

RICE UNIVERSITY

**WARPnet: A Platform for Clean-Slate Deployed Wireless  
Networks**

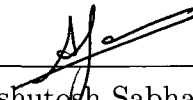
by

**Siddharth Gupta**

A THESIS SUBMITTED  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE

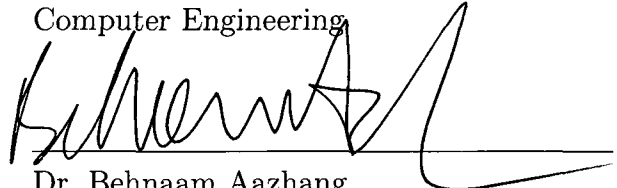
**Master of Science**

APPROVED, THESIS COMMITTEE:




---

Dr. Ashutosh Sabharwal, *Chair*  
Assistant Professor of Electrical and  
Computer Engineering



---

Dr. Behnaam Aazhang  
J.S. Abercrombie Professor of Electrical  
and Computer Engineering



---

Dr. Joseph R. Cavallaro  
Professor of Electrical and Computer  
Engineering

HOUSTON, TEXAS

DECEMBER 2009

UMI Number: 1486039

All rights reserved

**INFORMATION TO ALL USERS**

The quality of this reproduction is dependent upon the quality of the copy submitted.

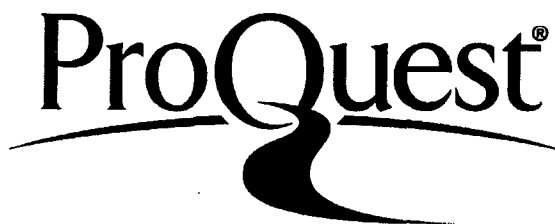
In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1486039

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

## ABSTRACT

WARPnet: A Platform for Clean-Slate Deployed Wireless Networks

by

Siddharth Gupta

There has been a recent paradigm shift within the wireless communications academic community towards implementation-based algorithm validation. In the past, this task was left to industrial affiliates but in order to close the theory to implementation loop faster research groups are actively developing proof-of-concept demonstrations of their theoretical protocols. In this work we present the Wireless Open-Access Research Platform for Networks (WARPnet) that provides all the computational power and data resources needed to prototype novel physical and MAC layers for emerging technologies. The platform is built to be deployed enabling large-scale network-wide experiments. Scheduling experiments and gathering data can be accomplished with a central server connected to the nodes. We characterize the dedicated control channel built for remote control and statistics aggregation, present frameworks for data transfer and implement example applications that show the methodology for benchmarking distributed wireless experiments.

## ACKNOWLEDGEMENTS

I thank my advisor and my committee for their guidance in making this work a reality. I am also very grateful to all my friends and colleagues who saw me through the long hours and late nights. A special thanks to Patrick and Charles for their patience and help in developing the hardware – their advice was incredibly valuable.

Most of all, thanks to my family for their infinite support and encouragement from across the world.

# Contents

---

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Wireless Experimentation . . . . .	1
1.2 WARPnet . . . . .	3
1.3 Related Work . . . . .	4
<b>2 Hardware Platform</b>	<b>7</b>
2.1 Virtex-4 FPGA Board . . . . .	8
2.1.1 Xilinx Virtex-4 FPGA . . . . .	8
2.1.2 Daughtercard Headers . . . . .	10
2.1.3 Gigabit Ethernet . . . . .	10
2.1.4 Dynamic RAM . . . . .	11
2.1.5 Multi-Gigabit Transceivers (MGTs) . . . . .	11
2.1.6 Reconfiguration . . . . .	12
2.1.7 Other Peripherals . . . . .	12
2.1.8 Power Considerations . . . . .	12

---

2.1.9	Hardware Design . . . . .	15
2.2	Backdoor Board . . . . .	15
2.2.1	Axis Etrax 100LX Multi-Chip Module . . . . .	16
2.2.2	Xilinx Spartan-3AN FPGA . . . . .	17
2.2.3	Backdoor Interfaces . . . . .	18
2.2.4	Other Peripherals . . . . .	19
2.2.5	Virtex-4 Link . . . . .	19
2.2.6	Power Considerations . . . . .	20
2.2.7	Board Architecture . . . . .	20
<b>3</b>	<b>System Implementation and Characterization</b>	<b>23</b>
3.1	Implementation Strategies . . . . .	23
3.2	Backdoor Link Performance . . . . .	27
3.2.1	Latency . . . . .	28
3.2.2	Throughput . . . . .	30
3.2.3	Maximum Range . . . . .	34
3.3	Frameworks . . . . .	37
3.3.1	Data Collection Architecture . . . . .	37
3.3.2	Reconfiguration . . . . .	39
3.3.3	Watchdog . . . . .	40
<b>4</b>	<b>Cognitive Radio</b>	<b>41</b>
4.1	Implementation . . . . .	43
4.2	Primary and Secondary Networks . . . . .	44
4.3	Distributed Cognitive Radio Example . . . . .	45
4.4	Centralized Example . . . . .	47
4.5	Performance . . . . .	48
<b>5</b>	<b>Conclusion and Future Work</b>	<b>50</b>

<b>References</b>	<b>51</b>
<b>A Schematics</b>	<b>53</b>
A.1 FPGA Board . . . . .	53
A.2 Backdoor Board . . . . .	76
<b>B Layout</b>	<b>87</b>
B.1 FPGA Board . . . . .	87
B.2 Backdoor Board . . . . .	113

## List of Figures

---

1.1	Characterization of Other Platforms . . . . .	5
2.1	Block Diagram of the Virtex-4 FPGA Board . . . . .	9
2.2	Front of FPGA Board . . . . .	13
2.3	Back of FPGA Board . . . . .	14
2.4	Block Diagram of Backdoor Board . . . . .	16
2.5	Backdoor Board . . . . .	21
3.1	Implementation on Platform . . . . .	24
3.2	Round-Trip Time for Two Nodes with Varying Distances . . . . .	29
3.3	Round-Trip Time for Two Transmitters Feeding One Destination . . . . .	30
3.4	Round-Trip Time for One Transmitter Feeding Two Destinations . . . . .	31
3.5	Throughput for Various Distances . . . . .	32
3.6	Throughput Scaling as Number of Transmitters Increases . . . . .	33
3.7	Data Collection Locations for Maximum Range Experiments . . . . .	35
3.8	Path Loss Measurements with Linear Regression . . . . .	36
3.9	Data Flow Decoupling . . . . .	38
4.1	Protocol Implementation on Platform . . . . .	43



---

4.2	Channel Access with Primary Only . . . . .	45
4.3	Division of Sensing Time . . . . .	46
4.4	Channel Access with Primary and Distributed Secondary Links . . .	47
4.5	Channel Access with Primary and Centralized Secondary Links . . .	48

## List of Tables

---

2.1	Resources available on Virtex-4 FX100 . . . . .	8
2.2	Resources available on Spartan-3AN . . . . .	17
3.1	Specifications of the 900 MHz radio . . . . .	27
3.2	Maximum Number of 160 bit Updates per Second . . . . .	33
3.3	Resource Utilization for Spartan Design . . . . .	39
3.4	Resource Utilization for FPGA Design . . . . .	39
4.1	Throughput for Cognitive Algorithm . . . . .	48

# Introduction

---

There has been a recent paradigm shift in wireless communications research towards hands-on proof-of-concept implementation. In the past, academic institutions primarily focussed on the theoretical aspects of research by advancing the bounds to performance. Companies on the other hand relied on this research to implement next-generation algorithms. Through industry partnerships, the theorists would advance and refine their models. However, an increasingly larger number of institutions want to implement the proof-of-concept design themselves, allowing them to close the theory – implementation loop faster.

In this work we present a new wireless communication platform that not only enables implementation of novel wireless algorithms but also provides all the functionality needed to prototype algorithms at a network-scale.

## 1.1 Wireless Experimentation

There are two schools of thought for validating algorithm performance: (1) model-based simulations and (2) real-time hardware implementations. While model-based simulations work well in checking the feasibility of implementing a theoretical framework, it has its limitations as not all real-world effects can be accurately represented

---

in these models. Thus the protocol implementations cannot be thoroughly validated.

Real-time validation of the algorithm requires that it be implemented on a testbed targeted at wireless communication. As a testbed will exhibit all real-world effects such as channel variability and RF behavior the real performance of the algorithm can be determined along with its sensitivity to such effects. Mass produced hardware fabricate their designs using the latest ASIC technology as this maximizes performance while reducing power usage. However, the time-to-market for an ASIC is on the order of years which is not ideal for prototyping a novel, previously untested algorithm. Therefore there is a need for a quick turn-around prototyping platform that presents real channel characteristics and environment effects to the algorithm designer.

Thus, we can break down the needs of the platform into three primary categories:

(1) flexibility, (2) observability and (3) control.

- *Flexibility:* The most important ability of the platform must be the flexibility to realize novel algorithms. As new and emerging technologies like cognitive radio are developed, the platform should not limit the possibilities that the technologies enable. Also, as cross-layer algorithms are gaining traction, the platform must enable prototyping at all layers, from the physical layers to the application layer.
- *Observability:* As with the development of any new protocol, there is a need to gather large quantities of data to verify the performance and to minimize inefficiencies. To debug the algorithm data generated at the bit-level, packet-level or application-level must be accessible. Real-time aggregation of statistics will also assist the designer in tracking down the bugs. Finally, this data must be available for download to perform offline analysis.
- *Control:* Once implemented, most algorithms need to be tweaked in-system.

Modifying system parameters at short timescales, creating heterogenous networks by remotely flashing firmware and modifying the network layout are control abilities that must be enabled by the platform. Additionally, these functions should be available in real-time. Once stable, stress-testing the implementation by conducting long-term experiments is crucial. Therefore, the platform must have the ability to schedule experiments and gather the results.

Many of the objectives above can claim have been met by other platforms, but we must accomplish these at a *network-scale*. Thus, a deployed network of nodes must maintain the same functionality that is often desired in a laboratory setting. An algorithm designer must have remote control of the full network of nodes from a central location including the abilities to modify parameters, gather performance numbers and update the firmware.

## 1.2 WARPnet

In this work we present the Wireless Open-Access Research Platform for Networks (WARPnet), a platform that enables network-scale wireless experimentation. Our contribution is developing the hardware platform and software frameworks to fulfill the requirements described above. To support the needs of wireless networks the platform is based around a Field Programmable Gate Array (FPGA), a reprogrammable and flexible device, that supports custom physical and MAC layers. Flexibility entails customization at every layer of the network stack, a feature that is enabled by the FPGA. The RF interfaces are added as daughtercards to the FPGA Board introducing modularity for future upgrades. Sharing observed data and sending control information require a reliable control channel; in-band and out-of-band dedicated control channels are enabled by the Backdoor Board. Additionally, a central server connecting to every node via the Backdoor Board can accomplish the experimentation

functions.

Chapter 2 describes the hardware in detail. We validate the performance of the platform and characterize the backdoor control channels in Chapter 3. In Chapter 4 we explore a cognitive radio experiment that exercises all the features of WARPnet with a real-time implementation. Finally, we conclude in Chapter 5.

## 1.3 Related Work

There are several platforms currently being used by researchers for the purposes of evaluating wireless algorithms. GNU Radio [1] is an open-source effort that implements the key components of a wireless stack on a host PC. The platform itself is responsible for filtering and transferring the raw filtered samples up to the PC for processing. While this platform does support custom physical and MAC layers, the latency from the antenna to the PC is significant; hence the node will not be able to return an acknowledgement in the time required by wireless standards. Therefore, unidirectional high bandwidth flows are possible but building networks of the nodes to perform real-time random access experiments is not feasible.

ORBIT [2] is another experimental testbed that uses a large number of indoor nodes to recreate several realistic scenarios. The nodes are based on a Linux PC and hence OSI stack layers 3 and above are flexible. The platform uses an off-the-shelf wireless card restricting user access to lower layers. While users can modify certain limited MAC parameters, evaluation of novel algorithms at the physical and MAC layers is not possible – a key to enabling future technologies. CogNet [3] is software protocol implementation for cognitive radio implemented for the GNU Radio platform and hence inherits its drawbacks. KUAR [4] is a hardware-based project but have not published results of their development work.

Rice University partnered with Technology For All to deploy a multi-hop wireless

network [5] in Houston hence it is capable of network-wide experiments. However, it is built using off-the-shelf hardware and thus the protocols and algorithms are fixed to the wireless cards installed. Another project at Rice University is the Wireless Open-Access Research Platform (WARP) [6] which was the forerunner to this work. While it has much of the capability that is needed for custom wireless algorithms, it does not have the functionality to monitor and control a deployed network of nodes.

There are a few European projects also focussed on testbed development. Caban et. al. [7] decouple the algorithm from the antenna where all the processing is done offline in MATLAB while using generic wireless interfaces to capture actual channel effects. While this allows for deployments, the latency between the antenna and offline processor is too large to enable real-time network-level communication. The WINNER project [8] does hardware development but its primary thrust is development of channel models related various standards.

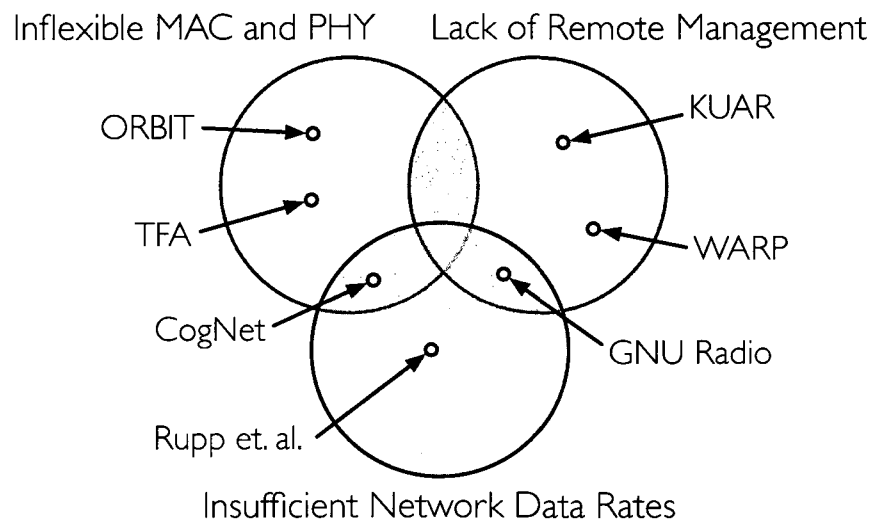


Figure 1.1: Characterization of Other Platforms

Figure 1.1 graphically illustrates the shortcomings of the various platforms discussed above. The three categories are: (1) the inability to implement novel physical and MAC layers, (2) the lack of remote control and observation and (3) the data rates sustained in real-time are not comparable to current generation wireless systems.

Much research has been done comparing protocols where a single radio is used [9] for the control and data communication or a dedicated radio interface is used [10]. WARPnet is designed with a dedicated control channel that is on an orthogonal frequency channel (900 MHz) and does not utilize the FPGA for communication. There are two advantages to this: (1) the physical and MAC layers are not interrupted by transmitting or receiving control information and (2) the FPGA is not required to implement a second physical layer in the same logic resources as the experimental physical layer.



## Hardware Platform

---

To support our goals of an experiment-capable network-wide wireless communications platform, we have developed a new hardware platform, Wireless Open-Access Research Platform for Networks (WARPnet). This highly reconfigurable and flexible system is modular and supports a myriad of setups, each tailored to user needs. In this chapter we showcase the architectural details of the WARPnet hardware.

WARPnet is built around three major boards. The Virtex-4 FPGA Board provides the processing horsepower to implement novel wireless physical and MAC layers. Up to 4 RF interfaces or Radio Boards can be installed on the FPGA Board to provide the RF upconverters etc. The Radio Boards developed for the first generation WARP project [6] are compatible with this platform and hence not described here. Finally, the Backdoor Board enables remote control and observation functionality and provides the hooks for extracting experiment information. The Backdoor Board has its own processor, allowing it to run independently of the FPGA Board while maintaining reliable communication channels.

## 2.1 Virtex-4 FPGA Board

Field Programmable Gate Array (FPGA) technology is well suited to prototype platforms. Implemented algorithms can take advantage of hardware parallelism that is commonly available in ASIC designs. The FPGA affords a short design cycle as its logic blocks are reconfigurable speeding up prototype iteration time. This reconfigurability also lowers long-term cost as the same hardware can perform multiple functions without need to recreate a physical chip. The key to maximizing the flexibility is building a modular system where the FPGA has access to multiple and selectable off-chip resources. Thus, WARPnet is based around a large FPGA, while the RF and experimentation hardware connect as daughtercards to the main board. Future upgrades of RF chips does not require a redesign of the FPGA Board itself.

### 2.1.1 Xilinx Virtex-4 FPGA

The foundation of WARPnet is an FPGA from Xilinx. It belongs to the Virtex-4 FX series [11] and is one of the largest Virtex-4 devices available on the market. FPGAs provide direct access to a large number of I/O pins, have reprogrammable logic resources and an onboard processor to implement OS functionality.

Logic Slices	42,176
XtremeDSP Slices	160
Block RAM	376
PowerPCs	2
Ethernet MACs	2
RocketIO Transceivers	20
User I/O	768

Table 2.1: Resources available on Virtex-4 FX100

The key resources available on the Virtex-4 are shown in Table 2.1. The flexibility of the FPGA is realized in the slice resources. Each slice is composed of two 4-input look-up tables (LUTs) and associated logic. These can implement arithmetic

functions or act as distributed RAM. The slices are laid out in an array-like structure and each can be reconfigured to form larger complex systems. FPGA logic design is controlled at the bit level, giving the user the power to decide what resources to use, placement of the design in hardware and the maximum sustainable clock frequency.

Other resources such as XtremeDSP slices, Block RAM, Ethernet MACs and RocketIO Transceivers are hardened logic for specific functions. For example, the XtremeDSP slice is an 18 bit x 18 bit multiplier followed by an accumulator. If a design requires the use of a multiplier, one of these will be instantiated. All these resources are spread out across the chip, easing the load on the interconnect that links all the resources.

The FPGA also includes two onboard PowerPC processors. Standard processor buses allow custom user code to connect with designs implemented in logic. This is extremely useful in rendering the node as a standalone device, as the physical layer can use the flexible logic while the MAC can both utilize processor functionality and be connected to the physical layer. Additionally, embedded operating systems, which can execute in the PowerPC, provide easy access to all the layers of the OSI network stack.

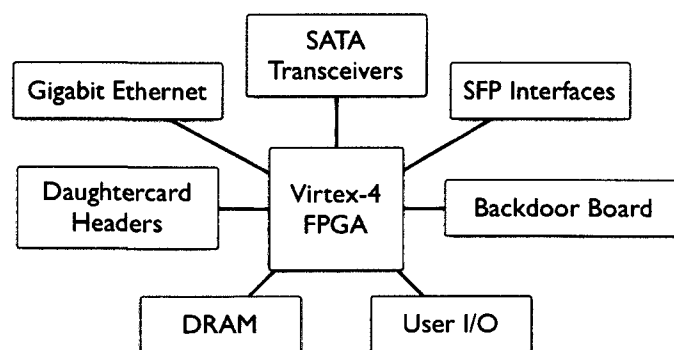


Figure 2.1: Block Diagram of the Virtex-4 FPGA Board

The hardware peripherals attached to the FPGA are shown in the block diagram in Figure 2.1. The FPGA provides the raw processing power to tackle real-time wireless

algorithms, while the peripherals are the conduits for the data to be processed. Each peripheral plays a specific role to create a standalone node.

### **2.1.2 Daughtercard Headers**

To support multiple standards, especially in the evolving portions of spectrum, RF interfaces are built as modules. A very general daughtercard interface is provided, which has an open specification to encourage the community to design custom boards to suit their needs. Four of these slots are present on the board. Currently, several daughtercards are available, like the Radio Boards and the Analog Board (digital-to-analog converter board). The interface itself is a 124 bit parallel bus connected directly to the FPGA general I/O. Here we take advantage of the flexibility of the FPGA as the I/O pins can be configured on a per design basis, we let the designer define the I/O pin functionality based on the daughtercard being used. This is key in maintaining modular architecture that enables future use and encourages new daughtercard designs. Additionally, the FPGA Board provides dedicated 5 V, -5 V and common ground to each slot. Up to 18 A can be drawn by the daughtercards in total, to use as desired. The significant high-speed bypass on the FPGA Board reduces noise on the power planes generated by the daughtercards.

### **2.1.3 Gigabit Ethernet**

In addition to possessing wireless capability each node needs connectivity to source and sink real data. The Board is built with the Marvell Alaska 88e1111 gigabit Ethernet transceiver. This assists in developing higher performance nodes that bridge wired and wireless domains. The Virtex-4 includes hardened Tri-mode Ethernet MACs that implement the functionality of Ethernet connectivity for a PowerPC processor in hard logic. They utilize considerably less logic than if implemented in fabric and reduce the load on user designs. We also support two additional gigabit Ethernet interfaces

which will be covered in Section 2.1.5 on *Multi-Gigabit Transceivers*.

### **2.1.4 Dynamic RAM**

As the two primary goals of the platform are deployment and experimentation, taking measurements and storing data are important. To sustain such data collection the FPGA Board has a DDR2 SO-DIMM slot that can support up to 2 GB of dynamic RAM. Other than data storage, operating systems, such as Linux, can utilize this RAM for program and data. Traditionally, designing efficient memory controllers has been a design challenge unto itself, but fortunately Xilinx tools provide the relevant controllers to make RAM usage seamless.

### **2.1.5 Multi-Gigabit Transceivers (MGTs)**

The MGTs are high-speed serial data links that support several standards of serial communication. Each transceiver is capable of up to 6.5 Gbps, encompassing links like SATA, PCI Express, optical fibre, etc. On this board we connect the MGTs to three interfaces: SATA, SFP and HSSDC2. To aid in non-volatile data storage such as hard drives, SATA target and host functionality is built onto the Board. Users can thus store several streams of data for offline analysis. Two additional gigabit Ethernet ports can connect via Small Form-factor Pluggable (SFP) jacks. By tripling the number of Ethernet connections, the system can support router-like functionality to test more complex wired and wireless networks. And finally, if some user designs are too big and require two or more FPGAs to split the processing between them, there needs to be a high-throughput link between them to transfer the large amount of data. Four HSSDC2 jacks provide this functionality by allowing a custom design to transfer information. The latency across this link can be reduced to a few cycles if the MGT clocks are shared. A flexible clocking scheme is present to fulfill the myriad demands of MGT usage, and that allows two boards to share their reference clocks.

### **2.1.6 Reconfiguration**

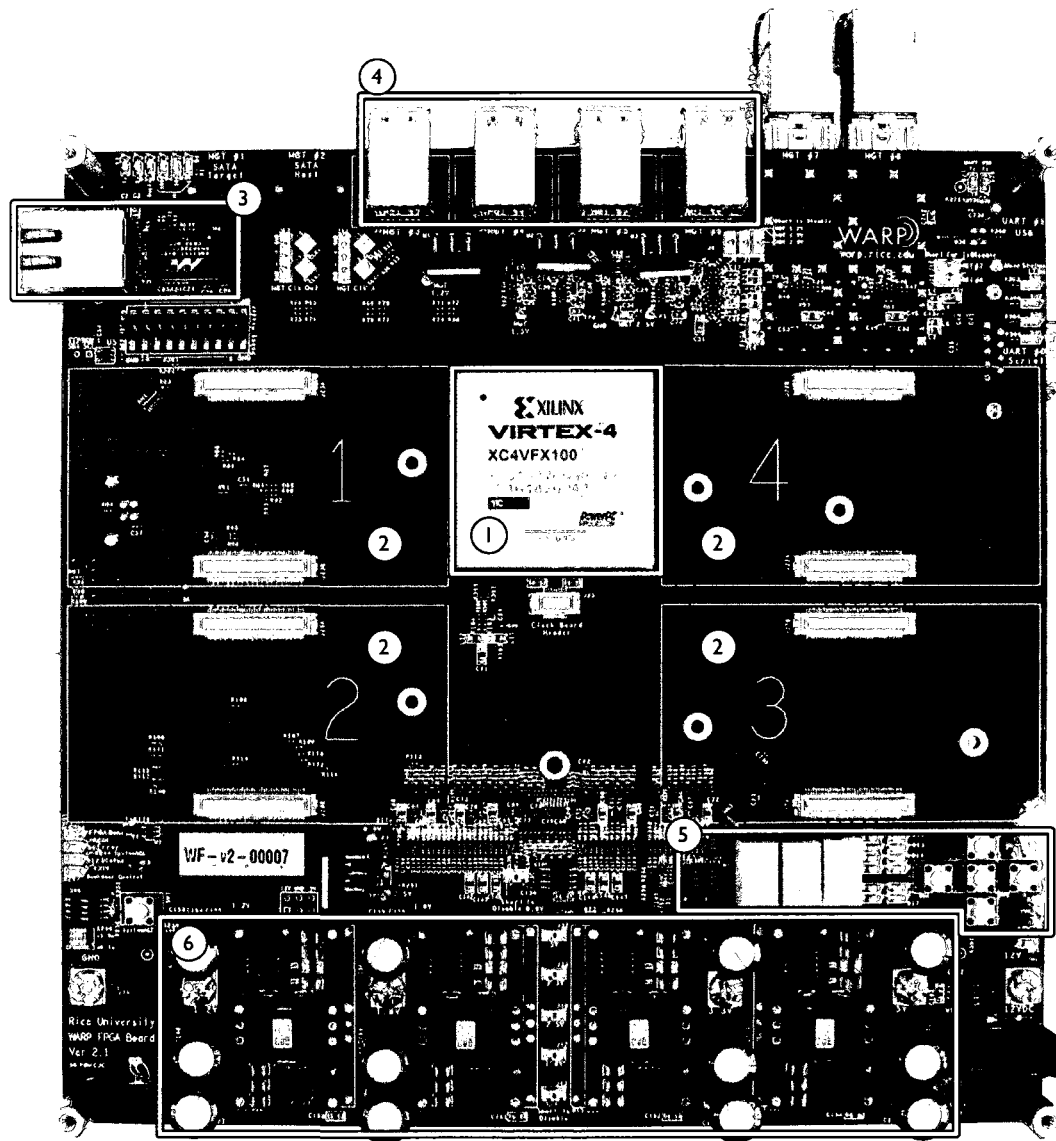
We designed the FPGA Board with three methods of reconfiguration for the Virtex-4. In a lab scenario where computers are easily available, the board can be reprogrammed using JTAG or USB cables. This allows for quick iterations as designs can be downloaded right away. However, if the board is deployed, up to 200 bitstreams can be saved on a CompactFlash card and downloaded using software calls. Lastly, the Backdoor Board, as we shall see in Section 2.2, can reconfigure the FPGA through the Slave Serial interface and control the CompactFlash card.

### **2.1.7 Other Peripherals**

While off-the-shelf hardware provides limited visibility into its inner workings, prototype hardware needs a low-level access to signals to enable debug of real-time designs. One of the most commonly used methods for debug purposes is using digital I/O to view signal interactions on an oscilloscope. 16 bits of digital I/O are provided for use. User LEDs, push buttons and serial ports can provide additional information as well.

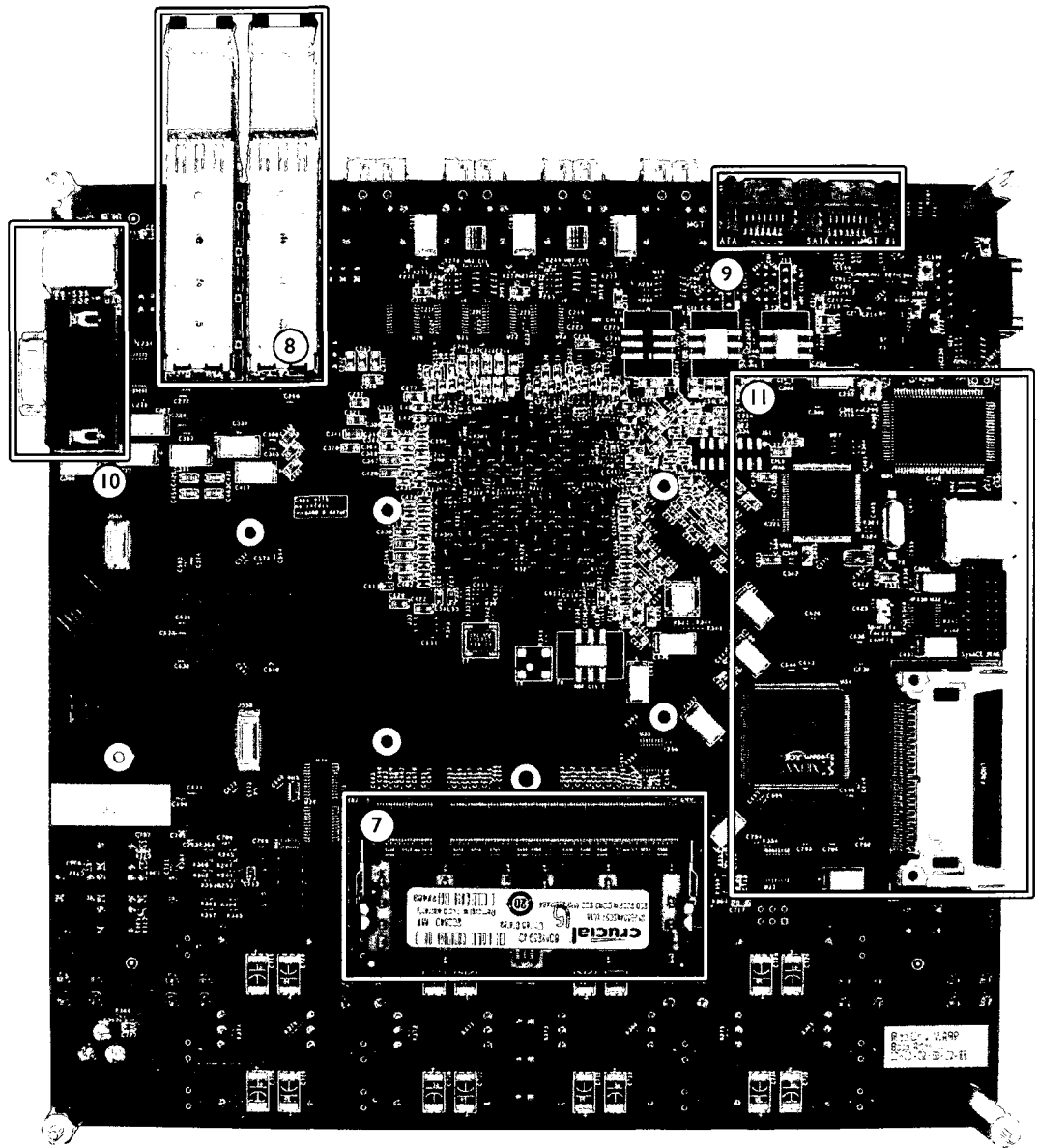
### **2.1.8 Power Considerations**

The board is powered by a single 12 V DC input supply. All other voltages are derived from this single source with the use of linear and switching regulators. There is a central controller that can inhibit all the supplies, effectively shutting down the board. This functionality is useful for performing remote hard resets of an entire node as the Backdoor Board (discussed in Section 2.2) has control of the inhibit control.



- |                              |                         |
|------------------------------|-------------------------|
| ① Xilinx Virtex-4 FX100 FPGA | ④ HSSDC2 MGT Connectors |
| ② Daughtercard Slots         | ⑤ User I/O              |
| ③ Gigabit Ethernet           | ⑥ Power Circuit         |

Figure 2.2: Front of FPGA Board



- |   |                     |   |                         |
|---|---------------------|---|-------------------------|
| ⑦ | DDR2 SDRAM          | ⑩ | User I/O                |
| ⑧ | SFP MGT Connectors  | ⑪ | Reconfiguration Circuit |
| ⑨ | SATA MGT Connectors |   |                         |

Figure 2.3: Back of FPGA Board



### 2.1.9 Hardware Design

The completed FPGA Board is shown in Figures 2.2 and 2.3. The different pieces of the system discussed above have been annotated on the figure.

The printed circuit board (PCB) design is 8 in x 8 in and composed of 18 copper layers – 8 power and ground and 10 signal layers. The stackup of the board is shown in Appendix B.1. Every signal layer has at least one ground layer adjacent to it. This improves signal integrity as the ground plane is available for signal return. As all the FPGA I/O pins are used, the fanout from the FPGA requires 7 signal layers. There are 10 different voltages on the board each with multiple drop locations. The power planes for all but 3.3 V are concentrated in the two central layers (layers 9 and 10) that are surrounded by ground planes as well. This protects the signals from having to use the split voltage planes as reference. The vicinity of the ground planes also provides capacitance augmented by the capacitors already present on the board.

The top and bottom layers are impedance matched to  $50\ \Omega$  for a 5 mil copper width providing excellent signal integrity for the high-speed signals on these layers. Therefore, all the MGT signals and clocks are on placed on the top and bottom. All I/O signals are designed with a maximum of two layer crossings (vias). As DDR2 memory controllers have a low delay tolerance, the signals carrying control and data for the DDR2 memory are within 0.027% (1 mil) of their total etch length.

The board is designed for a RoHS compliant manufacturing process. The internal dielectric (IT180) supports the higher temperatures and every part used for assembly is lead free. The complete schematics for the Board are available in Appendix A.1.

## 2.2 Backdoor Board

The second piece of the platform is the Backdoor Board. As mentioned, it serves two primary purposes: (1) a dedicated control channel and (2) the control and monitoring

hub. The control channel problem can be addressed in many ways. One could add an additional radio to the system and utilize the same band as the primary network [12] or use a dedicated control channel orthogonal to the experimental network itself [13, 14]. WARPnet supports both, as up to four radios can be connected to the FPGA Board. The Backdoor Board provides dedicated control channels that are also available to the FPGA. Next, having the ability to control every board in a network from a central server is important in scheduling experiments and observing network-level performance. Physical access to the boards may be difficult in deployed networks and hence remote reprogramming is an integral part of WARPnet. All the above functionality is provided by the Backdoor Board.

The block diagram of the Backdoor Board is shown in Figure 2.4. There are two main processors on the board each serving a different purpose. The first is a Linux System-on-Chip (SoC), the Axis ETRAX 100LX, and the second is a Xilinx Spartan-3AN FPGA.

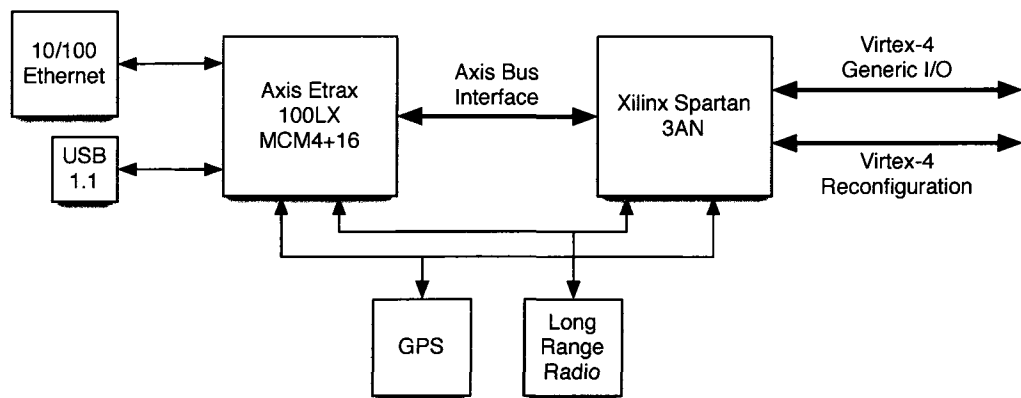


Figure 2.4: Block Diagram of Backdoor Board

### 2.2.1 Axis Etrax 100LX Multi-Chip Module

The design of the board is centered around an embedded Linux device server, the Axis Etrax 100LX MCM4+16 [15]. This is a multi-chip module with four major com-

ponents: an Axis Etrax 100LX processor, 4 MB flash memory, 16 MB RAM and an Ethernet transceiver (discussed in Subsection 2.2.3). The internal flash memory can store a boot image, which is executed at power up. This has two major benefits: it eliminates the need for users to reprogram the device every time, and eases deployment, since a stable image can be saved and used as many times as required without requiring physical access to the board.

The Etrax uses a custom, well supported, community driven, Linux distribution. It includes all the tools that are useful for debugging and development. It also reduces design cycle time as standard C/C++ programs can be used and no additional language needs to be learned. Additionally, the development environment is open source.

### 2.2.2 Xilinx Spartan-3AN FPGA

Connecting the Axis Etrax chip to the main Virtex-4 on the FPGA Board is a Xilinx Spartan-3AN (XC3S700AN) FPGA [16]. The resources available on the FPGA are listed in Table 2.2. Unlike the Virtex-series it has no embedded processor but a soft processor (e.g. MicroBlaze) can be implemented using the Xilinx design flow. The Spartan FPGA is set up to act as a translation device between the Axis Etrax processor and the Virtex-4. As both of those devices do not share common bus interfaces, some data translation is necessary for seamless transfer of data. The Spartan FPGA implements this functionality.

Logic Slices	5,888
Multipliers	20
Block RAM	360 Kb
User I/O	372

Table 2.2: Resources available on Spartan-3AN

Another key feature of the Spartan device, that is aimed at deployable networks, is the presence of non-volatile flash memory. Just like the Axis device, boot images

can be stored on the Spartan, and at power up can be ready for data transfer. From a network design perspective this provides additional reliability. If there is loss of power to the node, even though the Virtex-4 may lose its configuration, the two devices on the Backdoor Board that allow remote connections will return to a usable state.

### **2.2.3 Backdoor Interfaces**

We have implemented two functional and several potential backdoor communication interfaces. The type of deployment dictates the interface that is used in the network.

Consider the scenario where the deployment is within a building and wired Ethernet connections are available in the vicinity of each node. Here a wired backdoor link would be ideal. To fulfill this scenario a 10/100 Ethernet interface is provided giving direct access to the Etrax chip. As mentioned, the physical layer is part of the Etrax multi-chip module.

The second and more flexible implementation involves a long-range radio. This is tailored to outdoor deployments that spread over larger areas without wired connectivity at every node. The implementation uses a 900 MHz radio that can communicate over long distances with a highly reliable but low data rate link. The use of 900 MHz for the control channel eliminates all interference between the backdoor interface and the experimental wireless links.

Other potential backdoor interfaces can utilize the USB or Ethernet interfaces as supported by Linux. A Wi-Fi card connected to the USB 1.1 port could provide functionality similar to a laptop's wireless interface. The user must ensure that the channel usage is orthogonal to the experimental network. This solution is in between the long range and wired connection in terms of range and data rate. Any other external device connecting over USB or Ethernet can be utilized to provide the backdoor functionality.

## 2.2.4 Other Peripherals

The Backdoor Board has a GPS receiver attached. This is extremely useful for MAC layer algorithms as it can provide information such as topology and a precise globally-synchronized time. To perform scheduled access algorithms all the nodes in the system need a common time reference. The GPS time is accurate enough for this application [17]. This is also useful when the user is trying to conduct large-scale experiments where events need to occur at certain precise times.

One requirement that was mentioned previously was data collection. The board has a standard USB hub enabling users to connect flash drives to collect large amounts of data. A USB spectrum analyzer could augment the spectrum sensing abilities of the FPGA Board for certain applications.

## 2.2.5 Virtex-4 Link

The Backdoor Board is connected to the base Virtex-4 FPGA board using two low-profile headers. There are four types of signals passed up through them:

- Power Pins: The 12 V that powers the base FPGA Board is available for the Backdoor Board to use as its own source (discussed further in *Power Considerations*).
- General I/O Pins: A 22-pin bus is connected directly to the Virtex-4 is presented at the Spartan FPGA. As the signals are just general I/O they can be configured either as parallel data and address buses or individual signals.
- Reconfiguration Pins: One of the methods of reprogramming a Virtex-4 the Slave Serial interface. This is a four pin interface, two of them being clock and data. The Spartan has direct control of these pins allowing it reconfigure the Virtex-4 as needed.

Additionally, the Board has control over the CompactFlash reset interface. The user can reprogram the node from one of the many bitstreams stored on the CompactFlash card.

- **Inhibit:** Finally, the Board controls the Inhibit pin of the Virtex-4 power supplies. This means that the FPGA Board can be completely power-cycled allowing the user to hard reset. The Backdoor Board does not lose power during this process as both boards are powered by the 12 V that comes directly from the off-board source.

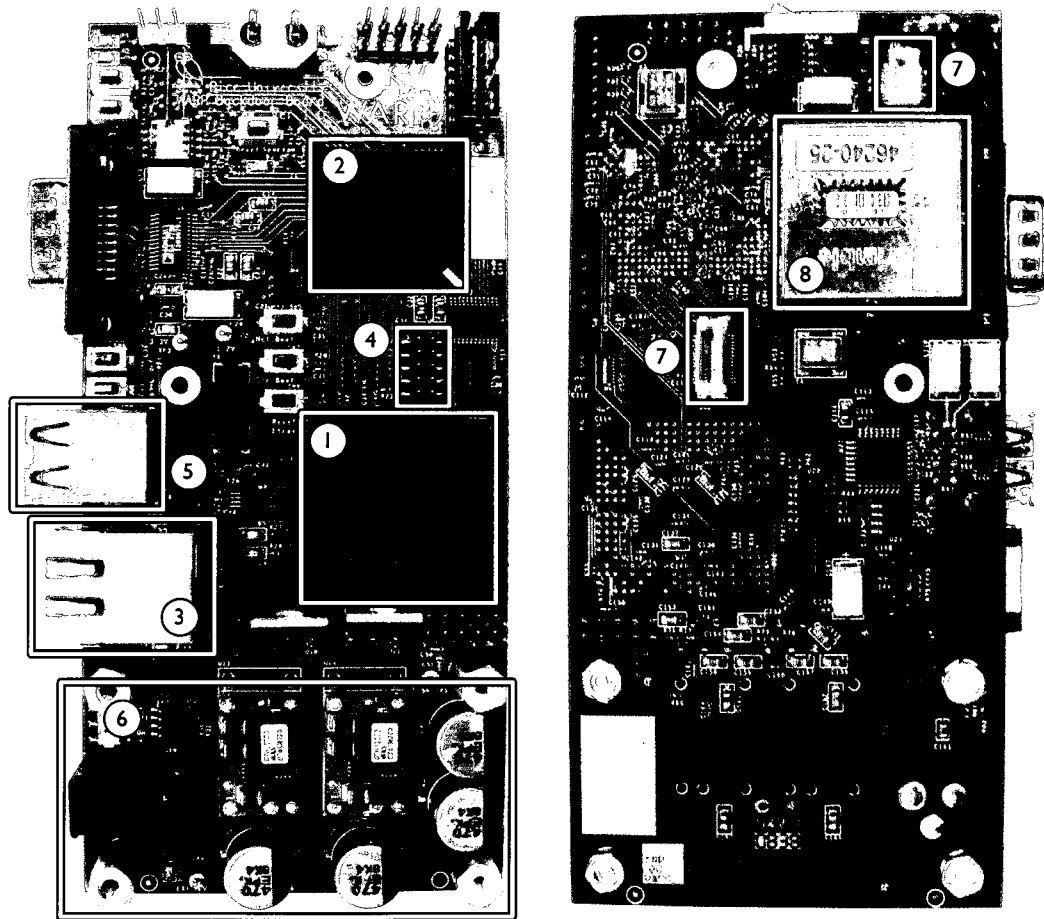
### **2.2.6 Power Considerations**

The Board is powered by 12 V DC input. As mentioned above, the Virtex-4 board makes this available on the auxiliary header. Additionally, in order to facilitate standalone operation, the board has a 12 V input jack (similar to the base board) that can be used instead. All other voltages (5 V, 3.3 V, 1.2 V) required to power devices are derived from this 12 V input.

### **2.2.7 Board Architecture**

The Axis Etrax chip acts as the heart of the design. Given its ease of use and C programmability, the user's primary concern is to utilize its resources. Its interface to the Virtex-4 has been abstracted as a bus transaction ("memory-mapped") through the Spartan device. Additionally, the Spartan has control over the Virtex-4 reprogramming and power cycling; it can monitor the Etrax bus for certain instructions enabling these features.

The 900 MHz radio serial interface is connected to both the Etrax as well as the Spartan chips. This enables two different modes; the Etrax uses the link as the control and measurement interface or the signals are passed up to the Virtex-4 that can use



- |                             |                                  |
|-----------------------------|----------------------------------|
| ① Axis Etrax 100LX MCM 4+16 | ⑤ USB 1.1                        |
| ② Spartan-3AN FPGA          | ⑥ Power Circuits                 |
| ③ 10/100 Ethernet           | ⑦ Virtex-4 FPGA Board Connectors |
| ④ 900 MHz Radio Connector   | ⑧ GPS                            |

Figure 2.5: Backdoor Board

it as an additional link to share routing information for example. Similarly, the GPS is also connected to both the devices.

The Etrax and Spartan have control of the reset and programming interfaces of each other. This allows both to perform a soft reset of either chip if necessary. As both chips have non-volatile storage, they always start up in a stable state.

The final key feature is the ability to act as a standalone device. To complete this network of deployed nodes, a central controller is required with the same interfaces as the Backdoor Boards. As the Backdoor Board itself is capable of acting in standalone mode it can be connected to a standard PC to perform all the processing for complex algorithms.

The Board itself is shown in Figure 2.5 with the various elements highlighted. The design of the PCB is composed of 8 layers – 4 signal layers and 4 power and ground layers. The stackup is shown in Appendix B.2. The board dimensions are 5.40 in x 2.58 in and it attaches to the bottom of the FPGA Board over the auxiliary header.



## System Implementation and Characterization

---

In the previous chapter we presented the hardware details of WARPnet. The FPGA Board, Backdoor Board and Radio Boards are assembled together to form a complete WARPnet kit. In this chapter we discuss possible implementation strategies, ways to partition a wireless communication algorithm and use cases for the hardware. In order for network designers to efficiently use the hardware, we characterize the following behavior of the 900 MHz backdoor link: round-trip time, throughput and maximum range. Finally, we present an example framework that decouples the control and observation engine and PHY and MAC along with the methodology to remotely reconfigure and monitor the Virtex-4 FPGA.

### 3.1 Implementation Strategies

In Chapter 1 we laid our goals to implement a novel wireless communications algorithm on a hardware platform. As we saw, the custom algorithm may modify the behavior from data type to the bit-level and each layer of abstraction must be modifiable. In the previous chapter we detailed the hardware architecture of the platform that we developed. The next step is to map the goals of custom wireless on to the specific hardware described and software frameworks that need to be developed.

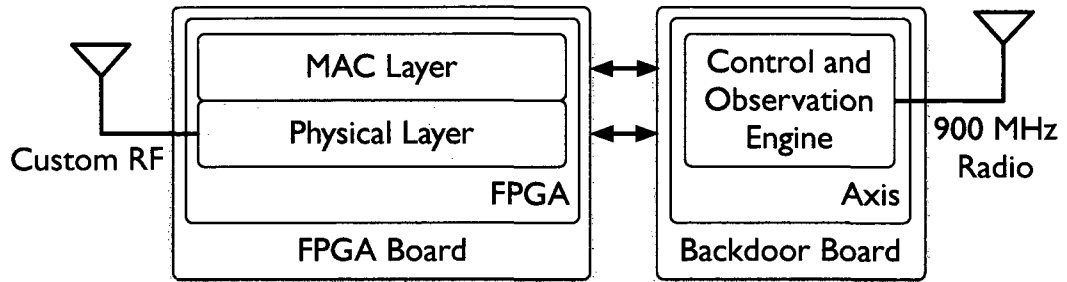


Figure 3.1: Implementation on Platform

The hardware itself provides several processors and interconnections. While this can be fairly complicated, each part of the wireless algorithm can be implemented on a particular processor, creating a decoupled system that provides all the functionality we need. Consider Figure 3.1 as an example implementation of the full stack where the corresponding pieces have been mapped onto hardware. As understood, there are three major pieces of the stack: the physical layer (PHY), the medium access layer (MAC) and the control and observation engine that must interact with both.

1. **Physical Layer:** The PHY is the lowest-level algorithm of a wireless network that deals with bits and waveforms. Off-the-shelf hardware utilize ASICs for the physical layer itself, but with our goal of being customizable and experimental the PHY can be implemented in the logic of the FPGA. As we saw in the previous chapter, the Virtex-4 FPGA is abundant in reprogrammable slices. Slices are parallel, high-speed and controllable at the bit level. In addition, the PHY will have access to other memory and I/O resources in real-time.
2. **Medium Access Layer:** One level above the PHY is the MAC layer that is responsible for scheduling and medium control. It must have low-level control of the physical layer to carry out its functionality. The MAC is implemented in the hardened PowerPC processors available on the Virtex-4 FPGA. There are two advantages to this: (1) the MAC has visibility into the workings of the PHY, and (2) it can access any other resources that enabling scheduling e.g.

---

timers.

- 3. Control and Observation Engine:** While just an FPGA Board would suffice for implementing current generation wireless protocols, the control and observation engine needs the functionality of the Backdoor Board. Using the custom interface between the FPGA Board and the Backdoor Board, the engine has visibility into the workings of the PHY and MAC. Parameters and statistics can be easily extracted to be passed onto other devices over the network of Backdoor Boards (backdoor network). An example framework is presented in Section 3.3 where control and observation engine and MAC/PHY are decoupled to avoid affecting the wireless data performance. The control engine talks to a central WARPnet controller or other WARPnet nodes over either the wired Ethernet connection or the long-range 900 MHz radio. As both of these are orthogonal to the 2.4 GHz and 5 GHz ISM bands, the control channel will not interfere with the data communication.

All of the ideas presented in Chapter 1 were to apply to network-wide experiments. This includes scheduling of node behavior and gathering statistics for performance evaluation. All three ideas of remote control, observation and reprogramming utilize the backdoor network. The example framework presented in Section 3.3 that decouples the MAC and PHY from the control and observation engine does the same for the experimentation functionality.

Keeping these possible implementations in mind, let us consider a few realistic use cases. These will also help in determining the metrics that need characterization for the evaluation of the platforms.

- 1. Cognitive Deployment:** The first scenario is for the platform to be deployed as a cognitive radio network. Wireless nodes, including the FPGA and Backdoor Boards, are spread out and connected in an indoor or outdoor environment. The

---

backdoor network is used to share cognitive information. If wired Ethernet is unavailable, all control packets utilize the 900 MHz radio interface. In such a scenario, the control data should be limited to short packets so as not to saturate the limited rate of the long-range radio. Latency of the data from source to destination will be important as update rate will determine how quickly a system can adapt to changing conditions. This scenario is tackled in more detail in Chapter 4.

2. **Data Gathering:** The second scenario is an algorithm evaluation mode where nodes are constantly reporting their performance to a central server. Just like the previous scenario, the nodes can be deployed in an indoor or outdoor environment. The data statistics are communicated over wired Ethernet, if available, else the 900 MHz radio is used. The scalability of such a solution will depend on the throughput that the 900 MHz radio can sustain in multi-source to single-destination setups.
3. **Centralized Controller:** The third use case is equivalent to system simulation. Algorithms implemented for the nodes, can be controlled from a central WARPnet controller. This amounts to ‘God-mode’ where network-wide information is available at all nodes. This is used to test the ideal performance of an algorithm with perfect knowledge of topology etc. This scenario will require constant data transfer from the central controller which will overwhelm the limited rate of the 900 MHz radio. Thus, wired Ethernet is preferred. Tethered mode is a second example of ‘God-mode’. In this scenario, the primary and secondary networks are all controlled by the central controller. The controller runs targeted experiments with specific network conditions to evaluate performance, similar to the functionality provided by WARPLab [18].

The wired Ethernet connection has been well characterized in practice. It can

sustain 30+ Mbps with very low latency (order of a millisecond). The interface that needs characterization is the long range 900 MHz radio. Its data carrying capability must be compared to wireless medium access time so algorithm designers can take its performance into account.

## 3.2 Backdoor Link Performance

An important aspect of the platform is data collection. The control channel to implement this would either be in the same band as the data network or orthogonal in frequency to it. The first option does not require the use of the Backdoor Board for communication but would intrude on the experimental spectrum. The second option is implemented with the 900 MHz radios as the data control channel. However, before utilizing the radios for sending control information, it is important to understand its limitations. In a rate-limited wireless link, the factors determining performance are latency and throughput. In addition, as the hardware is meant to be deployed, knowledge of the maximum range of communication is a prerequisite for building a large well-connected network.

The key specifications of the 900 MHz radio [19] are shown in Table 3.1.

	<b>Throughput Data Rate</b>	
	<b>9,600 bps</b>	<b>115,200 bps</b>
Maximum Transmit Power	1 W	1 W
Indoor Range	900 m	450 m
Outdoor line-of-sight Range w/ high-gain Antennas	40 miles	20 miles
Receiver Sensitivity	-110 dBm	-100 dBm

Table 3.1: Specifications of the 900 MHz radio

### 3.2.1 Latency

An important metric, especially for control information, is the latency from source to destination. Closed feedback control algorithms rely heavily on the latency of responses. For example, when deployed as a cognitive network, nodes will collaborate on parameters and update as necessary. However, the time for communication over the 900 MHz link would span several wireless data packets. Only if the time to update parameters on collaborating nodes is deterministic, coordination will be possible.

Therefore, the parameter of interest is the round-trip time from source to destination and back over the 900 MHz radio. This must not only include the channel usage time but also the time to update a parameter on the Virtex-4 FPGA Board. This is exactly the scenario in use case 1, where nodes must update each other on system parameters.

Our methodology to achieve this is as follows. The source node initiates a transmission with the current time as the data that is sent. The partner node captures the traffic, updates a parameter and replies with the original timestamp. The source can calculate the time for the round-trip based on its current time and the time the transmission was initiated. The clock used on the source is accurate to 10 ns, giving it high precision in calculating the round-trip time.

Let us consider variable distance as the first measure. Figure 3.2 shows the CDF of the arrivals with measured round-trip times for different distances. A round-trip time of 30 ms meant that just the source to destination time is 15 ms. The data from Figure 3.2 shows that 99+% of the arriving packets return in 52 ms while 98% return in 42 ms. Also, notice that the number of arrivals does not vary significantly over the various distances, so any algorithm utilizing the dedicated control channel does not need to modify its parameters based on the distance of the destination node.

While a point-to-point link is common in small node networks, scaling to anything larger would require the control channel to be point-to-multipoint. Figures 3.3

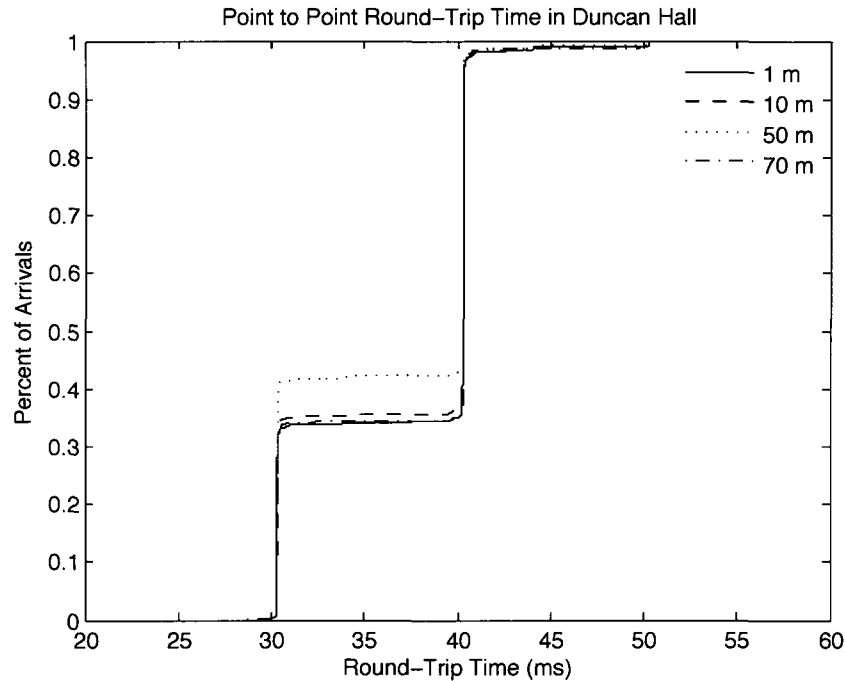


Figure 3.2: Round-Trip Time for Two Nodes with Varying Distances

and 3.4 show the round-trip time for a three node setup. First is the latency seen for two sources to a single destination while the second is for a single source and two destinations. The graphs show the CDF for both the links. Again, the variance between the two systems is minimal, further supporting the independence of the algorithm from the size of the network with respect to latency.

This outcome is expected, as the latency test does not stress the limits of medium usage. The real performance bottleneck will be seen when the number of nodes is increased and simultaneous throughput is measured.

While the absolute numbers provide an interesting look at the delay, we can gain perspective by comparing it to wireless data packet times. For the WARP OFDM Reference Design [20], a wireless OFDM data packet 1500 bytes long using QPSK modulation for its 48 data subcarriers requires 1.044 ms of medium usage time. In addition to that is the reply from the receiver node which returns in  $17 \mu\text{s}$  followed by  $64 \mu\text{s}$  of medium usage for the acknowledgement. On average the backoff

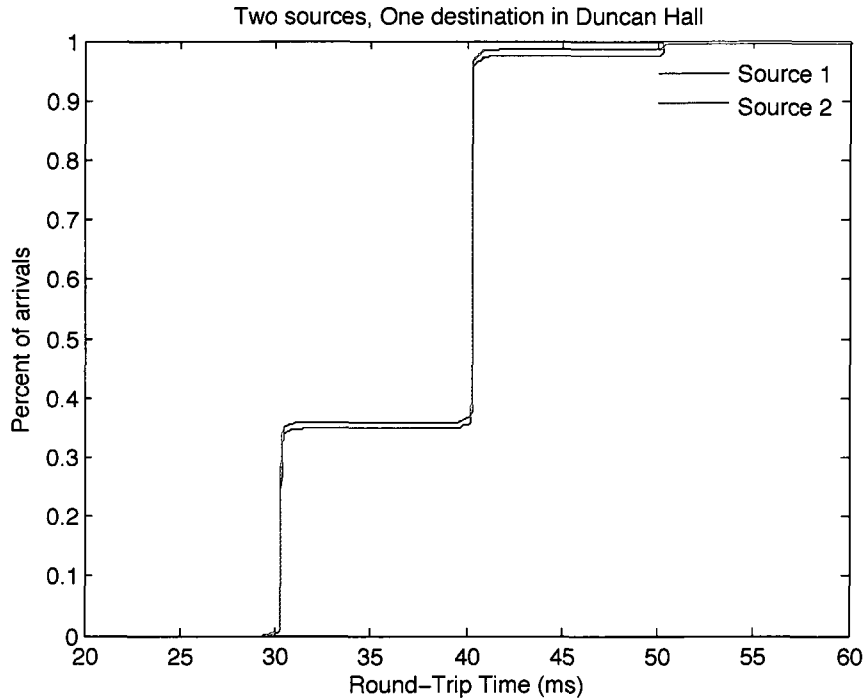


Figure 3.3: Round-Trip Time for Two Transmitters Feeding One Destination

window would utilize 8 time slots at  $20 \mu\text{s}$  each, adding a further delay of  $160 \mu\text{s}$ . Thus, each wireless data packet requires 1.285 ms between subsequent transmissions. On the other hand, the round-trip time for a 900 MHz acknowledgement is 42 ms. Approximately 33 wireless data transmissions can occur between subsequent control channel parameter updates.

The deterministic parameter update rate of 42 ms or 33 packet transmissions allows algorithm designers to tailor their feedback loop algorithms to perform at their peak.

### 3.2.2 Throughput

The next metric of importance is the throughput that can be sustained by the 900 MHz radio link. We showed use cases where the remote deployed nodes transfer information back to the central controller. The quantity and rate of that information will be determined by the Backdoor interface used to communicate that information.



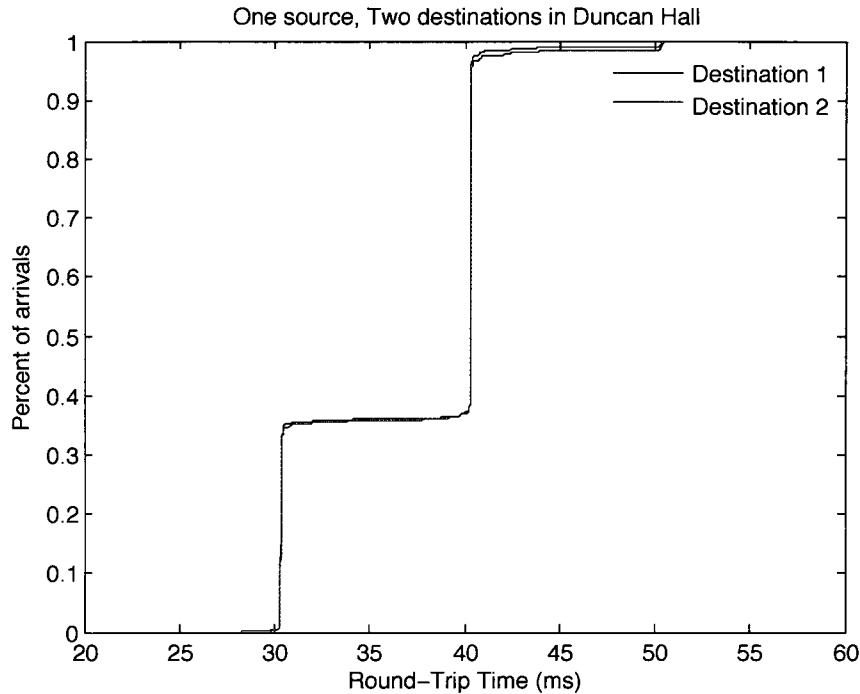


Figure 3.4: Round-Trip Time for One Transmitter Feeding Two Destinations

The 10/100 wired Ethernet on the Backdoor Board can support upwards of 30 Mbps. The WARP OFDM Reference Design can support up to 15 Mbps [20] while the maximum achievable physical layer throughput for 802.11a is  $\sim 54$  Mbps [21]. One in every two received data packet can trigger an update back to the central controller and this would be sustainable. For the long-range radio that is not the case. This radio is built as an alternative serial port, hence the throughput is limited to serial data rates of 115,200 bps. However, actual data throughput will be lower due to MAC overhead and retransmission latency. We can then compare this to the quantity of information that can be sent from source to destination.

Our methodology of characterizing this metric was to generate a constant rate of traffic from the source to the destination and keep track of the number of packets received. We used the wired Ethernet to trigger the beginning and end of the data gathering cycle. Each packet sent was 512 bits including control information.

Figure 3.5 shows the throughput seen across 5 distances in Duncan Hall. For all

the distances, the throughput is maintained at around 25 Kbps. Just as we noticed while characterizing latency, the performance variance is very low across the building. However, as noted in Table 3.1, the specified maximum range is 450 m while our measurements could only span 100 m due to the size of the building. Hence, any significant drop in throughput could not be empirically observed.

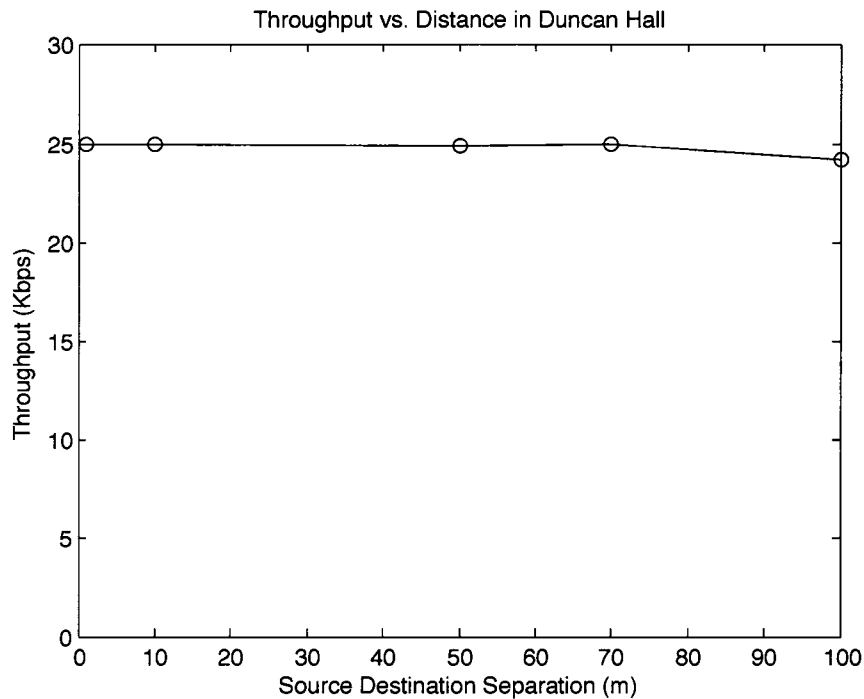


Figure 3.5: Throughput for Various Distances

Another metric within throughput that is important, especially in the data gathering scenario, is multiple nodes utilizing the same medium. We would expect a significant throughput loss as the total nodes increase. Figure 3.6 shows the average throughput for 1, 2 and 3 nodes transmitting to the destination. The graph with the squares is the ideal scheduled scenario throughput where the medium is perfectly partitioned amongst the participants. The graph with circles is the throughput in the implemented random access system. The dashed line shows an upper bound on potential future throughput. The assumption is that with a further increase in the number of nodes, the throughput loss remains constant. Thus the loss in throughput

with three transmitting nodes does not further increase. A factor to consider is that this test was conducted with the nodes always backlogged with data. If the nodes scaled down transmitted information based on the number of operating links, the scenario could be managed better.

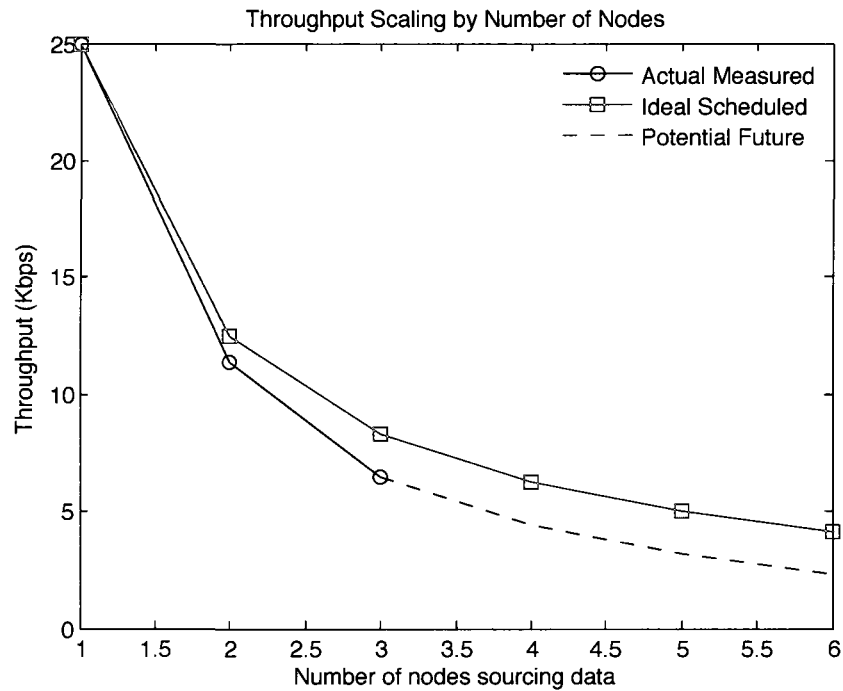


Figure 3.6: Throughput Scaling as Number of Transmitters Increases

Let us consider an example to see if 25 Kbps is sufficient for statistics updates in real-time. Each node keeps track of four counters: good and bad received packets and total received and transmitted bytes. Each counter is 32 bits long. Thus each update would require 128 bits. Adding an additional 32 bits for control information leads to a total of 160 bits to be transmitted.

Transmitting Nodes	Updates per second
1	160
2	72
3	41

Table 3.2: Maximum Number of 160 bit Updates per Second

From Figure 3.6 we know the maximum throughput that is sustainable by every

---

node. Using this information and the number of bits transmitted in every update, we can calculate the number of updates that the node is capable of sending to the central server.

Table 3.2 summarizes the results. We can see that for a three node setup, each node can send 41 updates per second. This detailed a look into the performance may not be needed in real-time.

### 3.2.3 Maximum Range

The hardware is built to be deployed in an indoor or outdoor environment while being controlled over the backdoor links. While Ethernet may not be available everywhere, the 900 MHz radio can be utilized to control the network. However, as expected, the long-range radio will have a finite range. For network designers, this is the key number in determining node placement. Exceeding this range can render the backdoor link unreliable, defeating its purpose. As indoor deployment is the primary goal, indoor range should be characterized.

There are several factors that contribute to the range of the 900 MHz links. The radio has two parameters that determine range: output power and wireless data rate. The output power can be controlled from 1 mW to 1 W while the RF data rate can either be 9,600 bps or 115,200 bps. As we would like to sustain as high a data rate as possible, we select the faster data rate and 1 W output power to perform our benchmark tests. Additionally, the module has an extremely sensitive radio that can receive data up to -110 dBm at 9,600 bps or -100 dBm at 115,200 bps as shown in Table 3.1. Thus, at 115,200 bps the maximum path loss that can be tolerated is 130 dB.

Our benchmarks are based on an empirical method. Using Duncan Hall as a representative indoor environment, measurements of path loss were taken across the building. Using linear regression, the path loss exponent is computed and potential

maximum range is determined. While the maximum distance is just a prediction, it matches closely with the expected distance based on the module's datasheet.

The measurement of the received power is the key to the empirical benchmark. Each module returns the received signal strength indicator (RSSI) value for every received packet. While the instantaneous RSSI may vary, its distribution should be log-normal [22]. The mean RSSI at each transmitter-receiver distance is used for the regression analysis. In order to provide a varied environment for analysis, the data was collected across the building, as shown in Figure 3.7, for several minutes. The coverage was limited to a single floor, but Duncan Hall's architecture allows for open access between the floors reducing the effect of floor crossings.

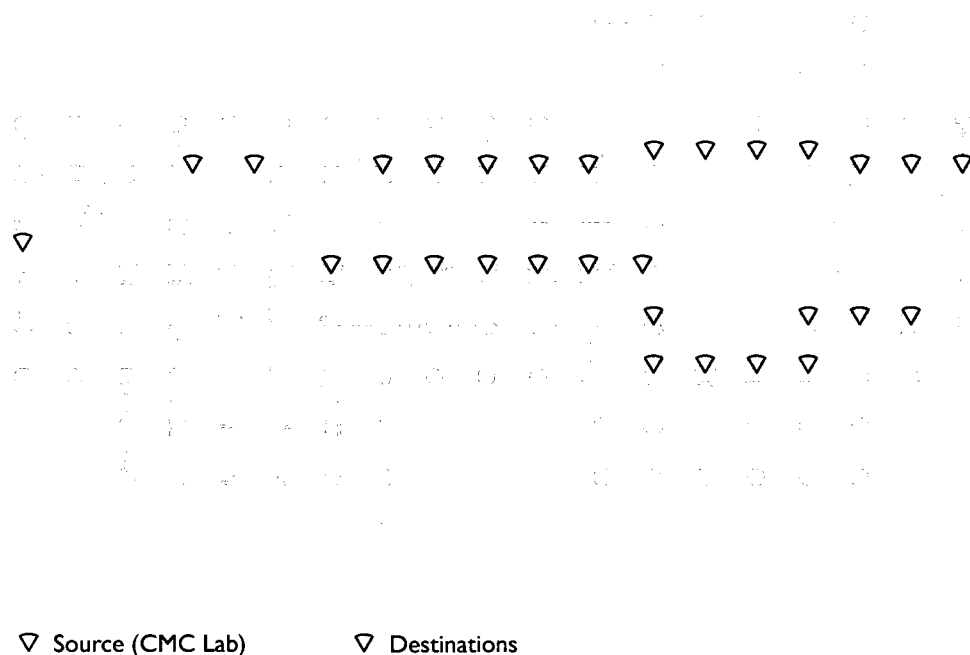


Figure 3.7: Data Collection Locations for Maximum Range Experiments

Taking into account the log-normal shadowing effects the path loss equation is given by

$$PL(d) = PL(d_0) + 10n \log \left( \frac{d}{d_0} \right) + X_\sigma \quad (3.1)$$

where  $d_0$  is the reference distance where the path loss is known,  $n$  is the path loss exponent and  $X_\sigma$  is the zero mean random variable representing the shadowing. By considering just the mean of all measurements at the same location, the random variable can be dropped. Thus,

$$PL(d) = PL(d_0) + 10n \log \left( \frac{d}{d_0} \right) \quad (3.2)$$

represents is the path loss model of interest.

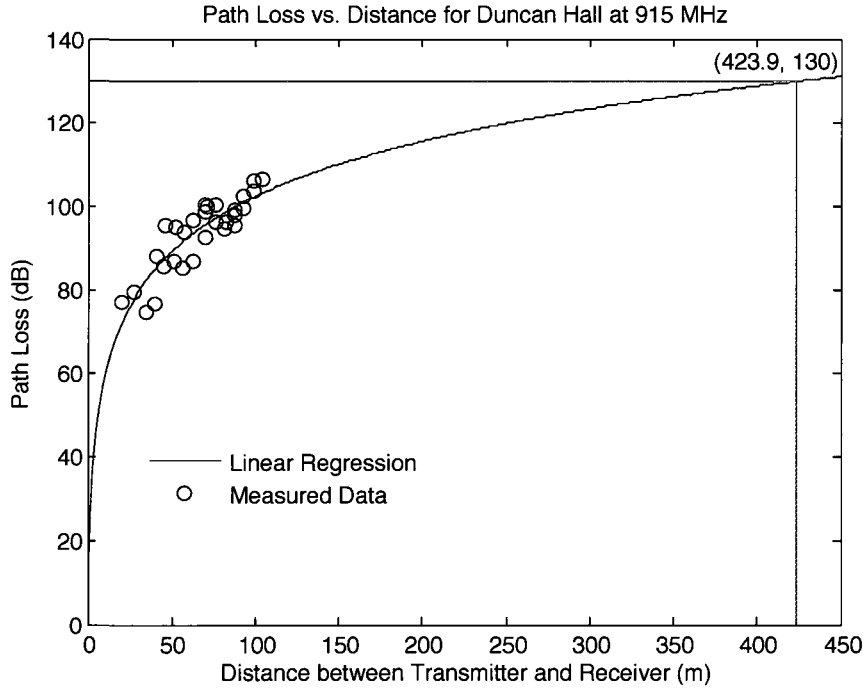


Figure 3.8: Path Loss Measurements with Linear Regression

The measured RSSI data is plotted in Figure 3.8 as the black scatter points. The distance range is limited to  $\sim 100$  m, as that is the length of the building. Also, the minimum path loss is around 70 dB as the radio modules can only report received power less than -40 dBm. Performing linear regression on the data gives the following equation,

$$PL(d) = 14.86 + 19.02 \cdot d \quad (3.3)$$

yielding a path loss coefficient of 1.90. The path loss coefficient for free space is 2; a higher number indicates obstructions and attenuation while a lower number indicates constructive interference. Using empirical data it has been shown [22] that ‘In building line of sight’ coefficients range between 1.6 and 1.8 and coefficients for retail stores range between 1.81 and 2.16 [23]. Duncan Hall’s architecture and open top floor do provide a warehouse-type indoor environment, indicating that the empirical coefficient is in the correct range.

Plotting Equation 3.3 on the same graph yields the potential maximum distance for indoor environments. At a path loss of 130 dB the distance is 424 m, extremely close to the datasheet mentioned 450 m, again validating our analysis. This maximum range would be the diameter of a fully connected network of nodes, where data communication between every module is possible.

### 3.3 Frameworks

The control and observation engine depends on having access to both the PHY and MAC layers running in the FPGA. Therefore we need to define a framework for data access from the FPGA to the central server. We present one possible implementation of decoupling the transfer between the Virtex-4 FPGA and the Axis processor.

#### 3.3.1 Data Collection Architecture

In the second use case, the node will be reporting statistics back to a central server. However, in the process of data collection, the performance of the wireless algorithm itself must not be affected. The data reporting must not steal any processing resources from the MAC layer algorithm. This would especially be an issue when using the 900 MHz radio as data transfer to the module is slow. In order to prevent MAC delays, the FPGA design should be decoupled from the process of transmitting the

data over to the server. The design in the Axis Etrax should be responsible for sending the data. Figure 3.9 shows an example architecture that achieves this decoupling.

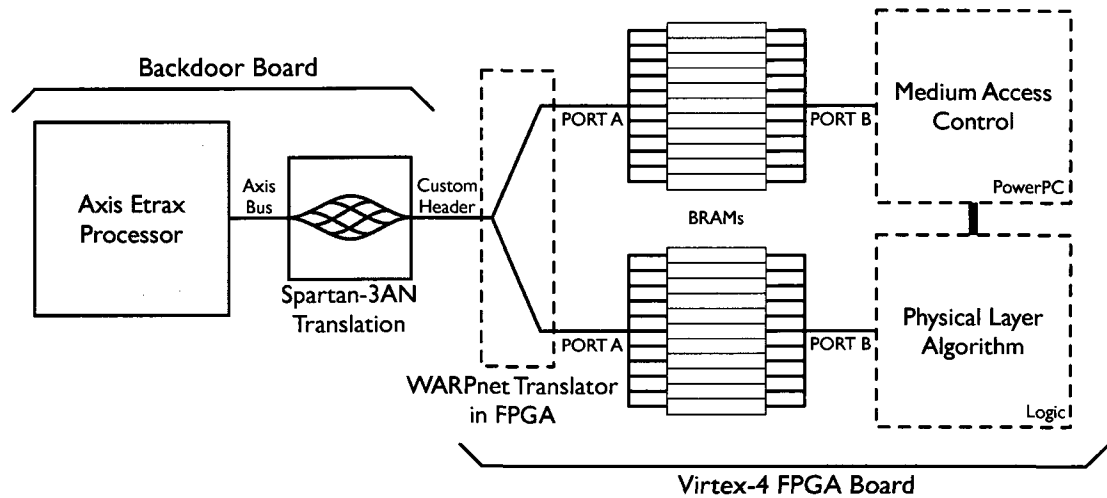


Figure 3.9: Data Flow Decoupling

The key to decoupling the architecture is to provide the Axis program direct access to the memory where the FPGA designs stores its statistics. For the architecture shown in Figure 3.9 the FPGA stores its data into a local BRAM. As all BRAM blocks are equipped with two ports, the Axis program gains access to the same memory over the second port. There is one major constraint to consider; the header between the Backdoor Board and FPGA Board is limited to 22 bits. Thus the 32 bit bus on the Axis must be converted to a double rate 16 bit bus. On the FPGA Board side the 16 bit data must be converted to 64 bits for BRAM port access.

The Spartan FPGA does the majority of the work to interface the two buses. It responds to both reads and writes, enabling bidirectional transfer of data. However, this transfer must always be initiated by the Axis design. The resource utilization for the Spartan design is shown in Table 3.3.

The FPGA-side design has to convert the custom bus from over the header to BRAM compatible transactions. The design must be lean so as to not steal any resources that the PHY might need. It utilization is shown in Table 3.4.



	Used	Available
<b>Slices</b>	518	5,888
<b>Block RAM</b>	11	20
<b>Slice Flip Flops</b>	591	11,776

Table 3.3: Resource Utilization for Spartan Design

	Used	Available
<b>Slices</b>	487	42,176
<b>Block RAM</b>	3	376
<b>Slice Flip Flops</b>	408	84,352

Table 3.4: Resource Utilization for FPGA Design

The natural decoupling of data across the BRAM interface is ideal. The MAC never realizes the presence of the Backdoor Board or that the data is being used elsewhere. The Etrax interactions are invisible to the MAC or PHY. This structure can also be used to update MAC and PHY parameters. The WARPnet translator in the FPGA provides the ability to read from up to 4 separate BRAMs. Thus, the MAC and PHY can store their statistics in two different BRAMs and the Axis can periodically gather new information at the rate it wishes to update the central controller.

### 3.3.2 Reconfiguration

The next important ability to enable remote control and measurement is the reconfiguration of the FPGA with updated designs. This will be done over the Backdoor Board network. The rate at which the boards are reconfigured will depend on the data transfer interface.

As mentioned in Section 2.2, the Backdoor Board is connected to the Slave Serial interface of the Virtex-4 FPGA. The Slave Serial interface is one of many ways to reprogram the FPGA. It is a two-wire serial interface with clock and data. The clock speed for configuration can be determined by the user application. There are three major steps to configuring the FPGA: (1) transfer the bitstream from the WARPnet

server to the Backdoor Board, (2) determine the integrity of the bitstream and (3) download the bitstream to the FPGA.

In order to perform any integrity checks on the transferred data, the bitstream needs to be buffered at the Backdoor Board. The Axis device has up to 16 MB of storage memory where the 4048 bytes bitstream can be stored. Once a cyclic redundancy check has been performed, the file can be streamed to the Virtex-4 FPGA. The Slave Serial pins are controlled by the Spartan-3AN, hence a simple state machine in the Spartan-3AN can implement the bit transfer functionality.

### **3.3.3 Watchdog**

As the Backdoor network is a monitor for the Virtex-4 FPGA, it is important to have the ability to know the state of the FPGA. A simple watchdog implementation can be used to assist in this process. As the link between the Backdoor Board and the Virtex-4 FPGA is custom, we can dedicate one pin for message passing. The FPGA can indicate that it is operational at regular intervals. In case of an error, the Backdoor Board can inform the central server and reset the FPGA restoring the design from a known image.

## Cognitive Radio

---

In this chapter we present the basics of cognitive radio, an emerging wireless communication technology. We elaborate on the functionality required for a cognitive radio implementation and the resources that WARPnet provides to facilitate such an implementation. In addition, we present an example cognitive radio application implemented in two different modes to show the flexibility of the platform. In Chapter 3 we showed the capabilities of the 900 MHz radio. Taking these limits into account we implement a partially distributed cognitive algorithm for channel selection of a secondary network. The same algorithm is also implemented in a centralized genie-aided mode where network information is known everywhere. These two examples intend to show the flexibility of the platform and a methodology for performance benchmarking. This application is not intended as a novel cognitive algorithm but as an example of using the functionality that is provided by the hardware and software of WARPnet.

Cognitive radio is based on current wireless algorithms where the physical and MAC layers play the paramount role of data transfer itself. It supplements this technology with the idea of ‘cognition’ that can assist the physical and MAC layers to perform at peak rate by making intelligent real-time decisions. Cognitive radio

devices must opportunistically adapt to the existing users of the spectrum. The three primary mechanisms that form the crux of cognitive radio are sense, adapt and share.

- *Sense:* The idea of sensing is not unique to cognitive radio; current generation technologies like CSMA sense the medium prior to transmission to protect against packet collisions. cognitive radio extends the idea from instantaneous medium knowledge to long-term information aggregation. For example, the medium may be sensed over several packet cycles to calculate average busy air-time, over several minutes to understand user behavior, etc. The local information per node is specific to each node due to location dependence of wireless propagation. As a result, each node has a different view of the network, which in turn necessitates coordination of measured data from several nodes to make network-wide decisions.
- *Adapt:* Once sensed data is available, a node adapts its system settings and behavior to increase its performance. Just like sensing, adaptation is commonly employed in current wireless systems. Cognitive radio extends the idea as the data is distributed and available of fast time-scales. For example, if a node sees that a particular spectral band is especially occupied over time, it can relocate itself. Or if it notices that other users have periodic bursts of traffic it can request less medium usage in that period. As the decision taken by the nodes is based on the sensed data, only local environmental changes will cause a reaction. While this may suit that particular node, it may degrade the performance of the network as whole. Thus, the adaptation has to be performed jointly to ensure each node can get maximal performance.
- *Share:* A major limitation of traditional wireless algorithms, that cognitive radio tackles, is the response to network-wide performance degradation. Thus, a node must not only gather information about the local medium and take a

decision on the best method to improve performance but also share this information with other nodes. This spread of information helps in iteratively selecting the ideal system parameters and keeping all nodes abreast of network variations. Coordination of the nodes helps in maximizing the performance across the network. For example, two nodes transmitting to each other can decide on a channel that is mutually agreeable rather than ideal for one particular node.

## 4.1 Implementation

As discussed in Chapter 3 a wireless implementation would use the FPGA for the physical and MAC layers. However, a cognitive radio algorithm requires another layer of intelligence to parse information over the cognitive network as well as gather information from the physical and MAC layers. Starting with the block diagram from Chapter 3 we can add in the functionality for the cognitive algorithm. As can be seen from Figure 4.1 the Backdoor Board can be re-used to not only provide the control and observation engine hooks but also implement the cognitive algorithm.

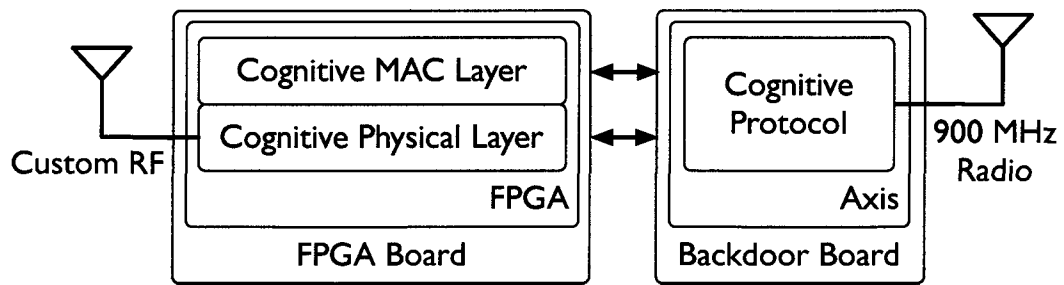


Figure 4.1: Protocol Implementation on Platform

Hence, we have extended the capabilities of the original platform by utilizing the hardware in a novel implementation.

---

## 4.2 Primary and Secondary Networks

The system implemented considers the often cited example of a primary and a secondary network. The primary network is a legacy system whose behavior is well-defined but has no ability to adapt to other wireless links. Consider the primary network as the owner of the spectrum and hence not aware of the presence of the secondary nodes in the same spatial location. The secondary network that is deployed must actively avoid the transmissions of the primary while trying to sustain the maximum possible throughput. While the algorithm implemented is not perfect, it is meant to exercise all the features of the platform and give an example of implementing distributed and centralized algorithms for benchmarking.

The primary network is a node coordinated frequency hopping system. The network utilizes the 2.4 GHz ISM band covered by the WARP Radio Boards. Every one second, the transmission hops to a new channel. This channel is randomly selected but known between the two primary nodes in order to preserve the link.

An example hopping order is shown in Figure 4.2. The primary switching order is channels 2, 4, 6 and then 8. The experiment does not mandate the need for a linear switching order; it has been done to visualize the behavior of the primary and secondary nodes. Each channel is 10 MHz wide, and since the transmissions are 10 MHz in bandwidth all the selectable channels are frequency orthogonal.

The secondary network is limited to the same four channels to which the primary has access. Every time-slot it must pick a channel that allows it to continue transmitting without interference and interruption. While ordinarily, the secondary network could adapt parameters such as bandwidth and subcarriers used, transmission power, etc., in this particular example just the channel is being actively adjusted.

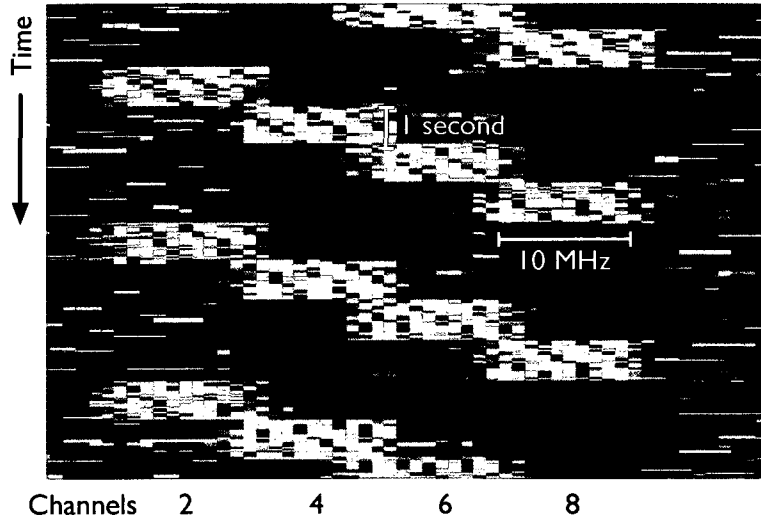


Figure 4.2: Channel Access with Primary Only

### 4.3 Distributed Cognitive Radio Example

As discussed in Chapter 1, every cognitive algorithm is divided up into three actions: sensing, adapting and sharing. We implemented the cognitive channel selection secondary network using the same principles.

The system actions are divided on a time-slot basis. The primary link's actions are controlled by the central controller at one second intervals. At the same time, the central server informs the secondary network of the beginning of a time-slot. The secondary nodes set aside 100 ms for sensing before continuing to transmit their own data. During this period of sensing, no secondary transmissions are permitted. Once the new channel has been selected and the information shared, the link returns to normal. The sensing time can be further divided into three major tasks. Figure 4.3 shows how the 100 ms is divided among these tasks.

- *Sensing*: In the implementation, a designated secondary node senses the medium for traffic. As there are four possible frequency bands that can be used by the primary network, the node scans all four channels in each sensing slot. The most commonly used metric for channel usage is received signal strength (RSSI). In-

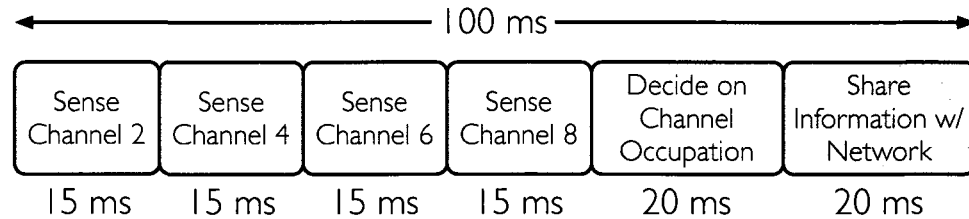


Figure 4.3: Division of Sensing Time

stead of considering instantaneous RSSI, where other traffic or noise can cause spurious readings, the sensing algorithm averages the RSSI for 7 ms. Within 15 ms two readings can be gathered, eliminating any effects of the channel switch in the first reading. The sensing utilizes a second radio interface, but can just as easily be multiplexed with the primary radio. The Virtex-4 FPGA, with direct access to the RSSI, is used to run the data capture algorithm.

- *Adapting:* Once the averaged RSSI data is known in the FPGA, the algorithm running on the Axis device reads in the data using the framework described in Section 3.3. A total of 60 ms is allocated to sensing the four channels before a decision is made (4 readings with 15 ms for each). The secondary link assumes that the primary network is the only traffic on the channel; thus to reduce interference it selects the channel that is furthest from the primary. When the primary is transmitting on channels 2 or 4, channel 8 will be selected. If the primary is on channel 6 or 8, the secondary selects channel 2. The location of the primary is determined by the averaged RSSI from the sensing stage.
- *Sharing:* The selected channel is communicated to the other node of the secondary network using the 900 MHz long range radio. As we showed in Section 3.2, 20 ms is needed for one way transmission of a packet. Thus the last 20 ms of the 100 ms is allocated to sharing the information.

Figure 4.4 shows a snapshot of the primary and secondary networks utilizing the channel. As seen, the secondary link picks a channel that is not occupied by the



primary. However, it does not always pick the channel furthest away. This is usually due to RSSI measurements. Further performance analysis and algorithm extensions are discussed in Section 4.5.

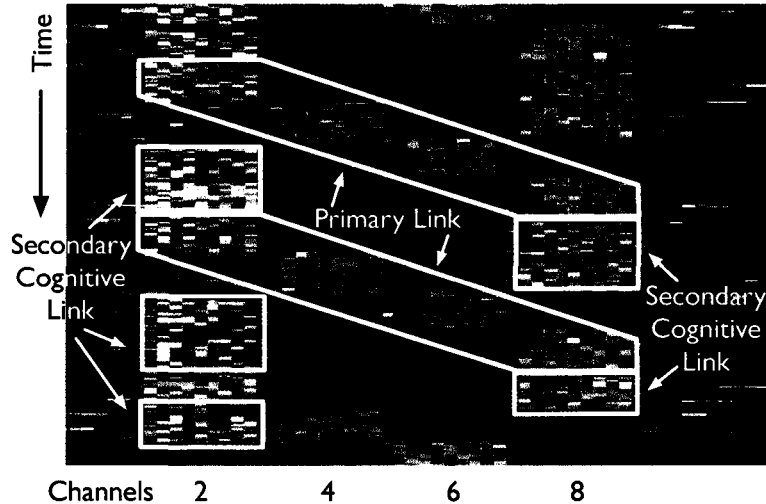


Figure 4.4: Channel Access with Primary and Distributed Secondary Links

## 4.4 Centralized Example

The distributed algorithm is the future of cognitive applications. However, it may have performance bottlenecks that have not been observed empirically. Benchmarking it against a perfect information centralized system will help in exposing these limitations.

We implemented a ‘God-mode’ application where the central server directs all the nodes. The server already controls the primary network by selecting its next channel; therefore it is always aware of the primary’s location. In addition to signaling the start of a time-slot, the server also communicates the primary channel to the secondary nodes. Despite the knowledge of the primary being available, the secondary network pauses for 100 ms to make the comparison with the distributed algorithm fair.

Figure 4.5 presents a snapshot of the channel access in centralized server scenario.

The secondary network always chooses the correct channel without overlap with the primary.

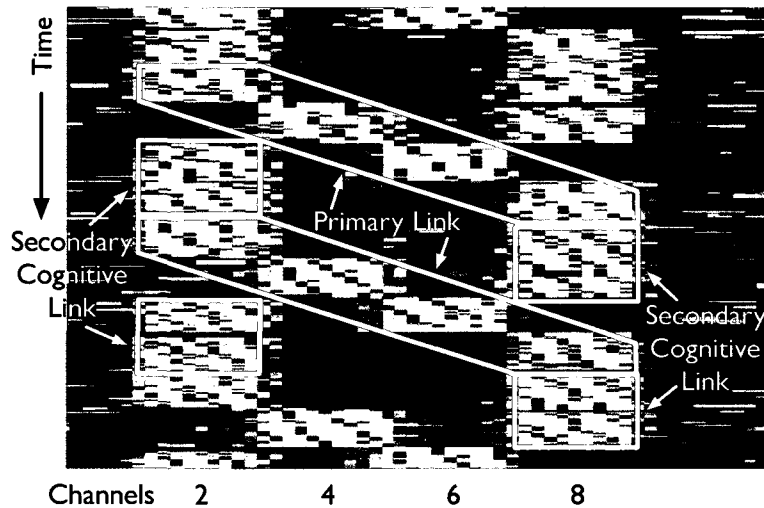


Figure 4.5: Channel Access with Primary and Centralized Secondary Links

## 4.5 Performance

An important metric for a wireless network is throughput. As expected, the secondary network is trying to maximize the data pushed through the network and minimize the disruption of the primary link. Table 4.1 summarizes the throughput achieved in the centralized and distributed algorithms implemented.

	Primary (Mbps)	Secondary (Mbps)
Primary Only	5.50	-
Secondary Only	-	4.25
Distributed	5.15	4.19
Centralized	5.40	4.20

Table 4.1: Throughput for Cognitive Algorithm

The benchmark for performance of the primary and secondary links is the when each is the sole user of the channel. The primary network can sustain 5.5 Mbps while the secondary link can sustain 4.25 Mbps. Both links are using the same

physical layer and the difference in throughput is because the secondary sets aside 100 ms for sensing. For the distributed implementation, the primary suffers 0.35 Mbps throughput loss while the secondary only loses 0.06 Mbps. The primary throughput loss can be attributed to the secondary picking the incorrect channel. As all the boards constantly report statistics to the central server we can track the number of time the secondary link picks the incorrect channel; 1.1% of the time the secondary will use the same channel as the primary channel. The performance of the distributed algorithm can be compared with the centralized where the perfect channel is always chosen. Here the primary sees only a 0.10 Mbps drop in throughput. Therefore, there is scope for improvement for the distributed algorithm.

These two implementation examples were developed to show the flexibility available on the board and exercise the various features of the hardware and software. They have not been designed to show a novel cognitive algorithm. Real cognitive radio algorithms will be mapped to the resources they require, and designers can use the characterization as a guide to optimizing their system performance.

## Conclusion and Future Work

---

In this work we presented a wireless communications platform that enables network-wide experimentation. Other platforms in the field are focussed on just laboratory-based experiments where there is ease of observing the performance of custom algorithms. Our platform not only enables the functionality to implement novel and custom algorithms, but also scales the system to a deployed network. We incorporate the frameworks needed achieve such deployments, have a fine-grained temporal view into system performance and control the behavior for scheduled experiments.

We described the hardware architecture of the platform in detail along with the software frameworks built to enable the above functionality. The platform has all the horsepower needed to implement custom physical, MAC and other layers of the networking stack. We characterized the backdoor control channel and presented frameworks for data gathering and controlling network deployments.

In addition, we tackled an emerging wireless technology implementation, cognitive radio, on the platform itself. We showed that the hardware can be extended beyond its original purpose and ably support the needs of new technologies.

## References

---

- [1] E. Blossom, “GNU Radio.” [Online]. Available: <http://www.gnuradio.org> 1.3
- [2] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, “Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols,” in *Proceedings of IEEE Wireless Communications and Networking Conference*, 2005. 1.3
- [3] D. Raychaudhuri, N. B. Mandayam, J. B. Evans, B. J. Ewy, S. Seshan, and P. Steenkiste, “CogNet: an architectural foundation for experimental cognitive radio networks within the future internet,” in *MobiArch '06: Proceedings of first ACM/IEEE international workshop on Mobility in the evolving internet architecture*. New York, NY, USA: ACM, 2006, pp. 11–16. 1.3
- [4] G. J. M. et. al., “KUAR: A flexible software-defined radio development platform,” in *DySPAN*, 2007. 1.3
- [5] J. Camp, E. Knightly, and W. Reed, “Developing and deploying multihop wireless network for low-income families,” in *Proceedings of Digital Communities 2005*, Napoli, Italy, June 2005. 1.3
- [6] P. Murphy, A. Sabharwal, and B. Aazhang, “Design of WARP: A flexible wireless open-access research platform,” in *Proceedings of EUSIPCO*, 2006. 1.3, 2
- [7] S. Caban, C. Mehlführer, R. Langwieser, A. L. Scholtz, and M. Rupp, “Vienna MIMO testbed,” *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 54868, 2006. [Online]. Available: [http://publik.tuwien.ac.at/files/pub-et\\_10929.pdf](http://publik.tuwien.ac.at/files/pub-et_10929.pdf) 1.3
- [8] W. Mohr, “The WINNER project – development of a radio interface for systems beyond 3G,” in *International Journal of Wireless Information Networks*, 2007. 1.3
- [9] J. Zhao, H. Zheng, and G.-H. Yang, “Distributed coordination in dynamic spectrum allocation networks,” in *New Frontiers in Dynamic Spectrum Access Net-*

- 
- works, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, Nov. 2005, pp. 259–268. 1.3
- [10] P. Bahl, A. Adya, J. Padhye, and A. Wolman, “Reconsidering wireless systems with multiple radios,” in *ACM SIGCOMM Computer Communications Review*, 2004. 1.3
- [11] “Xilinx Virtex-4 FPGA,” <http://xilinx.com/virtex4>. 2.1.1
- [12] J. So and N. H. Vaidya, “Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver,” in *Mobile Adhoc Networking and Computing (MobiHoc)*, 2004. 2.2
- [13] P. Kyasanur and N. Vaidya, “Routing and interface assignment in multi-channel multi-interface wireless networks,” in *IEEE Wireless Communications and Networking Conference*, 2005. 2.2
- [14] S.-L. Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu, “A new multi-channel mac protocol with on-demand channel assignment for multi-hop mobile ad hoc networks,” in *Int’l Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN)*, 2000. 2.2
- [15] “Axis Etrax 100LX MCM4+16,” [http://www.axis.com/products/dev\\_etrax\\_100lx\\_mcm/](http://www.axis.com/products/dev_etrax_100lx_mcm/). 2.2.1
- [16] “Xilinx Spartan-3AN FPGA,” <http://www.xilinx.com/spartan3an>. 2.2.2
- [17] J. Wang, Y. Fang, and D. Wu, “SYN-DMAC: a directional MAC protocol for ad-hoc networks with synchronization,” in *Military Communications Conference*, 2005. 2.2.4
- [18] “WARPnet online repository,” <http://warp.rice.edu/trac>. 3
- [19] *9Xtend OEM Module Product Manual*, Digi International Inc. 3.2
- [20] “WARP OFDM reference design.” 3.2.1, 3.2.2
- [21] Y. Xiao, “IEEE 802.11n: Enhancements for higher throughput in wireless LANs,” in *IEEE Wireless Communications*, 2005. 3.2.2
- [22] T. S. Rappaport, *Wireless Communications: Principles & Practice*. Prentice Hall. 3.2.3, 3.2.3
- [23] S. Y. Seidel and T. S. Rappaport, “914 MHz path loss prediction models for indoor wireless communications in multifloored buildings,” in *IEEE Transactions on Antennas and Propagation*, 1992. 3.2.3

## Schematics

---

The schematics for the FPGA Board and Backdoor Board are attached in this appendix. The Cadence Design Entry CIS tool was used to create these schematics.

### A.1 FPGA Board


The schematics for FPGA Board v2.2.

# WARP FPGA BOARD

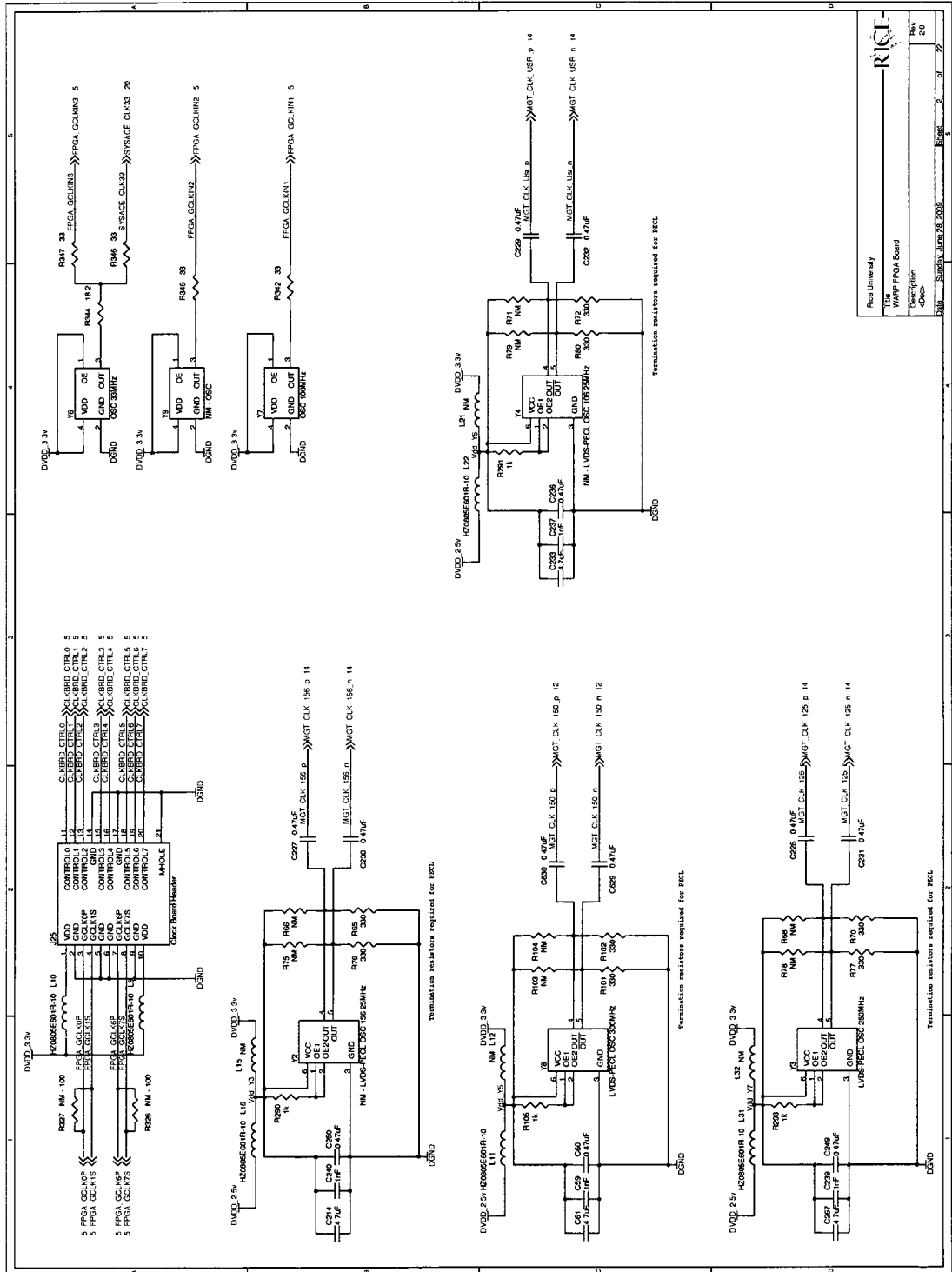
Version 2.2

Siddharth Gupta

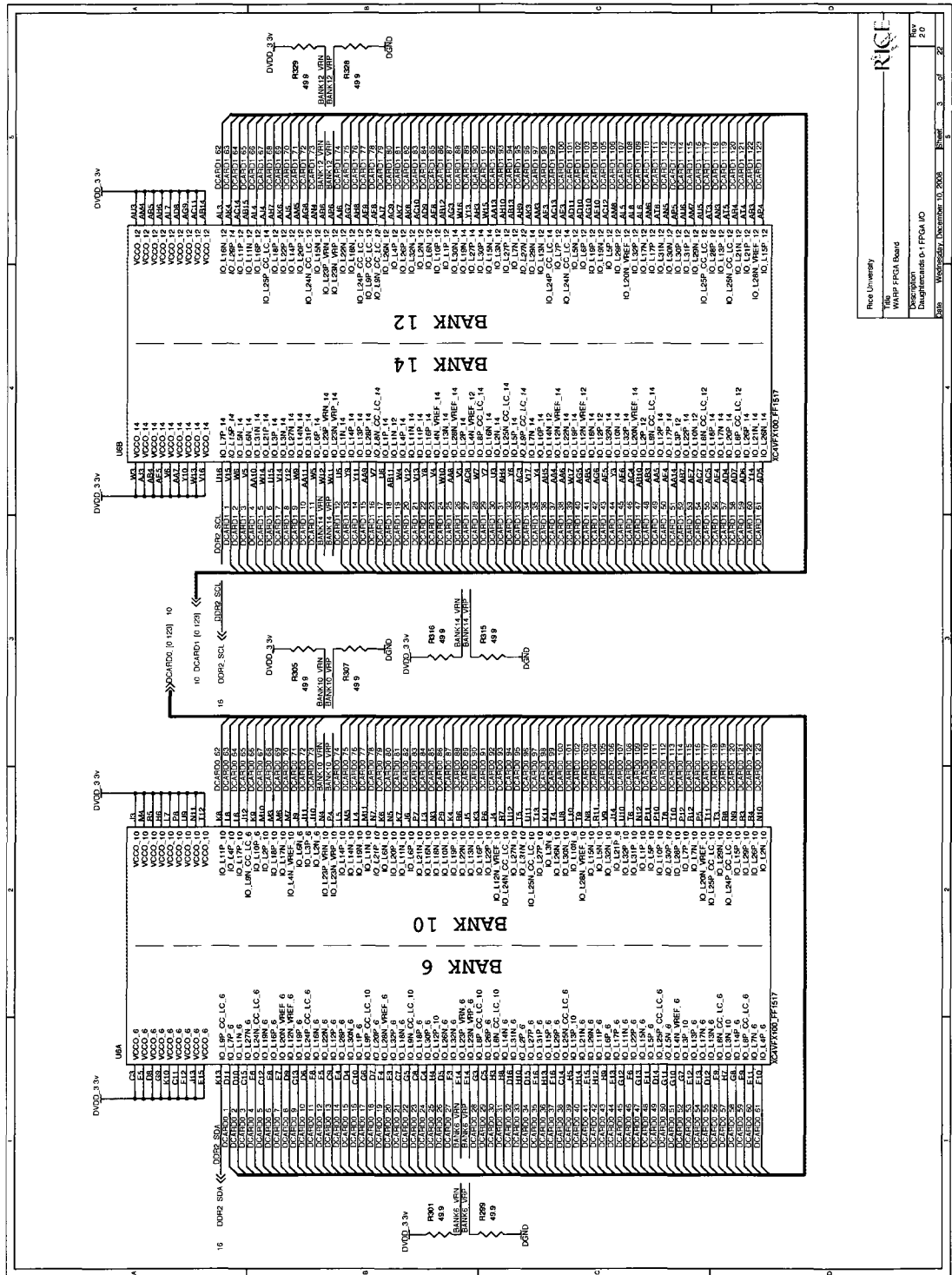
- 1 - Table of Contents
- 2 - Clocks
- 3 - FPGA Banks 6,10,12,14
- 4 - FPGA Banks 5,9,11,13
- 5 - FPGA Banks 1,2,3,4
- 6 - FPGA Banks 7,8,0
- 7 - FPGA Power
- 8 - Bypass Capacitors for FPGA
- 9 - Power Regulators
- 10 - Daughtercard Headers 0-1
- 11 - Daughtercard Headers 2-3
- 12 - Multi-Gigabit Transceivers: HSSDCII, SFP
- 13 - Multi-Gigabit Transceivers: SATA
- 14 - Multi-Gigabit Transceivers: Clock Distribution
- 15 - Gigabit Ethernet
- 16 - DDR2 SODIMM Memory: Control/Data
- 17 - DDR2 SODIMM Memory: Termination
- 18 - DDR2 Power and User I/O
- 19 - User I/O
- 20 - SystemACE Controller
- 21 - Auxiliary Header
- 22 - USB Configuration

	
Rice University	
Title	WARP FPGA Board
Description	
Doc#	
Rev	4.0
File	Schematic_06_27_2009
Sheet	1 of 22

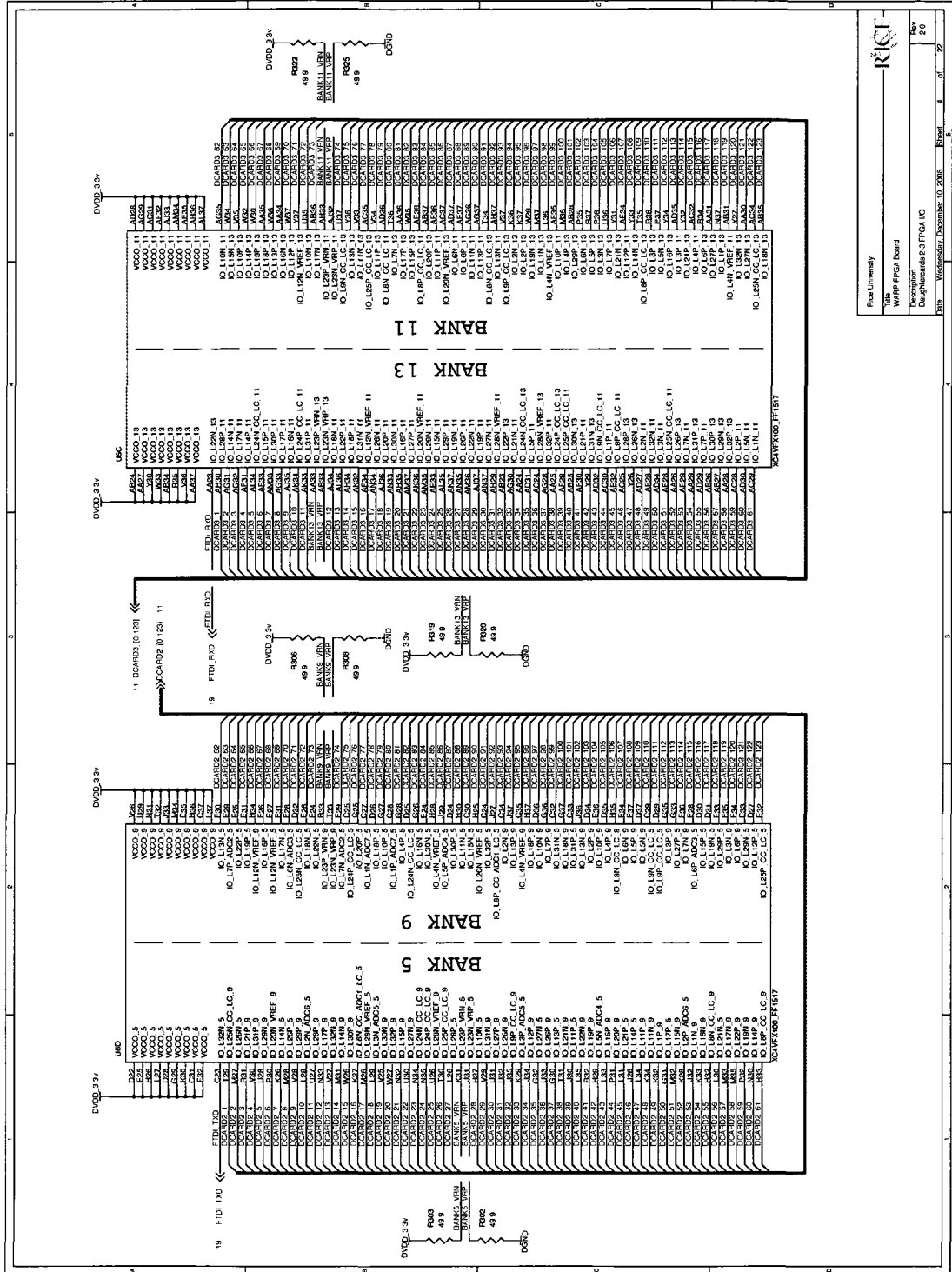




Rice University  
WARP FPGA Board  
Schematic  
Date: 06/18/2009  
Page: 2 of 22

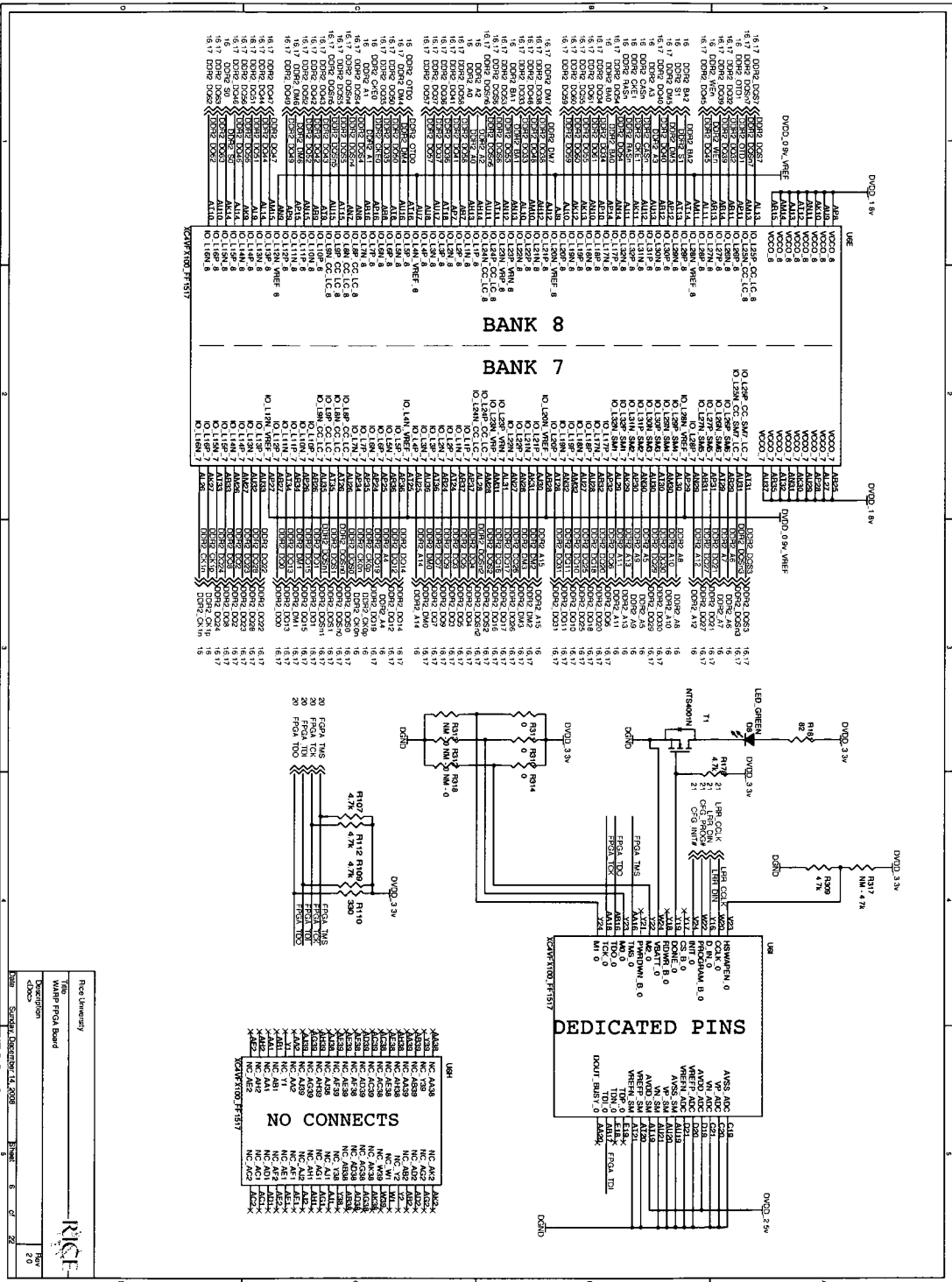


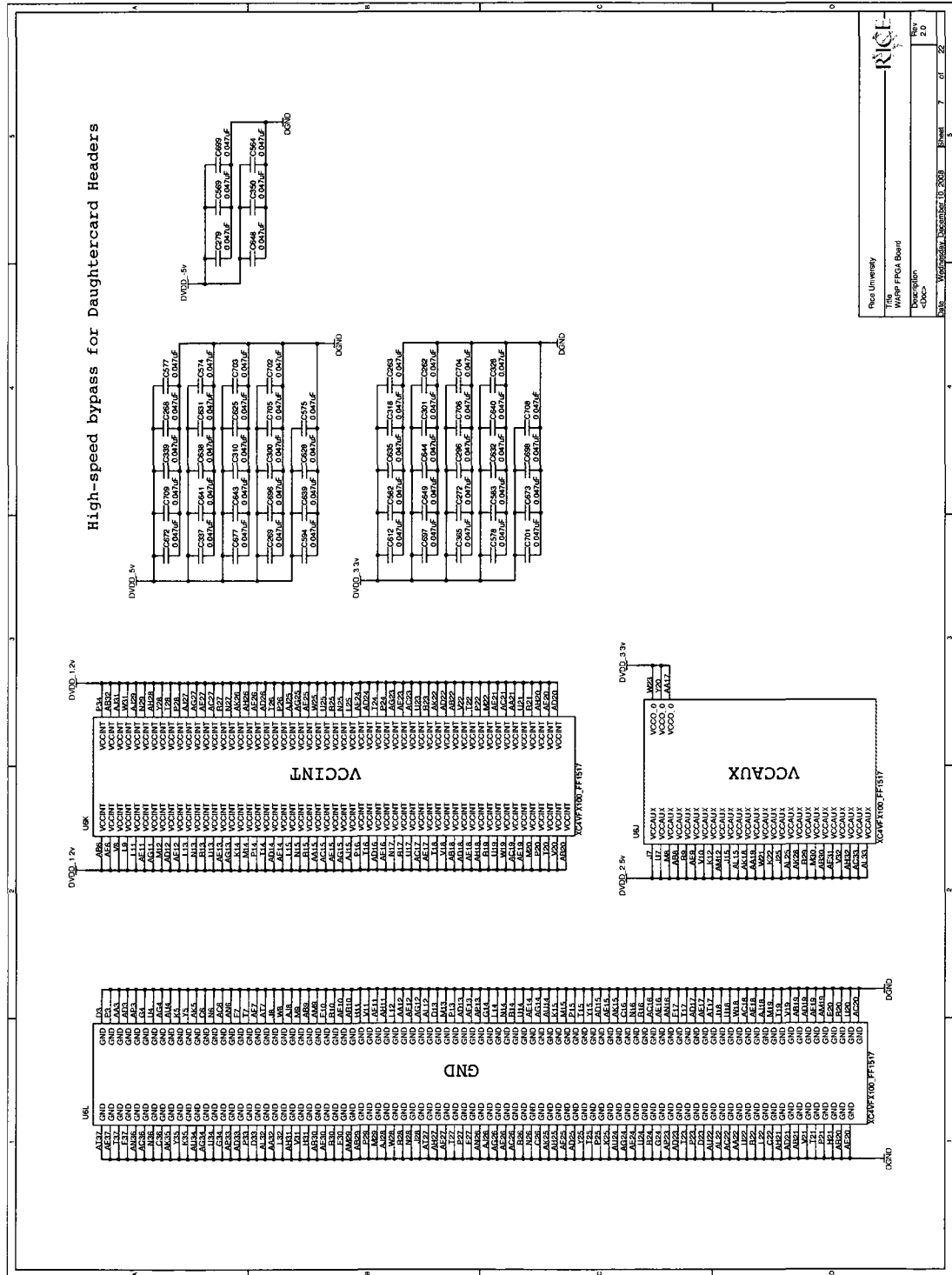
RICE University  
WARP FPGA Board  
Schematic of 0-1 FPGA IO  
Date: 10/26/2009, December 10, 2009, Sheet: 3 of 22




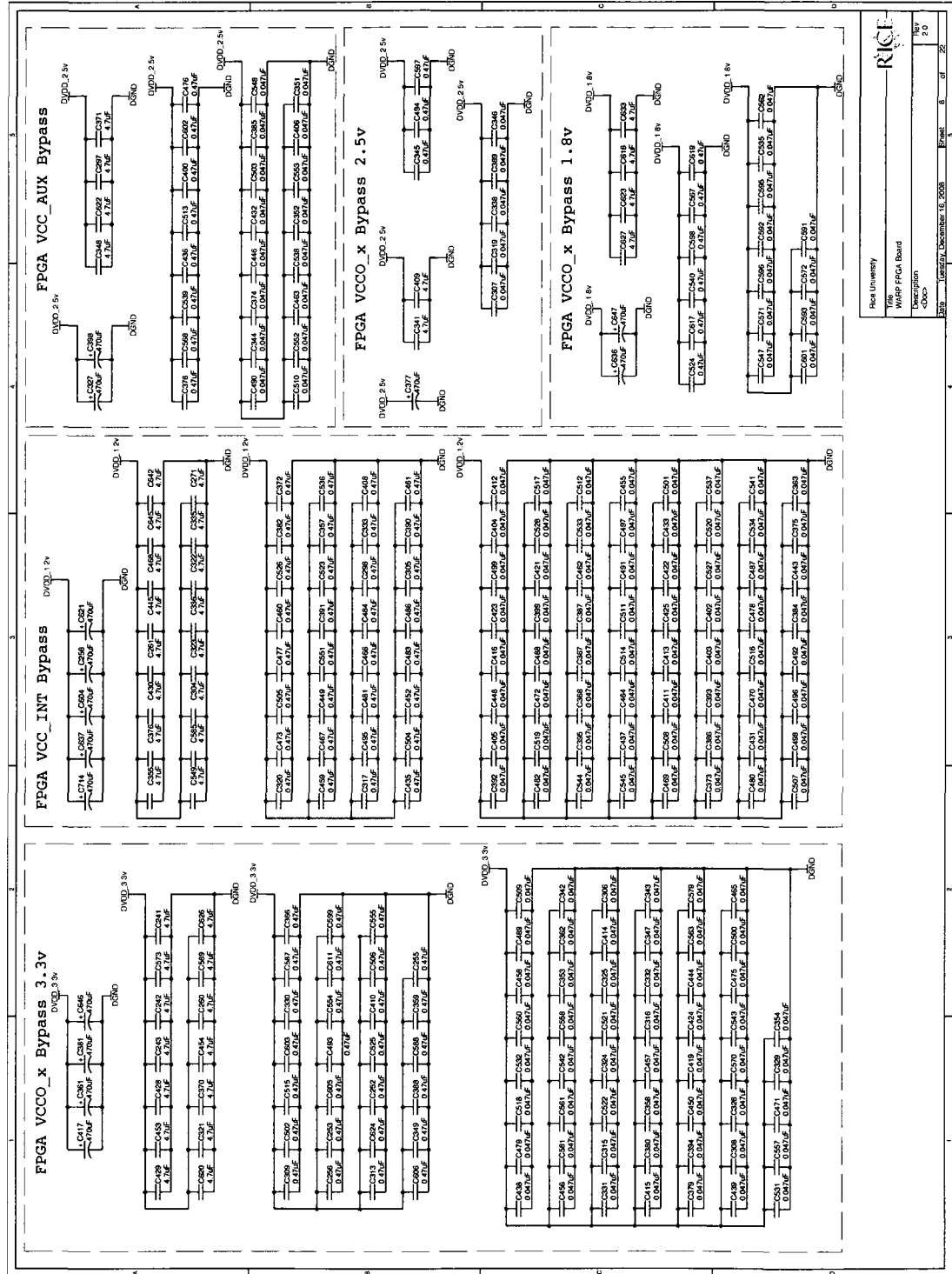
RIET  
Rice University  
Warp FPGA Board  
Kucukozkan  
Dağdelençulu, December 10, 2008  
Rev. 1.0  
Page 70

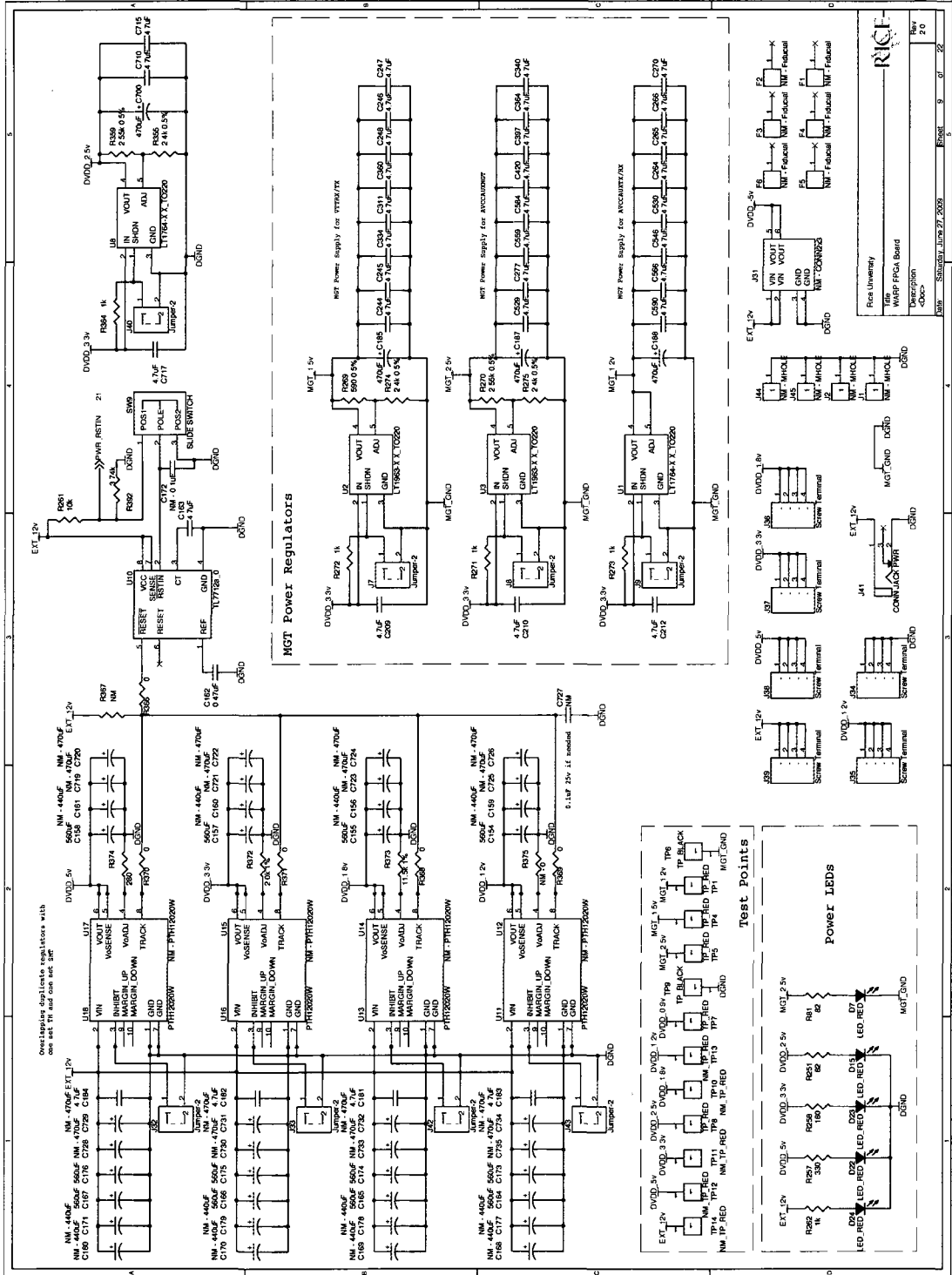




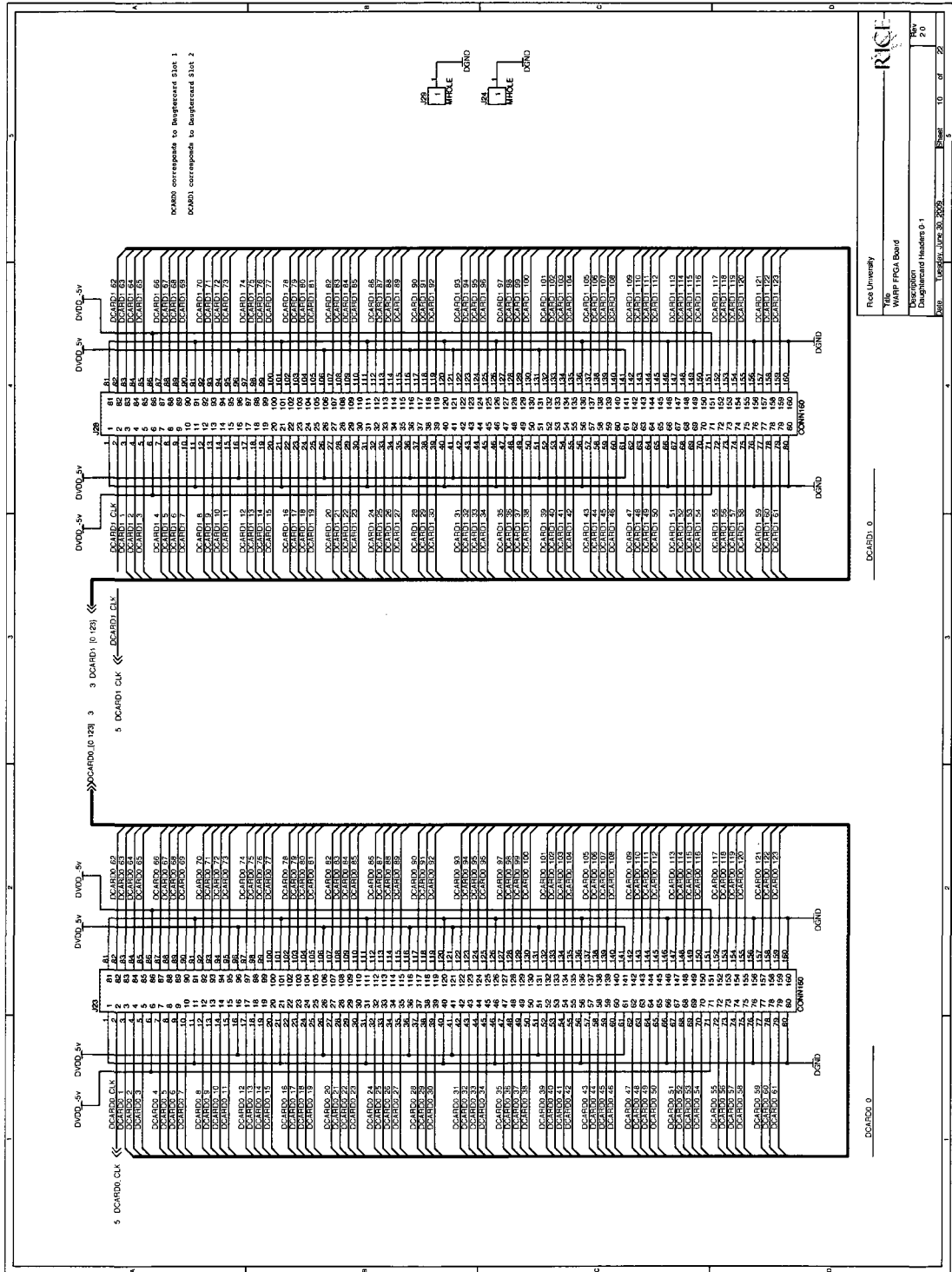


  
 Rice University  
 The WARP Project Board  
 Revision  
 1-00-0  
 Date: 11/28/2008 10:28:00 AM 7 of 22

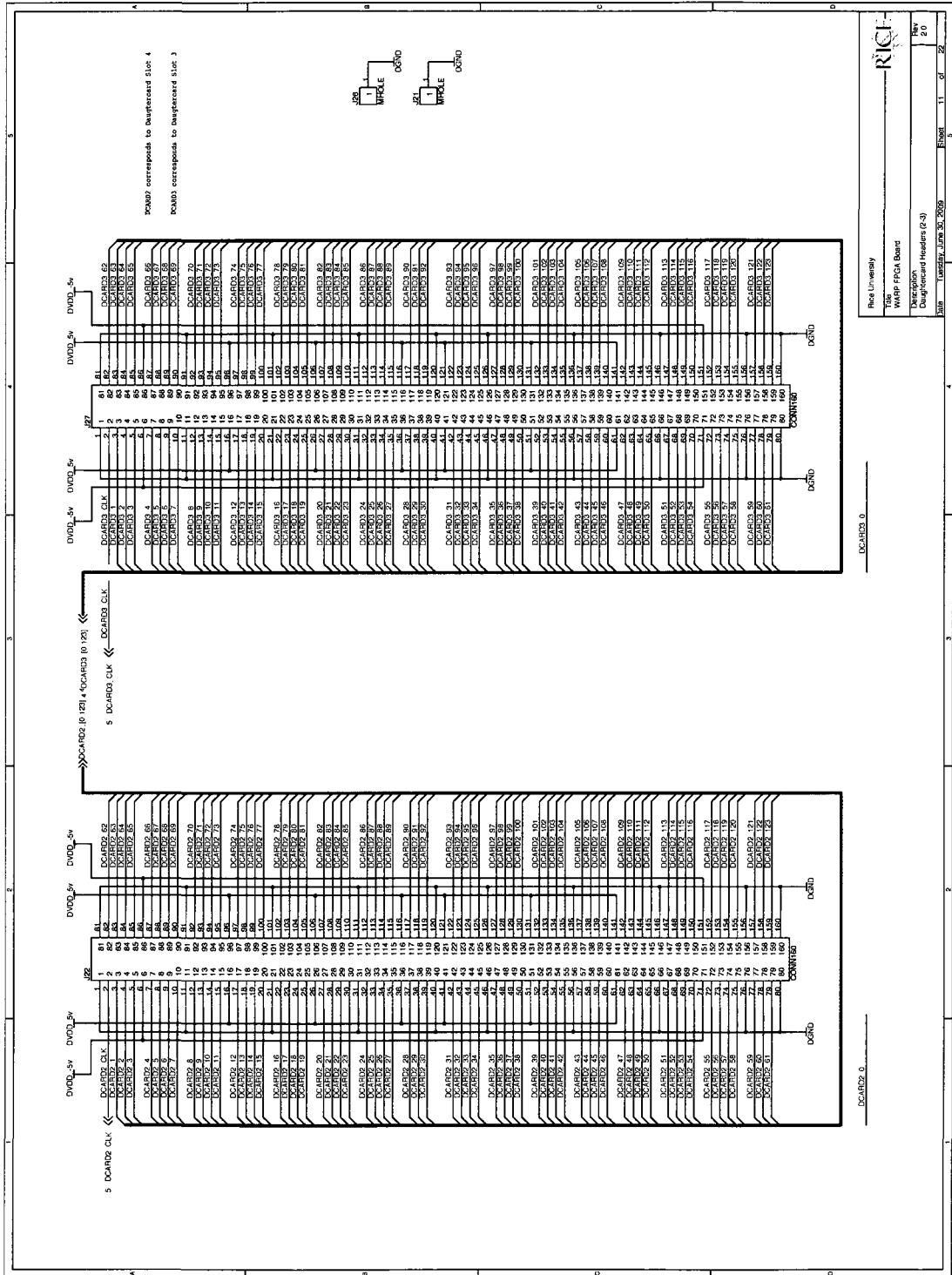


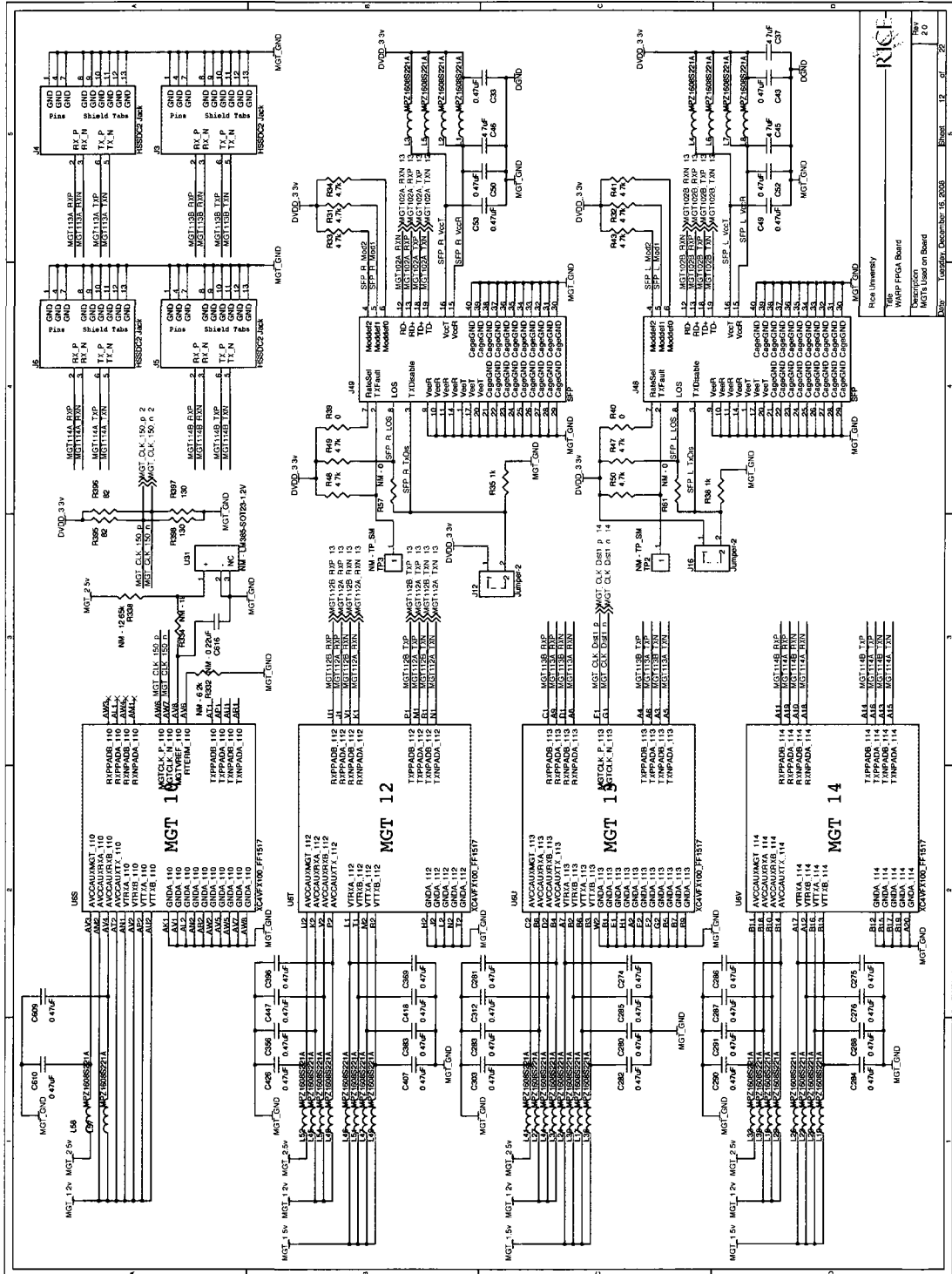






RICE  
 Rice University  
 Title: WARP FPGA Board  
 Description: Hardware Header 0 1  
 Date: Tuesday, June 30, 2009  
 Page: 10 of 22  
 Rev: 2.0



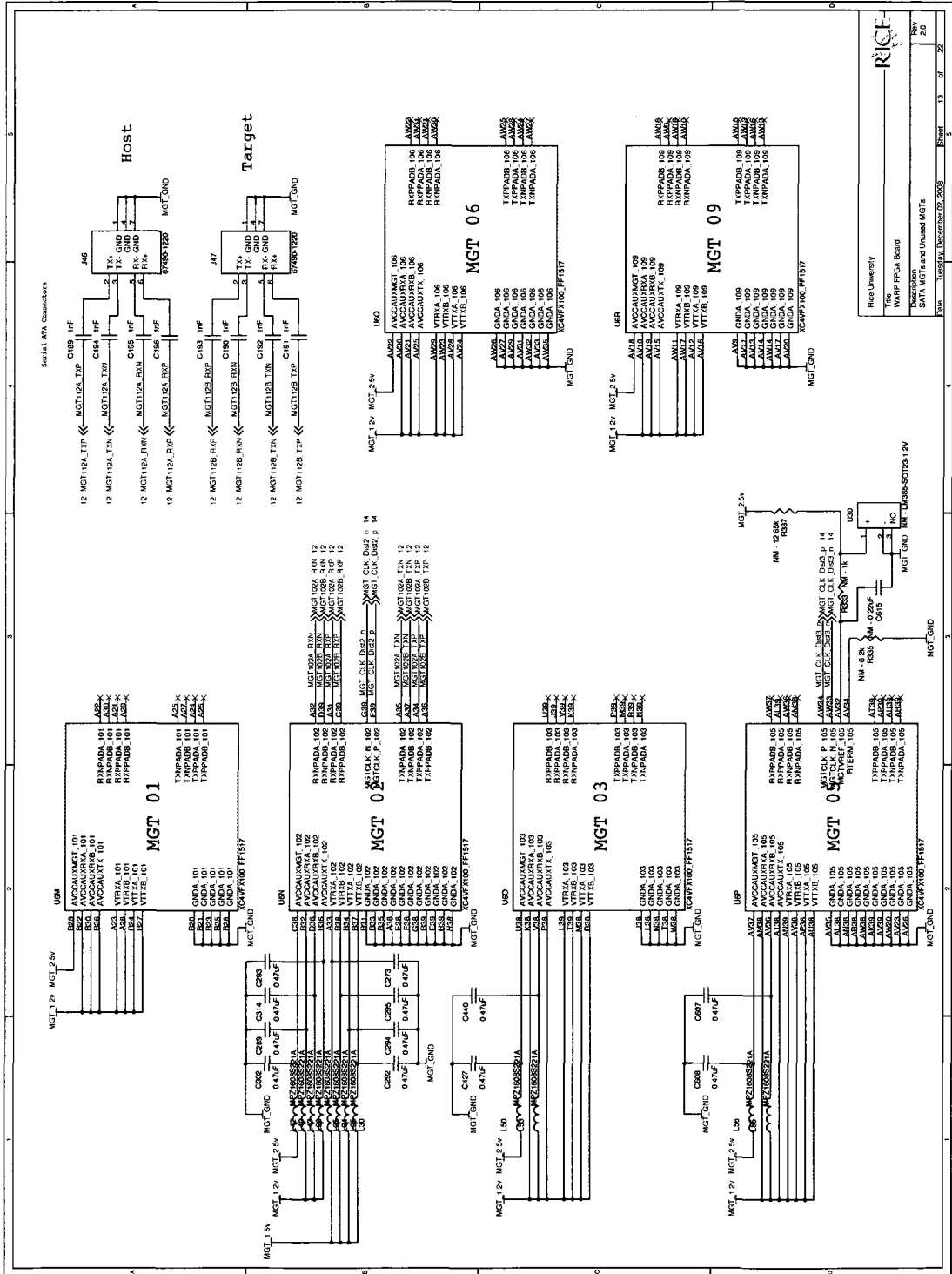


**Revision History**

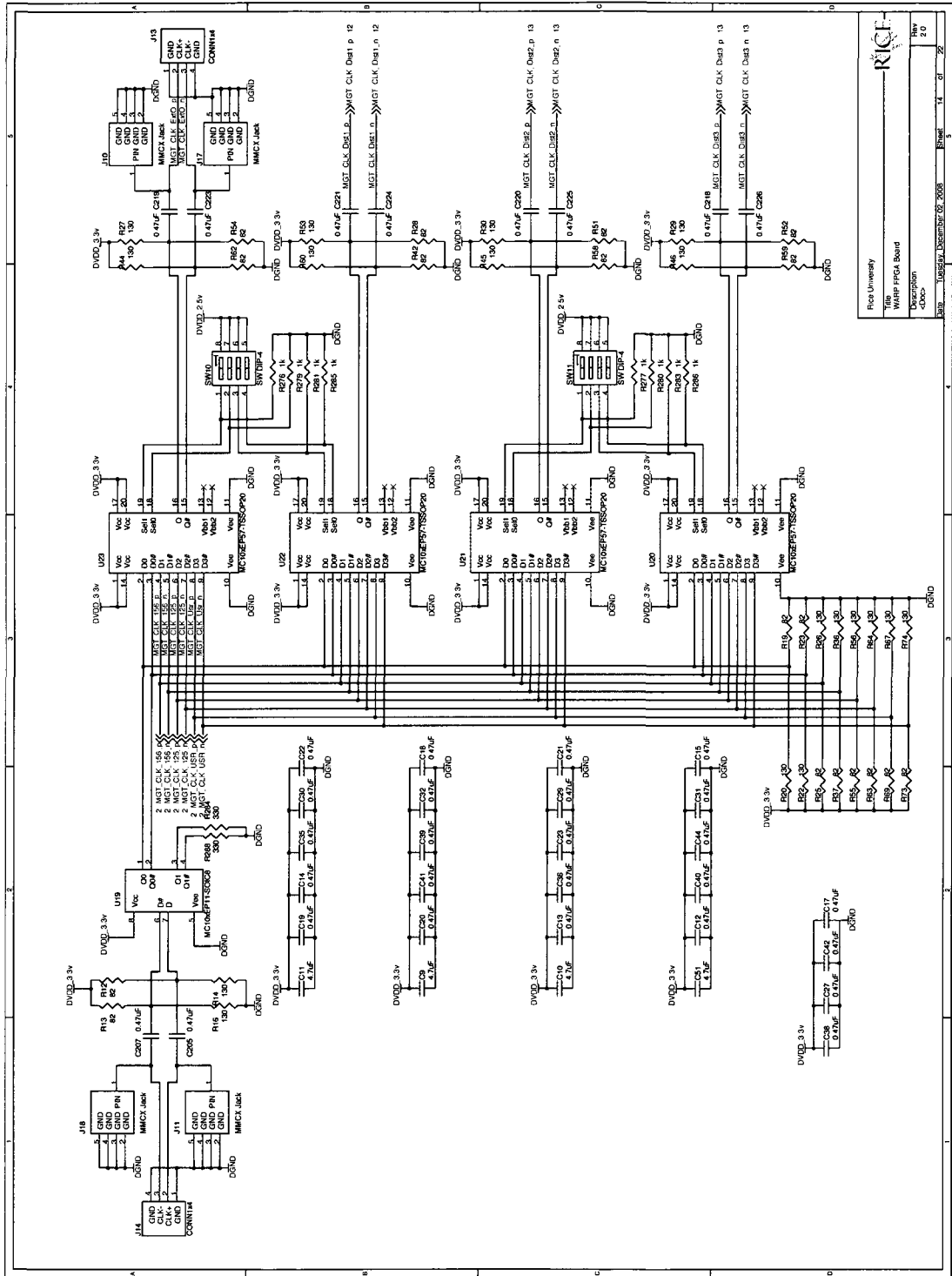
Rev	Description	Date
1.0	Initial Release	12/12/08
2.0	WARP FPGA Board	12/12/08

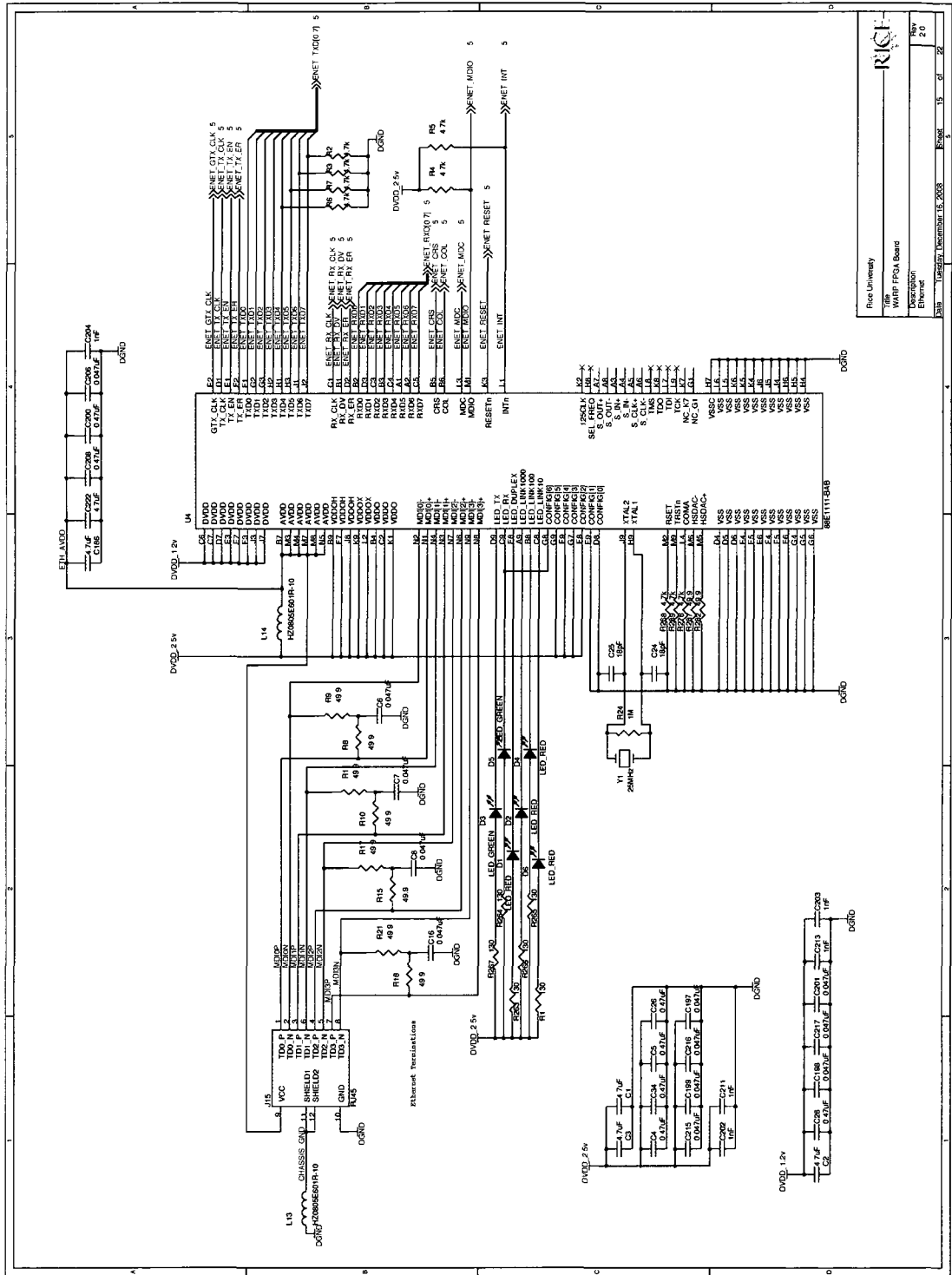
WARP FPGA Board  
MGTs Used on Board

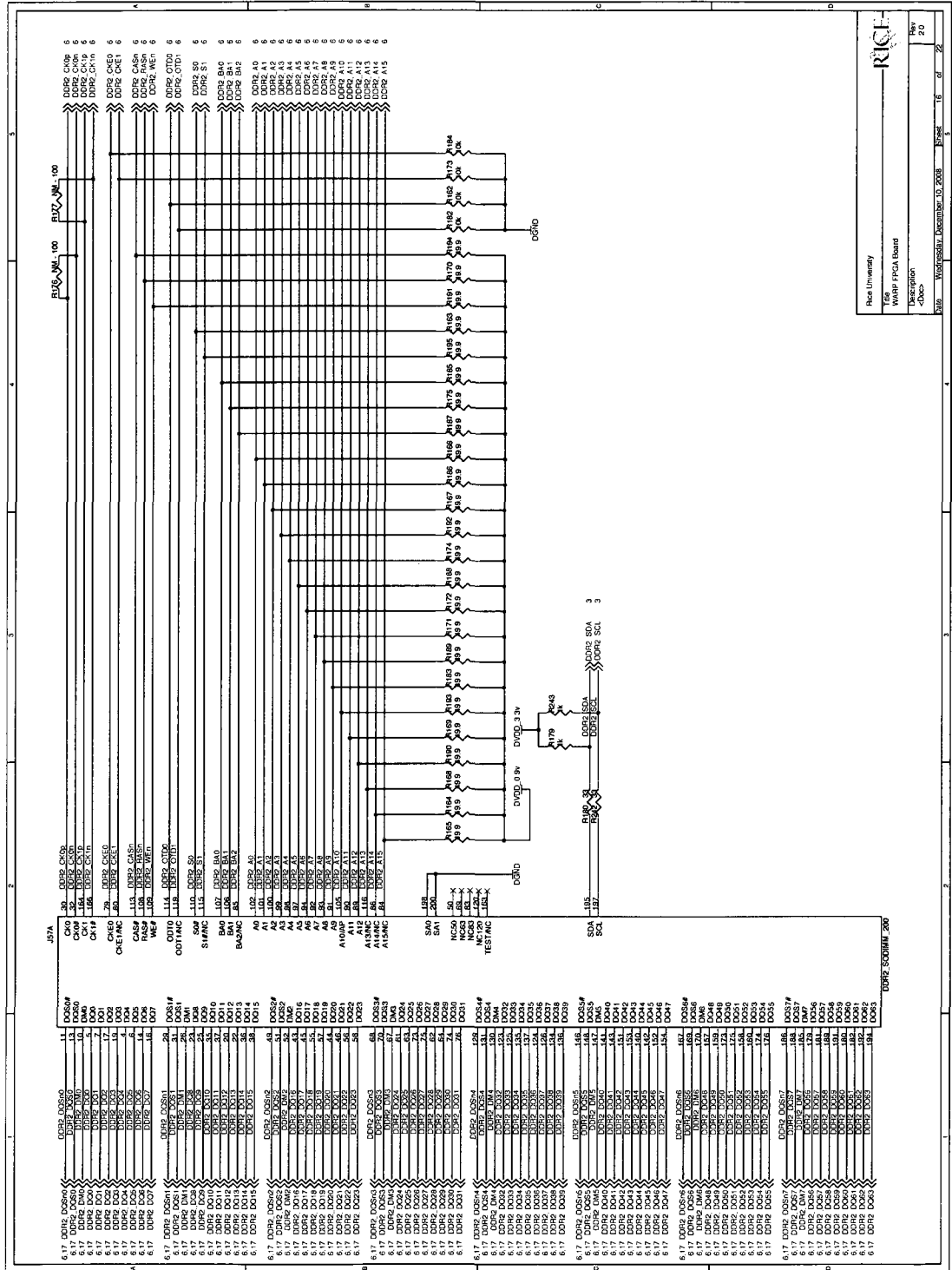
Revision History  
Rev 2.0  
12/12/08



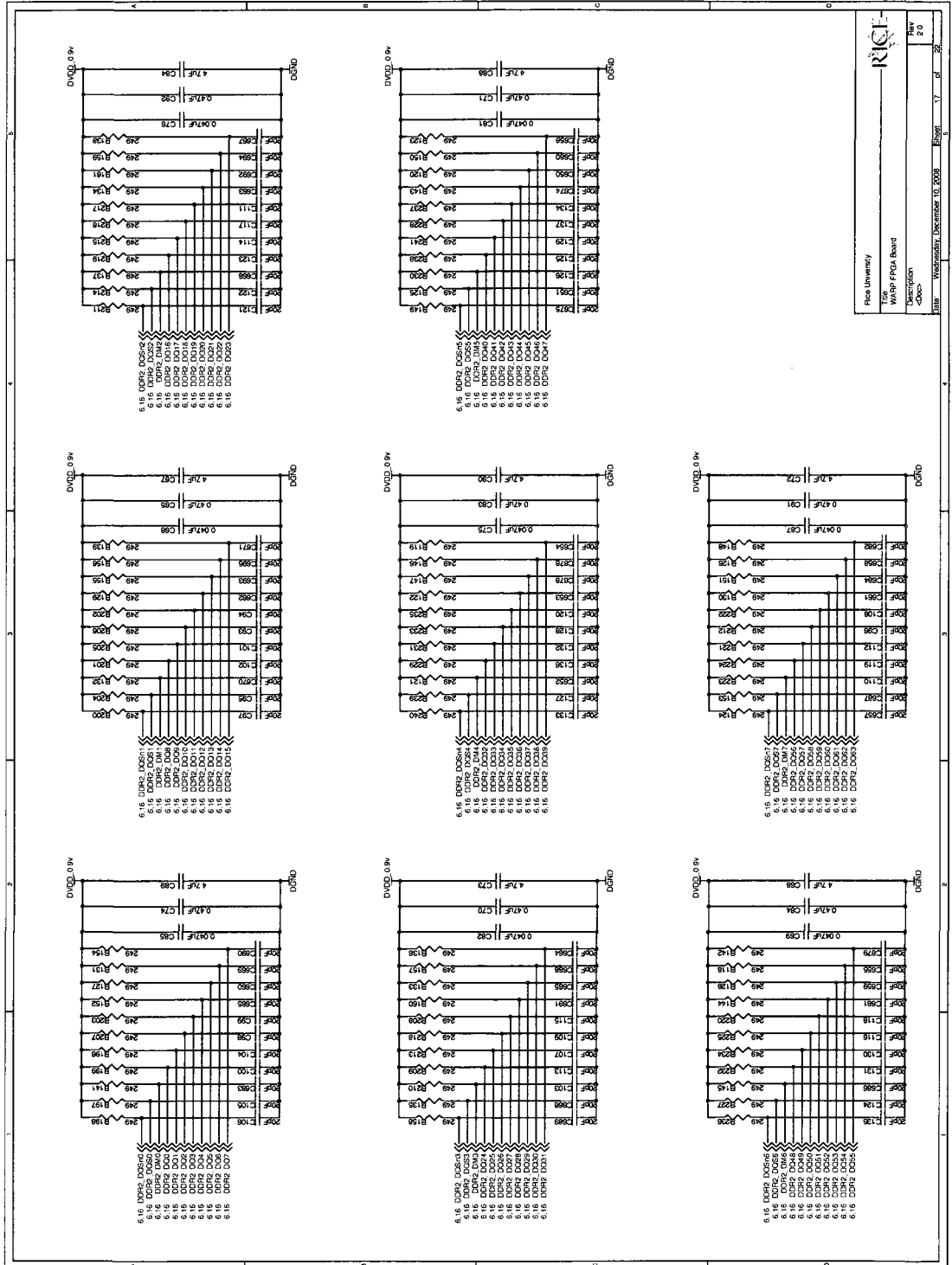
RIET  
Rice University  
Title: WARP FPGA Board  
Author: [Name]  
Date: [Date]  
Rev: [Revision]  
Page: 13 of 22



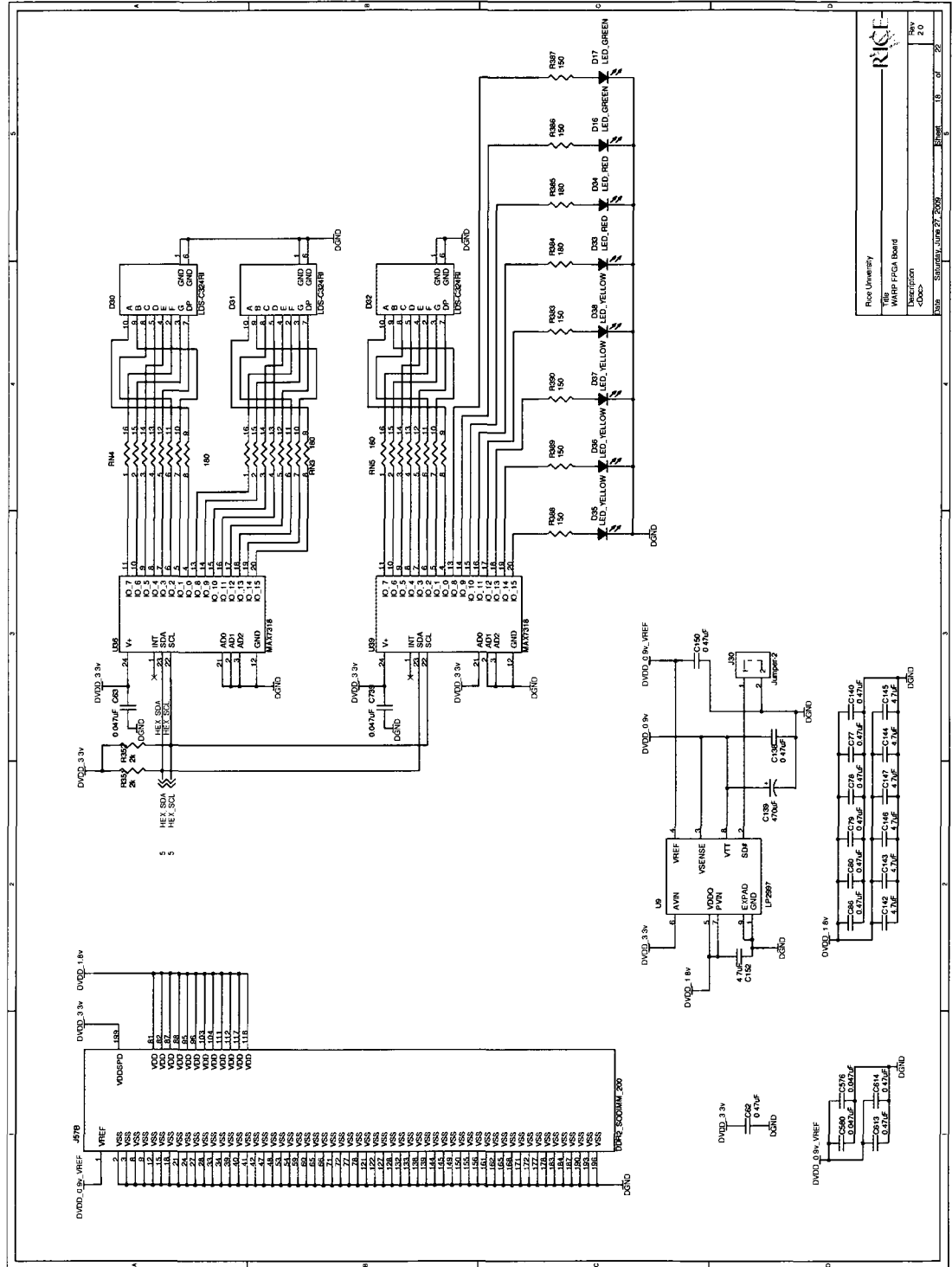




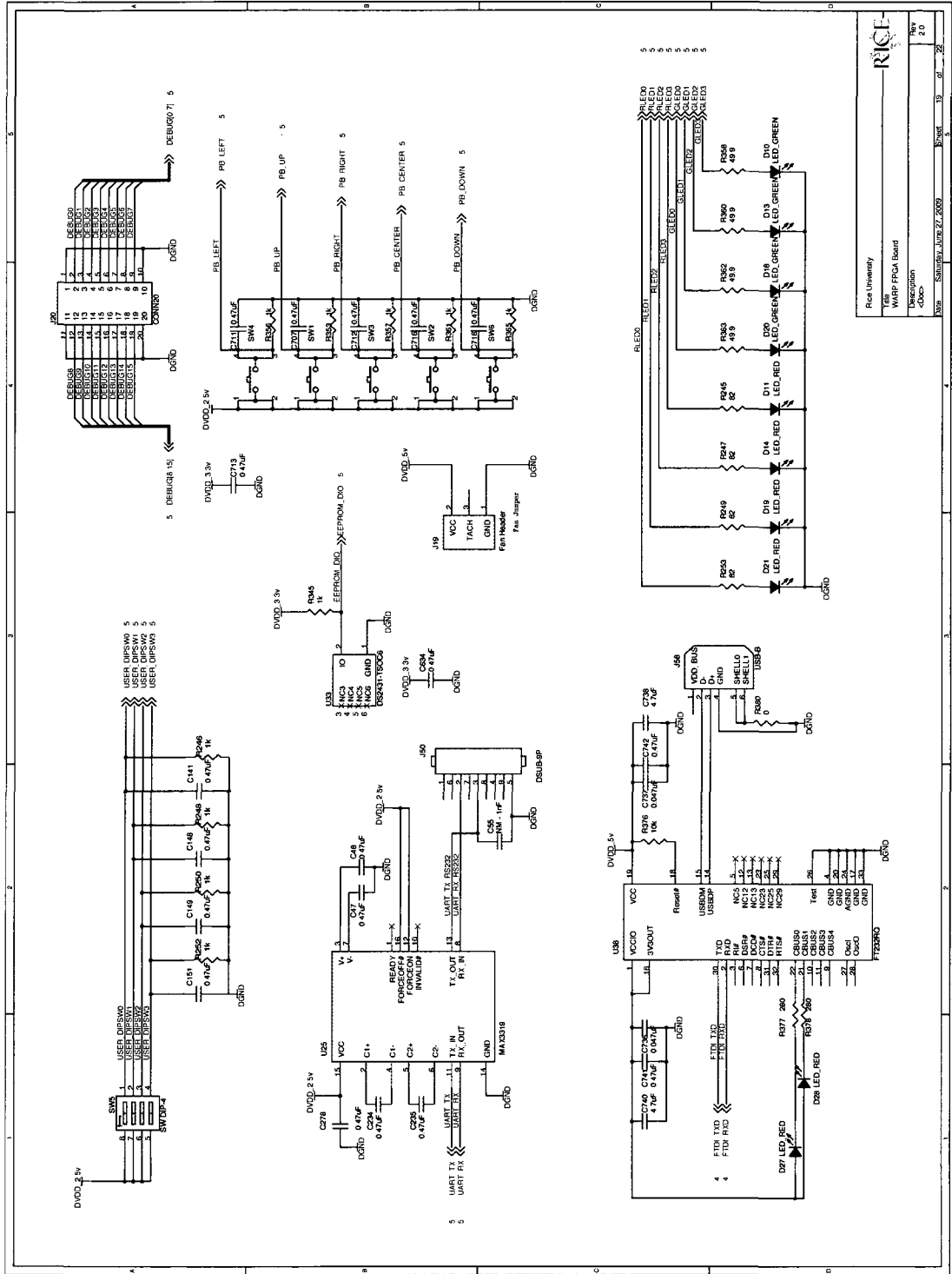
Rice University
   
 Title: WARP FPGA Board
   
 Revision:
   
 Date: 10/26/2008
   
 Page: 16 of 28
   
 710



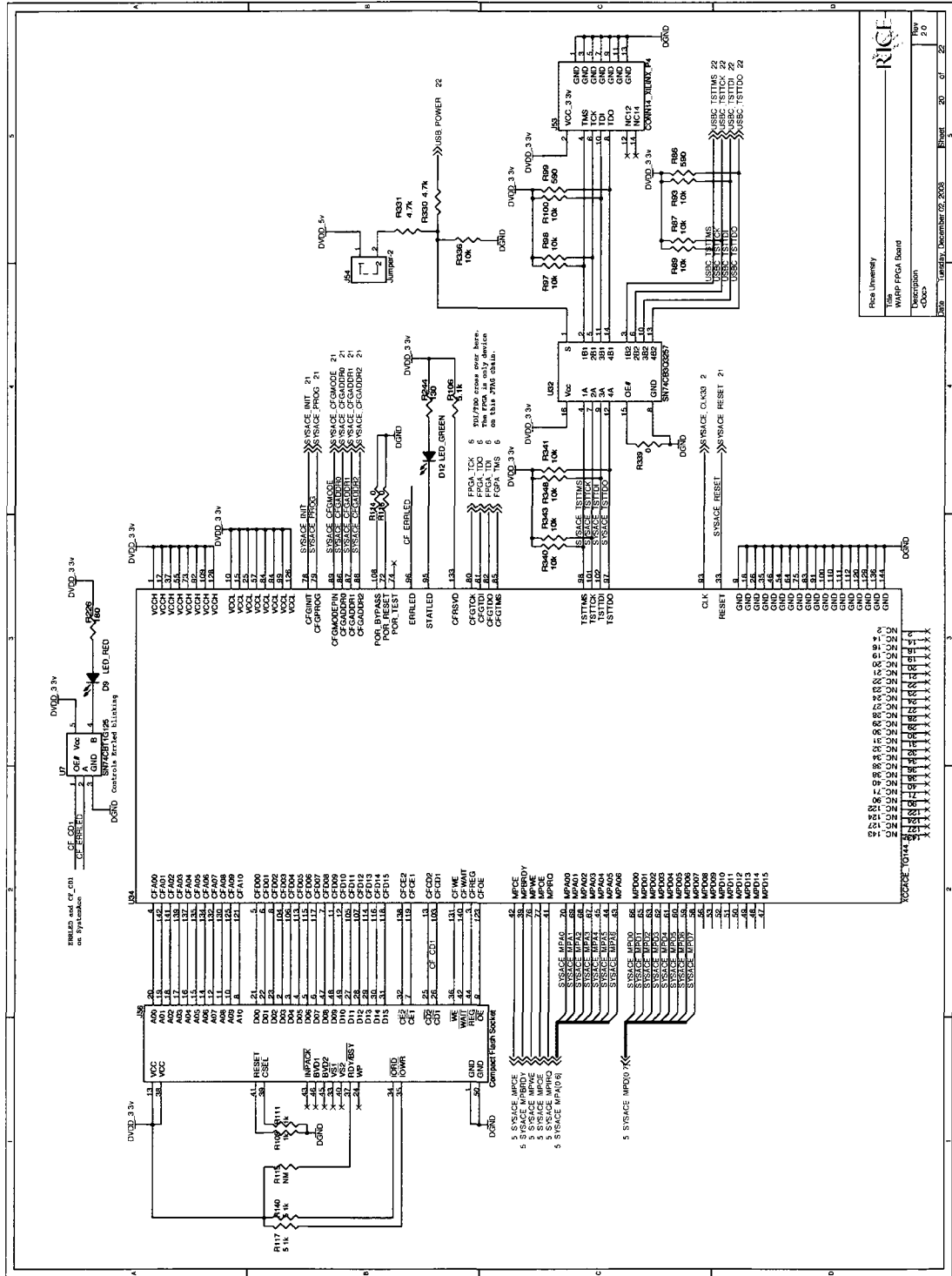




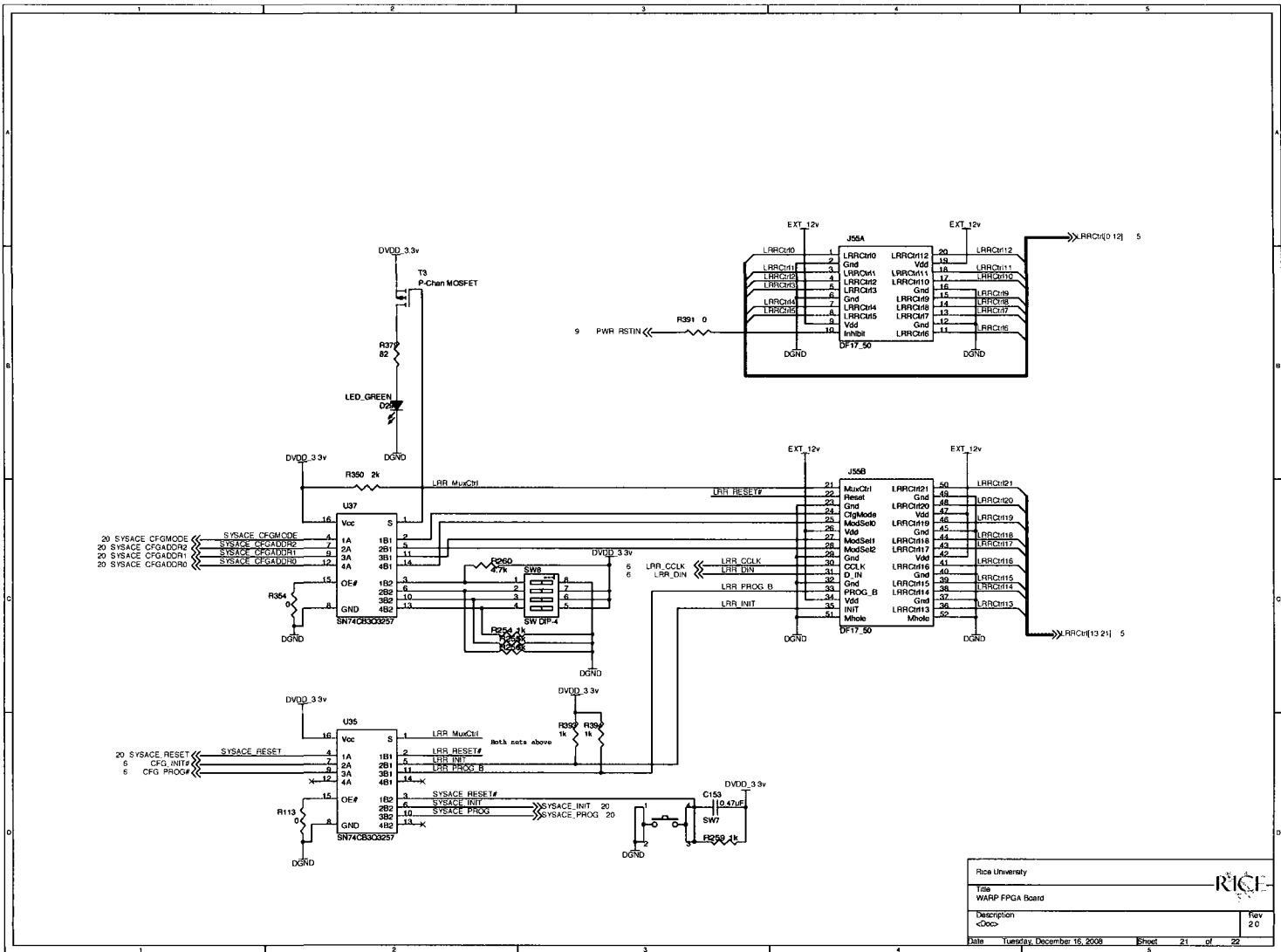
Rice University	Sheet 18 of 22
Title: WARP FPGA Board	
Description:	
Date: 2/1	Rev: 2
Author: Siddhant Kumar	Date: 2/1

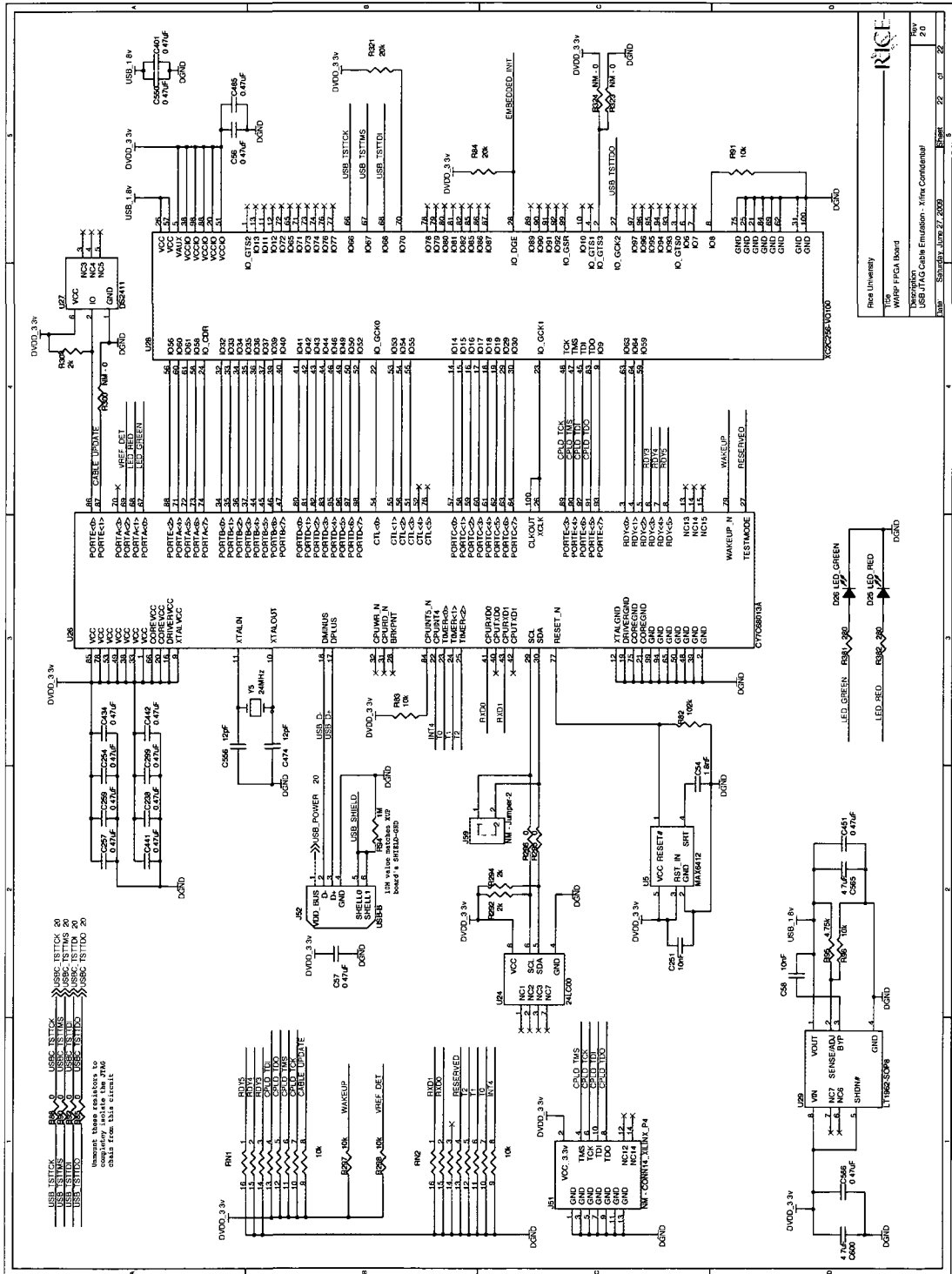


Rice University  
 Viterbi FPGA Board  
 Revision 2.0  
 Date: 10/27/2009



Rev	Desc	Date
1.0	WARP FPGA Board	2008-12-08
2.0	WARP FPGA Board	2008-12-08





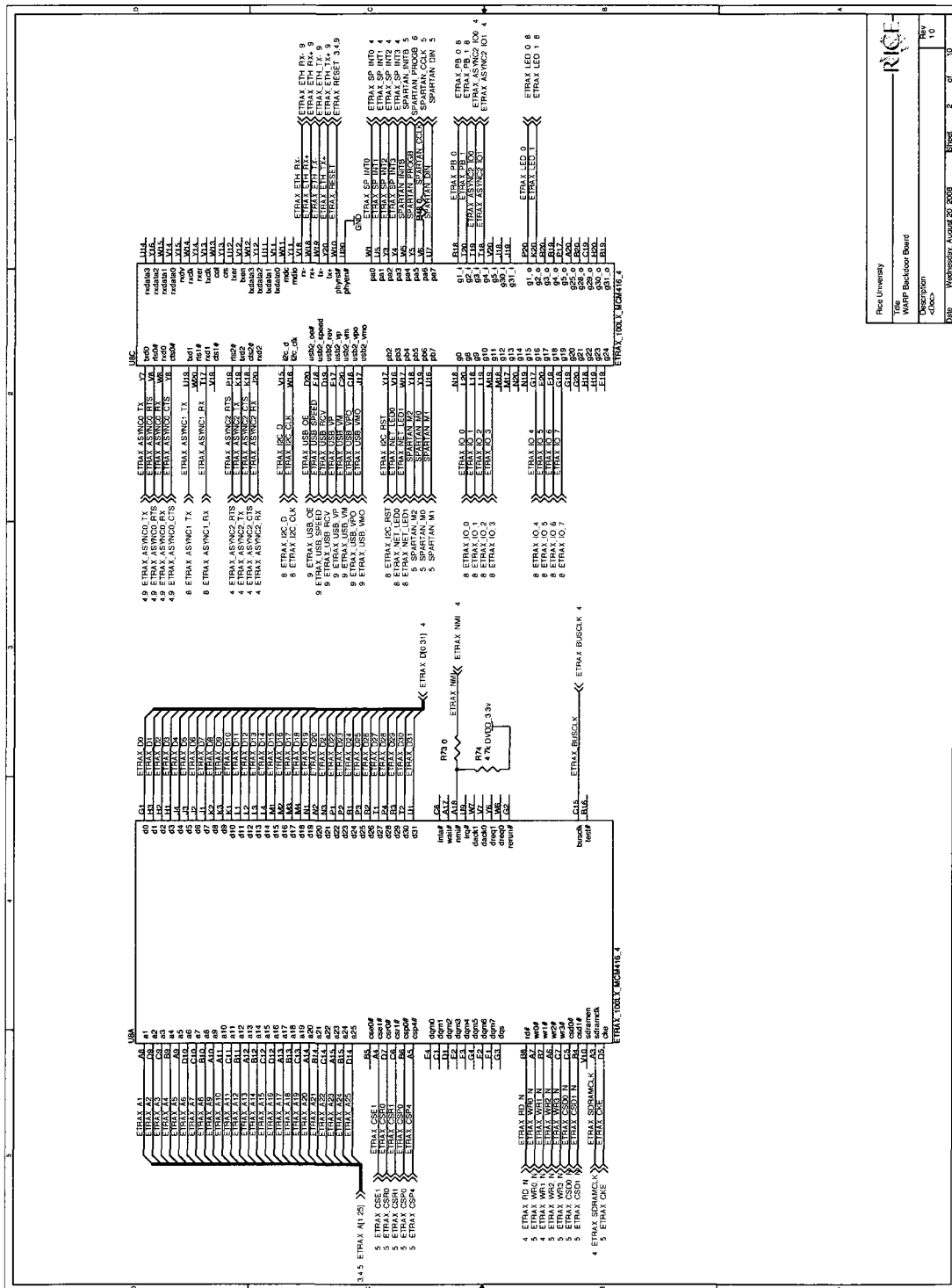
## A.2 Backdoor Board

The schematics for Backdoor Board v1.0.

WARP Backdoor Board

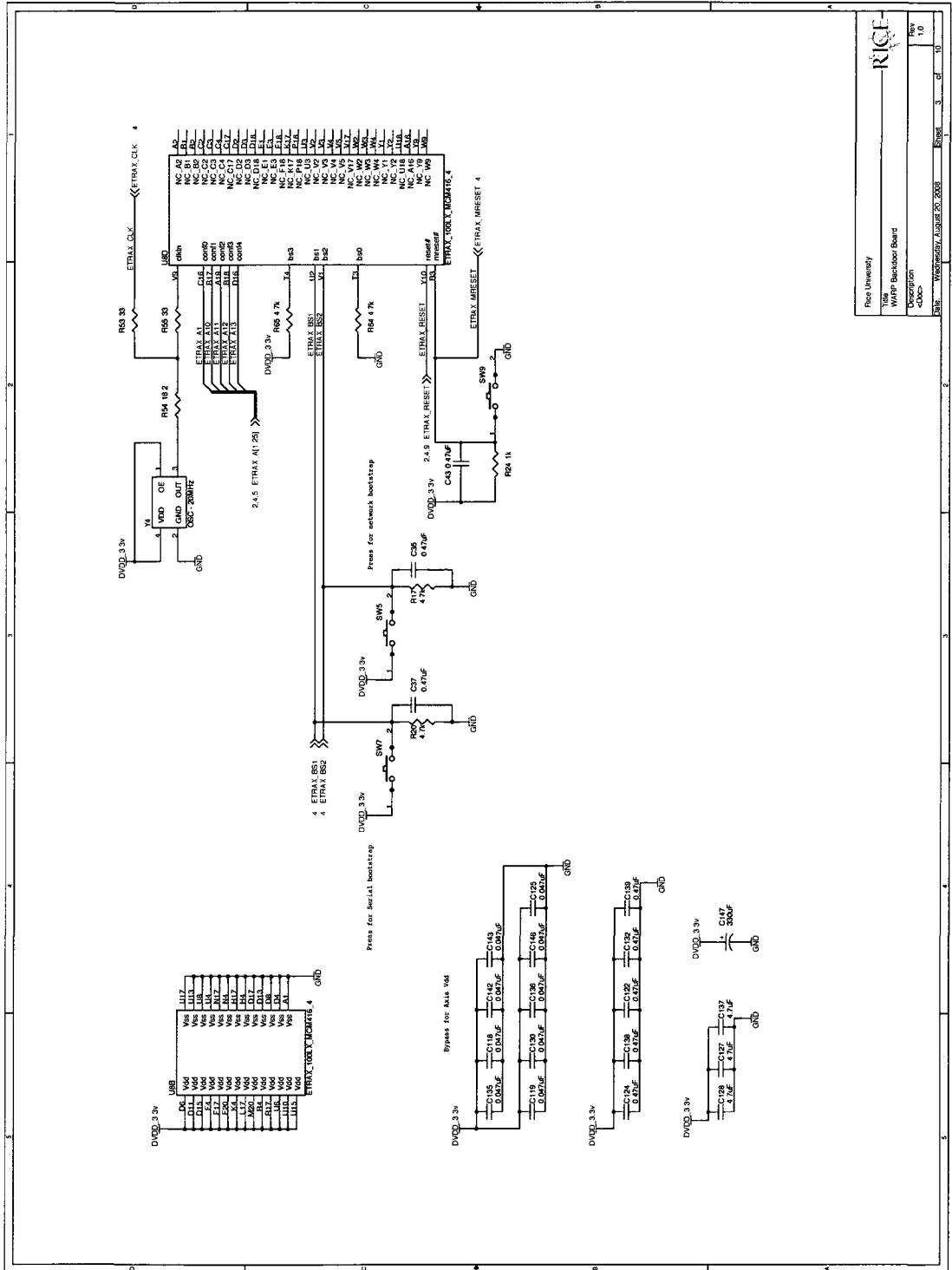
- 1 - table of contents
- 2 - axis io, bus
- 3 - axis power, control
- 4 - spartan io 0,1
- 5 - spartan io 2,3
- 6 - spartan power, config
- 7 - power
- 8 - gps, maxstream
- 9 - usb, ethernet, serial
- 10 - fpga header

Rice University	
Warp Backdoor Board	
Revision	1.0
Date	Wednesday, August 29, 2006
Page	1 of 10

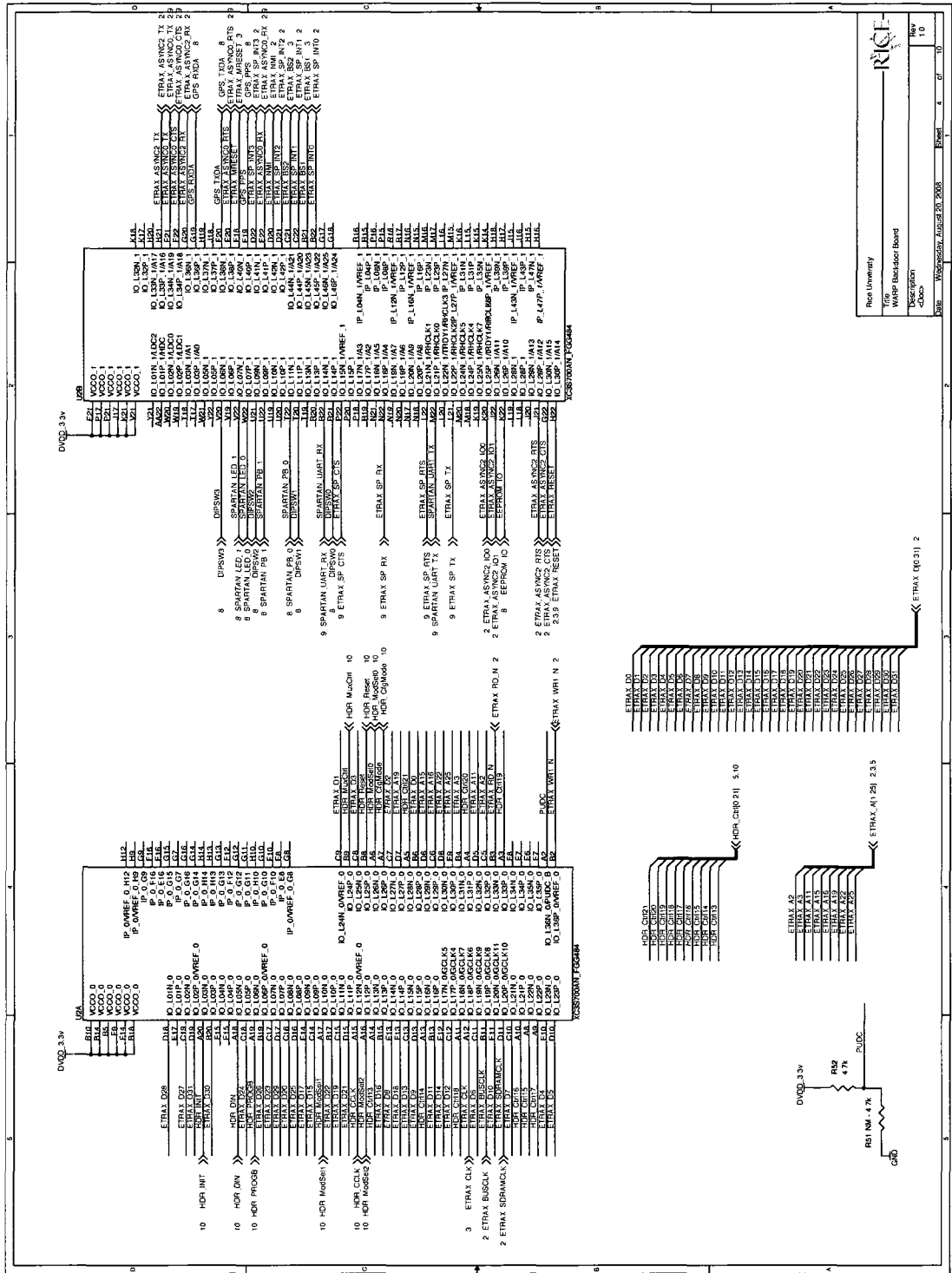


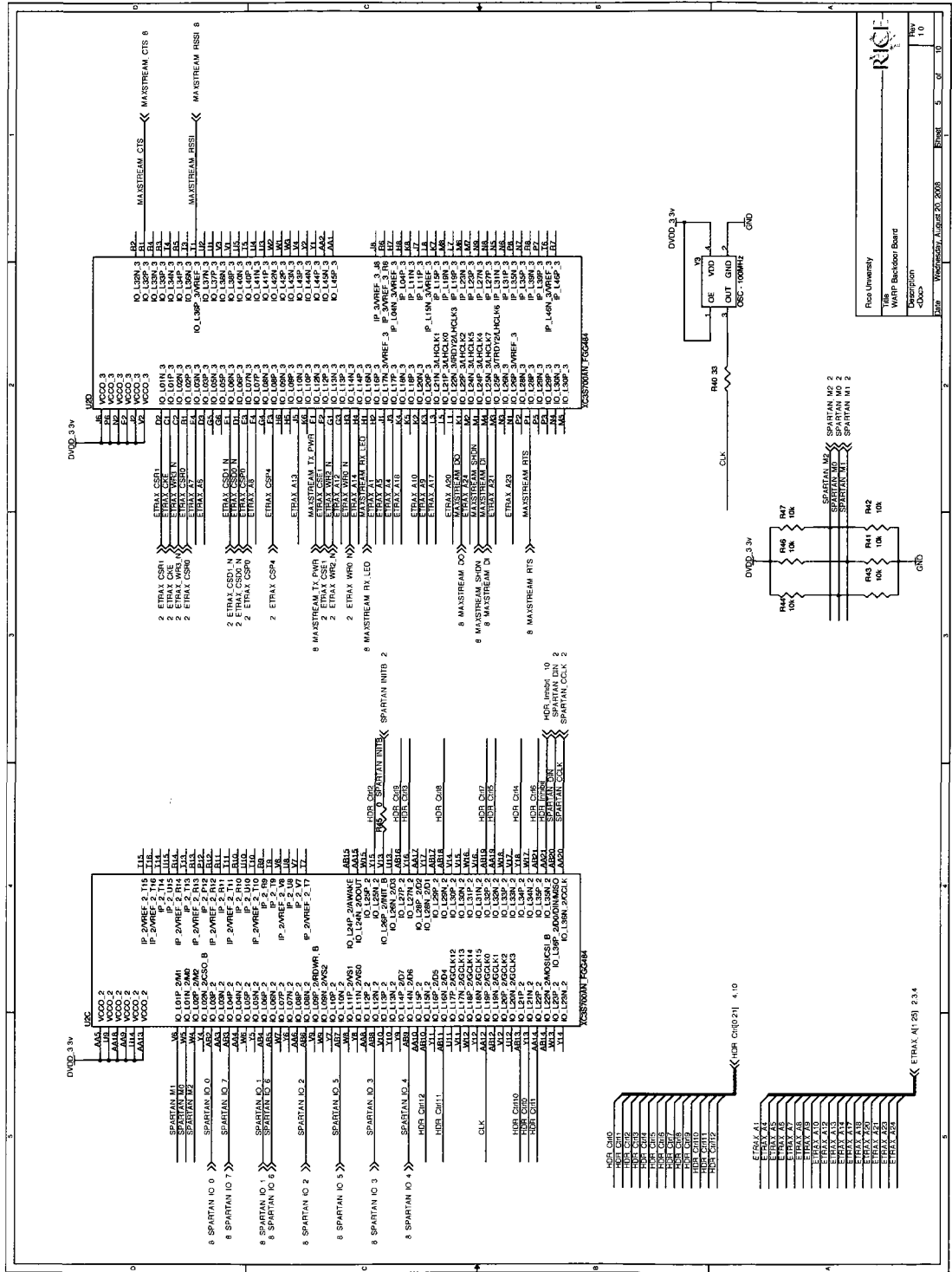
RICE  
Rice University  
File: WARP Backdoor Board  
Description: WARP  
Date: Wednesday, August 20, 2008 8:28 AM  
Page: 78 of 10



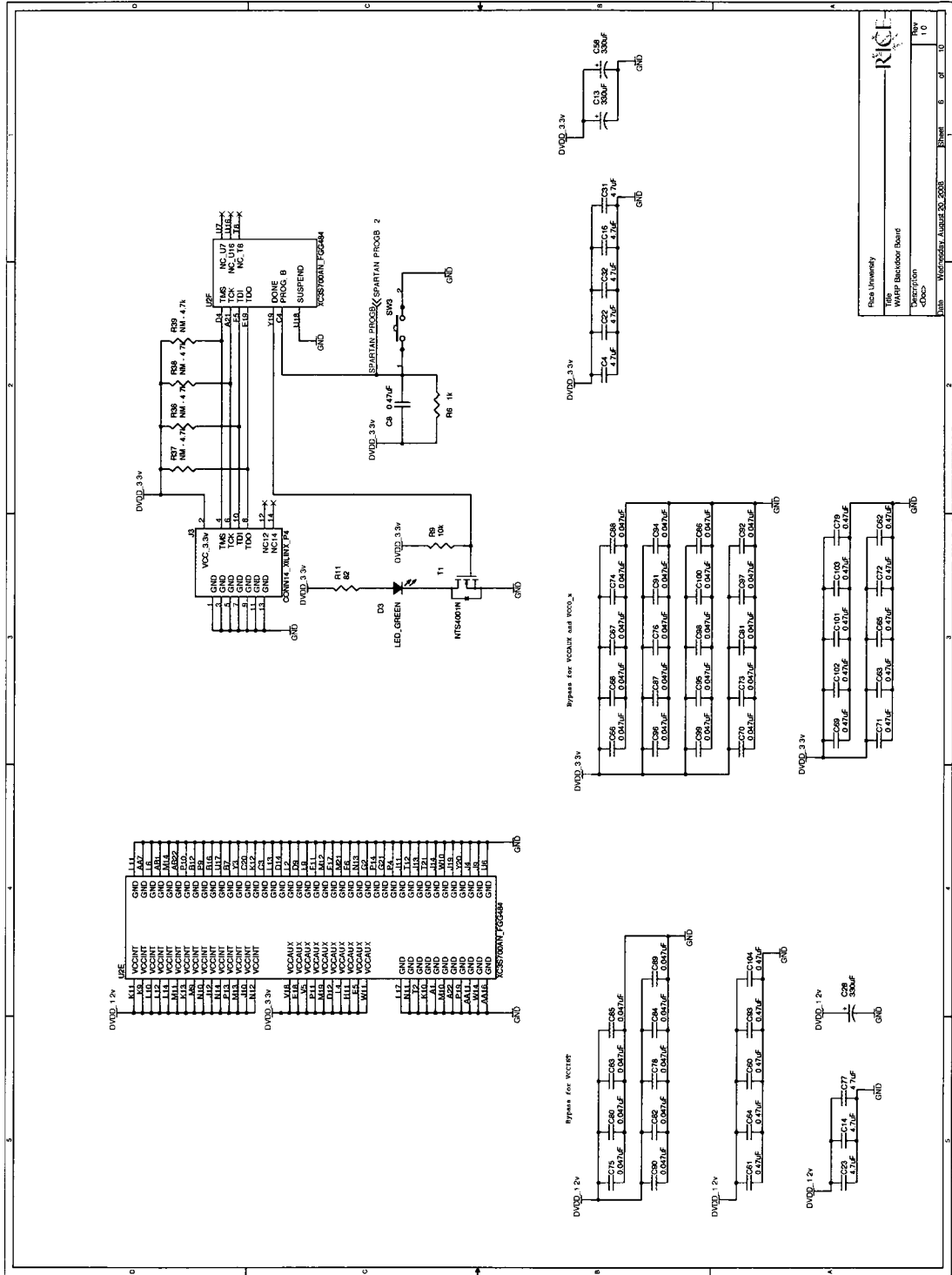


RICE  
 Rice University  
 Title: WARP Backdoor Board  
 Description: Coob  
 Date: Wednesday, August 30, 2006  
 Sheet: 3 of 10  
 Rev: 1.0



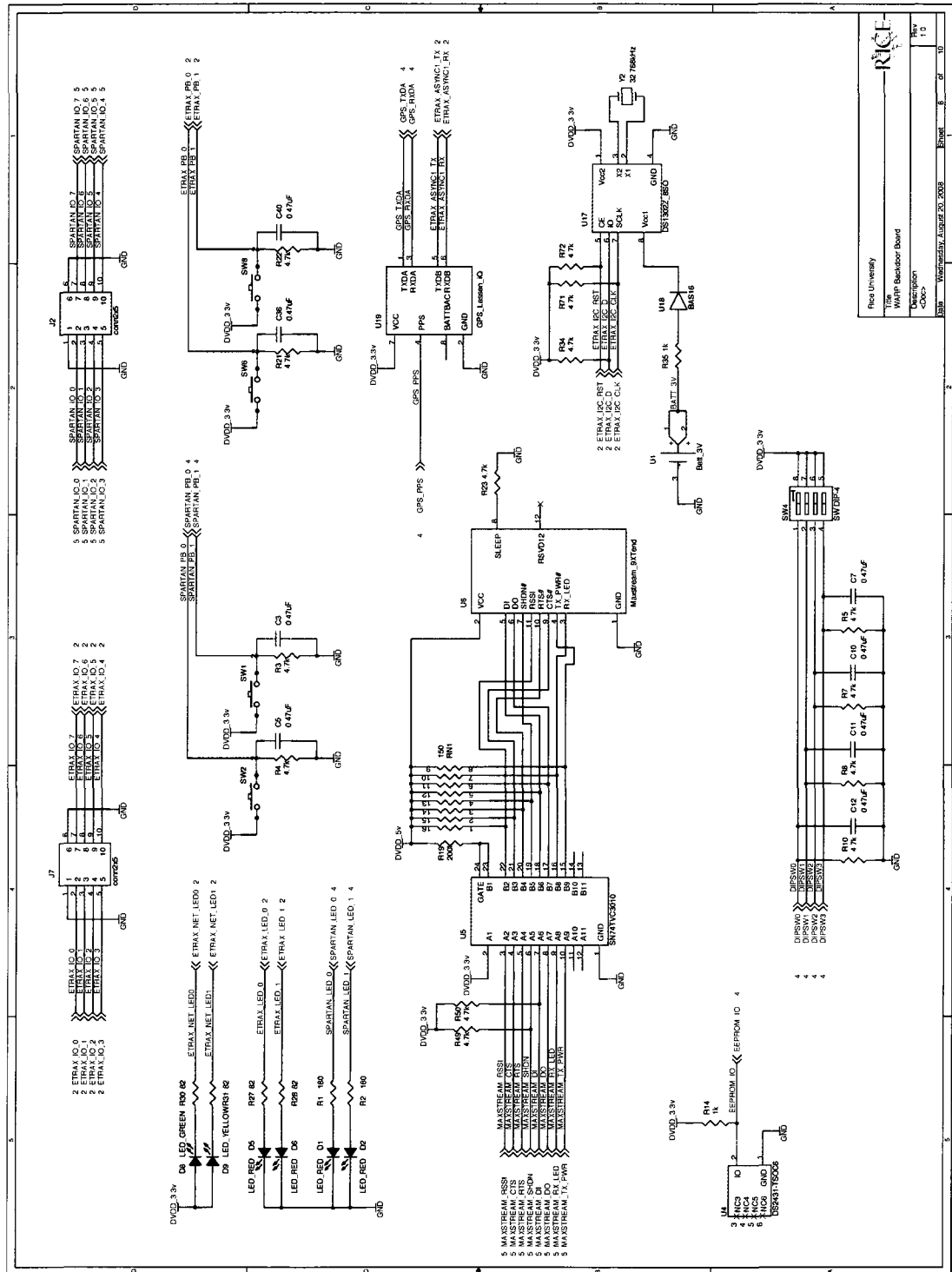


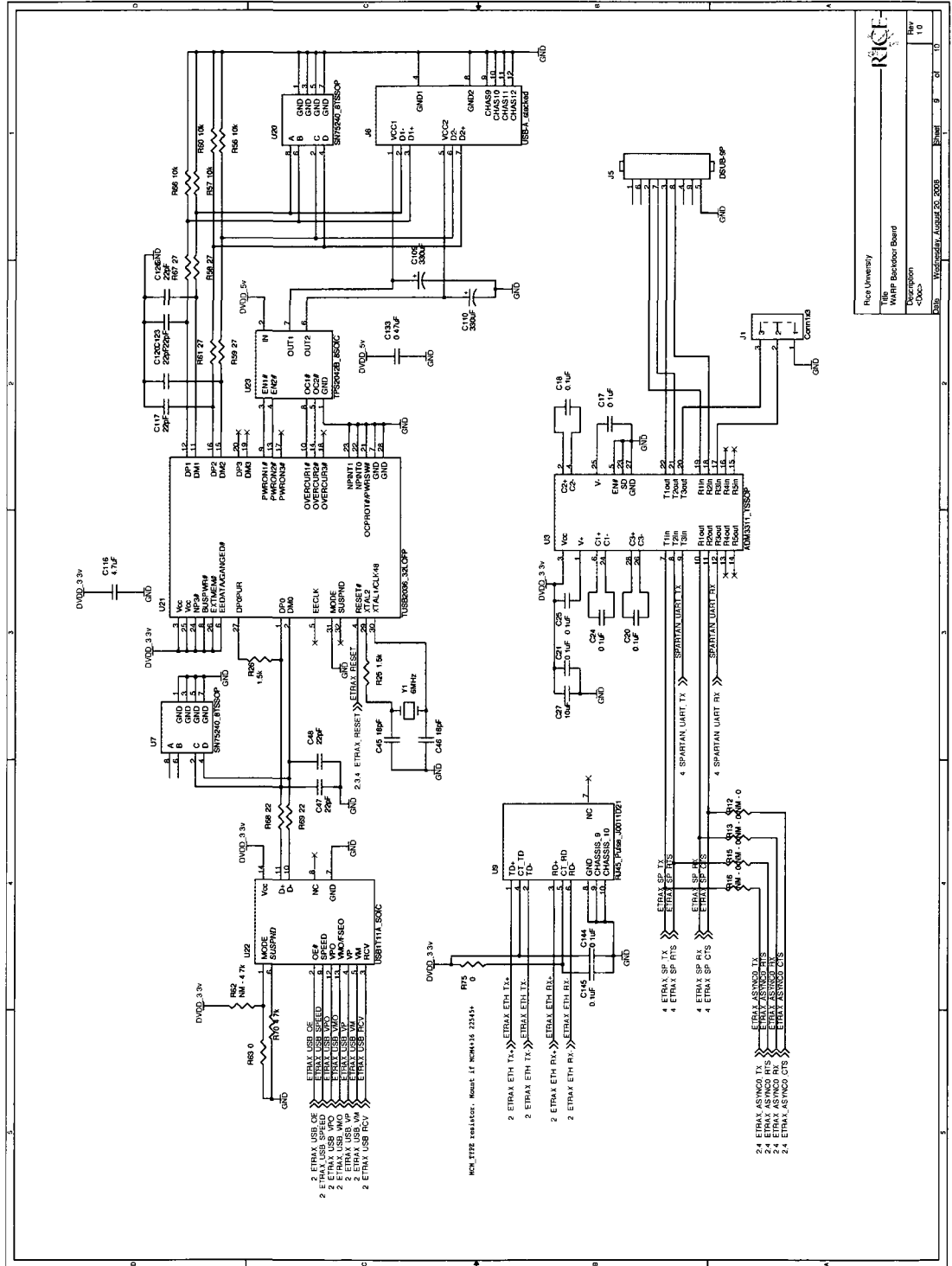
Rice University
   
 Title: WARP Backdoor Board
   
 Description: -Doc-
   
 Date: Wednesday, August 20, 2008
   
 Sheet: 5 of 10
   
 Rev: 1.0



Rev	1.0
Description	WARP Backdoor Board
Author	RC
Date	Wednesday, August 30, 2006
Sheet	6 of 10







Rice University  
WARP Backdoor Board  
Description  
Date: Wed, 14 Aug 2008 09:21:10  
Rev: 1.0





## Layout

---

Included in the next few pages are the layout gerbers for both the FPGA Board and the Backdoor Board. The layout was done in Cadence Allegro 16.01. In addition to the copper layers, also attached are the solder mask, the solder paste and silkscreen layers for both top and bottom. The stackup for each board is shown prior to the layout gerbers.

### **B.1 FPGA Board**

Stackup and layout gerbers for the FPGA Board.

Impedance Control 50Ω

Layer 1 (Top)

Thickness

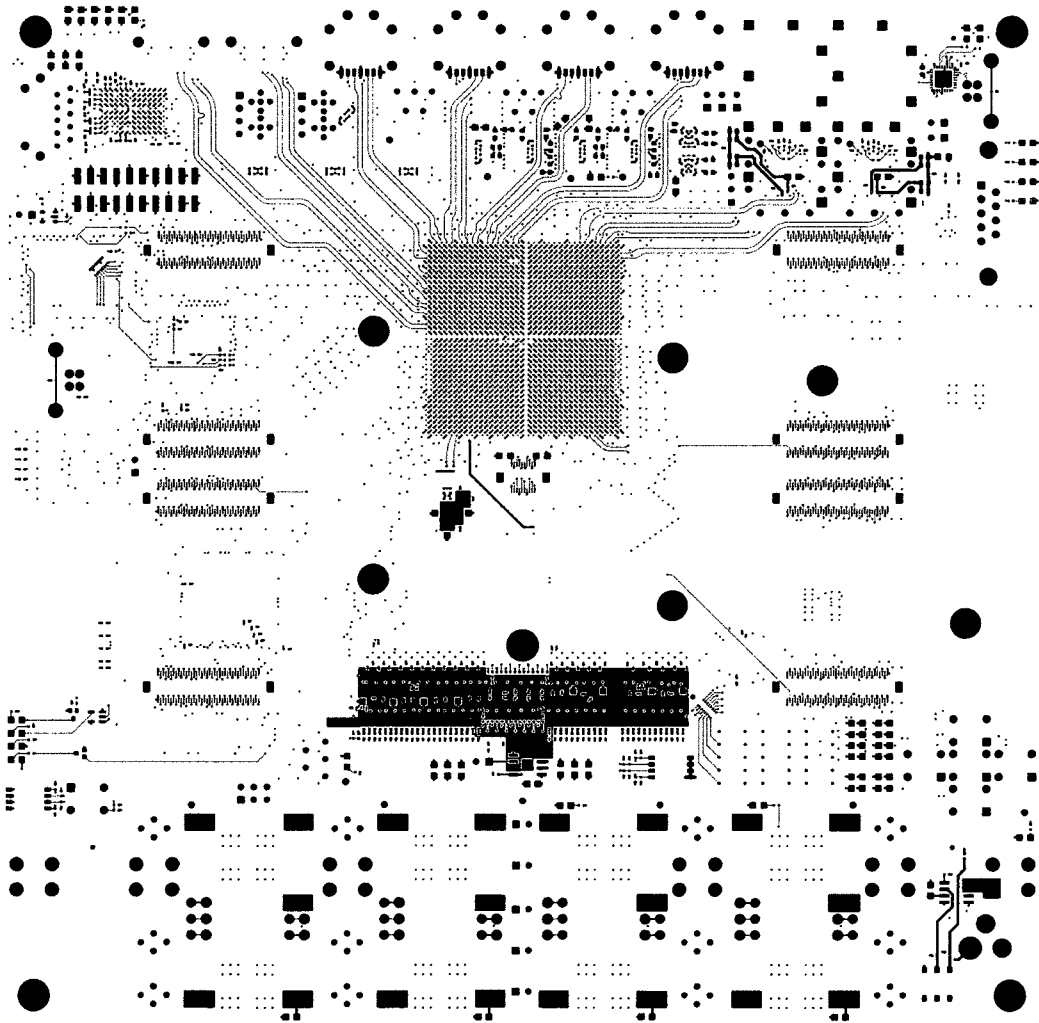
Weight

Layer 1 - Top	Signal	0.6mils	0.5oz
IT180	Dielectric	4.2mils	
Layer 2 - PG1	Plane	0.6mils	0.5oz
IT180	Dielectric	5mils	Core
Layer 3 - INT1	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	
Layer 4 - INT2	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	Core
Layer 5 - PG2	Plane	0.6mils	0.5oz
IT180	Dielectric	5mils	
Layer 6 - INT3	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	Core
Layer 7 - INT4	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	
Layer 8 - PG3	Plane	0.6mils	0.5oz
IT180	Dielectric	8mils	Core
Layer 9 - PG4	Plane	0.6mils	0.5oz
IT180	Dielectric	5mils	
Layer 10 - PG5	Plane	0.6mils	0.5oz
IT180	Dielectric	8mils	Core
Layer 11 - PG6	Plane	0.6mils	0.5oz
IT180	Dielectric	5mils	
Layer 12 - INT5	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	Core
Layer 13 - INT6	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	
Layer 14 - PG7	Plane	0.6mils	0.5oz
IT180	Dielectric	5mils	Core
Layer 15 - INT7	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	
Layer 16 - INT8	Signal	0.6mils	0.5oz
IT180	Dielectric	5mils	Core
Layer 17 - PG8	Plane	0.6mils	0.5oz
IT180	Dielectric	4.2mils	
Layer 18 - Bottom	Signal	0.6mils	0.5oz

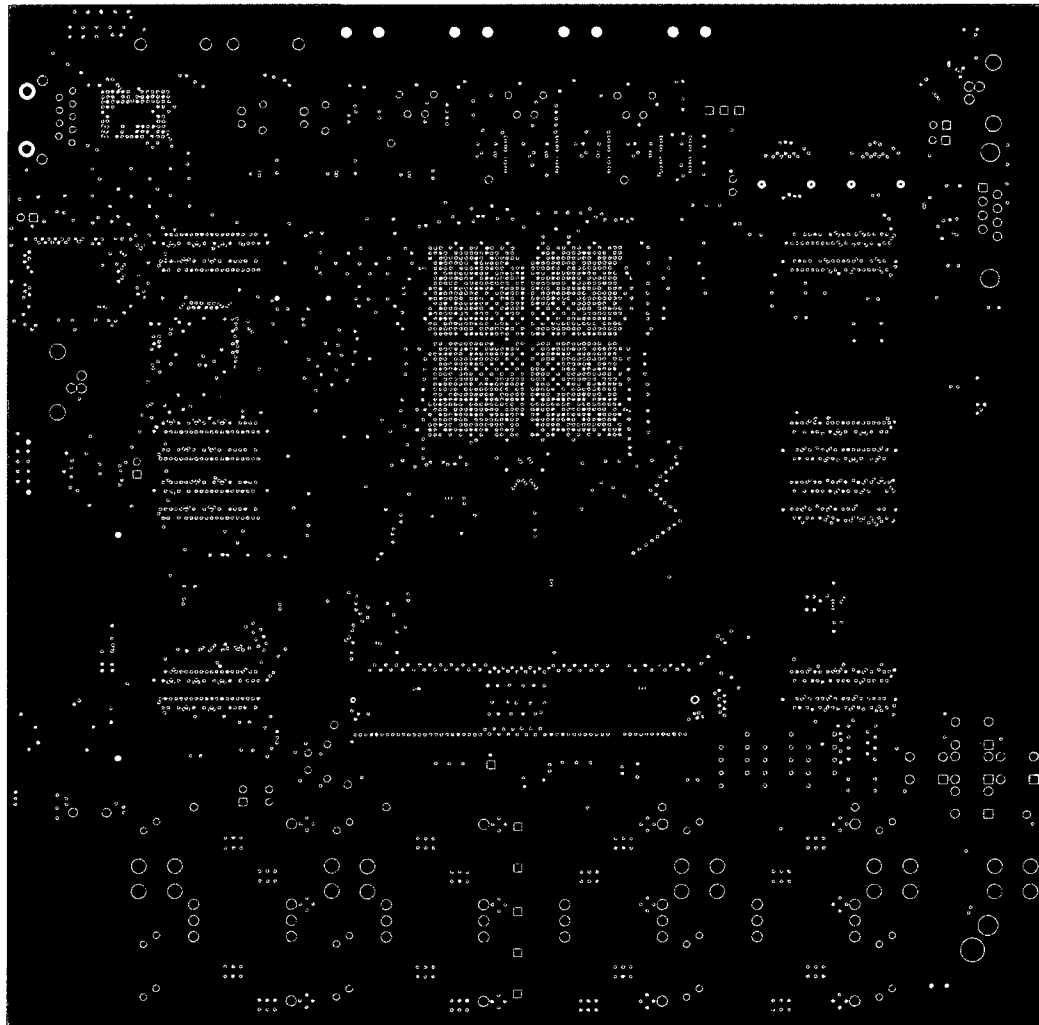
Impedance Control 50Ω

**100.2mils**

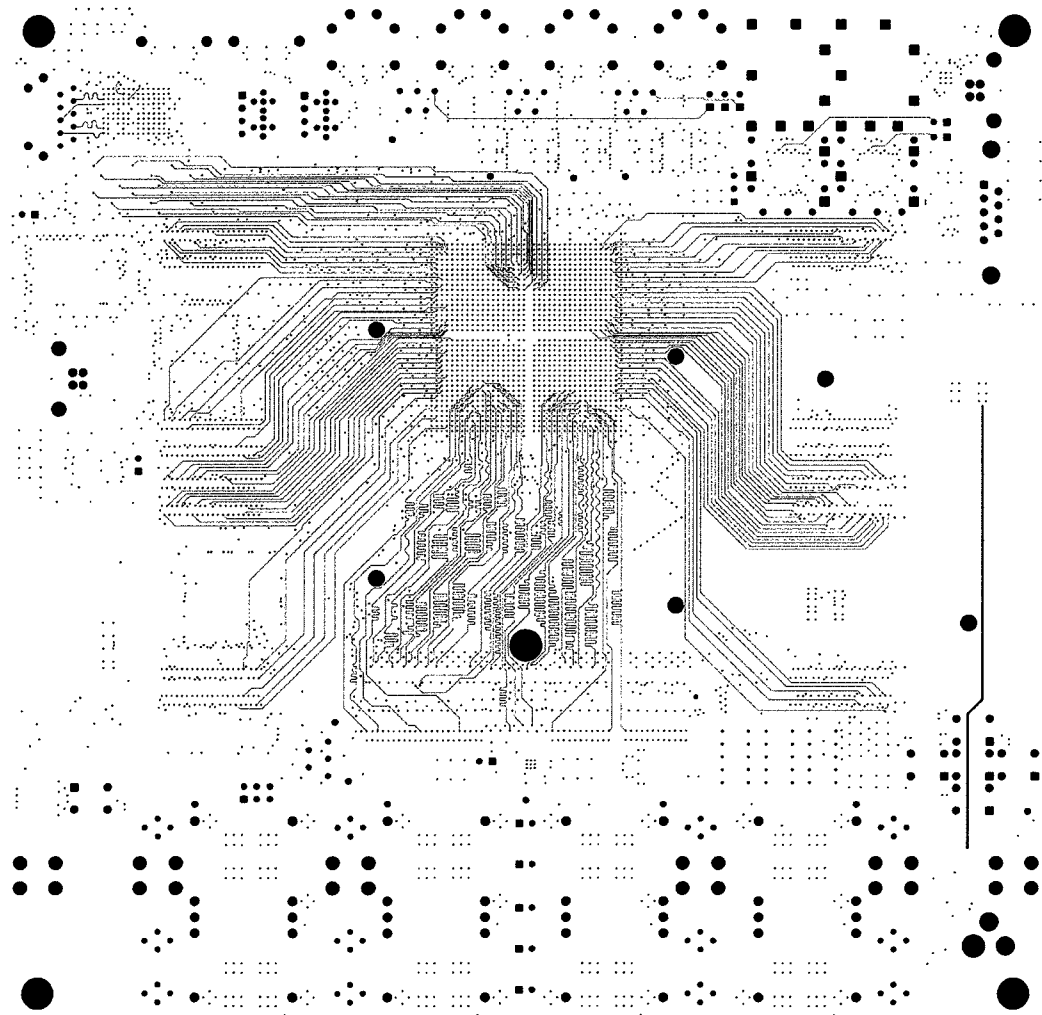
Layer 18 (Bottom)



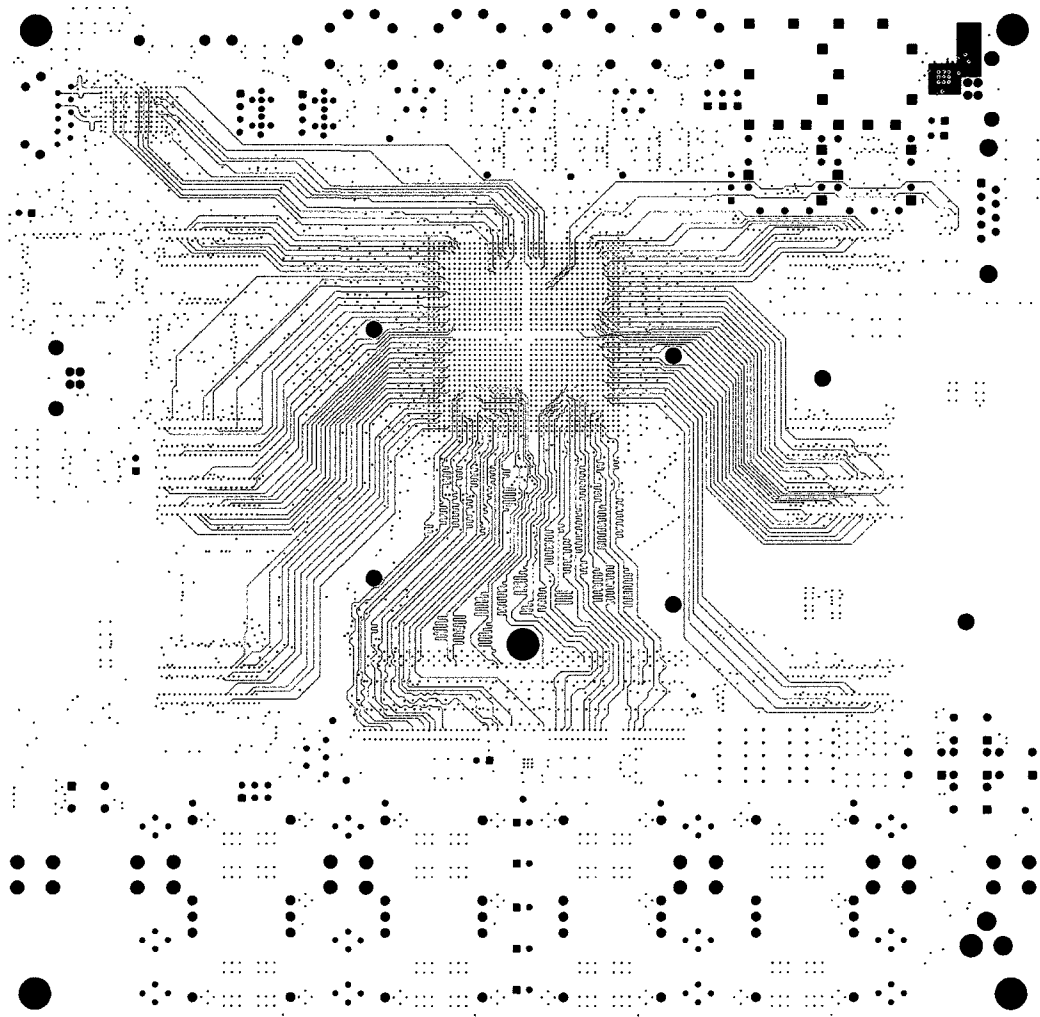
Top (Layer 1) - Signal Layer



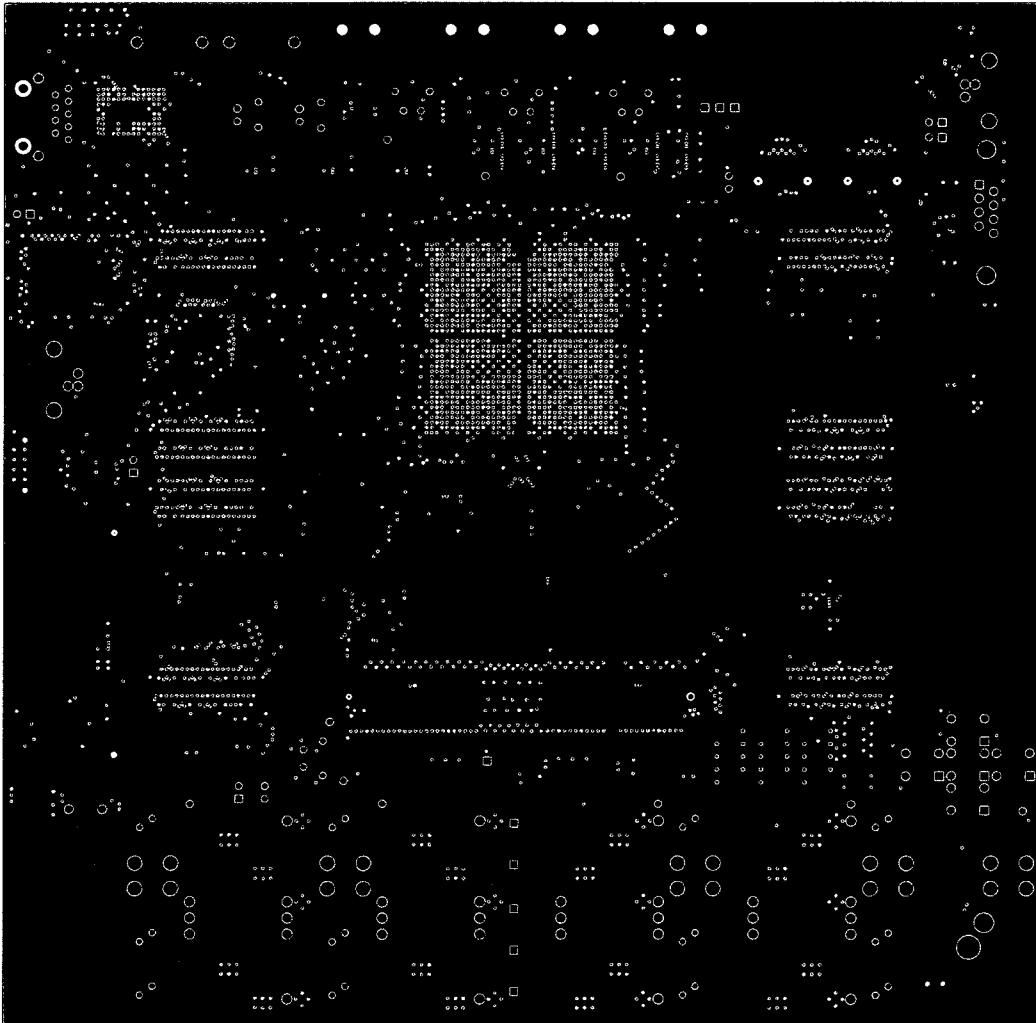
PGI (Layer 2) - Ground Plane



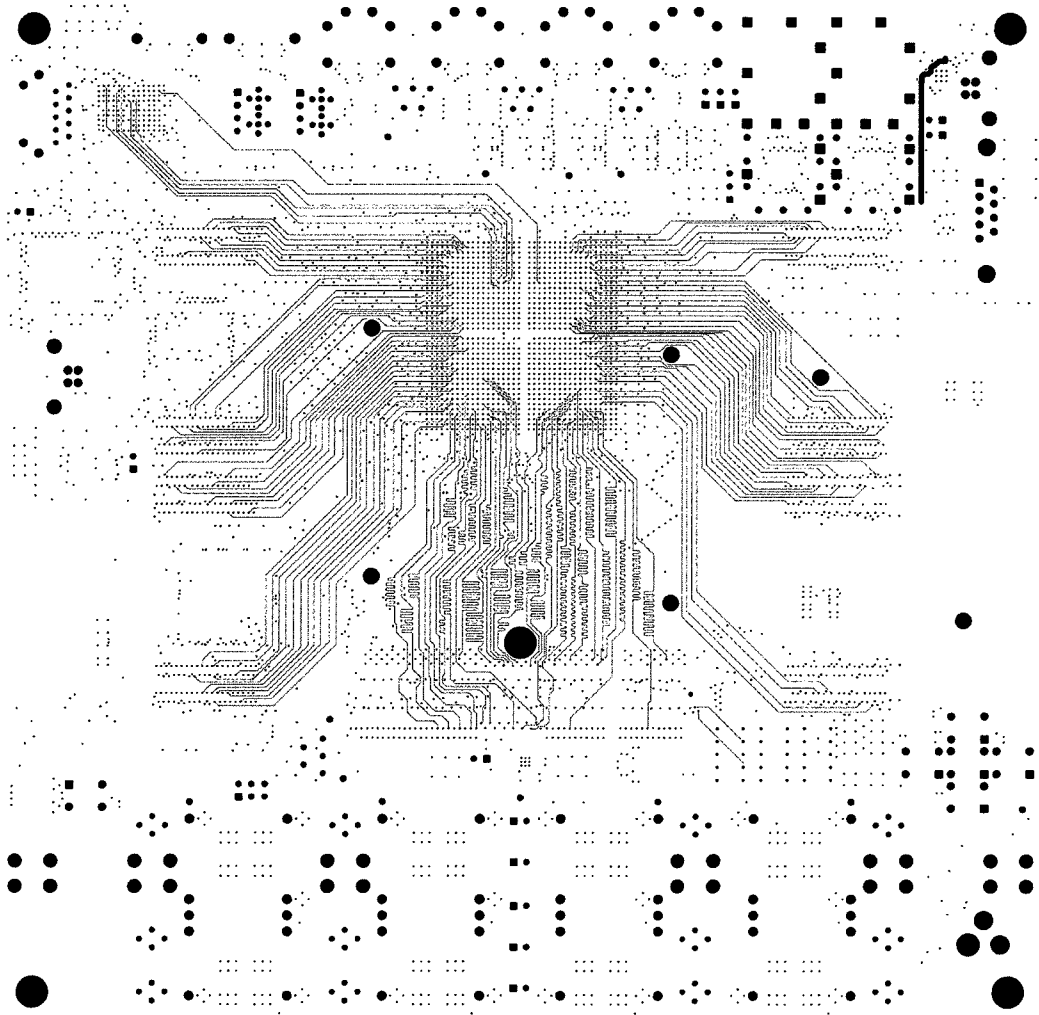
INTI (Layer 3) - Signal Layer



INT2 (Layer 4) - Signal Layer

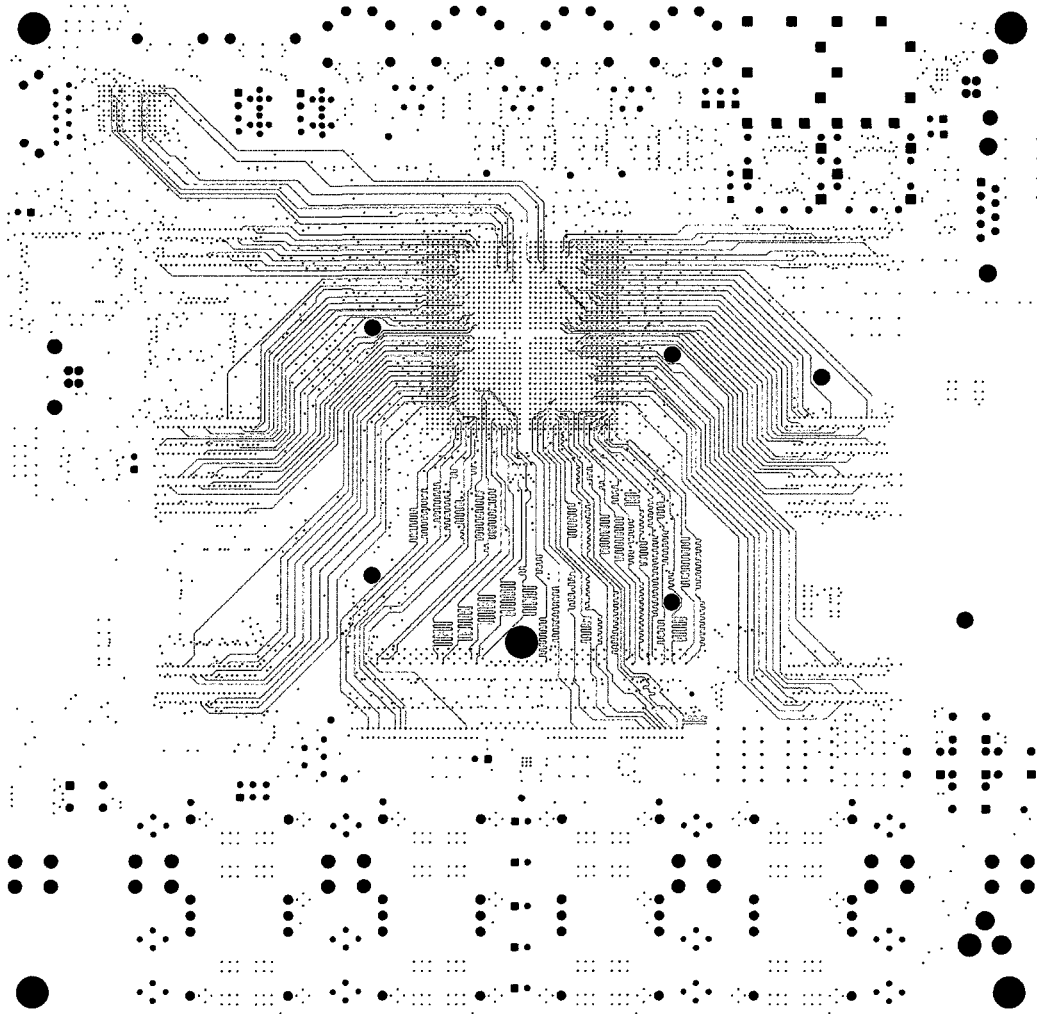


PG2 (Layer 5) - Ground Plane



INT3 (Layer 6) - Signal Layer

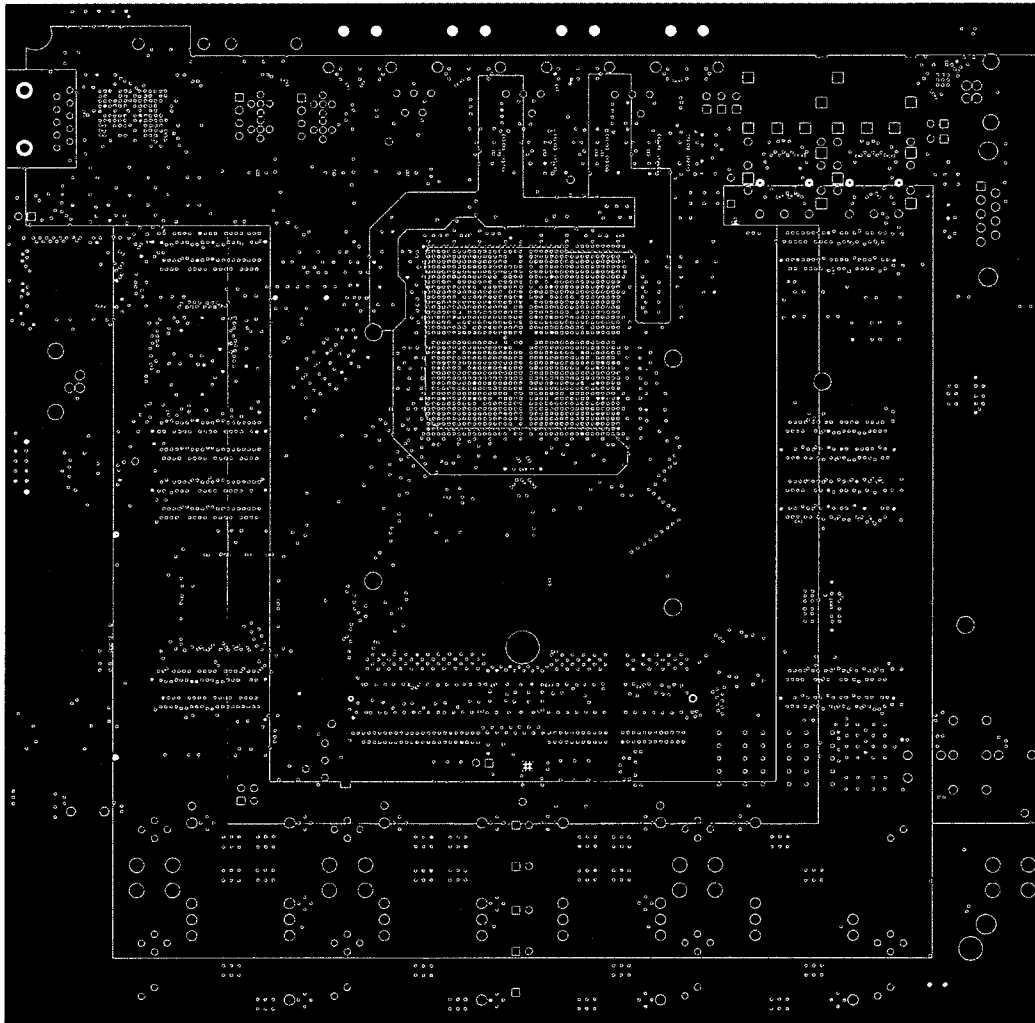




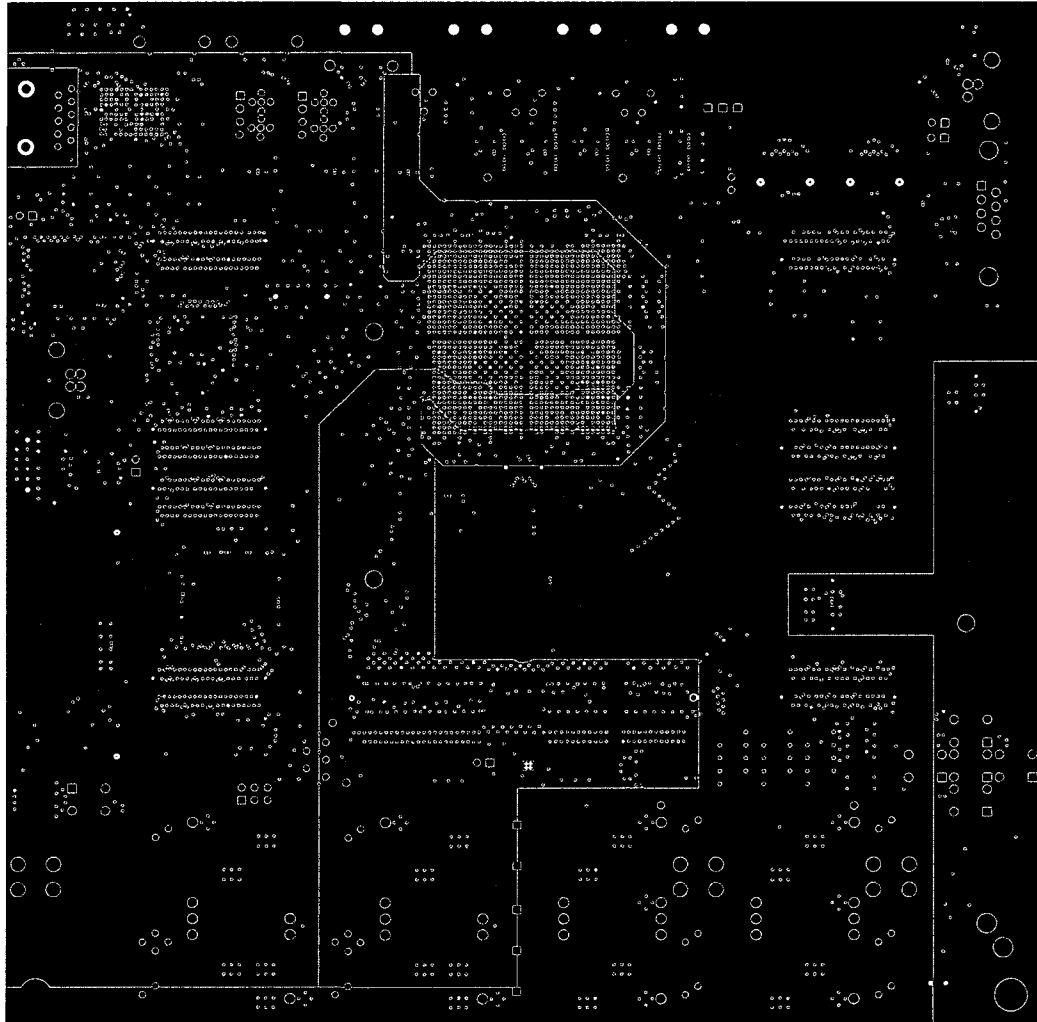
INT4 (Layer 7) - Signal Layer



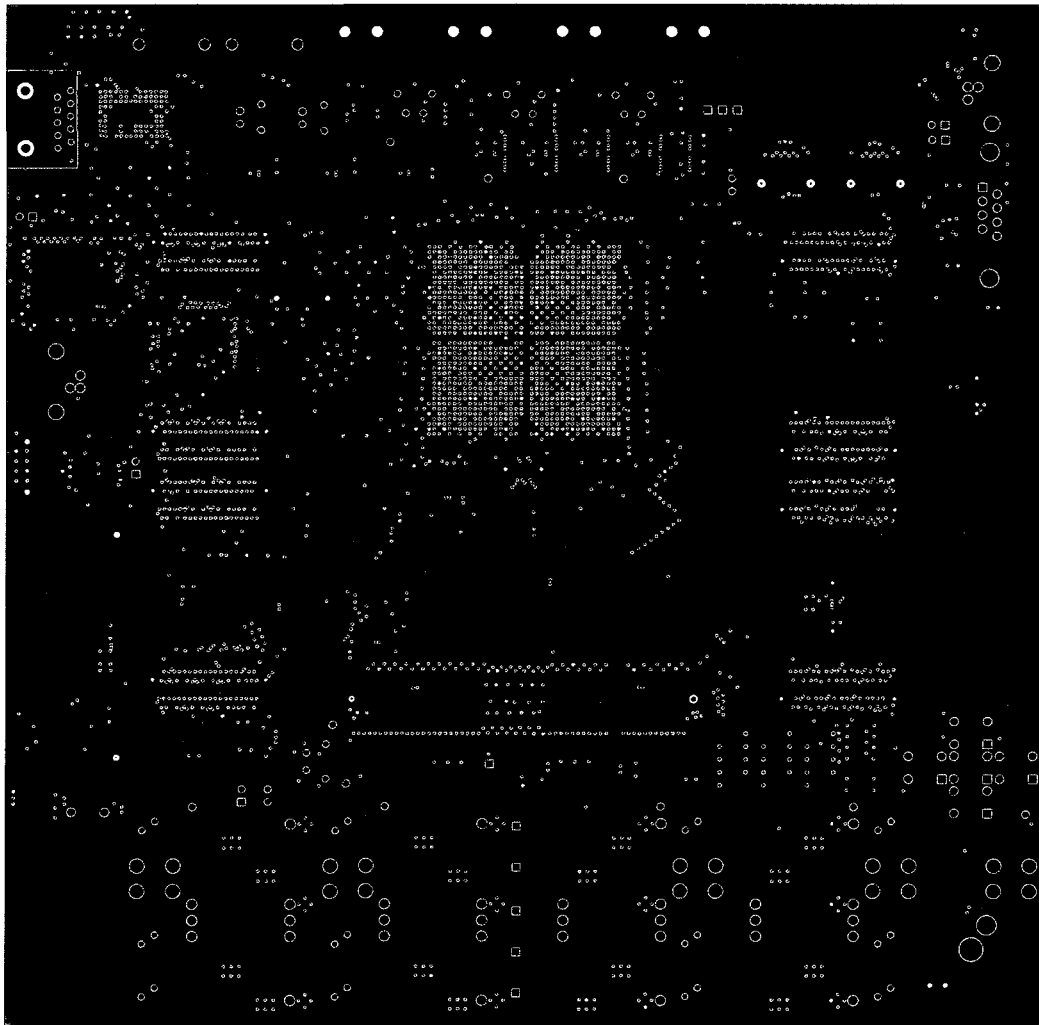
PG3 (Layer 8) - Ground Plane



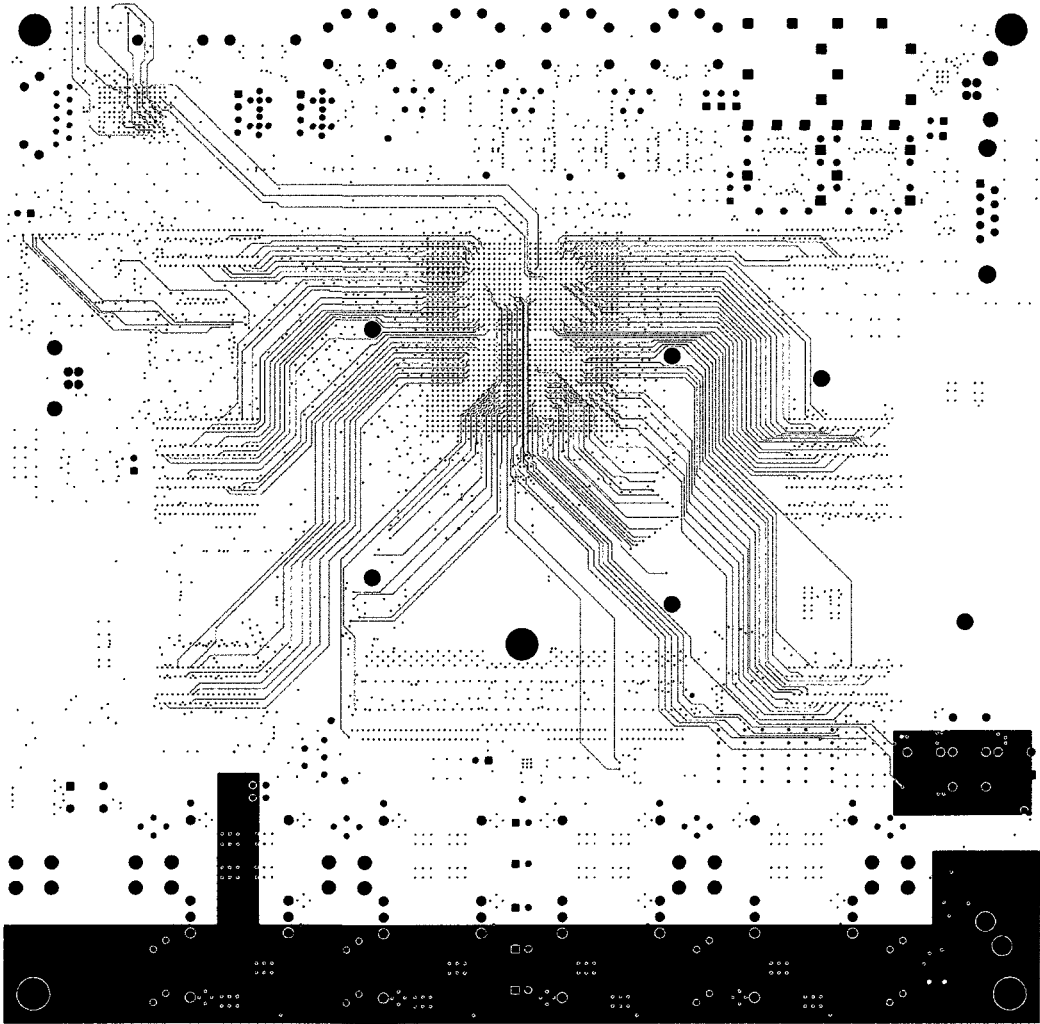
PG4 (Layer 9) - Power Plane



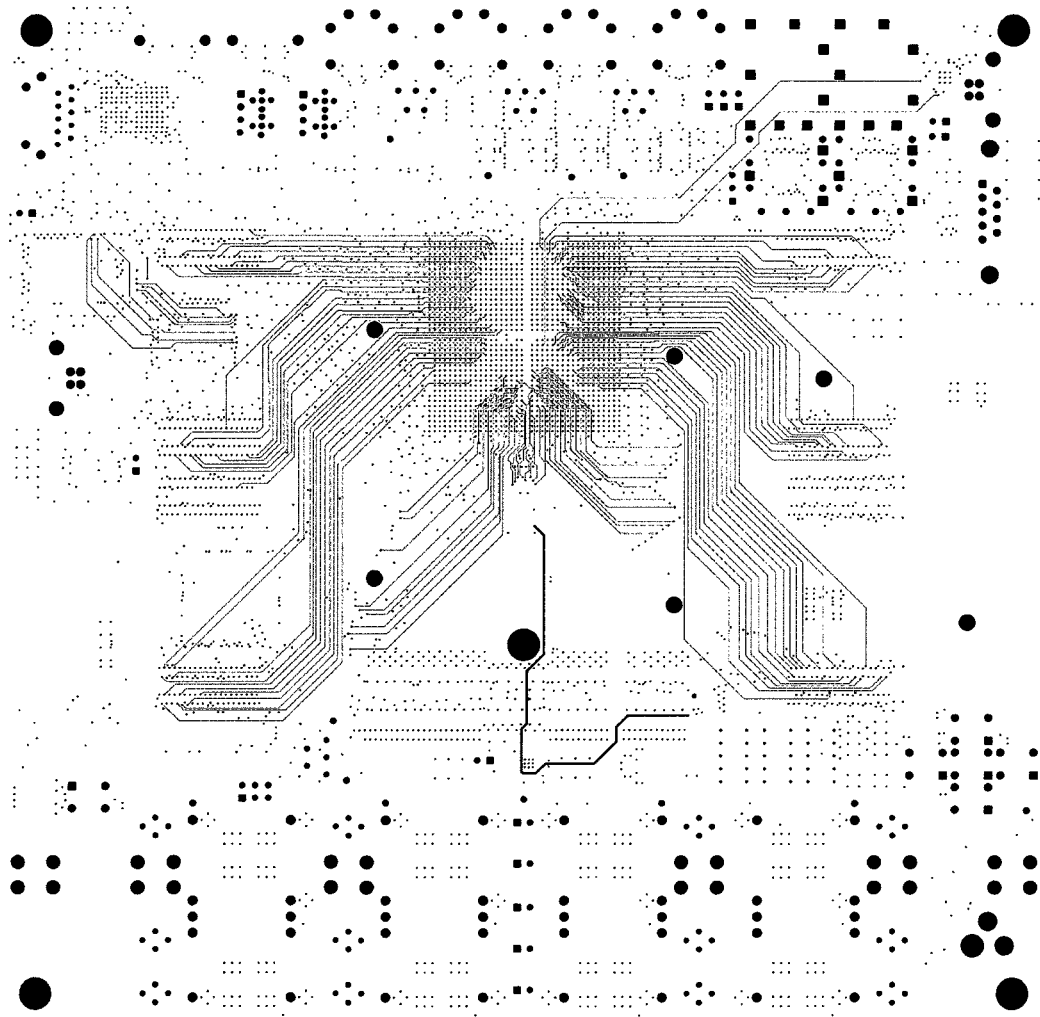
PG5 (Layer 10) - Power Plane



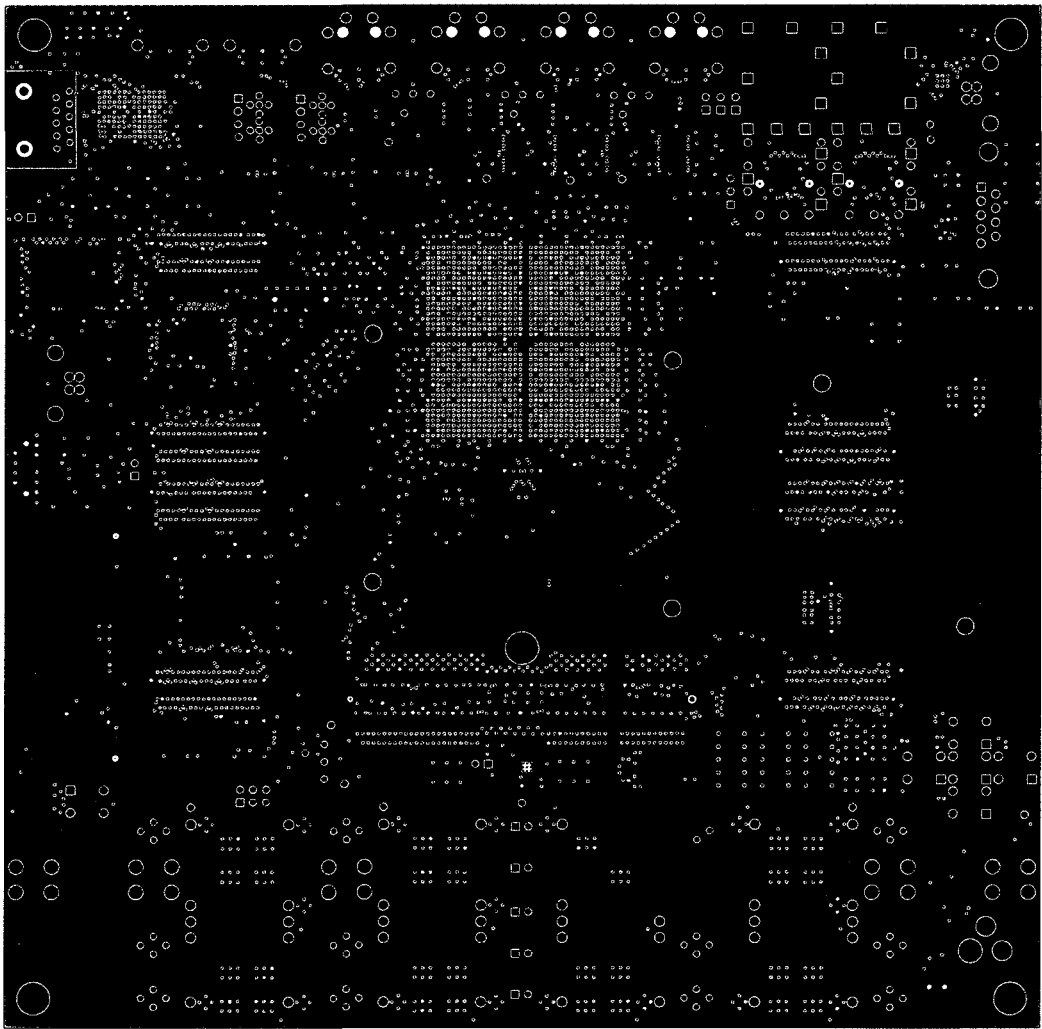
PG6 (Layer 11) - Ground Plane



INT5 (Layer 12) - Signal Layer

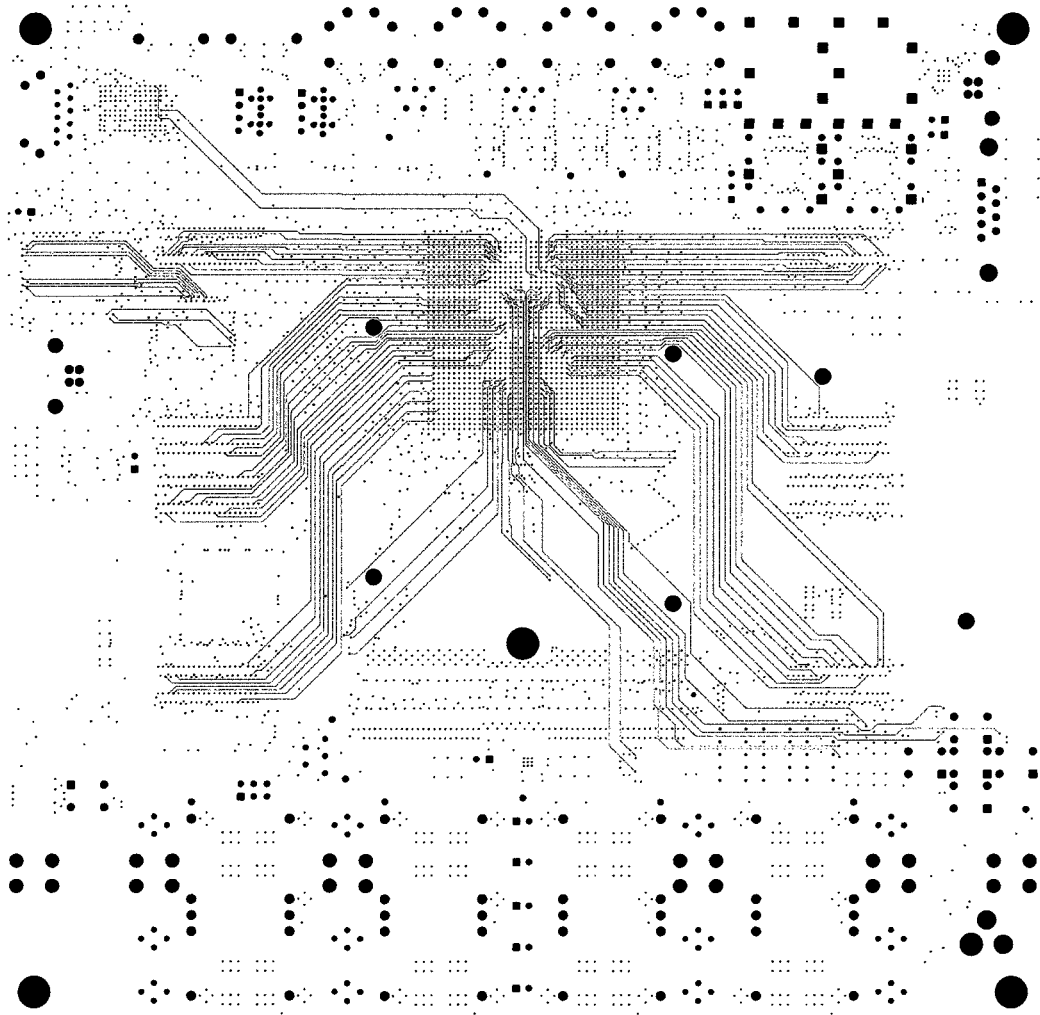


INT6 (Layer 13) - Signal Layer

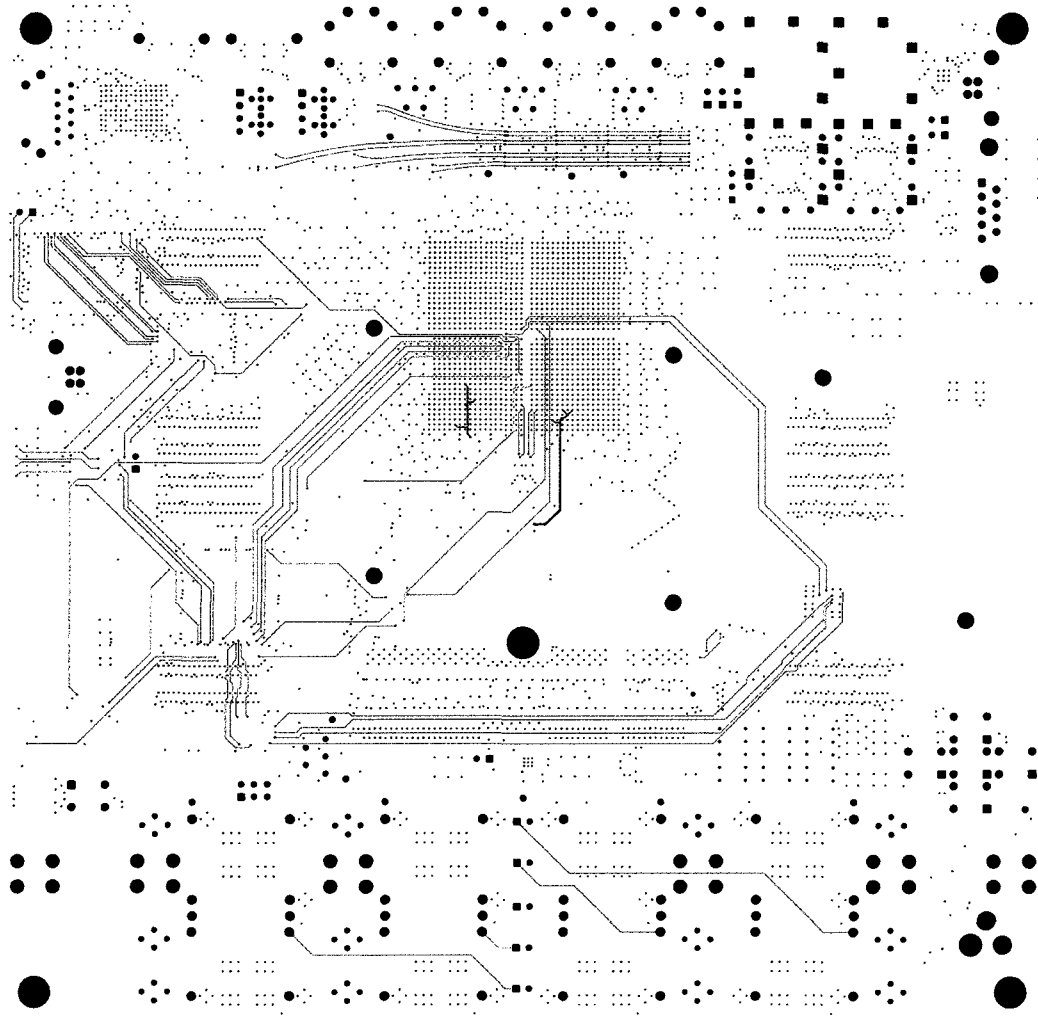


PG7 (Layer 14) - Power Plane

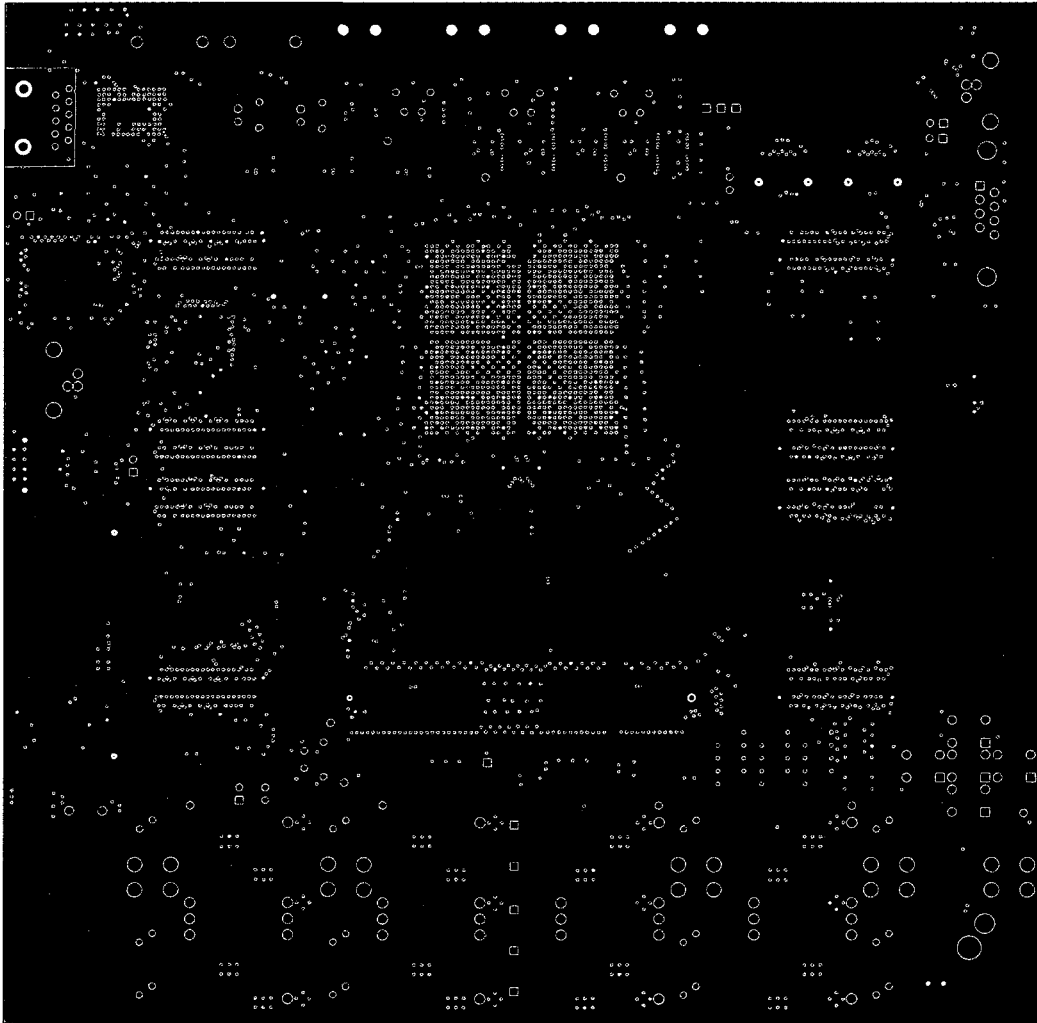




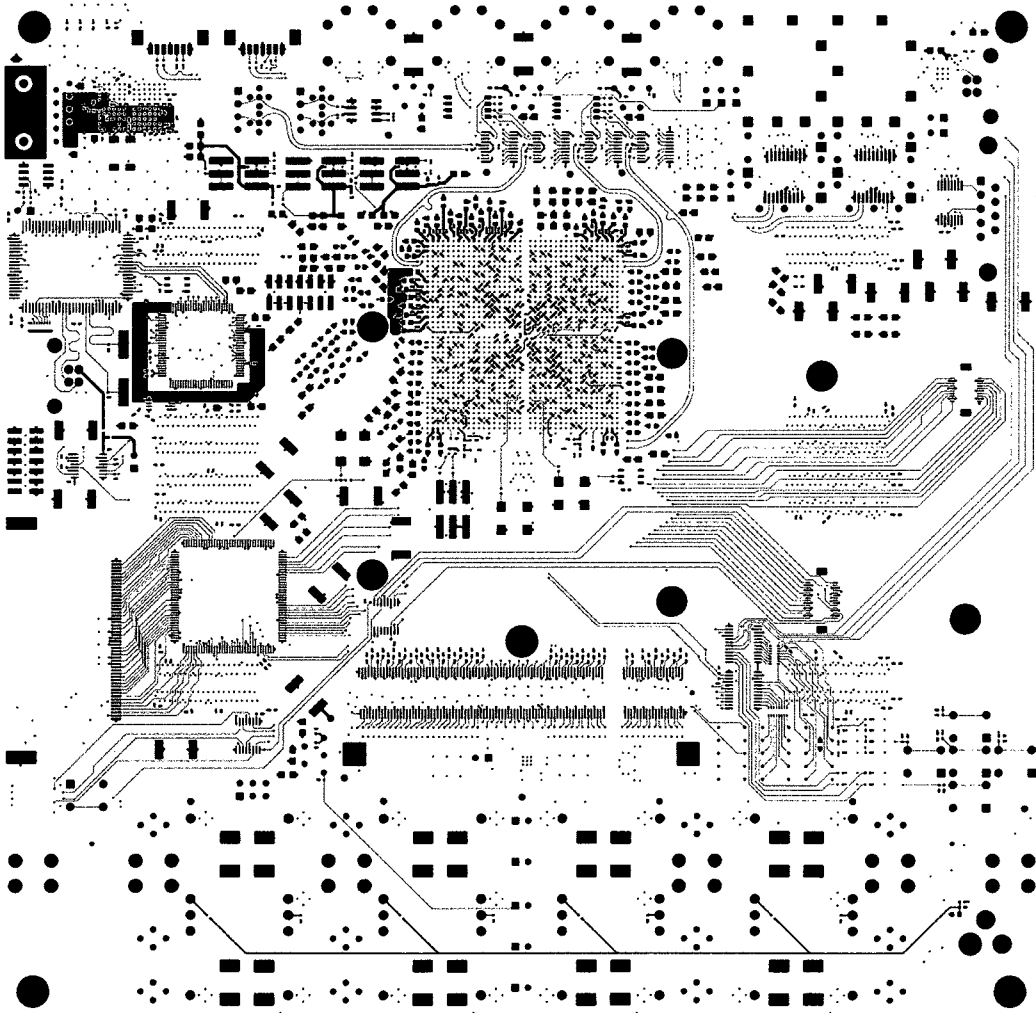
INT7 (Layer 15) - Signal Layer



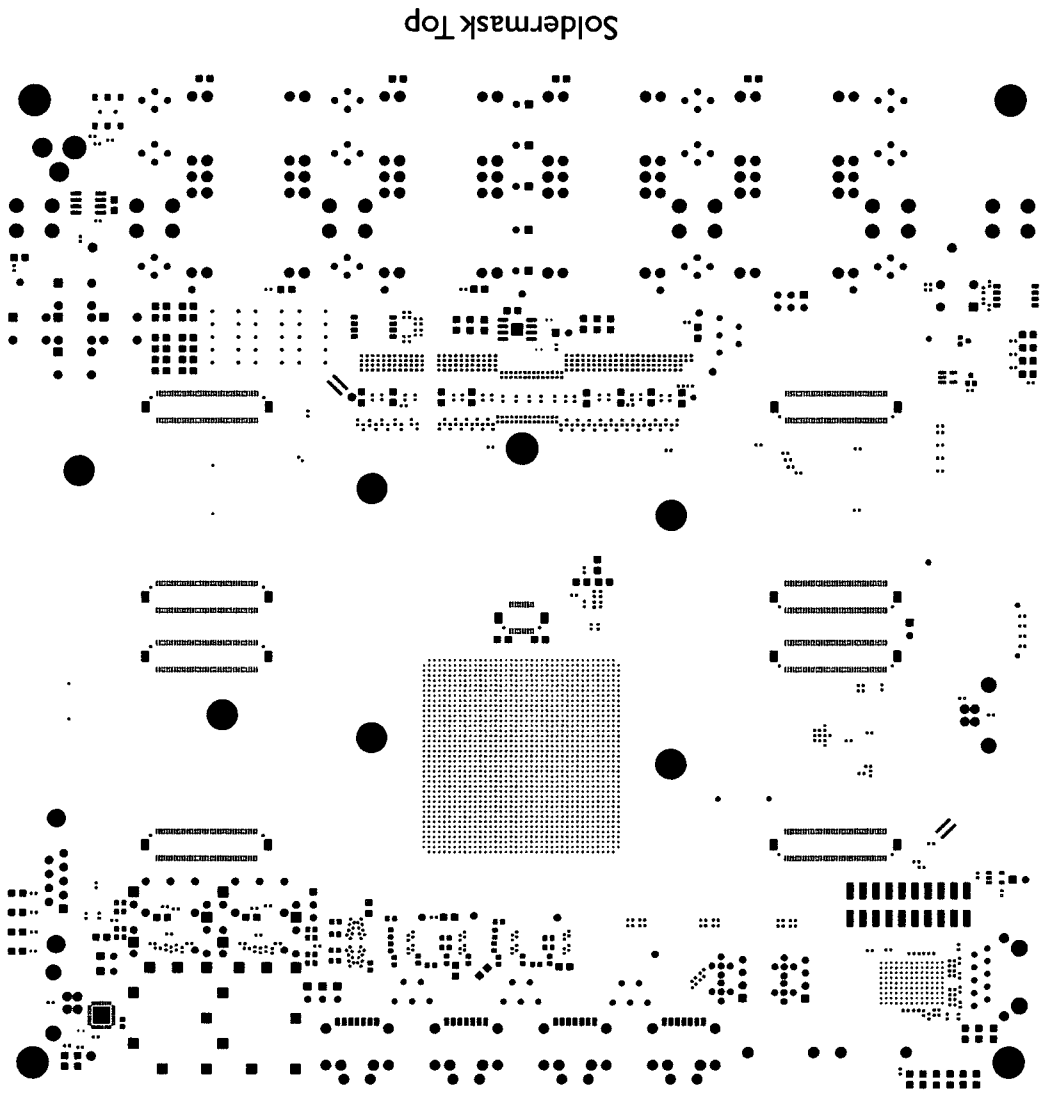
INT8 (Layer 16) - Signal Layer

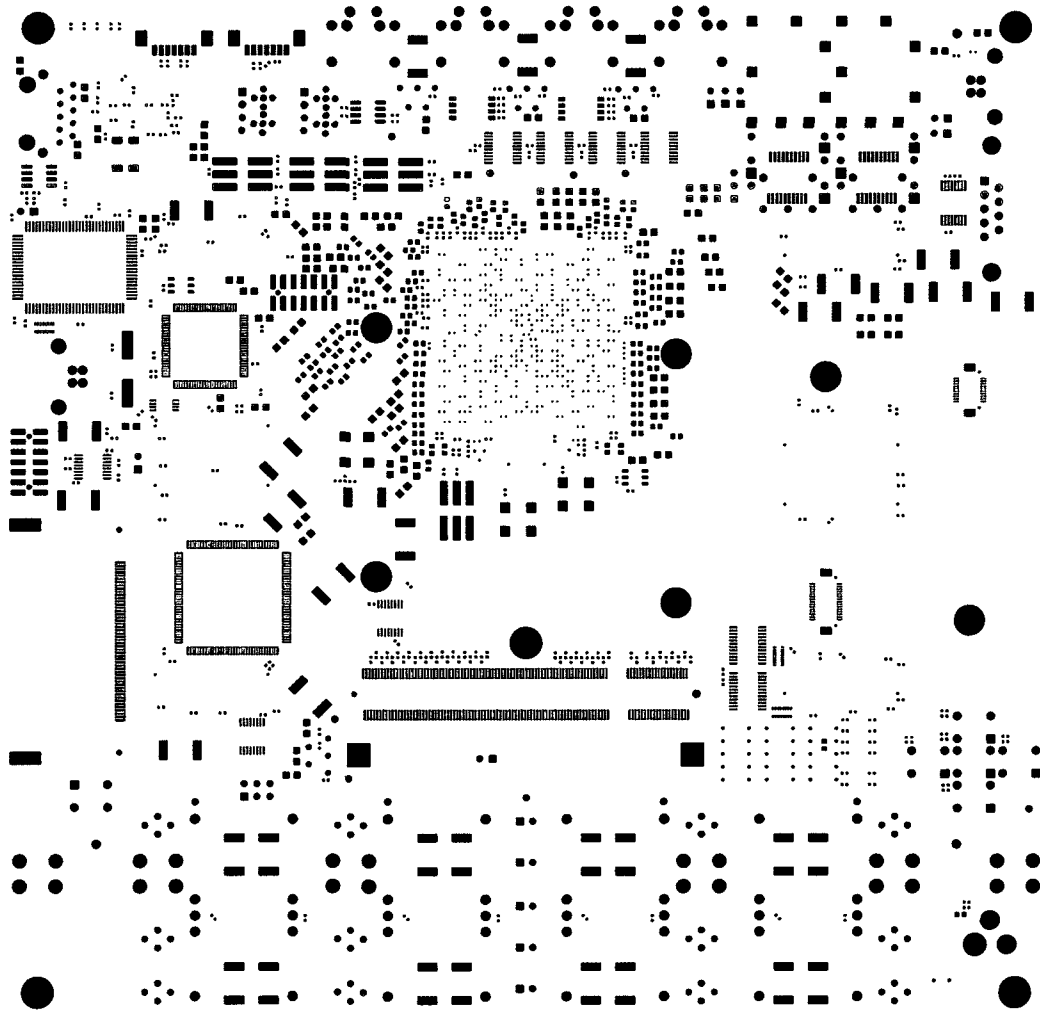


PG8 (Layer 17) - Ground Plane

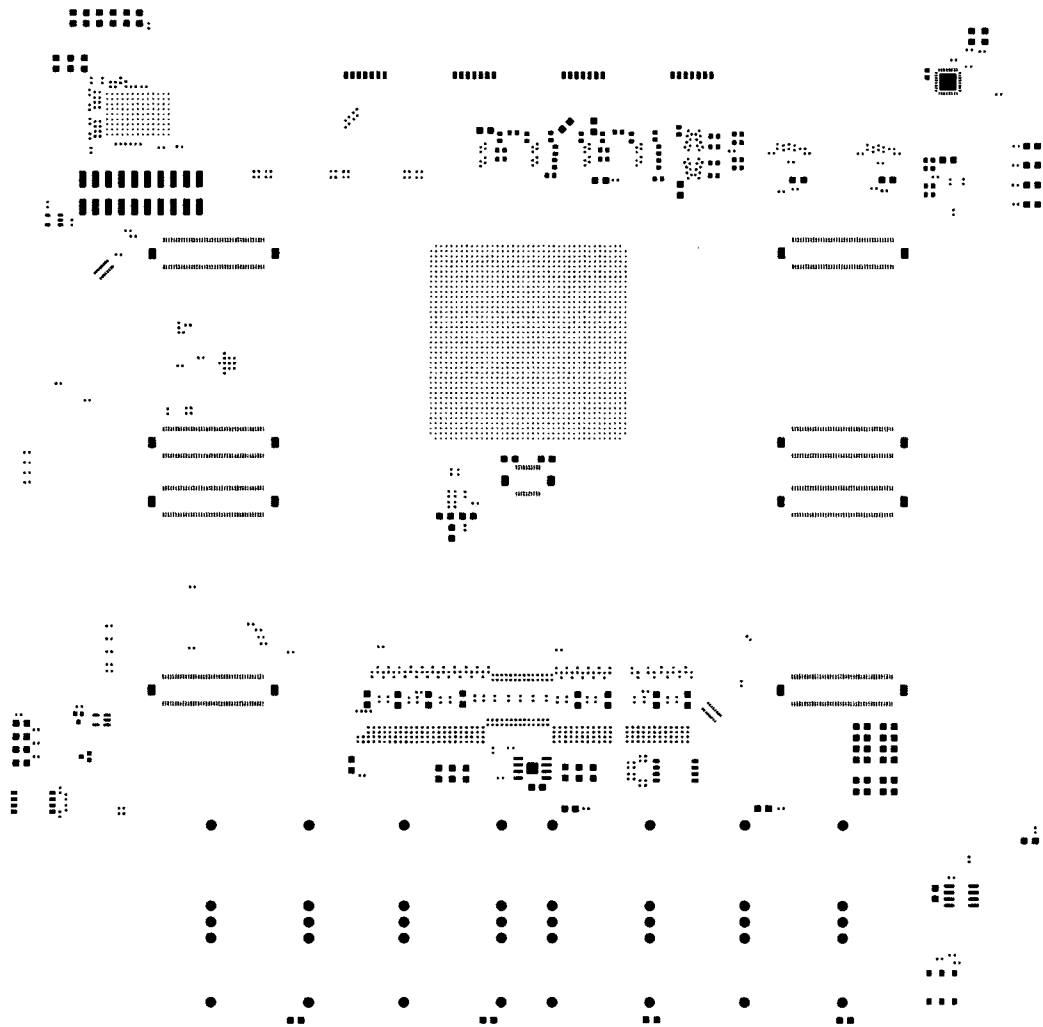


Bottom (Layer 18) - Signal Layer

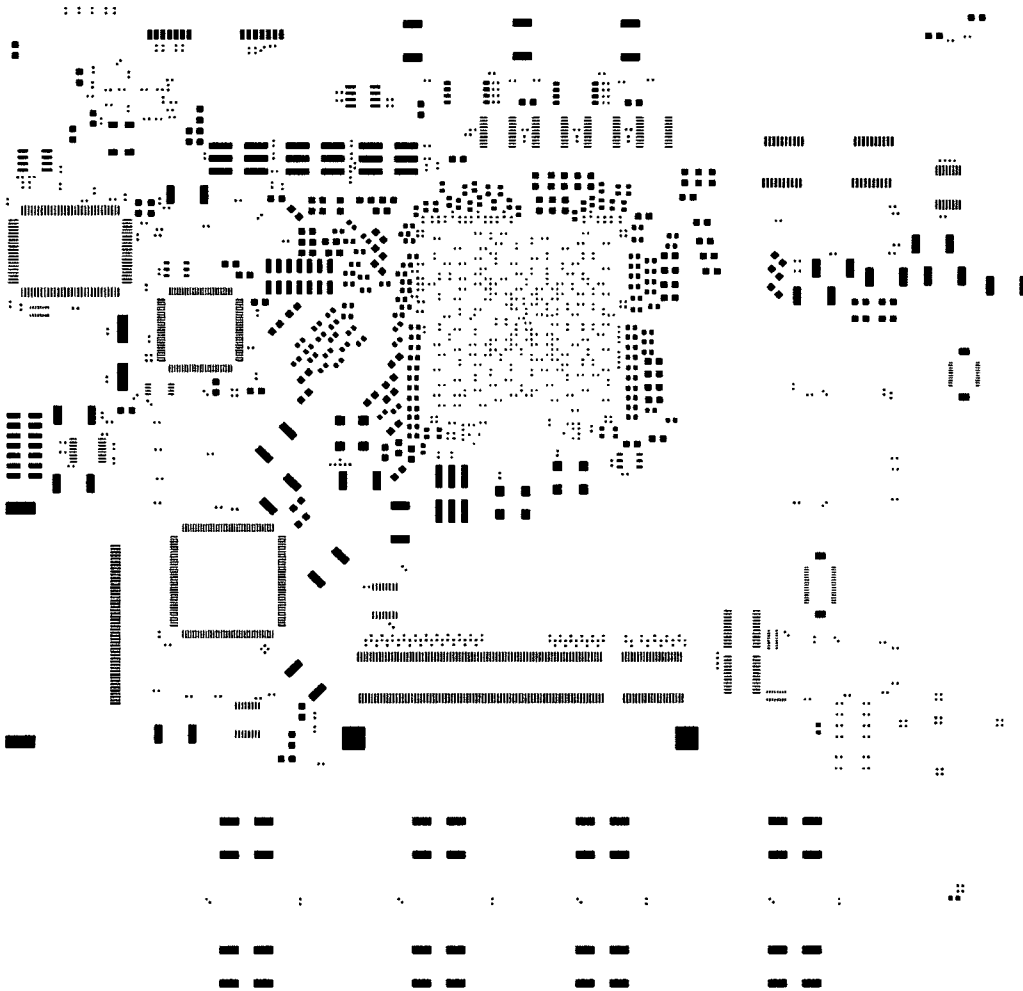




Soldermask Bottom

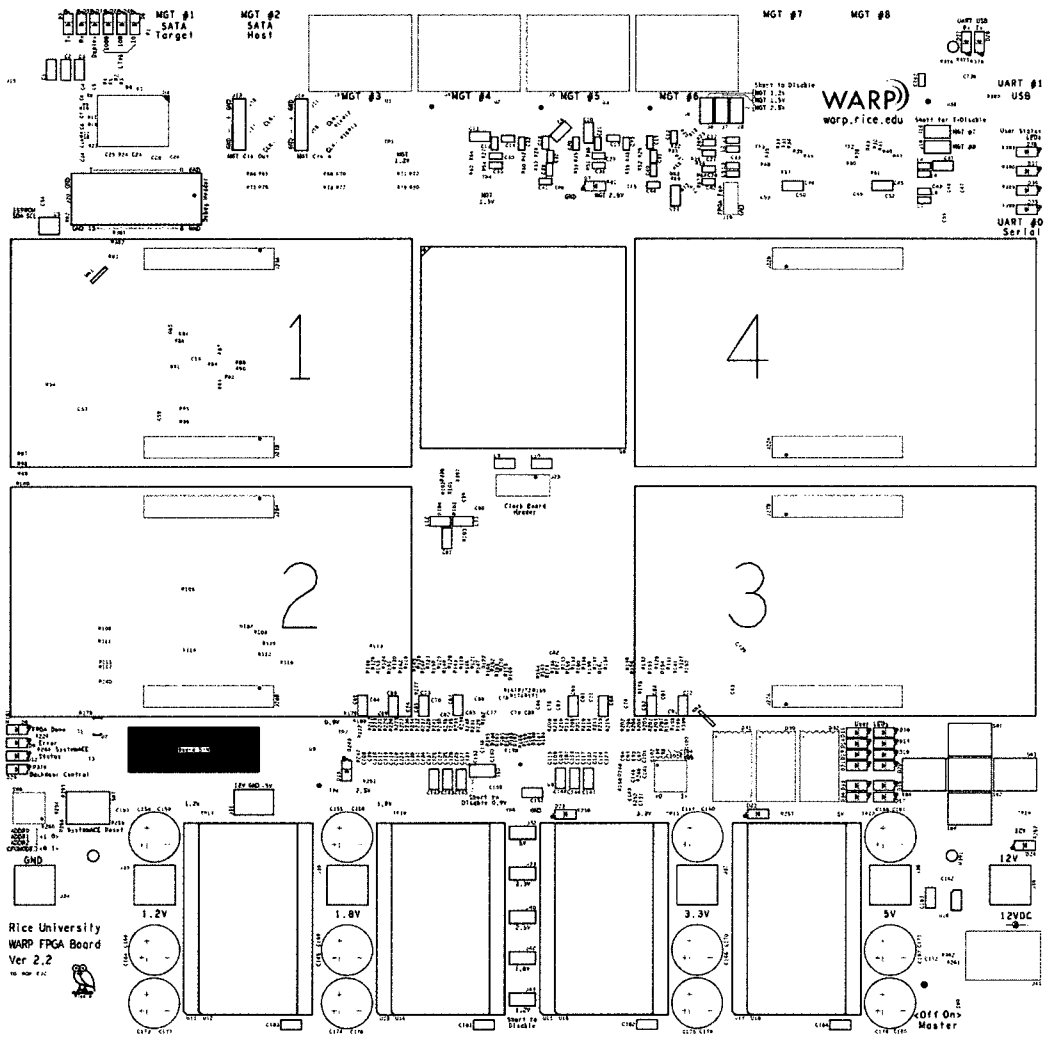


Pastemask Top

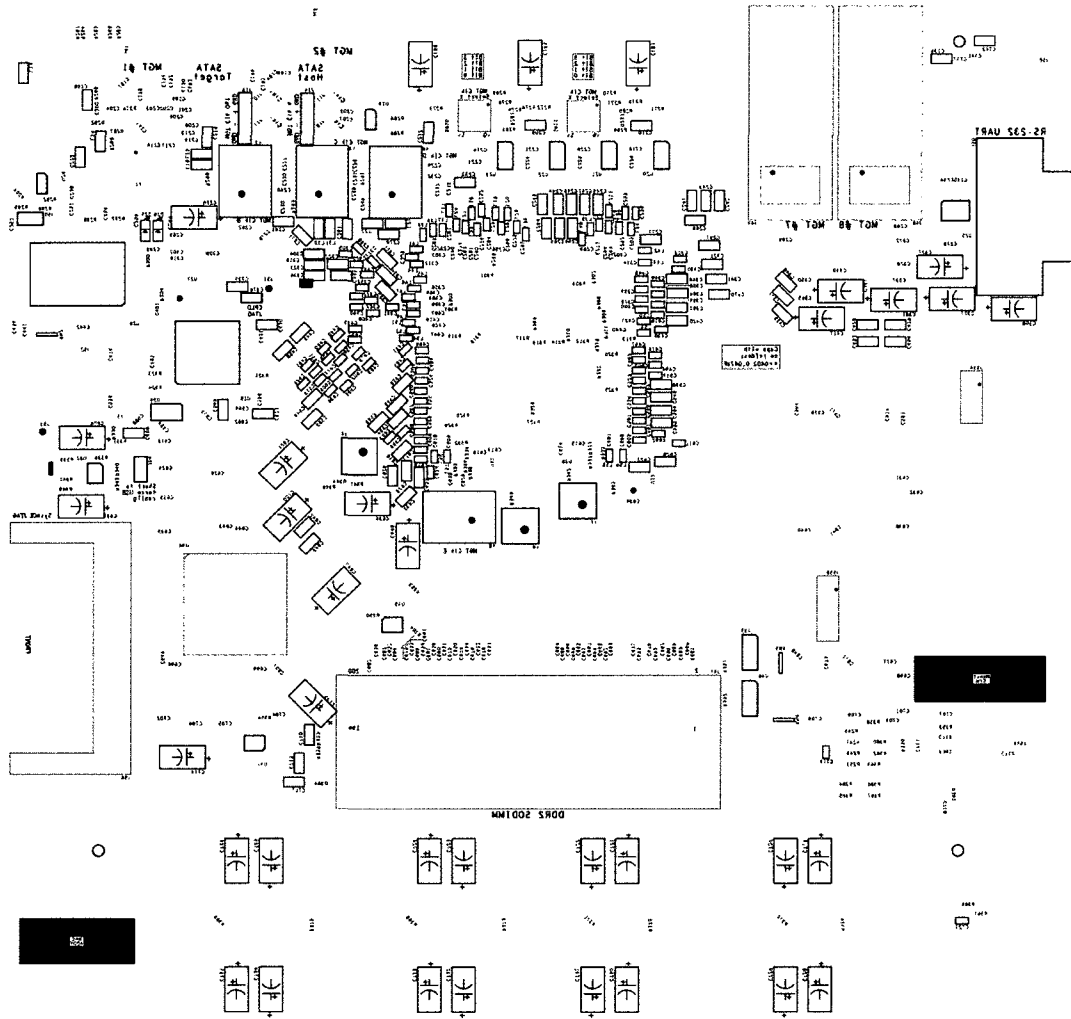


Pastemask Bottom





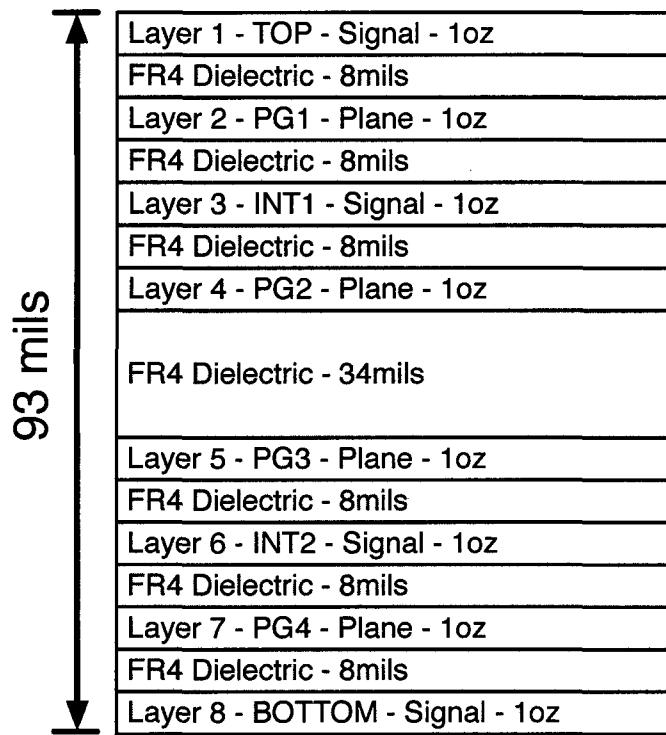
Silkscreen Top

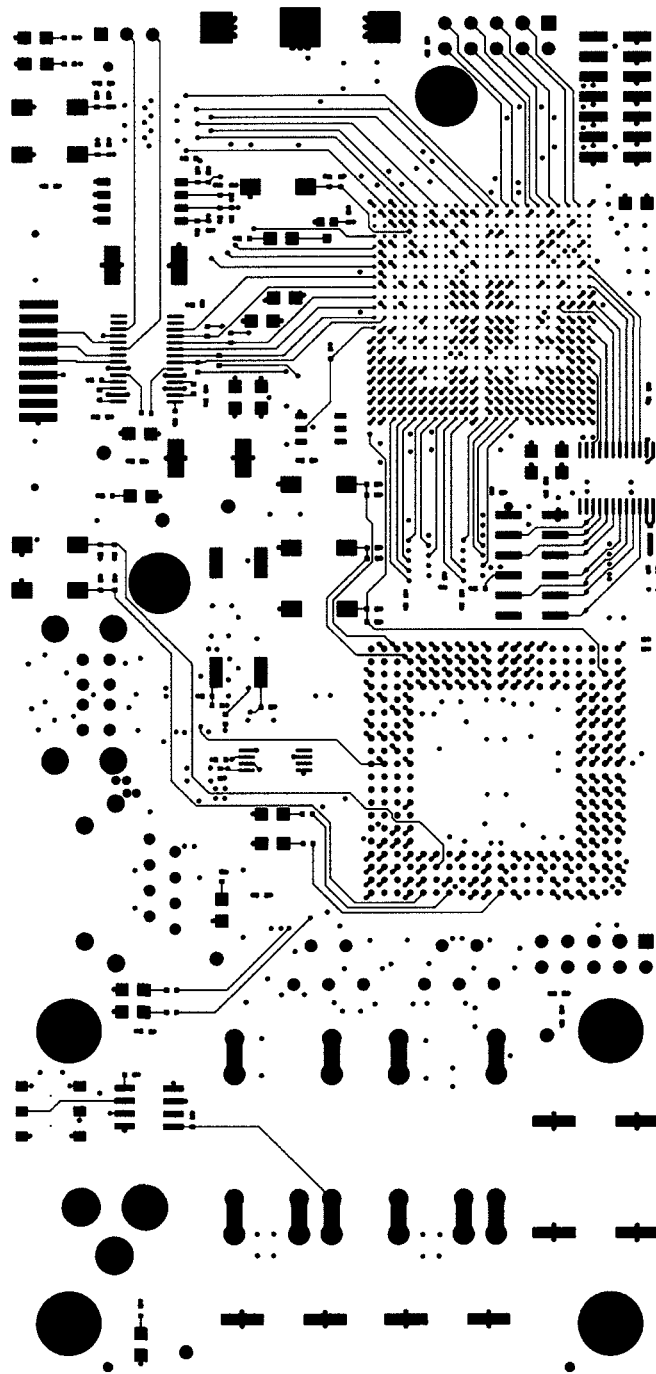


Silkscreen Bottom

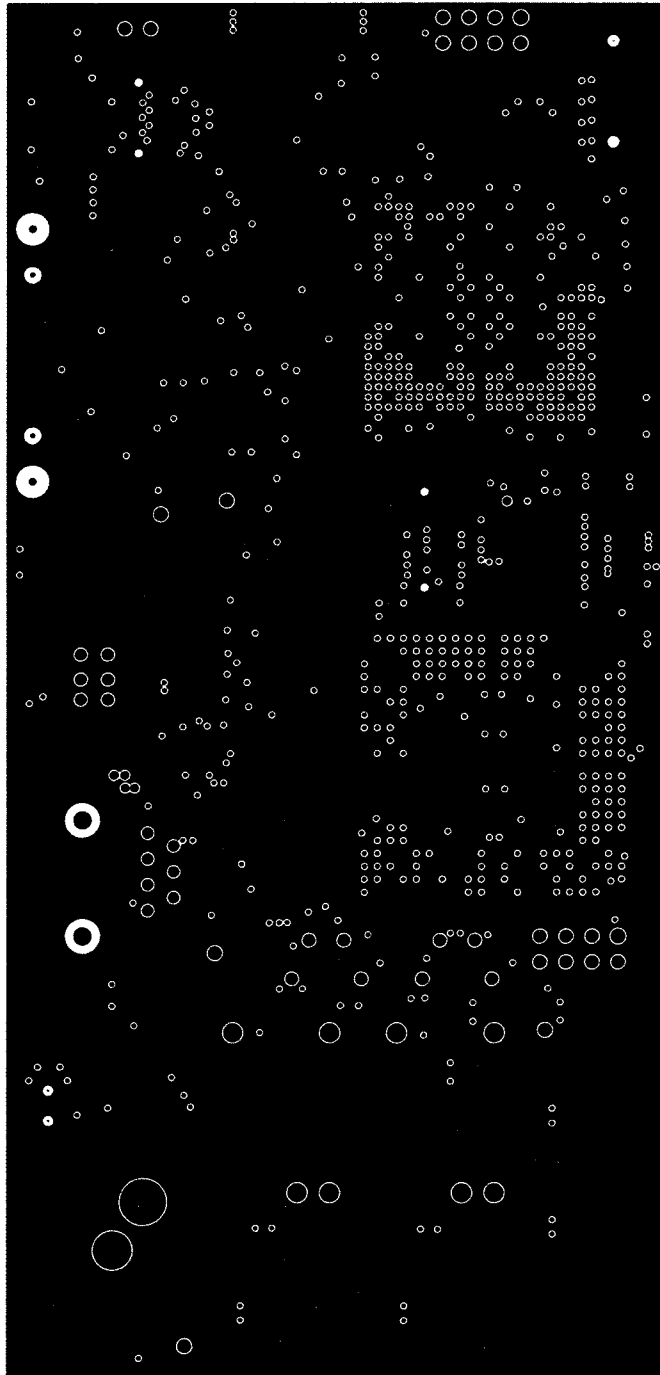
## **B.2 Backdoor Board**

Stackup and layout gerbers for the Backdoor Board.

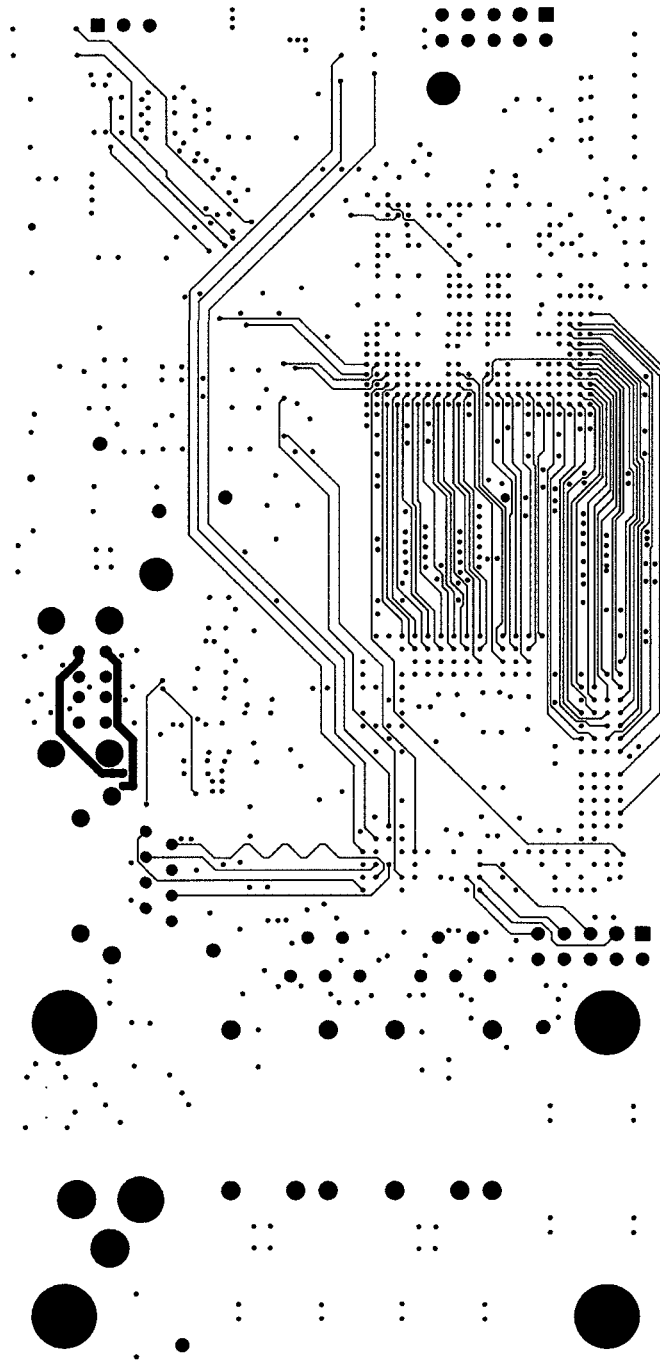




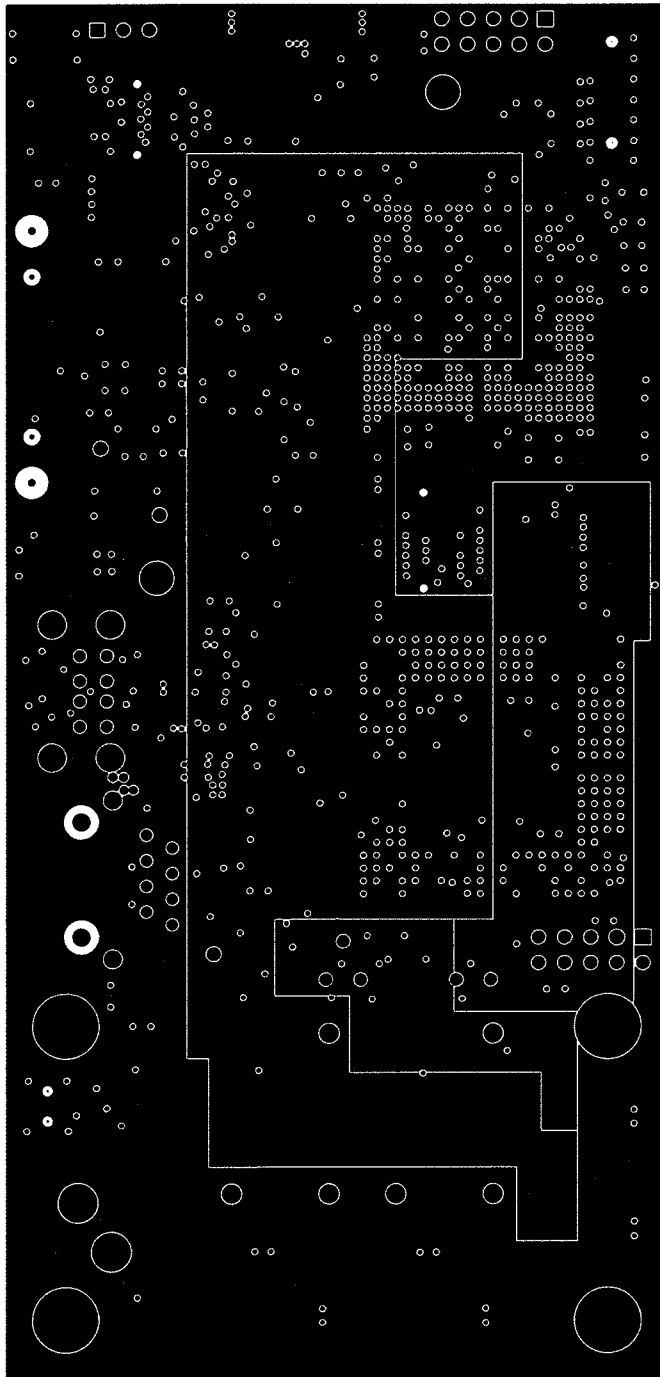
Top (Layer 1) - Signal Layer



PGI (Layer 2) - Ground Plane

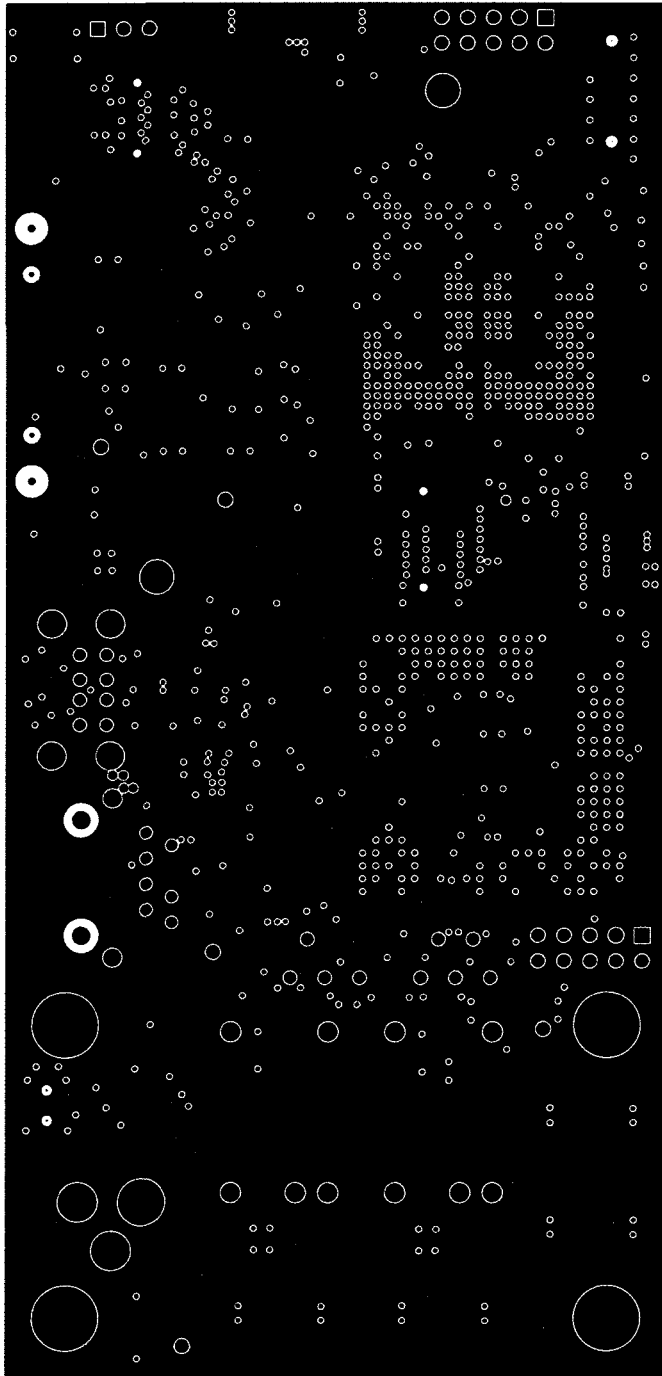


INT1 (Layer 3) - Signal Layer

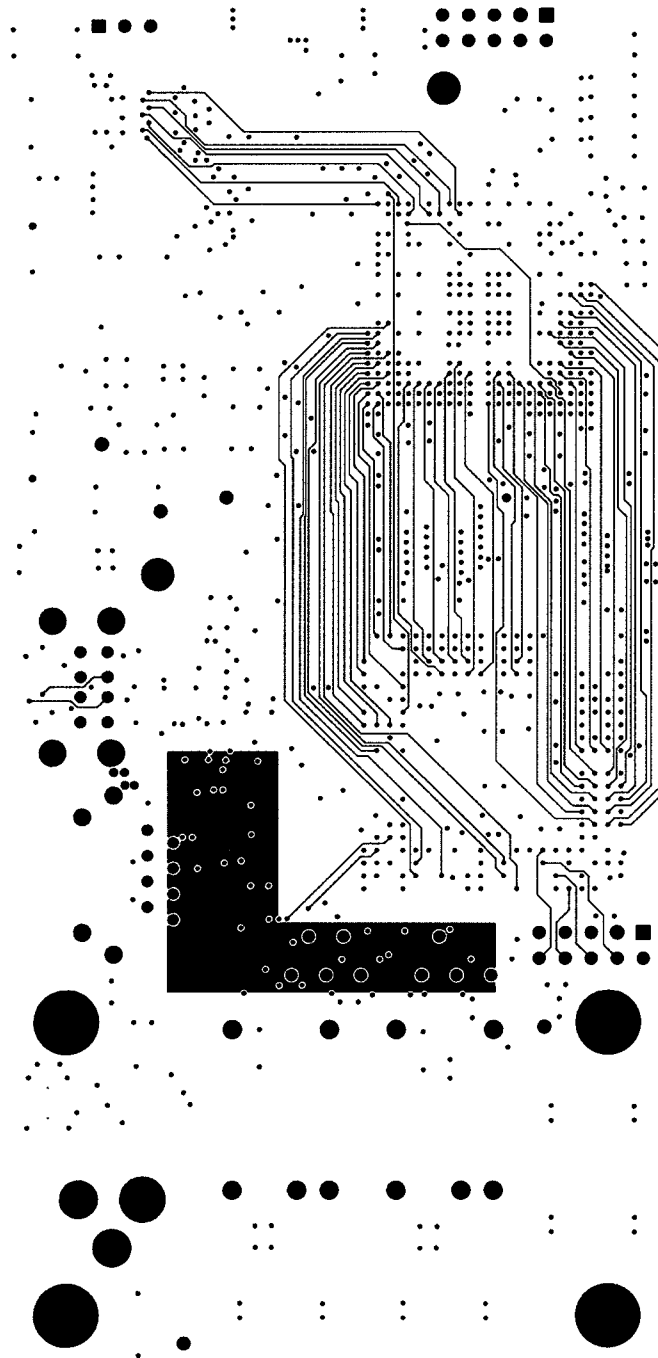


PG2 (Layer 4) - Power Plane

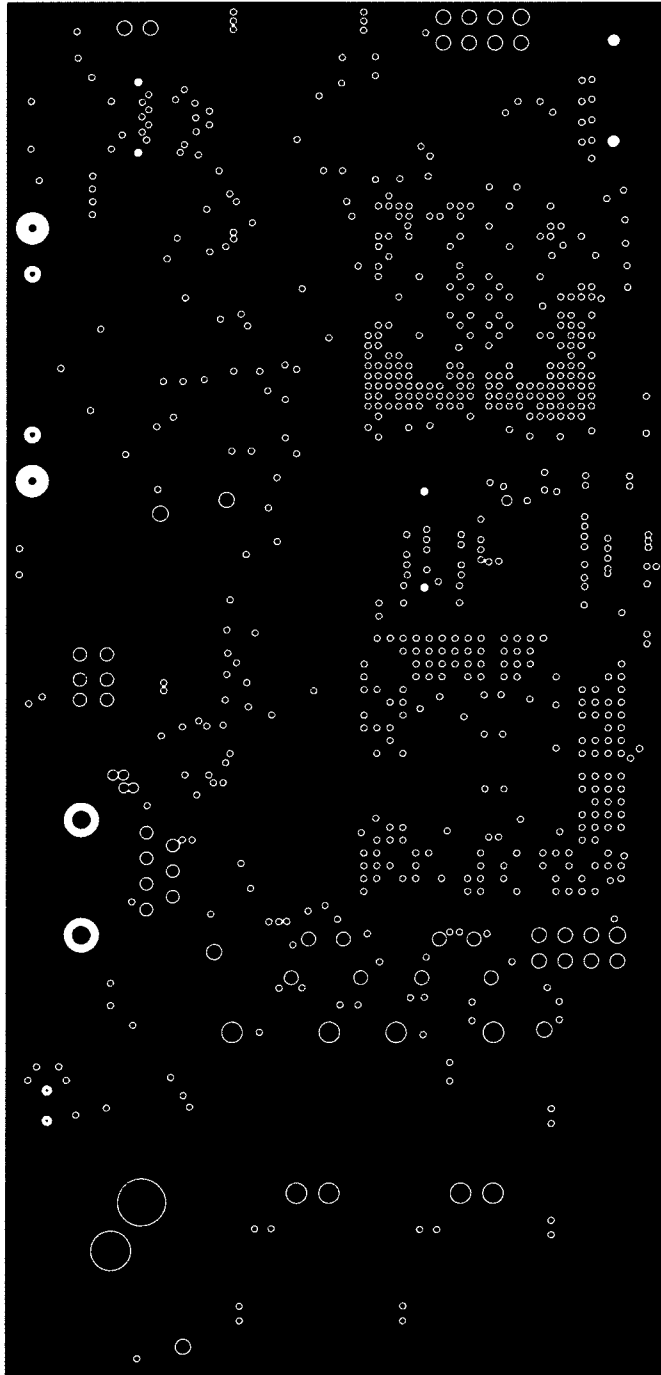




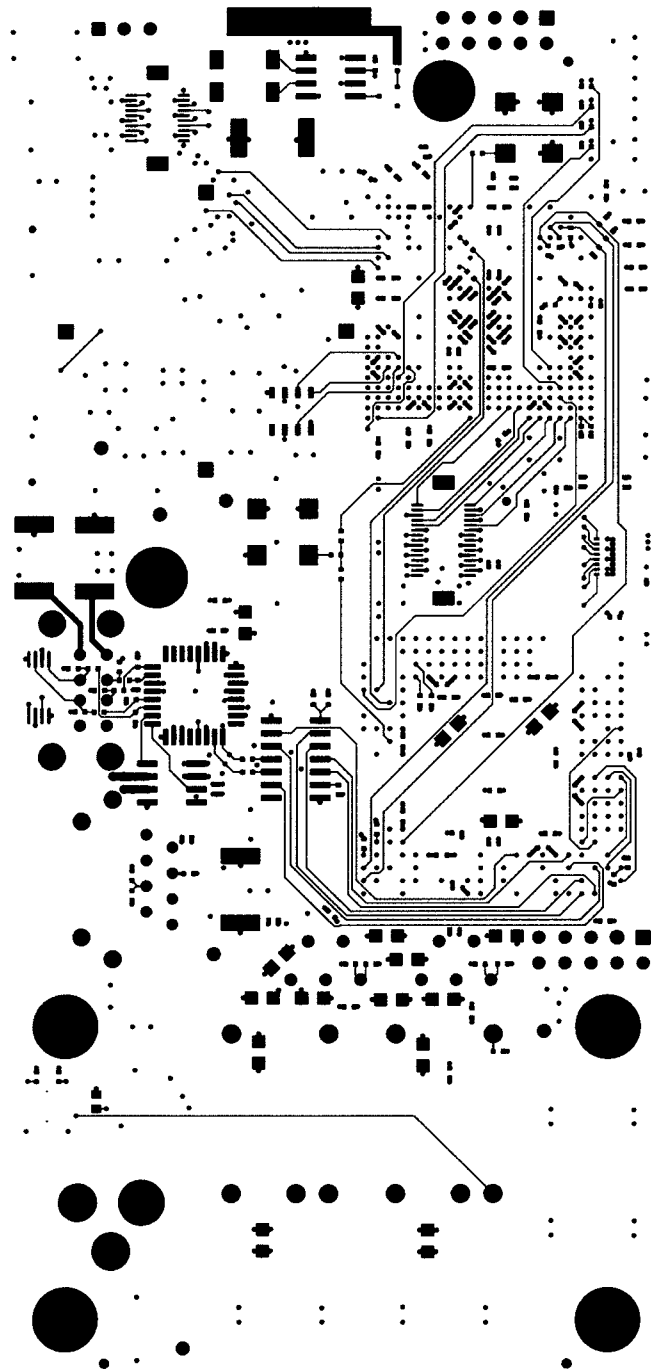
PG3 (Layer 5) - Power Plane



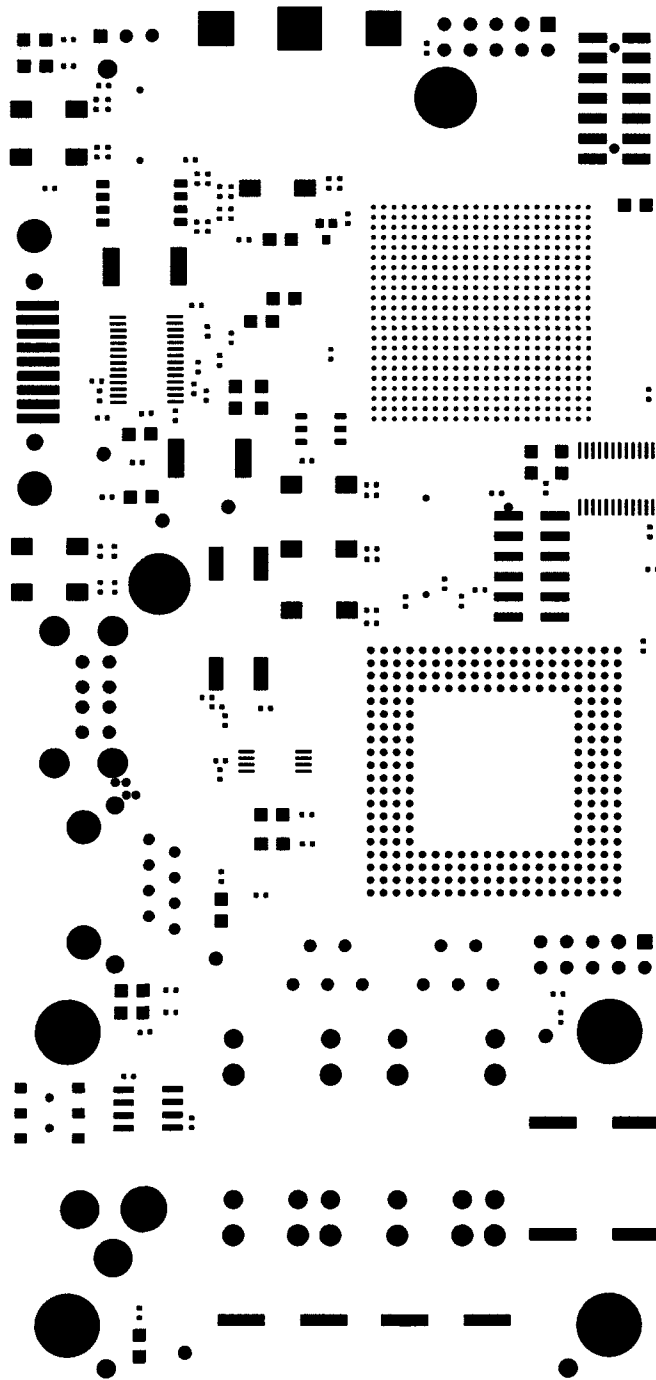
INT2 (Layer 6) - Signal Layer



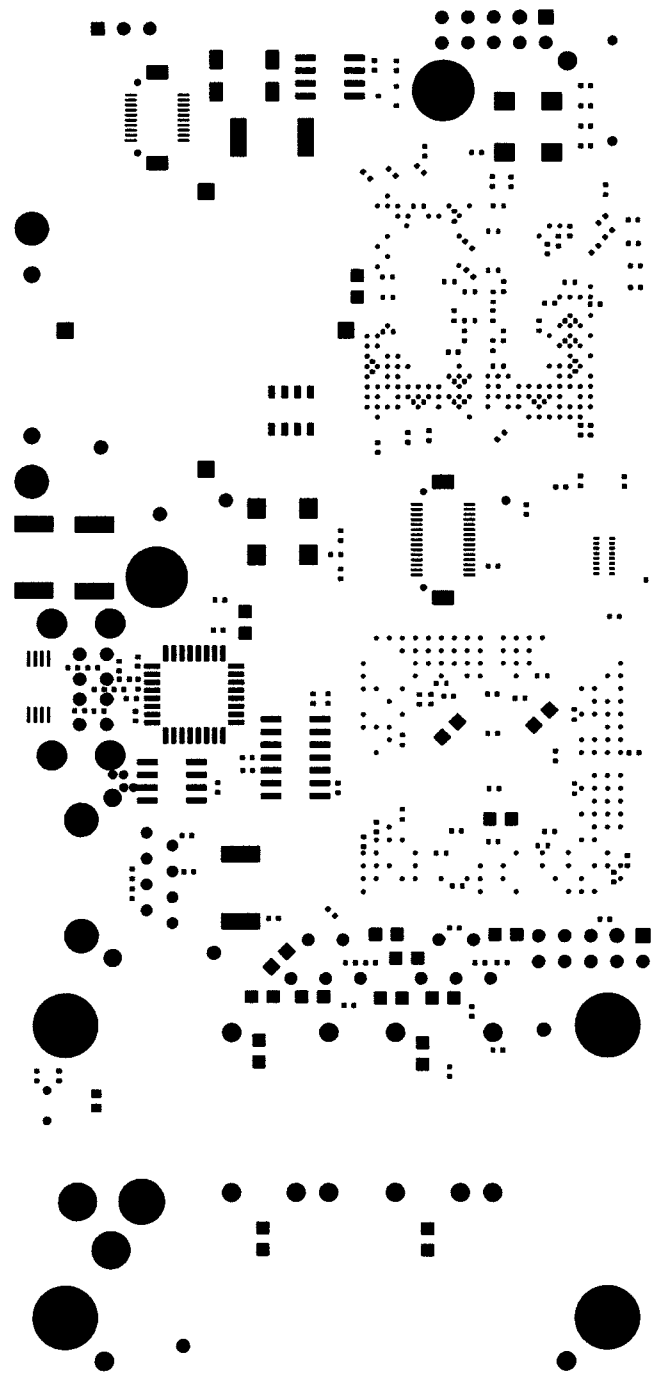
PG4 (Layer 7) - Ground Plane



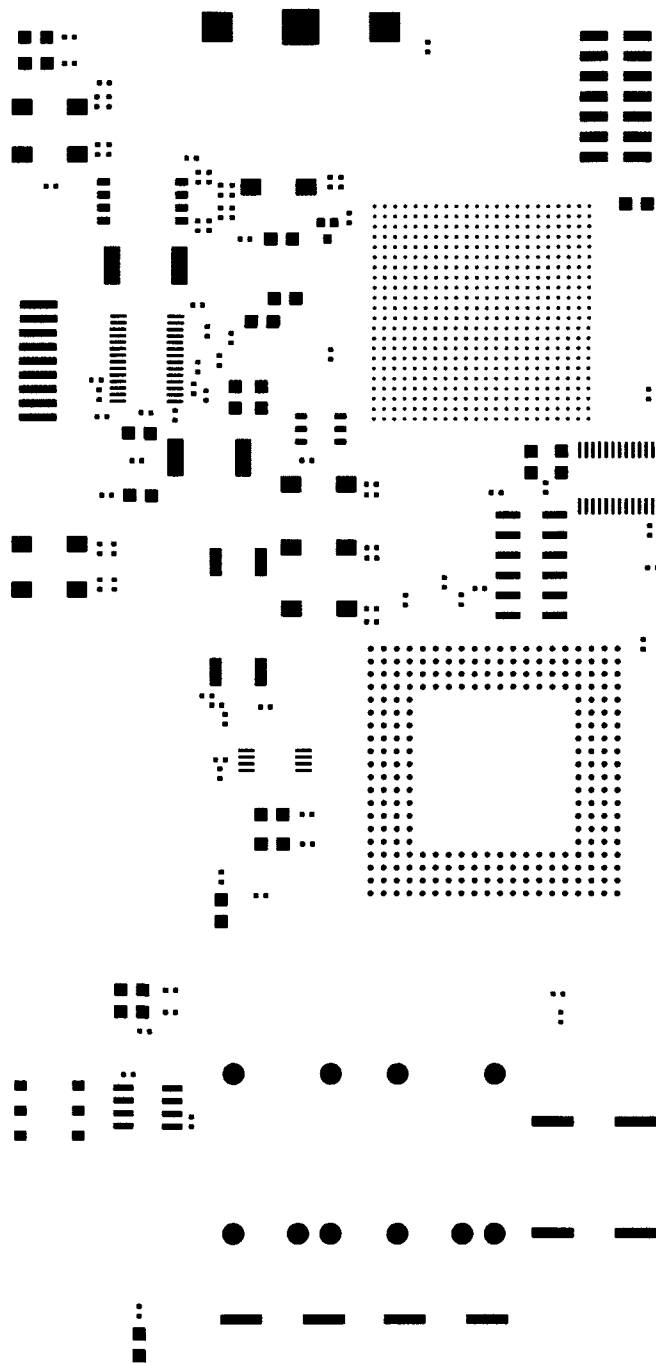
Bottom (Layer 8) - Signal Layer



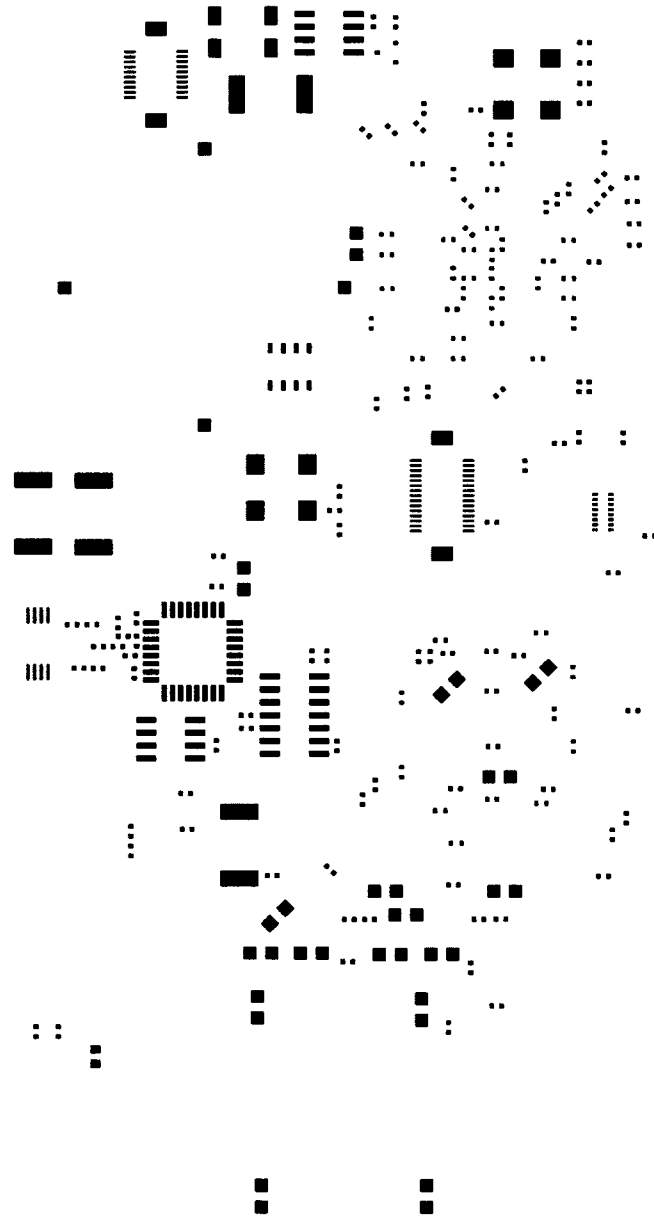
Soldermask Top



Soldermask Bottom

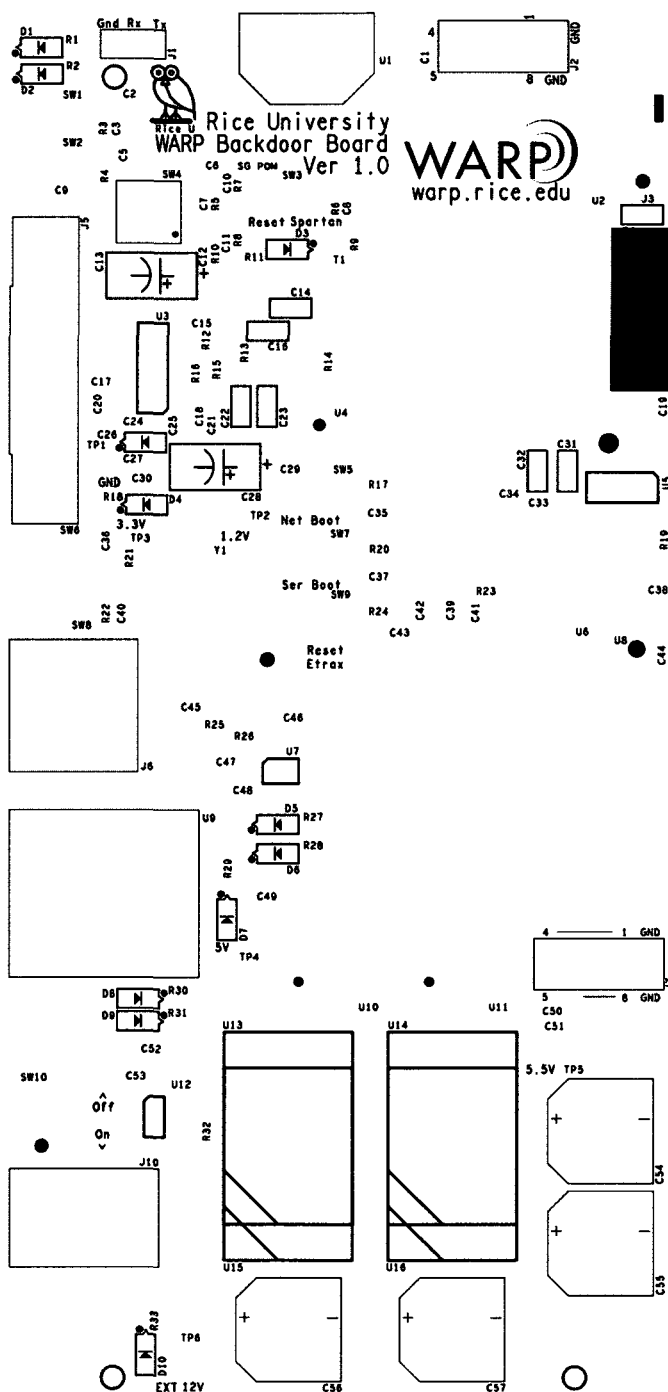


Pastemask Top

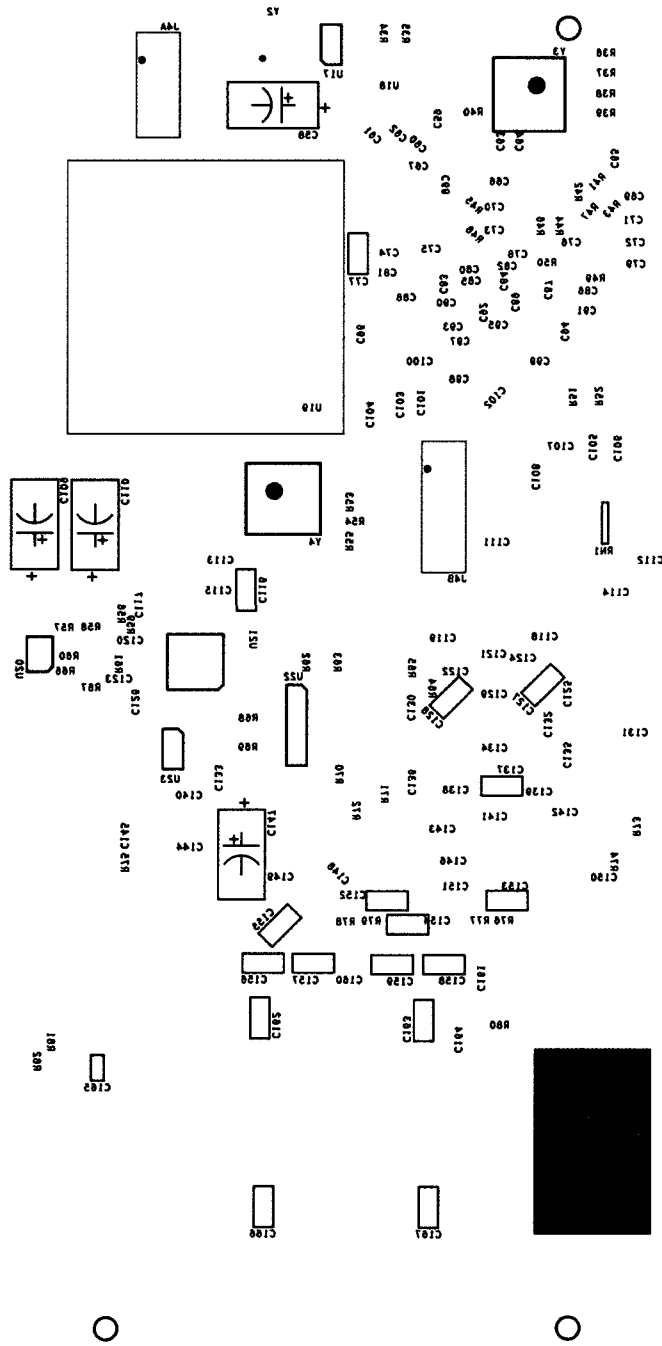


Pastemask Bottom





Silkscreen Top



Silkscreen Bottom