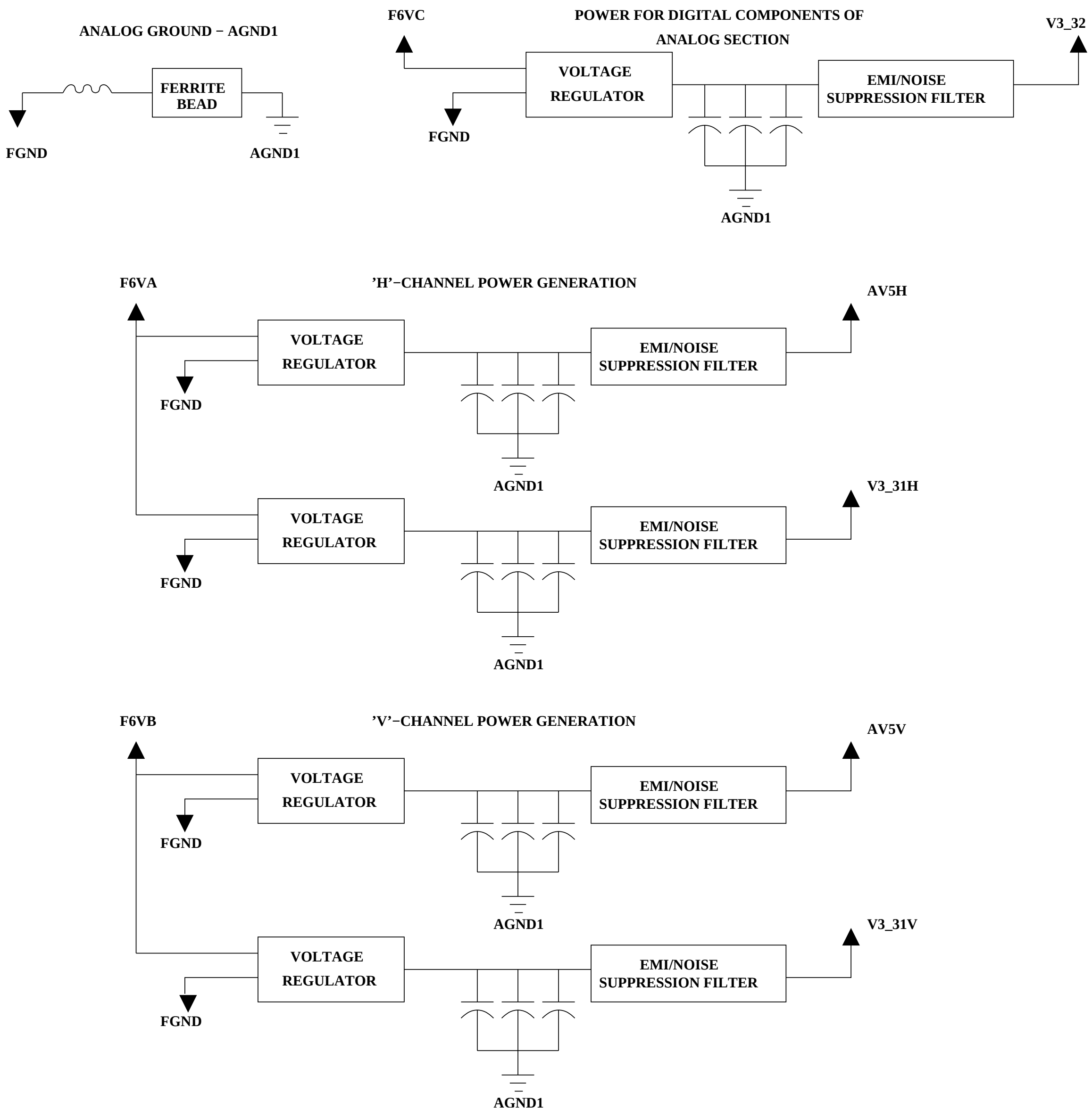

ANALOG GROUND − AGND1
FERRITE BEAD
FGND
AGND1
F6VC
POWER FOR DIGITAL COMPONENTS OF ANALOG SECTION
V3_32
VOLTAGE REGULATOR
EMI/NOISE SUPPRESSION FILTER
FGND
AGND1
F6VA
'H'−CHANNEL POWER GENERATION
AV5H
VOLTAGE REGULATOR
EMI/NOISE SUPPRESSION FILTER
FGND
AGND1
V3_31H
VOLTAGE REGULATOR
EMI/NOISE SUPPRESSION FILTER
FGND
AGND1
F6VB
'V'−CHANNEL POWER GENERATION
AV5V
VOLTAGE REGULATOR
EMI/NOISE SUPPRESSION FILTER
FGND
AGND1
V3_31V
VOLTAGE REGULATOR
EMI/NOISE SUPPRESSION FILTER
FGND
AGND1

Figure 6.5. Voltages for Analog section.

ensures high noise immunity and thus, superior performance and high resolution even at high speeds.

6.3 Isolation Shields

In addition to the conductive noise (via conductive paths) discussed in the section above, the analog section is susceptible to coupled noise too. These could be due to adjacent traces or coupling due to reflections from surfaces. Separate isolation enclosures for the two A to D channels and an enclosure for the complete analog section shields it against such noise. Further, this system shall be enclosed in a metal casing, which will form a good chassis ground and isolate the hardware from such noise sources.

6.4 External Power Supply Lugs

As an alternative to the power distribution and filtering scheme discussed in this chapter, power-lugs for external power supplies have been included in the design. In the event of a problem with the implemented filtering scheme, the system can be powered from external sources utilizing these power-lugs. More information on these power-lugs can be obtained from page 11 of the schematics included in appendix A.

Thus, we observe that by implementing a well designed, elaborate and highly structured filtering and distribution mechanism, twenty-two different power voltages (inclusive of intermediate voltage values) and five different ground signals are incorporated successfully in this design. Pages 2, 3, 11 and 25 of the schematics in appendix A deal with the circuitry for this filtering and distribution mechanism.

CHAPTER 7

CLOCK DISTRIBUTION MECHANISM

In the following sections we discuss the clock distribution scheme that has been developed for different modules included in this design.

7.1 Microcontroller clock distribution mechanism

The internal clock circuitry of the microcontroller consists of an integration of two oscillators and two PLLs. This circuitry generates the ARM processor clock, clocks for the different peripherals and four programmable clocks that can be used as outputs on pins to feed external devices. The two oscillators are namely the Main Oscillator and the Slow Clock Oscillator. A 32.768 kHz crystal is connected to the input pins of the Slow Clock Oscillator. The Main Oscillator is designed for a 3 to 20 MHz fundamental crystal. In this design we have connected a 18.432 MHz crystal to its input pins. A block diagram view of the microcontroller internal clock circuitry has been shown in figure 7.1.

Depending on how the microcontroller has been programmed, it works either in the Normal Mode or the Slow Clock mode. In the Normal mode the Main Oscillator and PLLs generate the peripheral clocks depending on application requirements and the ARM processor clock. In the Slow Clock mode the Main Oscillator and the PLLs are switched off to save power and the processor and peripherals run directly via the Slow Clock Oscillator. The Slow Clock mode is essential as at the time of power-up or reset the PLLs need to be configured to generate the processor and peripheral clocks.

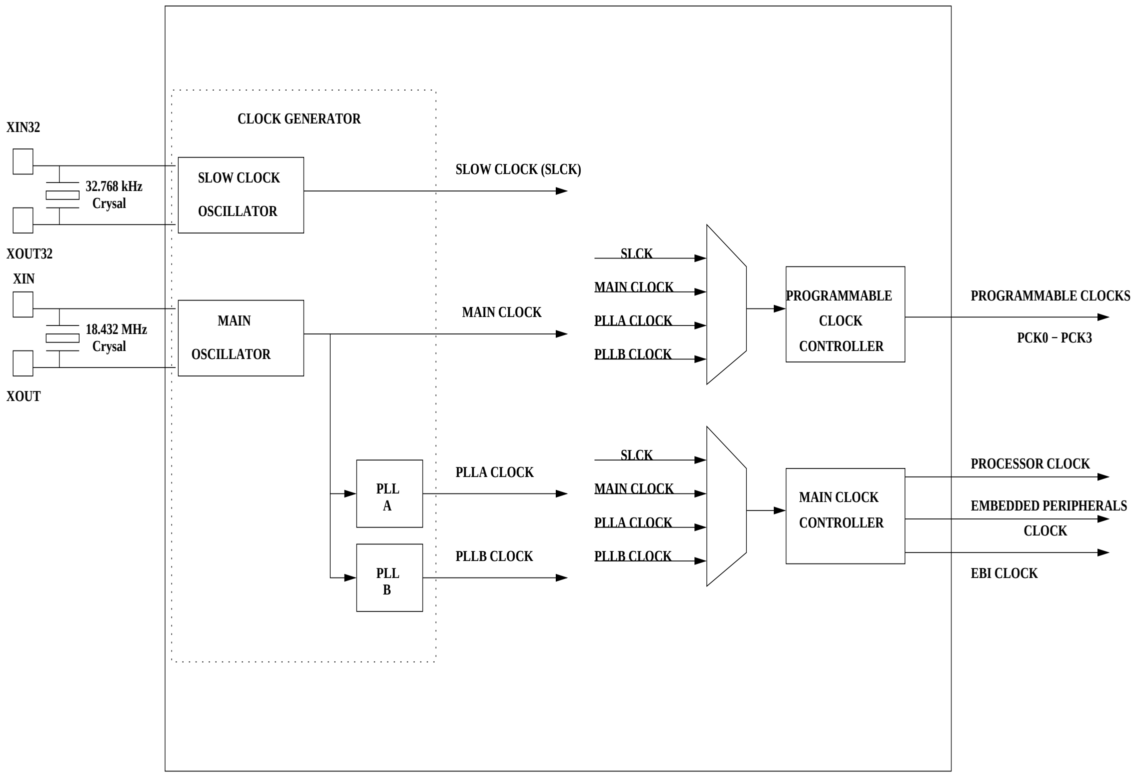


Figure 7.1. Microcontroller Clocking mechanism.

Once these PLLs have been configured, the system can then move into Normal mode of operation.

The processor clock can be configured to run at a maximum speed of 209 MHz. The peripheral clocks, which include the clocks to the External Bus Interface (EBI), can be configured to run at a maximum speed of 80 MHz. This EBI clock is utilized by the memory modules that interface to the microcontroller.

7.2 Clock distribution mechanism for FPGAs and their peripherals

Motorola MPC9447 [21], a 1:9 clock fanout buffer, is utilized to distribute a clock source to the PLLs of the two FPGAs. The outputs of these PLLs are utilized by the FPGAs for data processing and for their interfaces with external memory modules. Input to the clock buffer can be selected from either a 100 MHz oscillator or a programmable clock output from the microcontroller. At the time of system power-up, jumper JP20 on the schematics depicted in Appendix A is configured to select any one of these two inputs, as shown in figure 7.2. Under normal operation, the 100 MHz oscillator input is selected for distribution. Distribution of clock outputs CLK1 to CLK9 from the buffer has been discussed below. Figures 7.3 and 7.4 show this clock distribution scheme adopted for FPGA1 and FPGA2 respectively.

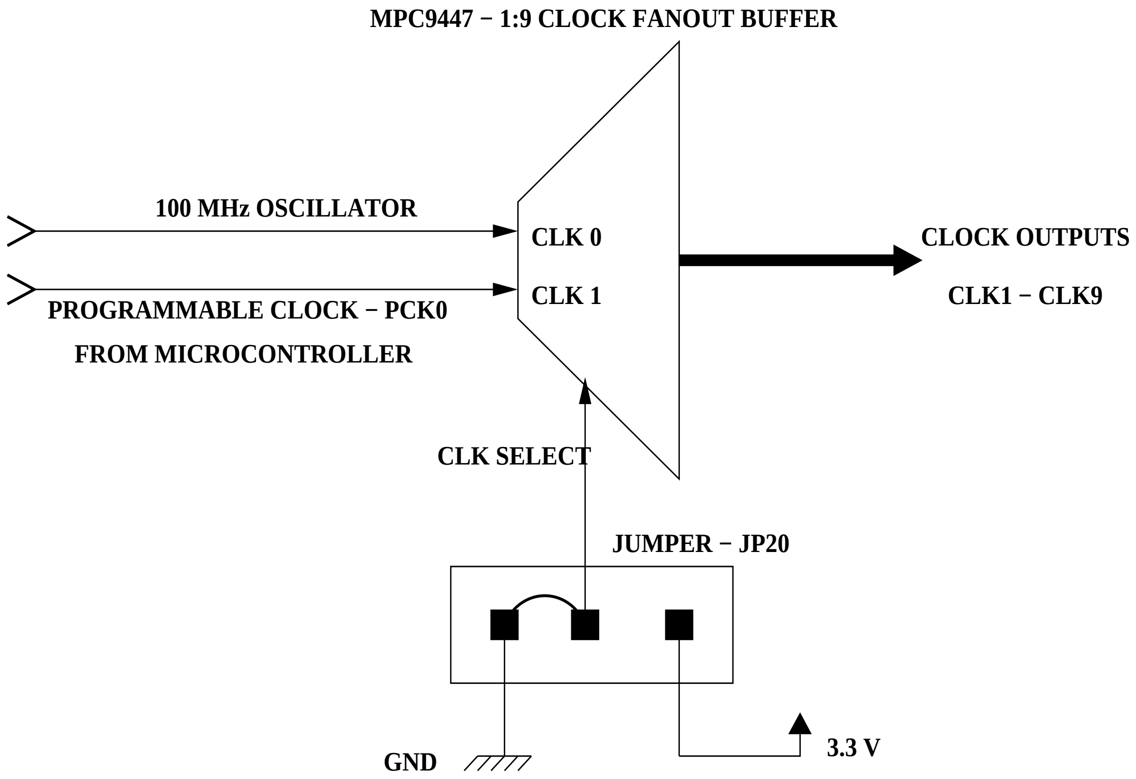


Figure 7.2. Clock distribution mechanism - Clock Fanout Buffer.

1. CLK1 $\rightarrow$ to PLL-12 on FPGA2. PLL-12 internal outputs are utilized for DPRAM communication and data processing, while external output is used to clock the DPRAM side interfacing with FPGA2.

2. CLK2 $\rightarrow$ to PLL-12 on FPGA1. The external clock output from this PLL is utilized to develop an alternate path for generating an external sampling clock input for this architecture as explained in section 7.3.

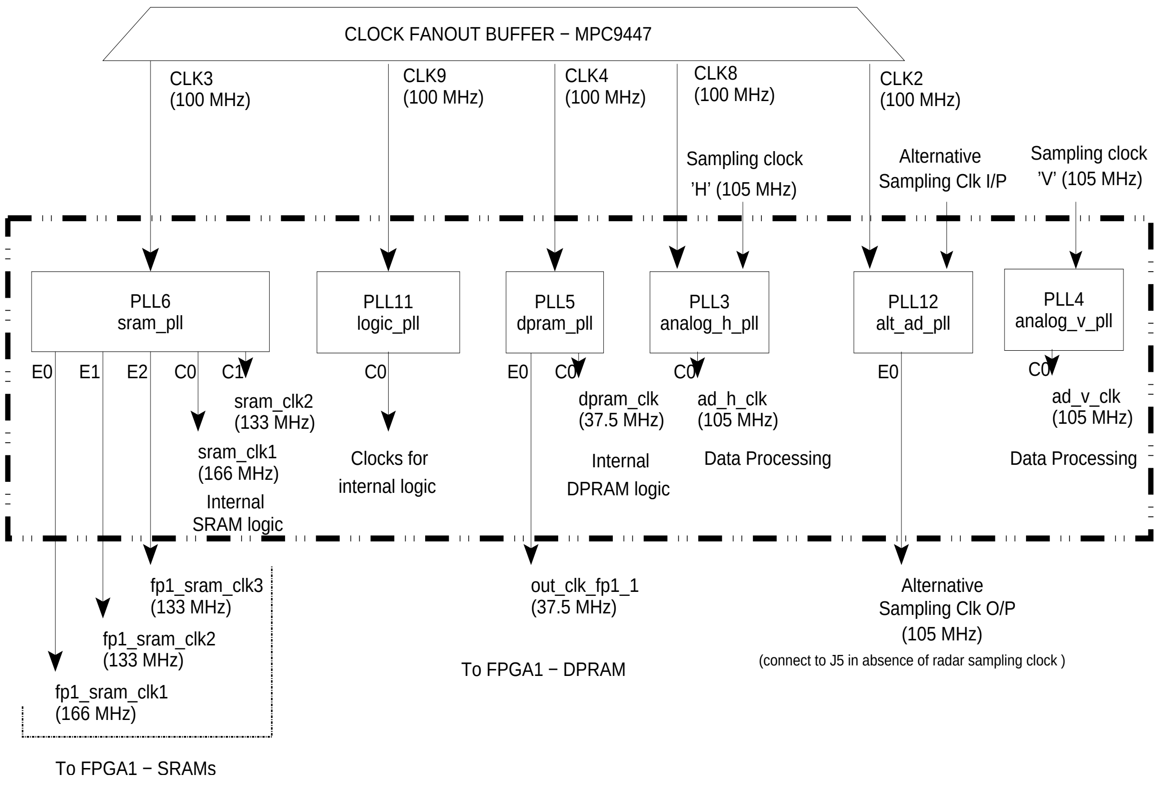


Figure 7.3. Clock distribution mechanism - FPGA1 PLL clock distribution (Adapted from Luko Krnan).

3. CLK3 $\rightarrow$ to PLL-6 on FPGA1. The outputs of this PLL are internally utilized by FPGA1 for its interface with the SRAM modules, and the external outputs provide the clocks to the these three SRAM modules coupled to FPGA1.

4. CLK4 $\rightarrow$ to PLL-5 on FPGA1. The internal outputs of this PLL clock FPGA1 transactions with the external DPRAM, while its external output is utilized by the DPRAM side interfacing with FPGA1.

5. CLK5 $\rightarrow$ to PLL-5 on FPGA2. The Gigabit Ethernet core clock inputs are generated by this PLL and has been discussed in detail in section 7.4.

6. CLK6 $\rightarrow$ to PLL-6 on FPGA2. The outputs from this PLL serve as clock inputs to the external SRAM chip coupled with FPGA2 and its internal logic pertaining to this chip.

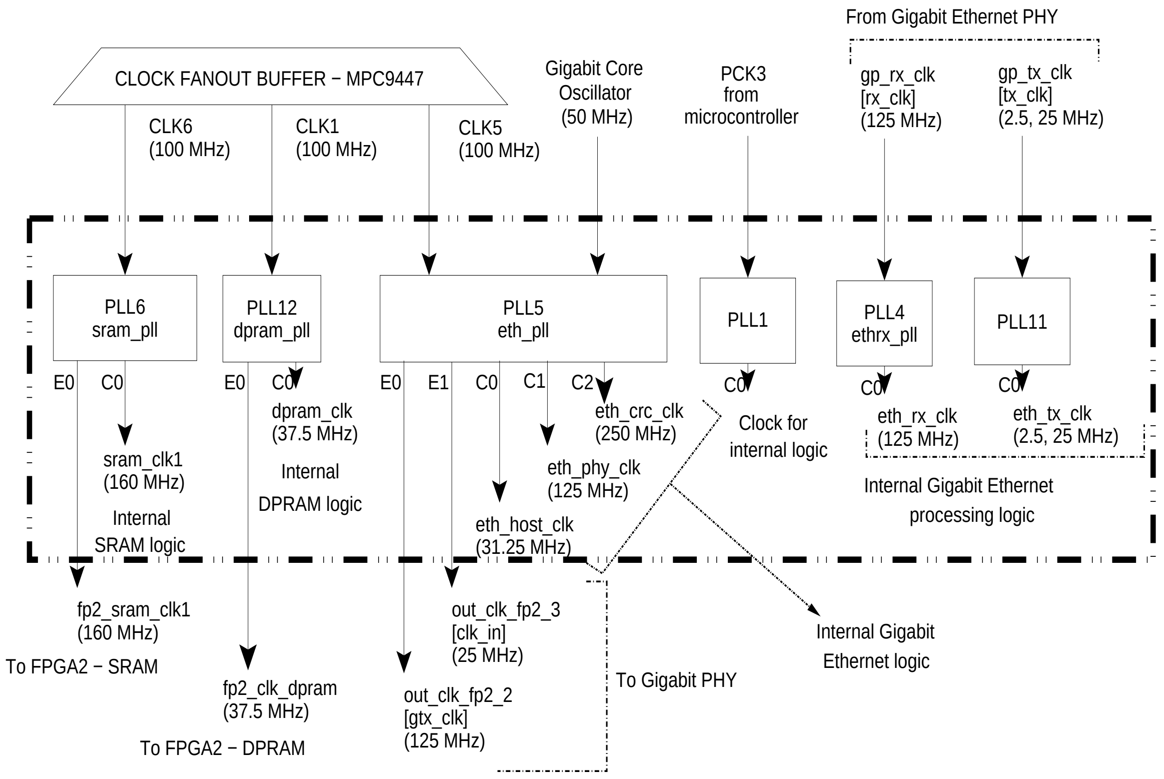


Figure 7.4. Clock distribution mechanism - FPGA2 PLL clock distribution (Adapted from Luko Krnan).

7. CLK7 → is routed to the Programmable Logic Device (PLD) for the FPGA configuration operations performed by it.

8. CLK8 → to PLL-3 on FPGA1. CLK8 is an alternative to the sampling clock input to FPGA1. By programming PLL-3, we can choose between an external sampling clock from radar input channel 'H' or CLK8 input to FPGA1 for its data processing needs.

9. CLK9 → to PLL-11 on FPGA1. Clocks for internal logic in FPGA1 can be generated from this PLL.

Besides the clock inputs enumerated above, for the ease of flexibility and clock distribution, additional PLLs on the two FPGAs have been given other clock inputs such as programmable clock output PCK1 from the microcontroller to PLL-1 on

FPGA1, external sampling clock from radar input channel 'V' to PLL-4 on FPGA1 and programmable clock output PCK3 from the microcontroller to PLL-1 on FPGA2.

7.3 Sampling Clock distribution mechanism

The mechanism devised for the distribution of the sampling clock is shown in figure 7.5. External connector J5 shown in the schematics of Appendix A, can be used by the radar system to feed a sampling clock to this system. This clock can then be utilized by the two A to D channels for data digitization and by FPGA1 for its internal logic and data processing, as shown in figure 7.3. This distribution route has all passive components and thus provides a low phase noise path.

As an alternative to this path, in the event that the radar system does not generate the sampling clock and if the system can tolerate phase noise, an additional connector J1 (refer Appendix A for schematics) is provided on this board. This connector can be used to feed a clock to PLL-12 on FPGA1, as shown in figure 7.3. This PLL can be programmed to select either this input or CLK2 from the clock distribution buffer and generate the required sampling clock frequency on its output. This output in turn is available as an external output on connector J3 (refer Appendix A for schematics). Connecting the output from J3 to the input of J5 can re-create the sampling clock distibution mechanism in the event of unavailability of a sampling clock from the radar unit.

7.4 Gigabit Ethernet core clock distribution mechanism

A 50 MHz clock input is required for the operation of the Gigabit Ethernet core embedded in FPGA2. PLL-5 on FPGA2 selects one of its two inputs, a 50 MHz oscillator or CLK5 from the clock distribution buffer to generate this clock. Further, different other clocks required for the Gigabit Ethernet core (embedded in FPGA2) interface to the Gigabit Ethernet physical layer device (PHY), are generated utilizing

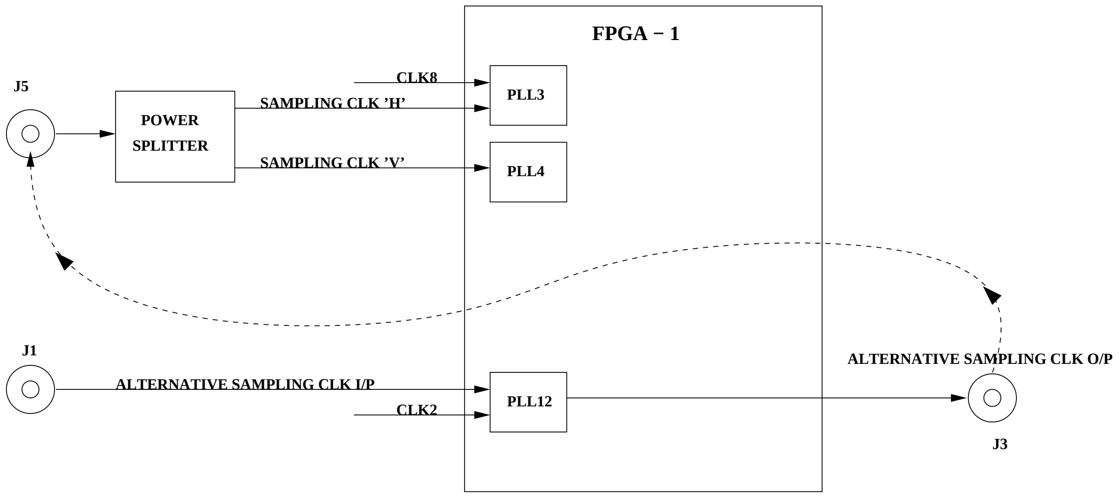


Figure 7.5. Clock distribution mechanism - Sampling Clock clock distribution mechanism.

the outputs from this PLL and the clocks generated by the Gigabit Ethernet PHY [24]. This clock distribution is a part of figure 7.4.

More information on the frequency values of these different clocks and their distribution can be gained through the respective manufacturer datasheets [3][9][20][6][24][21].

CHAPTER 8
SYSTEM BOOTUP STRATEGY

This chapter explains the procedure followed by the microcontroller while loading its boot program. This discussion shall be followed by a description of the different configuration schemes for the two FPGAs.

8.1 Microcontroller Bootup

The boot sequence for the microcontroller begins by executing a boot program. On power-up, depending on the state of the Boot Mode Select (BMS) input pin at the time of reset, the boot sequence shall commence by executing the contents stored either on the on-chip ROM (BMS = 1) or an external 8-bit parallel Flash chip (BMS = 0) connected to the External Bus Interface (EBI) on chip select signal zero (NCS0).

In the boot sequence, on selecting the external Flash chip, the bootloader is activated first. The first operation the bootloader does is the search for a valid program, a special sequence of eight valid ARM exception vectors. These vectors are used to store important information such as the size of the image to download and the type of Flash device. If a valid sequence is found, code is downloaded into the internal SRAM. This application may be the application code or a second-level bootloader. This is followed by a remap and a jump to the first address of the SRAM.

On selecting the on-chip ROM, the boot uploader is started. It initializes the Debug Unit serial port (DBGU). It then waits for any transaction and downloads a piece of code into the internal SRAM via XMODEM protocol for the DBGU. After the end of the download, it branches to the application entry point at the first address of the SRAM.

8.2 FPGA configuration

This architecture has three different schemes that can be followed for loading the configuration files into the respective FPGAs. The board has two non-volatile 'Software' Flash chips (32M x 8 bits each) to hold the FPGA configuration files and an Altera EPM7125AE Programmable Logic Device (PLD). The PLD and the two FPGAs are connected in a chain that terminates at a JTAG port. The Flash chips and the PLD have an interface to the microcontroller. The three configuration approaches devised for this design are discussed below.

8.2.1 Configuration via JTAG port

The first approach involves configuring the FPGAs directly thorough the JTAG interface. The two FPGAs and the PLD are connected in a JTAG chain that terminates in the JTAG port. A ByteBlasterMV cable connected between this JTAG port and a host computer running the Altera Quartus software can be used to download configuration files directly into the respective FPGAs. This scheme is employed during the testing and debugging phase of the system. The FPGA configuration attained through this approach is lost after power down. Further, during field operation, it is unfeasible for the user to access the JTAG port and configure the FPGAs, every time the system is powered-up or when the FPGAs need to be reconfigured. Thus, alternative schemes are required for automatic configuration of the FPGAs on system power-up.

8.2.2 Dynamic Reconfiguration

In the second approach, the configuration files are downloaded into the Flash chip via the 10/100 Mbps Ethernet interface to the microcontroller. At power-up, the microcontroller can accesses these configuration files from the Flash chip and instruct the PLD to load them onto the respective FPGAs. The Flash chip being non-volatile holds the configuration files even when the power is down. It needs to be re-programmed only when the configuration changes. Further, the Flash can store different configuration files performing different functions. As a result a remote user, via the microcontroller 10/100 Mbps Ethernet interface, can select from the available configuration files and instruct the PLD to load it into the concerned FPGA, providing us with the option of *dynamic reconfiguration* of the system.

8.2.3 Remote-User based, Dynamic Reconfiguration

Further, if the file-system of the operating system is run on a remote host PC and accessed by the microcontroller via its 10/100 Mbps Ethernet interface, the configuration files for the FPGAs too can be stored or generated as and when required on this remote host PC and downloaded into the FPGAs (via the PLD) to reconfigure them. Thus, this configuration scheme has the added benefit of *remote-user based dynamic reconfiguration* of the system.

CHAPTER 9
SIMULATION SETUP

The board schematics were simulated completely to check for functional and timing errors before sending the board out for fabrication.

9.1 Simulation Flow

The simulations target all the digital components of the board. Figure 9.1 depicts the simulation setup and flow.

The schematics for the board are done using Cadence Concept-HDL [11]. Concept-HDL creates a Verilog-HDL [25] netlist of the whole design from the schematics. This netlist includes all the components that are put on the board. These components are represented as black boxes in the netlist. Models based on the timing and functional behavior of these components can be plugged into respective black boxes to simulate them. The interconnection between these components is the same as the interconnection drawn in the schematics. The RTL design for different parts of the FPGAs was entered in Verilog-HDL. The whole design (FPGA and board design) was divided into modules and test-benches were devised for each of these modules. Each module of the board was simulated to verify its functional and timing related behavior. Once the modular testing was completed successfully, the whole board was simulated using a Verilog test-bench.

The next step in the simulation process is to synthesize and place and route the RTL designs for the FPGAs using Leonardo Spectrum/Altera Quartus [19] [2]. These RTL designs are demo algorithms used to verify the functionality of the design.

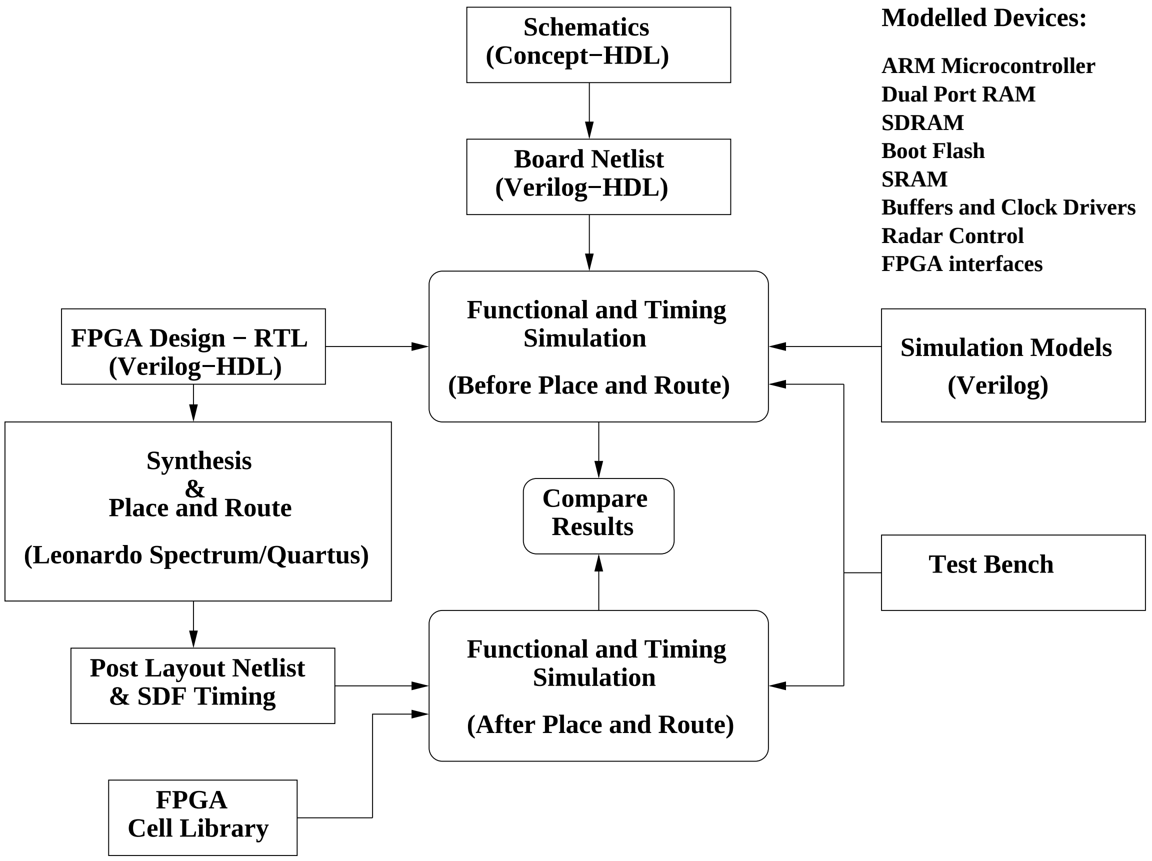

Schematics (Concept−HDL)
Board Netlist (Verilog−HDL)
Modelled Devices:
ARM Microcontroller
Dual Port RAM
SDRAM
Boot Flash
SRAM
Buffers and Clock Drivers
Radar Control
FPGA interfaces
FPGA Design − RTL (Verilog−HDL)
Functional and Timing Simulation (Before Place and Route)
Simulation Models (Verilog)
Synthesis & Place and Route (Leonardo Spectrum/Quartus)
Compare Results
Test Bench
Post Layout Netlist & SDF Timing
Functional and Timing Simulation (After Place and Route)
FPGA Cell Library

Figure 9.1. Simulation Setup.

During this stage, various timing constraints are defined depending on the design timing requirements. The timing results generated by the Quartus Static Timing Analyzer tool are compared to these constraints to verify that the design specifications are met.

Once the synthesis and place and route steps are performed successfully, the RTL design is then expressed as a Verilog-HDL netlist with the delay information in Standard Delay Format (SDF). The board simulations are re-run using the same test-bench and Verilog models for the components, but with the place and routed netlist for the FPGAs instead of the RTL design. The results are compared to the ones generated using the RTL design, for functional and timing behavioral verifications. In the advent of any discrepancies, the synthesis and place and route steps are repeated. If the specifications are still not satisfied, the RTL design is changed and the whole process repeated.

9.2 Simulation Modules

The components and the modular approach that was followed to test and simulate the design are enumerated and briefly explained below:

1. FPGA1 interface to the radar.

   Models: Flip-Flops, Voltage-level shifters and the part of FPGA1 talking to the radar.

   In this module, FPGA1 receives data and data-processing control signals from the radar. Some basic data processing in the form of coherent-integration is modeled in FPGA1. Testing in terms of functionality and delay statistics is performed.

2. FPGA1 interface to DPRAM.

   Models: DPRAM and the part of FPGA1 that interfaces with the DPRAM.

   In this module data transfer between FPGA1 and the external DPRAM shall be simulated.

3. FPGA2 interface to DPRAM.

   Models: DPRAM and the part of FPGA2 that interfaces with the DPRAM.

   Here data transfers from DPRAM to FPGA2 is simulated and tested.

4. Interface between FPGA1 and FPGA2.

   Models: Parts of FPGA1 and FPGA2 that talk to each other.

   Data lines running between FPGA1 and FPGA2 are modeled and simulated under this test.

5. FPGA1 interface with its SRAMs.

   Models: SRAM chips and the part of FPGA1 that interfaces with its SRAM chips.

   SRAM chips are referred to by FPGA1 for data-processing related parameters and values. This interface is tested through this module.

6. FPGA2 interface with its SRAM.

   Models: SRAM chip and the part of FPGA2 that interfaces with its SRAM chip.

   FPGA2 accesses its SRAM chip for intermediate data storage. These data transactions are simulated for functionality and timing verification through this module.

7. FPGA1 interface to the Microcontroller.

   Models: Part of FPGA1 that interfaces with the microcontroller and a dummy verilog model for the corresponding part of the microcontroller chip.

   The microcontroller controls and supervises the working of FPGA1 (and FPGA2), which is simulated and verified here.

8. FPGA2 interface to the Microcontroller.

   Models: Part of FPGA2 that interfaces with the microcontroller and a dummy verilog model for the corresponding part of the microcontroller chip.

   The microcontroller controls and supervises the working of FPGA2 (and FPGA1), which is simulated and verified here.

9. Microcontroller interface to its boot-flash chips.

   Models: Boot-Flash chips and a dummy verilog model for this microcontroller interface.

   Boot-Flash chip models simulated with a dummy microcontroller verilog model is used to test the functionality of this interface.

Tool	Vendor	Design Process
Concept-HDL ver 15.1	Cadence	Schematic Capture
Verilog-XL ver 3.11	Cadence	Verilog-HDL Compiler
Leonardo Spectrum ver 2002a	Mentor Graphics	FPGA Synthesis, Technology Mapping
Quartus ver 4.0	Altera	FPGA Place & Route, Timing Analysis
ADS ver 1.2	ARM	ARM Compiler/Debugger

Table 9.1. Design Tools

10. Micro-controller interface to its SDRAM chips.

    Models: SDRAM chips and a dummy verilog model for this microcontroller interface.

    Dummy microcontroller verilog model coupled with SDRAM models designed for this interface is used to test its functionality.

Modular testing helps accomplish a thorough functional verification of the system.

9.2.1 Tools Used

Various Electronic Design Automation (EDA) tools have been used during the design and testing of this architecture. Table 9.1 lists these tools.

CHAPTER 10

CONFIGURATION STRAPS AND JUMPER SETTINGS

Before powering-up the system, some configuration steps are required to be completed. Following is a list of configuration straps and jumper settings to be performed on the data acquisition board and their default values (if any). Refer to the schematics in appendix A for more information.

1. JP21 and JP22 on page-1 of the schematics are left un-soldered for the first two prototype boards. Thus, we need to solder and connect the input (DRY_H/V or U27/32) that we intend to connect to the net - BUFLAT_H/V. Once JP21 and JP22 are soldered in, the jumper needs to be set to select one of these two inputs. For the prototype boards, the U27/U32 inputs were soldered to the BUFLAT_H/V net.

2. JP19 on page-3 of the schematics can be used to turn off the voltage regulator, supplying power to the analog section of the system. On shorting pins 1 and 2 on this strap, the INHIBIT input of voltage regulator-VR1 is left floating, required for its normal operation. Shorting pins 2 and 3 pulls this input low, causing the voltage regulator to shutdown.

3. JP20 on page-4 of the schematics selects either a 100 MHz oscillator output or the PCK0 - programmable clock output from the microcontroller as an input to the clock buffer U9.

   Default: Short pins 2 and 3 to pull pin3 (Clock Select) on U9 low and select 100 MHz oscillator output as the input to the clock buffer.

4. JP11 on page-7 of schematics determines if the Gigabit PHY advertises Master (multiple nodes) or Slave (single node) priority during 1000BASE-T Auto-Negotiation.

   '1' selects multiple node priority (switch or hub).

   '0' selects single node priority (NIC).

   Default: '0'.

5. JP15 on page-7 of the schematics controls the automatic pair swap (Auto-MDIX) of the MDI/MDIX interface.

   '1' enables pair swap mode.

   '0' disables the Auto-MDIX and defaults the Gigabit PHY into the mode preset by the MAN_MDIX_STRAP pin (JP9 and JP8 combination).

   Default: '1'.

6. JP12 and JP10 combination on page-7 of the schematics initializes Auto-Negotiation Enable bit for the Gigabit PHY.

   '1' enables Auto-Negotiation.

   '0' disables Auto-Negotiation.

   Default: '1'. Hence, short pins 1 and 2 on both JP12 and JP10.

7. JP9 and JP8 combination on page-7 of the schematics sets the status of Speed[0] strap. JP7 and JP6 combination, again on page-7 of the schematics sets the status of Speed[1] strap. Together these two act as the speed-select strap and have two different functions depending on whether Auto-Negotiation is enabled or not (JP12 and JP10 combination). Different combinations and the resultant speed selected are shown in tables 10.1 and 10.2.

   Since, Auto-Negotiation has been enabled:

   Default: '10'. Hence, short pins 1 and 2 for JP7 and JP6 and pins 2 and 3 for JP9 and JP8.

Speed[1]	Speed[0]	Speed Enabled
1	1	Reserved
1	0	1000BASE-T
0	1	100BASE-TX
0	0	10BASE-T

Table 10.1. Speed-strap settings for Auto-Negotiation disabled

Speed[1]	Speed[0]	Speed Enabled
1	1	1000BASE-T, 10BASE-T
1	0	1000BASE-T
0	1	1000BASE-T, 100BASE-TX
0	0	1000BASE-T, 100BASE-TX, 10BASE-T

Table 10.2. Speed-strap settings for Auto-Negotiation enabled

8. JP5 and JP4 combination on page-7 of the schematics sets the value for the duplex mode for the Gigabit PHY.

   '1' enables Full Duplex mode.

   '0' enables Half Duplex mode.

   Default: '1'. Hence, short pins 1 and 2 on both jumpers.

9. JP3 and JP2 combination on page-7 of the schematics sets the value for manual MDI/MDIX configuration.

   '1' Gigabit PHY is manually set to cross-over mode (MDIX).

   '0' Gigabit PHY is manually set to straight mode (MDI).

   Default: '0'. Hence, short pins 2 and 3 on both jumpers.

10. JP1 on page-7 of the schematics acts as the Gigabit PHY configuration strap to enable Non IEEE Compliant Mode. This mode allows interoperability with certain non-IEEE compliant 1000BASE-T transceivers.

    '1' enables IEEE compliant operation and non-compliant operation.

'0' enables IEEE compliant operation but inhibits non-compliant operation. *Default:* '1'. Hence, short pins 1 and 2 on both jumpers.

11. JP13, JP14, JP17 and JP18 on page-8 of the schematics determine the base address of the Gigabit PHY. The LSB of the base address as been set to '1' (pin 13 of the Gigabit PHY has been pulled high in the design). JP13 forms the MSB of this address. Gigabit PHY base address:
    MSB ———————— LSB
    JP13 JP14 JP17 JP18 1
    During the testing phase, these straps were randomly set to values:
    JP13 - 1
    JP14 - 0
    JP17 - 1
    JP18 - 0

12. JP16 on page-11 of the schematics can be used to power down the system. Shorting pins 2 and 3 on this jumper activates the power-supply by pulling the PS-ON input on the ATX power connector to ground, powering-up the data acquisition system (normal operation). Connecting pins 1 and 2 on this jumper shorts the 5V standby output and the PS-ON input, causing the system to power-down.

13. JP24 on page-18 of the schematics acts as the Boot Mode Select (BMS) strap for the microcontroller. Depending on whether the microcontroller boots from the internal SRAM or external FLASH chip, leave the jumper open (BMS = 1) or closed (BMS = 0) respectively.

14. JP23 on page-19 of the schematics configures the JTAGSEL input to the microcontroller. If pin 2 and 3 are shorted on this strap, the JTAGSEL signal is pulled low enabling ICE mode of functioning of the microcontroller for debug-

ging purposes. On shorting pins 1 and 2, the JTAGSEL signal is pulled high, enabling JTAG mode for the microcontroller.

15. JP27 on page-19 of the schematics on being closed connects the microcontroller reset signal 'NRST' to its ICE/JTAG socket.

16. JP26 on page-19 of the schematics is not an actual jumper, but a header with some pins of the Reset Controller-U66 being made available externally for future usage. (The reference designator should be changed for future boards to starting with a 'P' as in the case of normal signal headers.)

17. JP25 on page-20 of the schematics is again not an actual jumper, but a header for accessing the pins of the USB device interface of the microcontroller for future usage. (The reference designator should be changed for future boards to starting with a 'P' as in the case of normal signal headers.)

Follwing the steps listed above completes the preliminary configuration and jumper setting setup for the hardware system.

CHAPTER 11

TESTING AND DEBUGGING THE HARDWARE SYSTEM

Once the design process and the board physical layout and manufacturing is accomplished, the next step is to test and debug the hardware to confirm that all the modules and the interfaces in the design are working accurately. In this chapter we discuss the debugging features of the system and debug tools that can be utilized, followed by the steps pursued to test and debug the system.

11.1 Debugging features

Features that have been incorporated to facilitate debugging and testing the system are discussed in this section.

11.1.1 External Power supply mechanism

This hardware system has power-lugs to feed the different required voltages from external voltage sources. In the event of failure of the on-board power filtering and distribution scheme or the scheme not performing as per the requirements, these power-lugs can be utilized to power-up the system.

11.1.2 Debugging features of the design

The design has a 34 channel Samtec connector that is used to connect the board to a Logic Analyzer [1]. Critical signals from the two FPGAs and the microcontroller have been routed to this connector, and can thus be monitored to debug problems in the system. The board also has test points for ground, clock and other control signals.

As explained in section 5.1.4, the two-pin UART comprised in the debug unit of the microcontroller, acts as an effecient channel to debug the microcontroller and its interfaces. Further, the debug unit JTAG interface can be used to monitor the microcontroller via a JTAG-In Circuit Emulator (ICE), as explained in the section below.

11.2 Debugging tools

As mentioned above, the Logic Analyzer interface is utilized to monitor signals from the FPGAs and the microcontroller. To debug the microcontroller, the development package distributed by Atmel [10] can be used to build testing algorithms. The ARM tools required for these test steps include the ARM Developer Suite (ADS) [8], a software tool-set for ARM application development, which includes a C/C++ compiler, code generator and a simulator. The other tool is the JTAG-In Circuit Emulator (ICE), a hardware module that acts as an interface between the PC that runs ADS and the JTAG port of the microcontroller. This setup can be used to download a test code from the host computer into the microcontroller's internal memory [8].

Utilizing these debug features and tools, we follow the steps listed below to test and debug the system and ascertain that it meets our requirements.

11.3 Preliminary verification

The testing/debugging of the system involves basic verifications as listed below:

1. Perform physical verification of the system such as: confirm that during the manufacturing process, components that were not to be populated, have actually been left unpopulated on the boards and check the polarities of all electrolytic (all polarity sensitive) capacitors.

2. Follow the instructions in chapter 10 to set all configuration straps.
3. Measure the impedance between different voltage power lugs and ground to confirm that they are isolated.
4. Considering that crystal Y3 was incorrectly placed on the prototype boards, check the placement of crystals Y3 and Y5. These crystals clock the Microcontroller.
5. Next we test the voltage levels at the power-stubs in the digital section of the design: Disable the power supply to the analog section of the design by shorting pins 2 and 3 on jumper JP19 that inhibits the output of voltage regulator VR1. Power-up the digital section of the system via the ATX power supply. Keep a watch for any visual abnormalities and overheated components. Measure the voltage at the power stubs (part of the digital power circuitry) to confirm proper voltage values and then power-down the system.
6. Next we test the voltage levels at the power-stubs in the analog section of the design: Leave the jumper JP19 open so that the INHIBIT input pin to VR1 is left floating, as required for its normal operation. Power-up the system and keep a watch for any visual abnormalities and overheated components. Measure the voltage at the power stubs (part of analog power circuitry) to confirm proper voltage values.

The steps mentioned above complete the initial verification of the system. We now test the microcontroller and its interfaces.

11.4 Testing the Microcontroller and its interfaces

For testing the microcontroller, we upload the boot program - *u-boot*, an open source code available at http://sourceforge.net/projects/u-boot [28], into the external

SDRAMs and then into the external FLASH chips. We then test the 10/100 Mbps Ethernet interface by downloading the Linux kernel from an external host PC onto the microcontroller via this Ethernet interface.

11.4.1 Loading the Boot code

The first step in testing the microcontroller is to load the boot code. The files, 'loader.bin' provided with the Atmel package [10] and 'u-boot.bin' an open source code [28] are transferred to the microcontroller. The setup is shown in figure 11.1 and the steps that we need to follow are enumerated below:

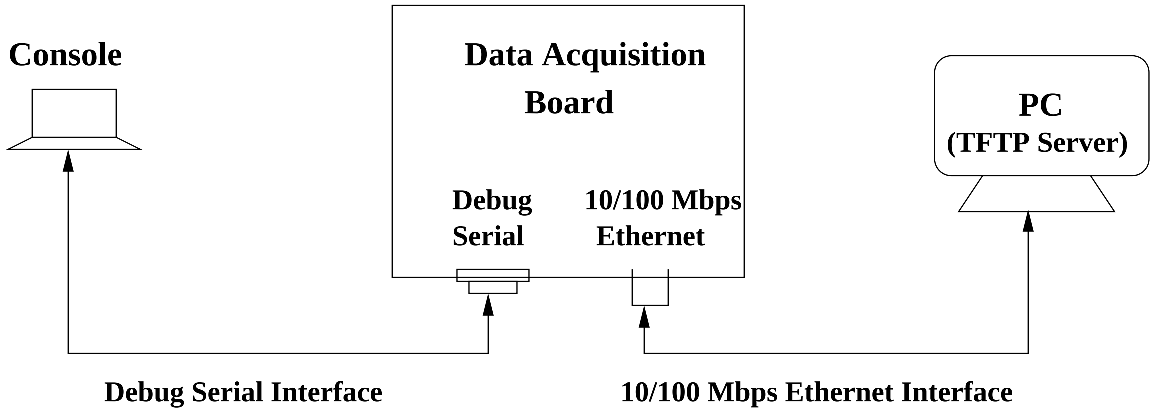


Figure 11.1. Test setup - Boot code and 10/100 Mbps Ethernet.

1. Leave the jumper JP24 open as we need to begin with booting off the internal SRAM of the microcontroller. Leaving JP24 pulls the Boot Mode Select (BMS) pin high at reset and selects the on-chip Boot mode for the microcontroller (Refer section 8.1 for more information).

2. Connect a cross-over (null modem) cable between the debug serial port P14 and a console and power-up the system.

3. Launch a console application with the parameters:

   - Bits per second: 115200

- Data bits: 8
- Parity: None
- Stop bits: 1

4. From the console, send the file "loader.bin" with the Xmodem protocol.

5. Once this transaction is complete, from the console, send the file "u-boot.bin" with the Xmodem protocol.

   After the end of the download, the Boot Program branches to the application entry point at the first address of the SDRAM, and a prompt (Uboot>) is displayed on the console. Next we copy the boot image to the external on-board FLASH.

6. At the prompt (Uboot>) on the console, enter the following commands:

   ```
   Uboot> protect off all
   Un-protect Flash Bank # 1
   Uboot> erase all
   Erasing sector 1  ok
   ...
   Erasing sector 39  ok
   Done
   ```

7. At this step the flash is completely erased. The next step is to load the boot image in SDRAM and to copy it in flash. This is done through the serial port (Kermit protocol on the console). The first file to load is the boot image "boot.bin" [10] (path: CDROM:/Linux-arm9/uboot/) and the set of commands are listed below.

```
Uboot> loadb 20000000
## Ready for binary (Kermit) download ...
## Start Addr =0x20000000
```

8. At this step the boot image is loaded in SDRAM. The next command copies the SDRAM (20000000) to FLASH(10000000).

```
Uboot>cp.b 20000000 10000000 5FFF
Copy to flash... done.
Uboot>protect on 10000000 10005FFF
Protected 3 sectors
Uboot>
```

9. Once Primary Bootstrap is loaded, u-boot gzipped image must be copied into Flash. The principle is exactly the same as for the primary bootstrap. The file to load is the uboot gzipped image "u-boot.gz" [10] (path: CDROM:/Linux-arm9/uboot/).

```
Uboot> loadb 20000000
## Ready for binary (Kermit) download ...
## Start Addr =0x20000000
Uboot>cp.b 20000000 10010000 FFFF
Copy to flash... done.
Uboot>protect on 10010000 1001FFFF
Protected 2 sectors
Uboot>
```

u-boot image has been successfully flashed. Set the jumper JP24 to pull BMS low at the time of reset and thus boot from the external on-board FLASH.

11.4.2 Testing the 10/100 Mbps Ethernet interface and launching Linux on the microcontroller

Network downloading is one of the most powerful capabilities of the u-boot. It provides the system with the capability to download binary data from a TFTP server connected to the network. Thus, once u-boot, the boot program, has been loaded onto the external FLASH chips, we test the 10/100 Mbps interface of the microcontroller by downloading the Linux kernel onto the SDRAM from a PC running a TFTP server. As shown in figure 11.1, connect the 10/100 Mbps interface to a PC on the network running a tftpserver. File 'uImage' from CDROM:/Linux-arm9/ has to be transferred via this setup.

The steps that were followed are listed below:

1. Power-up the system. The microcontroller boots up from the external FLASH and a prompt (Uboot>) is displayed on the console.

2. Follow the instructions given in the Linux Installation document provided with the Atmel package [10] and u-boot documentation [28] to set certain parameters such as the MAC and IP address of the board and the IP address of the PC running the TFTP server and other system parameters. Type commands to run the kernel. These commands shall download the gzipped kernel files to the SDRAM and shall uncompress them.

3. A set of system information messages are displayed on the console and Linux is launched on the system.

   A script that includes all these commands can be generated and stored into the FLASH chips, so that these commands are automated and executed once we power-up the system.

These steps confirm that the 10/100 Mbps Ethernet interface is working appropriately and Linux has been launched successfully.

11.5 Testing the FPGAs and PLD connectivity

Next we confirm that the FPGAs and the PLD have been powered properly. Connect a ByteBlasterMV cable between a host PC that runs Altera Quartus and the board via the JTAG connector P11. Power-up the system and view the components seen in the JTAG chain to confirm that these three devices have powered up properly.

11.6 Testing the Microcontroller - FPGA interface

We now implement algorithms to test the microcontroller interface to FPGAs. Data transactions between the microcontroller and the FPGAs were implemented successfully.

11.7 Testing the Analog section and FPGA1

The test setup for testing the analog section and FPGA1 of the design in shown in figure 11.2. The list of steps to be followed is given below.

1. Download the demo sampling code written by Luko Krnan into FPGA1. This algorithm routes the output onto the Logic Analyzer interface. Hook-up the logic analyzer to this interface using the Samtec probes.

2. Apply a sampling clock to J5 and analog signals to J4 and J6, and sample some data onto the analyzer to confirm the proper working of FPGA1 and the analog section of the design.

The analog input was sampled and data stored in FPGA1. This data was transferred to a PC via the microcontroller 10/100 Mbps Ethernet interface and analysed. Results generated from this analysis have been discussed in section 12.4.

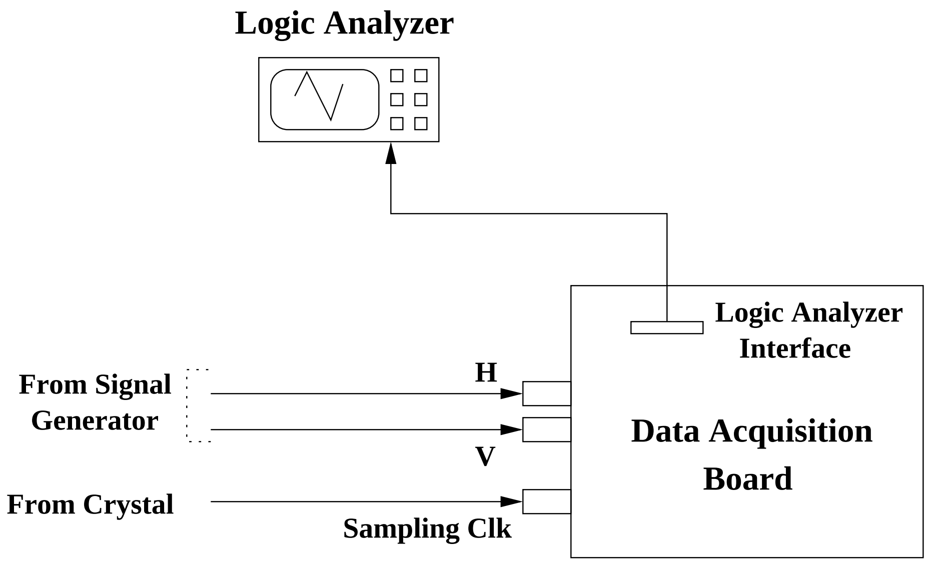


Figure 11.2. Test setup - FPGA1 and Analog section.

11.8 Testing the Gigabit Ethernet interface and FPGA2

We now test the functioning of the Gigabit Ethernet interface that involves a Gigabit Ethernet core embedded in FPGA2 and a Gigabit Ethernet PHY circuitry hooked up to a RJ45 connector. Figure 11.3 depicts the test setup for this task.

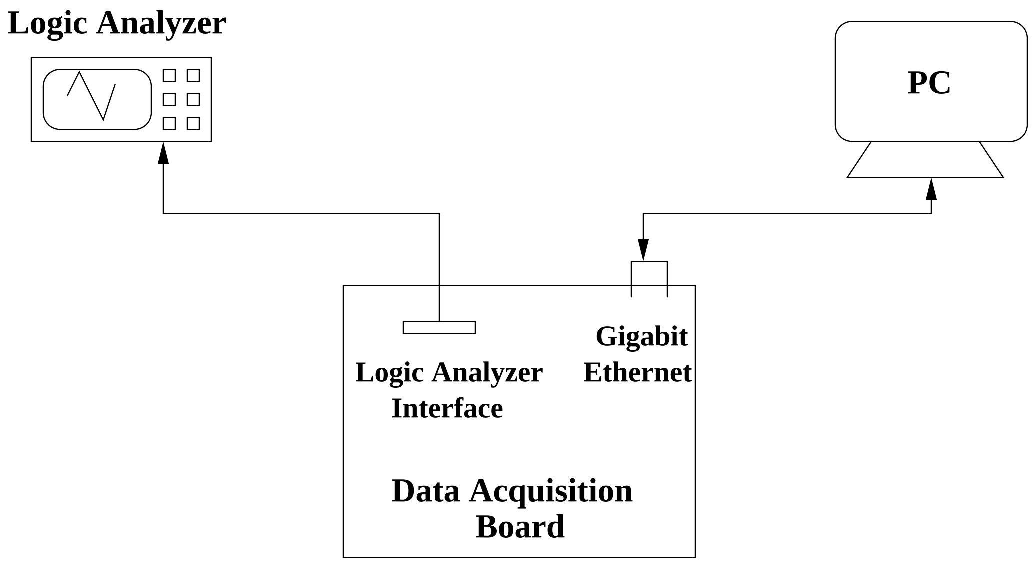


Figure 11.3. Test setup - FPGA2 and Gigabit Ethernet Interface.

1. Power-down the system and connect the ByteBlasterMV between JTAG interface P11 and a PC running Altera Quartus software. Connect an Ethernet cable

between the Gigabit Ethernet interface and the host PC (with Gigabit Ethernet functionality) and power-up the system.

2. Next download the Gigabit Ethernet interface demo test algorithm generated by Luko Krnan, into FPGA2. This algorithm imitates the Gigabit Ethernet core in the loop-back mode.

3. Run the test program on the PC (written by Luko Krnan). This setup successfully accepts data sent by a host PC hooked on to the Gigabit Ethernet interface and loops it back to the PC.

4. Monitor the output on the PC to confirm the proper working of this interface.

The successfully implementation of the steps mentioned above confirms that the different hardware modules, inter-modular interfaces and the communication interfaces incorporated in this design are faultless and operational. Further tests and detailed experimentation shall confirm the working of other interfaces such as the USB and Serial port for the microcontroller and the programming of the FPGAs via the PLD or remote-reconfiguration via the 10/100 Mbps Ethernet interface.

CHAPTER 12

RESULTS

This chapter lists the preliminary results that have been generated while testing the functionality of the data acquisition board and its interfaces.

12.1 Power distribution design

The voltage values at the power-lugs and power input pins to different components were in accordance with the requirements, ensuring the proper functioning of the power filtering and distribution scheme.

12.2 Results for microcontroller and its interfaces

Tests performed to confirm the functionality of the microcontroller and its interfaces include loading the boot code onto the internal SRAM and then the external FLASH chip via the debug serial port, launching Linux via the 10/100 Mbps Ethernet interface and testing the microcontroller - FPGA interface.

12.2.1 Loading microcontroller boot code via debug serial port

As described in section 11.4.1, microcontroller boot code was copied into the internal SRAM and then to the external FLASH chips, via the debug serial port interface. The board now boots automatically on power-up. This ensures that the microcontroller boots successfully and the debug serial port works as per our requirements.

12.2.2 10/100 Mbps interface

As discussed in section 11.4.2, the kernel and ramdisk for Linux was successfully downloaded into the microcontroller SDRAMs from a TFTP server connected to the network, ensuring that the 10/100 Mbps Ethernet interface is operational.

12.2.3 Porting Linux to the microcontroller

The kernel and ramdisk downloaded above were uncompressed and Linux was successfully launched on the microcontroller.

12.2.4 Microcontroller - BootFlash and SDRAM interface

Tests discussed in sections 12.2.1 and 12.2.2 required transactions over the microcontroller - BootFlash and microcontroller - SDRAM interfaces. Successful completion of these procedures authenticates the functionality of these interfaces and memory chips.

12.2.5 Microcontroller - FPGA interface

Dummy test algorithms created by Luko Krnan, to transfer data and control parameters between the microcontroller and the FPGAs, were successfully executed affirming that these interfaces are flawless.

12.3 PLD - FPGA1 - FPGA2 JTAG chain

As described in section 11.5, the PLD-FPGA1-FPGA2 JTAG chain was established successfully.

12.4 Results for the A to D channels and FPGA1

Figure 12.1 shows the Magnitude (in dbFS) Vs Frequency (in MHz) graph for a demo input to the analog channel. Input signal has a bandwidth of 10 MHz. This input was sampled utilizing a 100 MHz clock input to the system. The signal at 0 dBFS and the harmonics can be seen in the graph. The data generated had a Signal to Noise Ratio (SNR) of 75.052 dB, equivalent to the typical value quoted in the A to D converter datasheet [6]. Further, as evident from the graph, the worst harmonic is at about -85 dB.

These results confirm that the analog section of the design works properly. In addition, the high SNR, weak harmonics and low noise power asserts a high performance analog section and a successful power-filtering scheme.

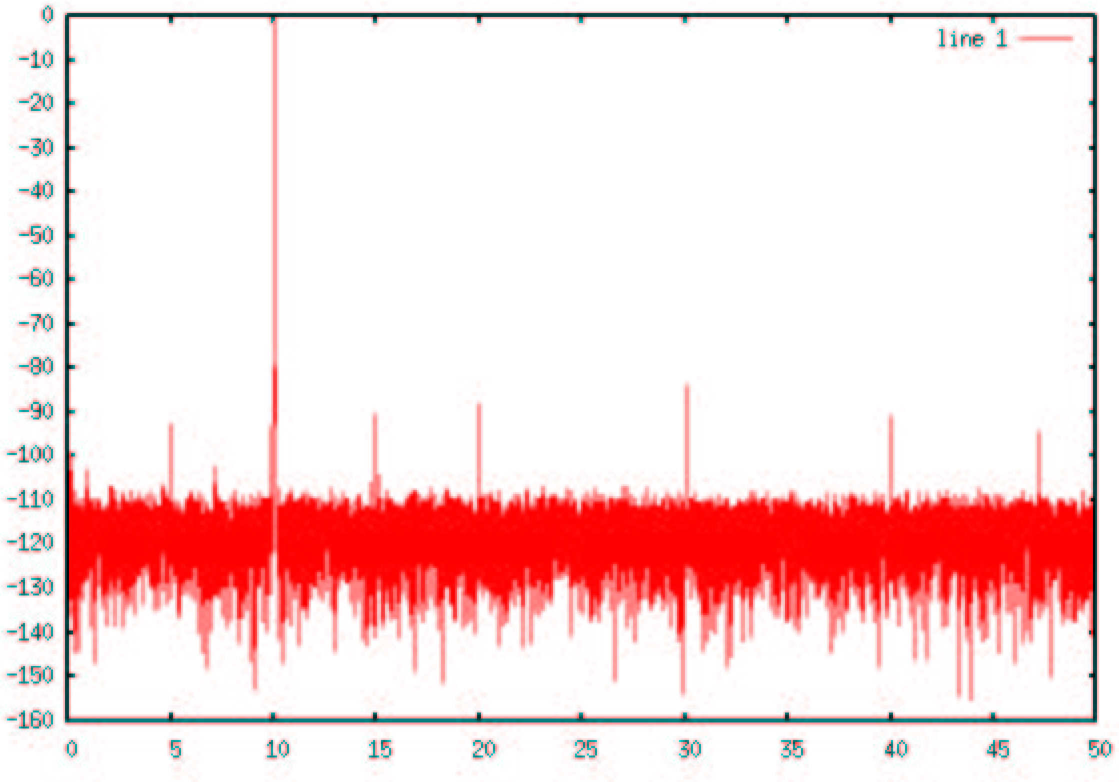


Figure 12.1. Analog section test result

12.5 Results for FPGA2 and Gigabit Ethernet interface

A demo test algorithm imitating the functionality of a Gigabit Ethernet core in the loop-back mode was generated by Luko Krnan and implemented in FPGA2. This setup successfully accepted data sent by a host PC hooked on to the Gigabit Ethernet interface and looped it back to the PC. This asserts the proper functionality of this interface and of FPGA2.

Thus, these results ascertain the affirmative status and functioning of all the modules and components included in the design and its communication interfaces.

CHAPTER 13
SUMMARY

13.1 Conclusion

This document describes a reconfigurable data acquisition system design for weather radar applications. The architecture for this design, to be integrated with the Distributed Collaborative Adaptive Sensing (DCAS) radars being developed by the Engineering Research Center (ERC) for Collaborative Adaptive Sensing of the Atmosphere (CASA) at the University of Massachusetts-Amherst, has been developed. Early and accurate detection of tornados, hazardous weather and floods are the applications this radar system will be applied for. These applications demand a high data throughput system, capable of processing bulk data quantities with complex signal processing algorithms at approximately 100 Mbits per second [13] in real-time. Further, reliable high-speed communication interfaces to transport finished data products to the end-user are expected.

A single circuit card implementation has been developed that meets these radar requirements. A reconfigurable architecture with high-density, high-performance FPGAs as the processing elements and an intelligent, feature-rich, high-speed microcontroller acting as the master control to the system has been designed. Two high-speed and high-resolution A to D channels that supersede the sampling requirements have been implemented in this architecture. Abundant on-chip and on-board memory resources and the presence of contemporary fast, reliable communication interfaces such as the Gigabit Ethernet, Fast (10/100 Mbps) Ethernet, USB, Serial ports and IDE channels, renders the proposed design suited for the intended application. Remote-host based control of this system is possible due to the presence of various network interfaces. Presence of an operating system alleviates the control and tool development process. Remote-user based reconfiguration of the design is a potent feature enabled in this design. Testing and debugging features in the form of dedicated Serial interface, JTAG ports and support for logic analyzer based monitoring of the system, further strengthen the assertion of a powerful, apt solution to the radar requirements.

Board design work for this design has been completed. The hardware system, its modules and interfaces have been tested successfully by implementing various test procedures. Linux has been successfully ported onto the microcontroller and thus, remote-user based control and reconfiguration features have been enabled. The design, layout, manufacturing and testing process has been completed well within the stipulated time frame, thus, providing the DCAS radar team with ample time for integrating the acquisition unit with the DCAS radars and field-testing the weather monitoring system.

13.2 Future Work

This architecture needs to be integrated with the DCAS radar units. Components that need to be developed and designed for its integration with the DCAS radars have been discussed in this section.

1. The Linux kernel needs to be configured to include device drivers for the USB host and USART interfaces to render these interfaces active.

2. Algorithm for the microcontroller to control the PLD and it configuring the FPGAs has to be developed.

3. In order to integrate the data acquisiton unit with the DCAS radars, signal processing algorithms need to be devised and implemented in the FPGAs. Control

and distribution algorithms for the microcontroller and the different interfaces need to be developed as per the radar requirements.

4. The USB device port pins for the microcontroller have been routed to an external header. A daughter-board that implements USB device port logic can be integrated with the data acquisition board for future applications.

APPENDIX A
SCHEMATICS

The Schematics for the Data Acquisition Board is given in Appendix A. It spans from Figure A.1 to Figure A.25.

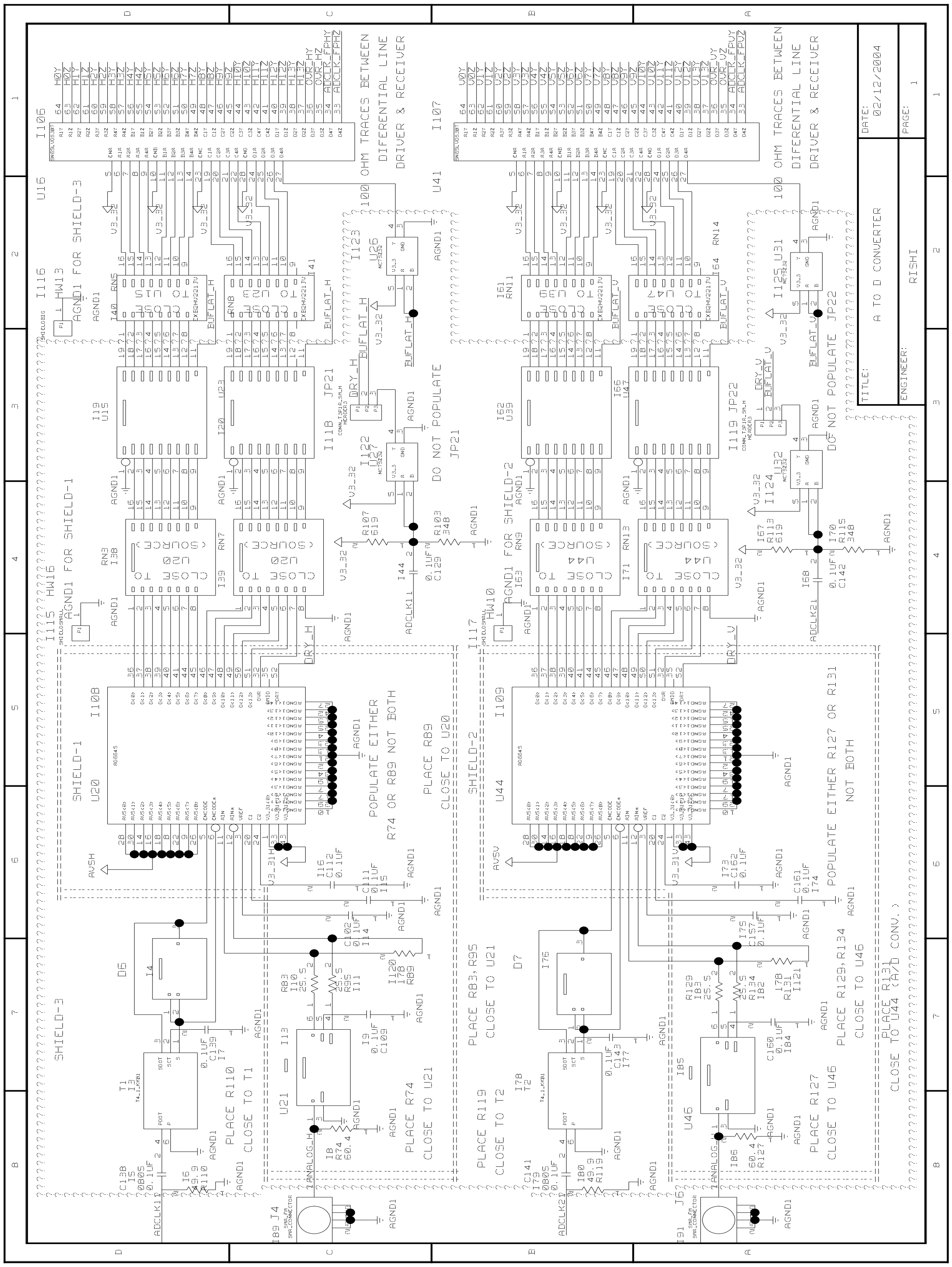


Figure A.1. AtoD Converter.

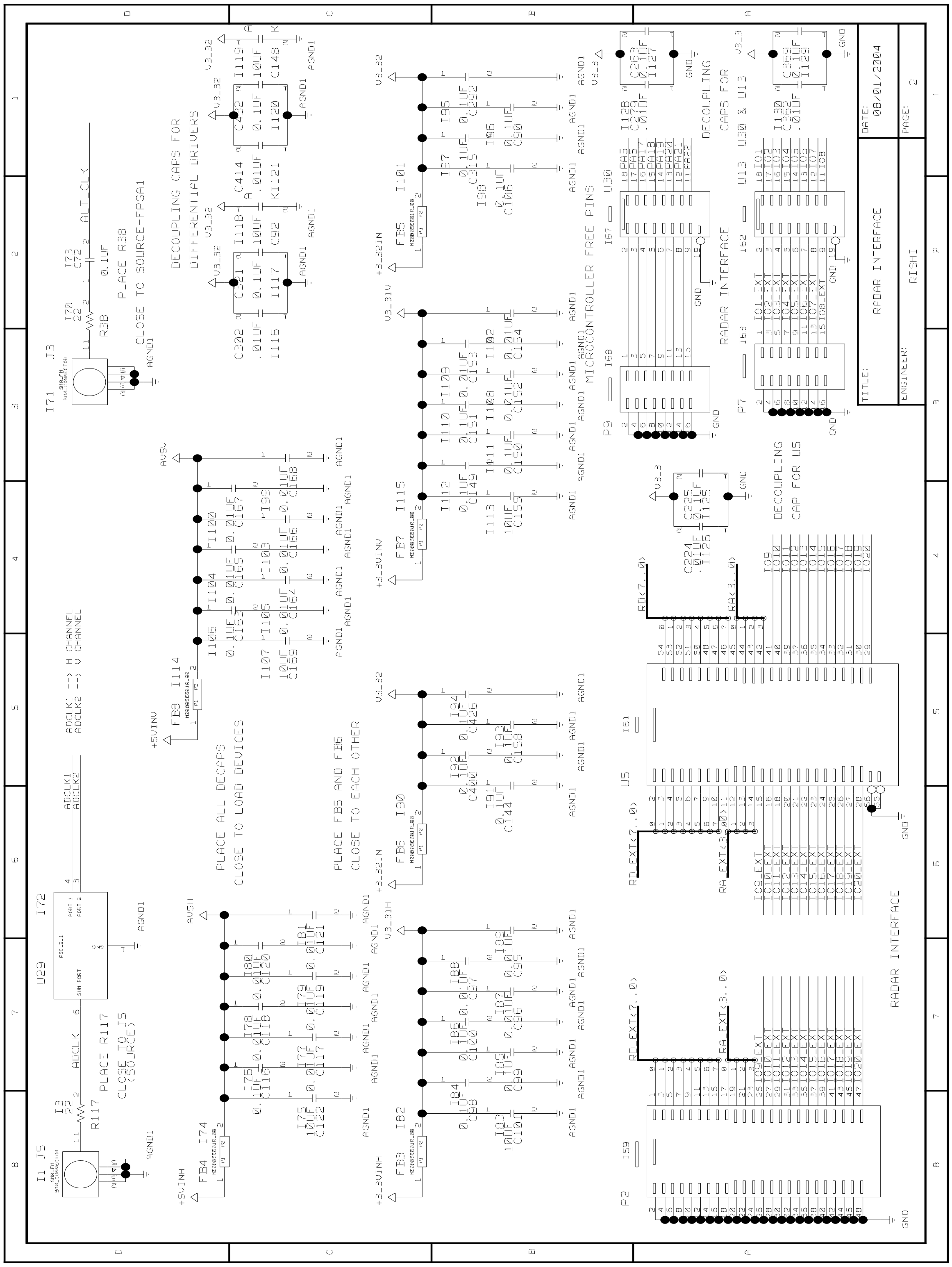


Figure A.2. Radar Interface & Power filtering (Analog section).

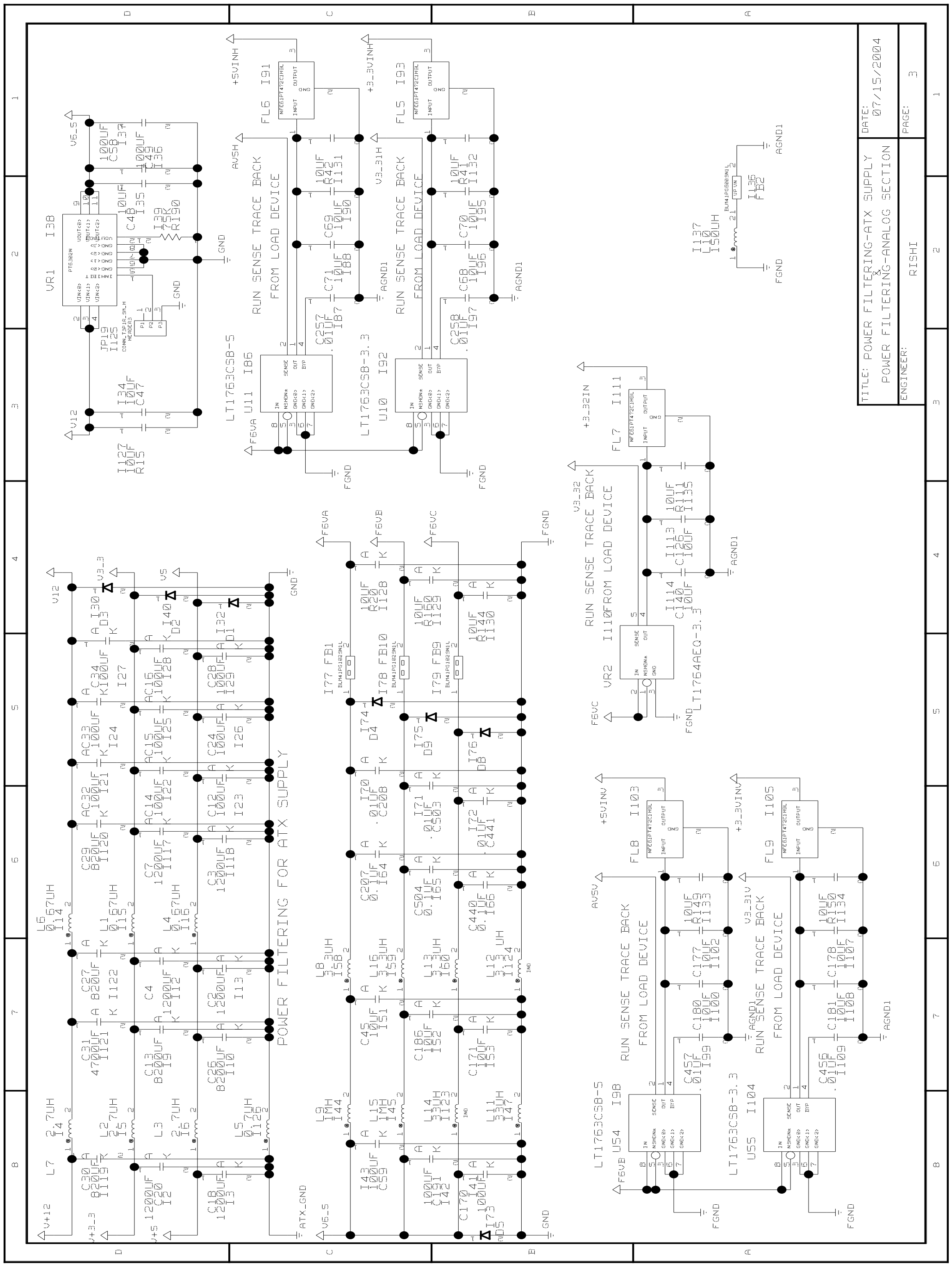


Figure A.3. Power filtering (Analog section).

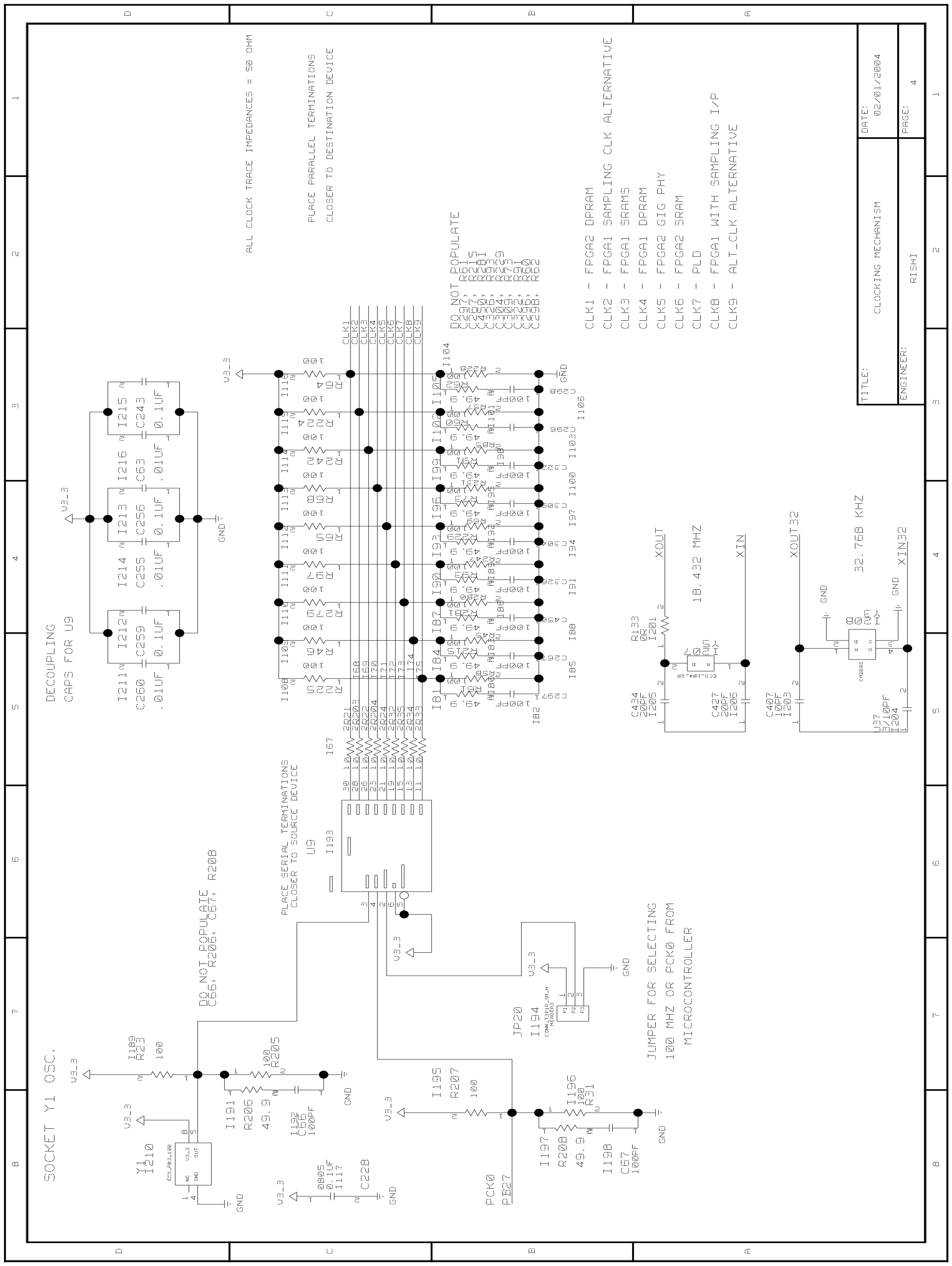

ALL CLOCK TRACE IMPEDANCES = 50 OHM
PLACE PARALLEL TERMINATIONS CLOSER TO DESTINATION DEVICE
DO NOT POPULATE
CLK1 - FPGA2 DPRAM
CLK2 - FPGA1 SAMPLING CLK ALTERNATIVE
CLK3 - FPGA1 SRAMS
CLK4 - FPGA1 DPRAM
CLK5 - FPGA2 GIG PHY
CLK6 - FPGA2 SRAM
CLK7 - PLD
CLK8 - FPGA1 WITH SAMPLING I/P
CLK9 - ALT_CLK ALTERNATIVE
TITLE: CLOCKING MECHANISM
DATE: 02/01/2004
PAGE: 4
ENGINEER: RISHI
DECOUPLING CAPS FOR U9
18.432 MHZ
32.768 KHZ
XOUT
XIN
XOUT32
XIN32
PLACE SERIAL TERMINATIONS CLOSER TO SOURCE DEVICE
U9
DO NOT POPULATE C66, R206, C67, R208
JUMPER FOR SELECTING 100 MHZ OR PCK0 FROM MICROCONTROLLER
JP20
SOCKET Y1 OSC.
PCK0
PB27
V3_3
GND

Figure A.4. Clock distribution mechanism.

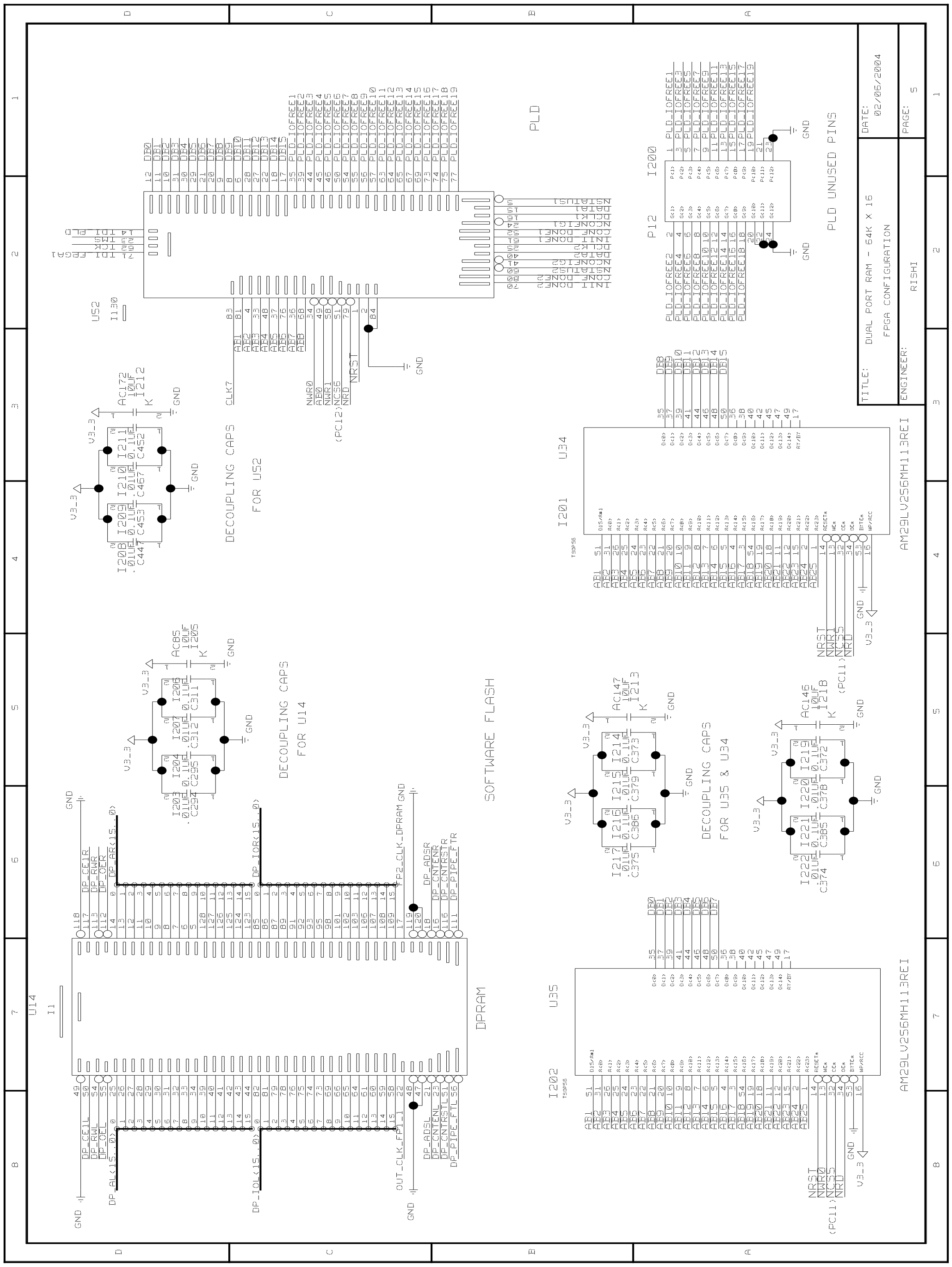


Figure A.5. Dual-Port RAM and FPGA Config. (PLD & FLASH).

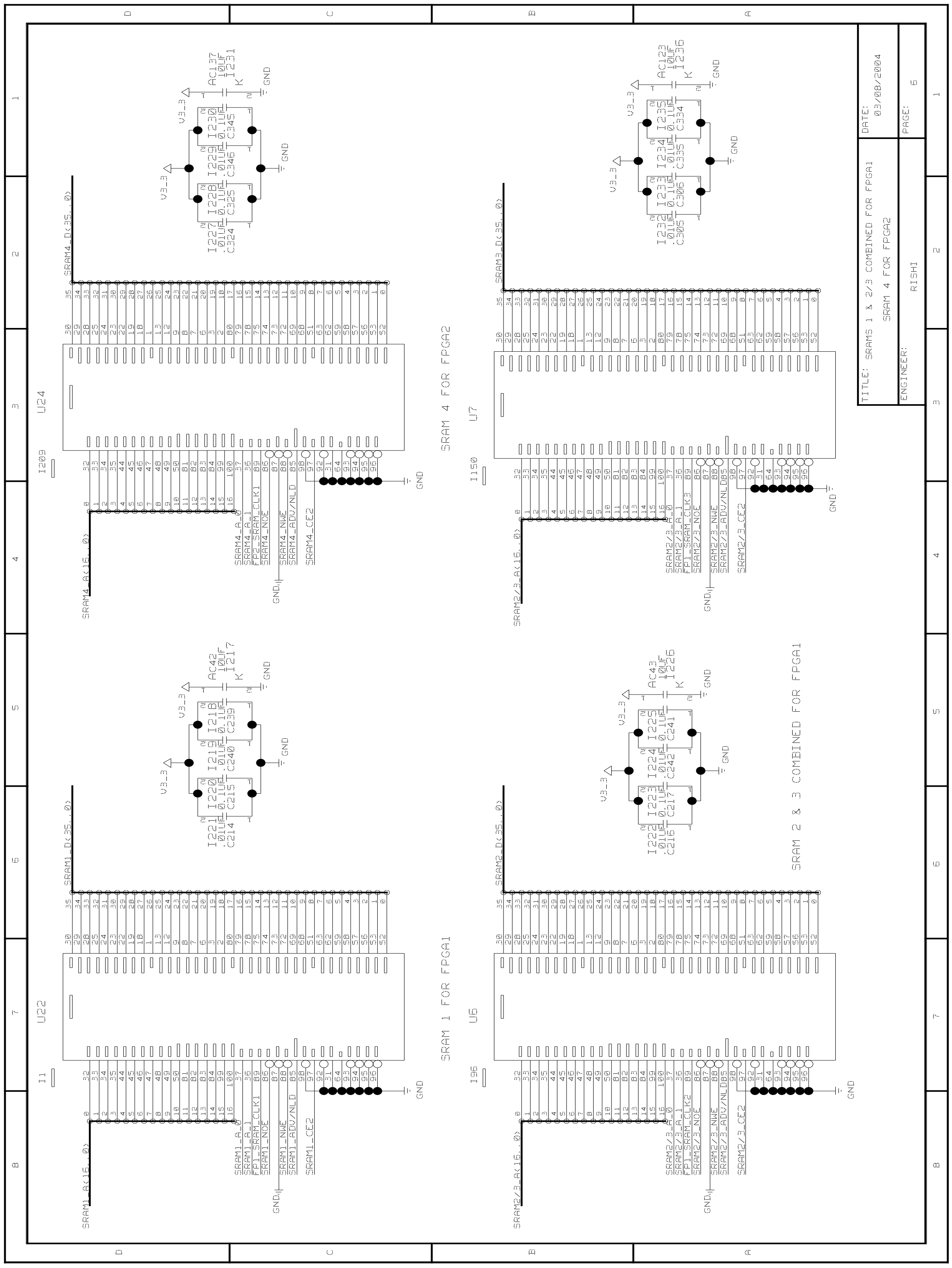

SRAM 1 FOR FPGA1
SRAM 4 FOR FPGA2
SRAM 2 & 3 COMBINED FOR FPGA1
U22
U24
U6
U7
TITLE: SRAMS 1 & 2/3 COMBINED FOR FPGA1 SRAM 4 FOR FPGA2
DATE: 03/08/2004
PAGE: 6
ENGINEER: RISHI

Figure A.6. FPGA SRAMs.

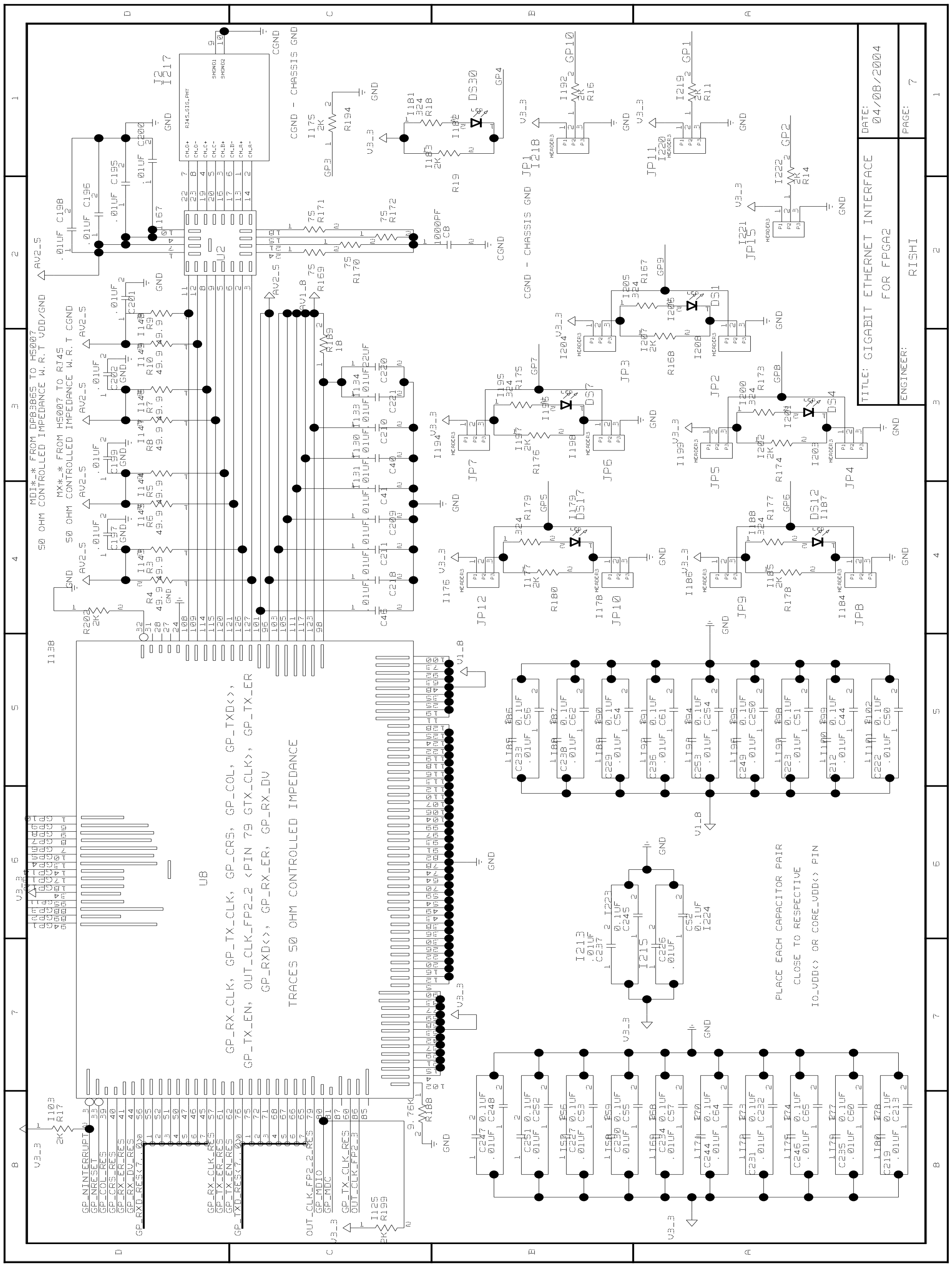


Figure A.7. Gigabit Ethernet Interface.

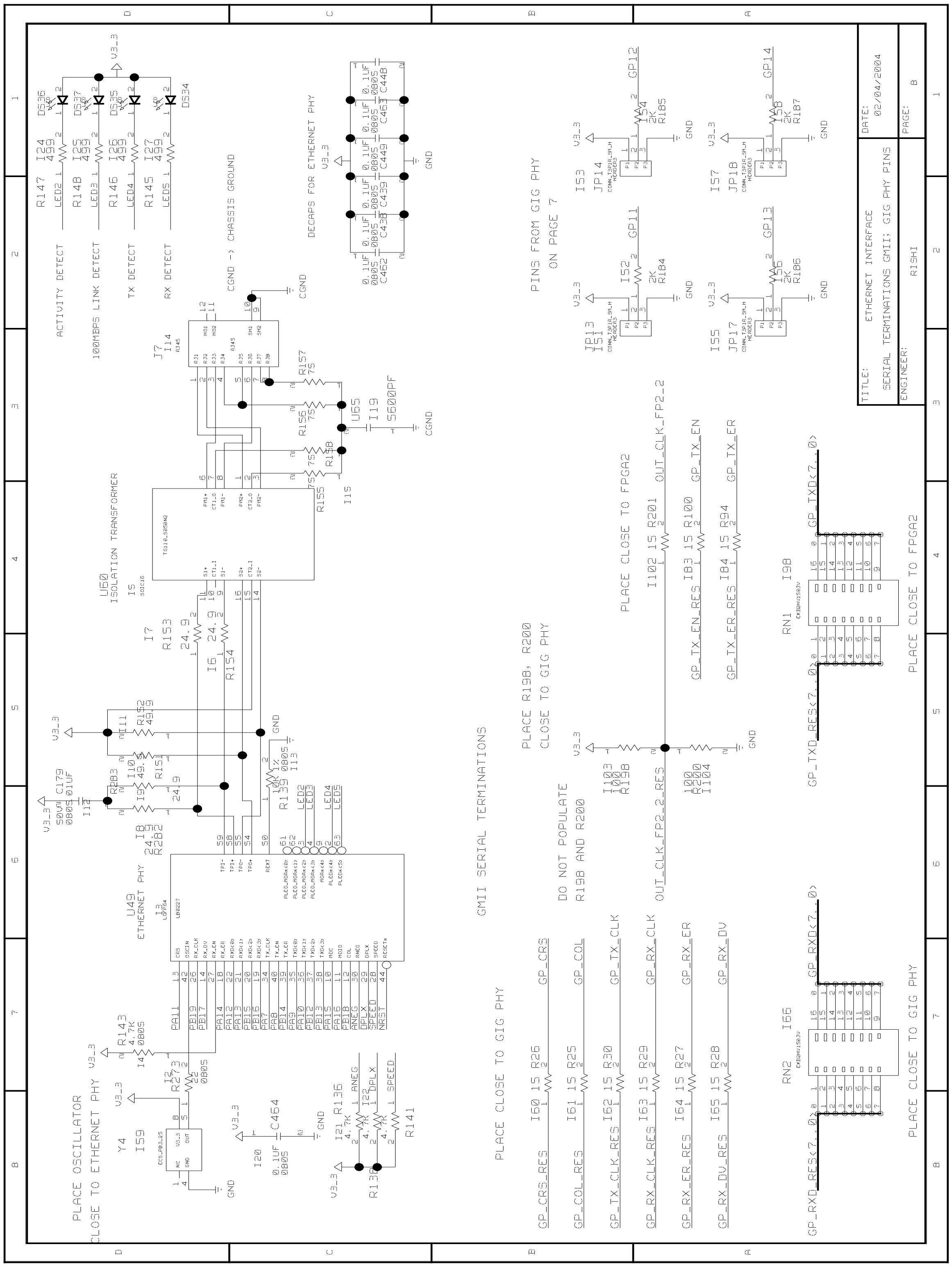


Figure A.8. Ethernet Interface & GMII terminations.

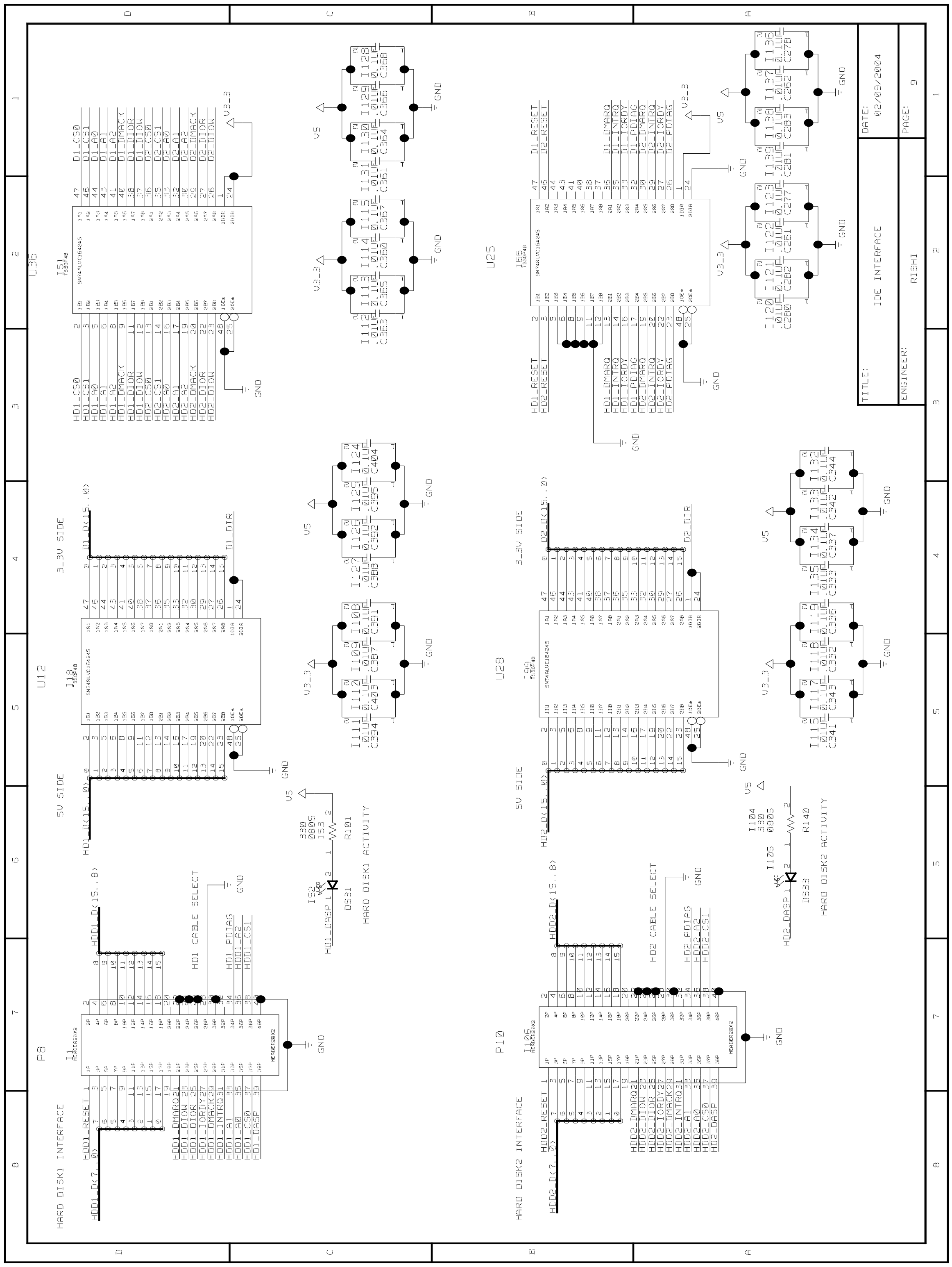


Figure A.9. IDE Interface.

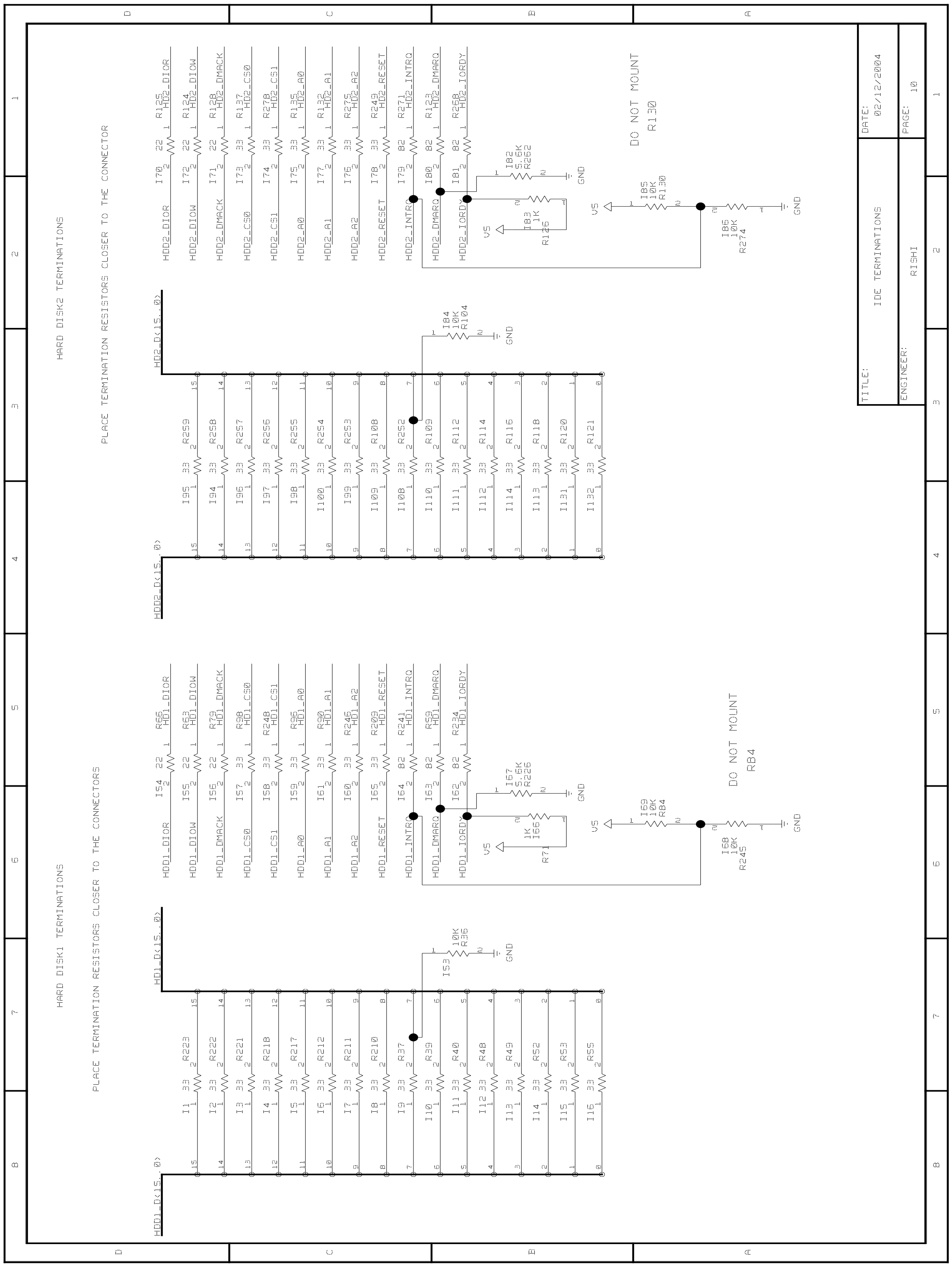
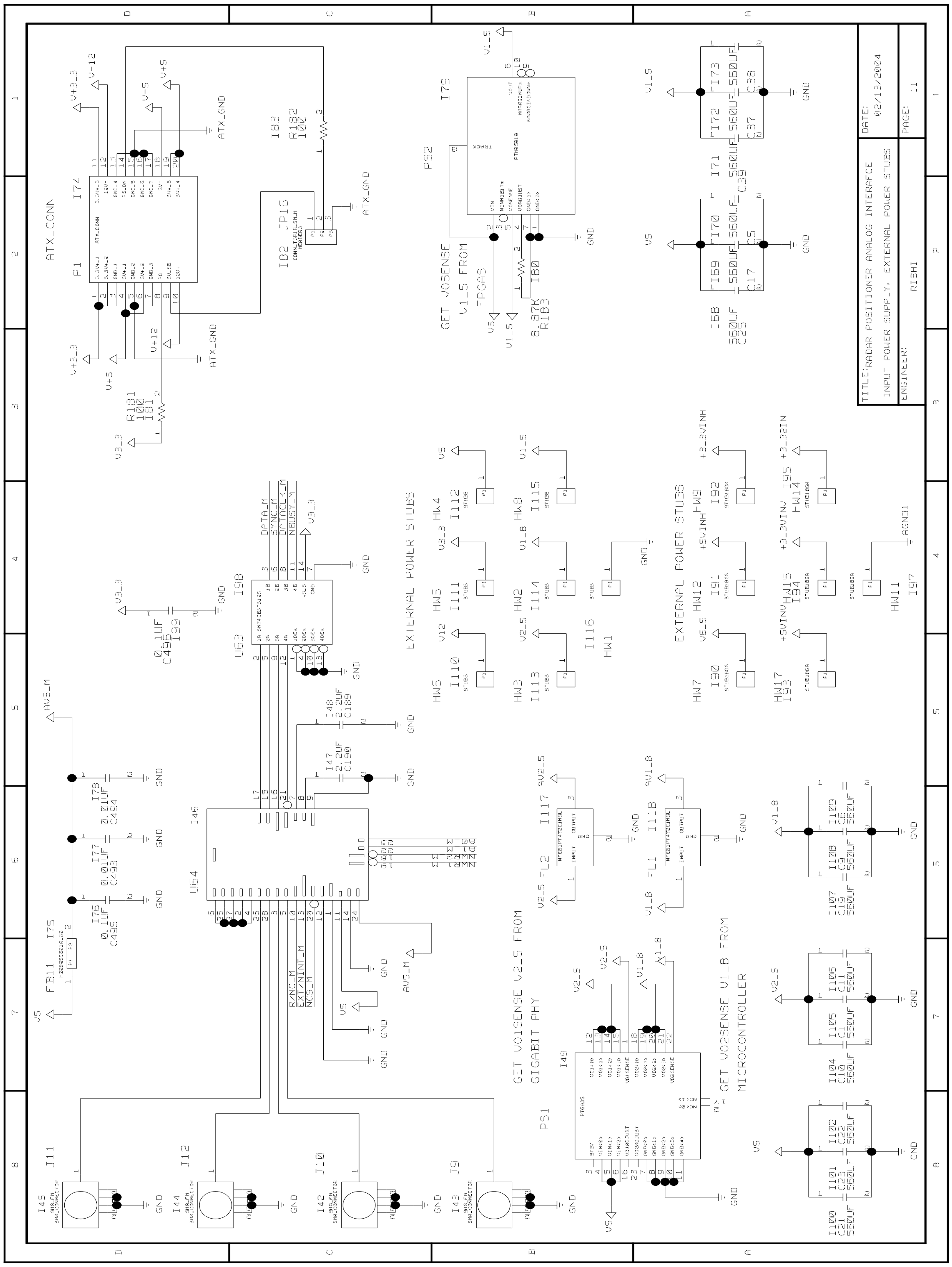


Figure A.10. IDE Terminations.



TITLE: RADAR POSITIONER ANALOG INTERFACE
INPUT POWER SUPPLY, EXTERNAL POWER STUBS
DATE: 02/13/2004
PAGE: 11
ENGINEER: RISHI
ATX_CONN
EXTERNAL POWER STUBS
GET VOSENSE V1_5 FROM FPGAS
GET VO1SENSE V2_5 FROM GIGABIT PHY
GET VO2SENSE V1_8 FROM MICROCONTROLLER

Figure A.11. Radar positioner analog channel, Input Power supply and External Power stubs.

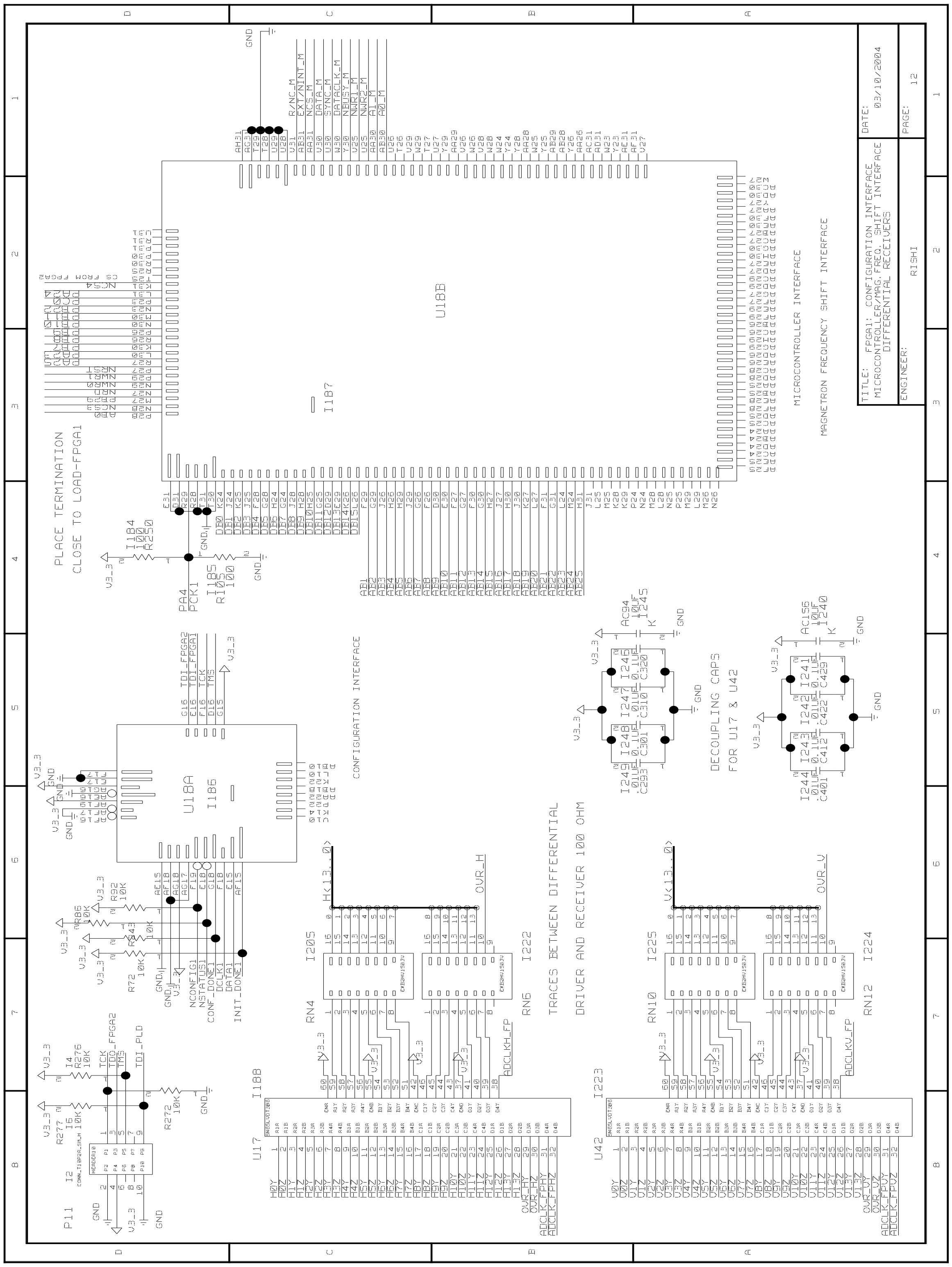


Figure A.12. FPGA1: Configuration, Microcontroller & Radar Positioner interface and Diferential Receivers.

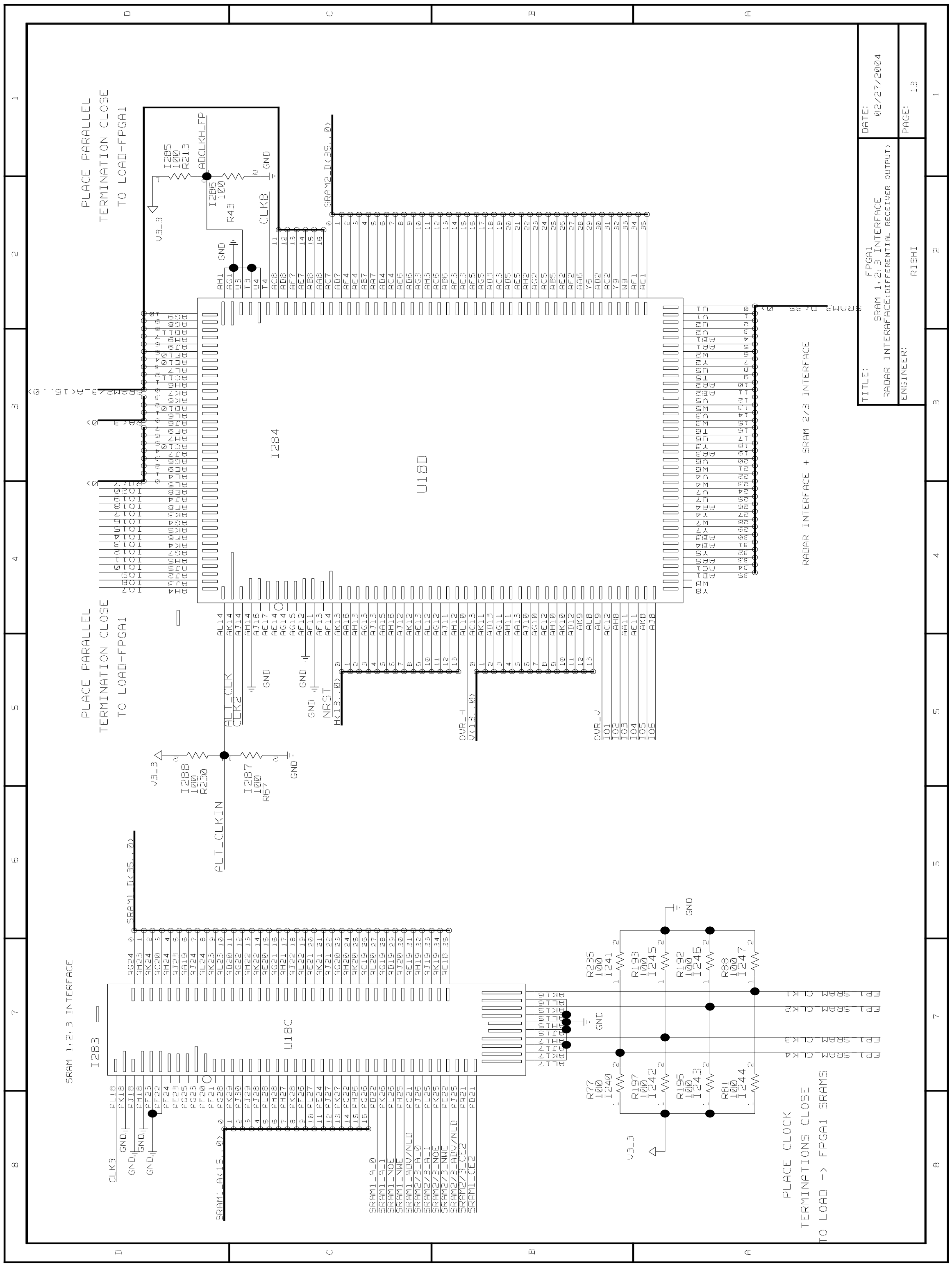

TITLE: FPGA1 SRAM 1,2,3 INTERFACE
RADAR INTERAFACE(DIFFERENTIAL RECEIVER OUTPUT)
DATE: 02/27/2004
PAGE: 13
ENGINEER: RISHI
PLACE PARALLEL TERMINATION CLOSE TO LOAD-FPGA1
U1BD
I284
RADAR INTERFACE + SRAM 2/3 INTERFACE
SRAM 1,2,3 INTERFACE
U1BC
I283
PLACE CLOCK TERMINATIONS CLOSE TO LOAD -> FPGA1 SRAMS
ALT_CLKIN
ADCLKH_FP
CLKB
CLK3
GND
V3_3

Figure A.13. FPGA1: SRAM 1,2 & 3 and Radar interface.

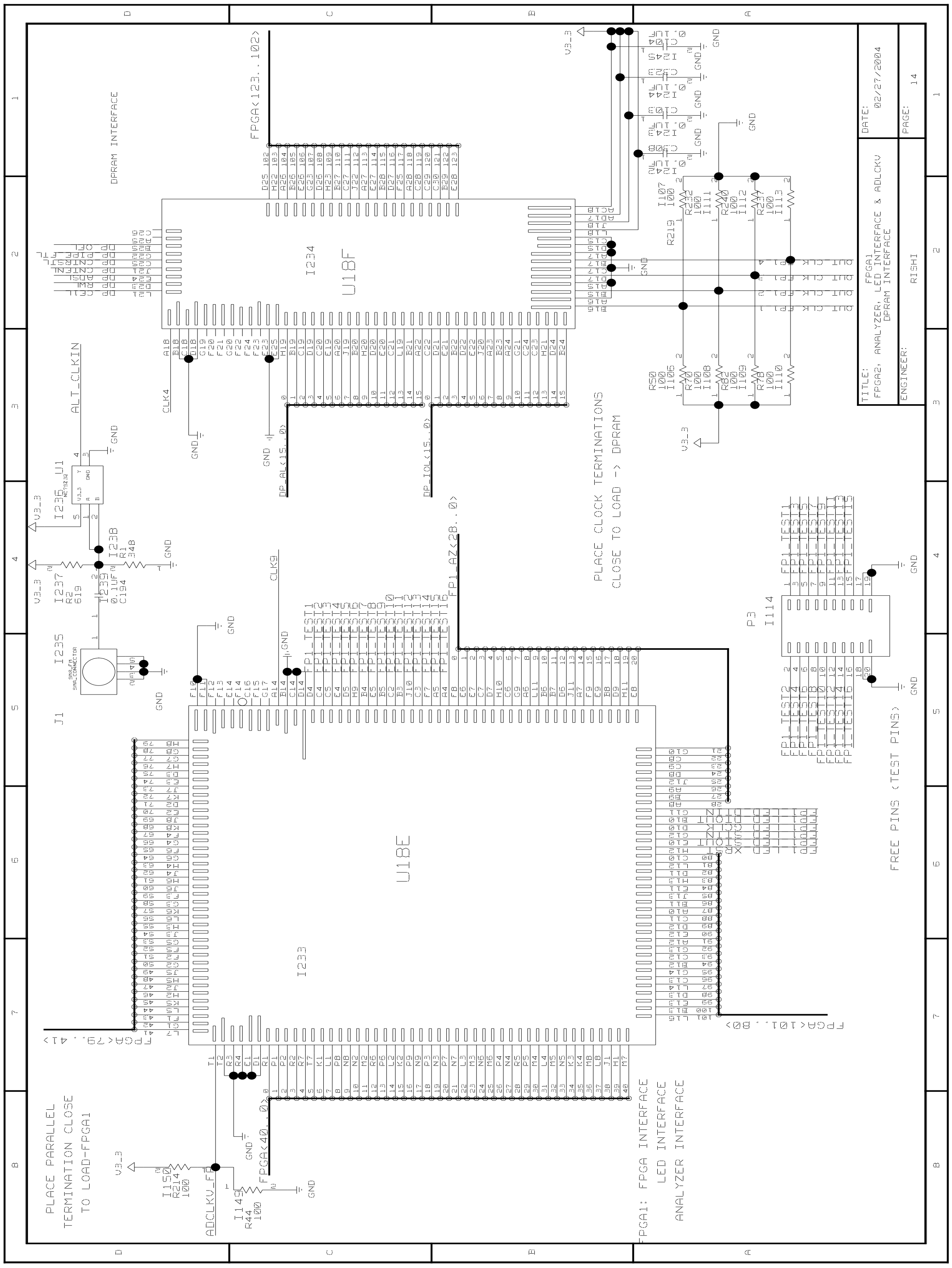


Figure A.14. FPGA1: Analyzer, FPGA2 and LED interface.

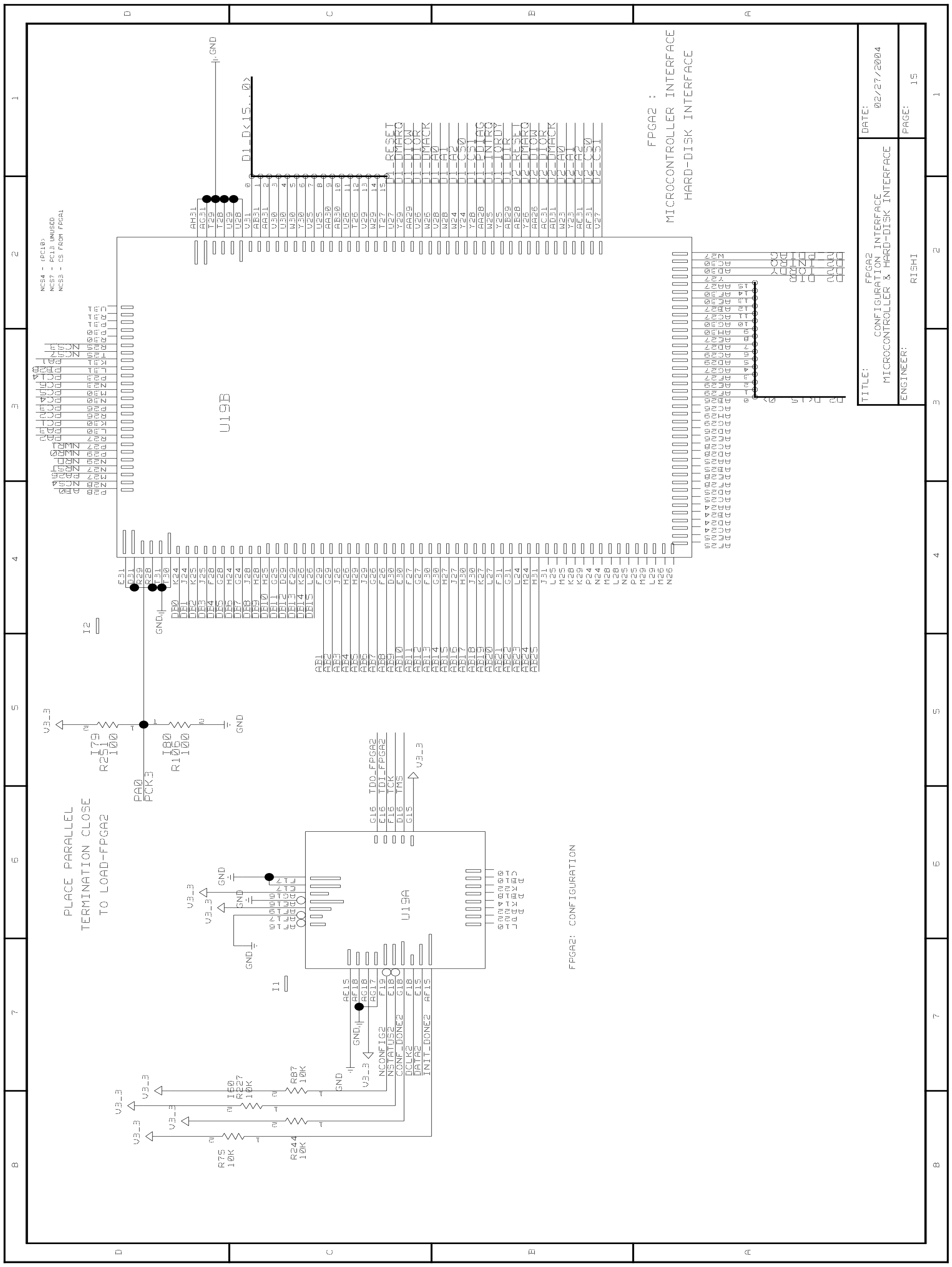


Figure A.15. FPGA2: Configuration, Microcontroller and IDE interface.

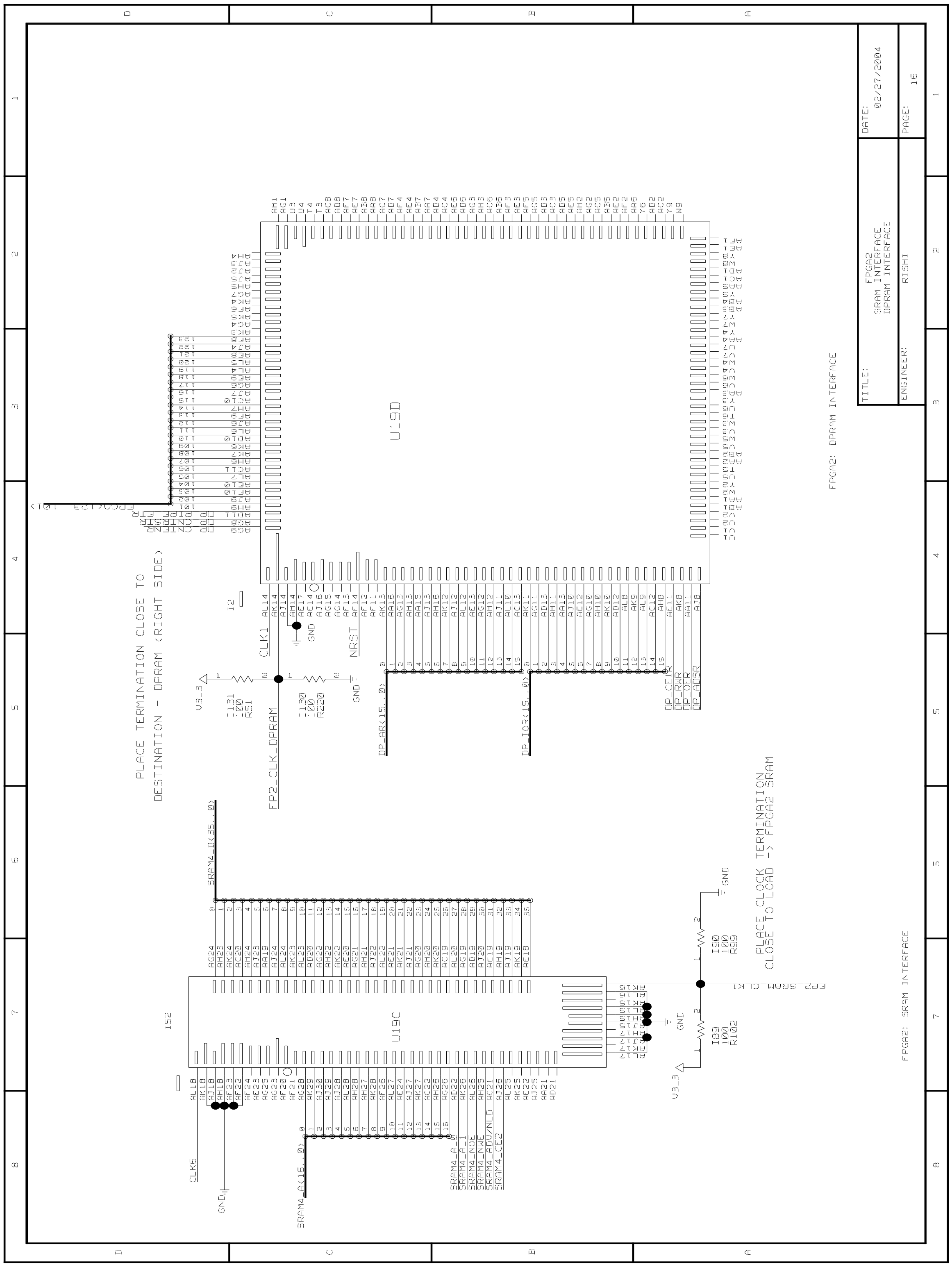
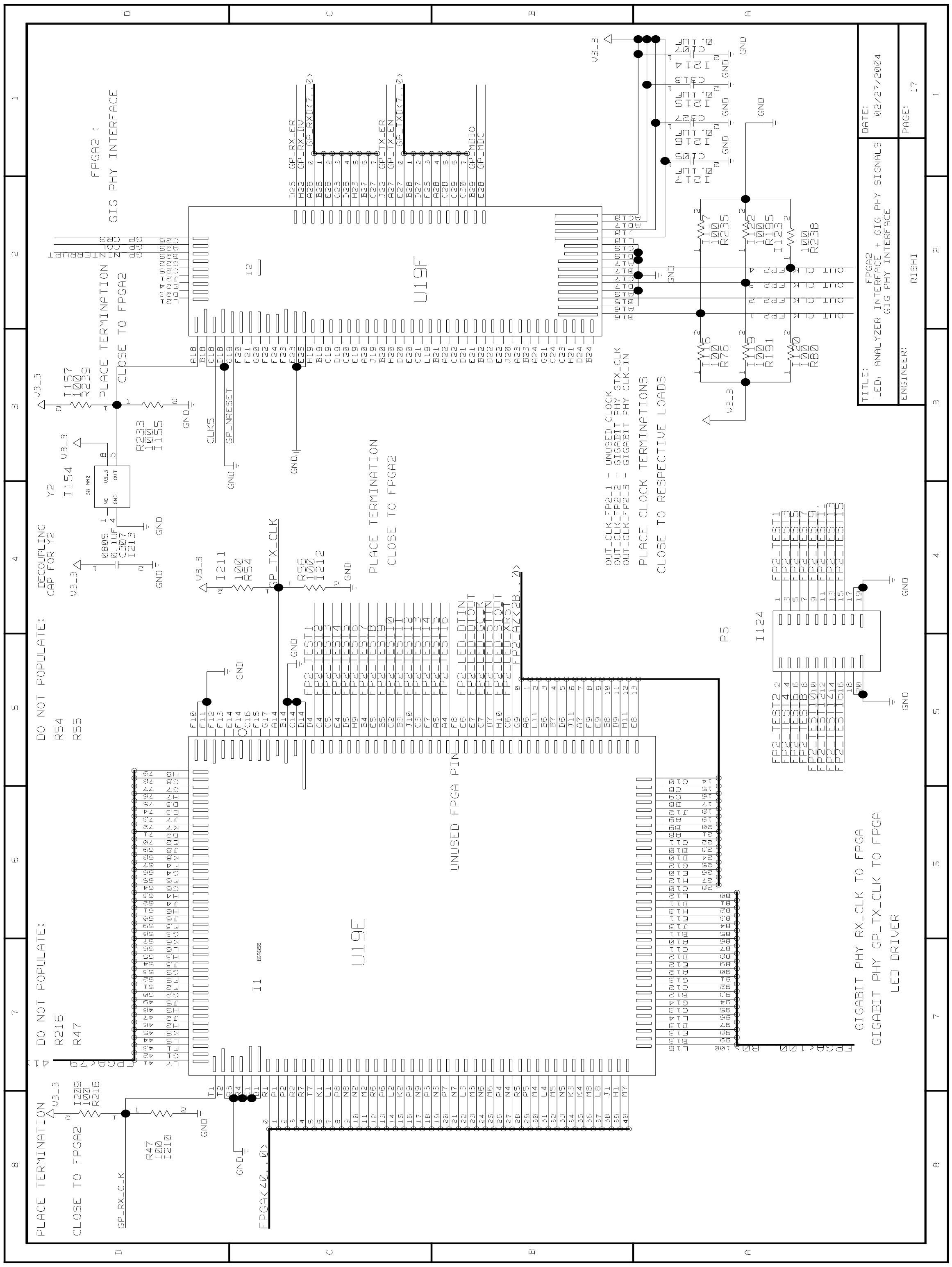


Figure A.16. FPGA2: SRAM and DPRAM interface.



FPGA2 : GIG PHY INTERFACE
U19F
U19E
UNUSED FPGA PIN
PLACE TERMINATION CLOSE TO FPGA2
DO NOT POPULATE: R54 R56
DO NOT POPULATE: R216 R47
DECOUPLING CAP FOR Y2
PLACE CLOCK TERMINATIONS CLOSE TO RESPECTIVE LOADS
OUT_CLK_FP2_1 - UNUSED CLOCK
OUT_CLK_FP2_2 - GIGABIT PHY GTX_CLK
OUT_CLK_FP2_3 - GIGABIT PHY CLK_IN
GIGABIT PHY RX_CLK TO FPGA
GIGABIT PHY GP_TX_CLK TO FPGA
LED DRIVER
TITLE: FPGA2 LED, ANALYZER INTERFACE + GIG PHY SIGNALS GIG PHY INTERFACE
DATE: 02/27/2004
PAGE: 17
ENGINEER: RISHI

Figure A.17. FPGA2: Analyzer, LED and Gigabit Ethernet PHY interface.

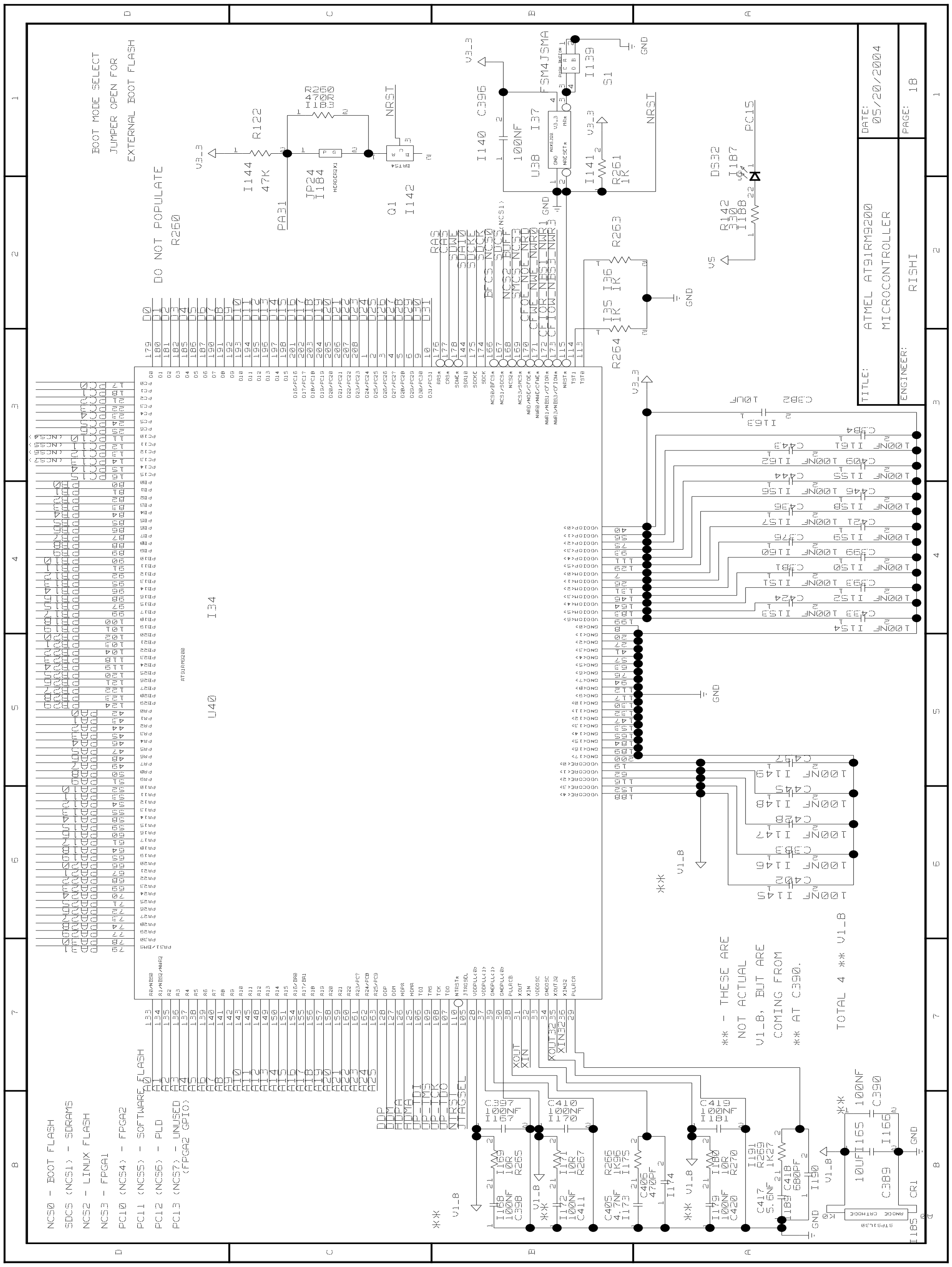


Figure A.18. AT91RM9200 - Microcontroller.

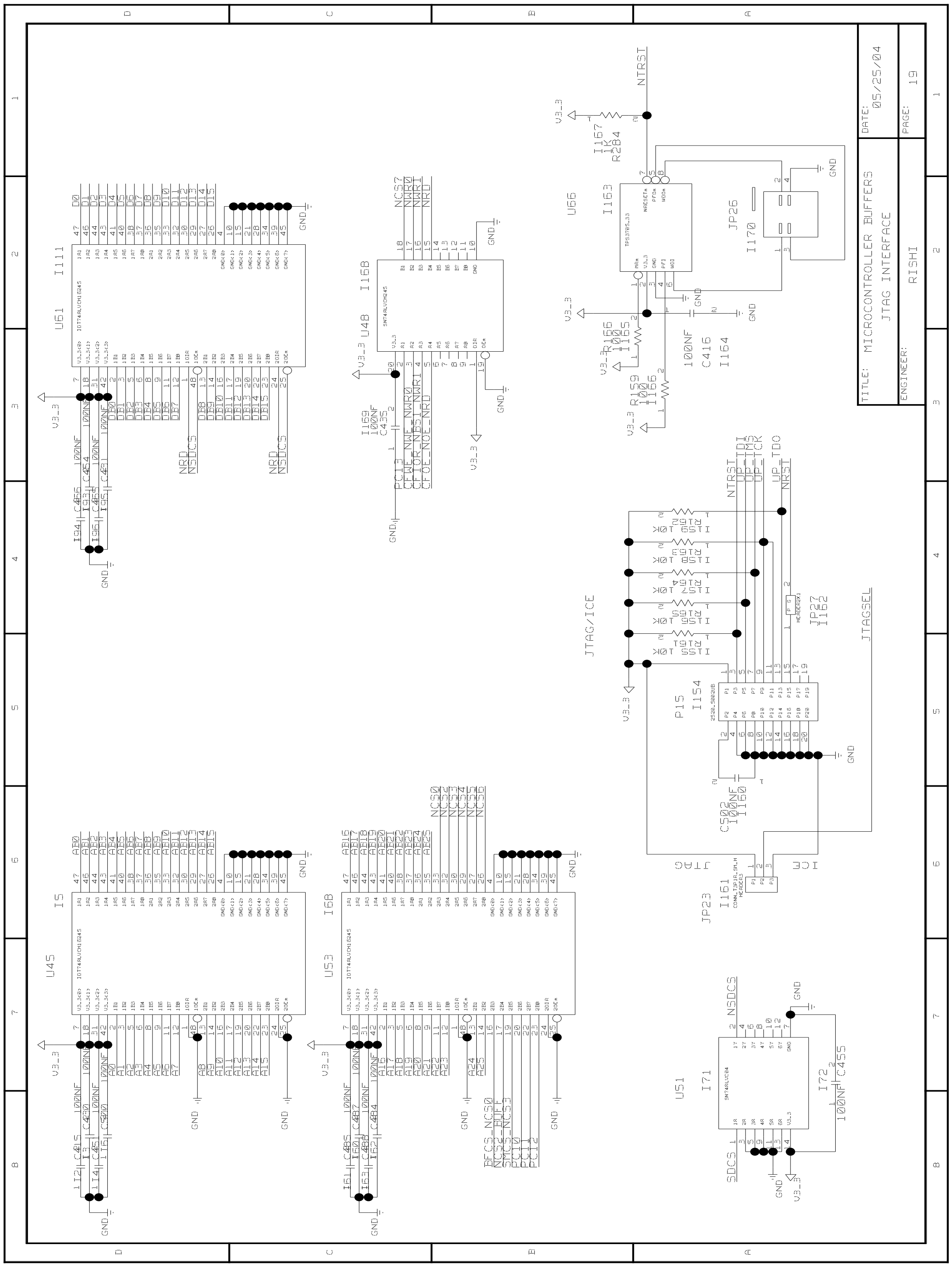


Figure A.19. Microcontroller: Bus buffers, JTAG interface and Reset controller.

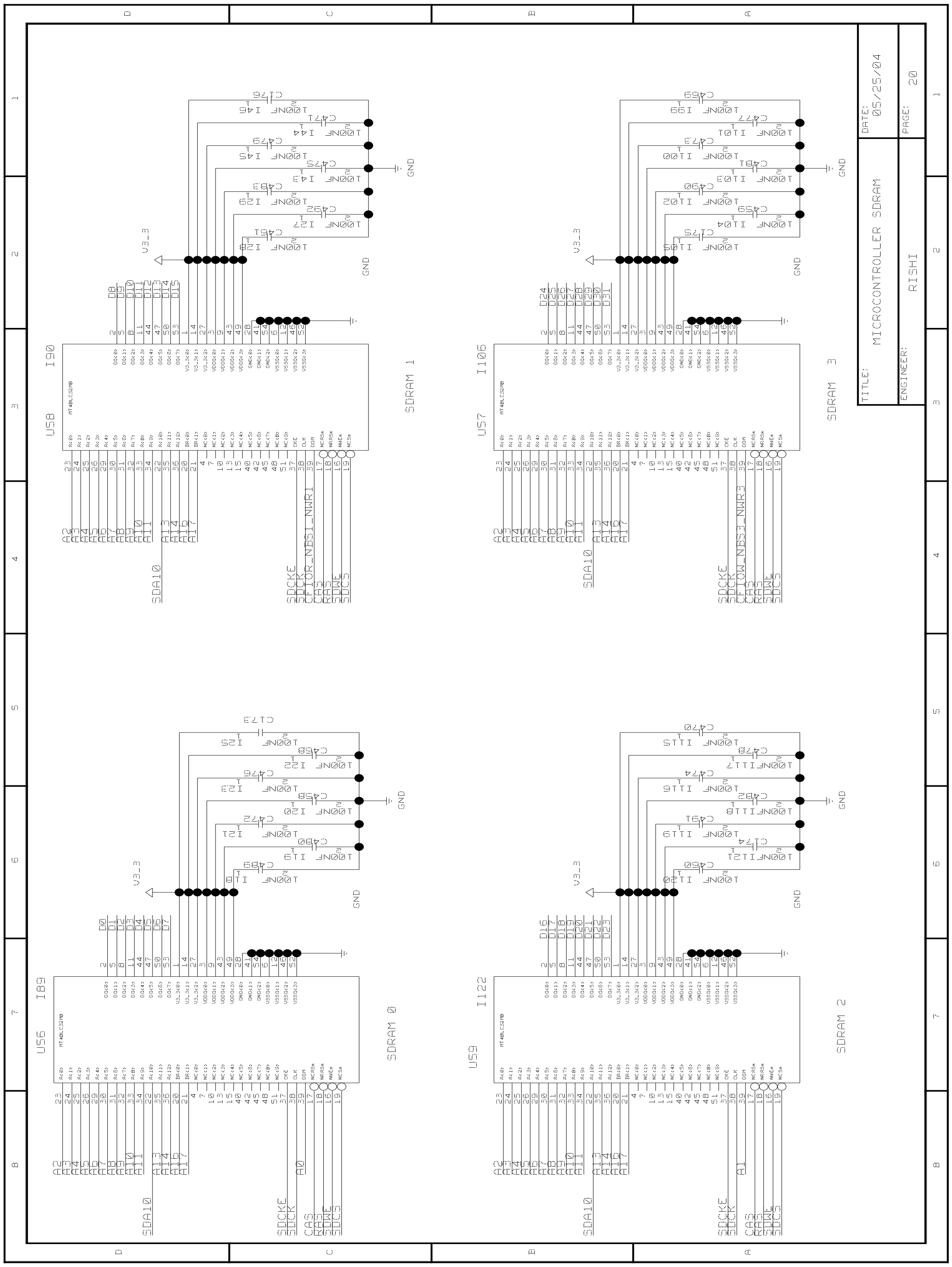


Figure A.20. Microcontroller SDRAMs.

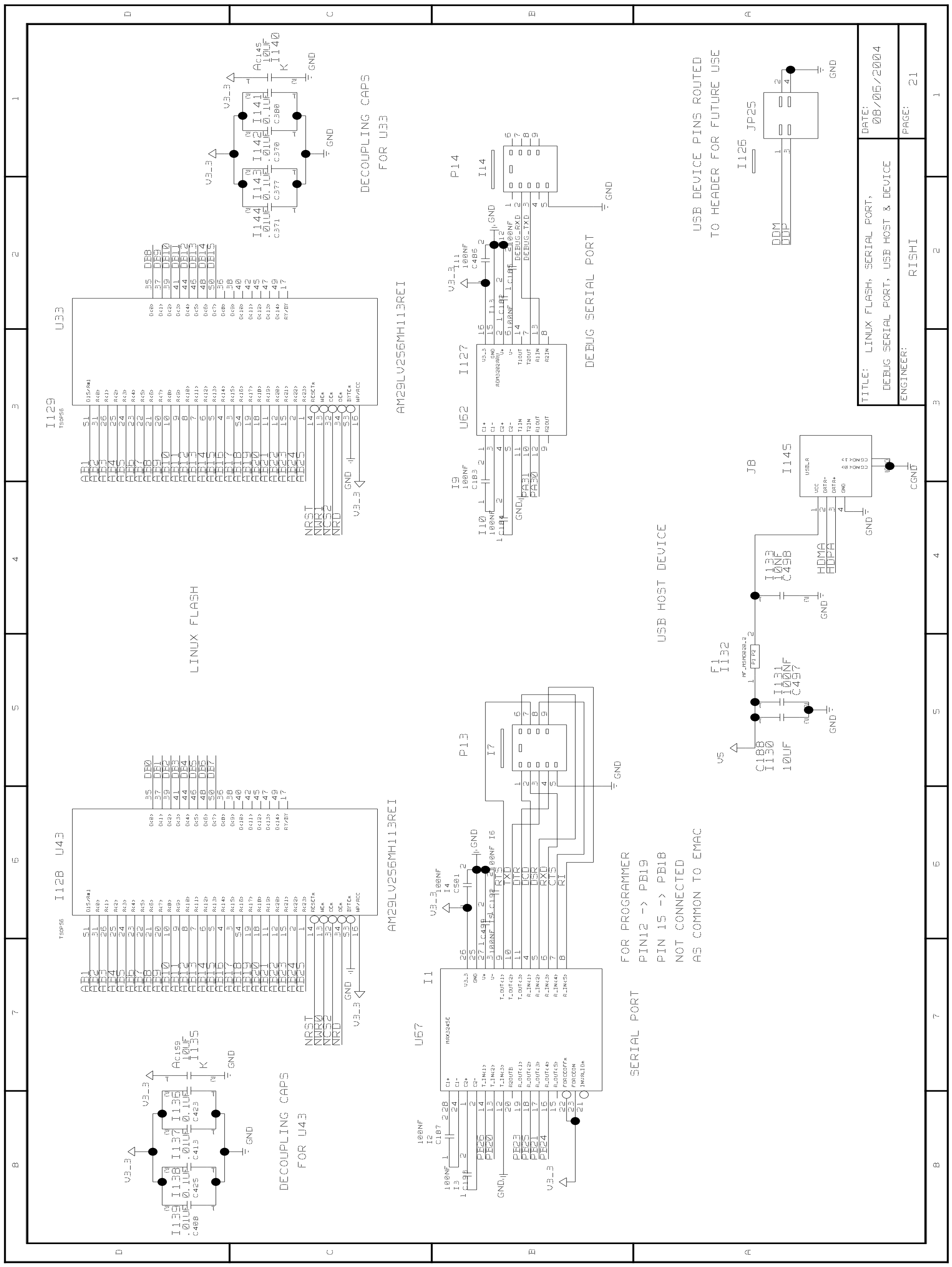

LINUX FLASH
U33
U43
AM29LV256MH113REI
DECOUPLING CAPS FOR U33
DECOUPLING CAPS FOR U43
DEBUG SERIAL PORT
SERIAL PORT
USB HOST DEVICE
USB DEVICE PINS ROUTED TO HEADER FOR FUTURE USE
FOR PROGRAMMER PIN12 -> PB19 PIN 15 -> PB18 NOT CONNECTED AS COMMON TO EMAC
P13
P14
U62
U67
JP25
J8
TITLE: LINUX FLASH, SERIAL PORT, DEBUG SERIAL PORT, USB HOST & DEVICE
DATE: 08/06/2004
PAGE: 21
ENGINEER: RISHI

Figure A.21. Linux FLASH, Serial & Debug Serial ports and USB Host & Device interfaces.

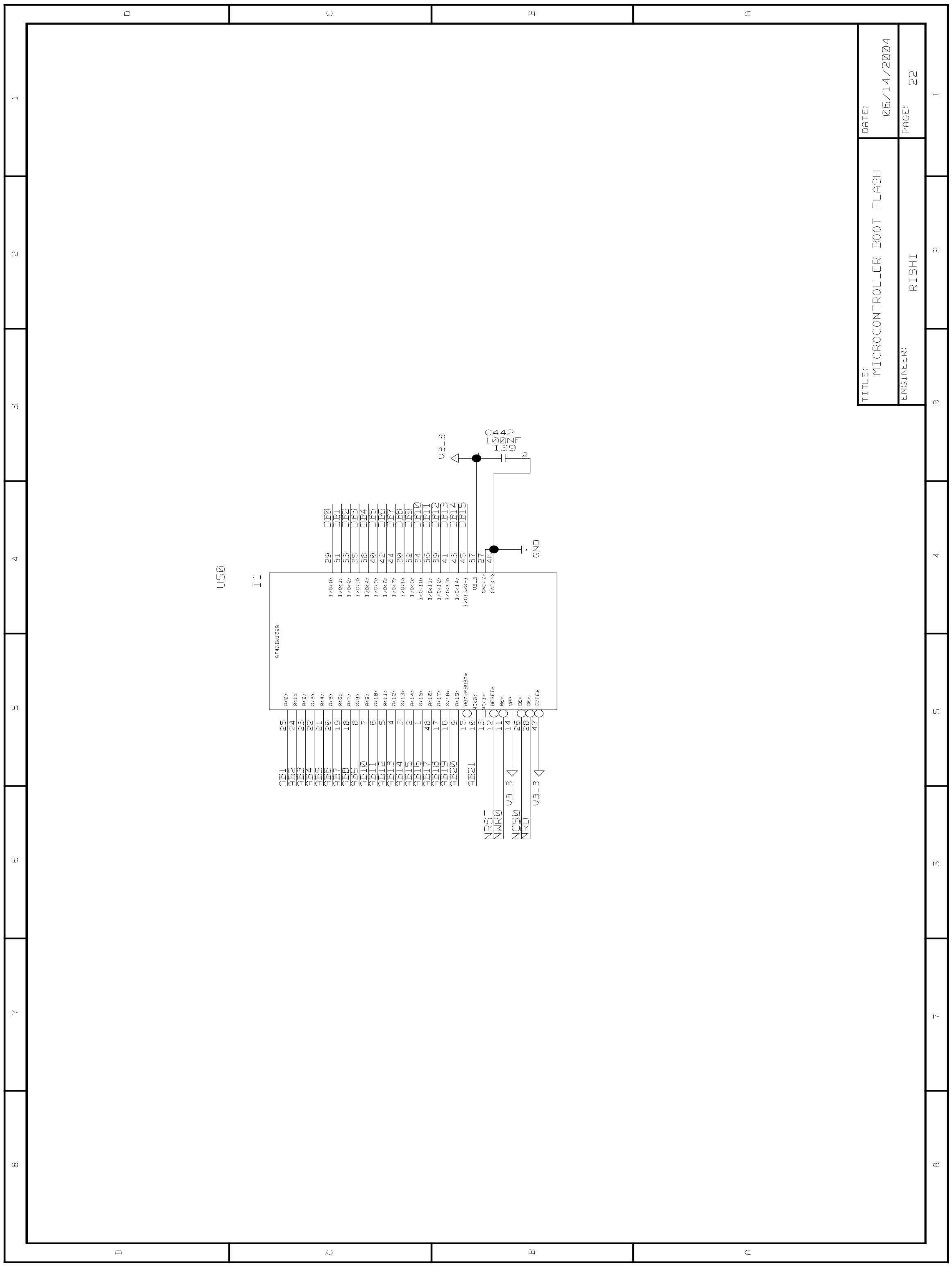


Figure A.22. Microcontroller Boot FLASH.

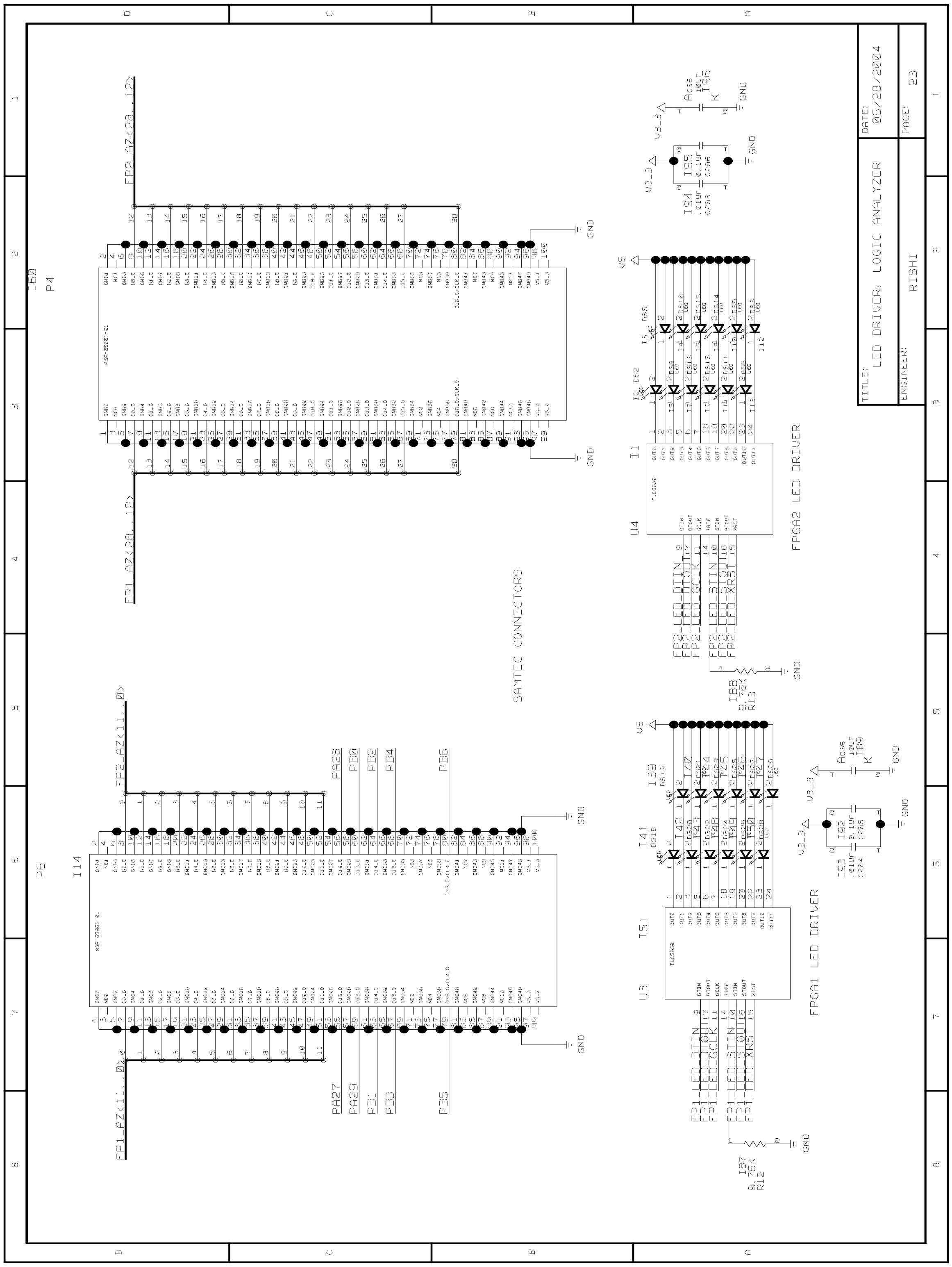


Figure A.23. Samtec connectors for Logic Analyzer and LED drivers.

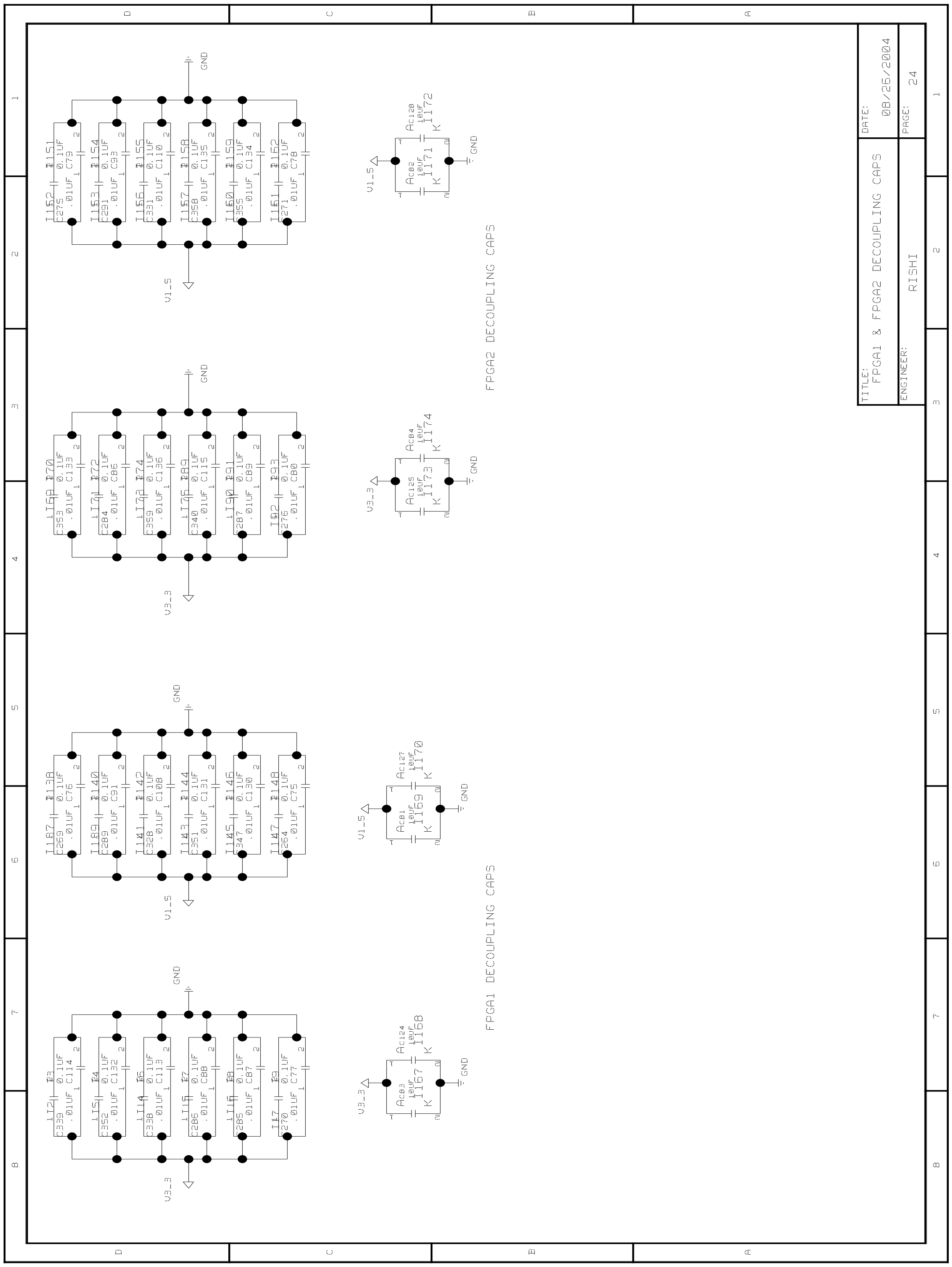


Figure A.24. FPGA1 and FPGA2 decoupling capacitors.

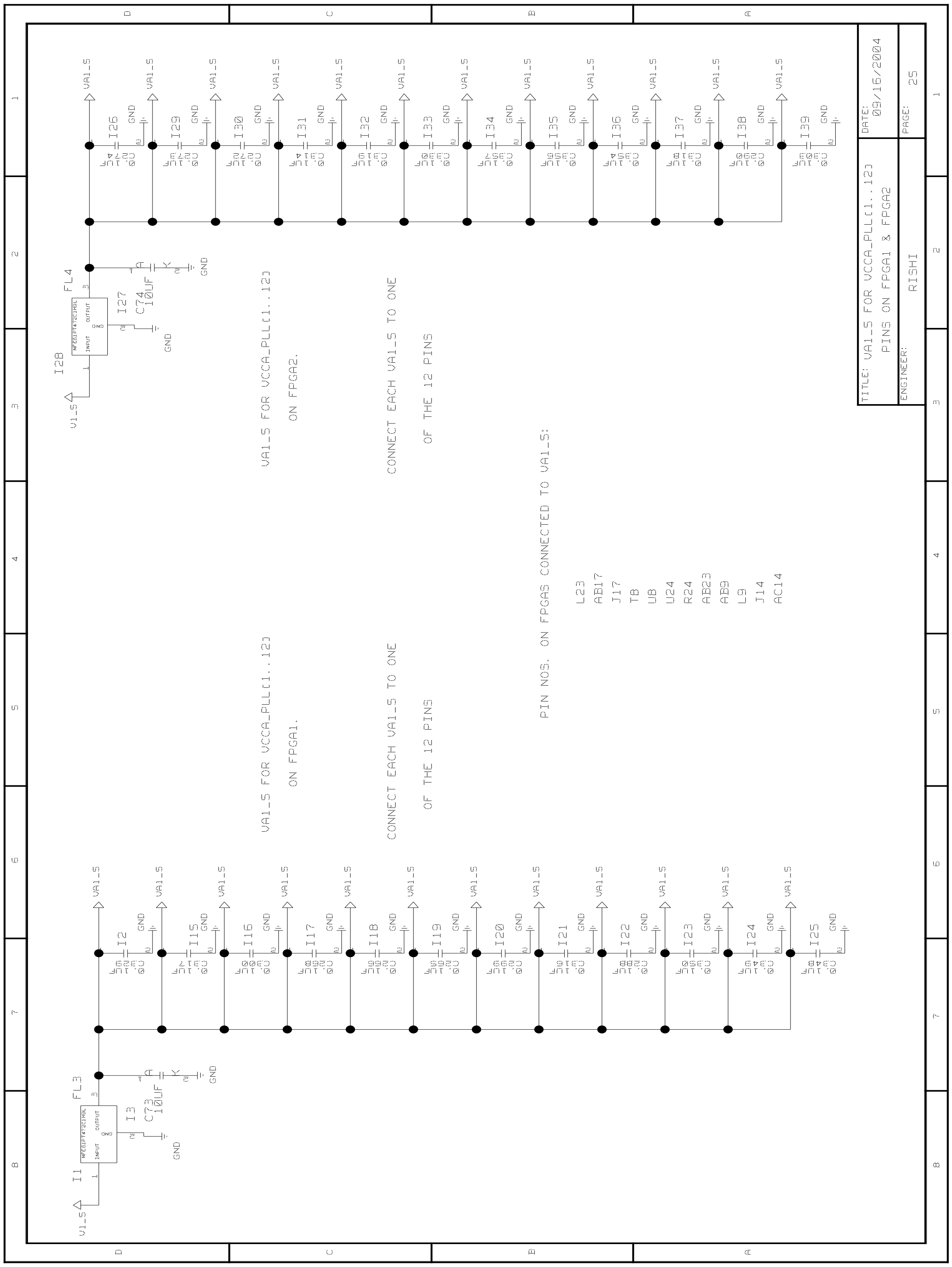

VA1_5 FOR VCCA_PLL[1..12]
ON FPGA2.
CONNECT EACH VA1_5 TO ONE
OF THE 12 PINS
VA1_5 FOR VCCA_PLL[1..12]
ON FPGA1.
CONNECT EACH VA1_5 TO ONE
OF THE 12 PINS
PIN NOS. ON FPGAS CONNECTED TO VA1_5:
L23
AB17
J17
T8
U8
U24
R24
AB23
AB9
L9
J14
AC14
FL4
FL3
C74 10UF
C73 10UF
V1_5
VA1_5
GND
0.1UF
TITLE: VA1_5 FOR VCCA_PLL[1..12] PINS ON FPGA1 & FPGA2
DATE: 09/16/2004
PAGE: 25
ENGINEER: RISHI

Figure A.25. FPGA1 and FPGA2 PLL power filtering.

APPENDIX B

BOARD PICTURE

The picture of the Data Acquisition Board is shown in Figure B.1.

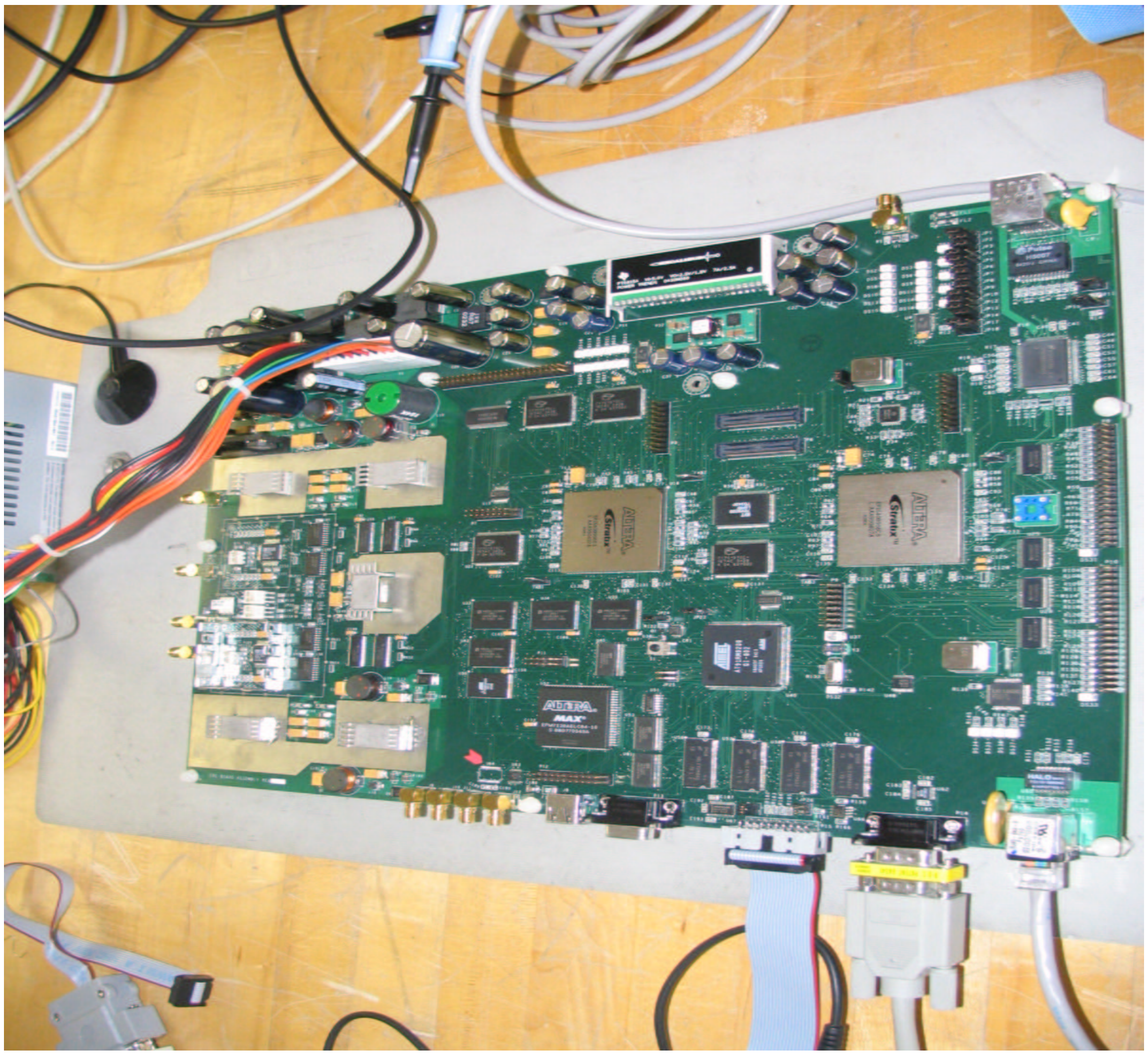


Figure B.1. Data Acquisition Board.

BIBLIOGRAPHY

[1] Agilent, http://agilent.com. *Agilent Technologies 16900 Series Logic Analysis Systems.*

[2] Altera Corporation, http://www.altera.com. *Quartus Development Software*, 2002.

[3] Altera Corporation, http://www.altera.com. *Stratix EP1S40 Device Manual*, July 2003.

[4] Altera Corporation, http://www.altera.com. *Stratix EP1S40 Device Manual*, July 2003.

[5] Altera Corporation, http://www.altera.com. *MAX 7000A Programmable Logic Device Data Sheet*, September 2003.

[6] Analog Devices, http://analog.com. *AD6645 - 14-Bit, 80/105 MSPS A/D Converter Datasheet.*

[7] A.Ramanathan, R.Tessier, D.McLaughlin, J.Carswell and S.Frasier. Acquisition of Sensing Data on a Reconfigurable Platform. In *Proc. of International Geoscience and Remote Sensing Symposium,Sydney, Australia* (July 2001).

[8] ARM Inc., http://www.arm.com. *Advanced Developer Suite Reference Manual.*

[9] Atmel Corporation, http://www.atmel.com. *AT91RM9200 - ARM RISC Microcontroller Manual.*

[10] Atmel Corporation, http://www.atmel.com. *Documentation for Atmel AT91RM9200DK evaluation board.*

[11] Cadence Design Systems Inc., http://www.cadence.com. *Concept-HDL User Manual.*

[12] Capdevila, Juan. Design and Implementation of a Dual Beam Interferometer for Ocean Surface Current Vector Mapping. Master's thesis, University of Massachusetts, Amherst, May 2001.

[13] CASA, http://www.casa.umass.edu/. *System Requirements, CASA ERC, IP1 - Phase A.*

[14] Engineering Research Center (ERC) for Collaborative Adaptive Sensing of the Atmosphere (CASA), http://www.casa.umass.edu/. *Information on Distributed Collaborative Adaptive Sensing (DCAS) Radars.*, May 2004.

[15] GV and Associates Inc, http://www.gvassociates.com. *GVA-375 Reference manual*, April 2004.

[16] J.Hartman, Ahren. A Polarimetric Bistatic Radar Test Bed for Signal-to-Clutter Enhancement Studies. Master's thesis, Northeastern University, Boston, August 1993.

[17] L.Krnan. IP1 Data Acquisition and Real-time Processing Node, Firmware Algorithm Documentation. Tech. rep., ERC, University of Massachusetts, Amherst, March 2004.

[18] L.Krnan. CASA IP1 Digital Receiver Hardware Issues. Tech. rep., ERC, University of Massachusetts, Amherst, November 2004.

[19] Mentor Graphics Corporation, http://www.mentor.com. *Leonardo Spectrum for Altera Reference Manual.*

[20] Mentor Graphics, http://www.mentor.com. *Documentation for Gigabit Ethernet MAC - PE-GMAC0.*

[21] Motorola Inc., http://www.motorola.com/. *MPC9447 - 1:9 LVCMOS Cock Fanout Buffer.*

[22] M.Petronino, R.Bambha, J.Carswell and W.Burleson. An FPGA-based Data Acquisiton System for a 95 Ghz W-band Radar. . In *Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP),Munich, Germany* (April 1997).

[23] M.Rabaey, Jan. *Digital Integrated Circuits, A Design Perspective.* Prentice Hall, 1996.

[24] National Semiconductors, http://www.national.com/. *DP83865 - Gig PHYTER V 10/100/1000 Ethernet Physical Layer.*

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

[26] Ramanathan, Arunachalam. Acquisition of Remote Sensing Data on a Reconfigurable Platform. Master's thesis, University of Massachusetts, Amherst, February 2003.

[27] R.Tessier, and W.Burleson. Reconfigurable Computing and Digital Signal Processing. *Journal of VLSI Signal Processing* (September 2000).

[28] SourceForge.net,http://sourceforge.net/projects/u-boot. *Documentation u-boot, an open source boot program.*

[29] T13, http://www.t13.com. *Information Technology - AT Attachment with Packet Interface - 5 (ATA/ATAPI-5), Rev.3*, February 2000.