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

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

List of Figures

List of Tables

Chapter 1

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 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.

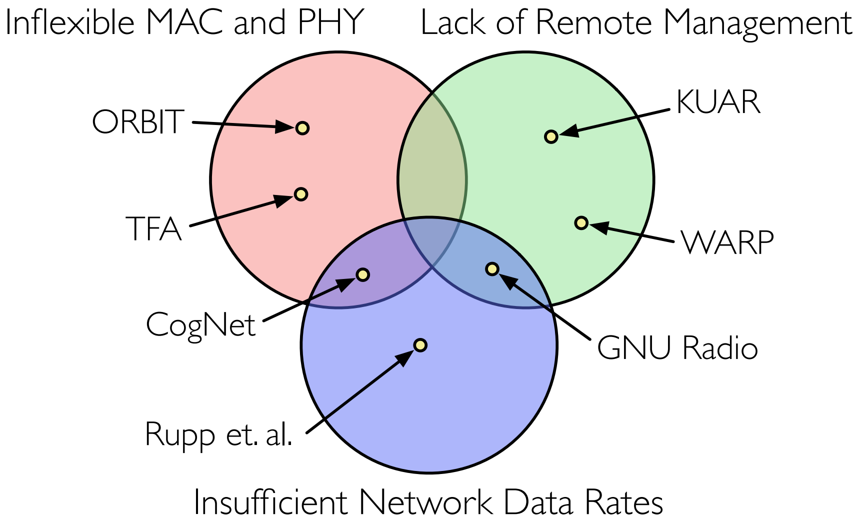


Figure 1.1: Characterization of Other Platforms

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.

Chapter 2

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.

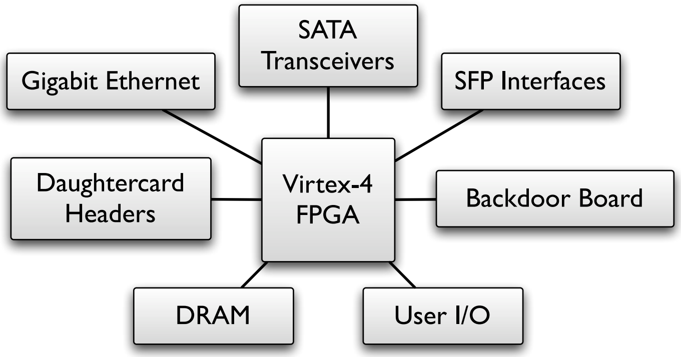


Figure 2.1: Block Diagram of the Virtex-4 FPGA Board

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 maintaing 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 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.

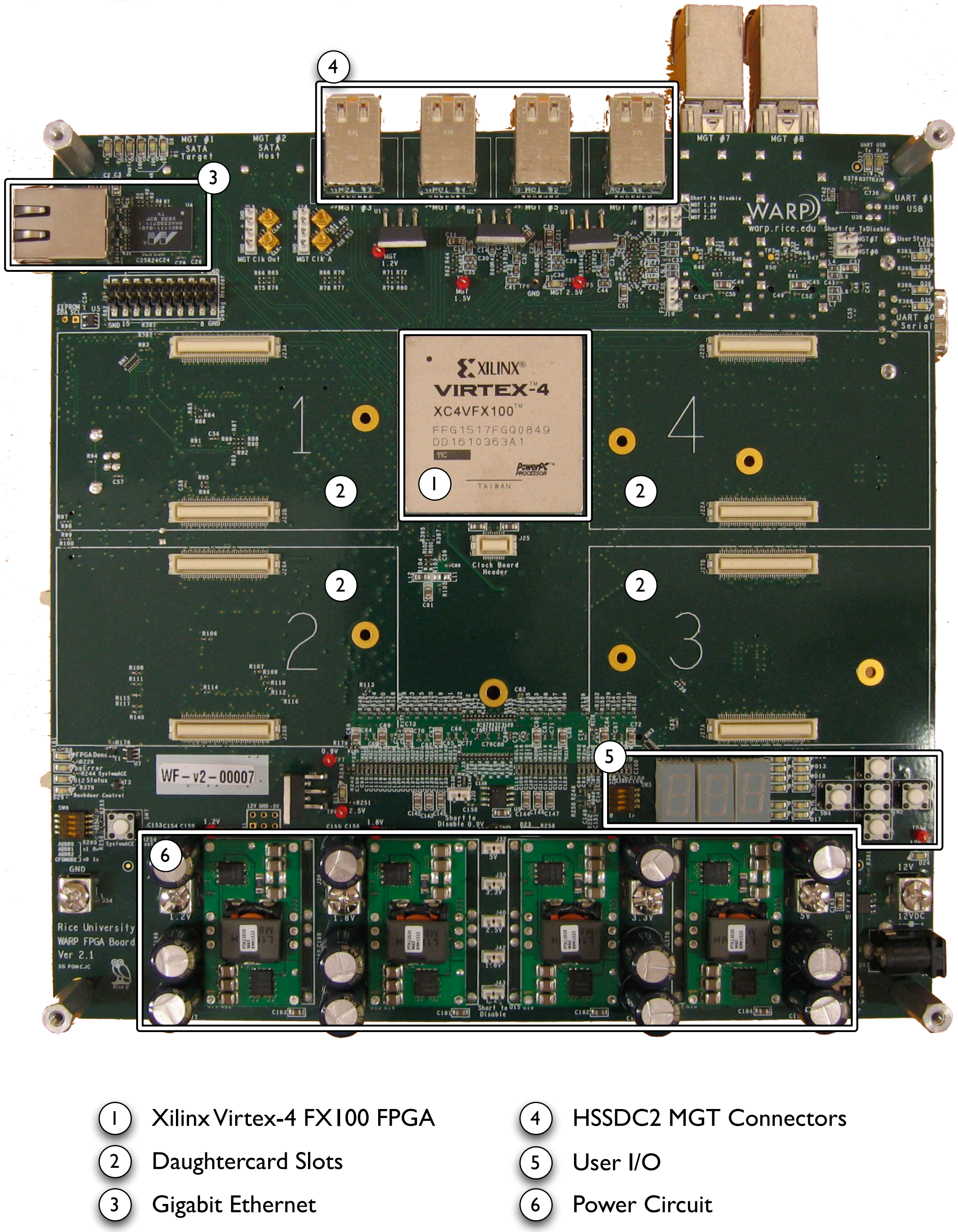
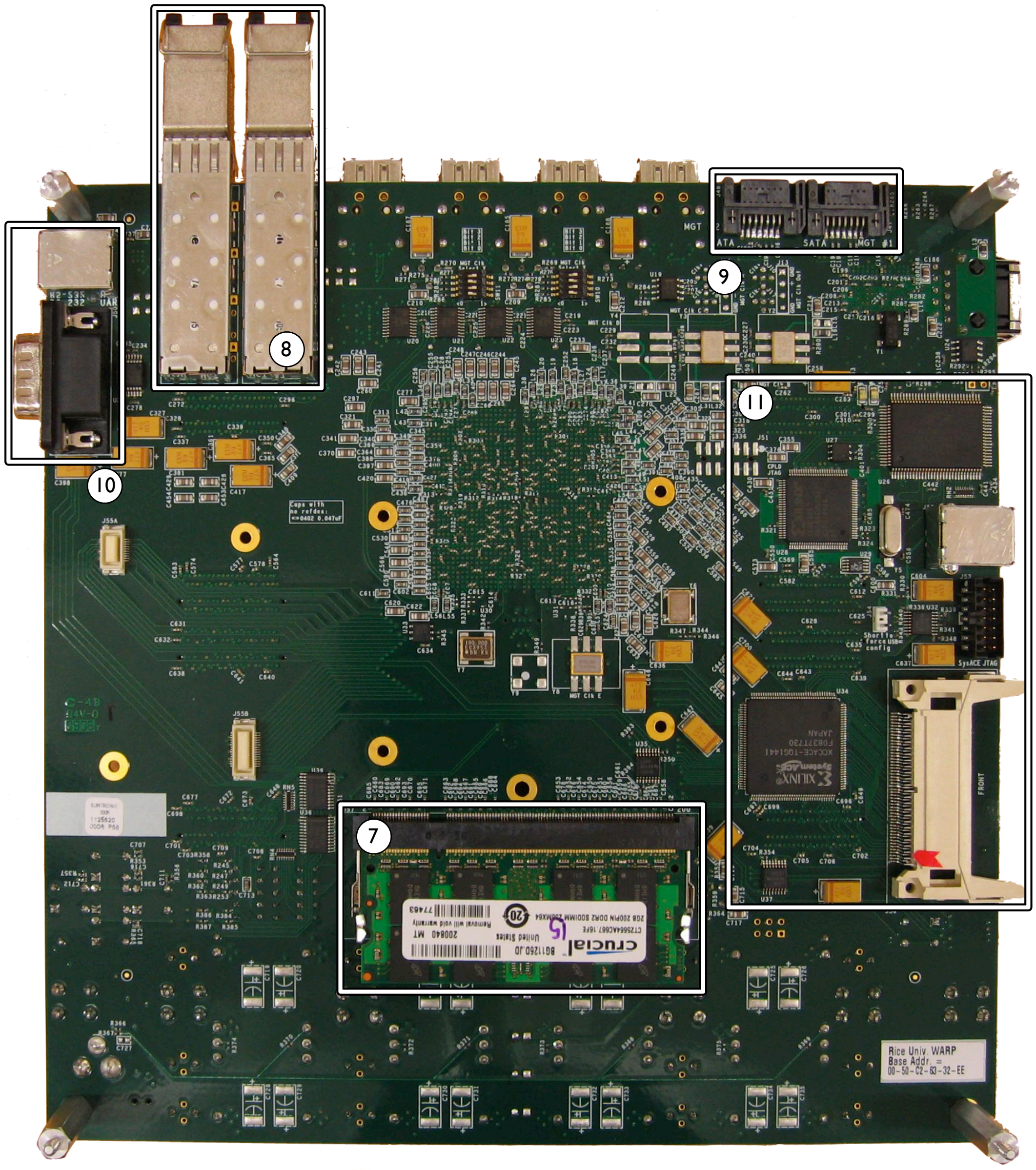


Figure 2.2: Front of FPGA Board



(7) DDR2 SDRAM

(8) SFP MGT Connectors

(9) SATA MGT Connectors

(10) User I/O

(11) Reconfiguration Circuit

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 Ω 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.

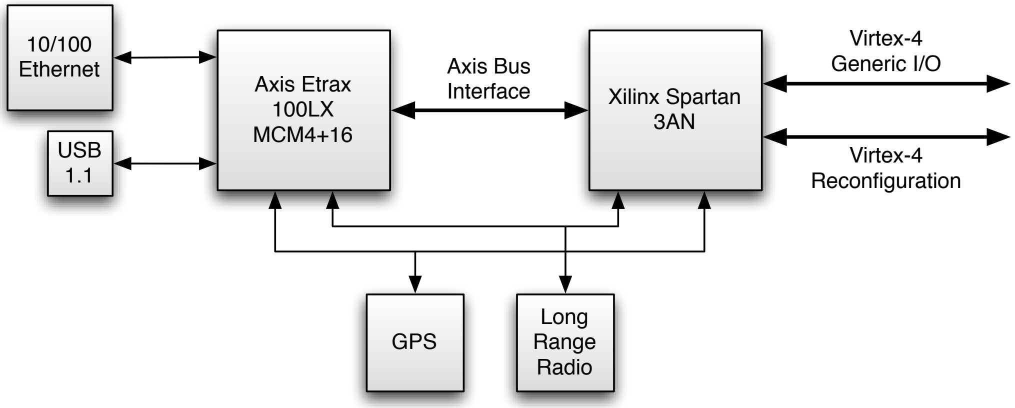


Figure 2.4: Block Diagram of Backdoor Board

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 components: 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

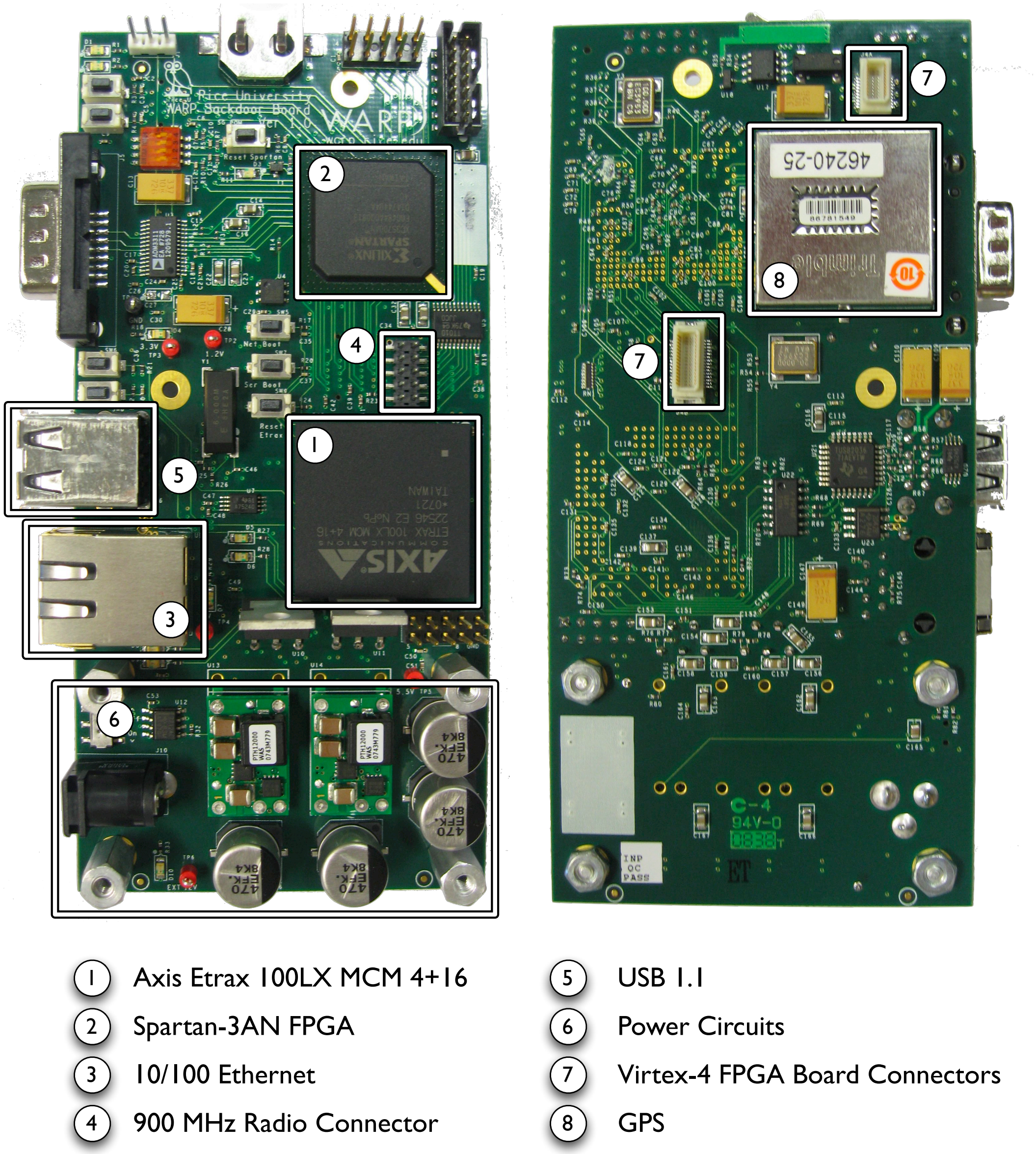

1 Axis Etrax 100LX MCM 4+16
2 Spartan-3AN FPGA
3 10/100 Ethernet
4 900 MHz Radio Connector
5 USB 1.1
6 Power Circuits
7 Virtex-4 FPGA Board Connectors
8 GPS

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.

Chapter 3

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.

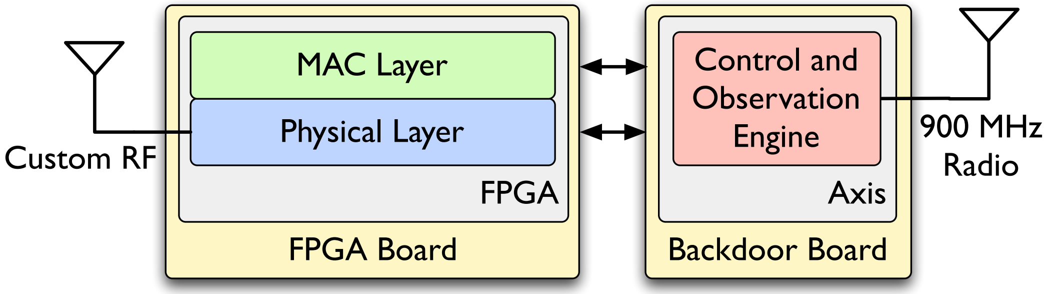


Figure 3.1: Implementation on Platform

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 enginer 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 is 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.

	Throughput Data Rate	
	9,600 bps	**115,200 bps**
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

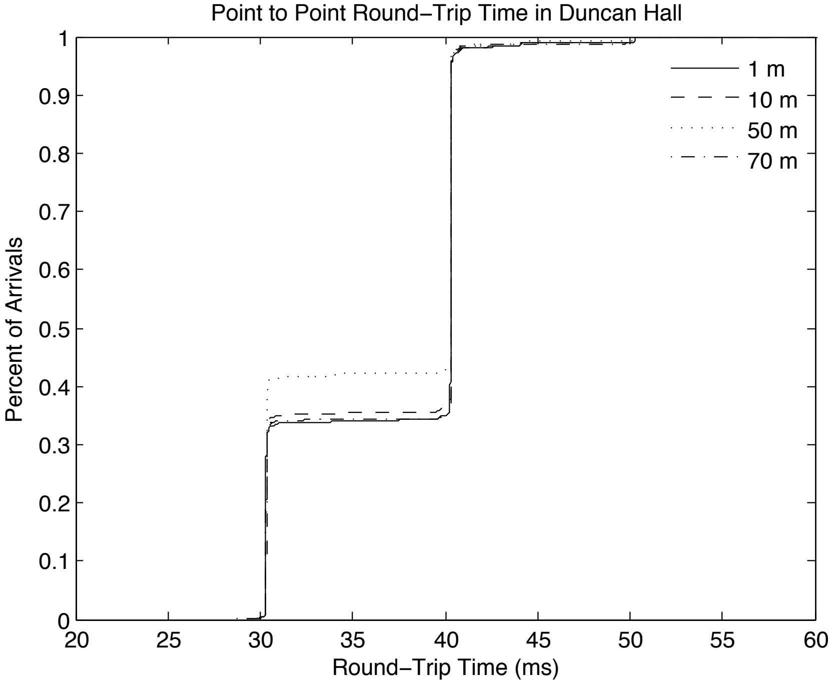


Figure 3.2: Round-Trip Time for Two Nodes with Varying Distances

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 μs followed by 64 μs of medium usage for the acknowledgement. On average the backoff

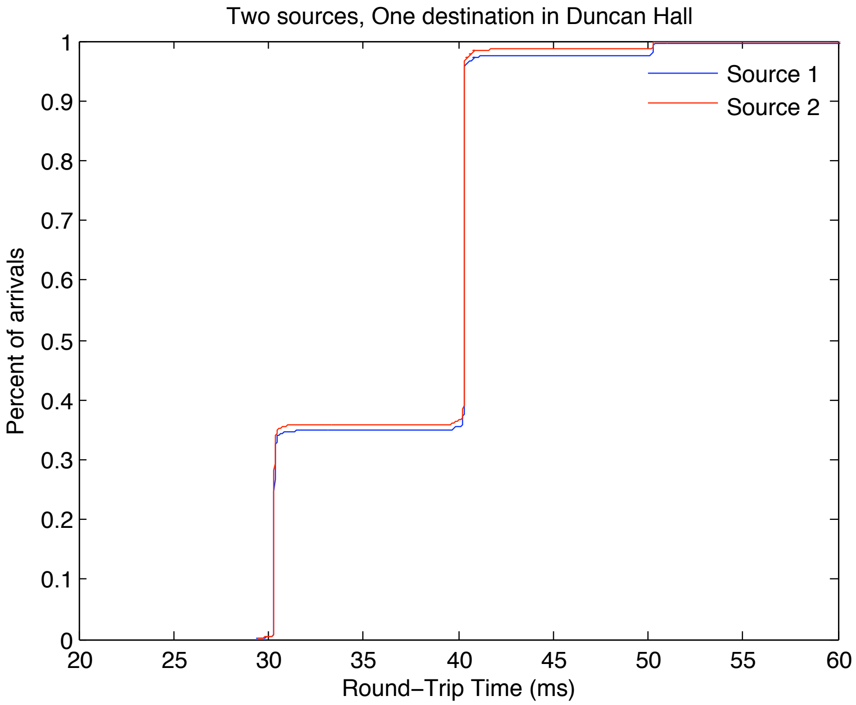


Figure 3.3: Round-Trip Time for Two Transmitters Feeding One Destination

window would utilize 8 time slots at 20 μs each, adding a further delay of 160 μs. 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.

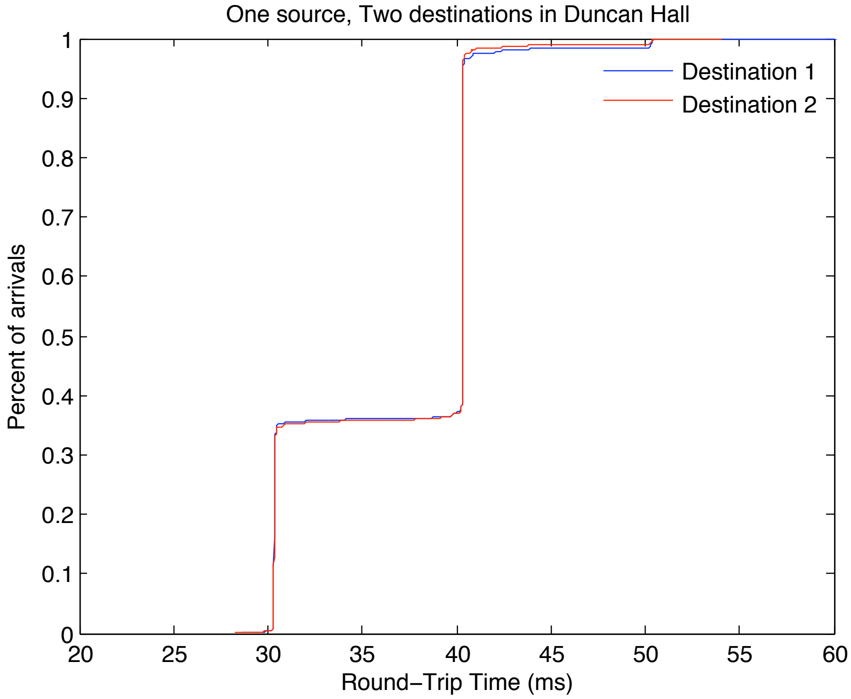


Figure 3.4: Round-Trip Time for One Transmitter Feeding Two Destinations

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 ∼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.

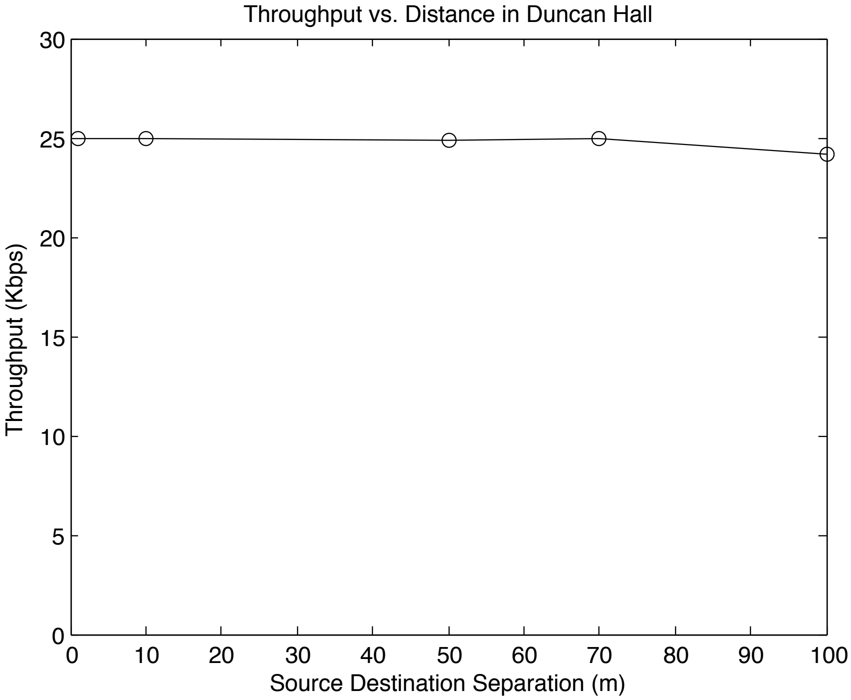


Figure 3.5: Throughput for Various Distances

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.

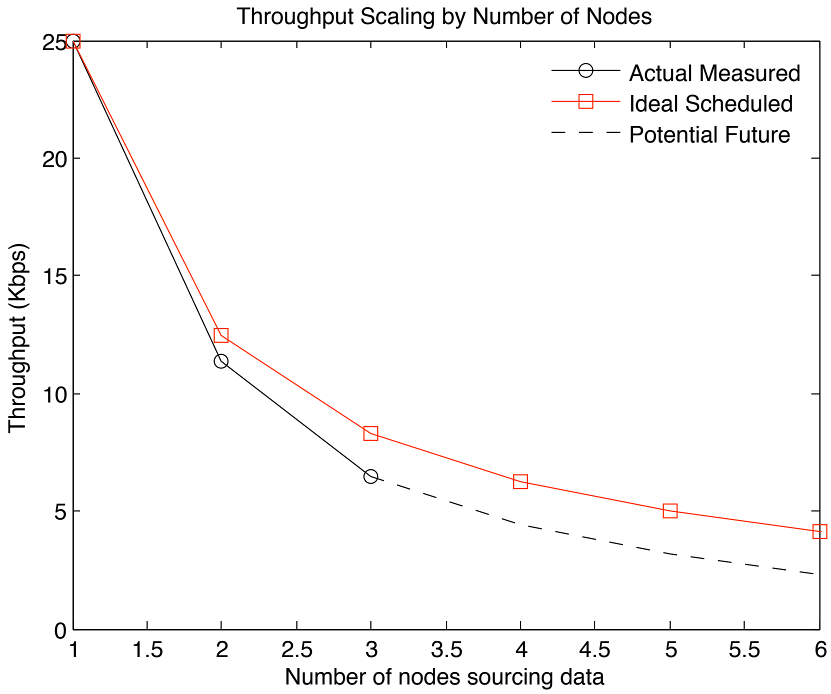


Figure 3.6: Throughput Scaling as Number of Transmitters Increases

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.

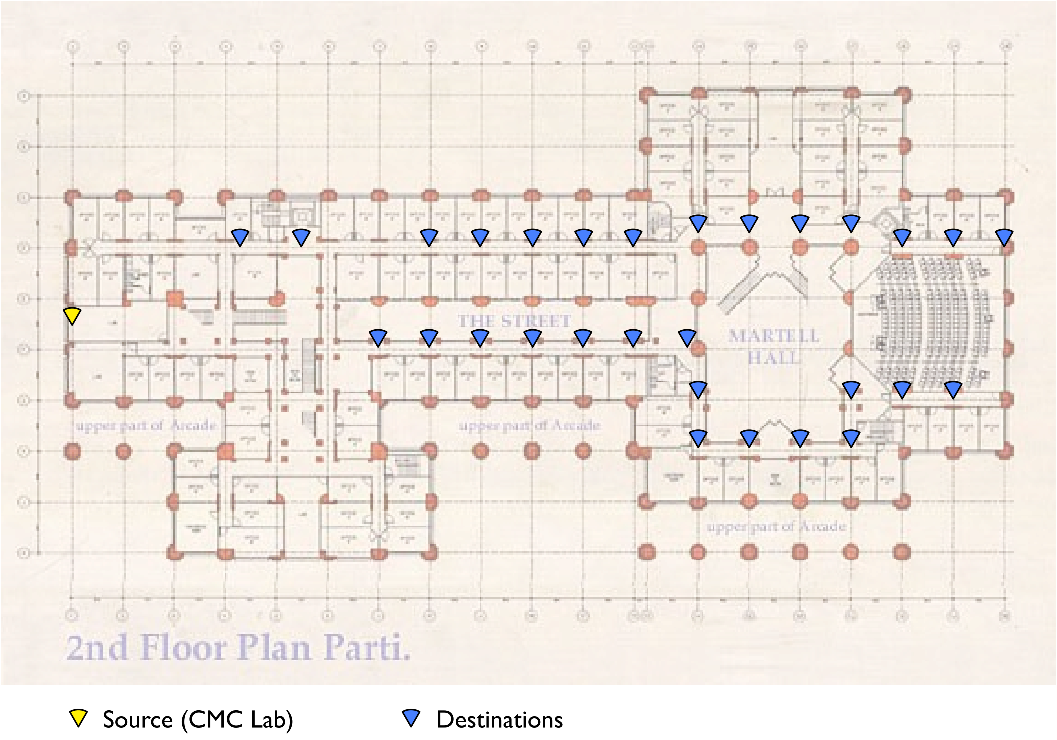


Figure 3.7: Data Collection Locations for Maximum Range Experiments

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

$$PL(d) = PL(d_0) + 10nlog\left(\frac{d}{d_0}\right) + X_\sigma \tag{3.1}$$

where d_0 is the reference distance where the path loss is known, n is the path loss exponent and X_σ 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,

$$PL(d) = PL(d_0) + 10nlog\left(\frac{d}{d_0}\right) \tag{3.2}$$

represents is the path loss model of interest.

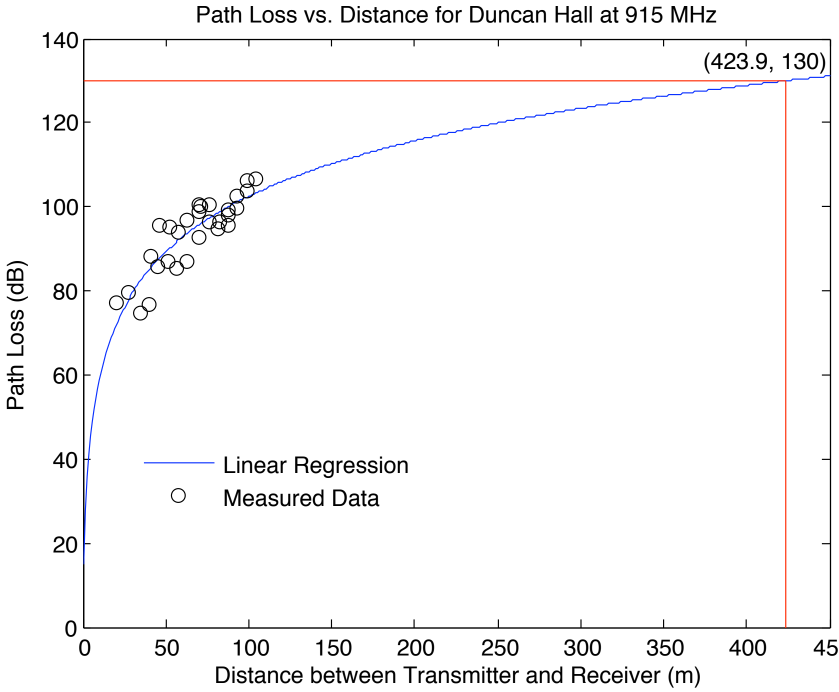


Figure 3.8: Path Loss Measurements with Linear Regression

The measured RSSI data is plotted in Figure 3.8 as the black scatter points. The distance range is limited to ∼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,

$$PL(d) = 14.86 + 19.02 \cdot d \tag{3.3}$$

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.

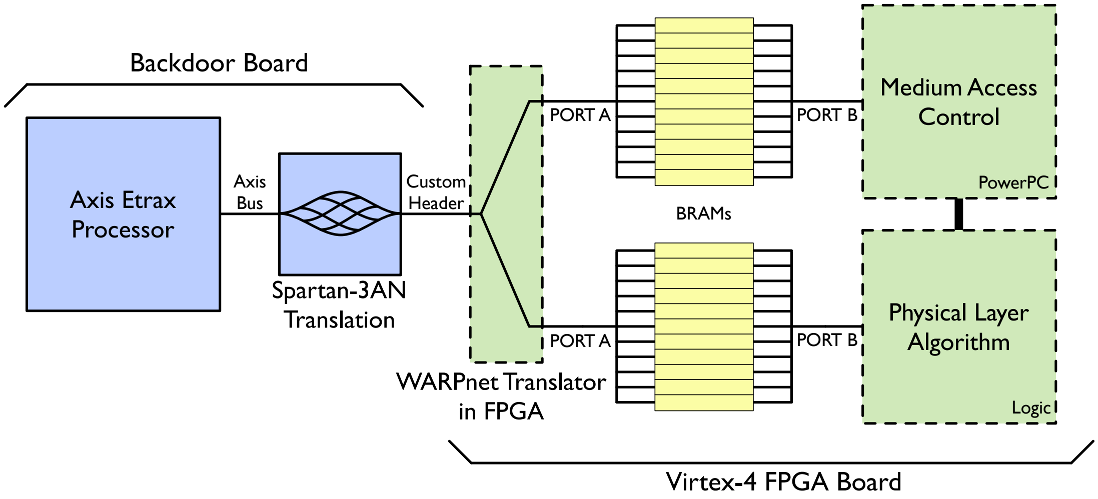


Figure 3.9: Data Flow Decoupling

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**
Slices	518	5,888
Block RAM	11	20
Slice Flip Flops	591	11,776

Table 3.3: Resource Utilization for Spartan Design

	Used	**Available**
Slices	487	42,176
Block RAM	3	376
Slice Flip Flops	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.

Chapter 4

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.

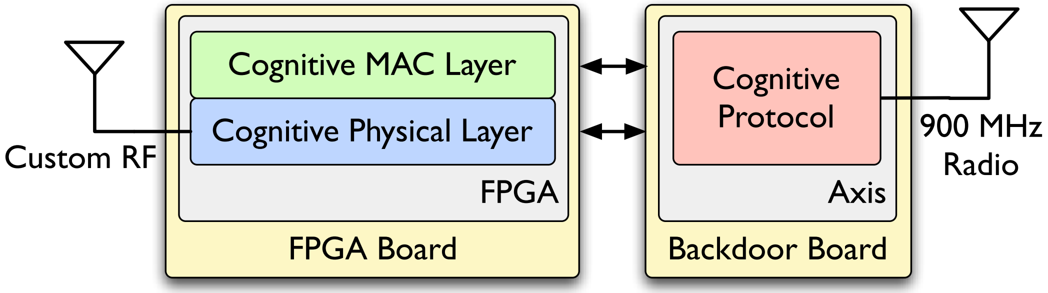


Figure 4.1: Protocol Implementation on Platform

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.

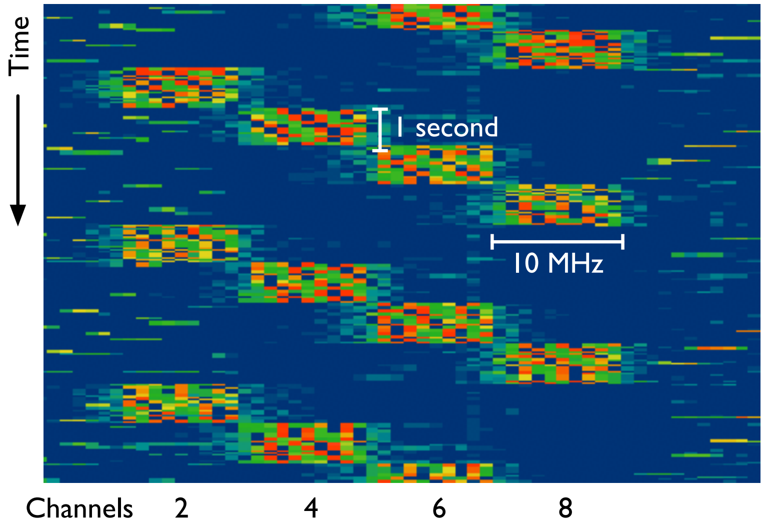


Figure 4.2: Channel Access with Primary Only

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-

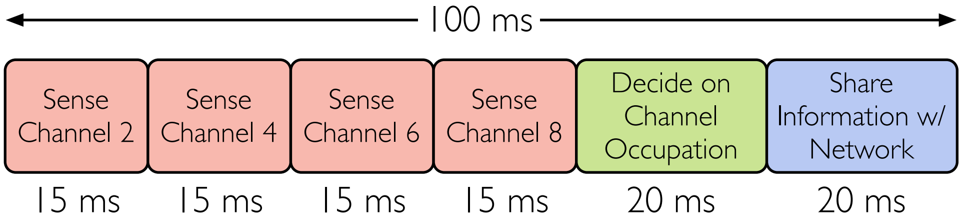


Figure 4.3: Division of Sensing Time

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 is due to RSSI measurements. Further performance analysis and algorithm extensions are discussed in Section 4.5.

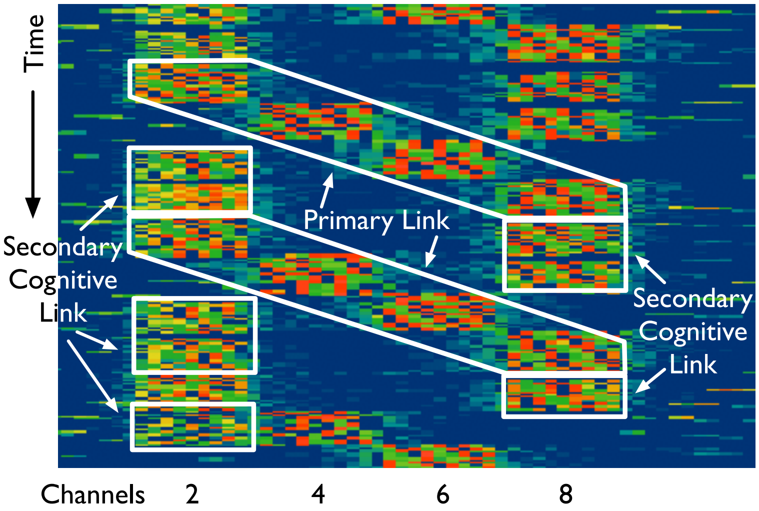


Figure 4.4: Channel Access with Primary and Distributed Secondary Links

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.

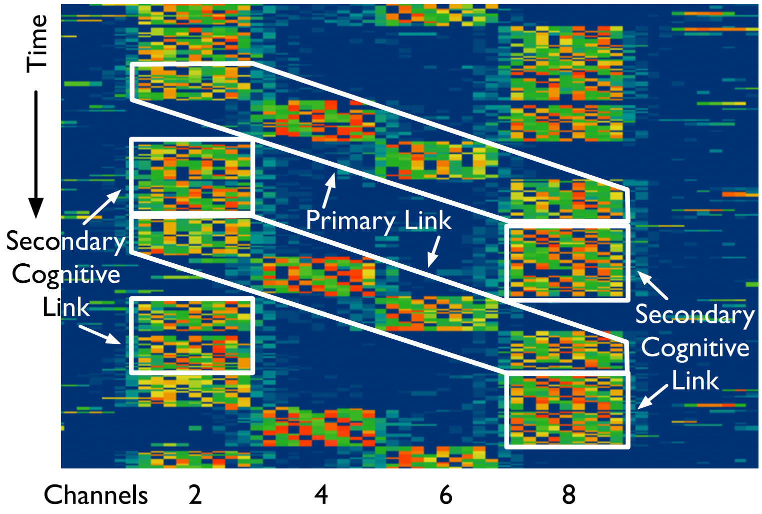


Figure 4.5: Channel Access with Primary and Centralized Secondary Links

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 is 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.

Chapter 5

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 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/. 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: Enahncements 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 Propogation*, 1992. 3.2.3

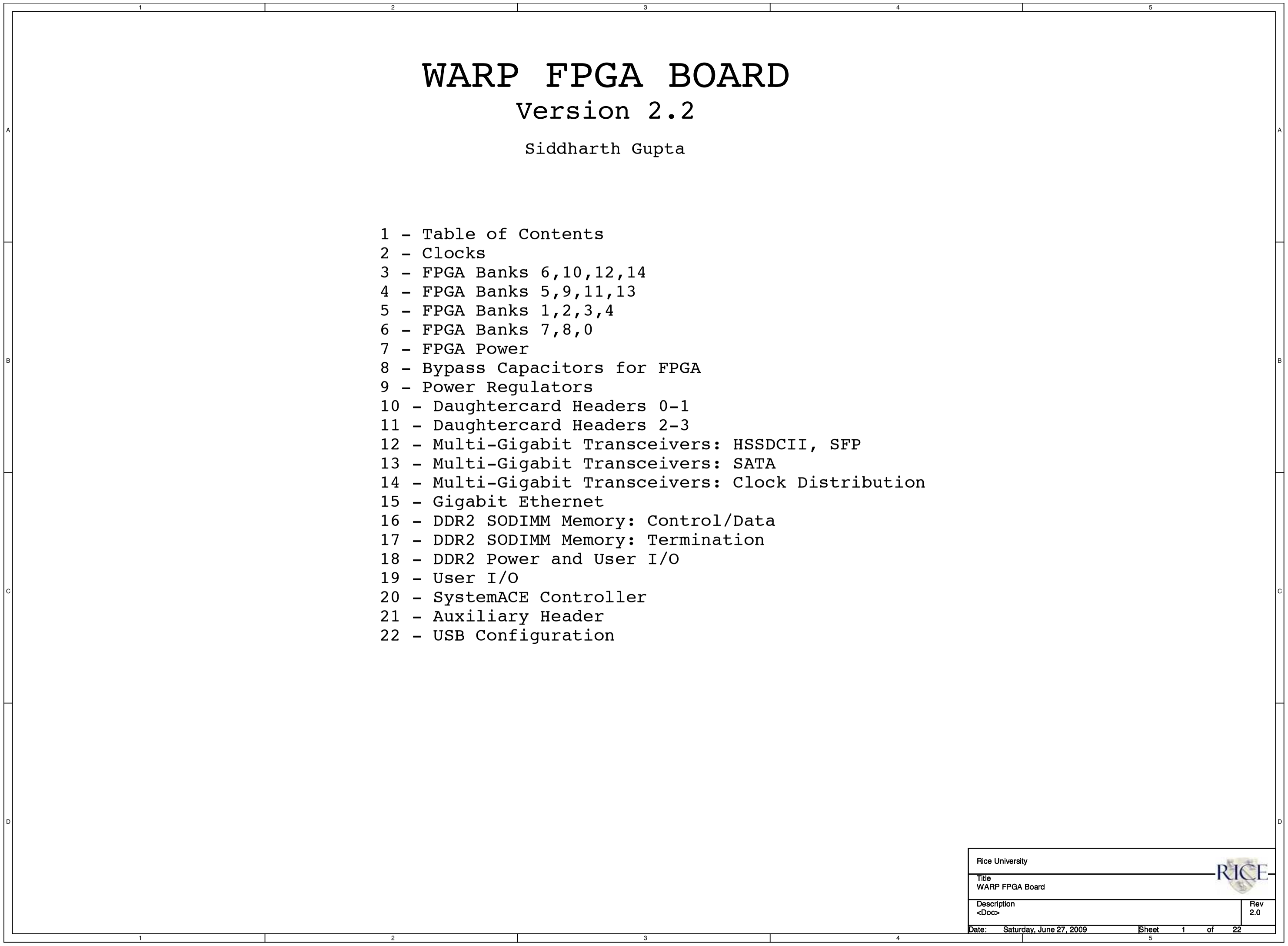
Appendix A

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

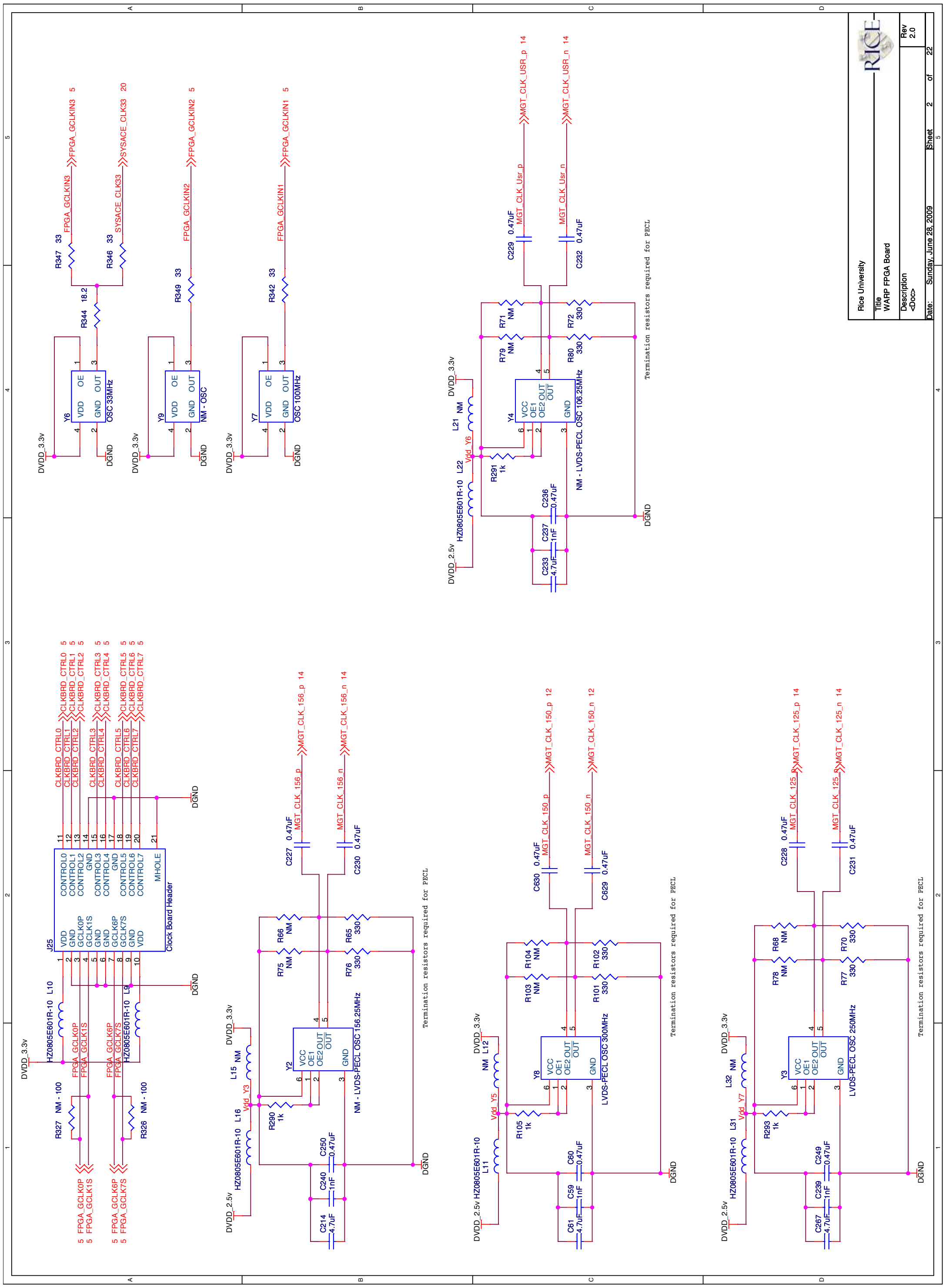
Rice University

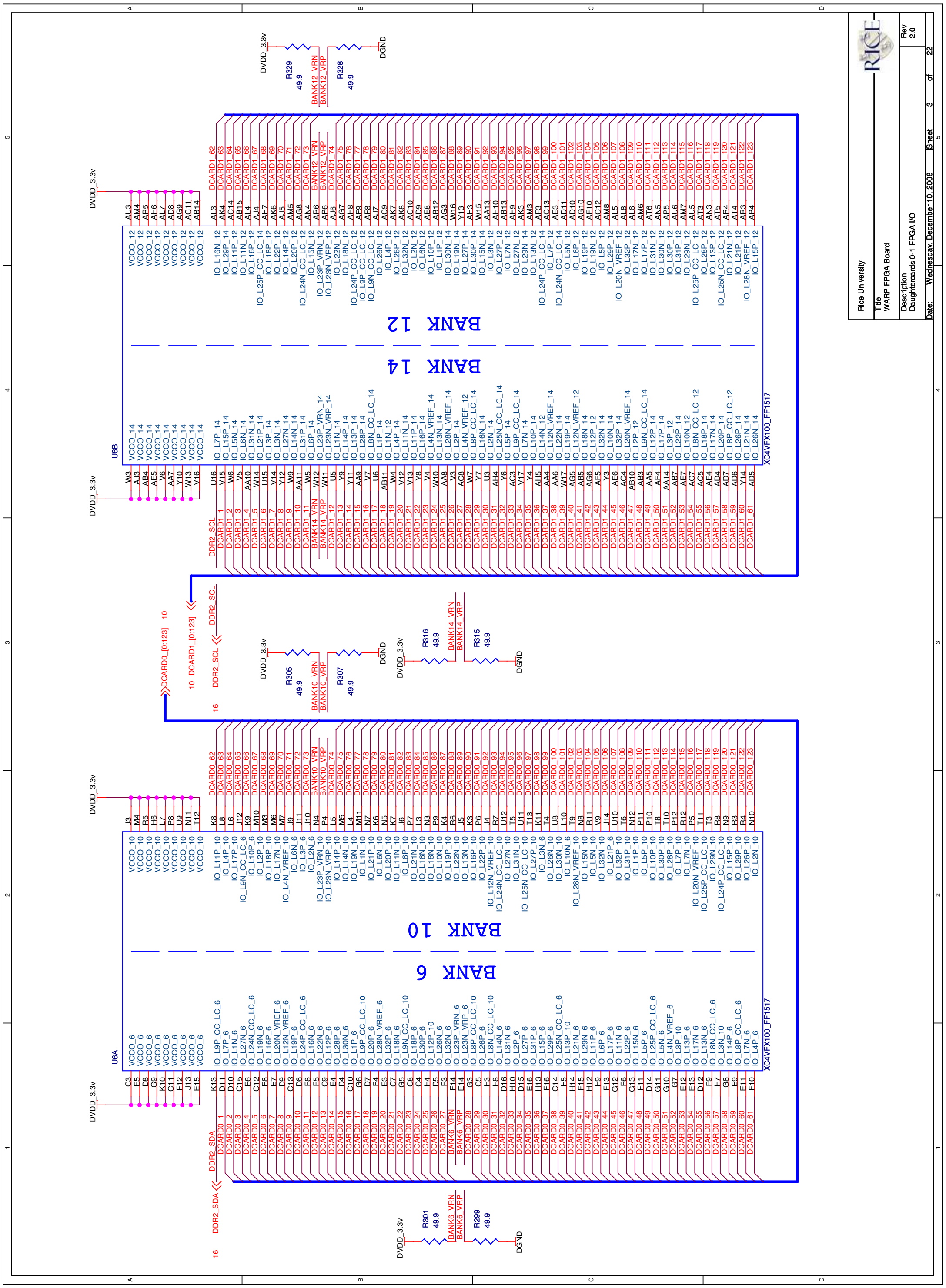
Title
WARP FPGA Board

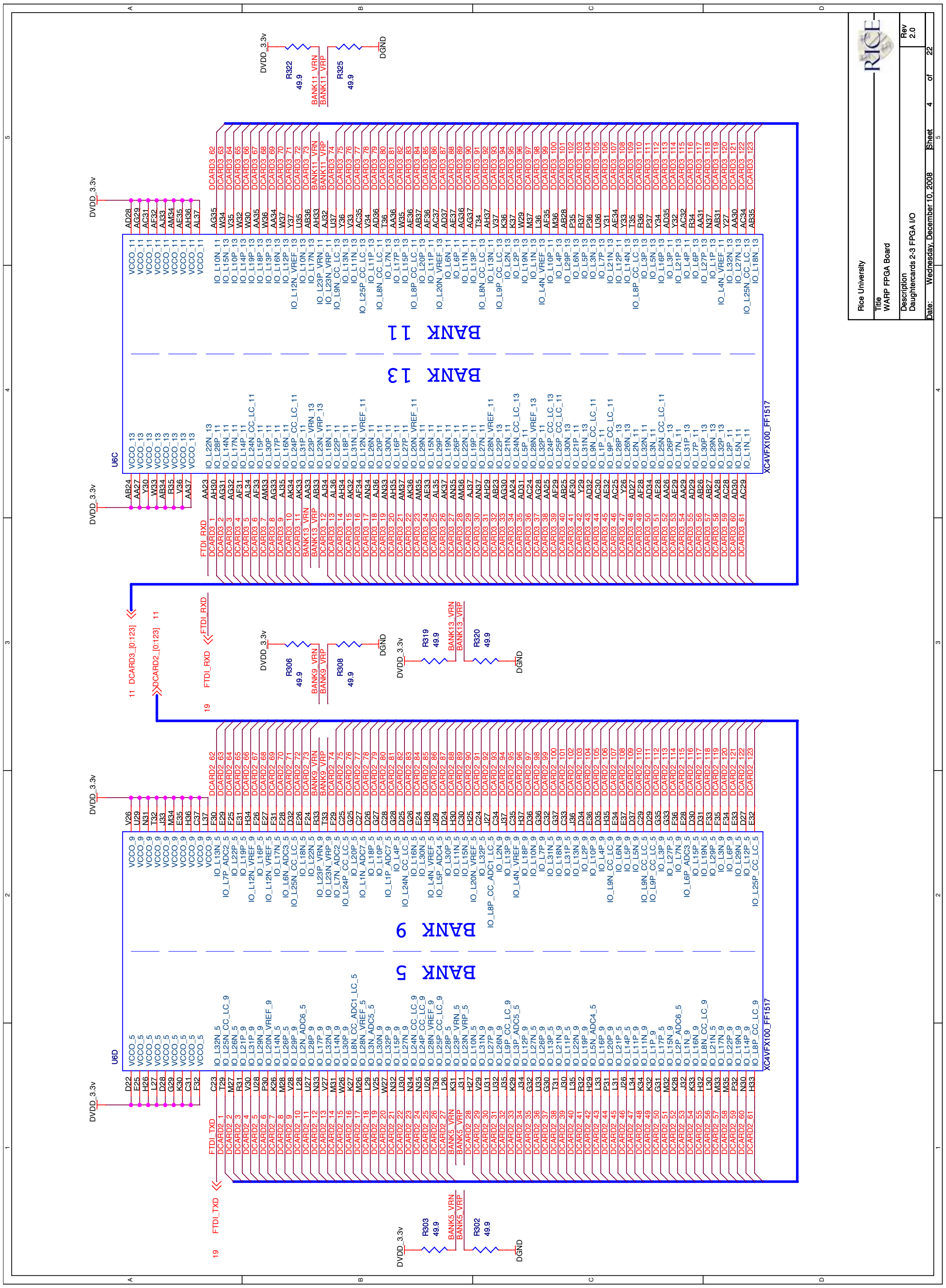
Description
<Doc>

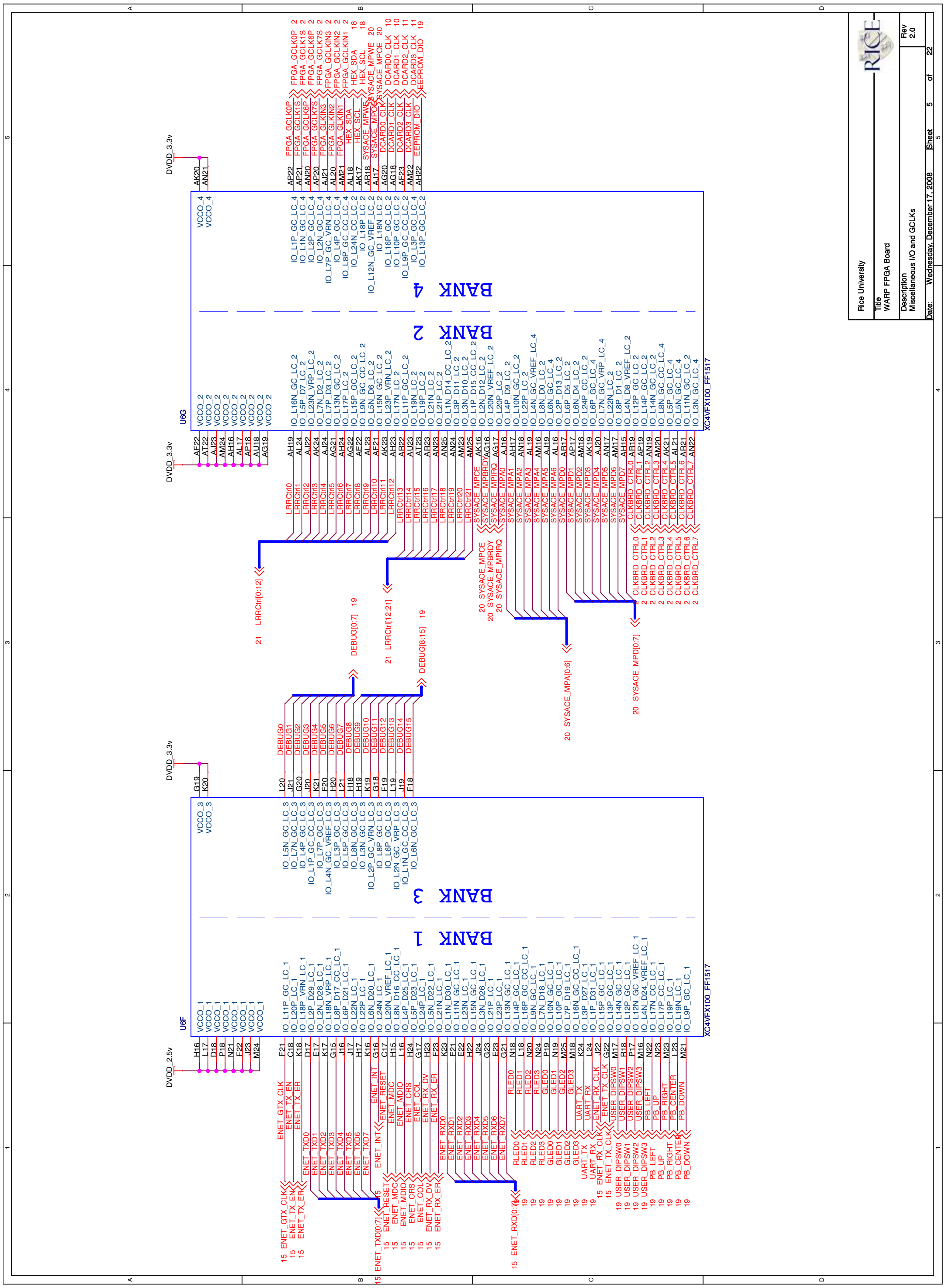
Rev
2.0

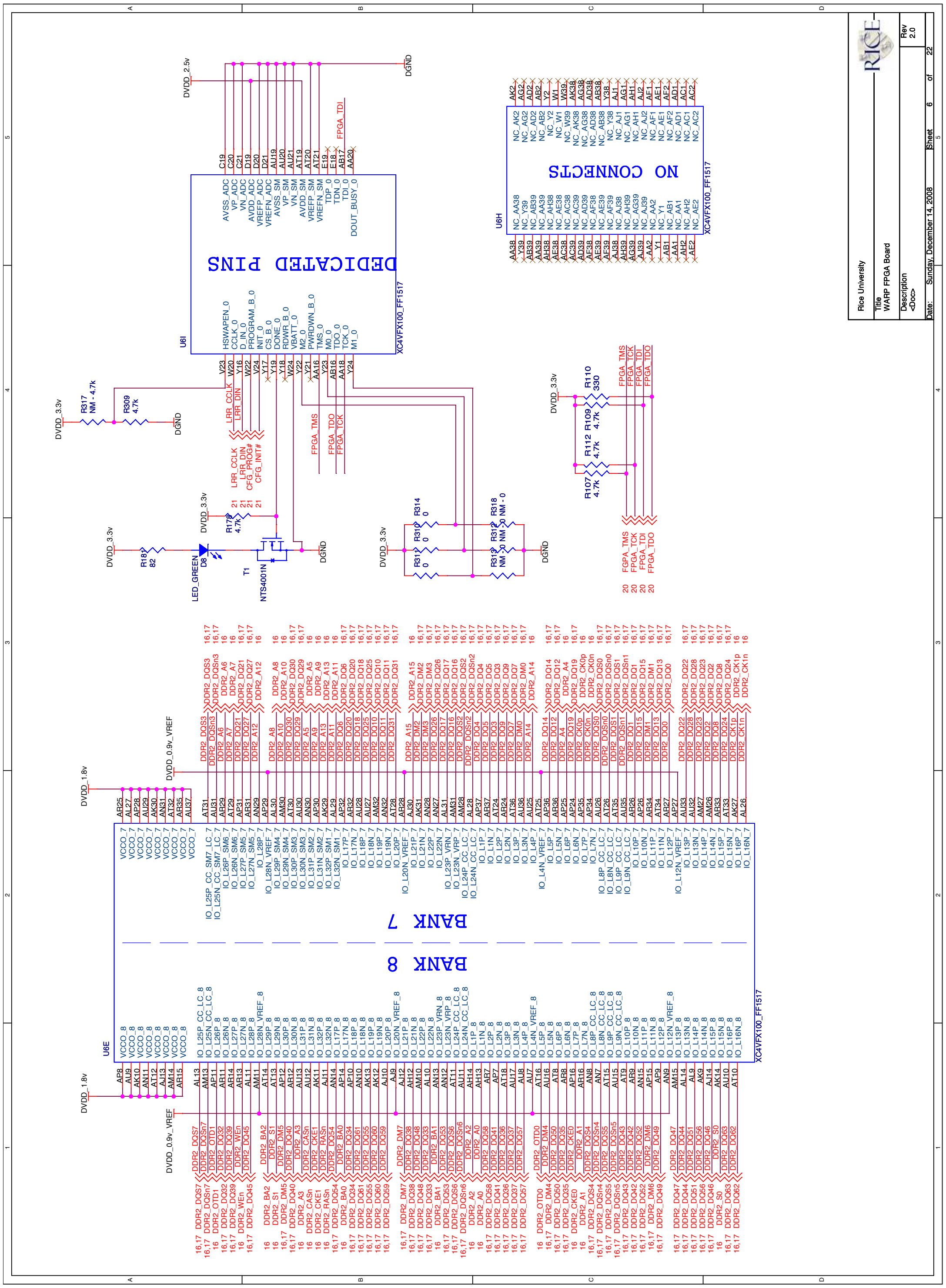
Date: Saturday, June 27, 2009 Sheet 1 of 22

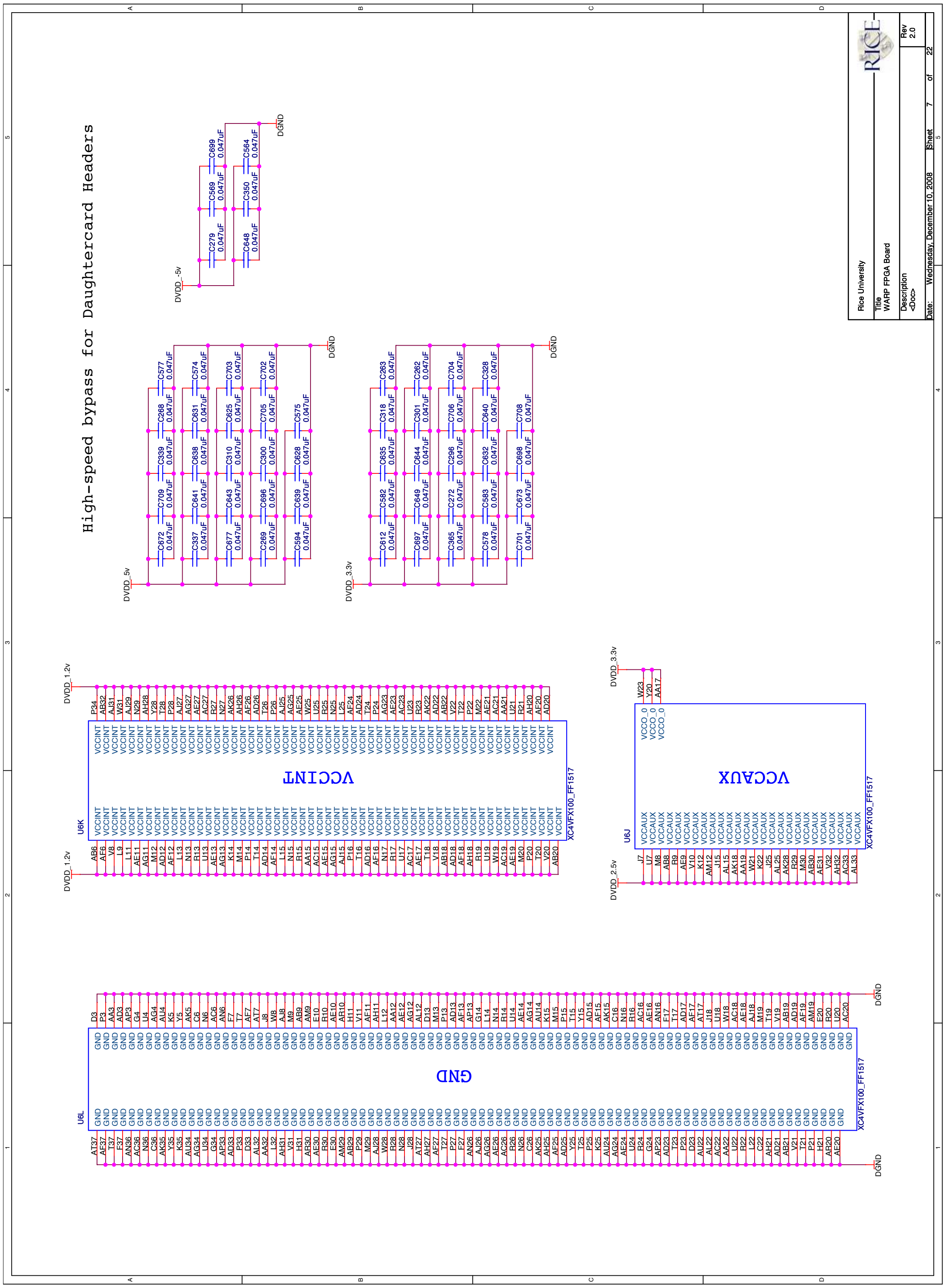

Y6
OE
VDD
OUT
GND
OSC 33MHz
DVDD_3.3v
DGND
R344 18.2
R347 33
R346 33
FPGA_GCLKIN3
FPGA_GCLKIN3 5
SYSACE_CLK33
SYSACE_CLK33 20
Y9
NM - OSC
R349 33
FPGA_GCLKIN2
FPGA_GCLKIN2 5
Y7
OSC 100MHz
R342 33
FPGA_GCLKIN1
FPGA_GCLKIN1 5
Y4
VCC
OE1
OE2
OUT
OUT
GND
NM - LVDS-PECL OSC 106.25MHz
L21 NM
L22
HZ0805E601R-10
DVDD_2.5v
Vdd_Y6
R291 1k
C236 0.47uF
C237 1nF
C233 4.7uF
R71 NM
R79 NM
R72 330
R80 330
C229 0.47uF
C232 0.47uF
MGT_CLK_Usr_p
MGT_CLK_Usr_n
MGT_CLK_USR_p 14
MGT_CLK_USR_n 14
Termination resistors required for PECL
J25
Clock Board Header
VDD
GND
GCLK0P
GCLK1S
GND
GND
GCLK6P
GCLK7S
GND
VDD
CONTROL0
CONTROL1
CONTROL2
GND
CONTROL3
CONTROL4
GND
CONTROL5
CONTROL6
CONTROL7
MHOLE
L10
L9
HZ0805E601R-10
R327 NM - 100
R326 NM - 100
FPGA_GCLK0P
FPGA_GCLK1S
FPGA_GCLK6P
FPGA_GCLK7S
5 FPGA_GCLK0P
5 FPGA_GCLK1S
5 FPGA_GCLK6P
5 FPGA_GCLK7S
CLKBRD_CTRL0
CLKBRD_CTRL1
CLKBRD_CTRL2
CLKBRD_CTRL3
CLKBRD_CTRL4
CLKBRD_CTRL5
CLKBRD_CTRL6
CLKBRD_CTRL7
CLKBRD_CTRL0 5
CLKBRD_CTRL1 5
CLKBRD_CTRL2 5
CLKBRD_CTRL3 5
CLKBRD_CTRL4 5
CLKBRD_CTRL5 5
CLKBRD_CTRL6 5
CLKBRD_CTRL7 5
Y2
NM - LVDS-PECL OSC 156.25MHz
L15 NM
L16
Vdd_Y3
R290 1k
C250 0.47uF
C240 1nF
C214 4.7uF
R66 NM
R75 NM
R65 330
R76 330
C227 0.47uF
C230 0.47uF
MGT_CLK_156_p
MGT_CLK_156_n
MGT_CLK_156_p 14
MGT_CLK_156_n 14
Y8
LVDS-PECL OSC 300MHz
L12 NM
L11
Vdd_Y5
R105 1k
C60 0.47uF
C59 1nF
C61 4.7uF
R104 NM
R103 NM
R102 330
R101 330
C630 0.47uF
C629 0.47uF
MGT_CLK_150_p
MGT_CLK_150_n
MGT_CLK_150_p 12
MGT_CLK_150_n 12
Y3
LVDS-PECL OSC 250MHz
L32 NM
L31
Vdd_Y7
R293 1k
C249 0.47uF
C239 1nF
C267 4.7uF
R68 NM
R78 NM
R70 330
R77 330
C228 0.47uF
C231 0.47uF
MGT_CLK_125_p
MGT_CLK_125_n
MGT_CLK_125_p 14
MGT_CLK_125_n 14
Rice University
Title
WARP FPGA Board
Description
<Doc>
Date: Sunday, June 28, 2009
Sheet 2 of 22
Rev 2.0
A
B
C
D
1
2
3
4
5

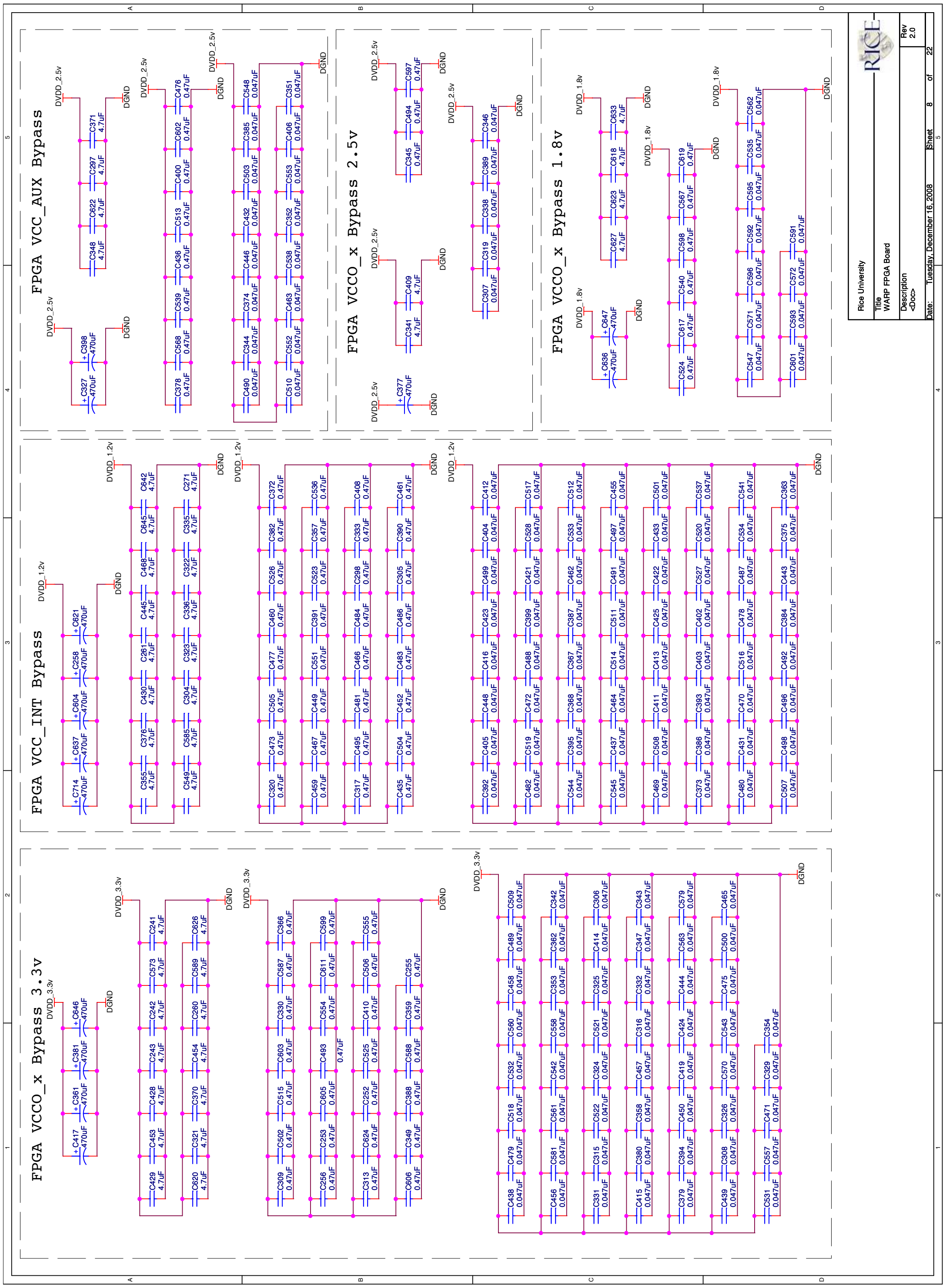

BANK 12
BANK 14
BANK 10
BANK 6
U6B
U6A
XC4VFX100_FF1517
DVDD_3.3v
DGND
R329 49.9
R328 49.9
BANK12_VRN
BANK12_VRP
R305 49.9
R307 49.9
BANK10_VRN
BANK10_VRP
R316 49.9
R315 49.9
BANK14_VRN
BANK14_VRP
R301 49.9
R299 49.9
BANK6_VRN
BANK6_VRP
DCARD0_[0:123]
DCARD1_[0:123]
DDR2_SCL
DDR2_SDA
Rice University
Title WARP FPGA Board
Description Daughtercards 0-1 FPGA I/O
Date: Wednesday, December 10, 2008
Sheet 3 of 22
Rev 2.0

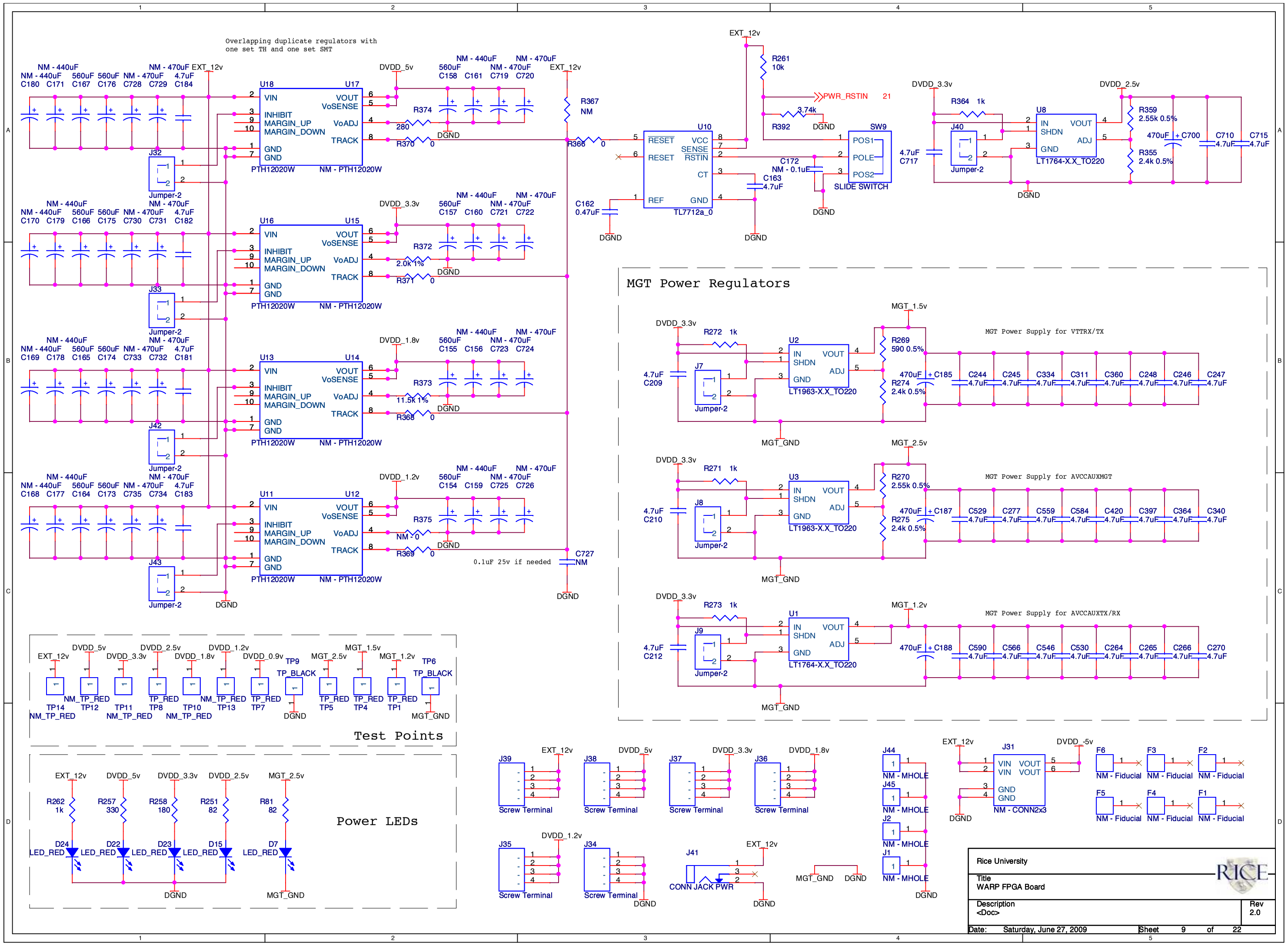

BANK 11
BANK 13
BANK 9
BANK 5
U6C
U6D
XC4VFX100_FF1517
DVDD_3.3v
DGND
R322 49.9
R325 49.9
R306 49.9
R308 49.9
R319 49.9
R320 49.9
R303 49.9
R302 49.9
BANK11_VRN
BANK11_VRP
BANK13_VRN
BANK13_VRP
BANK9_VRN
BANK9_VRP
BANK5_VRN
BANK5_VRP
FTDI_RXD
FTDI_TXD
DCARD3_[0:123]
DCARD2_[0:123]
Rice University
Title
WARP FPGA Board
Description
Daughtercards 2-3 FPGA I/O
Date: Wednesday, December 10, 2008
Sheet 4 of 22
Rev 2.0

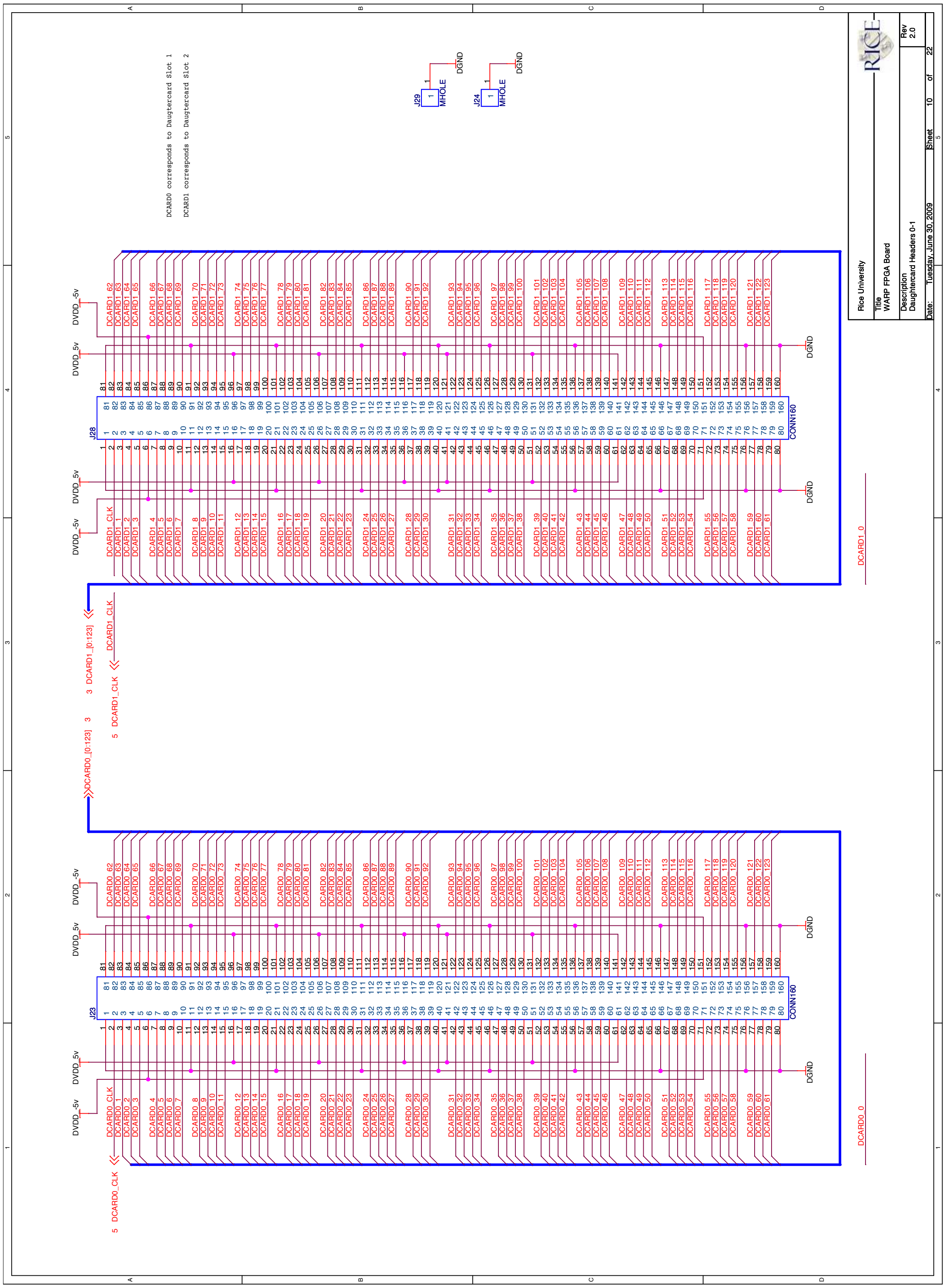

BANK 1
BANK 2
BANK 3
BANK 4
U6F
U6G
XC4VFX100_FF1517
DVDD_2.5v
DVDD_3.3v
LRRCtrl[0:12]
LRRCtrl[12:21]
DEBUG[0:7]
DEBUG[8:15]
SYSACE_MPA[0:6]
SYSACE_MPD[0:7]
ENET_TXD[0:7]
ENET_RXD[0:7]
Rice University
Title
WARP FPGA Board
Description
Miscellaneous I/O and GCLKs
Rev
2.0
Date: Wednesday, December 17, 2008
Sheet 5 of 22

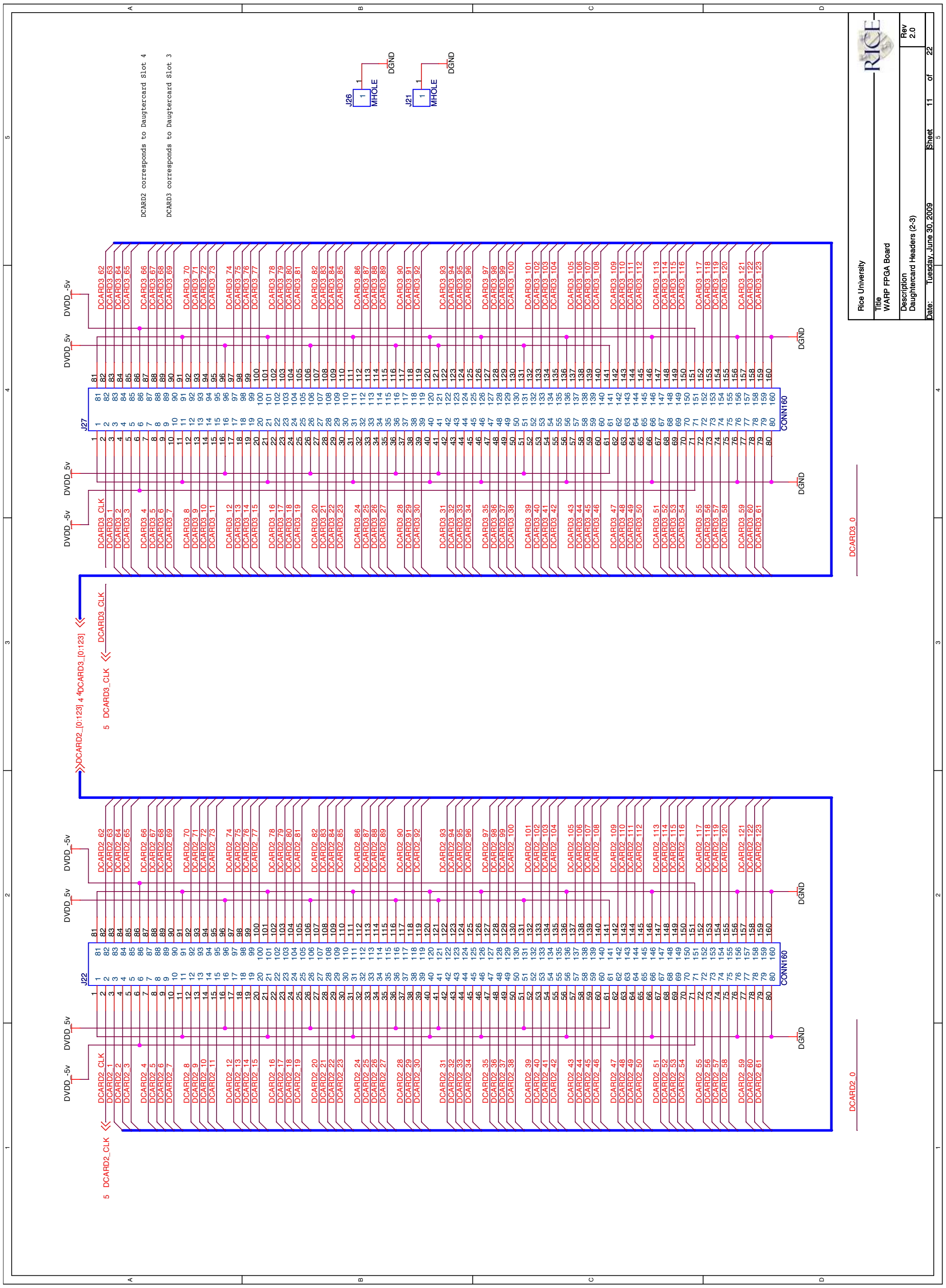

DEDICATED PINS
U6I
XC4VFX100_FF1517
HSWAPEN_0
CCLK_0
D_IN_0
PROGRAM_B_0
INIT_0
CS_B_0
DONE_0
RDWR_B_0
VBATT_0
M2_0
PWRDWN_B_0
TMS_0
M0_0
TDO_0
TCK_0
M1_0
AVSS_ADC
VP_ADC
VN_ADC
AVDD_ADC
VREFP_ADC
VREFN_ADC
AVSS_SM
VP_SM
VN_SM
AVDD_SM
VREFP_SM
VREFN_SM
TDP_0
TDN_0
TDI_0
DOUT_BUSY_0
FPGA_TDI
DVDD_2.5v
DGND
NO CONNECTS
U6H
XC4VFX100_FF1517
DVDD_3.3v
R317
NM - 4.7k
R309
4.7k
LRR_CCLK
LRR_DIN
CFG_PROG#
CFG_INIT#
FPGA_TMS
FPGA_TDO
FPGA_TCK
R178
4.7k
R18
82
D8
LED_GREEN
T1
NTS4001N
R314
0
R318
NM - 0
R310
0
R312
NM 0
R311
0
R313
NM 0
R110
330
R109
4.7k
R112
4.7k
R107
4.7k
FPGA_TMS
FPGA_TCK
FPGA_TDI
FPGA_TDO
BANK 7
BANK 8
U6E
XC4VFX100_FF1517
DVDD_1.8v
DVDD_0.9v_VREF
Rice University
Title
WARP FPGA Board
Description
<Doc>
Rev
2.0
Date: Sunday, December 14, 2008
Sheet 6 of 22

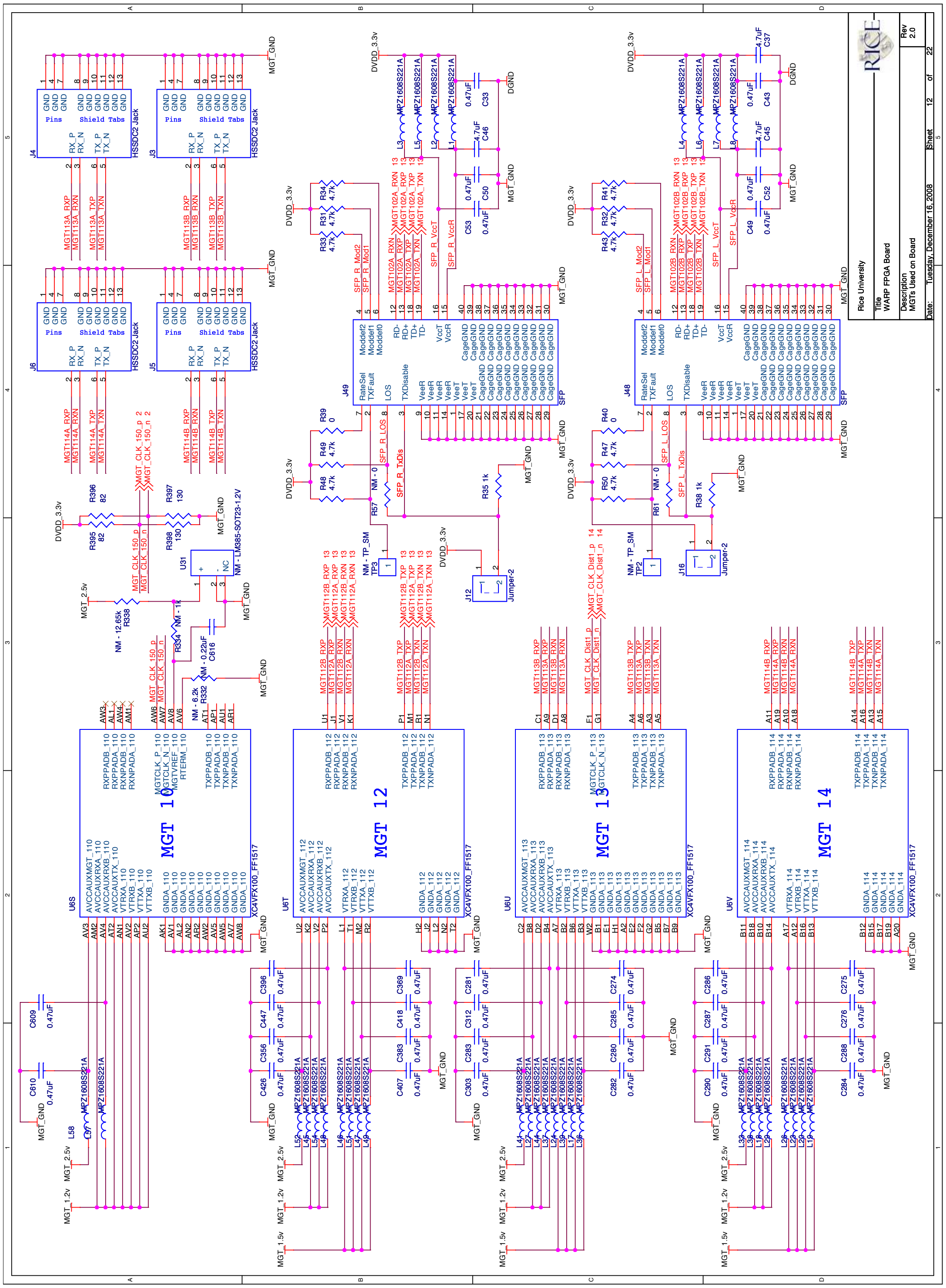

High-speed bypass for Daughtercard Headers
DVDD_5v
C672 0.047uF
C709 0.047uF
C339 0.047uF
C268 0.047uF
C577 0.047uF
C337 0.047uF
C641 0.047uF
C638 0.047uF
C631 0.047uF
C574 0.047uF
C677 0.047uF
C643 0.047uF
C310 0.047uF
C625 0.047uF
C703 0.047uF
C269 0.047uF
C696 0.047uF
C300 0.047uF
C705 0.047uF
C702 0.047uF
C594 0.047uF
C639 0.047uF
C628 0.047uF
C575 0.047uF
DGND
DVDD_3.3v
C612 0.047uF
C582 0.047uF
C635 0.047uF
C318 0.047uF
C263 0.047uF
C697 0.047uF
C649 0.047uF
C644 0.047uF
C301 0.047uF
C262 0.047uF
C365 0.047uF
C272 0.047uF
C296 0.047uF
C706 0.047uF
C704 0.047uF
C578 0.047uF
C583 0.047uF
C632 0.047uF
C640 0.047uF
C328 0.047uF
C701 0.047uF
C673 0.047uF
C668 0.047uF
C708 0.047uF
DGND
DVDD_-5v
C279 0.047uF
C569 0.047uF
C699 0.047uF
C648 0.047uF
C350 0.047uF
C564 0.047uF
DGND
U6K
VCCINT
XC4VFX100_FF1517
DVDD_1.2v
DVDD_1.2v
U6J
VCCAUX
VCCO_0
XC4VFX100_FF1517
DVDD_3.3v
DVDD_2.5v
U6L
GND
XC4VFX100_FF1517
DGND
DGND
Rice University
Title
WARP FPGA Board
Description
<Doc>
Rev
2.0
Date: Wednesday, December 10, 2008
Sheet 7 of 22
A
B
C
D
1
2
3
4
5

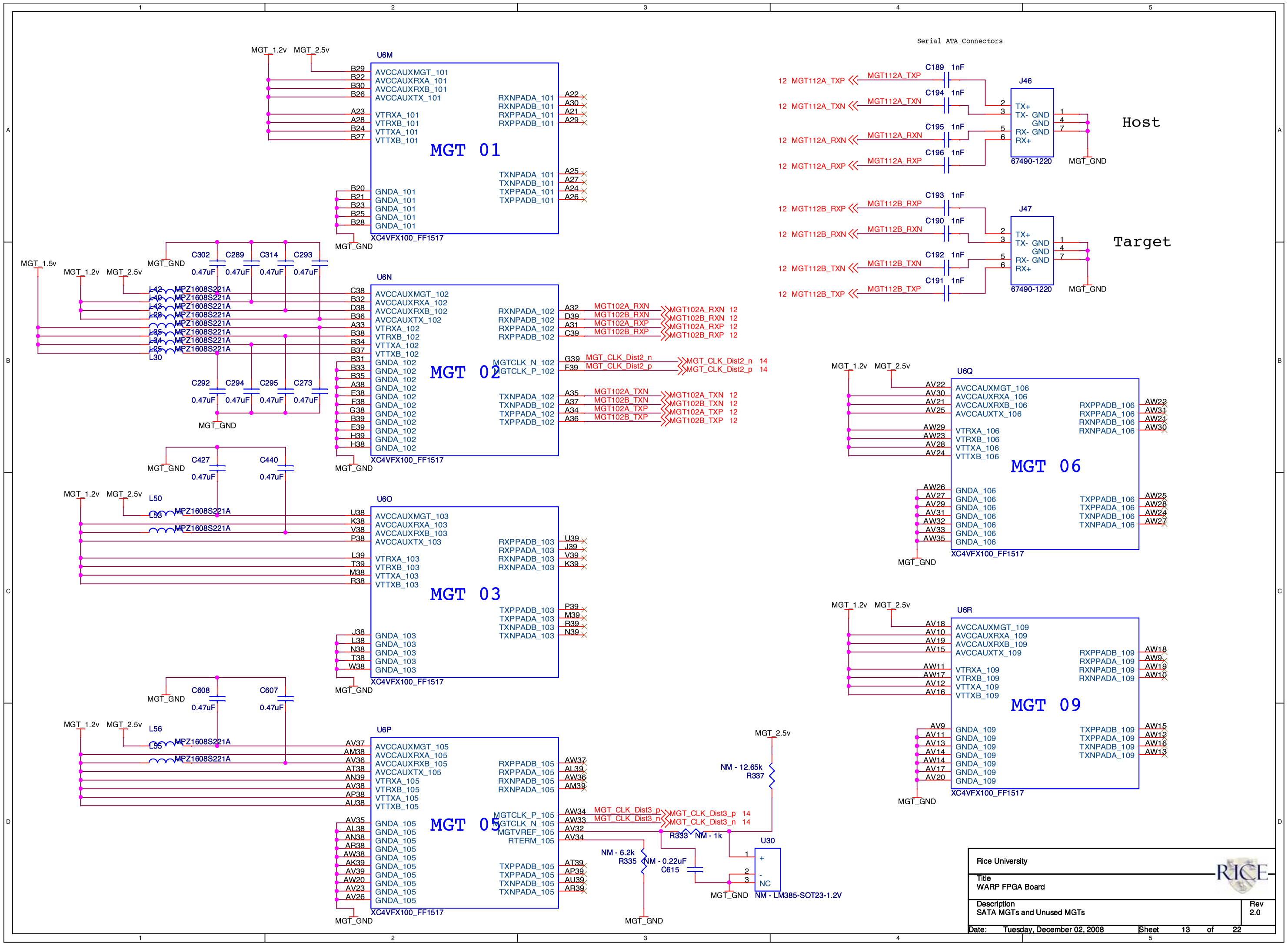

FPGA VCC_AUX Bypass
FPGA VCCO_x Bypass 2.5v
FPGA VCCO_x Bypass 1.8v
FPGA VCC_INT Bypass
FPGA VCCO_x Bypass 3.3v
DVDD_2.5v
DVDD_1.8v
DVDD_1.2v
DVDD_3.3v
DGND
470uF
4.7uF
0.47uF
0.047uF
Rice University
Title
WARP FPGA Board
Description
<Doc>
Date: Tuesday, December 16, 2008
Sheet 8 of 22
Rev
2.0
A
B
C
D
1
2
3
4
5

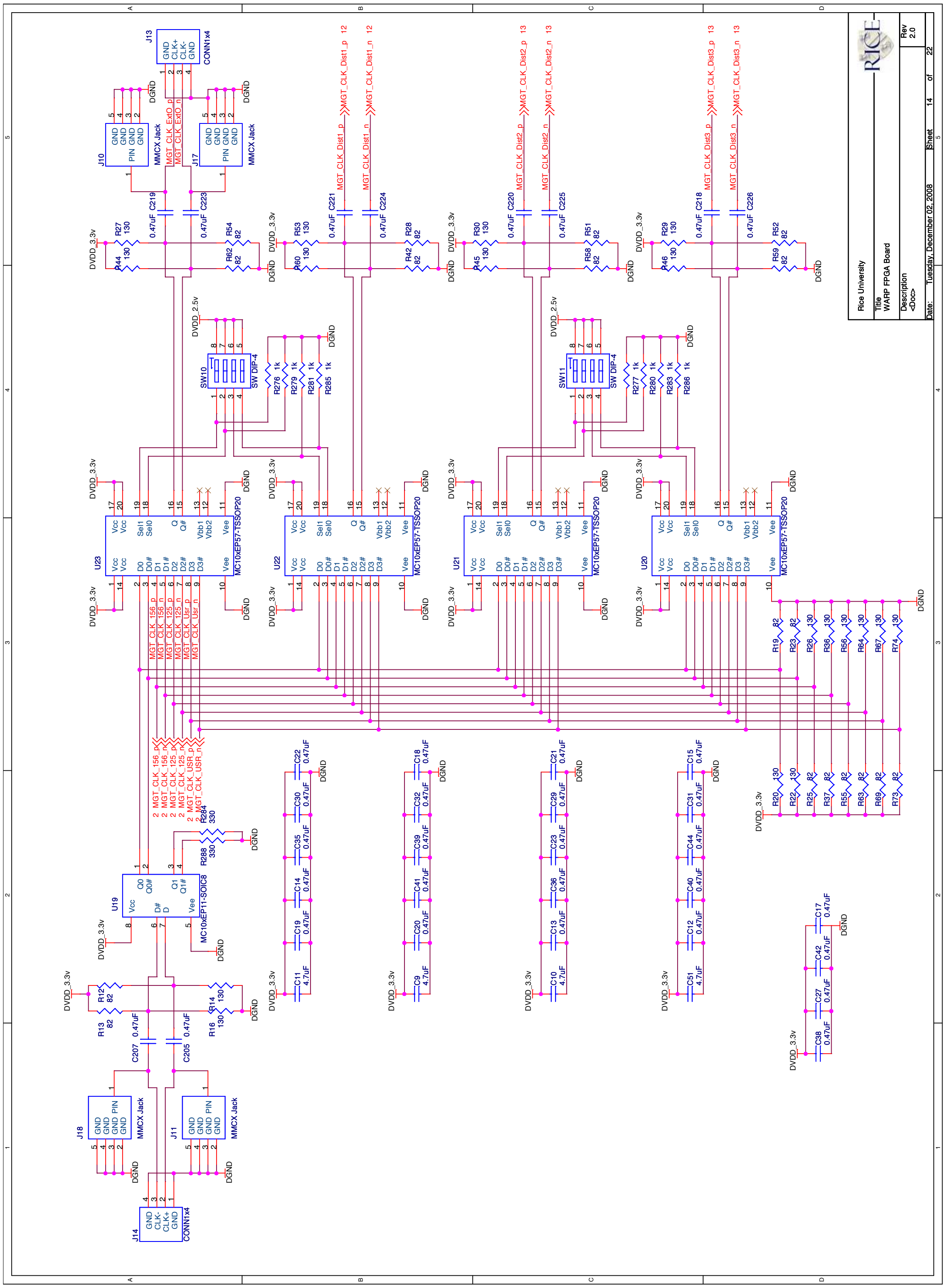

Overlapping duplicate regulators with one set TH and one set SMT
U18
U17
VIN
VOUT
VoSENSE
INHIBIT
MARGIN_UP
MARGIN_DOWN
VoADJ
TRACK
GND
PTH12020W
NM - PTH12020W
U16
U15
U13
U14
U11
U12
J32
J33
J42
J43
Jumper-2
DVDD_5v
DVDD_3.3v
DVDD_1.8v
DVDD_1.2v
EXT_12v
R374 280
R370 0
R372 2.0k 1%
R371 0
R373 11.5k 1%
R368 0
R375 NM - 0
R369 0
R367 NM
R366 0
C727 NM
0.1uF 25v if needed
NM - 440uF
560uF
NM - 470uF
4.7uF
C180 C171 C167 C176 C728 C729 C184
C170 C179 C166 C175 C730 C731 C182
C169 C178 C165 C174 C733 C732 C181
C168 C177 C164 C173 C735 C734 C183
C158 C161 C719 C720
C157 C160 C721 C722
C155 C156 C723 C724
C154 C159 C725 C726
DGND
R261 10k
PWR_RSTIN 21
R392 3.74k
U10
RESET
VCC
SENSE
RSTIN
CT
REF
GND
TL7712a_0
C172 NM - 0.1uF
C163 4.7uF
C162 0.47uF
SW9
POS1
POLE
POS2
SLIDE SWITCH
R364 1k
J40
C717 4.7uF
U8
IN
SHDN
ADJ
LT1764-X.X_TO220
DVDD_2.5v
R359 2.55k 0.5%
R355 2.4k 0.5%
470uF C700
C710 4.7uF
C715 4.7uF
MGT Power Regulators
MGT Power Supply for VTTRX/TX
MGT Power Supply for AVCCAUXMGT
MGT Power Supply for AVCCAUXTX/RX
MGT_1.5v
MGT_2.5v
MGT_1.2v
MGT_GND
R272 1k
R271 1k
R273 1k
J7
J8
J9
U2
U3
U1
LT1963-X.X_TO220
R269 590 0.5%
R274 2.4k 0.5%
R270 2.55k 0.5%
R275 2.4k 0.5%
470uF C185
470uF C187
470uF C188
C209 4.7uF
C210 4.7uF
C212 4.7uF
C244 C245 C334 C311 C360 C248 C246 C247
C529 C277 C559 C584 C420 C397 C364 C340
C590 C566 C546 C530 C264 C265 C266 C270
Test Points
TP14 NM_TP_RED
TP12 NM_TP_RED
TP11 NM_TP_RED
TP8 TP_RED
TP10 NM_TP_RED
TP13 NM_TP_RED
TP7 TP_RED
TP9 TP_BLACK
TP5 TP_RED
TP4 TP_RED
TP1 TP_RED
TP6 TP_BLACK
DVDD_0.9v
Power LEDs
R262 1k
R257 330
R258 180
R251 82
R81 82
D24 D22 D23 D15 D7
LED_RED
J39 J38 J37 J36 J35 J34
Screw Terminal
J41
CONN JACK PWR
J44 J45 J2 J1
NM - MHOLE
J31
NM - CONN2x3
DVDD_-5v
F6 F3 F2 F5 F4 F1
NM - Fiducial
Rice University
Title
WARP FPGA Board
Description
<Doc>
Rev
2.0
Date: Saturday, June 27, 2009
Sheet 9 of 22
RICE
A B C D
1 2 3 4 5

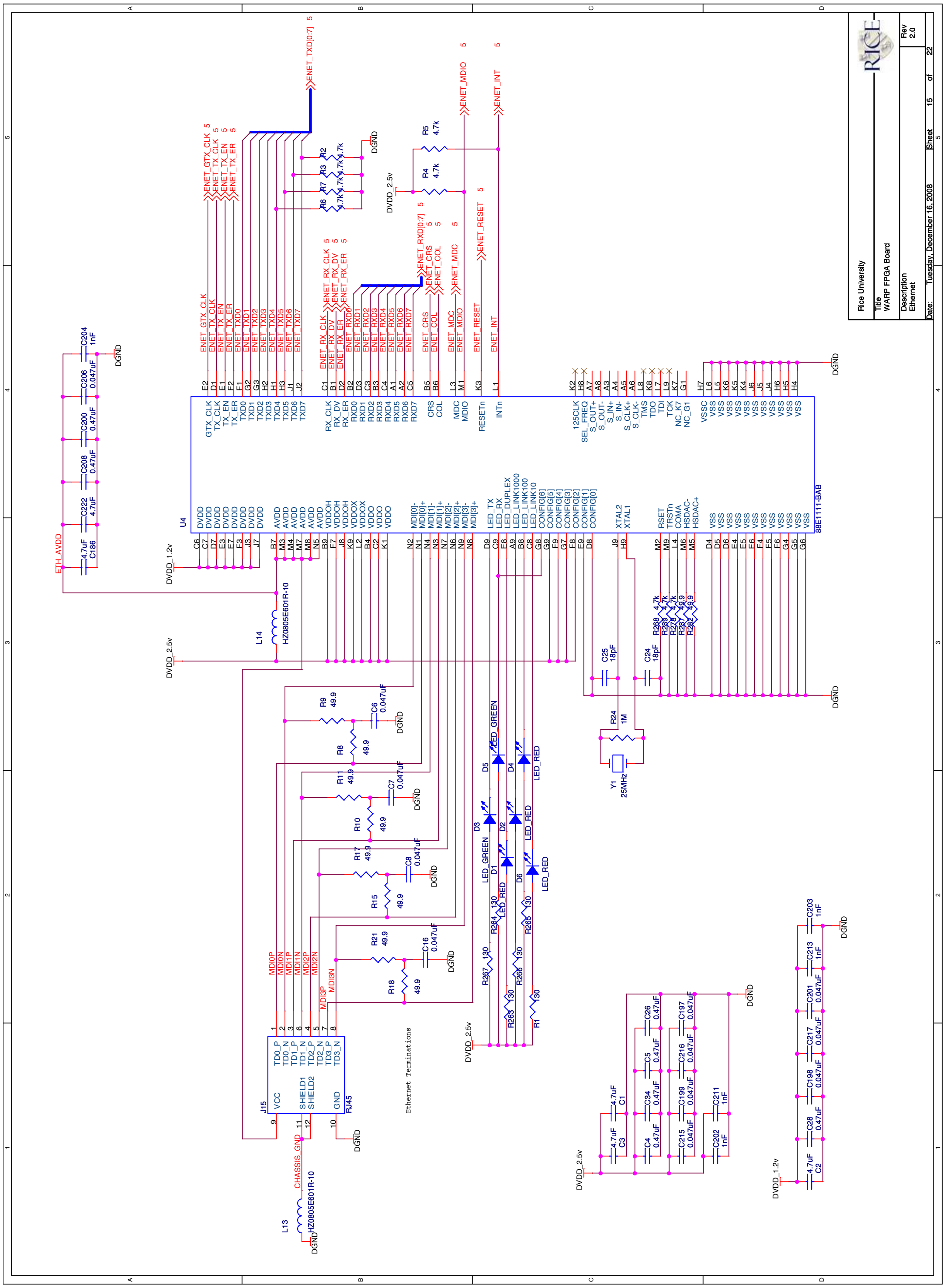

DCARD0 corresponds to Daugtercard Slot 1
DCARD1 corresponds to Daugtercard Slot 2
J29
1
MHOLE
1
DGND
J24
1
MHOLE
1
DGND
J28
CONN160
J23
CONN160
DVDD_5v
DGND
DCARD1_CLK
DCARD0_CLK
5 DCARD1_CLK
5 DCARD0_CLK
3 DCARD1_[0:123]
DCARD0_[0:123] 3
DCARD1_0
DCARD0_0
DCARD1_1 … DCARD1_123
DCARD0_1 … DCARD0_123
1 … 160
Rice University
Title
WARP FPGA Board
Description
Daughtercard Headers 0-1
Date: Tuesday, June 30, 2009
Sheet 10 of 22
Rev
2.0
A
B
C
D
1
2
3
4
5

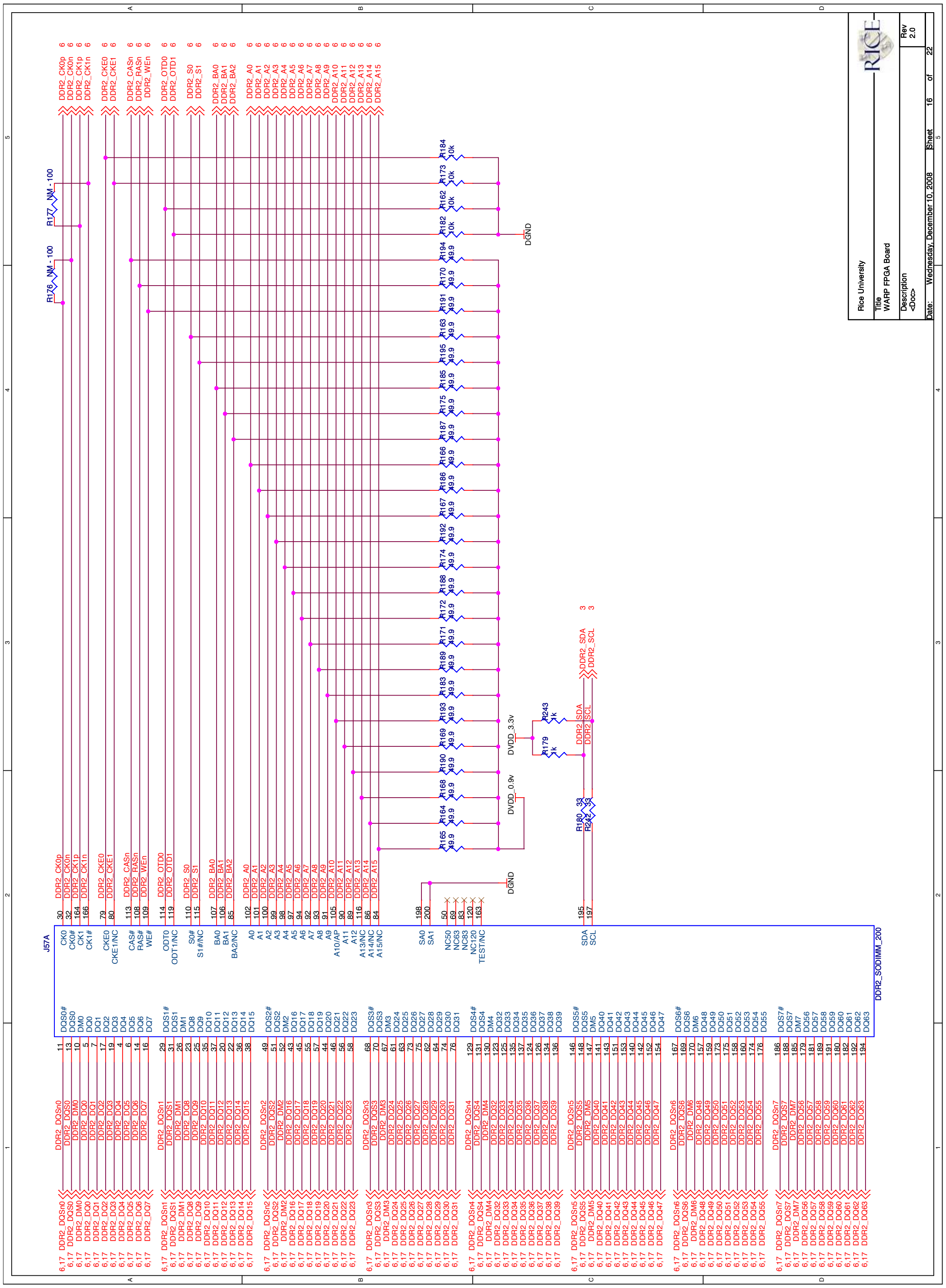

DCARD2 corresponds to Daugtercard Slot 4
DCARD3 corresponds to Daugtercard Slot 3
J26
J21
1
MHOLE
DGND
J27
J22
CONN160
DVDD_5v
DVDD_-5v
DCARD3_CLK
DCARD2_CLK
DCARD3_[0:123]
DCARD2_[0:123]
DCARD3_0
DCARD2_0
Rice University
Title
WARP FPGA Board
Description
Daughtercard Headers (2-3)
Rev
2.0
Date: Tuesday, June 30, 2009
Sheet 11 of 22

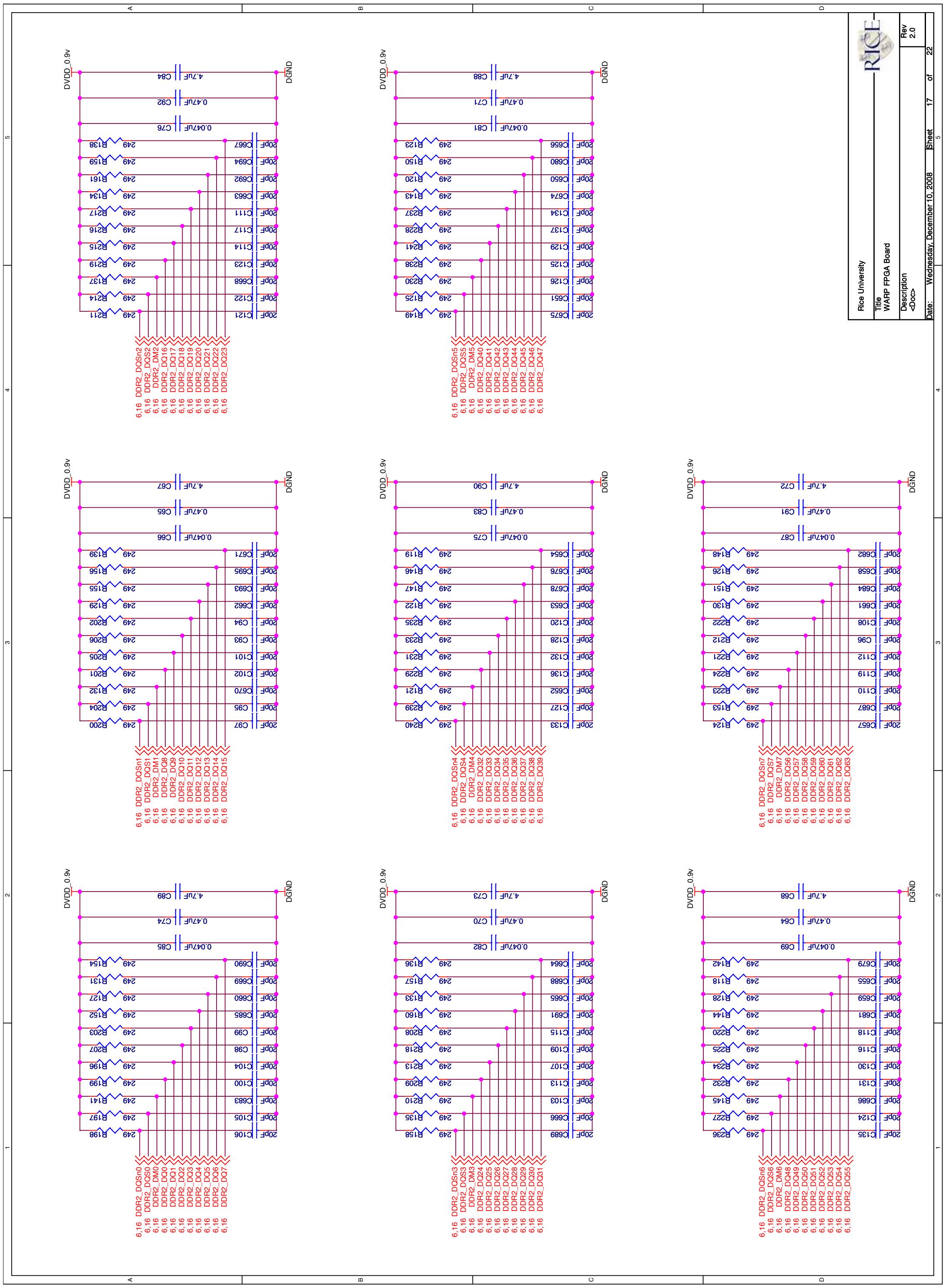

Rice University
Title
WARP FPGA Board
Description
MGTs Used on Board
Date: Tuesday, December 16, 2008
Sheet 12 of 22
Rev 2.0
MGT 10
MGT 12
MGT 13
MGT 14
U6S
U6T
U6U
U6V
XC4VFX100_FF1517
J3
J4
J5
J6
HSSDC2 Jack
J48
J49
SFP
J12
J16
Jumper-2
U31
NM - LM385-SOT23-1.2V
DVDD_3.3v
MGT_GND
DGND
MGT_1.2v
MGT_2.5v
MGT_1.5v

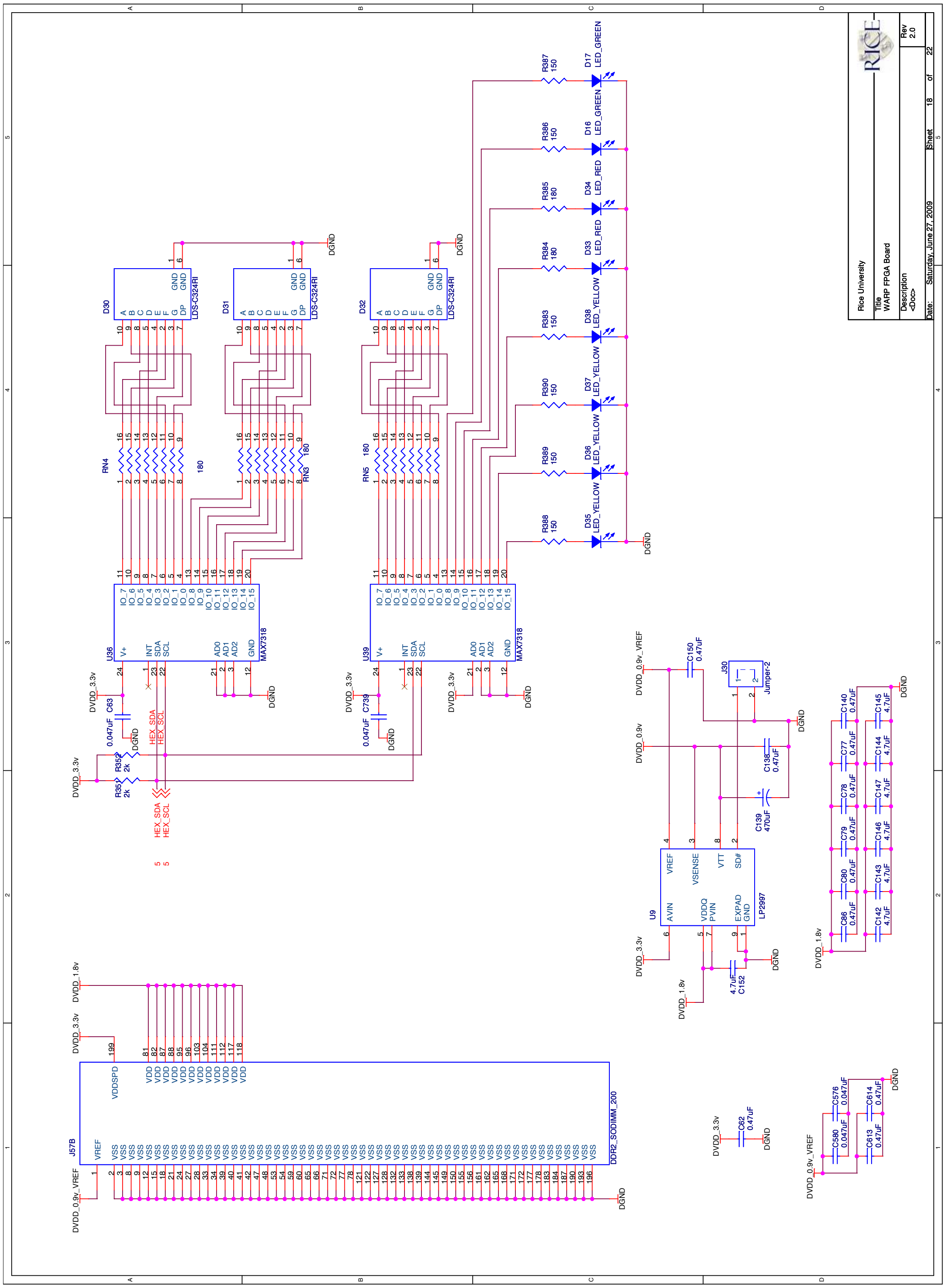

Serial ATA Connectors
Host
Target
J46
J47
67490-1220
TX+
TX-
GND
RX-
RX+
MGT_GND
C189 1nF
C194 1nF
C195 1nF
C196 1nF
C193 1nF
C190 1nF
C192 1nF
C191 1nF
12 MGT112A_TXP
12 MGT112A_TXN
12 MGT112A_RXN
12 MGT112A_RXP
12 MGT112B_RXP
12 MGT112B_RXN
12 MGT112B_TXN
12 MGT112B_TXP
MGT 01
U6M
AVCCAUXMGT_101
AVCCAUXRXA_101
AVCCAUXRXB_101
AVCCAUXTX_101
VTRXA_101
VTRXB_101
VTTXA_101
VTTXB_101
GNDA_101
RXNPADA_101
RXNPADB_101
RXPPADA_101
RXPPADB_101
TXNPADA_101
TXNPADB_101
TXPPADA_101
TXPPADB_101
XC4VFX100_FF1517
MGT 02
U6N
AVCCAUXMGT_102
AVCCAUXRXA_102
AVCCAUXRXB_102
AVCCAUXTX_102
VTRXA_102
VTRXB_102
VTTXA_102
VTTXB_102
GNDA_102
RXNPADA_102
RXNPADB_102
RXPPADA_102
RXPPADB_102
MGTCLK_N_102
MGTCLK_P_102
TXNPADA_102
TXNPADB_102
TXPPADA_102
TXPPADB_102
MGT102A_RXN 12
MGT102B_RXN 12
MGT102A_RXP 12
MGT102B_RXP 12
MGT_CLK_Dist2_n 14
MGT_CLK_Dist2_p 14
MGT102A_TXN 12
MGT102B_TXN 12
MGT102A_TXP 12
MGT102B_TXP 12
C302 C289 C314 C293 0.47uF
C292 C294 C295 C273 0.47uF
L42 L40 L43 L28 L35 L34 L25 L30
MPZ1608S221A
MGT_1.5v
MGT_1.2v
MGT_2.5v
MGT 03
U6O
AVCCAUXMGT_103
AVCCAUXRXA_103
AVCCAUXRXB_103
AVCCAUXTX_103
VTRXA_103
VTRXB_103
VTTXA_103
VTTXB_103
GNDA_103
RXPPADB_103
RXPPADA_103
RXNPADB_103
RXNPADA_103
TXPPADB_103
TXPPADA_103
TXNPADB_103
TXNPADA_103
C427 C440 0.47uF
L50 L36
MGT 05
U6P
AVCCAUXMGT_105
AVCCAUXRXA_105
AVCCAUXRXB_105
AVCCAUXTX_105
VTRXA_105
VTRXB_105
VTTXA_105
VTTXB_105
GNDA_105
RXPPADB_105
RXPPADA_105
RXNPADB_105
RXNPADA_105
MGTCLK_P_105
MGTCLK_N_105
MGTVREF_105
RTERM_105
TXPPADB_105
TXPPADA_105
TXNPADB_105
TXNPADA_105
MGT_CLK_Dist3_p 14
MGT_CLK_Dist3_n 14
C608 C607 0.47uF
L56 L55
NM - 12.65k R337
R333 NM - 1k
NM - 6.2k R335
NM - 0.22uF C615
U30
NC
NM - LM385-SOT23-1.2V
MGT 06
U6Q
AVCCAUXMGT_106
AVCCAUXRXA_106
AVCCAUXRXB_106
AVCCAUXTX_106
VTRXA_106
VTRXB_106
VTTXA_106
VTTXB_106
GNDA_106
RXPPADB_106
RXPPADA_106
RXNPADB_106
RXNPADA_106
TXPPADB_106
TXPPADA_106
TXNPADB_106
TXNPADA_106
MGT 09
U6R
AVCCAUXMGT_109
AVCCAUXRXA_109
AVCCAUXRXB_109
AVCCAUXTX_109
VTRXA_109
VTRXB_109
VTTXA_109
VTTXB_109
GNDA_109
RXPPADB_109
RXPPADA_109
RXNPADB_109
RXNPADA_109
TXPPADB_109
TXPPADA_109
TXNPADB_109
TXNPADA_109
Rice University
Title
WARP FPGA Board
Description
SATA MGTs and Unused MGTs
Rev 2.0
Date: Tuesday, December 02, 2008
Sheet 13 of 22

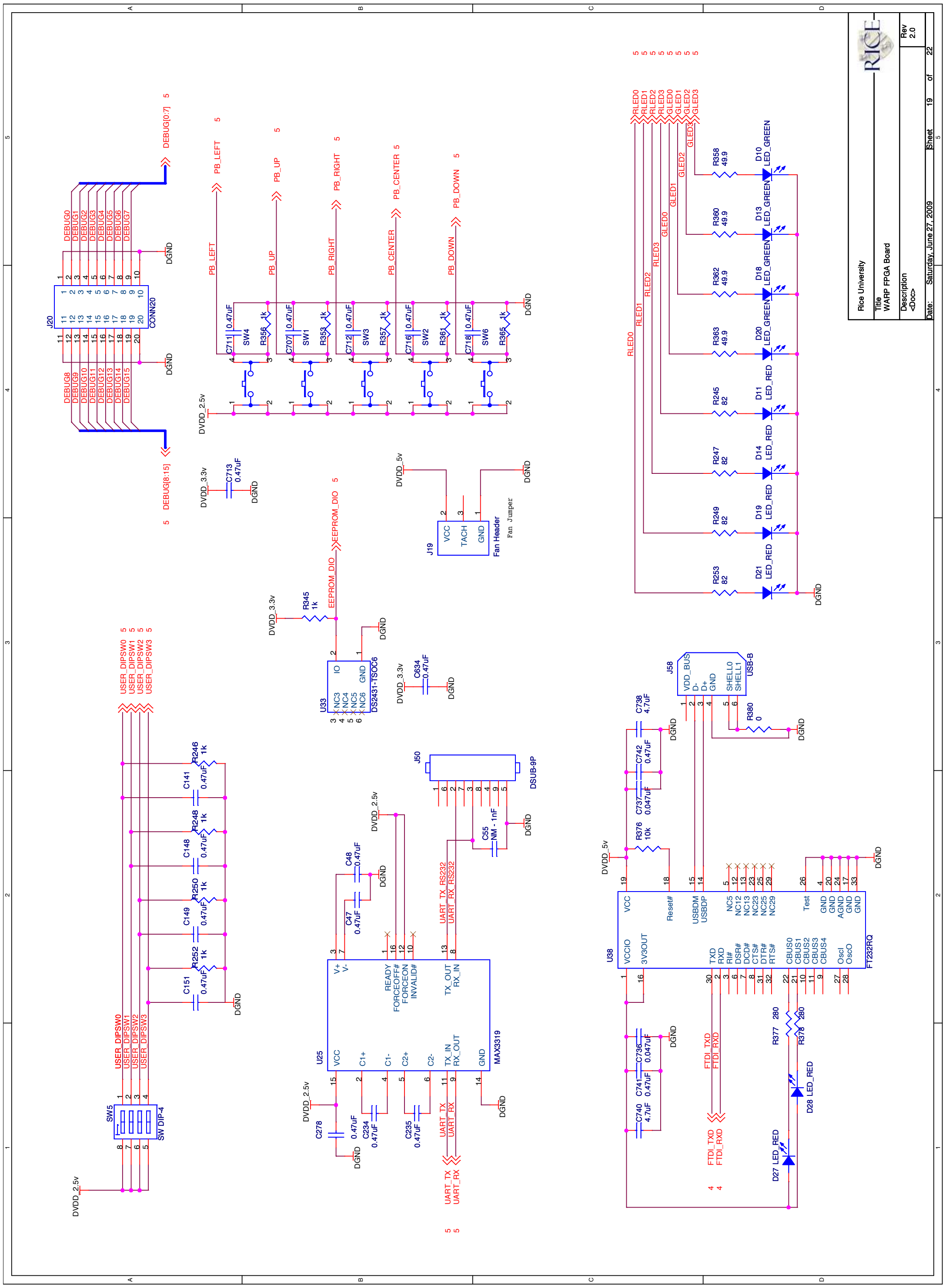

Rice University
Title
WARP FPGA Board
Description
<Doc>
Date: Tuesday, December 02, 2008
Sheet 14 of 22
Rev 2.0
J13 CONN1x4
J14 CONN1x4
J10 MMCX Jack
J17 MMCX Jack
J18 MMCX Jack
J11 MMCX Jack
MGT_CLK_ExtO_p
MGT_CLK_ExtO_n
MGT_CLK_Dist1_p 12
MGT_CLK_Dist1_n 12
MGT_CLK_Dist2_p 13
MGT_CLK_Dist2_n 13
MGT_CLK_Dist3_p 13
MGT_CLK_Dist3_n 13
SW10 SW DIP-4
SW11 SW DIP-4
U23 MC10xEP57-TSSOP20
U22 MC10xEP57-TSSOP20
U21 MC10xEP57-TSSOP20
U20 MC10xEP57-TSSOP20
U19 MC10xEP11-SOIC8
MGT_CLK_156_p
MGT_CLK_156_n
MGT_CLK_125_p
MGT_CLK_125_n
MGT_CLK_Usr_p
MGT_CLK_Usr_n
DVDD_3.3v
DVDD_2.5v
DGND
0.47uF
4.7uF

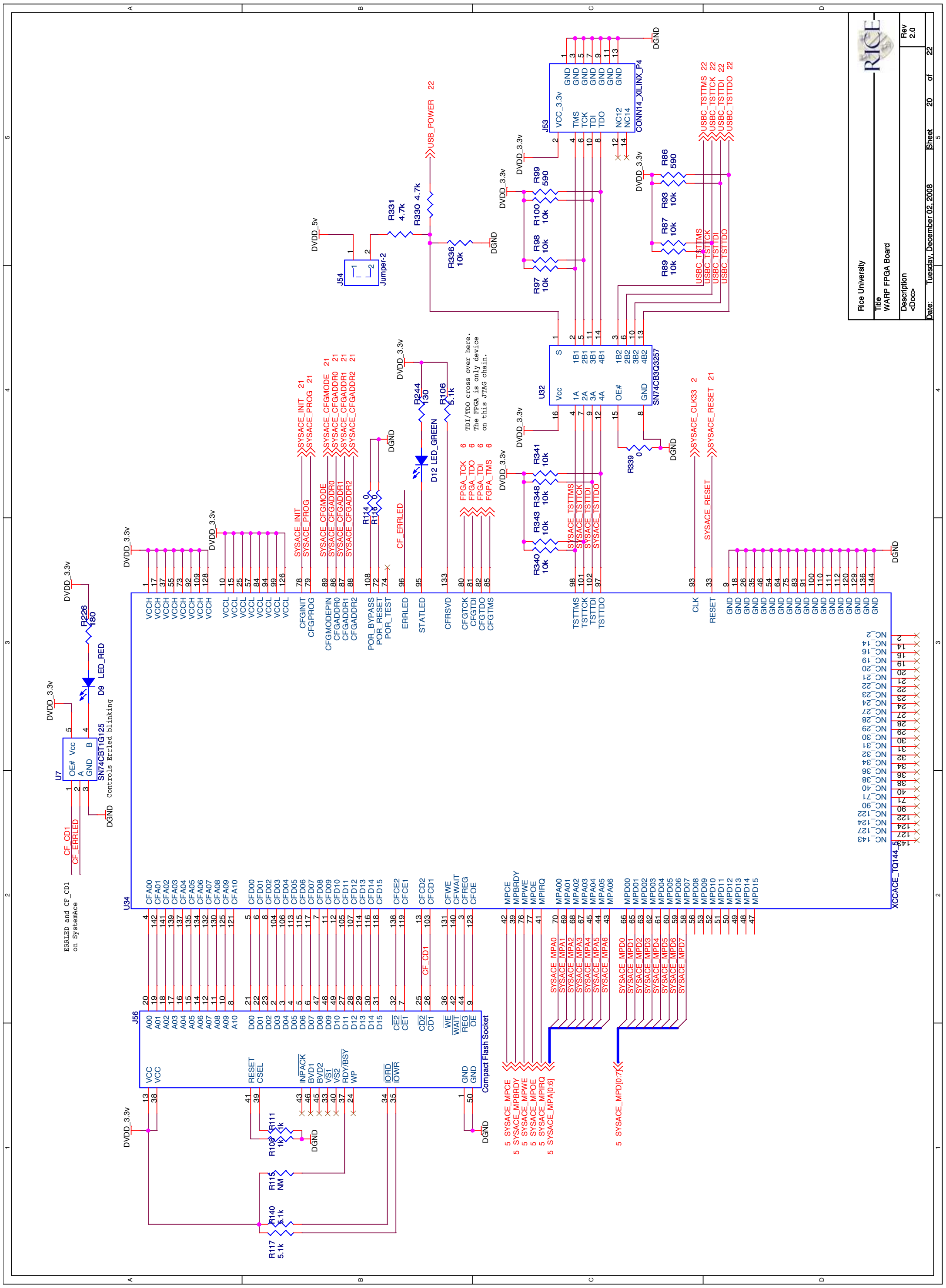

U4
88E1111-BAB
GTX_CLK
TX_CLK
TX_EN
TX_ER
TXD0
TXD1
TXD2
TXD3
TXD4
TXD5
TXD6
TXD7
RX_CLK
RX_DV
RX_ER
RXD0
RXD1
RXD2
RXD3
RXD4
RXD5
RXD6
RXD7
CRS
COL
MDC
MDIO
RESETn
INTn
125CLK
SEL_FREQ
S_OUT+
S_OUT-
S_IN+
S_IN-
S_CLK+
S_CLK-
TMS
TDO
TDI
TCK
NC_K7
NC_G1
VSSC
VSS
DVDD
AVDD
VDDOH
VDDOX
VDDO
MDI[0]-
MDI[0]+
MDI[1]+
MDI[1]-
MDI[2]-
MDI[2]+
MDI[3]-
MDI[3]+
LED_TX
LED_RX
LED_DUPLEX
LED_LINK1000
LED_LINK100
LED_LINK10
CONFIG[6]
CONFIG[5]
CONFIG[4]
CONFIG[3]
CONFIG[2]
CONFIG[1]
CONFIG[0]
XTAL2
XTAL1
RSET
TRSTn
COMA
HSDAC-
HSDAC+
ENET_GTX_CLK 5
ENET_TX_CLK 5
ENET_TX_EN 5
ENET_TX_ER 5
ENET_TXD[0:7] 5
ENET_TXD0
ENET_TXD1
ENET_TXD2
ENET_TXD3
ENET_TXD4
ENET_TXD5
ENET_TXD6
ENET_TXD7
ENET_RX_CLK 5
ENET_RX_DV 5
ENET_RX_ER 5
ENET_RXD[0:7] 5
ENET_RXD0
ENET_RXD1
ENET_RXD2
ENET_RXD3
ENET_RXD4
ENET_RXD5
ENET_RXD6
ENET_RXD7
ENET_CRS 5
ENET_COL 5
ENET_MDC 5
ENET_MDIO 5
ENET_RESET 5
ENET_INT 5
R2 4.7k
R3 4.7k
R7 4.7k
R6 4.7k
R4 4.7k
R5 4.7k
DVDD_2.5v
DVDD_1.2v
DGND
ETH_AVDD
C186 4.7uF
C222 4.7uF
C208 0.47uF
C200 0.47uF
C206 0.047uF
C204 1nF
L14 HZ0805E601R-10
L13 HZ0805E601R-10
CHASSIS_GND
R268 4.7k
R269 4.7k
R270 4.7k
R271 49.9
R272 49.9
C25 18pF
C24 18pF
R24 1M
Y1 25MHz
R9 49.9
R8 49.9
C6 0.047uF
R11 49.9
R10 49.9
C7 0.047uF
R17 49.9
R15 49.9
C8 0.047uF
R21 49.9
R18 49.9
C16 0.047uF
Ethernet Terminations
MDI0P
MDI0N
MDI1P
MDI1N
MDI2P
MDI2N
MDI3P
MDI3N
J15
RJ45
TD0_P
TD0_N
TD1_P
TD1_N
TD2_P
TD2_N
TD3_P
TD3_N
VCC
SHIELD1
SHIELD2
GND
D5 LED_GREEN
D4 LED_RED
D3 LED_GREEN
D2 LED_RED
D1 LED_RED
D6 LED_RED
R264 130
R265 130
R266 130
R267 130
R263 130
R1 130
C3 4.7uF
C1 4.7uF
C4 0.47uF
C34 0.47uF
C5 0.47uF
C26 0.47uF
C215 0.047uF
C199 0.047uF
C216 0.047uF
C197 0.047uF
C202 1nF
C211 1nF
C2 4.7uF
C28 0.47uF
C198 0.047uF
C217 0.047uF
C201 0.047uF
C213 1nF
C203 1nF
Rice University
Title
WARP FPGA Board
Description
Ethernet
Date: Tuesday, December 16, 2008
Sheet 15 of 22
Rev 2.0
A
B
C
D
1
2
3
4
5

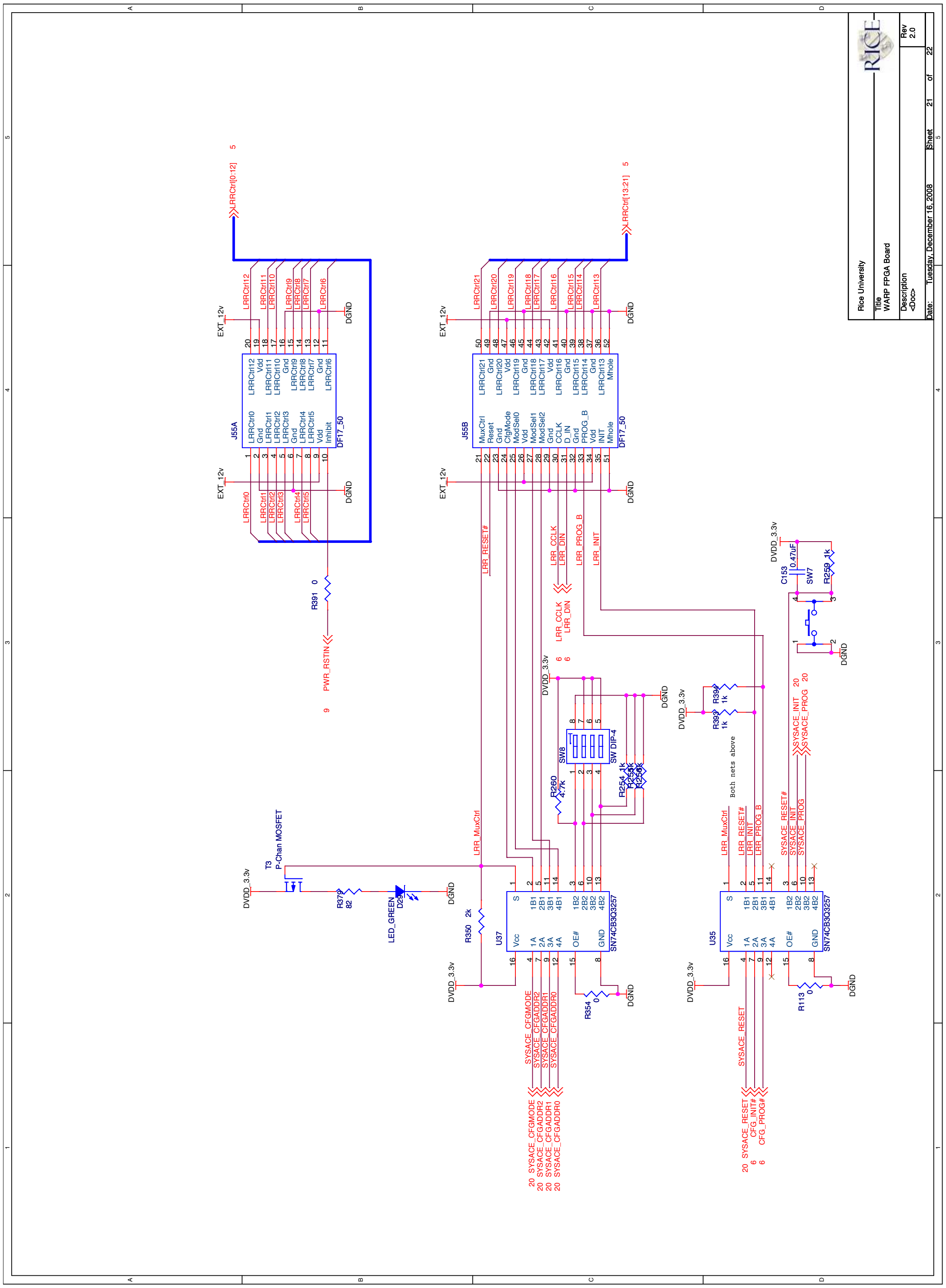

J57A
DDR2_SODIMM_200
DGND
DVDD_3.3v
DVDD_0.9v
R176 NM - 100
R177 NM - 100
R179 1k
R243 1k
R180 33
R242 33
DDR2_SDA 3
DDR2_SCL 3
Rice University
Title
WARP FPGA Board
Description
<Doc>
Rev
2.0
Date: Wednesday, December 10, 2008
Sheet 16 of 22

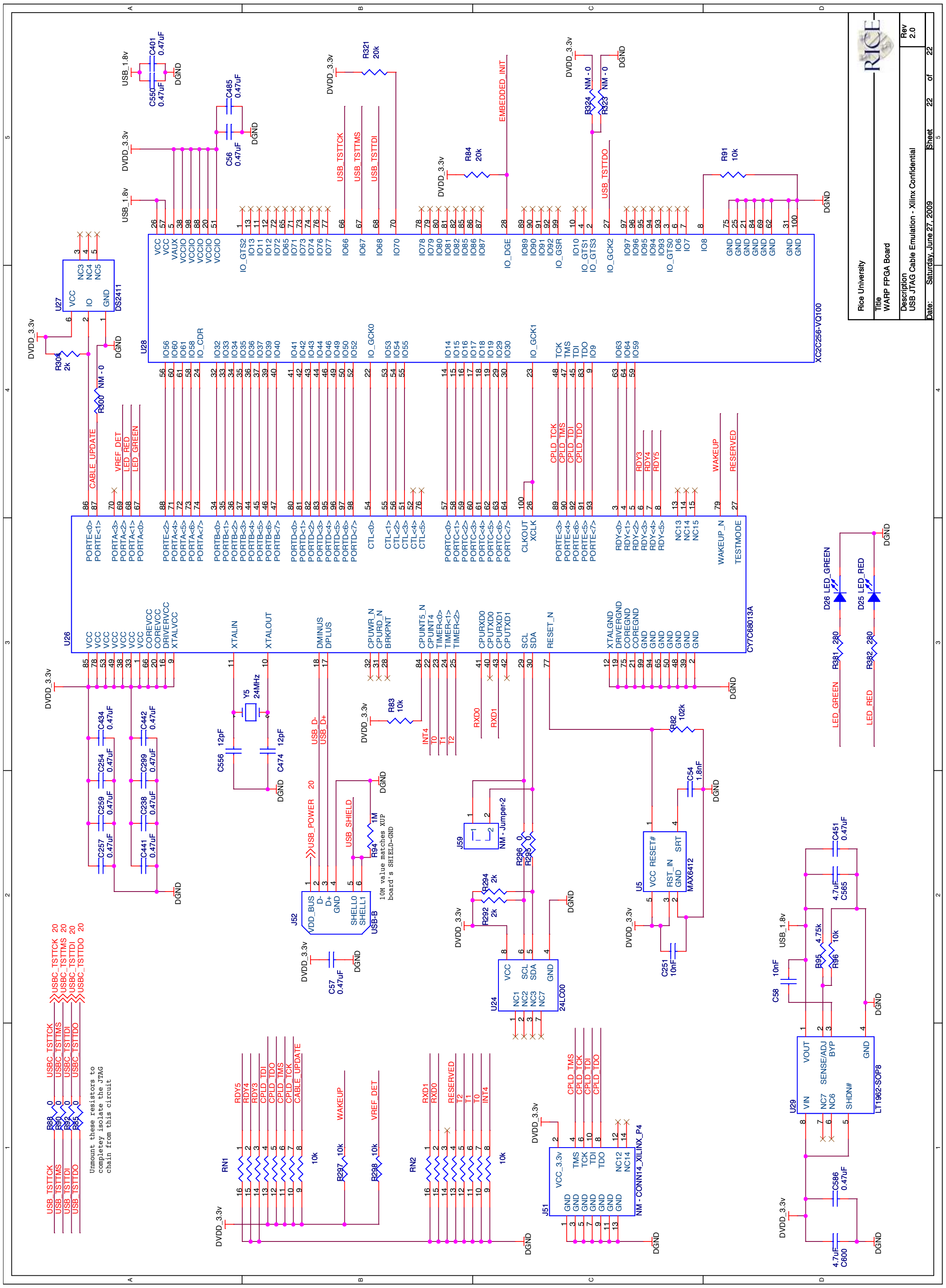

DVDD_0.9v
DGND
4.7uF
0.47uF
0.047uF
249
20pF
DDR2_DQSn0
DDR2_DQS0
DDR2_DM0
DDR2_DQSn1
DDR2_DQS1
DDR2_DM1
DDR2_DQSn2
DDR2_DQS2
DDR2_DM2
DDR2_DQSn3
DDR2_DQS3
DDR2_DM3
DDR2_DQSn4
DDR2_DQS4
DDR2_DM4
DDR2_DQSn5
DDR2_DQS5
DDR2_DM5
DDR2_DQSn6
DDR2_DQS6
DDR2_DM6
DDR2_DQSn7
DDR2_DQS7
DDR2_DM7
6,16
Rice University
Title
WARP FPGA Board
Description
<Doc>
Date: Wednesday, December 10, 2008
Sheet 17 of 22
Rev
2.0
RICE
A
B
C
D
1
2
3
4
5


U36
MAX7318
V+
INT
SDA
SCL
AD0
AD1
AD2
GND
IO_0 … IO_15
U39
MAX7318
D30
D31
D32
LDS-C324RI
A B C D E F G DP
GND
RN3
RN4
RN5
180
C63
0.047uF
C739
0.047uF
R351
R352
2k
HEX_SDA
HEX_SCL
DVDD_3.3v
DGND
R383 150
R384 180
R385 180
R386 150
R387 150
R388 150
R389 150
R390 150
D17 LED_GREEN
D16 LED_GREEN
D34 LED_RED
D33 LED_RED
D38 LED_YELLOW
D37 LED_YELLOW
D36 LED_YELLOW
D35 LED_YELLOW
U9
LP2997
VREF
VSENSE
VTT
SD#
AVIN
VDDQ
PVIN
EXPAD
GND
DVDD_0.9v_VREF
DVDD_0.9v
DVDD_1.8v
C150 0.47uF
C138 0.47uF
C139 470uF
C152 4.7uF
J30
Jumper-2
C140 0.47uF
C77 0.47uF
C78 0.47uF
C79 0.47uF
C80 0.47uF
C86 0.47uF
C145 4.7uF
C144 4.7uF
C147 4.7uF
C146 4.7uF
C143 4.7uF
C142 4.7uF
C62 0.47uF
C576 0.047uF
C580 0.047uF
C614 0.47uF
C613 0.47uF
J57B
DDR2_SODIMM_200
VREF
VSS
VDD
VDDSPD
Rice University
Title
WARP FPGA Board
Description
<Doc>
Rev
2.0
Date: Saturday, June 27, 2009
Sheet 18 of 22
A
B
C
D
1
2
3
4
5


Rice University
Title
WARP FPGA Board
Description
<Doc>
Date: Saturday, June 27, 2009
Sheet 19 of 22
Rev 2.0
J20
CONN20
DEBUG[0:7]
DEBUG[8:15]
DEBUG0
DEBUG1
DEBUG2
DEBUG3
DEBUG4
DEBUG5
DEBUG6
DEBUG7
DEBUG8
DEBUG9
DEBUG10
DEBUG11
DEBUG12
DEBUG13
DEBUG14
DEBUG15
DGND
PB_LEFT
PB_UP
PB_RIGHT
PB_CENTER
PB_DOWN
SW1
SW2
SW3
SW4
SW6
C711 0.47uF
C707 0.47uF
C712 0.47uF
C716 0.47uF
C718 0.47uF
R356 1k
R353 1k
R357 1k
R361 1k
R365 1k
DVDD_2.5v
DVDD_3.3v
C713 0.47uF
DVDD_5v
J19
VCC
TACH
GND
Fan Header
Fan Jumper
RLED0
RLED1
RLED2
RLED3
GLED0
GLED1
GLED2
GLED3
R358 49.9
R360 49.9
R362 49.9
R363 49.9
R245 82
R247 82
R249 82
R253 82
D10 LED_GREEN
D13 LED_GREEN
D18 LED_GREEN
D20 LED_GREEN
D11 LED_RED
D14 LED_RED
D19 LED_RED
D21 LED_RED
EEPROM_DIO
R345 1k
U33
DS2431-TSOC6
IO
GND
NC3
NC4
NC5
NC6
C634 0.47uF
USER_DIPSW0
USER_DIPSW1
USER_DIPSW2
USER_DIPSW3
SW5
SW DIP-4
C141 0.47uF
C148 0.47uF
C149 0.47uF
C151 0.47uF
R246 1k
R248 1k
R250 1k
R252 1k
J50
DSUB-9P
C55 NM - 1nF
U25
MAX3319
VCC
C1+
C1-
C2+
C2-
TX_IN
RX_OUT
GND
V+
V-
READY
FORCEOFF#
FORCEON
INVALID#
TX_OUT
RX_IN
UART_TX_RS232
UART_RX_RS232
UART_TX
UART_RX
C278 0.47uF
C234 0.47uF
C235 0.47uF
C47 0.47uF
C48 0.47uF
J58
USB-B
VDD_BUS
D-
D+
GND
SHELL0
SHELL1
R380 0
C738 4.7uF
C742 0.47uF
C737 0.047uF
R376 10k
U38
FT232RQ
VCCIO
3V3OUT
VCC
Reset#
USBDM
USBDP
NC5
NC12
NC13
NC23
NC25
NC29
Test
GND
AGND
TXD
RXD
RI#
DSR#
DCD#
CTS#
DTR#
RTS#
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
OscI
OscO
FTDI_TXD
FTDI_RXD
C740 4.7uF
C741 0.47uF
C736 0.47uF
R377 280
R378 280
D27 LED_RED
D28 LED_RED

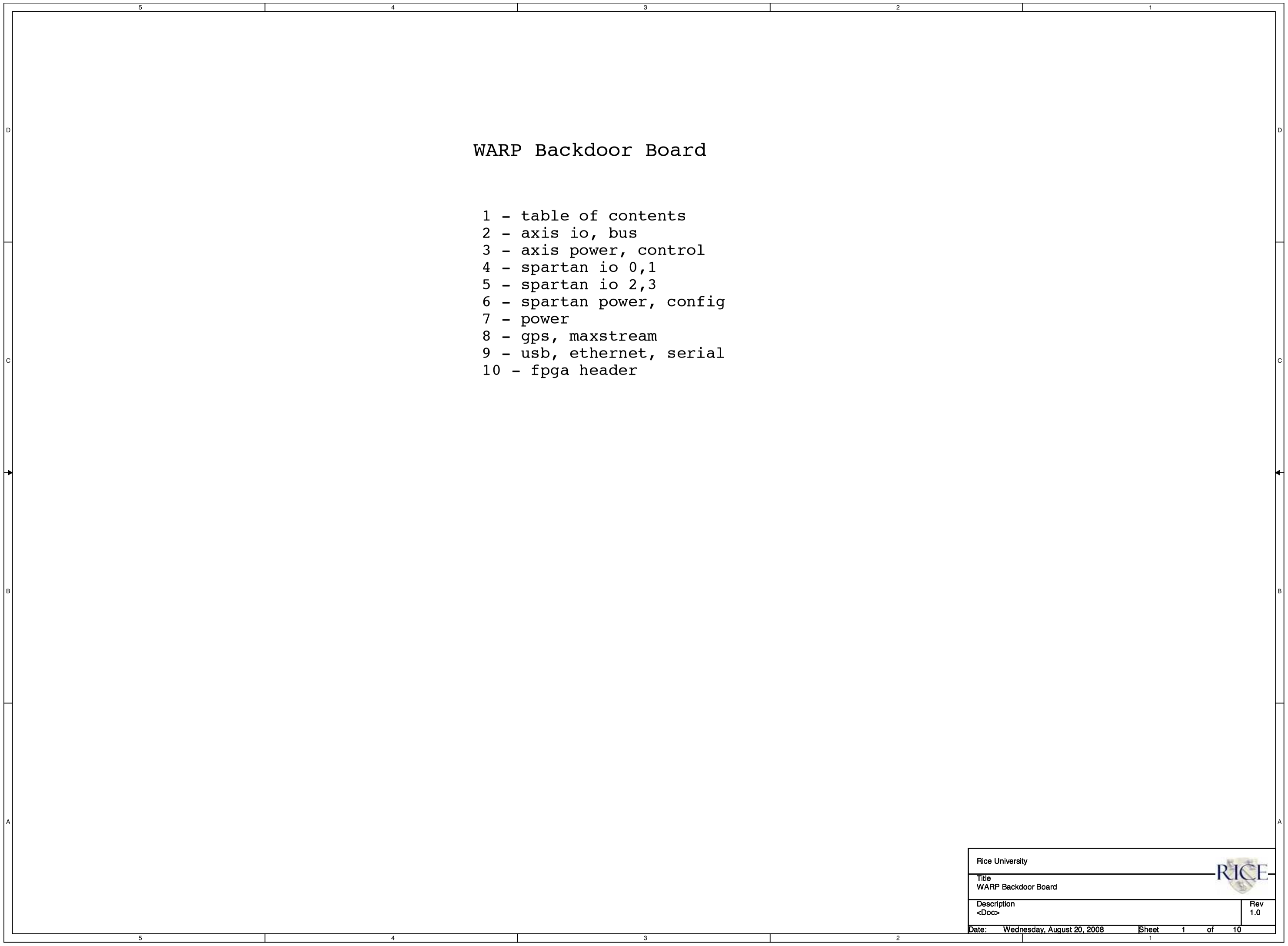

Rice University
Title
WARP FPGA Board
Description
<Doc>
Rev
2.0
Date: Tuesday, December 02, 2008
Sheet 20 of 22
U34
XCCACE_TQ144
J56
Compact Flash Socket
U32
SN74CB3Q3257
J53
CONN14_XILINX_P4
U7
SN74CBT1G125
Controls Erried blinking
ERRLED and CF_CD1 on SystemAce
TDI/TDO cross over here. The FPGA is only device on this JTAG chain.
J54
Jumper-2
D9 LED_RED
D12 LED_GREEN
USB_POWER 22
DVDD_3.3v
DVDD_5v
DGND
SYSACE_CLK33 2
SYSACE_RESET 21
SYSACE_INIT 21
SYSACE_PROG 21
SYSACE_CFGMODE 21
SYSACE_CFGADDR0 21
SYSACE_CFGADDR1 21
SYSACE_CFGADDR2 21
FPGA_TCK 6
FPGA_TDO 6
FPGA_TDI 6
FGPA_TMS 6
USBC_TSTTMS 22
USBC_TSTTCK 22
USBC_TSTTDI 22
USBC_TSTTDO 22
5 SYSACE_MPCE
5 SYSACE_MPBRDY
5 SYSACE_MPWE
5 SYSACE_MPOE
5 SYSACE_MPIRQ
5 SYSACE_MPA[0:6]
5 SYSACE_MPD[0:7]
CF_CD1
CF_ERRLED
R226 180
R244 130
R106 5.1k
R331 4.7k
R330 4.7k
R336 10k
R97 10k
R98 10k
R99 590
R100 10k
R86 590
R87 10k
R89 10k
R93 10k
R339 0
R340 10k
R341 10k
R343 10k
R348 10k
R114 0
R116 0
R108 1k
R111 1k
R115 NM
R117 5.1k
R140 5.1k


Rice University
Title
WARP FPGA Board
Description
<Doc>
Date: Tuesday, December 16, 2008
Sheet 21 of 22
Rev 2.0
J55A
DF17_50
J55B
DF17_50
LRRCtrl[0:12] 5
LRRCtrl[13:21] 5
EXT_12v
DGND
LRRCtrl0
LRRCtrl1
LRRCtrl2
LRRCtrl3
LRRCtrl4
LRRCtrl5
LRRCtrl6
LRRCtrl7
LRRCtrl8
LRRCtrl9
LRRCtrl10
LRRCtrl11
LRRCtrl12
LRRCtrl13
LRRCtrl14
LRRCtrl15
LRRCtrl16
LRRCtrl17
LRRCtrl18
LRRCtrl19
LRRCtrl20
LRRCtrl21
Gnd
Vdd
Inhibit
MuxCtrl
Reset
CfgMode
ModSel0
ModSel1
ModSel2
CCLK
D_IN
PROG_B
INIT
Mhole
R391 0
PWR_RSTIN 9
LRR_RESET#
LRR_CCLK
LRR_DIN
LRR_PROG_B
LRR_INIT
LRR_CCLK 6
LRR_DIN 6
LRR_MuxCtrl
DVDD_3.3v
SW8
SW DIP-4
R260 4.7k
R254 1k
R255 1k
R256 1k
R392 1k
R393 1k
Both nets above
C153 0.47uF
SW7
R259 1k
SYSACE_INIT 20
SYSACE_PROG 20
SYSACE_RESET#
SYSACE_INIT
SYSACE_PROG
T3
P-Chan MOSFET
R372 82
LED_GREEN
D29
R350 2k
U37
U35
SN74CB3Q3257
S
Vcc
1A
2A
3A
4A
OE#
GND
1B1
2B1
3B1
4B1
1B2
2B2
3B2
4B2
R354 0
R113 0
SYSACE_CFGMODE
SYSACE_CFGADDR2
SYSACE_CFGADDR1
SYSACE_CFGADDR0
20 SYSACE_CFGMODE
20 SYSACE_CFGADDR2
20 SYSACE_CFGADDR1
20 SYSACE_CFGADDR0
SYSACE_RESET
20 SYSACE_RESET
6 CFG_INIT#
6 CFG_PROG#
A
B
C
D
1
2
3
4
5


Rice University
Title
WARP FPGA Board
Description
USB JTAG Cable Emulation - Xilinx Confidential
Date: Saturday, June 27, 2009
Sheet 22 of 22
Rev 2.0
U26
CY7C68013A
U28
XC2C256-VQ100
U27
DS2411
U24
24LC00
U5
MAX6412
U29
LT1962-SOP8
J51
NM - CONN14_XILINX_P4
J52
USB-B
J59
NM - Jumper-2
Y5
24MHz
RN1
RN2
D26 LED_GREEN
D25 LED_RED
DVDD_3.3v
USB_1.8v
DGND
Unmount these resistors to completely isolate the JTAG chain from this circuit
10K value matches XUP board's SHIELD-GND
USB_TSTTCK
USB_TSTTMS
USB_TSTTDI
USB_TSTTDO
USBC_TSTTCK
USBC_TSTTMS
USBC_TSTTDI
USBC_TSTTDO
CABLE_UPDATE
VREF_DET
LED_RED
LED_GREEN
CPLD_TCK
CPLD_TMS
CPLD_TDI
CPLD_TDO
WAKEUP
RESERVED
EMBEDDED_INIT
USB_POWER
USB_SHIELD
USB_D-
USB_D+

A.2 Backdoor Board

The schematics for Backdoor Board v1.0.



WARP Backdoor Board

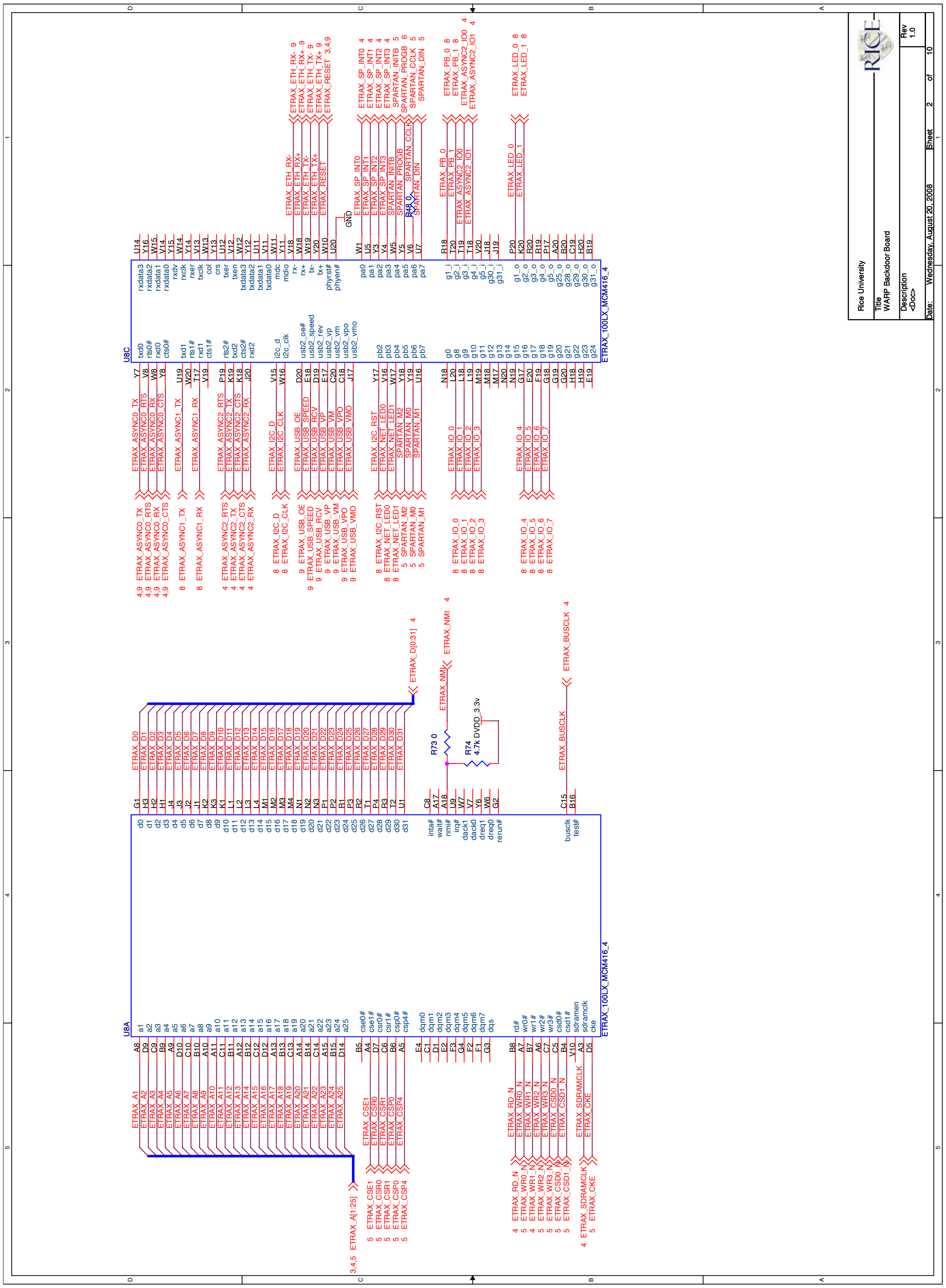
Rice University

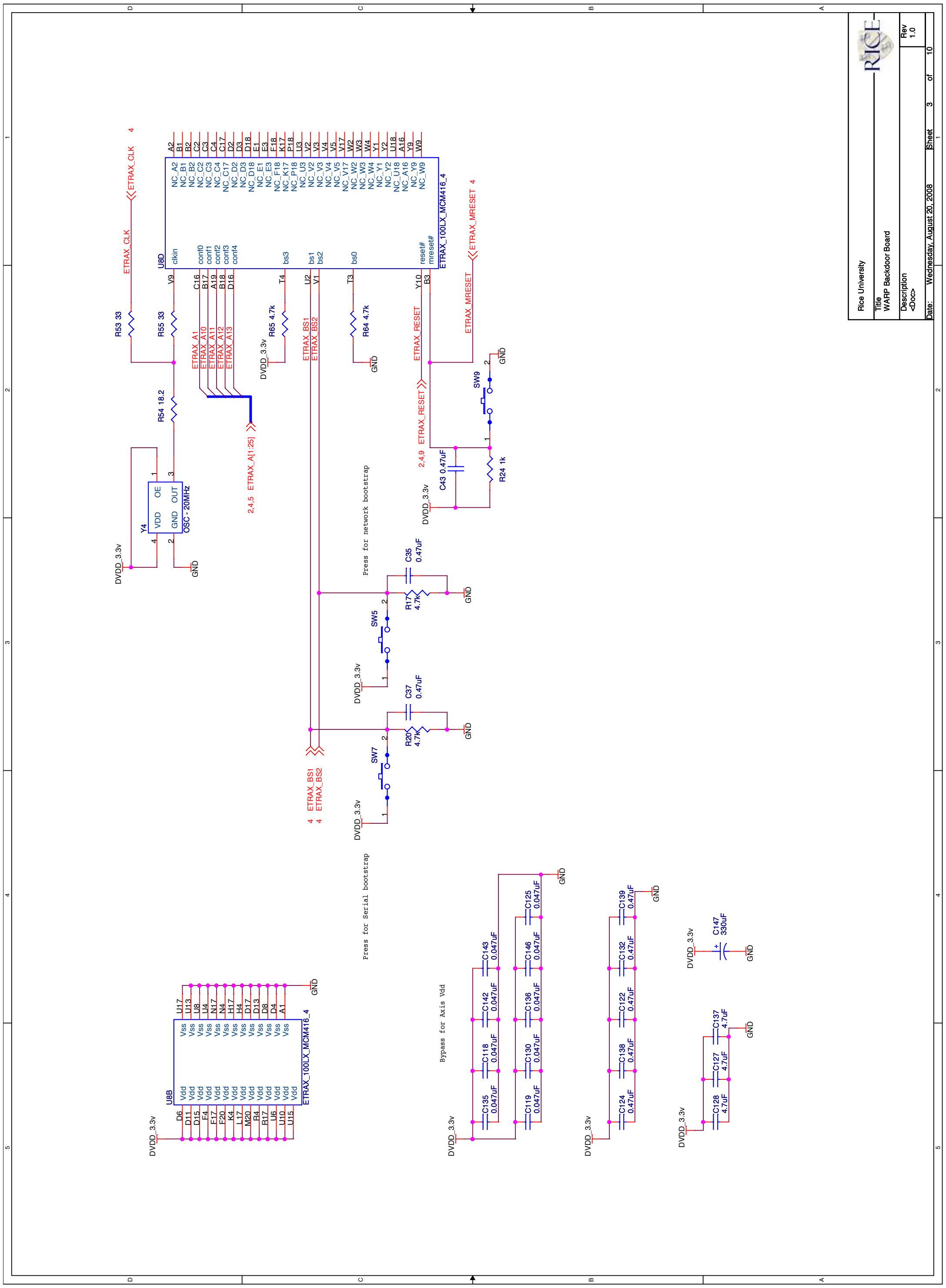
Title
WARP Backdoor Board

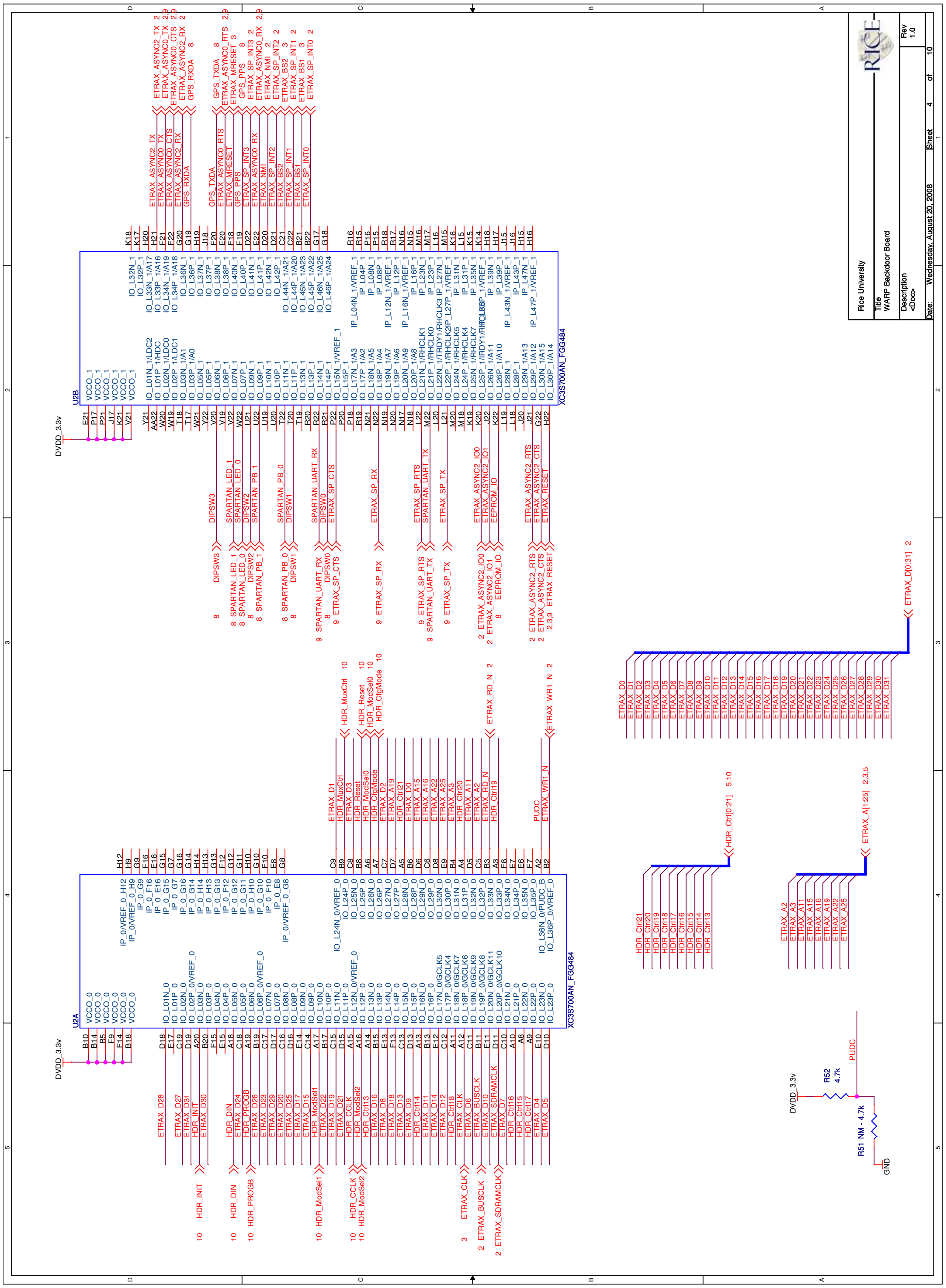
Description
<Doc>

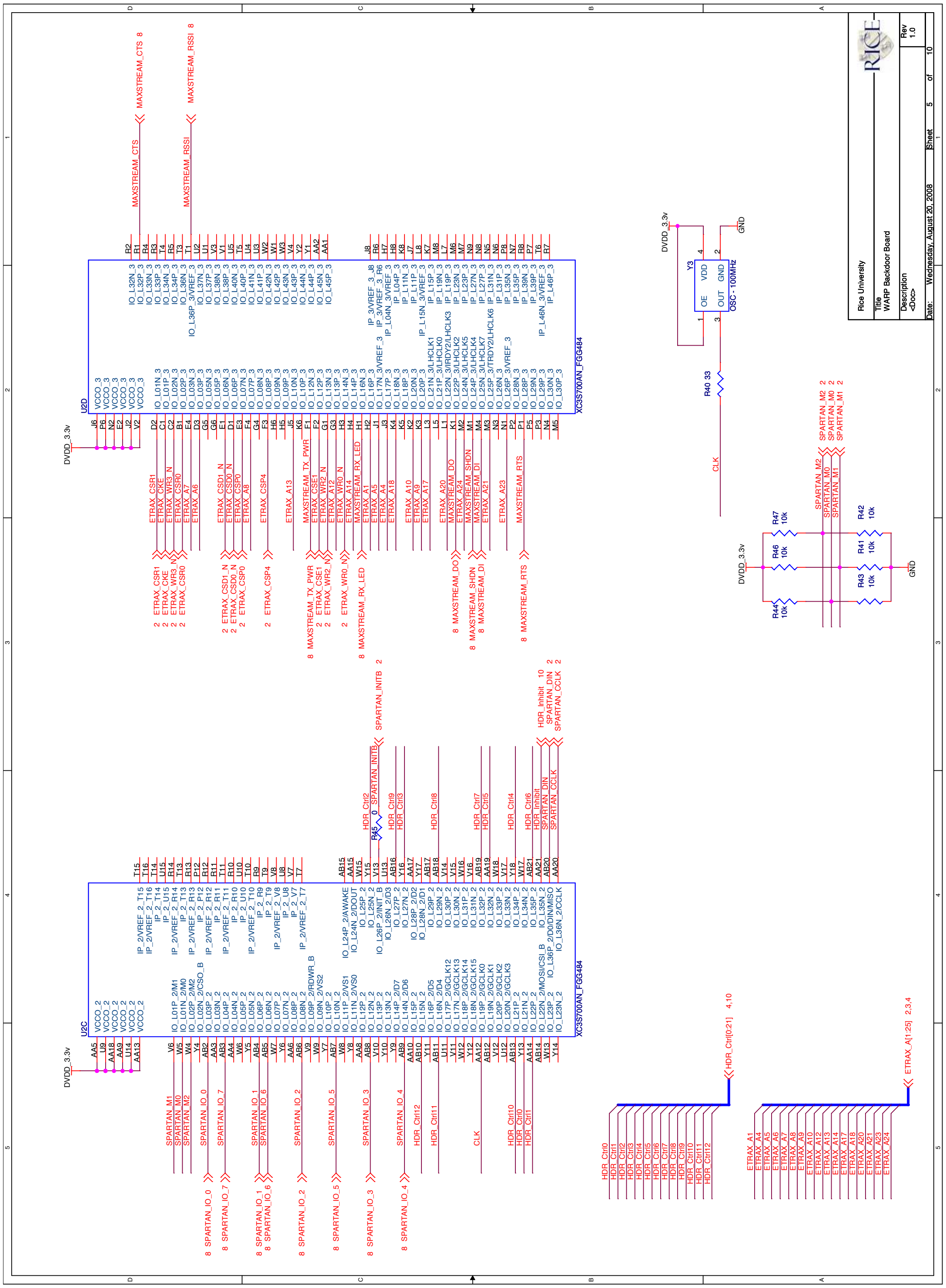
Rev
1.0

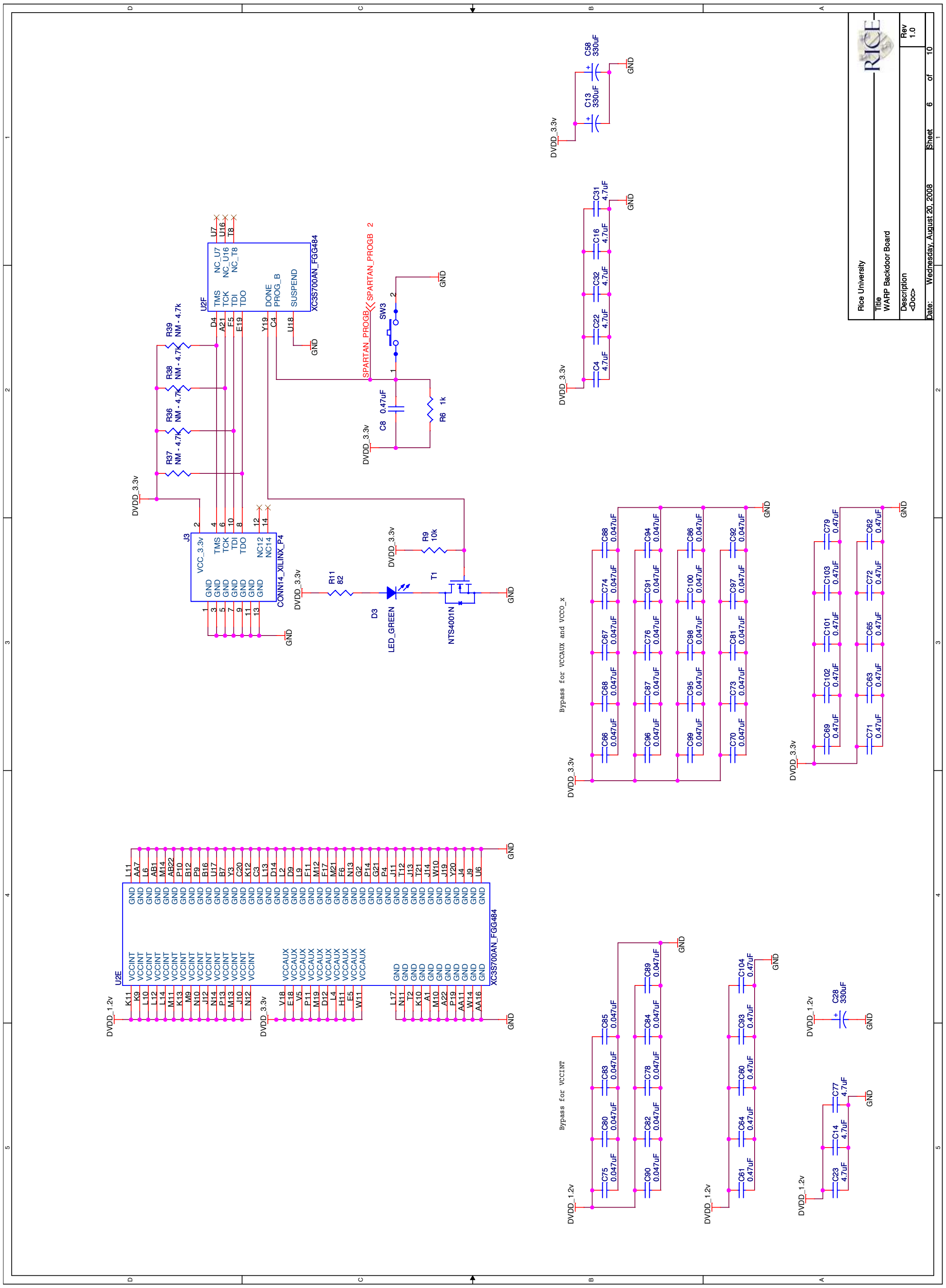
Date: Wednesday, August 20, 2008 Sheet 1 of 10

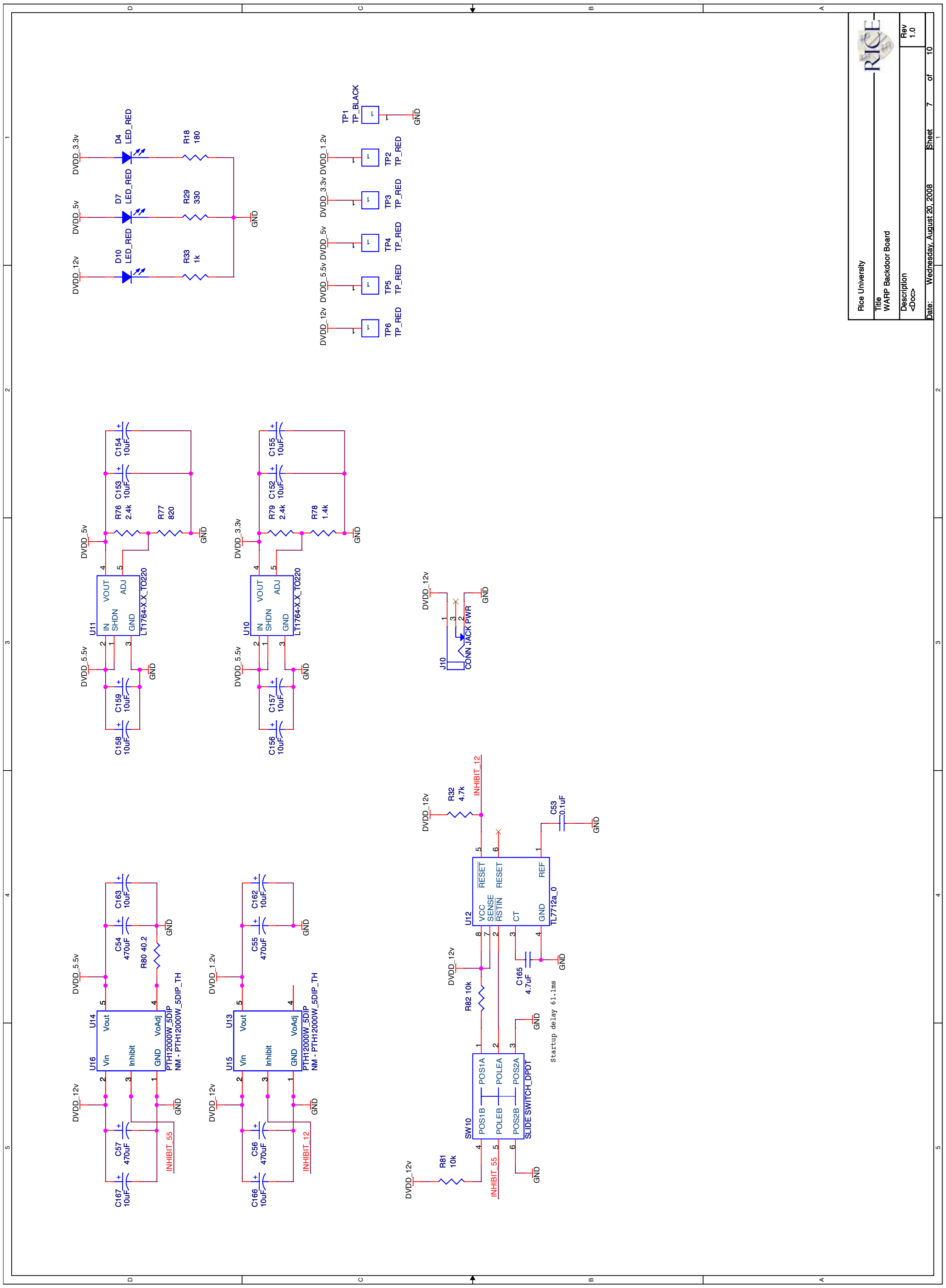

U8A
U8C
ETRAX_100LX_MCM416_4
ETRAX_A[1:25]
ETRAX_D[0:31]
ETRAX_CSE1
ETRAX_CSR0
ETRAX_CSR1
ETRAX_CSP0
ETRAX_CSP4
ETRAX_RD_N
ETRAX_WR0_N
ETRAX_WR1_N
ETRAX_WR2_N
ETRAX_WR3_N
ETRAX_CSD0_N
ETRAX_CSD1_N
ETRAX_SDRAMCLK
ETRAX_CKE
ETRAX_NMI
ETRAX_BUSCLK
R73 0
R74 4.7k
DVDD_3.3v
R49 0
GND
ETRAX_ASYNC0_TX
ETRAX_ASYNC0_RTS
ETRAX_ASYNC0_RX
ETRAX_ASYNC0_CTS
ETRAX_ASYNC1_TX
ETRAX_ASYNC1_RX
ETRAX_ASYNC2_RTS
ETRAX_ASYNC2_TX
ETRAX_ASYNC2_CTS
ETRAX_ASYNC2_RX
ETRAX_I2C_D
ETRAX_I2C_CLK
ETRAX_USB_OE
ETRAX_USB_SPEED
ETRAX_USB_RCV
ETRAX_USB_VP
ETRAX_USB_VM
ETRAX_USB_VPO
ETRAX_USB_VMO
ETRAX_I2C_RST
ETRAX_NET_LED0
ETRAX_NET_LED1
SPARTAN_M2
SPARTAN_M0
SPARTAN_M1
ETRAX_IO_0
ETRAX_IO_1
ETRAX_IO_2
ETRAX_IO_3
ETRAX_IO_4
ETRAX_IO_5
ETRAX_IO_6
ETRAX_IO_7
ETRAX_ETH_RX-
ETRAX_ETH_RX+
ETRAX_ETH_TX-
ETRAX_ETH_TX+
ETRAX_RESET
ETRAX_SP_INT0
ETRAX_SP_INT1
ETRAX_SP_INT2
ETRAX_SP_INT3
SPARTAN_INITB
SPARTAN_PROGB
SPARTAN_CCLK
SPARTAN_DIN
ETRAX_PB_0
ETRAX_PB_1
ETRAX_ASYNC2_IO0
ETRAX_ASYNC2_IO1
ETRAX_LED_0
ETRAX_LED_1
Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 2 of 10
A
B
C
D
1
2
3
4
5

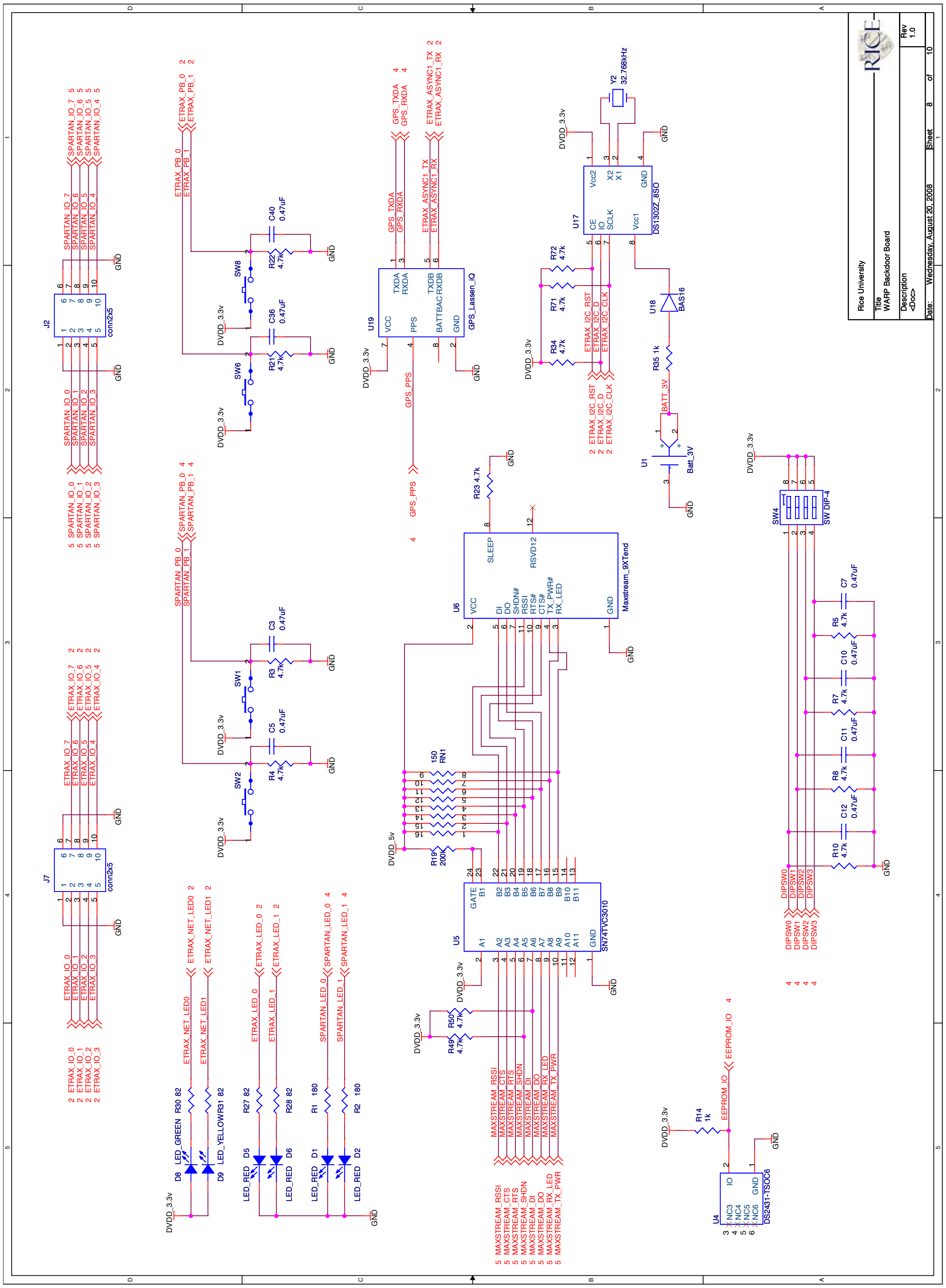

ETRAX_CLK 4
ETRAX_CLK
R53 33
R55 33
R54 18.2
Y4
VDD
OE
GND
OUT
OSC - 20MHz
DVDD_3.3v
GND
U8D
clkin
conf0
conf1
conf2
conf3
conf4
bs3
bs1
bs2
bs0
reset#
mreset#
ETRAX_100LX_MCM416_4
NC_A2
NC_B1
NC_B2
NC_C2
NC_C3
NC_C4
NC_C17
NC_D2
NC_D3
NC_D18
NC_E1
NC_E3
NC_F18
NC_K17
NC_P18
NC_U3
NC_V2
NC_V3
NC_V4
NC_V5
NC_V17
NC_W2
NC_W3
NC_W4
NC_Y1
NC_Y2
NC_U18
NC_A16
NC_Y9
NC_W9
ETRAX_A1
ETRAX_A10
ETRAX_A11
ETRAX_A12
ETRAX_A13
2,4,5 ETRAX_A[1:25]
R65 4.7k
R64 4.7k
ETRAX_BS1
ETRAX_BS2
ETRAX_RESET
2,4,9 ETRAX_RESET
ETRAX_MRESET
ETRAX_MRESET 4
SW9
C43 0.47uF
R24 1k
Press for network bootstrap
SW5
C35 0.47uF
R17 4.7k
SW7
C37 0.47uF
R20 4.7k
4 ETRAX_BS1
4 ETRAX_BS2
Press for Serial bootstrap
U8B
Vdd
Vss
ETRAX_100LX_MCM416_4
Bypass for Axis Vdd
C135 0.047uF
C118 0.047uF
C142 0.047uF
C143 0.047uF
C119 0.047uF
C130 0.047uF
C136 0.047uF
C146 0.047uF
C125 0.047uF
C124 0.47uF
C138 0.47uF
C122 0.47uF
C132 0.47uF
C139 0.47uF
C128 4.7uF
C127 4.7uF
C137 4.7uF
C147 330uF
Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 3 of 10

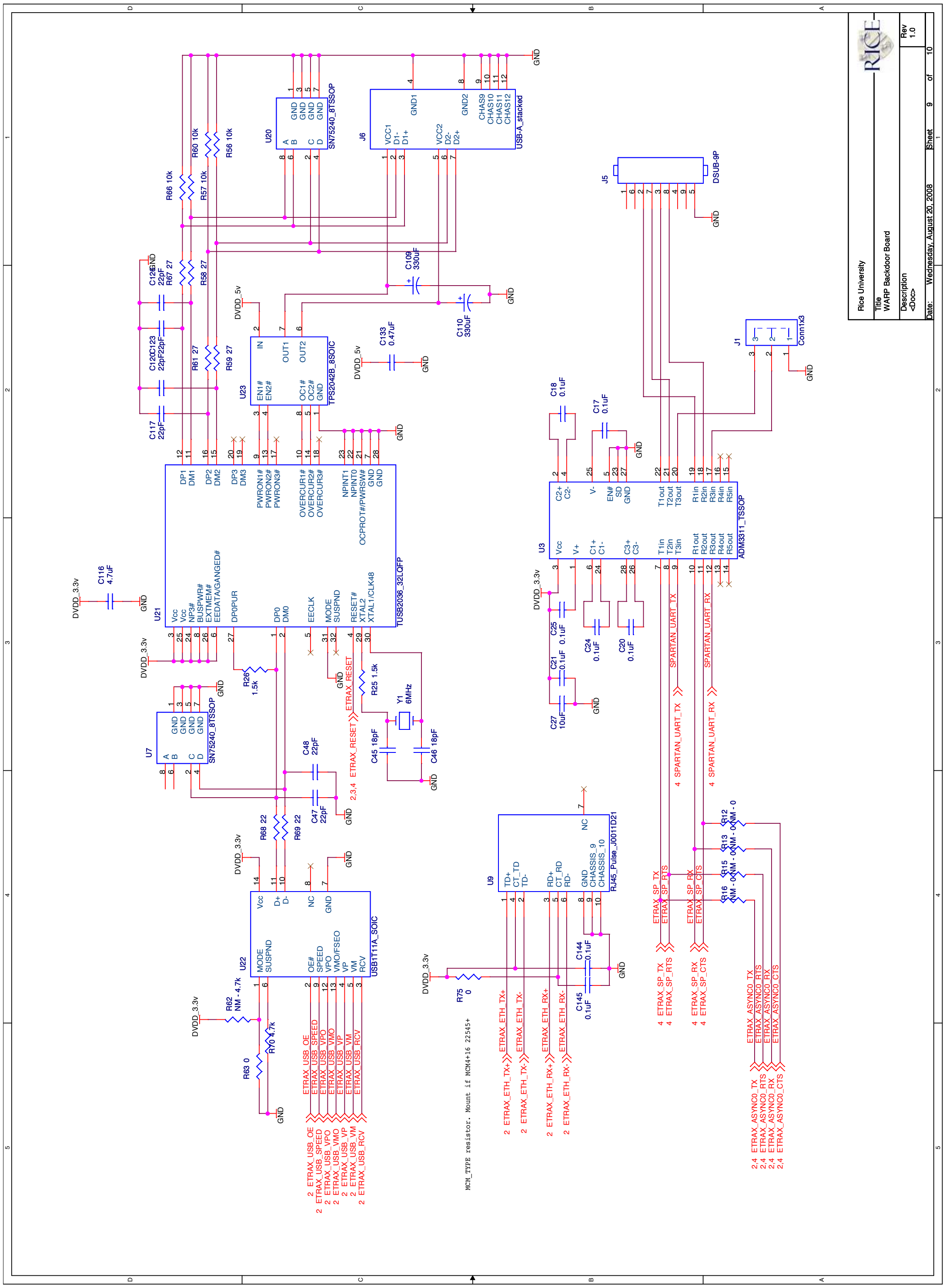

Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 4 of 10
U2A
XC3S700AN_FGG484
U2B
XC3S700AN_FGG484
DVDD_3.3v
R52
4.7k
R51 NM - 4.7k
PUDC
GND
ETRAX_D[0:31]
HDR_Ctrl[0:21]
ETRAX_A[1:25]

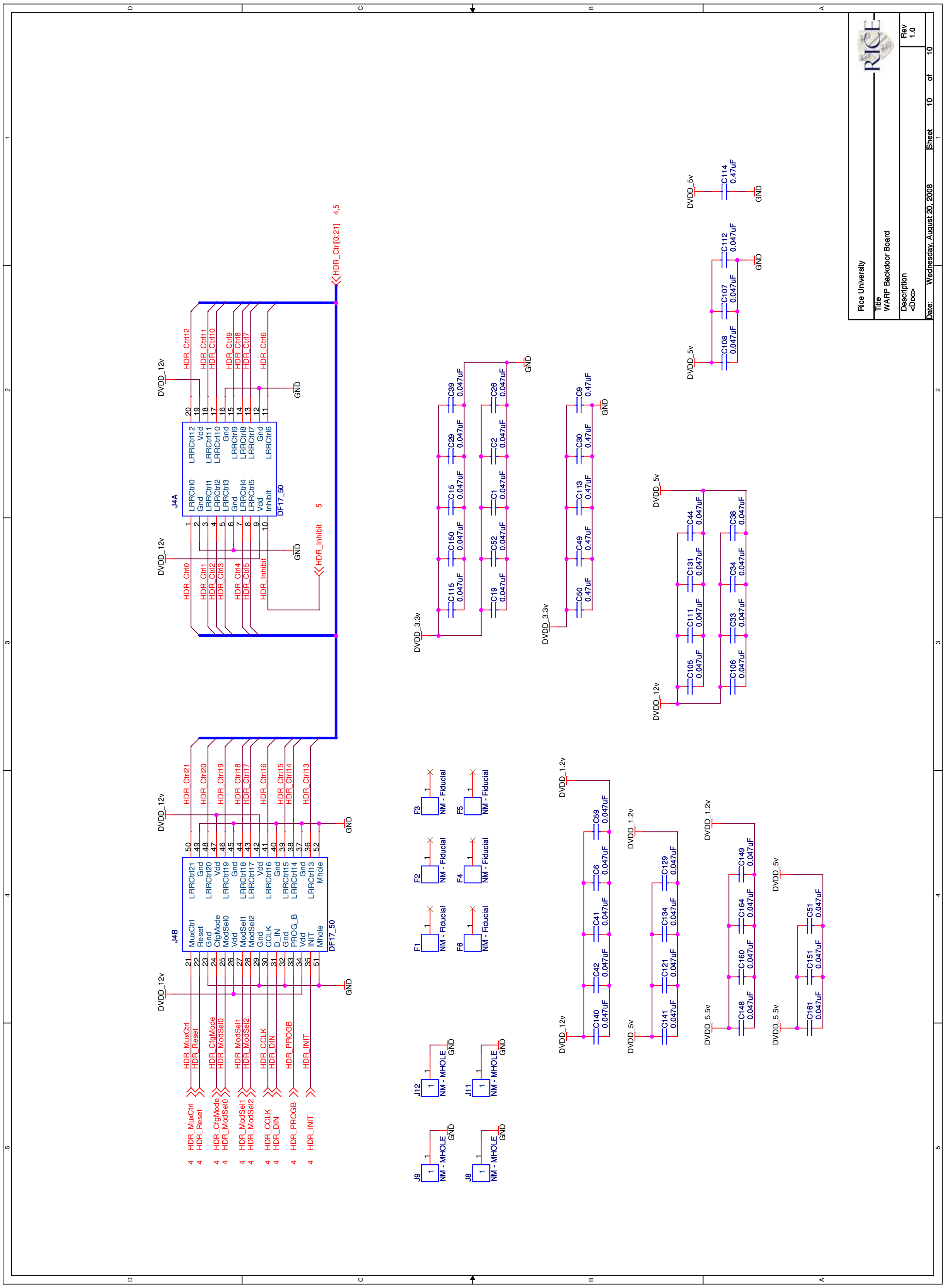

Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 5 of 10
U2D
XC3S700AN_FGG484
U2C
XC3S700AN_FGG484
Y3
OSC - 100MHz
OE
OUT
VDD
GND
DVDD_3.3v
GND
CLK
R40 33
R45 0
R41 10k
R42 10k
R43 10k
R44 10k
R46 10k
R47 10k
MAXSTREAM_CTS 8
MAXSTREAM_RSSI 8
SPARTAN_M2 2
SPARTAN_M0 2
SPARTAN_M1 2
SPARTAN_INITB 2
HDR_Inhibit 10
SPARTAN_DIN 2
SPARTAN_CCLK 2
HDR_Ctrl[0:21] 4,10
ETRAX_A[1:25] 2,3,4
8 SPARTAN_IO_0
8 SPARTAN_IO_7
8 SPARTAN_IO_1
8 SPARTAN_IO_6
8 SPARTAN_IO_2
8 SPARTAN_IO_5
8 SPARTAN_IO_3
8 SPARTAN_IO_4
2 ETRAX_CSR1
2 ETRAX_CKE
2 ETRAX_WR3_N
2 ETRAX_CSR0
2 ETRAX_CSD1_N
2 ETRAX_CSD0_N
2 ETRAX_CSP0
2 ETRAX_CSP4
8 MAXSTREAM_TX_PWR
2 ETRAX_CSE1
2 ETRAX_WR2_N
2 ETRAX_WR0_N
8 MAXSTREAM_RX_LED
8 MAXSTREAM_DO
8 MAXSTREAM_SHDN
8 MAXSTREAM_DI
8 MAXSTREAM_RTS


Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 6 of 10
U2F
XC3S700AN_FGG484
TMS
TCK
TDI
TDO
NC_U7
NC_U16
NC_T8
DONE
PROG_B
SUSPEND
D4
A21
F5
E19
Y19
C4
U18
U7
U16
T8
SPARTAN_PROGB
SPARTAN_PROGB_2
SW3
R39 NM - 4.7k
R38 NM - 4.7k
R36 NM - 4.7k
R37 NM - 4.7k
C8 0.47uF
R6 1k
DVDD_3.3v
GND
J3
CONN14_XILINX_P4
VCC_3.3v
NC12
NC14
R11 82
D3
LED_GREEN
R9 10k
T1
NTS4001N
U2E
XC3S700AN_FGG484
VCCINT
VCCAUX
DVDD_1.2v
Bypass for VCCAUX and VCCO_x
Bypass for VCCINT
C58 330uF
C13 330uF
C31 4.7uF
C16 4.7uF
C32 4.7uF
C22 4.7uF
C4 4.7uF
C88 0.047uF
C74 0.047uF
C67 0.047uF
C68 0.047uF
C66 0.047uF
C94 0.047uF
C91 0.047uF
C76 0.047uF
C87 0.047uF
C96 0.047uF
C86 0.047uF
C100 0.047uF
C98 0.047uF
C95 0.047uF
C99 0.047uF
C92 0.047uF
C97 0.047uF
C81 0.047uF
C73 0.047uF
C70 0.047uF
C79 0.47uF
C103 0.47uF
C101 0.47uF
C102 0.47uF
C69 0.47uF
C62 0.47uF
C72 0.47uF
C65 0.47uF
C63 0.47uF
C71 0.47uF
C85 0.047uF
C83 0.047uF
C80 0.047uF
C75 0.047uF
C89 0.047uF
C84 0.047uF
C78 0.047uF
C82 0.047uF
C90 0.047uF
C104 0.47uF
C93 0.47uF
C60 0.47uF
C64 0.47uF
C61 0.47uF
C28 330uF
C77 4.7uF
C14 4.7uF
C23 4.7uF


D4
LED_RED
D7
LED_RED
D10
LED_RED
R18
180
R29
330
R33
1k
DVDD_3.3v
DVDD_5v
DVDD_12v
GND
TP1
TP_BLACK
TP2
TP_RED
TP3
TP_RED
TP4
TP_RED
TP5
TP_RED
TP6
TP_RED
DVDD_1.2v
DVDD_3.3v
DVDD_5v
DVDD_5.5v
DVDD_12v
U11
LT1764-X.X_TO220
IN
SHDN
GND
VOUT
ADJ
U10
LT1764-X.X_TO220
C154
10uF
C153
10uF
C155
10uF
C152
10uF
R76
2.4k
R77
820
R79
2.4k
R78
1.4k
C159
10uF
C158
10uF
C157
10uF
C156
10uF
DVDD_5.5v
J10
CONN JACK PWR
U14
U16
PTH12000W_5DIP
NM - PTH12000W_5DIP_TH
Vin
Inhibit
GND
Vout
VoAdj
U13
U15
C163
10uF
C54
470uF
R80 40.2
C162
10uF
C55
470uF
C57
470uF
C167
10uF
C56
470uF
C166
10uF
INHIBIT_55
INHIBIT_12
DVDD_1.2v
U12
TL7712a_0
VCC
SENSE
RSTIN
CT
GND
RESET
RESET
REF
R32
4.7k
C53
0.1uF
C165
4.7uF
R82 10k
SW10
SLIDE SWITCH_DPDT
POS1A
POLEA
POS2A
POS1B
POLEB
POS2B
R81
10k
Startup delay 61.1ms
Rice University
Title
WARP Backdoor Board
Description
<Doc>
Date: Wednesday, August 20, 2008
Sheet 7 of 10
Rev
1.0
A
B
C
D
1
2
3
4
5


Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 8 of 10
RICE
J2
conn2x5
J7
SPARTAN_IO_0
SPARTAN_IO_1
SPARTAN_IO_2
SPARTAN_IO_3
SPARTAN_IO_4
SPARTAN_IO_5
SPARTAN_IO_6
SPARTAN_IO_7
ETRAX_IO_0
ETRAX_IO_1
ETRAX_IO_2
ETRAX_IO_3
ETRAX_IO_4
ETRAX_IO_5
ETRAX_IO_6
ETRAX_IO_7
ETRAX_PB_0
ETRAX_PB_1
SPARTAN_PB_0
SPARTAN_PB_1
SW1
SW2
SW6
SW8
C3 0.47uF
C5 0.47uF
C36 0.47uF
C40 0.47uF
R3 4.7k
R4 4.7k
R21 4.7k
R22 4.7k
DVDD_3.3v
DVDD_5v
GND
U19
GPS_Lassen_iQ
VCC
PPS
BATTBAC
TXDA
RXDA
TXDB
RXDB
GPS_TXDA
GPS_RXDA
GPS_PPS
ETRAX_ASYNC1_TX
ETRAX_ASYNC1_RX
U17
DS1302Z_8SO
CE
IO
SCLK
Vcc1
Vcc2
X1
X2
Y2
32.768kHz
R34 4.7k
R71 4.7k
R72 4.7k
ETRAX_I2C_RST
ETRAX_I2C_D
ETRAX_I2C_CLK
U18
BAS16
R35 1k
BATT_3V
U1
Batt_3V
U6
Maxstream_9XTend
SLEEP
RSVD12
VCC
DI
DO
SHDN#
RSSI
RTS#
CTS#
TX_PWR#
RX_LED
R23 4.7k
RN1
150
R19
200
U5
SN74TVC3010
GATE
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
R49 4.7k
R50 4.7k
MAXSTREAM_RSSI
MAXSTREAM_CTS
MAXSTREAM_RTS
MAXSTREAM_SHDN
MAXSTREAM_DI
MAXSTREAM_DO
MAXSTREAM_RX_LED
MAXSTREAM_TX_PWR
SW4
SW DIP-4
DIPSW0
DIPSW1
DIPSW2
DIPSW3
R5 4.7k
R7 4.7k
R8 4.7k
R10 4.7k
C7 0.47uF
C10 0.47uF
C11 0.47uF
C12 0.47uF
U4
DS2431-TSOC6
IO
NC3
NC4
NC5
NC6
R14
1k
EEPROM_IO
D8
LED_GREEN
D9
LED_YELLOW
R30 82
R31 82
ETRAX_NET_LED0
ETRAX_NET_LED1
D5
D6
LED_RED
R27 82
R28 82
ETRAX_LED_0
ETRAX_LED_1
D1
D2
R1 180
R2 180
SPARTAN_LED_0
SPARTAN_LED_1
A
B
C
D
1
2
3
4
5


Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 9 of 10
U21
USB2036_32LQFP
U20
SN75240_8TSSOP
U7
SN75240_8TSSOP
J6
USB-A_stacked
U23
TPS2042B_8SOIC
U22
USB1T11A_SOIC
U9
RJ45_Pulse_J0011D21
U3
ADM3311_TSSOP
J5
DSUB-9P
J1
Conn1x3
Y1
6MHz
C109 330uF
C110 330uF
C133 0.47uF
C116 4.7uF
C117 22pF
C120 22pF
C123 22pF
C126 22pF
C45 18pF
C46 18pF
C47 22pF
C48 22pF
C144 0.1uF
C145 0.1uF
C17 0.1uF
C18 0.1uF
C20 0.1uF
C21 0.1uF
C24 0.1uF
C25 0.1uF
C27 10uF
R25 1.5k
R26 1.5k
R56 10k
R57 10k
R60 10k
R66 10k
R58 27
R59 27
R61 27
R67 27
R68 22
R69 22
R62 NM - 4.7k
R70 4.7k
R63 0
R75 0
R12 NM - 0
R13 NM - 0
R15 NM - 0
R16 NM - 0
DVDD_3.3v
DVDD_5v
GND
ETRAX_RESET
2,3,4 ETRAX_RESET
SPARTAN_UART_TX
SPARTAN_UART_RX
4 SPARTAN_UART_TX
4 SPARTAN_UART_RX
ETRAX_SP_TX
ETRAX_SP_RTS
ETRAX_SP_RX
ETRAX_SP_CTS
4 ETRAX_SP_TX
4 ETRAX_SP_RTS
4 ETRAX_SP_RX
4 ETRAX_SP_CTS
ETRAX_ASYNC0_TX
ETRAX_ASYNC0_RTS
ETRAX_ASYNC0_RX
ETRAX_ASYNC0_CTS
2,4 ETRAX_ASYNC0_TX
2,4 ETRAX_ASYNC0_RTS
2,4 ETRAX_ASYNC0_RX
2,4 ETRAX_ASYNC0_CTS
ETRAX_ETH_TX+
ETRAX_ETH_TX-
ETRAX_ETH_RX+
ETRAX_ETH_RX-
2 ETRAX_ETH_TX+
2 ETRAX_ETH_TX-
2 ETRAX_ETH_RX+
2 ETRAX_ETH_RX-
ETRAX_USB_OE
ETRAX_USB_SPEED
ETRAX_USB_VPO
ETRAX_USB_VMO
ETRAX_USB_VP
ETRAX_USB_VM
ETRAX_USB_RCV
2 ETRAX_USB_OE
2 ETRAX_USB_SPEED
2 ETRAX_USB_VPO
2 ETRAX_USB_VMO
2 ETRAX_USB_VP
2 ETRAX_USB_VM
2 ETRAX_USB_RCV
MCM_TYPE resistor. Mount if MCM4+16 22545+
Vcc
NP3#
BUSPWR#
EXTMEM#
EEDATA/GANGED#
DP0PUR
DP0
DM0
EECLK
MODE
SUSPND
RESET#
XTAL2
XTAL1/CLK48
DP1
DM1
DP2
DM2
DP3
DM3
PWRON1#
PWRON2#
PWRON3#
OVERCUR1#
OVERCUR2#
OVERCUR3#
NPINT1
NPINT0
OCPROT#/PWRSW#
VCC1
D1-
D1+
GND1
VCC2
D2-
D2+
GND2
CHAS9
CHAS10
CHAS11
CHAS12
IN
OUT1
OUT2
EN1#
EN2#
OC1#
OC2#
D+
D-
NC
OE#
SPEED
VPO
VMO/FSEO
VP
VM
RCV
TD+
CT_TD
TD-
RD+
CT_RD
RD-
CHASSIS_9
CHASSIS_10
C2+
C2-
V-
EN#
SD
V+
C1+
C1-
C3+
C3-
T1out
T2out
T3out
T1in
T2in
T3in
R1in
R2in
R3in
R4in
R5in
R1out
R2out
R3out
R4out
R5out
A
B
C
D
1
2
3
4
5


Rice University
Title
WARP Backdoor Board
Description
<Doc>
Rev
1.0
Date: Wednesday, August 20, 2008
Sheet 10 of 10
J4A
DF17_50
J4B
DF17_50
HDR_Ctrl[0:21] 4,5
HDR_Inhibit 5
HDR_MuxCtrl
HDR_Reset
HDR_CfgMode
HDR_ModSel0
HDR_ModSel1
HDR_ModSel2
HDR_CCLK
HDR_DIN
HDR_PROGB
HDR_INIT
DVDD_12v
DVDD_5v
DVDD_3.3v
DVDD_1.2v
DVDD_5.5v
GND
F1 NM - Fiducial
F2 NM - Fiducial
F3 NM - Fiducial
F4 NM - Fiducial
F5 NM - Fiducial
F6 NM - Fiducial
J8 NM - MHOLE
J9 NM - MHOLE
J11 NM - MHOLE
J12 NM - MHOLE
0.047uF
0.47uF