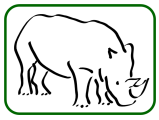
RHINO:

RECONFIGURABLE HARDWARE INTERFACE FOR COMPUTATIONAND RADIO



by

Simon Scott

A dissertation submitted to the Department of Electrical Engineering
in fulfilment of the requirements for the degree of

MSC ELECTRICAL ENGINEERING

at the

UNIVERSITY OF CAPE TOWN

Supervisors:

Dr Alan Langman
Prof Michael Inggs

March 2011

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

I declare that this project is my own unaided work; all sources that I have used have been referenced and are listed in the Bibliography. It is being submitted for the degree of Master of Science in Electrical Engineering at the University of Cape Town. This work has not been submitted before for any other degree or examination in any other university.

Signature of Author: .....

Cape Town

28 March 2011

ABSTRACT

Field-programmable gate arrays, or FPGAs, provide an attractive computing platform for software-defined radio applications. Their reconfigurable nature allows many digital signal processing (DSP) algorithms to be highly parallelised within the FPGA fabric, while their customisable I/O interfaces allow simple interfacing to analogue-to-digital converters (ADCs) and digital-to-analogue converters (DACs). However, FPGA boards that deliver sufficient performance to be useful in real-world applications are generally expensive. These high prices act as a barrier to entry to FPGA systems for smaller development teams and research groups.

In order to overcome this barrier, Rhino was developed. Rhino is an FPGA-based hardware processing platform that primarily supports software-defined radio applications. The goal of Rhino is to aid research and training in smaller university research groups and development teams with limited budgets. Rhino therefore aims to provide sufficient performance to be useful, while keeping the costs low.

The architecture for Rhino was designed so that it would best serve three target applications: radar, radio astronomy and bioinformatics. The architecture centres around a Xilinx Spartan-6 FPGA connected to an AM3517 ARM processor via a parallel bus. The FPGA is used for processing computationally-intensive algorithms, such as DSP operations for radar or matrix calculations for bioinformatics. The processor is used for coordinating the flow of data in and out of the FPGA, as well as for configuring the FPGA. The FPGA connects to ADC/DAC daughterboards using industry-standard FMC connectors. The FPGA can stream the data off the board via two 10Gbps Ethernet links, or buffer it in its 512MB of DDR3 SDRAM. A 1Gbps Ethernet link is also provided.

The processor runs BORPH, a Linux variant that allows users to communicate with the FPGA via the operating system's filesystem. The processor has a 100Mbps Ethernet link that allows users to access the board remotely and copy files to the processor over the network. The processor is supplied with 256MB of DDR2 SDRAM, and is able to boot from NAND flash memory, an SD card or from a network server. It also has USB host ports, a real-time clock and audio and video interfaces. The clocks on multiple Rhino boards can be synchronised to within 10ns of each other using the Precision Time Protocol, which runs over the Ethernet link. The entire board is monitored using power and temperature sensors.

The design of the manufactured Rhino board was verified using special test software and gateware. The board passed all software-based tests, proving that it performs as expected. Unfortunately, the FMC interface could only be tested at reduced speed, as no high-speed test hardware was available.

The final cost estimate for a complete Rhino system is under $1700, cheaper than similar FPGA boards that deliver much lower performance.

ACKNOWLEDGEMENTS

I would like to express my gratitude to the following people who assisted me during the course of this thesis.

Alan Langman for conceiving the idea of Rhino in the first place, and remaining involved throughout the duration of the project. I thank you for for always providing me with new ideas, for checking my work at each stage of the design process, and for the time you have spent organising the logistics of this project. I especially thank you for the all the stimulating discussions we have had over the past year. It has been a pleasure working with you.

Michael Inggs for overseeing the Rhino project and always ensuring that it stayed on track. Thank you also for all the input you provided during the editing stages of the dissertation. It is much appreciated.

Everyone who participated in the design reviews and provided such useful feedback. This includes, but is not limited to, **Henry Chen**, **Matt Dexter**, **David George**, **Philip Gibbs**, **David Hawkins**, **Francois Kapp**, **Pablo Sanchez** and **Jonathan Weintroub**.

The following organisations for funding this research through various scholarships: the **NRF** via the Prestigious Masters Scholarship, the **CSIR and the Department of Defense** via the Project Ledger Scholarship and the **University of Cape Town** through the Manuel & Luby Washkansky Scholarship.

Lastly, I would like to thank **SKA South Africa** for sponsoring the electrical components and the manufacturing of the PCBs. Without this funding, Rhino would not have been a reality.

GLOSSARY OF TERMS

ADC: Analogue to Digital Converter: a device that converts a continuous (analogue) signal to a discrete digital number. An analogue signal must be digitised by an ADC before it can be processed on an FPGA.

ASIC: Application-specific Integrated Circuit: an integrated circuit that is designed for a specific application, rather than for general use.

BGA: Ball Grid Array: a type of integrated circuit package containing an array of solder balls affixed to its underside.

DAC: Digital to Analogue Converter: a device that converts a digital signal into an analogue signal. It therefore performs the inverse operation of a ADC.

DIMM: Dual In-line Memory Module: a number of DRAM ICs mounted on a single PCB.

DSP: Digital Signal Processing involves the representation of signals as sequences of numbers, and these sequences are processed using mathematical algorithms.

FMC: FPGA Mezzanine Card: an ANSI standard that defines a standard mezzanine card form factor and connector interface to an FPGA located on a carrier board.

FPGA: Field Programmable Gate Array: an integrated circuit consisting of an array of logic cells, each of which can be programmed to perform a specific logic function.

Gateware: Gateware is for FPGAs what firmware is for embedded processors. It is the configuration data that is used to program/configure the FPGA.

HPC: High Pin Count (for FMC connectors): an FMC connector with 400 pins.

LPC: Low Pin Count (for FMC connectors): an FMC connector with 160 pins.

LVDS: Low voltage differential signalling: a standard for representing digital data using two separate voltage signals.

PCB: Printed Circuit Board. Also known as a printed wire board.

PLL: Phase-Locked Loop. A circuit that generates a clock signal with the same phase as an input reference signal, but with a frequency that may be a multiple of the reference frequency. Both the processor and the FPGA on Rhino contain PLLs for generating the correct internal clock frequencies.

Rhino: Reconfigurable Hardware Interface for computatioN and radiO. A low-cost FPGA board that targets software-defined radio, radio astronomy and bioinformatics applications. It is also the topic of this dissertation.

SDRAM: Synchronous Dynamic Random Access Memory: a type of RAM that requires regular refreshing (dynamic), and is synchronised to a system clock.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 THE NEED FOR NEW COMPUTING ARCHITECTURES

In the past, computer architects have always viewed increasing the clock frequency as the primary method of improving processor performance [1]. However, it is now generally accepted that this is no longer possible, due to what Asanovik et al. call the "power wall" [2]. This refers to the fact that by running microprocessors at multi-GHz clock speeds, semiconductor manufacturers have now hit the limit of the amount of power that a single integrated circuit can dissipate. Therefore, instead of trying to increase the clock speed of the processors further, semiconductor companies are instead improving performance by placing several smaller (and hence more power efficient) parallel processors on a single die [2].

Although Moore's Law (the trend of doubling the number of transistors in a microprocessor every two years) still holds, computer architects are no longer trying to double the number of transistors in a single microprocessor core. Instead, the new trend is to double the number of cores on a single multiprocessor die every two years. However, many believe that this approach of doubling the number of cores with each generation of processors will lead to diminishing returns once eight processors per die are exceeded [1]. Instead, it is believed that semiconductor companies should rather be designing multiprocessors that contain thousands of cores per die. This very large number of processors per die will result in both the highest performance (due to large number of parallel datapaths) and efficiency (as each core can be clocked at slower rate). Furthermore, the high level of parallelism will force programmers to stop thinking in terms of sequential processing, and rather write highly parallel software that makes the most efficient use of the computing resources available.

Unfortunately, processors with thousands of cores are probably still many years away. In the meantime, computer architects are looking at alternative processing devices that are inherently parallel in nature. Two such processing devices are graphics processors (formally known as graphics processing units, or GPUs) and field programmable gate arrays (FPGAs). An example of a high-end GPU is the NVIDIA Tesla C2050, which has 448 parallel processors [3]. Patterson [4] explains that this large number of parallel cores, coupled with the ease of use of GPUs, makes them an ideal platform on which to run software that is inherently parallel. GPUs are commonly used for digital signal processing applications (such as image and video processing), where speed-ups of ten times over traditional sequential CPU implementations are common. For example, Cope et al. recorded an 18 times throughput increase when using a GPU, instead of a CPU, to do primary colour correction [5].

The one disadvantage of GPUs is that they have a fixed hardware architecture. It is not possible to increase or decrease the number of cores as required, nor can one modify the datapath design of each processor to better suit the application. Furthermore, there is often a bottleneck transferring data in and out of the GPU over the

PCI and DRAM buses.

FPGAs, on the other hand, do not experience these shortcomings. They can be completely reconfigured to support a variable number of cores, and the datapath of each core is completely customisable. In fact, many manufacturers of embedded computing devices have already moved away from per-product ASIC designs to FPGA devices, due to "the need to amortize rapidly increasing design costs, the desire to minimize end-product design risk, and the value in providing flexible platforms" [2].

1.2 FPGAs as an Alternative Computing Architecture

A field programmable gate array (FPGA) can be defined as "a chip composed of an array of configurable logic cells (also called logic blocks). Each cell can be configured, or programmed, to perform one of a variety of simple functions... FPGA logic cells can be used as building blocks to implement any kind of functionality desired, from low-complexity state machines to complete microprocessors." [6]. An FPGA can therefore be regarded as a piece of programmable hardware, and since hardware is by its very nature parallel, FPGAs are inherently parallel processing devices. Depending on the application, an FPGA can be programmed to have one or one thousand parallel processing datapaths.

It is this reconfigurable parallel nature that makes FPGAs ideal for high-performance computing applications, such as physics simulations and DNA sequencing (see Yu et al. [7]). FPGA's also have a simple physical I/O interface, as each physical pin on the integrated circuit (IC) can be individually programmed to perform a specific function. This allows FPGA's to be easily connected to external components, such as memory ICs or analogue-to-digital converters (ADCs). Since software-defined radio applications are both DSP (digital signal processing)-intensive and require the use of ADCs and DACs, FPGAs are ideally suited to such applications. Examples of software-defined radio applications are radar and radio astronomy, where the FPGAs are typically connected to an analogue-to-digital converter (ADC) and are used to perform filtering and Fourier Transform operations on the sampled waveform.

However, there still remain definite barriers that hinder the development of FPGA-based systems. FPGA boards that deliver reasonable performance are very expensive. Roach, a Virtex 5 board used by the CASPER radio astronomy community [8], costs approximately $5300, while the USRP N210, the most readily available FPGA-based system for software-defined radio, costs $1700 [9]. The tools themselves are also very expensive: the basic development environment that is used for all Xilinx FPGAs (known as Xilinx ISE) costs $2995 per annum [10], while the MATLAB and Simulink tools required for developing systems for the CASPER boards cost $5250[1] [11].

These high prices act as a barrier to entry to FPGA systems for smaller development teams and research groups with limited budgets. Furthermore, many users of the MATLAB-Simulink environment in the CASPER community have complained about the long processing times when simulating their FPGA design, as well as the instability of the development environment [2], while the majority of USRP users never alter the gateware on the USRP's FPGA to better meet their application's needs, due to the steep learning curve involved in FPGA development.

From all the above mentioned reasons, one can only conclude that there is a clear need for an easy-to-use FPGA-based system that is low-cost, but still delivers sufficient performance to be useful.

[1] Prices were obtained directly from the relevant manufacturers, and were correct as of January 2011.

[2] These remarks regarding the MATLAB-Simulink environment were made numerous times during working-group discussions at the CASPER 2010 Workshop at the Harvard-Smithsonian Institute in Boston, USA in August 2010.

1.3 RHINO, A LOW-COST FPGA BOARD

In order to meet this need, the concept for Rhino was developed. Rhino is an FPGA-based hardware processing platform that primarily supports software-defined radio applications. The goal of Rhino is to aid research and training in smaller university research groups and development teams with limited budgets. Rhino therefore aims to provide sufficient performance to be useful in real-world applications, while keeping the costs low.

The original idea for Rhino was conceived by Dr Alan Langman, a researcher affiliated with the University of Cape Town. His outline for the project, which shall be called the "customer specification", stated that:

1. The board must contain both an FPGA and a processor, connected via a parallel bus. The architecture for the board should follow the style of other CASPER boards (such as Roach [8]), whereby the FPGA is used for data processing and the processor is used for control.
2. The FPGA must be the largest Xilinx Spartan-6 available, and the processor must be an Texas Instruments ARM processor.
3. Both the FPGA and processor must have RAM attached to them.
4. An I/O interface (for connecting ADCs and DACs) and a high-speed network link (for shipping the data off the board). must be connected to the FPGA.
5. The processor must run BORPH, a Linux-based operating system that allows users to communicate with externally connected FPGAs via the operating system's file system [12].
6. The processor must have an Ethernet connection to allow remote monitoring and control of the board.

1.4 RHINO TARGET APPLICATIONS

In order to keep the project focussed, three main applications were selected for the proposed Rhino system:

1. Radar
2. Radio astronomy
3. Bioinformatics

These three applications formed the focus of Rhino during the development process. All hardware design decisions were made by considering the effect it would have on these applications. Furthermore, the processing requirements of these applications were used to determine the hardware requirements for Rhino. Each of these applications are described in more detail below. Although the first two applications are often grouped under the common heading of "software-defined radio", they will be discussed separately here, as their processing requirements do vary.

1.4.1 Radar

Radar works by transmitting a radio-frequency (RF) signal towards a desired target, and observing the echo that is reflected back. This received echo signal needs to be processed before it can be of use. The processing typically involves down-conversion and matched filtering operations, which were traditionally performed using analogue circuits or custom ASICs [13]. Due to the high costs of producing ASICs, the low production

volumes of radar equipment, and the desire for flexibility in radar systems (for research applications), analogue circuits and custom ASICs are not ideal. Furthermore, according to Andraka and Berkin [13], typical radar digital signal processing (DSP) systems perform over 5 billion multiply operations per second, which is far more than a standard microprocessor-based system can handle. Since FPGAs are parallel devices, they can perform the large number of multiplies in parallel, and hence offer the performance of custom silicon, while maintaining the economies and flexibility of the microprocessor-based solutions.

In both radar and radio astronomy applications, an analogue-to-digital converter (ADC) is required to sample the received signal before it can be processed by the FPGA. The digital output of the ADC is typically connected directly to I/O pins on the FPGA. In some cases, the ADC samples the received waveform directly at the carrier frequency and the digital downconversion is performed in the FPGA, while in other cases an analogue front-end performs the downconversion before the signal is sampled. Furthermore, radar systems use digital-to-analogue converters (DAC) to generate the transmitted waveform. The digital inputs of the DAC are too driven by the FPGA.

1.4.2 **Radio Astronomy**

Radio astronomy can be regarded as the study of celestial bodies by detecting the radio waves that these bodies emit. As opposed to optical astronomy, which uses lenses and mirrors to capture the light waves emitted by celestial bodies, radio astronomy uses radio antennae to detect the radio waves.

Much like radar, traditional "radio astronomy instrumentation is highly specialized, with custom, complex, dedicated instruments being built for individual applications. Each instrument takes 3-5 years to design, construct, and debug, and by the time it is deployed, it has usually been made obsolete by the Moores Law growth of the electronics industry" [14]. Most radio astronomy applications require that several gigahertz of bandwidth be processed in realtime. Since microprocessor-based systems are non-parallel in nature, they would need a clock-rate higher than the datarate in order to process the data in real-time, which is not feasible.

However, due to their parallel nature, FPGAs are able to process the data in realtime, at a much slower rate, by splitting the data into a number of parallel processing paths. While this could also be done on a GPU, GPUs have a number of drawbacks, as was discussed in Section 1.1. FPGAs are typically used in radio astronomy to perform correlation, beamforming and wideband spectroscopy operations [14]. Such a FPGA-based wideband spectrometer is described by Chang, Wawrzynek and Brodersen [15]. Here the spectrometer is implemented using a polyphase filter bank (PFB), a corner turn and a Fast-Fourier Transform (FFT). The PFB is used to split the input signal into a number of coarse frequency channels (typically a few thousand channels). The output of the PFB is then recorded using a transpose matrix (called a corner turn), before it is fed into the FFT. The FFT then performs the detailed spectral analysis of the signal, typically providing 1 Hz (or better) resolution.

As was described in the previous section, the data is fed into the FPGA from an analogue-to-digital converter. In addition, high-speed data links (10Gbps) are also often required between different FPGA boards, when cross-correlation operations need to be performed, as described by Parsons, et al. [16].

1.4.3 **Bioinformatics**

Bioinformatics can be defined as the application of computer science and statistics to molecular biology. One of the most important research areas in bioinformatics is sequence analysis, where one attempts to determine the optimal alignment between two biological sequences (such as sequences of DNA nucleotides or sequences of proteins) [17]. This typically involves calculating the elements in dynamic programming matrix. Since a microprocessor can only calculate a single element at a time, while a FPGA can calculate multiple matrix

elements simultaneously using multiple datapaths, FPGAs can provide significant computational speedups over CPU implementations. Bioinformatics software tools that have been successfully accelerated on FPGAs include BLAST and HMMER2, where speedups of over 500 times for BLAST (by The Technical University of Crete [18]) and over 70 times for HMMER2 (by Derrien and Quinton [19]), have been recorded.

Obviously no ADCs or DACs are required for bioinformatics applications. However, a fast network is required (usually 1Gbps or higher) to transfer the sequence data between the PC (which stores the database of sequences on hard-disk) and the FPGA.

1.5 Analysis of the Requirements for Rhino

Now that the target applications for Rhino have been discussed, the exact requirements for such a low-cost FPGA board can be determined. These requirements were obtained by interviewing a range of specialists in the fields of radar, radio astronomy and bioinformatics from the University of Cape Town, SKA South Africa and Stanford University. The results of these interviews are summarised in the Table 1.1.

When reviewing these requirements, it is important to keep in mind that Rhino does not aim to provide the highest possible computing performance. There are a number of FPGA boards already available that do provide exceptionally high processing performance (such as the BEE3 [20]), but as a result are relatively expensive (the BEE3 costs around $25 000 with an academic discount). Furthermore, there are a number of FPGA development kits available from FPGA vendors that are reasonably cheap, and are very useful as development and test platforms, but unfortunately do not provide sufficient performance to do useful science in radio astronomy or achieve considerable speedups in bioinformatics applications. Instead, Rhino aims to provide sufficient performance to be useful in real-world applications, while keeping the costs low enough that it is affordable for smaller research groups and development teams with limited budgets.

Table 1.1: Requirements for Rhino for Target Applications

	Radar	**Radio Astronomy**	**Bioinformatics**
Performance	Synchronise multiple boards to within 10ns; read two 12 bit ADCs, each sampling bandwidth of 400MHz; 1Gbps network.	20Gbps network for inter-board communication; read two 8 bit ADCs with bandwidth of 500MHz; 20 Gbps DDR SDRAM interface[3]	1Gbps network to load database into DRAM. Minimum 1GB of DRAM.
I/O Interface	Minimum of 2 standardised interfaces for ADCs and DACs	Minimum of 2 standardised interfaces for ADCs, so that boards with different sampling rates can be used	None
Ease of use	Simple, rugged physical user interface (push buttons, etc.), as often used in field.	Simple toolchain that focuses on DSP aspects, and not technical details (most users are astronomers, not engineers)	Simple library support for DRAM and network access. No DSP support required.
Cost	Under $1800	Under $2000	Less than $8000[4]

No FPGA logic requirements, such as number of logic cells or DSP slices, have been given for each of the target applications. This is because the client specification stated that the Spartan-6 FPGA with the largest amount of logic resources must be used. It is therefore assumed that the largest device will have sufficient logic resources for each of the target applications.

By combining the different requirements for the target applications for a low-cost FPGA board, a single set of requirements can be produced. These combined requirements are shown in Table 1.2, and will be used later to define the detailed specification for Rhino.

In most cases, the combined requirements were determined by taking the maximum requirements in each category. However, in the case of the I/O Interface (input/output interface that is used to connect ADCs and DACs to the FPGA board), some calculations were required before the summarised requirements for Rhino could be determined. These calculations are shown in Appendix A, and assumed that the ADCs and DACs used I/Q sampling and LVDS data lines.

Table 1.2: Summary of Rhino Requirements

1	**FPGA resources**	Spartan-6 with largest amount of logic resources.
2	**SDRAM**	1GB of SDRAM with 20Gbps throughput.
3	**Networking**	20 Gbps high speed networking.
4	**Synchronisation**	Synchronise clocks on multiple Rhino boards to within 10ns.
5	**IO Interface**	Two independent ADC/DAC interfaces with standardised connectors. Each interface must support 32 LVDS pairs, and the FPGA must handle 400Mbps on each pair.
6	**Ease of use**	Minimalist physical user interface; simple toolchain that focusses on DSP application rather than technical details; easy-to-use software libraries for hardware peripherals.
7	**Cost**	Under $1800

1.6 SCOPE OF THIS THESIS

This thesis focusses on the design of the hardware for Rhino. This refers purely to the Rhino PCB itself, and does not include the design of any ADC/DAC cards or adaptor boards.

The software requirements were given in the preceding section, as they form part of the basic requirements for Rhino. However, the operating system and toolchain are beyond the scope of this thesis. They will only be discussed in so far as how they might influence the design of the hardware.

That said, low-level embedded software must still be run on the board to test the hardware and hence verify the design. Therefore, this thesis does cover the basic board-support software that was written to ensure that the hardware works correctly.

1.7 A NOTE ON NOMENCLATURE

Although Rhino can operate in standalone mode, more often than not it will be connected a desktop computer, laptop or server via USB or Ethernet. In order to improve readability, the phrase “personal computer” will be used to describe these computing platforms. Therefore, whenever the phrase “personal computer”, or the abbreviation “PC”, appears in this thesis, it may refer to a desktop computer, a laptop or a server.

[3] SDRAM data rate must be higher than combined ADC data rate.

[4] A typical commercial FPGA board for bioinformatics costs $16 000

1.8 STRUCTURE OF THE THESIS

Now that the requirements for Rhino and scope of this thesis have been defined, a review of existing FPGA boards that are used in software-defined radio applications is given in Chapter 2. Although the reviews focus on the hardware architecture of each board, a short explanation of the relevant software environment is given too. The FPGA boards that are included in this review are Roach, USRP N200 and BEE4. Although not strictly used for software-defined radio, the Xilinx Spartan-6 Development Board (SP605) is also reviewed, as it makes use of the Spartan-6 FPGA. Since none of these boards were found to meet the requirements for Rhino, the reviews are followed by a summary of the pros and cons of each board, and how each of the pros should be used, and each of the cons avoided, when designing a new FPGA platform.

By combining the customer specification from Section 1.3 and the requirements from Section 1.5, with the review of existing software-defined radio hardware in Chapter 2, a high-level architecture description for Rhino is developed in Chapter 3. This description specifies the main hardware elements for Rhino, at a block diagram level, without specifying which components will be used. The main hardware elements were found to be an FPGA with two ADC/DAC interfaces, at least 1GB of SDRAM and a high-speed network connection. The FPGA is connected to an ARM processor, via a bus, which contains USB, Ethernet and flash memory peripherals. This architecture description can be regarded as the "detailed specification" for Rhino. A brief description of the software architecture is given too, and how it affects the design of the hardware.

The architecture description is then followed by a comparison of the different component options in the second half of Chapter 3. These comparisons take the form of trade-off analyses, where the advantages of different component options are weighed up against each other. Analyses are done for digital and analogue power supplies, as well as for NAND and NOR flash memory. In these trade-off analyses, analogue power supplies were preferred over digital ones, and NAND flash memory was selected over NOR memory. The different component options for the FPGA, the processor, and the ADC I/O interface are also compared. The XC6SLX150T-4FGG676C was the chosen FPGA, the AM3517 was the chosen processor, and the FPGA Mezzanine Card (FMC) standard was selected for the ADC/DAC interface.

With the major components selected in Chapter 3, the detailed hardware design process can be described in Chapter 4. A detailed sub-system diagram is first developed, describing the four major sub-systems within Rhino (the FPGA sub-system, the processor sub-system, the debugging sub-system and the power management sub-system), and how they fit together. This sub-system diagram is then transferred to the schematics, the details of which are also discussed in this chapter. In these hardware discussions, emphasis is placed on how cost was minimised and testability was maximised at each stage in the design process. The chapter then ends off with an overview of the clocking infrastructure for the Spartan-6 on Rhino.

Chapter 5 describes the PCB-level design for Rhino. This is described in a separate chapter to the schematic-level design, as these represent two distinct design stages. Although the actual routing of the Rhino PCB did not form part of this thesis (it was outsourced), the rules and requirements for the layout and routing needed to be specified. This chapter therefore begins with a review of the theory behind the routing of high-speed digital signals. This is followed by the description of the PCB stackup (a 16 layer stackup was used for Rhino), and the routing rules that were defined. A discussion of the placement of the major components is also given. Finally, the procedure for simulating the high speed board traces is described, and the results of these simulations are given. These simulations showed acceptable signal integrity for all high-speed traces.

In order to verify that Rhino was designed and built correctly, a number of hardware and software tools were developed. Chapter 6 therefore describes the test rig that was built and the test programs that were written to check the hardware components on Rhino. These test programs took the form of standalone applications that ran on the processor, and basic gateware designs that ran on the FPGA. However, before the software tests

could be written, the basic board-support software needed to be developed. A discussion of the bootloader software and the low-level drivers is therefore also given.

The results of this hardware verification process are given in Chapter 7. This includes photographs of the completed Rhino PCB, the results of the testing that was performed during manufacturing, and the results of the software-based tests. The manufacturing tests, which included PCB trace impedance measurements and X-raying, all gave satisfactory results. All the software-based tests passed too, verifying that the board was designed and manufactured correctly. As with any large design, a few inevitable bugs in the hardware were uncovered during the testing process. These bugs are described, along with the hardware modifications that were made to correct the faults. Finally, a manufacturing cost analysis of the board is given, with the predicted cost sitting at $1635 for a complete Rhino system.

Chapter 8 concludes this thesis with a summary of the entire design process for Rhino. It also compares the final design to the original customer specification and requirements, showing that the board meets the cost, usability and functionality prerequisites.

In addition, Chapter 8 outlines the work that still needs to be done for Rhino. This is mainly the changes that need to be made to the PCB to correct errors in the current revision, before the next revision is released. Additional work that can be done to further reduce the board cost, such as reducing the number of PCB layers, is also described. Finally, the specifications for a rack-mount enclosure, which still needs to be built for Rhino, are given.

Four appendices have also been included. Appendix A outlines the calculations for determining the I/O requirements for Rhino, for each of the target applications. Appendix B gives the full Rhino schematics, while Appendix C gives the proposed layout of the Rhino Hide, the rack-mount enclosure for the Rhino board. Lastly, Appendix D lists the files on the attached CD.

CHAPTER 2

REVIEW OF EXISTING SOFTWARE-DEFINED RADIO HARDWARE

Before designing a new, low-cost FPGA board for software-defined radio applications, it would be sensible to investigate existing FPGA-based hardware that targets similar applications. The aim of such an investigation would be to identify the strengths and weaknesses of existing hardware, and build on these previous designs when developing a new system.

This chapter therefore reviews three FPGA boards: ROACH, USRP N200 and BEE4. These three platforms are all currently used for at least one of Rhino's target applications. Although not typically used for any real-world applications, the Xilinx SP605 is also reviewed. The SP605 is Xilinx's development board for Spartan-6 FPGAs, and hence contains many design elements that may be useful in a commercial Spartan-6 board.

Each of these four FPGA-based hardware platforms are reviewed below. While the emphasis of each review is on the hardware architecture of each board, the software environment is also briefly described.

2.1 ROACH

ROACH is a Virtex-5 based platform, designed by SKA SA (the organisation responsible for building the Square Kilometre Array in South Africa), primarily for radio astronomy applications. ROACH stands for "Reconfigurable Open Architecture Computing Hardware", and forms part of the collection of FPGA boards used for signal processing by the CASPER radio astronomy community. CASPER is an international collaboration of radio astronomers who aim to "streamline and simplify the design flow of radio astronomy instrumentation by promoting design reuse through the development of platform-independent, open-source hardware and software." [21]. A photograph of ROACH is given in Figure 2.1.

One of the main applications of ROACH is a packetised correlator. A standard radio astronomy correlator requires a large number of interconnections between different receiver boards. To simplify these interconnections, a packetised correlator uses a high speed network switch (typically 10Gbps Ethernet switch) to connect the FPGA boards together [22].

Since ROACH is aimed at large radio astronomy installations (such as the Greenbank Telescope[1] and MeerKAT[2]),

[1] http://www.gb.nrao.edu/gbsapp/
[2] http://www.ska.ac.za/meerkat/overview.php

it has been designed to process high data rates on both the ADC I/O side and on the network side. Furthermore, these large radio astronomy installations are typically very expensive, due to the cost of the antennas and the analogue electronics. Therefore, there has been little need to minimise the cost of ROACH.

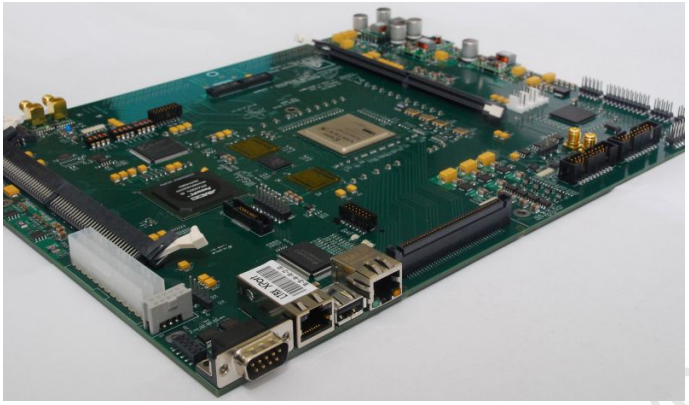


Figure 2.1: Photograph of a ROACH board
(Image courtesy of CASPER: http://casper.berkeley.edu/wiki/ROACH)

2.1.1 **Hardware Architecture**

The architecture of ROACH is shown in Figure 2.2. At the heart of ROACH lies a Virtex-5 FPGA, which is responsible for performing all the data processing on the board. The FPGA uses two Z-DOK+ connectors (each 40 differential data pairs) to interface with ADC and DAC cards. For networking, the board provides four CX4 connectors, each supporting 10Gbps Ethernet, resulting in a maximum network throughput of 40Gbps.

ROACH has both QDR II+ SRAM and DDR2 SDRAM connected to the FPGA. As stated on the ROACH website [8], the quad data rate (QDR) SRAMs provide 9MB of high speed memory for performing the previously-mentioned corner turn (matrix-transpose) operation that is used in radio astronomy designs. The DDR SDRAM, on the other hand, provides higher latency, but larger capacity (typically 1GB) memory, mainly for buffering. The FPGA also has a couple auxiliary SMA clock connectors, GPIO headers and LEDs.

In addition to the Virtex-5, ROACH also contains a separate PowerPC chip. The PowerPC is used to both program the FPGA and control the device once it has been programmed, via a parallel bus. Since the PowerPC runs Linux, users can easily communicate with the processor via its Ethernet connection. Furthermore, the PowerPC is running a modified version of Linux called BORPH, which allows users to access registers on the FPGA by reading and writing virtual files within Linux [12]. In terms of memory, the PowerPC uses a 64MB flash memory chip to store the bootloader and operating system, while the (typically) 1GB of DDR2 SDRAM helps to keep the operating system running. Note that no actual DDR2 SDRAM is provided on the ROACH board itself. Instead, DIMM (dual in-line memory module) sockets are connected to both the FPGA and processor, allowing users to plug in as much DDR2 SDRAM as they require.

The processor provides both USB and RS-232 interfaces for additional I/O. Additionally there is an SD card socket, allowing the PowerPC to load files off an SD card. However, the PowerPC does not natively support the SD card interface, and so an additional CPLD is required to perform the protocol conversion.

Lastly, the power management and monitoring of ROACH is powered by an Actel Fusion mixed-signal FPGA. This FPGA monitors all the voltages, currents and temperatures on the board. It also allows remote monitoring of the the health of the board, and will automatically shutdown the board in the case of over-/under-voltage, over-current or over-temperature.

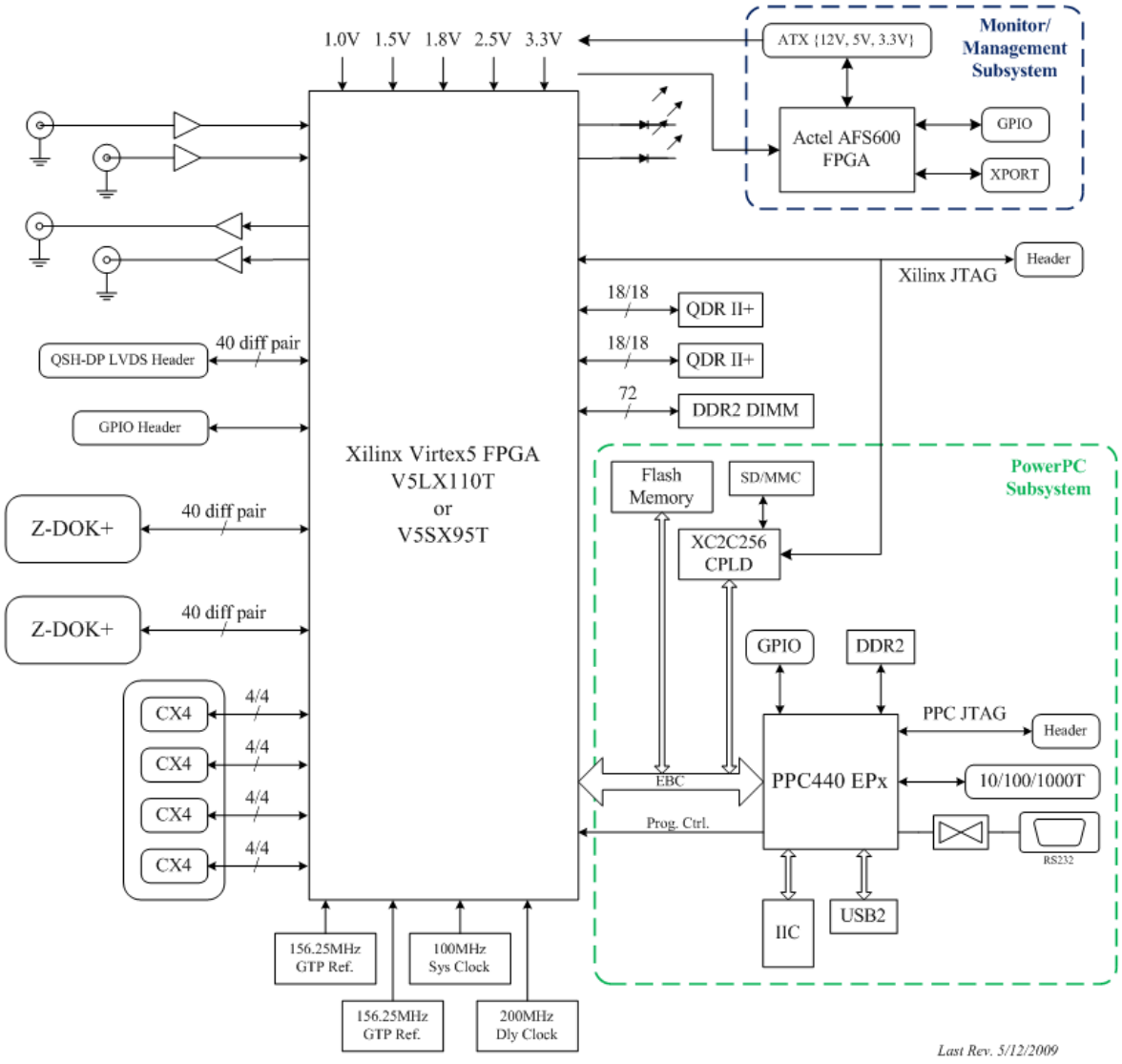


Figure 2.2: The architecture of ROACH
(Image courtesy of CASPER: http://casper.berkeley.edu/wiki/ROACH_Architecture)

2.1.2 Software Environment

As it ships, the following software is installed on the flash memory on ROACH:

1. U-boot, a simple bootloader environment
2. BORPH, a custom Linux kernel
3. Busybox, a massively stripped-down filesystem for Linux.

Since the BusyBox file system includes only the most essential files, users typically load their full filesystem over a network using NFS (a network filesystem protocol), from a USB flash drive, or off an SD card.

Probably the biggest highlight of the ROACH software environment is BORPH. BORPH is an "operating system that extends a standard Linux system with integrated kernel support for FPGA resources" [12]. This allows users to run FPGA hardware designs from within Linux, as standard Linux processes. The advantage of this is that the FPGA is able to access the Linux filesystem (via the bus between the FPGA and processor), reading and writing virtual files, and hence providing users with a simple interface to registers on the FPGA.

Gateware designs for the ROACH board are usually developed in Simulink (part of MATLAB), using Xilinx System Generator to convert the Simulink design into a bitstream that can be programmed onto the FPGA. The advantage of using Simulink is that users can build systems by graphically connecting together processing blocks, such as filters and FFTs. The disadvantage to this approach, however, is that large designs can become very difficult to manage, due to the sheer number of graphically interconnected blocks.

2.1.3 **Strengths of ROACH**

The following points can be regarded as the main strengths of the ROACH design:

- The Virtex-5 FPGA provides good performance in terms of speed and number of logic slices. The large volume of DDR2 SDRAM and QDR II SRAM means that ROACH should be able to meet the processing requirements of most applications.
- Having a separate processor on the board, running BORPH, provides the user with a simple interface to monitor and control the hardware design running on the FPGA. This also allows the FPGA to be programmed without the need for special JTAG programmers.
- The real-time monitoring of temperatures, voltages and currents provides assurance that the board is running correctly, and if not, it will turn off before any damage can be done.
- Simulink provides an easy-to-use environment in which new hardware designs can be rapidly built, as long as they are not too large or have multiple clock domains.

2.1.4 **Weaknesses of ROACH**

ROACH does however have a number of pitfalls, such as:

- The board is expensive, costing around $5300.
- It uses a non-standard ADC interface (the Z-DOK+ connectors). This means that only ADC/DAC boards designed specifically for the ROACH can be used.
- The processor bootloader on the flash memory can only be programmed via JTAG. This means that if the bootloader becomes corrupt (and reports on the CASPER mailing list confirm that this does happen), a JTAG programmer is necessary to reprogram the bootloader.
- While using DIMM connectors gives users freedom regarding the amount of SDRAM used on the board, it has been known to cause compatibility problems. Since ROACH has been designed to work with specific memory timings, there have been reports of commercially-available DIMMs that do not work with ROACH.
- A CPLD interface is required to connect the SD card to the PowerPC, making the design complex. Furthermore, the PowerPC has no DMA support for the SD card.

- The only method to ship large volumes of data in and out of the FPGA is via the CX4 connectors. This means that even at low data rates (such as 1Gbps), expensive 10Gbps Ethernet hardware (switches and PCI-Express cards for the PC) are required.
- ROACH has a complex "bringing up" process: the Atmel Fusion FPGA, the CPLD that interfaces to the SD card, and the flash memory attached to the PowerPC need to be programmed using JTAG programmers before the board can be used for the first time.
- The MATLAB, Simulink and System Generator tools are very expensive, costing $5250.
- Simulink is known to crash regularly, apparently on a daily basis for heavy users. Furthermore, the graphical designs in Simulink become unwieldy if the design is large.

2.2 USRP N200 AND N210

The USRP (Universal Software Radio Peripheral) N200 and N210 are FPGA boards, designed by Ettus Research, specifically for software-defined radio applications. According to the USRP N200 series datasheet [23], these applications include broadcast TV, mobile telephone network base-stations and satellite navigation, in both academic and industrial sectors. Both the N200 and N210 support a wide range of RF front-ends (known as "daughterboards"), allowing the USRP to handle radio applications at any frequency up to 6GHz.

Furthermore, the low cost of these devices (the N210 costs $1700), and the rugged, small form-factor enclosures (shown in Figure 2.3 below) make them popular development platforms in various software-defined radio research fields. Note that most of the information provided here regarding the USRP N200 and N210 has been taken from the USRP N200 series datasheet [23].

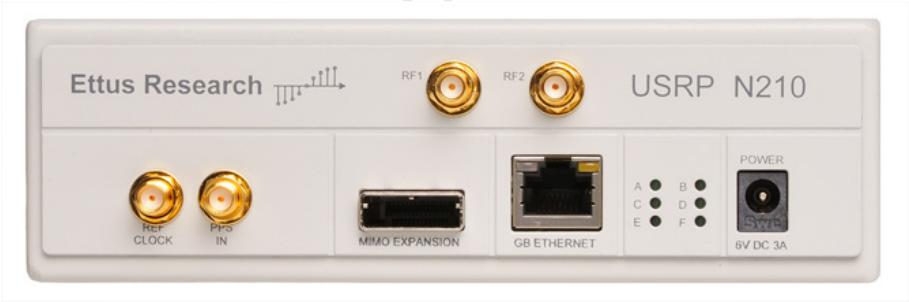


Figure 2.3: The USRP N210
(Image courtesy of Ettus Research LLC: http://www.ettus.com/products)

2.2.1 Hardware Architecture

Both the USRP N200 and N210 are powered by Xilinx Spartan-3A FPGAs. The only difference between these two devices is the resources available on the FPGA: the N200 contains a Spartan-3A XC3SD1800A (with 37 000 logic cells), while the N210 contains a Spartan-3A XC3SD3400A (with 54 000 logic cells). Both of these Spartan-3As are however still regarded as low-cost FPGAs. In both cases, the FPGA's block RAM (i.e. memory within the FPGA itself) is supplemented by a 1MB high-speed SRAM chip.

Looking at Figure 2.4, which shows the architecture for the USRP2 (which has the same architecture as the USRP N200 series), one immediately notices the lack of a separate processor. This is one of biggest differences between the USRP N200 series and the ROACH. The USRP N200 series uses a MicroBlaze soft-processor (i.e. a RISC processor programmed into the FPGA fabric) to manage Ethernet communications,

configure the daughterboards, and control the flow of data within the FPGA pipeline. The soft-core can also be used to perform true software processing of the data, by running C programs on the processor. The advantage of the soft-core approach is that it keeps costs down (fewer components and simpler PCB) and provides a more flexible interface between the FPGA datapath and processor. The disadvantage, however, is that the MicroBlaze core uses approximately a third of the logic and memory resources on the FPGA, reducing the amount of space available for signal processing.

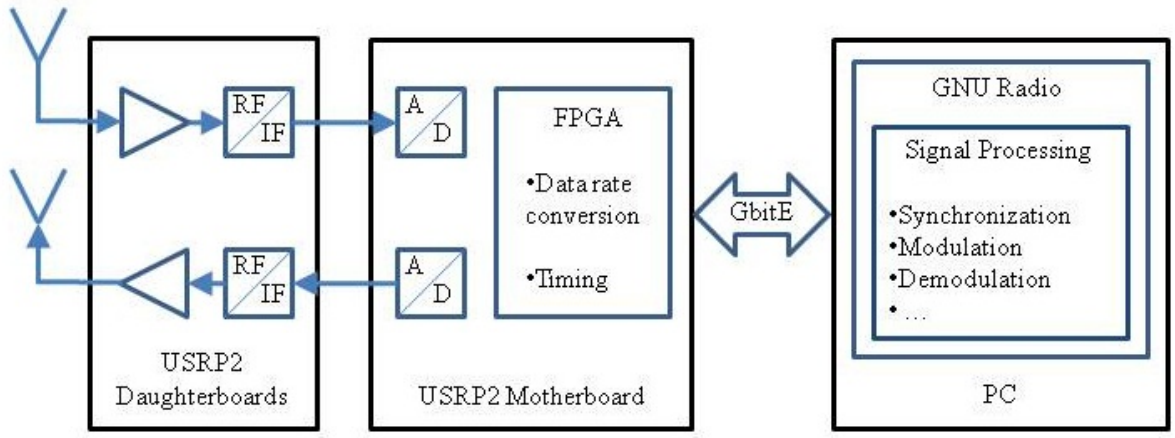


Figure 2.4: The USRP2 and USRP N200 series architecture
(Image courtesy of Amr El-Sherif, Qatar University:
http://confluence.qu.edu.qa/display/NPRPRESEARCH/USRP2+Testbed)

The other difference to ROACH is that the ADCs and DACs are built onto the board itself, while only the RF front-ends are placed on separate daughterboards. A large number of different RF front-ends are available, with (in some cases, tunable) centre frequencies up to 6GHz. All of these daughterboards plug onto the USRP mainboards using industry-standard PCI Mezzanine Card (PMC) connectors. On the receiver chain, the analogue baseband signals are sampled by two 14-bit, 100MS/s ADCs on the USRP main board. Two separate ADCs are provided to allow I/Q sampling. Similarly, two 16-bit, 100MS/s DACs on the USRP board provide the signal source for the transmit chain.

As mentioned previously, the Spartan-3A is a low-cost device with limited logic resources. Furthermore, a third of the FPGA's resources are consumed by the MicroBlaze soft-processor. This means that the FPGA is not able to perform any complex signal processing [24]. Therefore, the FPGA on the USRP N200 series typically just reads the ADCs, performs some basic filtering and sample-rate conversion, and then packetises the data for transmission over the 1Gbps Ethernet link. The data is streamed to a personal computer for further processing in software. The reverse occurs during data transmission.

Multiple USRP boards can be connected together using the MIMO expansion connector. This connector provides a high speed serial link and a reference clock to synchronise multiple boards. The USRP N200 series boards can also be clocked externally using the clock input SMA connectors on the front of the box, or can be synchronised using the add-on GPS-disciplined oscillator module.

2.2.2 **Software Environment**

Typically, little processing other than the standard filtering and sample-rate conversion is done on the FPGA. It has been reported that very few users modify the prebuilt FPGA design, due to its poor documentation and lack of API (gateware libraries) for custom designs. Although it is easier to write custom firmware (C code)

for the MicroBlaze, allowing the USRP to act as a standalone system, the USRP is still most commonly used with a host PC.

In this configuration, the FPGA and MicroBlaze processor are programmed with the prebuilt software, and the actual signal processing is done on the PC, within the free, open-source, GNU Radio framework. GNU Radio allows users to develop a signal processing system by connecting together signal processing "blocks" (such as filters, mixers, FFTs) using Python scripts [25]. GNU Radio also includes a number of easy-to-use graphical interfaces for visualising the data.

The FPGA bitstream and custom processor firmware are stored on the flash memory on the USRP board. The flash memory can be programmed over the network, making development a simple process.

2.2.3 Strengths of the USRP N200 Series

- Relatively low cost: the USRP N210 retails for $1700.
- The RF daughterboards use an industry-standard PMC connector.
- The USRP N200 series has a well-developed mechanism for synchronising multiple boards, using either the 1PPS clock input, the MIMO cable or the GPS-disciplined oscillator.
- The 1Gbps Ethernet support means that low-cost network hardware can be used.
- The hardware integrates seemlessly with GNU Radio, providing a easy-to-use [24] Python environment in which users can develop software-defined radio applications. Furthermore, GNU Radio is a completely free, open-source, software tool.
- The FPGA and processor can be easily reprogrammed over the network.

2.2.4 Weaknesses of the USRP N200 Series

- The USRP N200 series of FPGA boards can only process 100MHz of bandwidth within the FPGA (the ADCs sample at 100MS/s, with complex sampling).
- Only 50MHz of signal bandwidth can be streamed to a PC for further processing, due to the bandwidth limitations of the 1Gbps Ethernet link.
- The MicroBlaze soft-processor uses approximately 33% of the FPGA logic and memory resources. Since the FPGA is a low-cost device, its total resources are limited, which means that only the most basic signal processing can be performed on the FPGA.
- It can be difficult for the average user to develop their own custom FPGA and processor code, due to the lack of documentation and libraries.

2.3 BEE4

BEE4 stands for "Berkeley Emulation Engine 4". BEE was first developed as a processor emulation platform to speed up the development of new processor architectures. It is described by Davis, Thacker and Chang in the Microsoft Technical Report on BEE3 [26], as a platform where "researchers can rapidly prototype a variety of architectures in a relatively short amount of time by using a repository of low-level component designs" . The BEE platform was initially developed at the University of California, Berkeley, but the latest iterations (BEE3 and BEE4) have been developed by BEEcube.

Although the BEE platform was originally intended for processor emulation, people soon realised that it had enormous potential for software-defined radio applications. Hence, software-defined radio variants of both the BEE3 and BEE4 were released, named BEE3-W and BEE4-W, respectively. The BEE4-W (which shall be discussed in this review) is a stackable, multi-FPGA based prototyping platform, with integrated ADC and DAC modules, for applications such as radar, software-defined radio and communications [20]. The BEE4-W is pictured below in Figure 2.5.

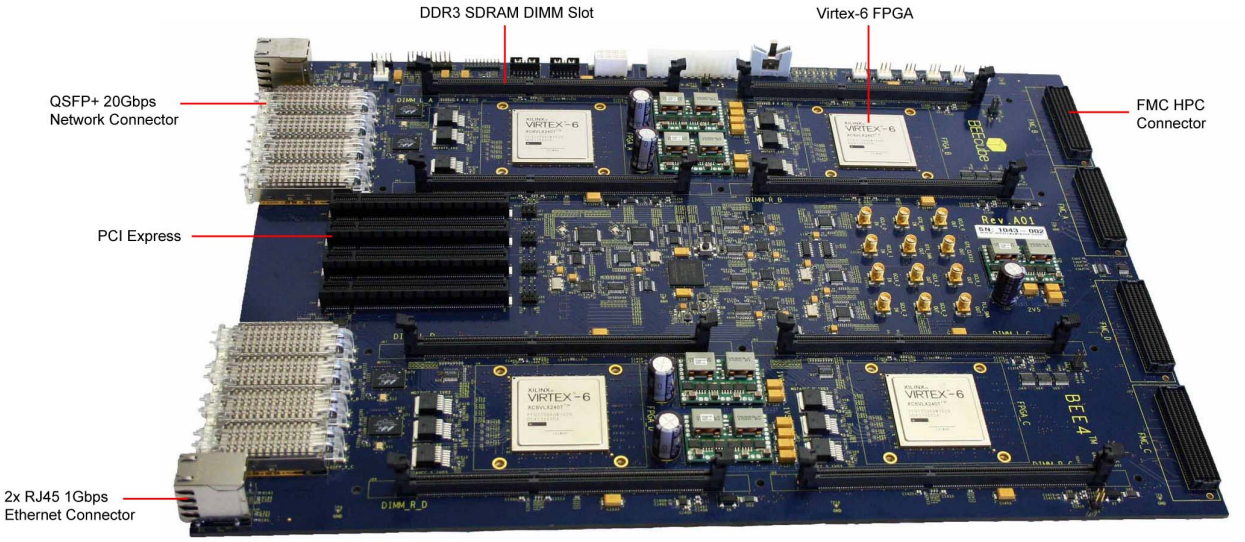


Figure 2.5: The BEE4 FPGA board
(Original image courtesy of BEEcube: http://beecube.com/products)

2.3.1 **Hardware Architecture**

The BEE4 contains four Virtex-6 FPGAs (Xilinx's latest family of performance FPGAs), interconnected via a ring bus. Each FPGA has its own independent memory and I/O, resulting in a symmetrical architecture of four identical processing cells, each containing an FPGA with its own resources. An Intel Atom processor is used to configure the FPGAs and monitor the hardware, while datapath control is carried out by a Microblaze softcore on one (or more) of the FPGAs.

The architecture of one of these processing cells is shown in Figure 2.6. Each cell contains a single Virtex-6 FPGA, connected to two DDR3 SDRAM DIMM slots. High-speed network access comes in the form of two QSFP+ (Quad Small Form-factor Pluggable) connectors. Each of these connectors support Ethernet up to 20Gbps, providing a total network bandwidth of 40Gbps per FPGA. Each cell also contains an FMC (FPGA Mezzanine Card) connector to interface to a single ADC or DAC card, supporting sample rates well over 2GS/s. Since FMC is an ANSI/VITA standard, there are a number of commercially available ADC and DAC cards that can be used with the BEE4-W, such as the FMC110 dual ADC/DAC FMC card from 4DSP [27].

Each processing cell also contains a single 1Gbps Ethernet link for control and monitoring, and for data transfer in applications where high-speed data rates are not required. A number of other peripherals (such as RS232, LEDs and PCI Express) are also included in each processing cell, and are shown in Figure 2.6.

As mentioned, the BEE4-W contains four identical FPGA cells connected in a ring topology. This provides the board with a total of 20 million configurable logic gates, 160Gbps network throughput and support for up to 128GB of DDR3 SDRAM [28]. Each board is also able to support up to four FMC ADC/DAC cards.

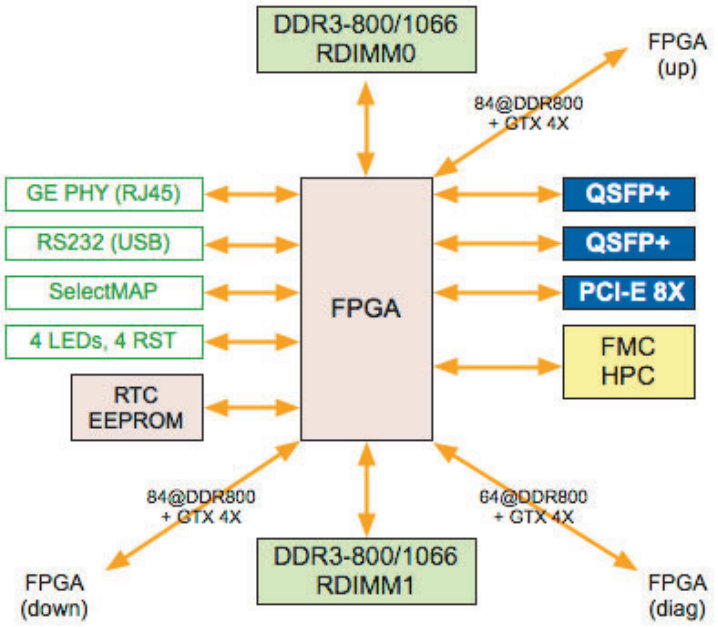


Figure 2.6: The architecture of a single FPGA processing cell on the BEE4-W
(Image courtesy of BEEcube)

Furthermore, the BBE4-W platform is scalable, allowing up to 80 boards to be interconnected via high-speed QSFP+ data links. At $25 000, the BEE4-W is clearly a performance system, rather than a low-cost system.

2.3.2 Software Environment

The development environment for the BEE4 is called BEEcube Platform Studio, or BPS. It is very similar to that of ROACH, in that new systems are developed using the Simulink block diagram tool within MATLAB, and the resulting FPGA bitstream is generated using Xilinx System Generator.

According the BEEcube, "months of engineering tasks to convert complex DSP algorithms to implementation can be achieved through BPS in a matter of days, all without requiring user knowledge of the low level FPGA implementation details" [29]. This is facilitated by the large number of Simulink library blocks written by BEEcube, for the SDRAM, network and I/O interfaces on BEE4. The user is also able to draw on the Xilinx DSP blocks available within Simulink.

This means that creating a new design for the BEE4-W is as simple as connecting together graphical blocks to carry out DSP, communication and I/O functions. However, like ROACH's development environment, this approach does result in unwieldy block diagrams when the designs are large.

2.3.3 Strengths of the BEE4

Some of the main strengths of the BEE4 are:

- The Virtex-6 FPGAs provide excellent performance (the logic can be clocked at up to 500MHz [20]), and lots of logic resources (20 million gates per board).
- The BEE4-W uses an industry standard connector (FMC) to interface to the ADC and DAC cards. As a

result, the board can be used with a number of commercially available ADC and DAC cards.

- The board provides both 1Gbps and 20Gbps Ethernet interfaces, giving users more flexibility.
- The use of QSFP+ network interfaces, rather than CX4, is an advantage, as many network switch manufacturers are no longer producing CX4-based hardware [30].
- The BEE4 can be fully operated and monitored remotely via Ethernet.
- The system is scalable: up to 80 boards can be connected together using the QSFP+ connectors.
- The use of a soft-processor (MicroBlaze) for datapath control, rather than a separate processor chip, is advantageous in this case, as the FPGAs are so large that the logic used by the MicroBlaze core is negligible. This approach simplifies the PCB design, as no external high-speed traces are required.
- Simulink, together with the pre-written library blocks, provides users with a rapid application development environment.

2.3.4 Weaknesses of the BEE4

The BEE4-W does however have two very big pitfalls:

- The main drawback of the BEE4 is the price. At approximately $25 000, it is accessible to only the largest research and development teams.
- Simulink is reportedly unstable at times and the block diagrams can become unwieldy for large designs.

2.4 XILINX SP605

The SP605 is Xilinx's evaluation board for the Spartan-6 FPGA. While it does not contain the fastest or largest Spartan-6 FPGA, it does contain many features common to FPGA-based embedded systems, such as DDR3 SDRAM, Ethernet support and a PCI Express interface.

Since the SP605 is not a software-defined radio platform, nor is it designed for real world applications, the strengths and weaknesses of the board will not be evaluated. Rather, the basic architecture of the SP605 will be described, as it serves as a base template for the design of other Spartan-6 based boards, including Rhino.

2.4.1 Hardware Architecture

The SP605 is shown in Figure 2.7. The core processing components of this board are a mid-range Spartan-6 (XC6SLX45T-3FGG484) FPGA and a 16-bit DDR3 SDRAM IC [31]. Since the Spartan-6 memory controller supports only discrete memory components (i.e. separate memory chips, and not DIMMs), it would not have been possible to place a DIMM connector on the board.

The SP605 has a wide range of peripherals to support a number of applications. These include an SFP connector for high-speed Ethernet, an RJ45 connector providing 1Gbps Ethernet and an FMC connector to support the use of ADC/DAC cards. The SFP and FMC connectors connect directly to the FPGA, while the 1Gbps Ethernet port is managed by a Marvell Ethernet PHY. There is also a DVI video connector, a USB connector (which provides a virtual serial port), and a PCI Express finger.

The FPGA is typically programmed via JTAG, using the Xilinx IMPACT software.

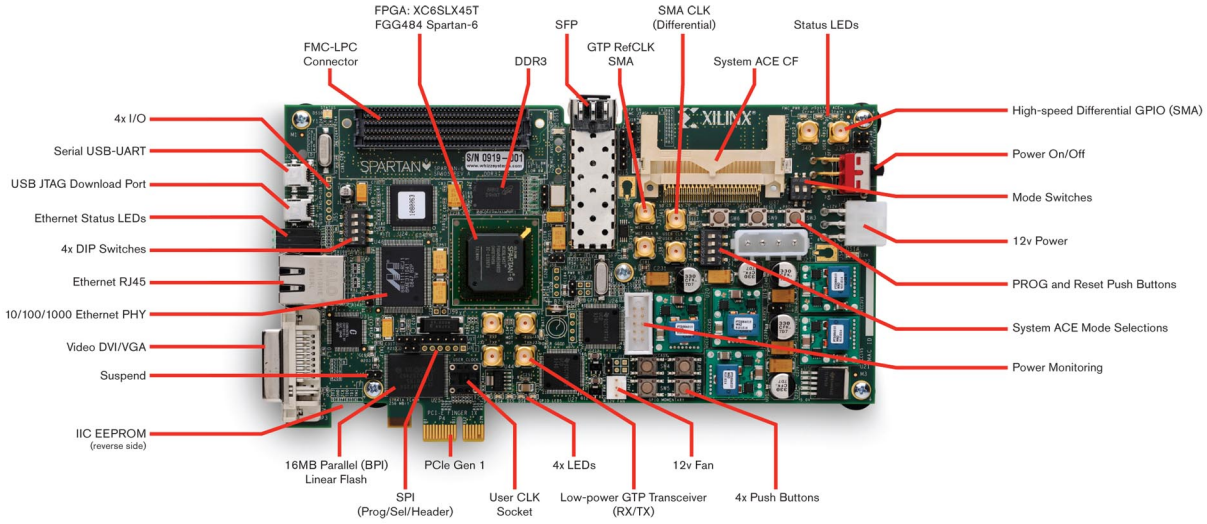


Figure 2.7: The Xilinx SP605 development board
(Image courtesy of Xilinx: http://www.xilinx.com/publications/prod_mktg/sp605_product_brief.pdf)

2.5 Outline for a New, Improved, Hardware Platform

Three commercially-available FPGA boards that target software-defined radio applications have been reviewed. However, none of these boards fully meet the requirement for a low-cost platform that has sufficient performance to be useful in real-world radar, radio astronomy and bioinformatics applications. The ROACH and BEE4 were both too expensive for smaller research and development teams, while the USRP does not provide sufficient performance nor resources.

Even though none of these boards sufficiently meet the requirements, the reviews have provided useful information on what should be included and what should be avoided when designing a new FPGA-based platform. This information is summarised below.

Hardware guidelines:

1. The FPGA should have sufficient logic resources, and speed grade, to meet the requirements of the application. The FPGA must be able to do more than simple filtering and sample-rate conversion. The Spartan-3A does not meet these requirements, while the Virtex-6 is too expensive for a low-cost board.

2. The board must also have a processor running Linux, providing users with a simple terminal interface to the board. The processor should be a physical processor chip, and not a soft-core on the FPGA.

3. One must be able to program the FPGA without the need for special hardware (such as a JTAG programmer). One solution would be to run a data bus and configuration bus between the processor and the FPGA, and use the processor to configure and control the FPGA. This could be extended further by running BORPH on the processor.

4. The board should be powered by an off-the-shelf power supply, simplifying power supply requirements.

5. All voltages, currents and temperatures should be monitored in real-time, and if any of them reaches a critical level, the board should be automatically switched off.

6. The FPGA board should be cheap enough for small research groups and development teams. Due to the popularity of the USRP with such groups, a price under $1800 would probably be appropriate.

7. The board should use an industry-standard connector for interfacing to ADC and DAC cards. For example, if the FMC interface was used, the FPGA board would support the large number of third-party cards available on the market.

8. Since the Spartan-6 (the specified FPGA for Rhino) supports only discrete SDRAM chips (and not DIMMs), discrete SDRAM chips must be used on the board. The advantage is that this avoids all DIMM compatibility problems. However, the SDRAM chips must provide sufficient capacity and throughput to meet buffering requirements (especially for radio astronomy).

9. The board should have both 1Gbps and 10Gbps Ethernet interfaces. This gives the user the choice between bandwidth and cost savings.

10. The "bringing-up" process after the board is first powered up should be made as simple as possible. This can be done my minimising the number of programmable devices on the board, and ensuring that all FPGAs, CPLDs and processors can be programmed via a single Ethernet connection.

11. There should be a well-managed mechanism for synchronising the clocks on multiple Rhino boards. This could either be done by providing auxiliary clock inputs on each board, or by using a cable similar to the MIMO one used on the USRP.

12. If possible, QSFP+ (or at least SFP+) connectors should be used for the high speed networking interfaces, rather than CX4. This is due to network hardware manufacturers no longer producing CX4 10Gbps Ethernet switches.

13. The new FPGA system must be scalable.

Software guidelines:

1. A software and gateware (FPGA logic) development environment should be provided with the board that is both easy-to-use and contains many library modules. The environment should allow users to simply plug together existing library blocks, facilitating rapid application development.

2. The environment should not however be graphical in nature, but rather text based. This will make the management of larger systems easier. One such possibility would be to use Python-based tools, such as MyHDL for FPGA gateware development or GNU Radio for PC-based post-processing.

3. Good documentation and support libraries should be provided to facilitate the development of new systems.

4. Free, open-source software should be used where possible.

5. Users must be able to fully monitor and control the board remotely. Users should be able to program the FPGA and get results back without ever having to be in the same room as the physical hardware.

Three existing FPGA boards have been reviewed in this chapter, all of which had applications in the field of software-defined radio. Although these boards varied in performance and cost, none of them met the specific cost and performance requirements for Rhino. Therefore, the strengths and weaknesses of each board have been combined to form the guidelines shown above. In the next chapter, these guidelines are combined with the original customer specification and requirements to determine the architecture for Rhino.

CHAPTER 3

THE ARCHITECTURE OF RHINO

After reviewing a number of existing FPGA boards in the previous chapter, the guidelines for a new low-cost FPGA board were drawn up. These guidelines can be combined with the customer specification and requirements from Chapter 1 to obtain a high-level block diagram for the proposed Rhino FPGA board. This high-level block diagram can be regarded as the "detailed specification" for Rhino.

Once the high-level block diagram has been developed, the major components for Rhino are selected in the latter half of this chapter. This includes decisions such as processor selection, type of I/O interface connector for ADC/DAC connectors and SDRAM architecture. These major components are then plugged back into the high-level block diagram to obtain the detailed architecture diagram for Rhino.

3.1 HIGH-LEVEL BLOCK DIAGRAM AND DETAILED SPECIFICATION FOR RHINO

The original customer specification (as described in Section 1.3) is now combined with the results of the requirements analysis (from Section 1.5) and the review of existing FPGA boards to obtain a high-level block diagram for Rhino. This diagram, which also shows the performance requirements for each component, is shown in Figure 3.1.

This diagram can be regarded as the "detailed specification" for Rhino. It is described in more detail below. Note that each point in the detailed specification below can be traced back to either the customer specification, the requirements analysis or the hardware guidelines that were obtained after reviewing similar FPGA boards. To facilitate the traceability of the detailed specification, each point below is accompanied by one or more of the following symbols: CS[1-6], RQ[1-7], HG[1-13] or SG[1-5]. They can be described as follows:

- CS[1-6] refers to the six customer specification points that were given in Section 1.3.
- RQ[1-7] refers to the seven requirements for a low-cost FPGA board that were summarised in Table 1.1.
- HG[1-13] refers to the thirteen hardware guidelines that were drawn up at the end of the previous chapter, after reviewing existing FPGA boards.
- SG[1-5] refers to the five software guidelines that were drawn up at the end of the previous chapter.

The use of these symbols allows each of the points in the following detailed specification to be traced directly back to either a customer specification, a performance requirement or a guideline.

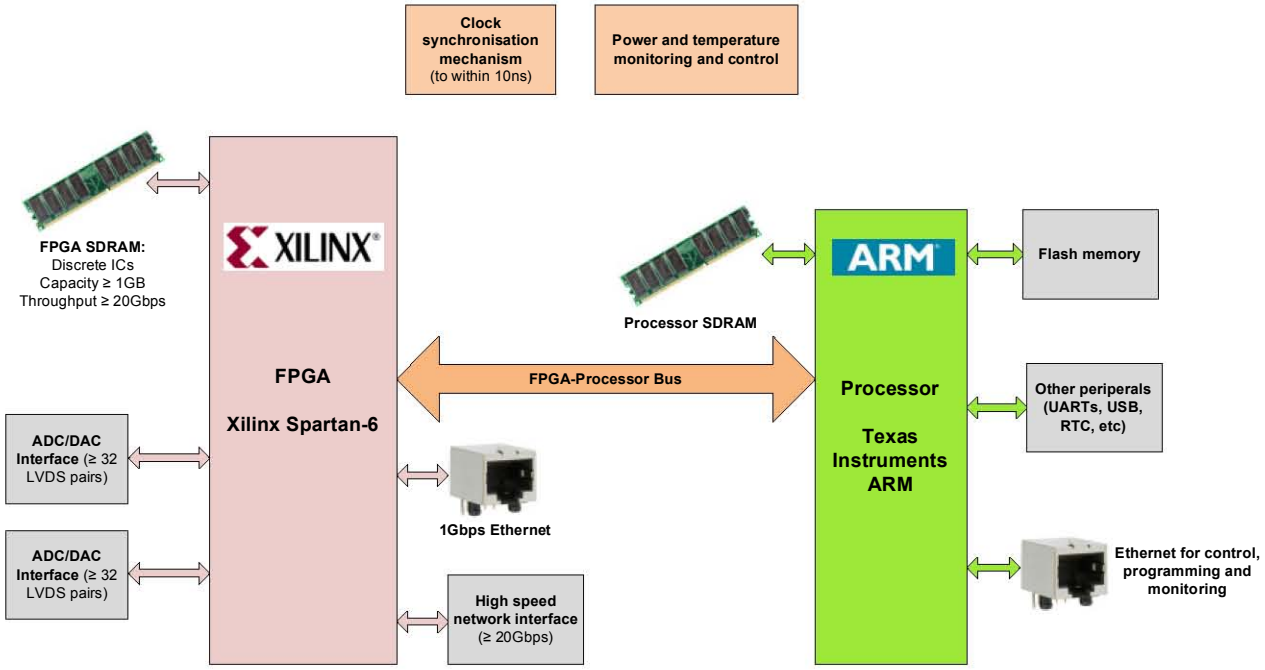


Figure 3.1: Rhino high-level block diagram, indicating performance requirements

- **Architecture:** Rhino contains an FPGA and a processor, connected via a parallel bus. The processor is used for coordinating the flow of data in and out of the FPGA, as well as for programming the FPGA [CS1, HG3].

- **FPGA:** A Xilinx Spartan-6 FPGA, with the maximum amount of logic resources available. The FPGA has sufficient I/O resources to support 64 LVDS pairs (connected to ADCs and DACs), each running at 400Mbps [CS2, RQ1, RQ5, HG1].

- **Processor:** Any ARM processor (a physical processor, not a softcore on the FPGA), manufactured by Texas Instruments, that is well supported by Linux [CS2, HG2].

- **Memory:** 1GB of SDRAM connected for the FPGA, with minimum throughput of 20Gbps. The SDRAM takes the form of discrete components, and not a DIMM. The processor has sufficient SDRAM to run Linux [CS3, RQ2, HG8].

- **Flash memory**: The processor has flash memory on which the operating system, custom firmware for the processor and FPGA configuration files can be stored.

- **Networking:** The FPGA has two types of network interfaces: a high-speed network connection with a throughput of 20Gbps, as well as a slower 1Gbps Ethernet link. The processor also has an Ethernet connection that allows remote control and monitoring of the board, and programming of the FPGA, but it may run at any speed [CS4, CS6, RQ3, HG9, HG12].

- **I/O Interface:** Two separate 32-bit LVDS interfaces for ADCs and DACs, using an industry-standard connector [CS4, RQ5, HG7].

- **Clock Synchronisation:** The Rhino board contains a mechanism to synchronise the clocks on different boards to within 10ns of each other [RQ4, HG11].

- **Power and Temperature Monitoring**: All the temperatures, voltages and currents on Rhino are monitored using a dedicated monitoring subsystem, and the results are reported to the processor. If any of the readings exceed the safety limits, either the offending component will be automatically switched off, or the board will shut down [HG5].

- **Cost:** Rhino must cost less than $1800 to manufacture [RQ7, HG6].

3.2 SOFTWARE ARCHITECTURE FOR RHINO

Although the software does not form part of this thesis, the proposed software architecture does in fact have some influence on the design of the board. It is therefore described briefly. Note that the same referencing symbols that were used in the previous section are used again here.

The processor will be running BORPH, a Linux variant with FPGA support [CS5]. This will allow users to remotely communicate with the board over the processor's Ethernet connection [SG5]. The bootloader and Linux kernel will be stored on the flash memory on the board, while the file system will most probably be hosted on a network server and transferred to the board using the NFS (Network Filesystem) protocol.

As has already been explained, BORPH allows users to program the FPGA with a given design/configuration, and run it as a software process within Linux. This simplifies both the FPGA configuration/programming process, and the communication between the FPGA and the processor [HG3]. The consequences of this is that:

a.) No provision needs to be made for dedicated JTAG interfaces for programming the FPGA.
b.) The processor must interface to the relevant programming pins on the FPGA, so that the FPGA programming process can be automated by BORPH.
c.) A parallel bus between the FPGA and processor is required for inter-device communication.

The exact architecture of the toolflow has yet to be determined. However, whatever form it may take, it will allow the user to focus on the signal processing aspects of the target applications, rather than on the technical workings of Rhino. This should be achieved by providing the user with number of library blocks (both signal processing and hardware I/O blocks) that can be simply connected together in a text-based environment, such as MyHDL[1] (a Python to VHDL/Verilog compiler) [RQ6, SG1-4].

For further signal processing on a host computer, GNU Radio will be used. A GNU Radio driver will be developed that allows Rhino to look like just another software-defined radio peripheral. Besides providing a simple framework for doing true software-based signal processing, GNU Radio will also allow the captured data to be visualised in real-time on a computer monitor.

3.3 SELECTION OF MAJOR COMPONENTS

Now that the architecture of Rhino has been determined, the main components must be selected. This needs to be done before the detailed hardware block diagram can be determined, as the choice of components can impact the detailed architecture. In most cases, the selection of major components was done by performing a trade-off analysis of the different options. The selection of the Spartan-6 FPGA, the ARM processor, the type of ADC/DAC interface connector, the high-speed network interface, and the type of flash memory and power supplies used is explained below.

[1] http://www.myhdl.org

3.3.1 Selection of Spartan-6 FPGA

There are number of devices in Spartan-6 family, each with varying amounts of logic resources, block RAM, I/O pins and peripheral support. The Spartan-6 devices can be divided into two main groups: LX and LXT. LX are general logic devices, while LXT are general logic devices with GTP transceivers. Since these transceivers are required for high speed networking (anything above 1Gbps), only the LXT devices within the Spartan-6 family will be considered.

The customer specification stated that the FPGA with the largest amount of logic must be used for Rhino. Therefore, the chosen FPGA will be the largest LXT device, the XC6SLX150T. The characteristics for this device are shown in Table 3.1. This table was taken from the Spartan-6 Family Overview datasheet [32].

Table 3.1: Characteristic of the XC6SLX150T Spartan-6 FPGA

Logic Cells	**CLB Slices**	**CLB Flip-Flops**	**CLB LUTs**	**DSP48A1 Slices**	**18kb BRAM Blocks**	**Max MCBs**	**Total I/O Banks**
147 443	23 038	184 304	92 152	180	268	4	6

These characteristics are explained below:

- *Number of logic cells:* The most basic programmable element in an FPGA. A logic cell contains a single look-up table (LUT) and a single flip-flop [33]. This metric is rarely used anymore.
- *Number of CLB slices:* A CLB is a configuration logic block, the preferred metric for measuring logic resources. Each CLB is made up of two slices. A single CLB slice contains four look-up tables (LUTs) and eight flip-flops, as this represents the physical structure within the FPGA die [32].
- *Number of DSP48A1 slices:* These are the digital signal processing (DSP) resources within the FPGA. Each DSP slice contains an 18x18 multiplier, an adder and an accumulator.
- *Number of BRAM blocks:* Block RAM is the fast RAM built into the FPGA itself. Each block is 18kb in size.
- *Maximum number of MCBs:* An MCB (memory controller block) is specialised circuitry on the FPGA, used for interfacing to external SDRAM ICs. Table 3.1 gives the *maximum* number of MCBs, as the MCB pins can also be used as normal I/O interfaces.
- *Number of GTP transceivers:* A GTP transceiver is a serial transmitter and receiver pair that is capable of operating at frequencies up to 3.125Gbps. GTP transceivers are typically used for high speed network interfaces. The number of GTP transceivers is not given in Table 3.1, as the actual number of transceivers present depends on the physical package used.
- *Number of I/O Banks:* An I/O bank is a group of user I/O pins, powered off a common voltage rail. Typically, the larger the number of I/O banks, the greater the total number of user I/O pins.

There are also a number of different physical packages for Spartan-6 FPGAs, and each in varying numbers of pins. The package determines the number of I/O pins and the number of transceivers (a bigger package has more I/O and more transceivers). The different package options are therefore shown in Table 3.2. Note that CSG*xxx* describes a 0.8mm pitch BGA (ball grid array) package with *xxx* balls, while FGG*yyy* describes a 1mm pitch BGA package with *yyy* number of balls.

Table 3.2: Spartan-6 XC6SLX150T Package Options

Package	**CSG484**		**FGG484**		**FGG676**		**FGG900**	
Device	**GTPs**	**User I/O**	**GTPs**	**User I/O**	**GTPs**	**User I/O**	**GTPs**	**User I/O**
XC6SLX150T	4	296	4	296	8	396	8	540

Before an FPGA package can be selected, the number of I/O pins required to interface to each FPGA subsystem peripheral must be determined. This is shown in Table 3.3 below. Although the calculations are not shown here, these values were determined by looking at similar FPGA boards (such as the boards reviewed in Chapter 2) and counting the number of I/O pins required for each type of peripheral.

Table 3.3: FPGA I/O Requirements for Each Peripheral

Peripheral	**Number of Pins**	**Notes**
SDRAM 0	49	18 address lines, 16 data lines, 15 control lines
SDRAM 1	49	18 address lines, 16 data lines, 15 control lines
ADC Interface 0	76	32 LVDS data pairs, 2 LVDS clock pairs, 8 control lines
ADC Interface 1	76	32 LVDS data pairs, 2 LVDS clock pairs, 8 control lines
Processor-FPGA Bus	45	10 address lines, 16 data lines, 19 control lines
1Gbps Ethernet PHY	28	16 data lines, 12 control lines
GPIO, LEDs, clocks	32	16 GPIO, 8 LEDs, 8 clocks
Total number of pins	355	

Although the Spartan-6 XC6SLX150T supports four memory controller blocks (and hence four SDRAM chips), only two have been included in the calculations above. The reason lies in the fact that each MCB is in a different I/O bank, and operates at the 1.5V SSTL signalling standard. Therefore, if all four MCBs were used, four of the six I/O banks would operate at 1.5V, which is unusable voltage for all other peripherals. Furthermore, the ADC interfaces can only be used on banks 0 and 2 (these are the only banks that support LVDS) and almost completely occupy these banks in both the FGG676 and FGG900 packages, leaving very few pins available for other peripherals. Therefore, only two MCBs are used.

From the table above, it is clear that both the FGG676 and FGG900 packages meet the I/O requirements. However, in order to keep manufacturing costs down (it is expensive and difficult to place large BGAs) and simplify the routing of the board, the FGG676 package shall be used. Therefore, the final part number of the Spartan-6 FPGA to be used is XC6SLX150T-4FGG676C.

3.3.2 Processor Selection

While the processor may not do any computationally-intensive processing, it is just as critical as the FPGA. It is used for programming the FPGA, monitoring the status of the board, and providing a simple interface to the FPGA logic. The requirements for the processor were given in Chapter 1, and are repeated here for clarity:

1. The processor must be an ARM device produced by Texas Instruments.
2. The processor must use external SDRAM.
3. The processor must have the necessary hardware to provide both an Ethernet connection (so that the user can talk to the processor) and an interface to program the FPGA.
4. The processor must be well supported by Linux.

Texas Instruments (TI) manufactures three types of processors with ARM cores: microprocessors, microcontrollers and DSPs. Since microcontrollers have on-chip RAM and program memory, their memory is generally very limited, while microprocessors use external memory components, allowing them to be supplied with larger volumes of memory. Microprocessors also typically run at higher clock speeds than microcontrollers. DSPs (such as the popular OMAP family), on the other hand, can use either microcontroller or microprocessor cores, but have specialised processing hardware to perform DSP operations very quickly.

It should be clear that TI's microprocessors are the only viable option for Rhino, as microcontrollers have insufficient memory and performance, while the DSPs contain hardware that is of no use in this application, and hence not financially economical. Table 3.4 (taken from the TI ARM Selection Guide [34]) shows the different families of ARM *microprocessors* that TI produces, and the typical applications of each.

Table 3.4: Texas Instruments ARM Microprocessors

Family	**Stellaris**	**TMS570**	**Sitara**	
ARM Core	Cortex-M3	Cortex-R4	Cortex-A8	ARM9
Description	Designed for low cost, low power applications, such as wireless sensor networks. No mainstream Linux support (only FreeRTOS).	Designed for safety critical systems, such as motor control. Therefore includes a number of safety features, such as dual CPUs and built-in self tests.	The Sitara processors are high-performance, low power devices, with clock speeds up to a substantial 1.5GHz. The devices have extensive peripheral integration and are fully supported by Linux.	

From this table it is clear that the Stellaris family is not suitable, due to the lack of Linux support, nor is the TMS570 family, since it sacrifices performance for reliability. Even though the Sitara family of devices are more expensive than the other two families, their higher performance, extensive integrated peripherals and excellent Linux support make the Sitara devices the obvious choice for Rhino.

Only certain devices within the Sitara family have Ethernet support. Since this is obviously a critical interface for the processor (as it allows remote access to the processor, and allows FPGA configuration files to be copied to the board), only devices within the Sitara family with Ethernet support were considered for Rhino. The list of possible devices is shown in Table 3.5 [2]. Note that at the time of the project commencement, not all these devices were available (indicated with a "No" in the "Available?" column). However, these devices which have become available after the processor was selected have still been included in the table (albeit greyed out) to show how rapidly technology advances within just 12 months.

Most of the entries in Table 3.5 should be self-explanatory. However, the "External Memory Interface" column may require some explanation. The external memory interface is used for interfacing to the flash memory and the FPGA. Sitara microprocessors offer two different types of memory interfaces: EMIF (External Memory Interface) and GPMC (General-purpose Memory Controller). EMIF has limited configurability, and supports only NAND flash, NOR flash and SRAM devices. GPMC, on the other hand, is fully configurable and can interface to just about any type of memory or processing device. Also note that the values in the "MMC/SD" columns in the table above indicate the number of SD card interfaces on the relevant device, while the values in the "USB" column indicate the number of supported USB host ports.

[2] Obtained by doing a parametric part search on the TI website (www.ti.com) on 22 February 2011.

Table 3.5: Texas Instruments Sitara Microprocessor Family (only devices with Ethernet support)

Device Name	Available?	Core Frequency	SDRAM Support	External Memory Interface	Video Output	Max Ethernet Speed	MMC/SD	USB
AM3894	No	1.5GHz	2x 32-bit (up to DDR3-1600)	16-bit GPMC	Yes	1000Mbps	1	2
AM3892	No	1.5GHz	2x 32-bit (up to DDR3-1600)	16-bit GPMC	Yes	1000Mbps	1	2
AM3517	Yes	600MHz	1x 32-bit (up to DDR2-333)	16-bit GPMC	Yes	100Mbps	3	3
AM3505	Yes	600MHz	1x 32-bit (up to DDR2-333)	16-bit GPMC	Yes	100Mbps	3	3
AM1808	Yes	456MHz	1x 16-bit (up to DDR2-333)	16-bit EMIF	No	100Mbps	4	2
AM1707	Yes	456MHz	1x 32-bit (up to 100MHz non-DDR)	16-bit EMIF	No	100Mbps	1	2
AM1705	Yes	456MHz	1x 32-bit (up to 100MHz non-DDR)	16-bit EMIF	No	100Mbps	1	1

From Table 3.5, it is obvious that the AM1707 and AM1705 devices are not suitable, due to their slow SDRAM interfaces. The AM35xx devices are preferable to the AM1808, as the GPMC interface is far more configurable than the EMIF interface, and hence there is a lower risk of compatibility problems when interfacing to the FPGA. The ARM Cortex-A8 (used by the AM35xx devices) is also much newer and provides better performance than the older ARM926E core (as used by the AM1808). The Cortex-A8 can run at 2000 DMIPS (Dhrystone Millions of Instructions Per Second), while the ARM926E can only provide 200 DMIPS [35] [36].

The only difference between the AM3517 and the AM3505 is that the AM3517 has an integrated 3D graphics accelerator, while the AM3505 does not. While not a critical feature, it is possible that Rhino may be used for video processing applications where the processor needs to do the final post-processing before outputting the video to a display. Since the AM3517 costs only $4 more than the AM3505[3], the minimal extra cost is worth the increased functionality. The AM3517 shall therefore be the chosen processor for Rhino. However, since the AM3505 and AM3517 are pin compatible, it is always possible to switch to the AM3505 in the future.

The full feature set for the AM3517, which was taken from the AM3517 datasheet [37], is given in Table 3.6.

3.3.3 Selection of I/O Connector for ADC and DAC Cards

Rhino includes two I/O connectors for connecting to ADC and DAC cards. The type of connector chosen affects the type of ADC/DAC cards that are compatible with Rhino, and is therefore an important decision. The two options for such a connector are Z-DOK+ and FMC. Both of these connectors are described below, and their relative advantages and disadvantages are compared.

ZDOK

The Z-DOK+ connector is a high-speed connector produced by Tyco Electronics, supporting 40 differential pairs and 6 utility contacts for power rails [38]. It is used on all CASPER boards (such as ROACH) and therefore compatible with CASPER ADC/DAC boards. The Z-DOK+ interface, as defined by CASPER, is shown in Table 3.7.

The two differential clocks are outputs on the ADC/DAC card and inputs on the FPGA. Some of the differential data pairs can be used for control, but there is no standard stating which pairs should be used for this.

[3] As listed on www.digikey.com on 22 February 2011, for single quantities

Table 3.6: Feature Set of the AM3517

CPU	ARM Cortex-A8, with NEON SIMD and vector floating point co-processors
CPU Frequency	600MHz
On-Chip L1 Cache	32 KB (ARM Cortex-A8)
On-Chip L2 Cache	256 KB (ARM Cortex-A8)
SDRAM Support	32-bit SDRAM Controller (SDRC), with support for LPDDR and DDR2 (up to DDR2-333), with 1GB address space
External Memory Interface	16-bit data, 26-bit address multiplexed (configurable) GPMC bus, with glueless support for NOR flash, NAND flash, OneNAND, SRAM, and easily configured to interface to an FPGA
DMA	32-Bit Channel SDMA
Video Ports	1 dedicated output,1 dedicated input, both capable of HD resolutions
Graphics Accelerator	POWERVR SGX graphics accelerator
CAN	1 interface
EMAC (Ethernet MAC)	10/100Mbps Ethernet MAC
MMC/SD	3 interfaces
McBSP (Multi-channel Buffered Serial Port)	5 ports
I2C	3 interfaces
McSPI (Multi-channel Serial Port Interface)	4 interfaces
UART	4 ports
USB	3 host interfaces, and 1 USB-OTG interface
Timers	12 32-Bit general-purpose timers
Supply Voltage	1.2V core, 1.8V or 3.3V I/O

Table 3.7: The Z-DOK+ Connector Pins

Pins	Voltage
38 differential data pairs	2.5V LVDS
2 differential clock pairs	2.5V LVDS
5V 1.5A supply	-
3.3V 1.5A supply	-
2.5V 1.5A supply	-
1.8V 1.5A supply	-

FMC

FMC, or FPGA Mezzanine Card, is a ANSI/VITA standard (VITA 57.1) developed by Xilinx, Curtiss Wright, Samtec and a number of other electronics companies, for connecting peripheral cards to FPGA boards. It defines the mechanical specifications for I/O cards (known as "mezzanine cards") that plug onto FPGA boards (known as "carrier cards") [39]. In the case of Rhino, the mezzanine cards are ADC or DAC cards, while the carrier card is the Rhino PCB. The FMC standard defines not only the mechanics of the mezzanine card, but also the electrical interface and mechanical dimensions of the connectors that interface the two boards. These connectors, manufactured by Samtec, are known as the FMC connectors.

There are two types of FMC connectors: low pin-count connectors (LPC connectors) and high pin-count con-

nectors (HPC connectors). LPC connectors contain 160 pins, while HPC connectors have 400 pins. However, both connectors have identical form factors and are mechanically compatible; LPC connectors merely do not have all the pins populated. The pins on the LPC connector are summarised in Table 3.8.

Table 3.8: The FMC LPC Connector Pins

Pins	**Voltage**
34 differential data pairs	2.5V LVDS
2 differential clock pairs	2.5V LVDS
5 JTAG lines	3.3V LVTTL
SCL and SDA lines for I2C	3.3V LVTTL
Power good signal	3.3V LVTTL
3 mezzanine-to-carrier signals that indicate if card is present, and address of EEPROM chip	3.3V LVTTL
3.3V 3A supply	-
12V 1A supply	-
3.3V 20mA auxiliary supply	-
Adjustable 2A supply (for differential data pairs, usually 2.5V)	-

The FMC specification allows for some, or all, of the 34 differential data pairs to be used as single-ended LVTTL signals. Although not shown in Table 3.8, the FMC LPC specification also defines two high-speed serial data pairs (up to 10Gbps). Since all the GTP transceivers on the FPGA are used for the high-speed network interface, there are no transceivers available for these two serial data pairs, and they are thus left disconnected.

The FMC HPC connector contains exactly the same pins as the LPC connector, in the same positions. The specification just defines an additional 46 differential data pairs, an additional 2 differential clocks and an additional 18 multi-gigabit high-speed serial data pairs.

Comparison of FMC and ZDOK

The FMC connector is selected over the Z-DOK+ connector as the I/O connector of choice for Rhino, for the following reasons:

- FMC is a well-defined ANSI/VITA specification, that defines the function of each and every pin on the connector, while there is little standardisation of the Z-DOK+ interface (not even all the CASPER ADC cards are pin compatible with each other).
- Since FMC is an industry-recognised standard, there are numerous FMC ADC and DAC cards available from third-party manufacturers, such as Curtiss Wright and 4DSP [27]. Z-DOK+ ADC and DAC cards are however only available from the CASPER community.
- The FMC specification has been drawn up by experts in the field, and as a result it provides guidelines for dealing with most thermal and isolation issues [39].

However, it would still be beneficial for Rhino to be compatible with the Z-DOK+ connector too, as this would allow Rhino to use the various CASPER ADC and DAC cards. This can be achieved by building an FMC-to-Z-DOK+ adaptor board. This would essentially be an FMC card with no components other than an FMC connector (which plugs onto Rhino) and a Z-DOK+ connector (which connects to the CASPER ADC/DAC card). The one problem is that the Z-DOK+ interface defines 38 differential data pairs, while the FMC LPC

connector only provides 34 data pairs. To solve this, Rhino actually uses an FMC HPC connector to provide the extra data pairs. Rhino uses the LPC pins on the HPC connector, and an additional 4 differential pairs (8 pins) from the HPC part of the connector. The 8 pins on the HPC part of the connector are only used for the Z-DOK+ adaptor board; in all other cases Rhino uses only the LPC pins on the FMC connector.

3.3.4 Selection of High-speed Network Interface

There are two types of connectors that can be used for the high speed (20Gbps) network interface on the FPGA: CX4 and SFP+. In both cases, four XAUI (10Gbps Attachment Unit Interface) lanes, each running at 3.125Gbps, connect the FPGA to the network connector. In a CX4 connector, these four XAUI lines are connected directly to separate copper wires within the cable. SFP+ (Small Form-factor Pluggable Plus), a newer standard, instead uses a PHY on the board to convert the four XAUI lanes to a single 10Gbps lane, which is wired to the connector. Furthermore, the connector on the cable itself usually contains a transceiver module that converts the single 10Gbps lane into a single copper or optical fibre line. Since SFP+ requires a PHY and a transceiver, while CX4 is just plain copper wires, the SFP+ option is obviously more expensive.

While it is tempting to use the latest technologies, in this case the increased cost does not outweigh the benefit. Furthermore, since Rhino is targeting smaller research groups, many of these groups have older CX4-based network switches, and due to the price of 10Gbps Ethernet hardware, do not have the funds to purchase new SFP+ switches. Rhino therefore uses CX4 connectors for the high-speed networking.

Note that a single CX4 connector supports speeds up to 10Gbps only. Since the required network bandwidth for Rhino is 20Gbps (see Table 1.2), two CX4 connectors are required.

3.3.5 Selection of SDRAM for the FPGA

The detailed specification at the beginning of this chapter states that Rhino should have 1GB of SDRAM attached to the FPGA, with a maximum throughput greater than 20Gbps. The specification also states that discrete SDRAM ICs must be used, and not DIMMs, as the memory controller blocks (MCBs) on the Spartan-6 FPGA support only discrete memory components.

Since only two of the four MCBs on the XC6SLX150T Spartan-6 FPGA can be used in this design (refer to Section 3.3.1 for the explanation of why the other two MCBs cannot be used), and each MCB only supports a single SDRAM IC, only two SDRAM ICs can be connected to the FPGA. The MCBs each support data bus widths up to 16-bits and memory clock speeds up to 400MHz (DDR3-800). The SDRAM ICs used on Rhino shall therefore be 16-bit DDR3 devices, to provide maximum memory bandwidth.

Micron SDRAM ICs were selected for use on Rhino due to the availability of low quantities of individual ICs [4] (many other SDRAM manufacturers only sell ICs in quantities of tens of thousands) and their good reputation for quality in the memory industry. The largest commercially available DDR3 SDRAM components from Micron, with 16-bit data buses, have a density of 2Gb [5]. If two of these devices are connected to the FPGA, the total density (size) becomes 4Gb, or 512MB.

Unfortunately, 512MB is only half the memory capacity that was specified for the FPGA at the beginning of this chapter. However, there are presently no larger devices with 16-bit data buses available from any manufacturer, nor can the other two MCBs on the Spartan-6 be used. Therefore, for now at least, Rhino is unable to meet the requirement for 1GB of FPGA SDRAM. Fortunately, Micron is currently providing

[4] Micron SDRAM ICs can be bought online from www.micron.com, www.digikey.com and www.arrownac.com

[5] This was determined by performing a parametric component search on the Micron website (www.micron.com) on 24 February 2011

samples of 4Gb DDR3 SDRAM ICs with 16-bit data buses. Hopefully these devices will enter production soon and can therefore be used on future builds of Rhino, providing the FPGA with the full 1GB of RAM. To make provision for this, Rhino will be compatible with both the 2Gb and 4Gb SDRAM components, allowing either component to be populated once the 4Gb devices become available.

The maximum throughput of the two DDR3-800 SDRAM ICs (they are actually DDR3-1066 devices that are run slightly slower, as DDR3-800 is the highest speed that the Spartan-6 supports) can be calculated as follows:

Throughput = (Memory clock) * DDR * 16 bits * (2 ICs)
= 400MHz * 2 * 16 * 2
= 25.6Gbps, which meets the Rhino FPGA memory specification of $\geq$ 20Gbps

3.3.6 Selection of Flash Memory Technology for the Processor

There exist two main flash memory technologies: NAND flash and NOR flash. NAND memory is typically used for multimedia storage, as it provides fast sequential access, but slow random access. NOR memory is typically used code storage, due to its fast random access times.

Therefore, it may seem counter-intuitive that NAND memory shall be used for code storage for the AM3517 on Rhino. The main reason for choosing NAND over NOR is that the AM3517 uses a common address/data multiplexed bus to communicate with the memory device. Since most NOR memory devices use separate buses for address and data signals, they are not compatible with the AM3517.

Spansion does however make a NOR memory device with a common address/data multiplexed bus (OR-NAND) [40]. The problem with these devices is that their maximum capacity is 64MB. It therefore appears that NAND memory would be a better option, as the larger capacity allows bigger Linux distributions and more FPGA bit files to be stored in the memory. The AM3517 also has considerable cache memory which should hide most of the NAND access delays.

3.3.7 Comparison of Analogue and Digital Power Supplies

When building a complex circuit board, it is crucial that the power supply system is carefully designed. Rhino is no exception to this rule, especially since the FPGA requires five different supply voltages and the processor three supply voltages.

In order to maximise power efficiency, switch-mode power supplies are used. There exist two main types of switch-mode power supply modules: digital supplies and analogue supplies. Analogue supplies are the most common and use an analogue feedback loop to regulate the switching of the supply. Digital supplies, however, use a microprocessor with an ADC to monitor the supply output and regulate the switching accordingly. This is a much newer technology and allows greater control over the supply switching.

Table 3.9 shows a comparison of these two different power supply technologies, with respect to the Rhino hardware design. In the purely analogue design, four analogue switch-mode power supplies and a single low-dropout (LDO) linear regulator are used to power the FPGA, while a 3-channel analogue switch-mode supply (TPS65023) powers the three processor voltage rails. A separate INA219 (voltage and current monitor IC) is used to monitor the power on each supply rail.

The digital power supply design uses four digital switch-mode power supplies and a single LDO supply to power the FPGA. These four digital power supplies are controlled and monitored by a single UCD9240

controller IC. The same 3-channel analogue switch-mode supply that was used in the analogue design to power the processor, is used again in this design. These analogue supply rails are once again monitored using INA219 power monitors.

As one can see from the table, the efficiency is almost identical for both the analogue and digital supplies (in the case of Rhino's power requirements). The main advantage of the digital supply option is actually the component count: a single UCD9240 power supply controller is able to monitor and control four independent supplies (both voltage and current monitoring), without the need for any external instrumentation amplifiers or ADCs. It is also able to control two fans, based on temperature readings. This reduced component count leads to reduced costs.

The advantage of the analogue approach is simplicity: there is no digital controller IC that requires programming. Routing is also simpler, as each voltage/current monitor can be placed directly at each current-sensing resistor. Furthermore, there are many more analogue supplies available on the market than digital ones.

The trade-off is therefore between cost and simplicity. As a result, it was decided that analogue power supply modules would be used, as greater simplicity means lower risk of design errors. However, as digital supplies become more abundant and easier to use, future revisions of the board may use digital supplies instead.

3.4 Updated High-level Block Diagram

Now that all the major components have been selected for Rhino, the high-level block diagram can be updated to reflect these decisions. This updated diagram is shown in Figure 3.2. Note that even though the SDRAM is represented as DIMMs in the diagram, these are actually discrete memory ICs.

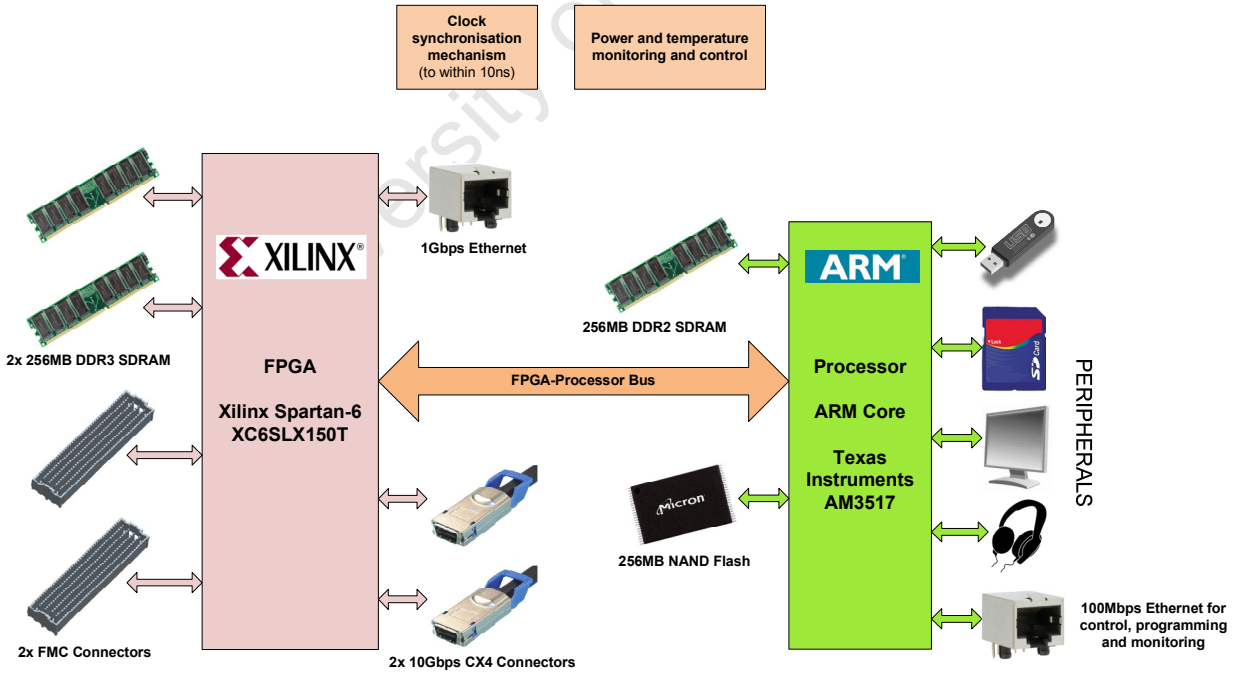


Figure 3.2: Updated Rhino high-level block diagram

This updated block diagram (and hence more detailed architectural description of Rhino) meets all of the specifications that were drawn up at the beginning of this chapter (see Section 3.1), with the exception of the SDRAM capacity for the FPGA (1GB). As has been already explained, there are no DDR3 SDRAM

components currently available that are big enough to meet this requirement. However, these components should become available soon and be incorporated into future revisions of Rhino.

With the main components selected, and the detailed architecture of Rhino determined in this chapter, the block diagram in Figure 3.2 needs to be fleshed out to a greater level of detail. This shall be done in the next chapter, after which the detailed design will be transferred to schematics.

Table 3.9: Comparison of Analogue and Digital Power Supplies for Rhino

	Analogue		**Mixed Digital and Analogue**	
Component Count	5x Analogue Supply Modules 1x Processor Supply Device 2x FPGA LDO Supply 9x INA219 Power Monitors 1x 4-channel Temperature Monitor 1x 2-channel Fan Controller **Total:** 8x Power Supplies 11x Monitoring ICs		4x Digital Supply Modules 1x Analogue Supply Modules 1x Analogue Processor Supply Device 2x FPGA LDO Supply 1x UCD9240 5x INA219 Power Monitors 1x 4-channel Temperature Monitor **Total:** 4x Digital Power Supplies 4x Analogue Power Supplies 7x Monitoring ICs	
Cost	Analogue Supplies (3x PTH08T230W, 2x PTH08T240W)	$90.60	Digital Supplies (2x PTD08A006, 2x PTD08A010)	$67.76
	Processor Supply (TPS65023)	$8.75	Analogue Supplies(1x PTH08T230W)	$16.52
	2x FPGA LDO Supply (TPS51200)	$6.30	Processor Supply (TPS65023)	$8.75
	9x INA219	$49.95	2x FPGA LDO Supply (TPS51200)	$6.30
	Temp. Monitor (MAX1668)	$9.86	1x UCD9240	$11.66
	Fan Controllers (TC654)	$2.00	5x INA219	$27.75
			Temp. Monitor (MAX1668)	$9.86
	Total	**$167.46**	**Total**	**$148.60**
Efficiency (at full load)	89.9%		90.3%	
Complexity	Simple enable line to switch each supply on/off. Easy to route, as monitor can be placed directly at supply		PMBus commands used to switch each supply on/off. 8 lines must be routed between each supply and the UCD9240	
Factory Setup	None		UCD9240 needs to be programmed via I2C interface before board can be used.	
Available Supply Sizes	1A, 3A, 4A, 6A, 10A, 16A, 30A, 50A, amongst others.		6A, 10A, 15A, 20A only	

CHAPTER 4

SCHEMATIC-LEVEL HARDWARE DESIGN OF RHINO

The previous chapter described the hardware architecture for Rhino. This architectural description forms the basis for the sub-system diagram, which is developed in this chapter. This diagram details all the different sub-systems that together form Rhino. Only once this detailed sub-system description is complete, can the design can be transferred to schematics.

Each of the schematic pages can be found in Appendix B. Since browsing through pages of schematics can be tedious, a discussion of the major components on each schematic page, and relevant design decisions that were made, has been included in the second half of this chapter.

4.1 DETAILED HARDWARE SUB-SYSTEM DESCRIPTION FOR RHINO

The final hardware architecture diagram (see Figure 3.2) is now expanded to form a detailed sub-system diagram. This diagram, shown in Figure 4.1, shows all the different sub-systems that together form Rhino. These sub-systems can be loosely grouped into four categories: the FPGA and its peripherals, the processor and its peripherals, the debugging sub-system, and the power generation and monitoring systems. The FPGA sub-systems are on the left-hand side of the diagram, the processor sub-systems on the right-hand side of the diagram, the debugging sub-systems in the top-right corner, and the power systems in the rest of the upper half of the diagram.

4.1.1 FPGA and its Peripherals

The main component of Rhino, the Spartan-6 XC6SLX150T FPGA, is aptly represented by the largest block in the sub-system diagram. Connected to the FPGA are two FMC connectors, each with 34 LVDS data pairs, 2 LVDS clock pairs and a couple control wires (including I^2C). The FPGA network support takes the form of two CX4 connectors (each with four XAUI lanes, providing a total bandwidth of 10Gbps per connector) and one RJ45 connector for the 1Gbps Ethernet. The 1Gbps link does however require a PHY to convert the parallel GMII (Gigabit Multimedia Independent Interface) bus into the necessary serial signals that can be sent down a CAT6 Ethernet cable.

Two 2Gb DDR3 SDRAM ICs provide large capacity memory for FPGA applications that need to buffer large volumes of data. These two ICs each have a 16-bit data bus, running at 400MHz DDR (i.e. 800Mbps per data line). This results in a total theoretical memory throughput 25.6Gbps.

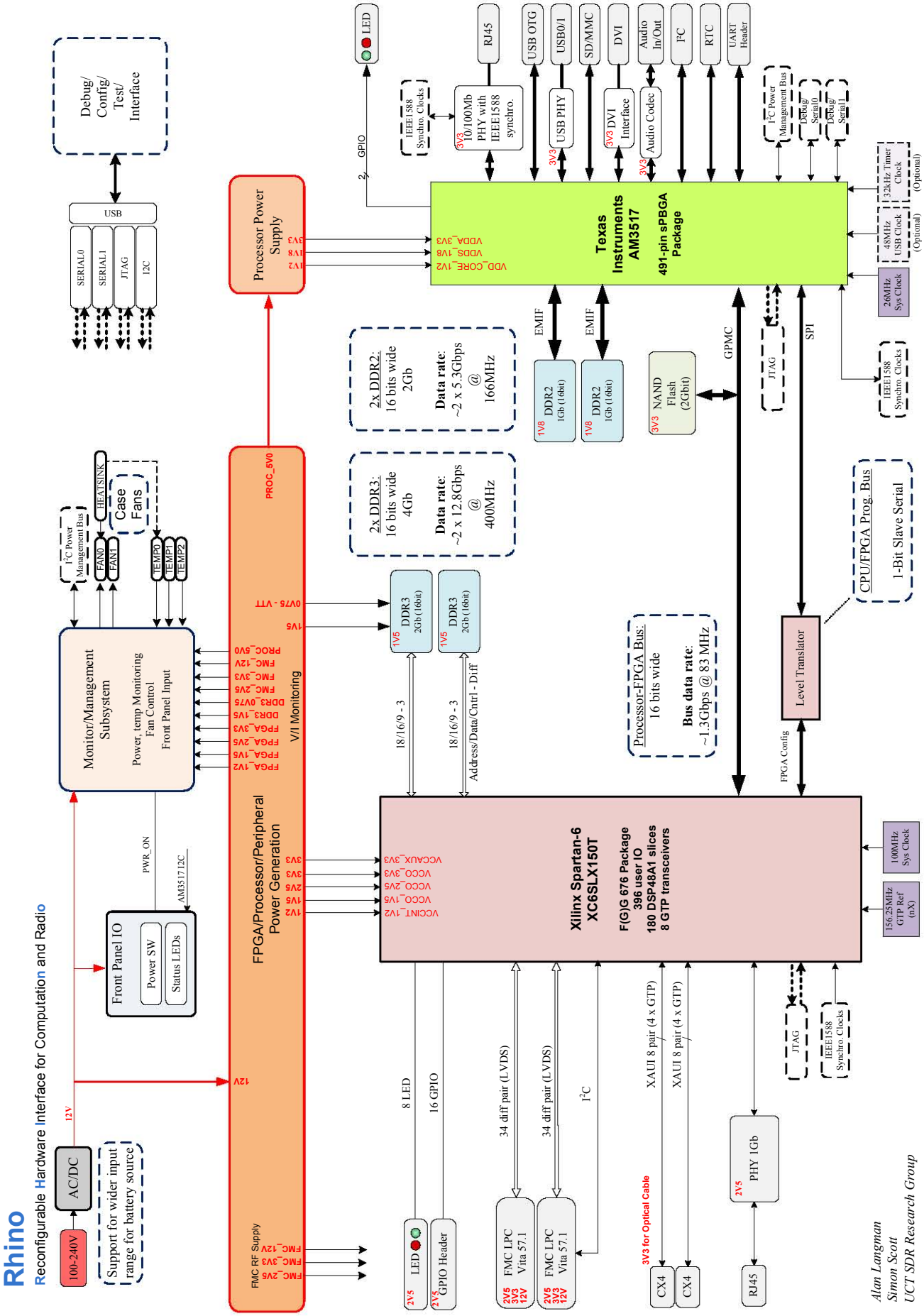


Figure 4.1: Sub-system diagram of Rhino

The FPGA is connected to the processor via a parallel bus (which connects to the GPMC interface on the processor side). This parallel bus allows BORPH to read and write registers on the FPGA [12]. This bus is not however used for programming/configuring the FPGA. That instead takes place via the Serial Configuration Interface on the FPGA, which is driven by a SPI port on the processor. The FPGA can also be programmed via a JTAG header, or via the board-wide JTAG bus (which shall be described later).

For debugging purposes, the FPGA is also supplied with 8 LEDs and a 16-bit GPIO header. Clocking takes the form of a 100MHz system clock and a 156.25MHz GTP reference clock. Furthermore, the clock lines on the FMC connector can also be used to clock the FPGA.

4.1.2 **Processor and its Peripherals**

The AM3517 processor is depicted by the large green block on the right-hand side of the diagram. Its two 1Gb DDR2 SDRAM ICs are wired in parallel and connected to the 32-bit EMIF (External Memory Interface). Each of these ICs have a 16-bit interface, running at 166MHz DDR (i.e. 333Mbps per line).

Both the NAND flash memory and the FPGA are connected to the GPMC (General Purpose Memory Controller) interface on the processor. This interface has 16 data lines (which can be multiplexed to act as address lines too), 10 dedicated address lines and a large number of control lines. This bus can be configured to support a number of different bus protocols, allowing glueless interfacing to both the NAND flash and the FPGA. As was mentioned previously, the FPGA is programmed using one of the SPI interfaces on the processor, via a 3.3V to 2.5V level translator.

The processor uses a DP83640 Ethernet PHY to provide 100Mbps Ethernet access. A PHY is essentially a transceiver that converts the parallel RMII (Reduced Media Independent Interface) signals from the Ethernet MAC (on the processor) to serial signals that can be transmitted down an Ethernet cable. One advantage of using this particular PHY is that it supports the IEEE1588 Precision Time Protocol (PTP). PTP allows the clocks on a number of devices (i.e. Rhinos), connected via a local network, to be synchronised to within 10ns of each other. This PHY therefore satisfies both the processor Ethernet support and the clock synchronisation requirements for Rhino.

Other major processor peripherals include the SD card (Secure Digital card) interface, the two USB host ports and the USB On-the-Go port. The processor is able to boot from any one of these peripherals, as well as the previously mentioned flash memory and Ethernet interface, without requiring any special firmware. Just like the Ethernet interface, the two USB host ports also each require a transceiver (or PHY).

The processor also has a video output interface (using a High Definition Multimedia Interface, or HDMI, connector); three audio connectors for line-in, line-out and headphones; a real-time clock chip; an RS-232 connector for communicating with devices like GPRS modems; and two LEDs for debugging. While none of these peripherals are critical, and Rhino would still meet all the requirements identified in Chapter 1 if none of these peripherals were used, they have been added to improve the usability and versatility of Rhino. Furthermore, this has been done at minimal extra cost, as all these hardware peripherals are natively supported by the AM3517 processor.

The processor is clocked by three different oscillators: a 26MHz system clock that is multiplied internally to generate the various CPU clocks; a 48MHz oscillator that clocks the USB sub-system on the processor; and a 32kHz clock that is used in low-power modes.

4.1.3 **Debugging Sub-Systems**

The FTDI USB-to-UART/JTAG/I^2C converter IC (shown in top-right corner of the sub-system diagram) provides most of the debugging facilities on Rhino. This single device has a USB port, which when connected to a personal computer (PC), creates two virtual serial ports, a virtual JTAG port and a virtual I^2C port. Applications running on the PC can read and write to these virtual ports as if they were separate physical ports on the computer. The two serial ports on the FTDI chip are wired to UARTs on the AM3517 processor. The JTAG port drives the board-wide JTAG chain on Rhino. Lastly, the I^2C port connects to the I^2C power management bus on Rhino, allowing all the power rails to be monitored from a PC, without needing the AM3517 processor.

4.1.4 **Power Generation and Monitoring Sub-Systems**

The power generation and monitoring sub-systems are shown in the top half of Figure 4.1. The two dark orange blocks represent the FPGA and processor power supplies. The lighter orange block at the very top depicts the monitoring sub-systems, which are responsible for monitoring all the power supplies, temperatures and fans.

The power generation and monitoring sub-systems are shown in more detail in Figure 4.2. Power is supplied to the board by an external 12V power supply, which directly powers all the on-board switch-mode power supplies. The large light-orange block on the left of the diagram (labelled "FPGA and FMC Power Generation") shows the four switch-mode power supplies for the 1.2V, 1.5V, 2.5V and 3.3V rails for the FPGA and FMC mezzanine cards. The 0.75V and 1.2V supplies (for the DDR3 SDRAM termination resistors and GTP transceivers, respectively) take the form of LDO (low drop-out) regulators to minimise ripple, and are powered off the 1.5V switch-mode supply. The FMC cards are not powered directly off the FPGA supplies, but via load-switches that allow the cards to be turned on only when needed. These load switches, along with the switch-mode supplies and LDO regulators are switched on and off by enable lines connected to GPIO pins on the AM3517 processor.

The processor is powered by a single 3-channel switch-mode power supply IC (TPS65023). This IC generates the 1.2V (core), 1.8V (SDRAM) and 3.3V (I/O) necessary to power the AM3517 processor [41]. Since this power supply cannot be powered directly off the input 12V, a secondary switch-mode power supply is used to step the input 12V down to 5V. This 5V is also used for powering the USB host ports.

All the power supplies on Rhino are monitored using INA219 power monitor ICs. These devices are able to monitor both the output voltage and current (using a current-sensing resistor) of a single supply, and report these results over an I^2C interface [42]. A 3-channel temperature monitor allows the temperature of the FPGA heatsink, the processor heatsink and the ambient air to be continuously measured and reported over I^2C. Lastly, a fan controller IC is used to both control and measure the speed of two case fans. All these board monitoring devices (power, temperature and fan) are shown in the light-purple block at the bottom of Figure 4.2, and are connected to a common I^2C bus, which is controlled by the AM3517 processor.

In a typical use-case scenario, the processor would read all the power supply voltages and currents once a second, and if it detected an error condition (such as over-voltage or over-current), it would shut down the offending power supply and log a fault to the SD card. The processor would also measure the temperatures once every five seconds, and if a device was running too hot, it would increase the fan speeds. If this did not help sufficiently, it would then switch the offending component off.

Lastly, a push-button controller is used to manage the power-up and power-down sequences on the board. This controller enables the processor 5V supply when the user presses the main power switch, hence booting the

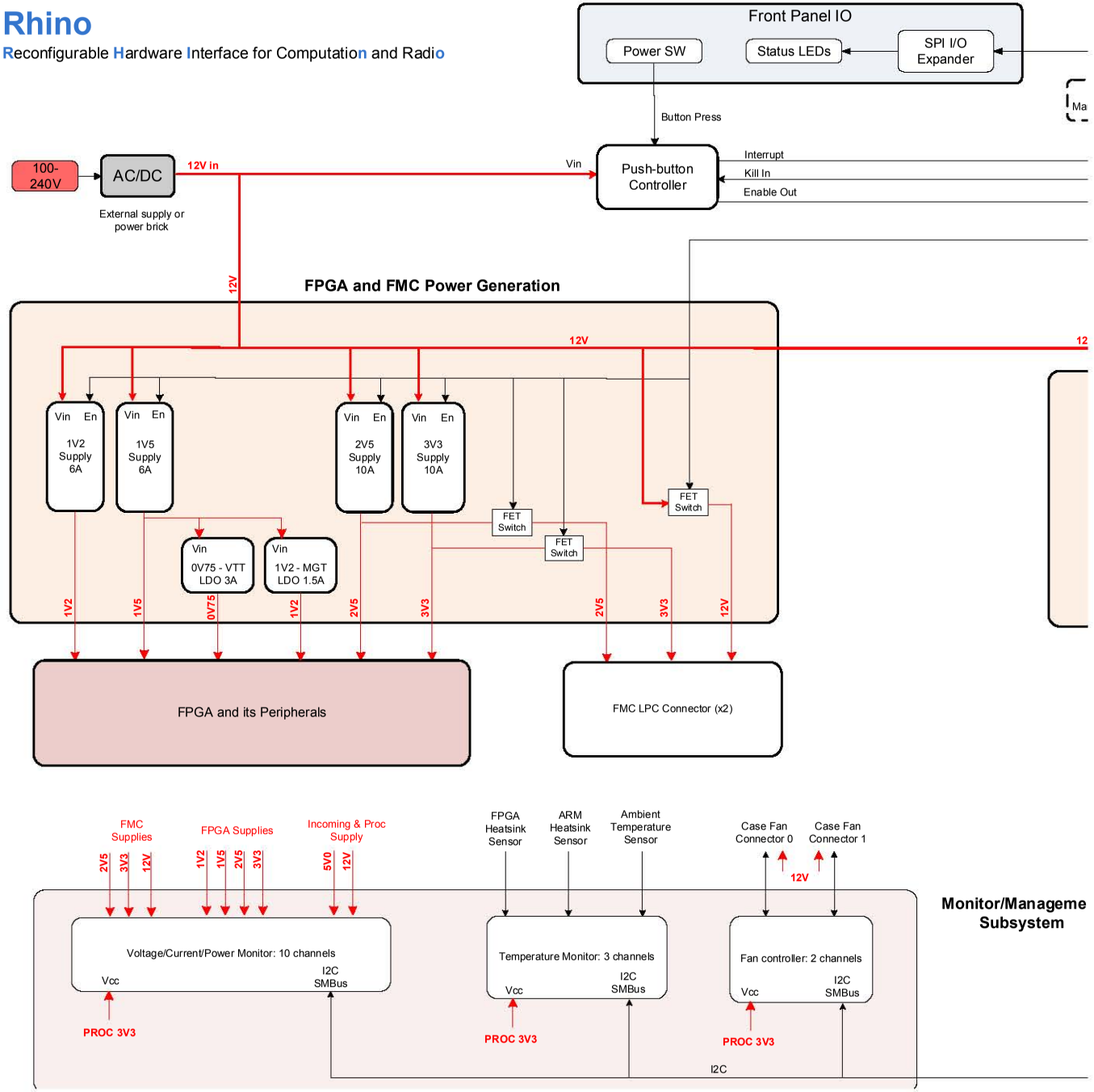


Figure 4.2: Power generation and management for Rhino

processor. If the user initiates a shutdown from within Linux, the processor would send a kill signal to the push-button controller, which in turn switches the processor supply off. If, however, the user switches off the board by pressing the main power button again, the processor is sent an interrupt signal, informing it that it has 4 seconds to save all data and gracefully shutdown before the supply is switched off.

4.2 THE RHINO SCHEMATICS

Now that the design of Rhino has been sufficiently detailed at a sub-system level, the design is transferred to schematics. The schematics, which have been included in Appendix B, were drawn up using Altium Designer. It should be mentioned that these schematics are hierarchical in nature. The first schematic sheet is the top-level sheet which contains component blocks (called sub-sheets), representing other lower-level schematic sheets. These schematics sheets can in turn contain other sub-sheets, creating a schematic hierarchy. The sub-sheets are shown as rectangular boxes (usually green in colour) with input and output ports that can be connected to other sub-sheets that occur in the same top-level sheet.

An example of this hierarchical schematic system is shown in Figure 4.3. Each of the dashed black boxes represents a separate schematic sheet. The *top_level* schematic sheet contains two sub-sheets: *switches* and *leds*. Each of these two sub-sheets have two ports, which are connected together in the *top_level* sheet. The inner workings of these two sub-sheets are shown on two separate schematic pages, with the same names: *switches* and *leds*. On the *switches* schematic, the two ports, *sw1* and *sw2* are outputs, while the two ports, *led1* and *led2* are inputs on the *leds* schematic sheet. Note that the *top_level* schematic is at a higher level in the schematic hierarchy than the other two schematic sheets.

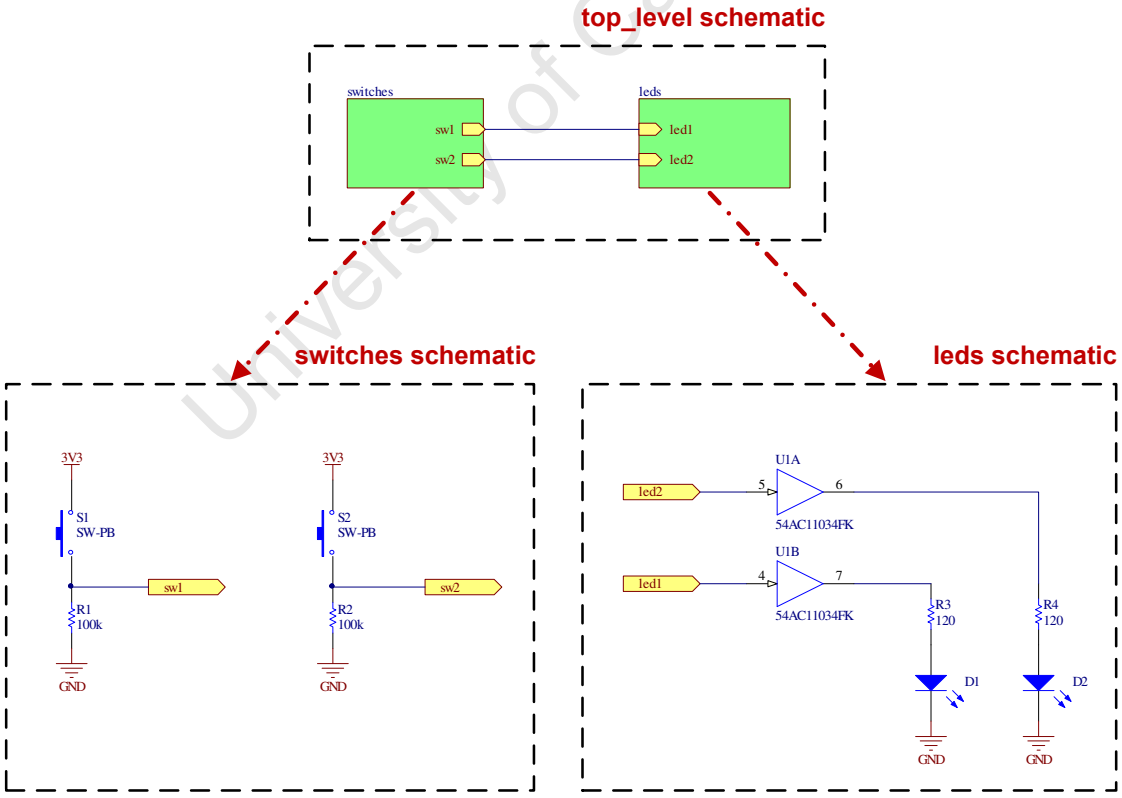


Figure 4.3: An example of Altium's hierarchical schematic structure

This hierarchical nature of Altium Designer simplified the process of converting the sub-system diagram to schematics. Each block in the sub-system diagram was drawn as a "block" (or sub-sheet) in the top-level

schematic, with the appropriate input and output ports. The internal workings of each of these sub-sheets were then later described in lower-level schematic sheets.

This section is best read in conjunction with the actual schematics in Appendix B. It explains the functions implemented by each schematic sheet and how the different sheets link together. It also aims to explain the reasoning behind the major design decisions that were made on each sheet.

4.2.1 Cover Page, Table of Contents and Rhino Power Distribution and Monitoring

Both the architectural block diagram and power generation diagram have been repeated on the first few pages of the schematics. A table of contents has also been included allow the reader to quickly find the pertinent schematic sheet. Note that the page numbers in the table of contents refer to the schematic sheet numbers (in the title box in the bottom right corner of each schematic sheet) and not the dissertation page number.

4.2.2 Rhino Overview

The Rhino Overview page is the top level schematic sheet for the entire schematic project. It has been drawn so that it mirrors the sub-system diagram as closely as possible (see Figure 4.1). Each block in the sub-system diagram has now been replaced with a sub-sheet, as represented by each of the coloured rectangles in the Rhino Overview schematic. For example, the purple Xilinx Spartan-6 block in the sub-system diagram has been replaced by the large, purple-pink *spartan6* sub-sheet block on the left-hand side of the Rhino Overview schematic.

As has already been explained, each of these blocks, or sub-sheets, represent actual schematic sheets lower down the schematic hierarchy. The schematic for the *spartan6* sub-sheet is shown on page 5 of the schematics.

The main differences between the original sub-system diagram and the Rhino Overview schematic are:

- The debugging sub-system is represented by the *usb_to_jtag_rs232* sub-sheet.
- To clearly illustrate how all the devices connect together to form the JTAG chain, a separate *jtag_chain* sub-sheet has been created. There is no such block in the sub-system diagram.
- There is only one sub-sheet for both of the processor's DDR2 SDRAM ICs, while the sub-system diagram had two separate blocks.
- All of the power generation and management blocks in the sub-system diagram have been replaced by the single *pwr_supply_management* sub-sheet. The reason for this is that the power nets are all global, and hence do not need to be represented by connections on the top-level schematic.

4.2.3 Spartan-6 (top-level)

This is the schematic for the *spartan6* sub-sheet in the top-level schematic. It itself contains a further 10 sub-sheets, represented by the 10 green boxes. These sub-sheets are for the six I/O banks, the configuration interface, the multi-gigabit transceivers (MGT), the power supply pins and the supply decoupling capacitors.

The ports (yellow and blue hexagons on the left and right edge of the schematic) represent the ports on the *spartan6* sub-sheet block on top-level schematic, which are wired to the different FPGA peripherals. On this schematic page, these ports are now wired directly to their relevant I/O bank, configuration interface or the transceivers.

It is useful to note that there are three types of connections (nets or wires) shown on this schematic page: thin blue wires, thick dark-blue wires and thick pale-blue wires. The thin blue wires are single nets. The thick, dark-blue wires are buses, which contain a number of wires with identical functions. An example of such a bus is the 8-bit USER_LED[7..0] bus, which drives the FPGA LEDs. The third type of connection is the thick, light-blue wires called harnesses. Harnesses are used to group together nets that perform related, but not identical functions. The FPGA_CONFIG_BUS harness is shown in the bottom left-hand corner of the page. This harness contains five different nets (DONE, PROGRAM_B, etc) that together form the FPGA configuration interface.

4.2.4 **Spartan-6 Bank 0**

The data bus for the second FMC connector (*FMC1*) is connected to I/O Bank 0 on the FPGA. Since the FMC interface uses LVDS, there are actually two separate buses: FMC1_LA_P[33..0] and FMC1_LA_N[33..0]. The eight pins on the HPC part of the FMC connector that are used purely for the FMC-to-Z-DOK+ adaptor board, are also connected to I/O Bank 0.

The Spartan-6 I/O Bank 0 contains eight global clock pins, indicated by "GCLK" in the pin name. These global clocks are connected to bits 0, 1 and 17 of the FMC bus, as the FMC specification states that these lines can be used for source-synchronous clocking. The 100MHz system clock, which is used for clocking the FPGA fabric, is also connected to one of the global clock inputs.

4.2.5 **Spartan-6 Bank 1**

Bank 1 has a myriad of peripherals connected to it, with the main one being the FPGA-processor bus. The signals that make up this interface are shown in the *FPGA_PROC_BUS* harness at the centre of the page (the large, pale-blue rectangle). The harness contains a 10-bit address bus, a 16-bit data bus, and a number of control signals.

There are also a number of FMC control signals wired to Bank 1 for the I^2C interface. As has been mentioned, bits 0, 1, 17 and 18 of the FMC data bus can be used for source-synchronous clocking. However, the dedicated clock lines, FMC0_CLK[1..0]_M2C_[N/P], can also be used to transmit the clock from the FMC card to the FPGA. Unfortunately, due to a shortage of differential global clock inputs on the FPGA, these differential signals must first converted to single-ended CMOS signals (by the *dual_lvds_receiver0* block) before they are connected to FPGA global clock inputs. The details of this differential-to-single-ended converter are shown on the next page of the schematics.

As was explained in Section 4.1.2, the DP83640 Ethernet PHY is used for the 100Mbps processor Ethernet, as it supports the Precision Time Protocol. The 1Hz precision output clock of the PHY is connected to Bank 1 of the Spartan-6 (1588_CLK). Furthermore, the Ethernet PHY has four GPIOs that can be programmed to signal events at certain times. These four lines (1588_GPIO[3..0]) are also connected to Bank 1.

4.2.6 **Dual LVDS Receiver 0**

This schematic shows the inner workings of the dual differential-to-single-ended converter that was used for the FMC clock signals on the previous schematic sheet. Two MAX9111 LVDS receivers are used to convert the two LVDS signals to two 3.3V CMOS signals. Since the MAX9111 is rated for clock signals up to 250MHz, it does place a limitation on the maximum frequency of the FMC mezzanine-to-carrier (M2C) clock signals. If data is transmitted on both the rising and falling edges of the clock, this allows data rates up to 500Mbps. Since the requirements for Rhino, in Table 1.2 of Chapter 1, state that the maximum data rate on the FMC interface is 400Mbps per line, this limitation should not however cause any clocking problems.

4.2.7 **Spartan-6 Bank 2**

The Spartan-6 I/O Bank 2 is nearly identical to Bank 0. Connected to Bank 2 are the data bits of the first FMC connector, *FMC0*. Again, bits 0, 1 , 17 and 18 of the FMC data bus are connected to global clock inputs on the FPGA. However, only four of the eight extra lines for the Z-DOK+ adaptor board are connected to Bank 2, as some of the bank's pins are used for the serial configuration interface (the interface used to program the FPGA). The configuration pins CONF_CCLK, CONF_DIN and CONF_INIT_B are all connected to the FPGA_CONFIG harness on the Spartan-6 (top-level) schematic (sheet 5), while the rest of the configuration pins are either pulled high or low with resistors.

4.2.8 **Spartan-6 Bank 3**

Bank 3 contains the GMII interface to the 1Gbps Ethernet PHY, the 8 User LEDs, the 16 GPIO lines, and the remaining four Z-DOK+ adaptor lines that did not fit into Bank 2. Since Bank 3 of the FPGA does not have internal termination for LVDS signals, external termination resistors are required for these four FMC_0_ZDOK_[P/N][1..0] lines.

The Gigabit Multimedia-Independent Interface (GMII) contains 8 data lines that carry data from the FPGA to the PHY (TXD[7..0]), and 8 data lines that carry data in the reverse direction (RXD[7..0]), allowing full duplex communication. The TX_CLK and RX_CLK clock signals are inputs to the FPGA, and are therefore connected to global clock pins. The slower, serial, management data interface used for controlling the Ethernet PHY is also connected to Bank 3. This interface consists of the MDIO, MDC, INT, RESET and COMA signals.

Just like in Bank 1, the two dedicated FMC clock pairs, FMC1_CLK[1..0]_M2C_[N/P], pass through a differential-to-single-ended converter before being tied to global clock inputs on the FPGA. Note that in Bank 1, the dedicated clock lines were for FMC connector 0, while in Bank 3 they are for FMC connector 1. Even though there are unused global clock inputs on Bank 3, the FMC clocks must still be converted to single-ended signals, due to restrictions on the routing of differential clocks within the FPGA. These restrictions will be explained in more detail later in this chapter.

4.2.9 **Dual LVDS Receiver 1**

Same as *Dual LVDS Receiver 0*.

4.2.10 **Spartan-6 Bank 4 and Bank 5**

In order to simplify the interfacing to external SDRAM devices, the Spartan-6 contains four Memory Controller Blocks (MCBs). The blocks take the form of special circuitry within the FPGA itself that is able to interface directly to external SDRAM components, via dedicated FPGA pins. Bank 4 of the Spartan-6 FPGA contains the connections for MCB 4 (which connects to DDR3 SDRAM 0), while Bank 5 contains the connections for MCB 5 (for DDR3 SDRAM 1). The SDRAM interface (represented by the *DDR3_BUS* harness), consists of a 16-bit data bus (DQ[15..0]), a 14-bit address bus (A[13..0]), 3 bank address lines (BA[2..0]), 2 differential data strobes (xDQS_[P/N]), a differential clock (CK_[P/N]) and 8 control lines.

4.2.11 **Spartan-6 Multi-Gigabit Transceivers**

The high-speed GTP transceivers on the Spartan-6 are often referred to simply as MGTs (Multi-Gigabit Transceivers). The XC6SLX150T-4FGG676C has 8 such transceivers, each consisting of a single transmitter and a single receiver. Since each of the transceivers runs at 3.125Gbps, differential signalling is used to minimise noise and improve signal integrity.

Four of the transceivers are connected to CX4 connector 0 (indicated by the top *MGT_BUS* harness), while the other four are wired to CX4 connector 1 (indicated by the bottom *MGT_BUS* harness). Capacitors are placed on all the receive lines to remove any DC offset that may have been injected by a network switch.

Since all the transceivers run at the same speed, only two reference clocks are used (see the MGT_CLKS_IN port), each running at 156.25MHz.

4.2.12 **Spartan-6 Configuration**

The preferred method of programming the Spartan-6 FPGA on Rhino is via the Serial Configuration Interface. The serial clock and data pins (CONF_CCLK and CONF_DIN) are actually part of I/O Bank 2, and were discussed earlier in this chapter. These pins are driven by an SPI port on the ARM processor. The DONE pin, which is shown on this schematic page, goes high when the FPGA is correctly programmed. This signal connects to both an LED (which is ON when the FPGA is *not* configured) and to the processor. The ARM processor pulls PROGRAM_B_2 to clear the FPGA and begin a new programming sequence. This pin is also pulled low by the temperature sensor (see RESET port) if the FPGA gets too hot, thereby resetting the FPGA.

The FPGA can also be configured using JTAG. This can either be done via the main JTAG chain, or via the dedicated FPGA JTAG header (connector J10 on this schematic page). A 4-bit, 2-to-1, bus selector is used to select either the JTAG chain or the JTAG connector as the source for the JTAG interface on the FPGA. The selector is controlled by a jumper (P1) on the Rhino PCB. The dedicated FPGA JTAG header is a fail-safe mechanism that allows the FPGA to be configured even if the board-wide JTAG chain is not working and the processor is not running. The JTAG header therefore helps to reduce risk and aid debugging.

4.2.13 **Spartan-6 Power and Supply Decoupling Caps**

The *Spartan-6 Power* schematic page shows all the power supply pins on the FPGA. I/O Bank 0 (with power supply pins VCCO_0) and I/O Bank 2 are both powered with 2.5V, as these banks connect to the LVDS data pairs on the FMC connectors. I/O Bank 1 runs at 3.3V so that it can interface with the processor's GPMC bus, while I/O Bank 3 runs at 2.5V so that it can interface with the 1Gbps Ethernet PHY. I/O Banks 4 and 5 are set to 1.5V for interfacing with the DDR3 SDRAM ICs.

The supply decoupling capacitors are shown on the following schematic page. These are placed directly underneath or around the FPGA on the PCB, as close to the power supply pins as possible.

4.2.14 **DDR3 RAM**

The FPGA's large capacity memory takes the form of two 2Gb DDR3 SDRAM ICs. These two ICs are shown on sheets 18 and 19 of the schematics. The *DDR3_BUS* harness is identical to the harness of the same name that was described on the *Spartan-6 Bank 4* schematic (sheet 12): 14 address (A) lines, 3 bank address (BA) lines, 16 data (DQ) lines and a number of control signals.

Since DDR3 technology supports on-die termination (ODT) for the data signals, no external termination resistors are needed for the DQ[15..0] bus. External termination resistors are however needed for the address, bank address and high-speed control lines.

The dotted red boxes placed around certain net names on the schematic sheet are PCB directives: rules that define how the nets must be routed on the PCB itself. Generally these rules are used to indicate that certain nets must be routed as traces with a specified impedance. The other type of PCB directive that is used in the schematics is shown in Figure 4.4. This symbol indicates that two nets must be routed as differential pairs.

Figure 4.4: The PCB directive for a differential pair

4.2.15 **FMC HPC Connector**

The schematics for the FMC HPC connectors are shown over sheets 20 to 27. Note that each sheet is shown twice, as there are two FMC connectors. Even though the two copies of each schematic sheet may look identical at first glance, the component designators (C265, J17, etc) are in fact different.

The *FMC HPC Connector (top level)* schematic is the only FMC schematic of interest; the rest simply show the *FMC_HPC_x* harnesses being trivially wired to each of the rows on the FMC connector. On the *FMC HPC Connector (top level)* schematic, one can see that the 34-bit data bus (LA_P[33..0]) is connected to rows C, D, G and H. These four rows form the LPC part of the connector. The other six rows, A, B, E, F, I and J form the HPC part of the FMC connector. There are also connections for power, I^2C, JTAG and the CLKx_M2C dedicated clock lines, all of which are on the LPC part of the connector.

The additional four data pairs for the FMC-to-Z-DOK+ adaptor board are connected to row E, which lies within the HPC section.

4.2.16 **CX4 10Gbps Ethernet Connector**

Again, there are two copies of the CX4 connector schematic (sheets 28 and 29), as there are two separate CX4 connectors on the board. The main component on the CX4 schematic page is the CX4 connector itself, shown on the upper half of the sheet. The connector has 16 signal pins, allowing 8 differential pairs, or 4 XAUI lanes (containing both a transmitter and a receiver pair) to be connected. PCB directives have been included on the schematic to show that these signals must be routed as differential pairs with 100 ohms differential impedance.

The lower half of this schematic page contains the circuitry necessary to power any CX4-to-optical adaptors that may be plugged into Rhino. This circuit was adapted from the datasheet for the Fujistu o-microGiGaCN device [43], one such CX4-to-optical adaptor. These adaptors have a standard CX4 connector that can plug into Rhino. However, laser diodes and detectors are placed directly inside the connector, and the copper cable is replaced with four optical fibre lines, allowing the cable to be as long as 300m.

In order to power these devices, four of the ground pins on the CX4 connector are reassigned to serve as power and control signals for the optical adaptor. One such pin is G7, which acts as a type-sense signal. When an optical adaptor is plugged into this socket, the adaptor sets this pin to approximately 1.66V. When the TYPE_SENSE net is at this voltage, both comparators in IC U66 are switched on, activating the 2A load switch, U67, which provides 3.3V to pin G8 on the CX4 connector, hence powering the optical adaptor.

It is important to note that this circuit is fully compatible with standard CX4 cables. If a standard CX4 cable is connected, the optical power is automatically switched off. A green status LED indicates when the optical power is switched on, while a red LED indicates if there is an optical fault.

4.2.17 **1 Gbps Ethernet PHY**

The Marvell 88E1111, depicted by the large yellow rectangle in the centre of the schematic, is the Gigabit Ethernet transceiver of choice for Rhino. The transceiver, also known as a PHY, is connected to the Spartan-6 FPGA by the Gigabit Multimedia Independent Interface (GMII), which consists of an 8-bit transmit bus,

an 8-bit receive bus and a number of clocks and control lines. Both the transmit bus and receive bus have source termination series resistors to help improve signal integrity. The optimum values of these resistors were determined by performing signal integrity simulations, which will be described in Chapter 5.

The transceiver converts these two 8-bit buses into four differential signals (MDI0 to MDI3) that can be transmitted down a Gigabit Ethernet cable. These four differential signals are connected to the RJ45 Ethernet jack on the right of the schematic. The transformers/magnetics on the MDIx lines, which are required for isolation and common-mode noise rejection, are integrated into the RJ45 connector itself.

The Marvell 88E1111 also has a number of LED outputs that can be programmed to perform different functions. The circuit has been designed with the following settings indicated in Table 4.1 in mind.

Table 4.1: LED Settings for the Marvell 88E1111 Gigabit Ethernet PHY

LED	**Function**
LED_LINK10	On if link up (at any speed), off if link down
LED_LINK100	Don't care
LED_LINK1000	On if link running at 1000Mbps, off if any other speed
LED_DUPLEX	Don't care
LED_RX	Idle high, pulse low if Ethernet packet received
LED_TX	Idle high, pulse low if Ethernet packet transmitted

The LEDs do not perform these functions by default; the PHY must be correctly programmed by the FPGA, using the serial management data interface (pins MDC and MDIO), before the LEDs will function as intended. The LED_LINK1000, LED_RX and LED_TX pins connect to LEDs on the PCB, while the LED_LINK10 drives the green LED on the RJ45 connector. The LED_RX and LED_TX signals are also OR'ed together using diodes so that they flash the yellow LED on the connector if a packet is transmitted or received.

The Marvell 88E1111 PHY includes 7 configuration pins (CONFIG0 to CONFIG6) that are used to set certain hardware settings at power-up. By tying these configuration pins to 2.5V, ground, or one of the six LED pins, each CONFIG input can be set to one of eight values. These hardware settings are explained in the Marvell 88E1111 datasheet [44]. The settings for Rhino's PHY are given in the "General Notes" box at the bottom of the schematic sheet.

The JTAG pins connect to the board-wide JTAG chain on Rhino, although this interface should never need to be used. The Marvell PHY also contains two debug pins, HSDAC+ and HSDAC-, whose output voltage indicates the type of fault that has occurred. These two signals are connected to test-points on the PCB.

4.2.18 **Spartan-6 GPIO Header and LEDs**

A GPIO header connector has been included on Rhino. This connector contains 16 GPIO lines that connect directly to 2.5V FPGA I/O pins. All pins are protected against short-circuits by 100 ohm series resistors, against ESD spikes by ESD protectors (ESDA5V3SC6), and against over-voltage inputs by 3.3V zener diodes.

There are also 8 yellow LEDs, connected directly to FPGA pins, to provide a simple debugging interface for FPGA gateware designs.

4.2.19 **Spartan-6 Clocks**

Besides the clocks on the GMII interface, the processor bus and the FMC cards, all of which can be used to clock the FPGA fabric, two dedicated oscillators are also used. The top oscillator on the *Spartan-6 Clocks*

schematic page (U57) runs at 156.25MHz and is used for clocking the GTP transceivers on the Spartan-6. Since two clock signals are required to clock all eight transceivers, a clock splitter (U36) is used to split the clock into two separate signals. These LVPECL clock signals are appropriately terminated and AC-coupled, as specific in the Spartan-6 GTP Transceivers User Guide [45].

The second oscillator is a 100MHz LVDS-output device that should be used for clocking most of the logic on the FPGA. All the clock signals are routed as 100 ohm, differential traces.

4.2.20 **AM3517 (top-level)**

This is the main schematic for the AM3517 ARM processor. It is represented by the large green sub-sheet on the top-level schematic (sheet 4). The AM3517 schematic contains four sub-sheets, which represent the different parts of the processor: *am3517_periph_a*, *am3517_periph_b*, *am3517_power* and *am3517_decoupling*.

Most of the ports on the left- and right-hand edge of schematic connect directly to ports on the appropriate AM3517 sub-sheet. There are, however, certain ports that are not directly mapped and require some explanation. The first ones are the FPGA_PROC_BUS and NAND ports. Since these two devices both connect to the GPMC (General Purpose Memory Controller) interface on the processor, the FPGA_PROC_BUS and NAND harnesses are expanded and both connected to the GPMC harness.

Since the real-time clock, audio codec, and front-panel PCB are all connected to the same SPI (serial peripheral interface) port, their data and clock lines are tied together. Each device obviously has its own chip-select line.

The last set of signals that require some explanation are the warning signals in the bottom-left corner of the schematic. Due to lack of processor pins, these signals are all connected to the SYS_BOOT pins on the processor. The SYS_BOOT input pins are used to specify booting options, such as which memory device from which to boot and which clock sources to use. Since the warning signals have unpredictable state, they obviously cannot be connected to the SYS_BOOT pins at startup. The warning signals are instead buffered before being wired to the SYS_BOOT pins, and this buffer is disabled at boot time.

4.2.21 **AM3517 Peripherals (Part A)**

This schematic shows most of the parallel interfaces on the processor. The SDRC (SDRAM Controller) bus consists of a 32-bit data bus, a 14-bit address bus, a 3-bit bank address bus, and a number of control signals. This bus is wired directly to the two DDR2 SDRAM ICs.

The GPMC interface is also shown, and has already been explained. The DVI (Digital Video Interface) harness is shown on the left of the schematic, and connects to a video transmitter IC. Both the DVI and the SDRC interfaces have 22 ohm source series termination resistors, to help prevent overshoot at the receiver.

The other smaller interfaces shown on this schematic include the FPGA configuration interface, which is connected to SPI port 2 and is used for programming the FPGA; the SD card interface; debug UART 1 which is connected to the FTDI USB-to-UART converter; and the UART for the modem serial port (RS232_DATA).

4.2.22 **AM3517 Peripherals (Part B)**

Most of the processor pins on this schematic page are used as GPIOs to control peripherals. The few pins that are used to perform other functions include the three I^2C interfaces (I2C_PWR_MAN, DDC_I2C and FMC_I2C); the 100Mbps Ethernet RMII interface (ETHER_RMII); the JTAG pins (PROC_JTAG); the two USB host ports (HS_USB1 and HS_USB2); and the multi-channel buffered serial port (McBSP2) which

streams audio samples to the audio codec (AUDIO_DATA).

The SYS_BOOT pins, which were described in Section 4.2.20, are also shown on this schematic page (bottom-right corner). Since the buffer for the warning signals is switched off at boot time, the levels of these pins are determined by the pull-up and pull-down resistors alone. The levels of SYS_BOOT3 and SYS_BOOT5 can however be set using two switches (S1), allowing the boot device order to be changed by the user. These two switches allow either the NAND flash memory, the MMC/SD card or the Ethernet link to be set as the preferred boot device. Once the processor has booted, the SYS_BOOT pins become normal GPIO pins, and the warning signal buffer is enabled.

4.2.23 **AM3517 Power and Supply Decoupling Caps**

The processor core runs at 1.2V, the SDRAM interface at 1.8V and the I/O pins at 3.3V. Just like with the FPGA, a large number of supply decoupling capacitors are required to keep the supply stable. These capacitors are placed directly underneath or around the processor on the PCB, as close to the supply pins as possible.

4.2.24 **AM3517 DDR2 RAM (top level)**

The processor has two 1Gb DDR2 SDRAM ICs connected to it. Each of these two ICs, which are represented by the two green sub-sheet blocks on this schematic, have a 16-bit data bus. The data buses of these two RAM chips are combined to form a single 32-bit data bus, which connects to the processor's SDRC (SDRAM Controller) interface. All the control signals (except the data masks, UDMx and LDMx, and the data strobes, UDQSx and LDQSx) and address signals are common for both SDRAM ICs.

4.2.25 **DDR2 RAM**

This schematic shows the internal workings of the *ram_ddr2* sub-sheets on the previous page. There are two *DDR2 RAM* schematic pages, as there are two RAM chips. At the centre of this schematic page is the Winbond 1Gb DDR2-800 SDRAM IC. Although this device is capable of running at DDR2-800 speeds, the processor is only able to clock the SDRC interface at DDR2-333 speeds.

The RAM chip has a 16-bit data bus, a 14-bit address bus, a 3-bit bank address bus, and a number of control lines. Although the AM3517 processor supports on-die termination for DDR2 SDRAM ICs, the data sheet still recommends that external *source series* termination is used on all data, address and control lines [37]. Therefore, 22 ohm series resistors have been placed on the address and control lines directly at the processor, while the series termination resistors on the data lines are placed directly at the RAM chip itself.

All DDR2 RAM traces are routed with 50 ohms characteristic impedance; or 100 ohm differential impedance in the case of the differential pairs.

4.2.26 **NAND Flash**

The Micron 256MB NAND flash chip is typically used for storing the processor bootloaders and Linux kernel, but may also be used for storing FPGA configuration files. The flash device connects to the processor GPMC interface, as was described in Section 4.2.20. It is interesting to note that the NAND flash memory does not have any address lines, just an asynchronous data interface (IO[15..0]) and control lines. The 16-bit data interface is multiplexed within the device to transfer commands, address and data as efficiently as possible.

4.2.27 **100Mbps Ethernet PHY**

A National Semiconductor DP83640 Ethernet PHY provides the physical layer interface between the ARM processor and the Ethernet cable. The PHY communicates with the processor over the Reduced Media Independent Interface (RMII), which consists of a two-bit transmit and a two-bit receive bus. A management data interface (MDIO and MDC) has also been provided to allow the processor to configure the PHY.

On the physical layer side of the device, two differential pairs (one transmit, one receive) connect to the RJ45 Ethernet jack. Again, the magnetics necessary for isolation and common-mode noise rejection are integrated into the RJ45 jack. The PHY also has three LED outputs: LED_LINK (indicates that link is up), LED_SPEED (on if link running at 100Mbps) and LED_ACT (indicates network activity).

As has been mentioned, this particular Ethernet PHY supports the IEEE1588 Precision Time Protocol. The purpose of this protocol "is to synchronize the time between different nodes on an Ethernet network" [46], with an accuracy of 10ns or less. The DP83640 Ethernet PHY has an internal clock that is synchronised to a master clock using the Precision Time Protocol, which is achieved by sending special packets over the Ethernet link. This internal, synchronised clock is used to generate the 1 pulse-per-second (PPS) clock output, which can be used to clock either the FPGA or the processor (see the 1588_CLK1 and 1588_CLK2 output ports in the top-right corner of the schematic). The Ethernet PHY can also be programmed, by the master clock, to generate an event at a particular time. This event is indicated by one of the GPIO outputs on the PHY (see output port 1588_GPIO[3..0]), which are connected to both the processor and the FPGA.

The PHY also includes a JTAG interface for debugging. The interface is connected to the Rhino board-wide JTAG chain, but it is not expected to ever be used.

4.2.28 **RS-232 Header for Peripherals**

In order to allow hardware peripherals with RS-232 interfaces (such as GPRS modems) to be connected to Rhino, a RS-232 serial port has been included. This serial port may be useful in network radar applications, where the easiest way to communicate with each node is via GPRS. The serial port supports the full RS-232 protocol, with hardware flow control and modem signals. A MAX3243 level converter is used to convert the 3.3V signals from the processor to RS-232 level signals that can be sent down a serial cable.

Due to lack of space along the connector edge of the Rhino PCB, no 9-pin D-SUB (DB-9) connector is included on the Rhino PCB. Instead, the signals are connected to a 10-pin header. The intention is that a panel-mount DB-9 connector will be installed on the back plate of the Rhino enclosure, and this connector will be connected directly to the PCB header with a ribbon cable.

4.2.29 **AM3517 Clocks**

Three oscillators are used to clock the AM3517 processor. The 26MHz system clock is used to clock the processor core and most of the on-chip peripherals; the 32kHz clock is used when the processor is in standby mode; and the 48MHz oscillator clocks the USB on-chip hardware on the AM3517.

4.2.30 **USB On-the-Go**

USB On-the-Go (OTG) is an extension of the USB 2.0 specification, that allows devices to act as *both* USB hosts *and* USB peripherals [47]. It also allows two OTG devices to be connected directly to each other with a standard USB cable, without needing a hub. This is useful in the case of Rhino, as it allows a Rhino to communicate with a PC at maximum USB speeds (480Mbps). It also allows two Rhinos to be connected

directly together with a simple USB cable, if network hardware is not available.

The AM3517 has an on-chip USB On-the-Go PHY, allowing a direct connection from the processor pins to the USB connector. A ESD protector has however been added to protect the processor from electro-static discharges. The processor is, however, not able to provide the 5V supply for the USB port. This is achieved using a high-side load switch (U56), which can be switched on and off by the processor.

4.2.31 **USB Host Transceivers**

Rhino is equipped with two USB Type-A host ports. This allows USB flash drives, keyboards and USB modems to be attached to the board. The processor has two high-speed (480Mbps) USB host controllers, but requires external transceivers before it can interface to USB peripherals. These transceivers (USB3320 devices manufactured by SMSC) interface to the processor via a 8-bit bi-directional data bus, running at 60MHz (60MHz * 8 bits = 480Mbps). This 8-bit data bus, together with the bus clock and three control signals, conform to the UTMI+ Low Pin Interface (ULPI) standard.

Besides converting the 8-bit parallel ULPI interface into the serial USB bus, the transceivers also handle the low-level USB protocol details, such as negotiating the link speed with the peripheral. Both transceivers are connected to the same dual-port stacked USB connector (J11). No external ESD protectors are required, as the transceivers contain both ESD and over-voltage protection circuitry. The USB power is provided by a dual-channel high-side load switch (U52), which is controlled by the transceivers. The load switch provides over-current protection, and the over-current warning signals, USB1_OC and USB2_OC, are connected to GPIO pins on the processor.

4.2.32 **SD Card and Real-time Clock**

The SD card connector allows SD (or MMC) cards, up to 32GB, to be plugged into Rhino. The SD card will typically contain the Linux filesystem and FPGA configuration files. However, it may also contain the bootloaders and Linux kernel, as the processor can be set to boot from the SD card using the boot switches. Since the AM3517 processor contains an internal SD card transceiver, no external circuitry other than the physical connector is required.

A real-time clock (RTC) has been included to provide the processor with the current date and time, in cases where there is no GPS or network clock available. This may be useful when Rhino is used in remote locations and the recorded data needs to be time-stamped. The processor communicates with the real-time clock over SPI. In order to ensure that the clock does not lose time when the Rhino board is unpowered, a 1.5 Farad super-capacitor provides backup power for the clock. This capacitor should be able to keep the clock running continuously for over a month if no other power is available.

4.2.33 **HDMI Video Transmitter**

As has already been discussed, the AM3517 contains a graphics accelerator and video output port, which in the case of Rhino, may be used for video signal processing applications. If a keyboard and mouse are plugged into the USB host ports on Rhino, then the video output port may also allow Rhino to act as a standalone computer with a graphical user interface.

A TFP410 video transmitter is required to serialize and encode the RGB (red, green, blue) pixel data streams. The transmitter connects to the processor via a 24-bit pixel bus (DATA0 to DATA23), with 8-bits per colour. The IDCK, VSYNC and HSYNC timing signals are used to align the pixel bitstream to form video frames. These three 8-bit buses are serialised and TMDS (transition minimised differential signalling) encoded to

form three serial signals (TX0 to TX2) and one clock signal (TXC), that are transmitted to the monitor via the HDMI (High Definition Multimedia Interface) connector (J14).

The HDMI connector also supports the Display Data Channel (DDC) protocol, which allows the monitor to inform the processor which display modes it supports. The data and clock signals for the DDC protocol (DDC_SDA and DDC_SCL) connect to an I^2C port on processor, after being level shifted from 5V to 3.3V.

4.2.34 **Audio**

Rhino is equipped with line-out, line-in and headphone audio jacks. Audio samples are streamed between the processor and the TLV320AIC23 audio codec (U10) over an Inter-IC Sound (I^2S) interface (AUDIO_DATA port in the schematic). The I^2S interface connects directly to a Multi-channel Buffered Serial Port (McBSP) on the processor. An additional SPI port (AUDIO_CTRL) is used for changing settings such as the headphone volume and the sample rate. The ADCs and DACs within the codec support sample rates up to 96kHz.

External resistor-capacitor circuits are used to filter the analogue inputs and outputs. The 1nF capacitors on the LINE_IN input remove any high-frequency noise above 100kHz, while the 2.2uF capacitors block any DC components in the signal. The RC circuits for both the PHONES channels and the LINE_OUT channels act as high-pass filters, removing any DC offset and noise below 30Hz. These filter circuits were tuned using SPICE simulations.

4.2.35 **USB to JTAG/I^2C/RS-232**

The FT4232 IC from FTDI is a USB to JTAG/I^2C/RS-232 converter. It contains a USB interface that is typically connected to a PC, and four independent ports that can be configured for JTAG, I^2C or UART operation. When connected to a PC, four virtual JTAG, I^2C or RS-232 ports are created to which application software can read or write. Drivers on the PC perform the translation from JTAG, I^2C or RS-232 to USB, while the FTDI chip on Rhino translates the data it receives on the USB back to JTAG, I^2C or RS-232.

On Rhino, the USB pins on the FT4232 IC are connected to a USB Type-B connector, similar to that used in printers. This allows the USB port to be connected to a PC with a standard USB printer cable.

The four multi-function ports on the FTDI chip (A, B, C and D) are shown on the right-hand edge of the IC symbol in the schematic. Port A is configured as a JTAG master; port B is configured for I^2C, and allows the power management I^2C bus to be monitored from the PC; and port C and D are both configured as UARTs and connected to the processor. The I^2C port aids board testability, as it allows the power management devices to be monitored even if the processor is not running correctly. The JTAG port also includes two GPIO pins that are used for switching the FPGA on, so that it can be programmed via JTAG.

Since the four ports are, by default, configured as UARTs at power up, and not as JTAG or I^2C interfaces, buffers are used to prevent port A and port B causing bus contention on either the JTAG chain or I^2C power management bus, while they are still configured as UARTs. Only once the device has been connected to a PC, and the correct driver has been installed, will the driver on the PC pull the JTAG_EN line low and the I2C_EN line high, enabling both buffers (U58 and U59).

An EEPROM chip, on which is stored the USB "device name" and other settings, is connected to the FTDI IC. Four orange LEDs have also been connected to the transmit and receive lines of the two UART ports, allowing traffic to be monitored.

4.2.36 JTAG Chain

This schematic contains seven JTAG harnesses, representing the seven devices on the Rhino JTAG chain. The JTAG master port, shown on the left, is connected to the JTAG port on the FTDI USB-to-JTAG/I^2C/RS-232 converter. The first two devices on the JTAG chain are the FPGA and processor, both of which can be programmed via JTAG. The next two devices are the FMC connectors. However, since it is possible that no FMC mezzanine card may be connected, two 1-bit bus selectors (U17 and U18) are used to selectively add or remove each FMC card from the JTAG chain, depending on whether the card is present or not. This is determined by looking at the FMC_x_PRSNT signals. The last two devices on the JTAG chain are the 100Mbps and the 1Gbps Ethernet PHYs.

4.2.37 Configuration Interface Level Translator

The Serial Configuration Interface on the FPGA is used for programming/configuring the FPGA. It is connected to an SPI port on the AM3517 processor. However, the FPGA configuration interface runs at 2.5V, while the processor's SPI port is at 3.3V. Therefore, voltage translators are used to perform the conversion. A description of each of the configuration signals is given in Table 4.2.

Table 4.2: FPGA Serial Configuration Interface Signals

Signal	Direction	Description
CCLK	Proc to FPGA	Programming clock
DIN	Proc to FPGA	Programming data
PROGRAM_B	Proc to FPGA	Indicates that programming is about to begin
DONE	FPGA to Proc	Indicates that FPGA has been correctly configured/programmed
INIT_B	Bi-directional	Processor can pull low to delay programming
		FPGA can pull low to indicate that a CRC error occured

4.2.38 Power Supply Management

This is the top level sheet for all the power supply and monitoring schematics. The four green boxes in middle of the page are the four power supply sub-sheets. The large green box on the right-hand side is the sub-sheet for power monitors, while the smaller sub-sheet beneath it is for the temperature monitors and fan controllers. The sub-sheet on the far right contains the power LEDs.

This schematic sheet also contains the push-button controller, shown in the top-left corner, which switches the processor on and off when the power push-button is pressed. When the board is switched off, and the push-button is initially pressed, the controller enables the processor power supply. If the push-button is later pressed for 200ms or longer, an interrupt (INT output) is sent to the processor, instructing the processor to prepare for shutdown. Once the processor has completed its shutdown tasks, it acknowledges the interrupt by pulling the PWR_KILL port low, and the controller switches the processor supply off. If, however, the processor does not respond to the interrupt within 4.4 seconds, the controller assumes that the processor has crashed, and it is switched off anyway.

The SUPPLY_EN port on the left of the schematic is a harness containing all the power supply enable lines. These enable lines are all connected to the processor, allowing the processor to switch off parts of the board when they are not needed, hence conserving power. There are also two FPGA power suppy enable lines coming from the FTDI USB converter (FTDI_SUPPLY_EN port). These allow the USB converter to power up the FPGA so that it can be programmed. A dual OR gate has been used to combine the FPGA power enable signals from the processor and the USB converter.

The warning outputs of each supply and power/fan/temperature monitor are placed on the PWR_WARNS harness in the top right corner of the schematic. This is wired to processor GPIO pins. Below this, test points have been added for each power supply rail, to aid debugging and improve testability.

Lastly, the front panel connector is shown in the bottom left corner. This header connector contains pins for the power push-button, reset button, power LED, and I^2C and SPI buses to control front panel LEDs.

4.2.39 Spartan-6 Power Supplies

The four switch-mode power supplies for the FPGA and its peripherals are shown: 1.2V for the FPGA fabric, 1.5V for the DDR3 SDRAM; 2.5V for the FMC cards and the 1Gbps Ethernet PHY; and 3.3V for the FMC cards and processor interface.

The calculations to determine the current requirement for each supply are shown in Table 4.3. The last two columns will be explained in the next section. Although analogue switch-mode power supplies are available in many different current ratings, the most common values are 3A, 6A and 10A. Therefore, 6A supplies were selected for the 1.2V and 1.5V rails, while 10A supplies were used for the 2.5V and 3.3V.

Table 4.3: Power Requirements for the FPGA and its Peripherals

Sub-System	**Component**	**Supply Current (mA)**					
		1.2V	**1.5V**	**2.5V**	**3.3V**	**0.75V VTT**	**1.2V MGT**
FPGA	Spartan-6 XC6SLX150T	4000	2500	2000	800		1000
DDR3	Micron MT41J128M16HA		551				
DDR3 Termination	N/A		630			630	
CX4 Optical	N/A				500		
1Gbps Eth PHY	Marvell 88E1111	250		381			
FMC Connectors	N/A			4000	6000		
Total Current		**4250**	**3681**	**6381**	**7350**	**630**	**1000**

4.2.40 Spartan-6 LDO Power Supplies

Besides the four main switch-mode power supplies that power the FPGA, an additional two low-dropout (LDO) regulators are also used. A 0.75V LDO regulator provides the DDR3 SDRAM termination voltage, while a 1.2V regulator powers the multi-gigabit transceiver circuitry on the FPGA. The current requirements for these two LDO regulators are shown in the last two columns of Table 4.3.

An LDO regulator, rather than switch-mode power supply, has been used for the DDR3 termination voltage, as an LDO is cheaper than a switch-mode supply, and efficiency is not critical when the current is small. Since the multi-gigabit transceivers require a supply with extremely low ripple, the existing 1.2V switch-mode supply was not suitable, and a separate LDO has been used instead.

4.2.41 FMC Power Supply Switches

The FMC mezzanine-cards require four different power supplies, as shown in Table 4.4. High-side load switches are used to selectively enable or disable each power rail.

The 3.3V auxiliary 2.5V supplies are switched using a commercial load switches (TPS22924C). However, this switch is only rated for 3.6V up to 2A, and therefore cannot be used for the 3.3V (3A) and 12V (1A) power rails. These power rails are instead switched using a MOSFET circuit, the details of which appear on the next few pages. The MOSFET circuit is represented on this schematic using *SI6463BDQ_switch* sub-sheets.

Table 4.4: FMC Mezzanine Card Power Requirements

Supply Voltage	Supply Current	Purpose
2.5V	2A	ADCs/DACs and LVDS data bus
3.3V	3A	ADCs/DACs and control
3.3V auxiliary	20mA	EEPROM chip
12V	1A	RF circuitry (if present)

4.2.42 **Si6463BDQ FET Load Switch**

A Si6463BDQ MOSFET is used to switch the 3.3V and 12V FMC power rails. To allow the P-channel MOSFET to be switched using 3.3V logic levels, an NPN transistor is used. Since the MOSFET requires a minimum load current to operate correctly, a 10k burden resistor has been connected to the output.

There are four copies of this schematic, as there are two FET load switches per FMC card. The operation of this circuit has been verified using SPICE simulations.

4.2.43 **AM3517 Power Supply**

Texas Instruments, the manufacturers of the AM3517 processor, also manufacture a companion 3-channel switch-mode power supply for this processor. This power supply (TPS65023) has three output channels [41]: 1.2V @ 1.7A, 1.8V @ 1A and 3.3V @ 1A. Unfortunately, Rhino's 12V DC input is above the maximum input voltage that this device can tolerate; therefore, a second switch-mode supply is used to convert the input 12V down to 5V (U21). The 5V is also used for powering the USB ports.

In order to ensure that this 3-channel power supply can deliver sufficient current to the AM3517 and *all* its peripherals, the power requirements need to be determined. These requirements are summarised in Table 4.5. From this table, it is clear that the TPS65023 power supply will meet the power requirements, with at least 8% headroom on each output.

Table 4.5: Power Requirements for the Processor and its Peripherals

Sub-System	Component	Supply Current (mA)			
		1.2V	1.8V	3.3V	5V
ARM Processor	TI AM3517	1500	300	300	
DDR2 SDRAM	Winbond W971GG6JB		502		
NAND Flash	Micron MT29F2G16AADWP			45	
Ethernet PHY	National Semiconductor DP83640			100	
USB Transceiver	SMSC USB3320		58	34	
USB Host and OTG Port	N/A				1500
Video Transmitter	TI TFP410			250	
Audio Codec	TI TLV320AIC23B			36	
USB Converter	FTDI FT4232H		70	100	
Total Current		**1500**	**930**	**865**	**1500**

If one then calculates the power consumption for each rail (including the 5V), and adds them together, the total power consumption for the processor sub-system can be determined (15.4W, assuming 90% supply efficiency). Furthermore, this means that the 5V switch-mode power supply must be able to handle 3.1A. Although one could probably use a 3A supply to provide the 3.1A, the supply would run hot and shorten its lifespan. Hence, a 6A switch-mode power supply is used to provide the 5V.

The power-up sequence of the processor power supply works as follows:

- The user presses the power push-button, which is detected by the push-button controller (on the Power Supply Management schematic), and the PROC_SUPPLY_EN port (on this sheet) goes high.
- This switches on the 5V switch-mode supply, which provides power to the TPS65023 supply inputs (VINDCDCx) and control circuitry.
- Channel 3 (DCDC3) turns on immediately, providing 1.8V to the SDRAM and processor.
- Channel 2 turns on 5ms after Channel 3, providing 3.3V to the processor's I/O circuitry and other peripherals.
- Channel 1 switches on 5ms later, supplying the processor core with 1.2V.
- Finally, the two LDO regulator channels turn on 5ms after Channel 1, providing 1.8V to the digital PLL circuitry within the processor, and 3.3V to the USB circuitry within the processor.

The sequencing of the power supplies is not arbitrary; the processor has very strict requirements regarding the sequence in which its supplies must be powered up [37]. The 5ms delays in the sequencing of each supply were achieved using RC delay circuits at the supply enable inputs (DCDCx_EN and LDO_EN).

The processor power requirements in Table 4.5 can be combined with the FPGA power requirements in Table 4.3 to determine the board's total power consumption. Again, these calculations assumed that the power supplies were 90% efficient, as per the calculated efficiency that was given in Chapter 3 (see Section 3.3.7).

Table 4.6: Rhino Power Consumption

Processor and peripherals	15.4W
FPGA and peripherals (excluding FMC)	23.4W
FMC mezzanine cards	57.1W
Total power (with FMC)	**95.8W**
Total power (without FMC)	**38.7W**

4.2.44 Power Monitors

Both the voltage and the current of all power rails within Rhino are monitored. This is achieved by placing 5 milli-ohm current sensing resistors on the output of each power supply, and then using INA219 power monitors to measure the output voltage, and the voltage across the current sensing resistor. Even though each INA219 can measure only one supply rail, this is actually an advantage, as it allows the INA219 to be placed directly at the current sensing resistor at the power supply output.

The processor communicates with the INA219 power monitors over a common power-management I^2C bus. Each INA219 can be programmed with the value of the current sensing resistor, allowing it to calculate the actual current and power.

4.2.45 Temperature Monitor and Fan Controller

A 4-channel temperature monitor is used to measure the temperature of the FPGA, the processor and the ambient air. Since neither the FPGA nor the processor contain internal temperature diodes, external temperature

sensors must be glued onto the FPGA and processor heatsinks. A third temperature sensor must be placed inside the box, near the air inlet, to measure the ambient air temperature. All three of these temperature sensors connect to the board with standard two-pin Molex KK-style connectors.

The processor is able to read the temperatures, and program warning conditions, via the common power-management I^2C bus. By design, the processor should program the temperature monitor, on power-up, to generate a warning condition if the FPGA temperature rises above 70°C. If such a condition does occur, the ALERT output pin goes low, resetting the FPGA.

The TC654 2-channel fan controller is used to control two 12V case fans. It uses slow pulse-width modulation (at approximately 30Hz) to control the fan speed, which can be set by the processor using the common power-management I^2C bus [48]. The fan controller measures the actual fan speed by counting current pulses, which the processor can read using the same I^2C bus. If a fault, such as a locked rotor, does occur, the fan controller immediately alerts the processor using its FAULT output, which is wired to a processor GPIO pin.

4.2.46 Power LEDs

Five power LEDs are used to indicate whether the four FPGA switch-mode power supplies and the processor 5V supply are turned on. Furthermore, two power LEDs are connected to the "power good" outputs of the two FPGA LDO regulators (see schematic sheet 55). This allows the user to immediately determine which power supplies are on and which are off.

4.2.47 Mounting Holes and Fiducials

The micro-ATX PCB mounting holes and the FMC mezzanine card mounting holes are shown on this schematic. The fiducials used to align the PCB layers have also been shown for completeness.

4.3 Clocking the Spartan-6 FPGA

The Spartan-6 has special, dedicated, clock pins (called "global clocks") that should be used for all clock input signals. These pins are connected to special global clock buffers within the FPGA (BUFGMUXs), which drive the clock signals onto low-latency clock routing networks [49]. This ensures that the clock signal experiences minimum skew as it travels through the FPGA.

A list of all the clock input signals to the FPGA is given in Table 4.7. Note that each FMC interface has six clock lines, as four of the data pairs (0, 1, 17 and 18) can also be used as source-synchronous clocks. As one can see, there are 13 differential clocks. However, the Spartan-6 can only support 8 simultaneous differential clocks, due to limitations on clock routing within the FPGA, as will be explained later. Therefore, some of the differential clocks are converted to single-ended clocks (see fourth column in the table).

Even though the Spartan-6 has 32 global clock inputs (GCLKs), not all of these clocks can be used simultaneously. The reason for this lies in how the global clock signals are routed within the FPGA. Each global clock is routed to four clock multiplexers (BUFGMUXs). The clock signal is then routed from the output of the BUFGMUX, via a low-skew clock network through FPGA, to the user logic. This is illustrated in Figure 4.5. The GCLK pins are shown at the top of the diagram, with their connections to the BUFGMUXs on the right-hand side. This diagram shows the connections for just I/O Banks 0 and 1. A similar routing network exists for Bank 2 and 3. No GCLKs are available on I/O Banks 4 and 5. Since this means that there are a total of just 16 BUFGMUXs, only 16 of the 32 global clock inputs (GCLKs) can be used at once.

Furthermore, each GCLK is wired to only four BUFGMUXs, meaning that only certain combinations of the

Table 4.7: List of Clock Inputs to the Spartan-6 FPGA on Rhino

Sub-System	Clock Signal	Differential?	Convert to Single-Ended?
FMC0	FMC0_LA_0_CC	Yes	No
	FMC0_LA_1_CC	Yes	No
	FMC0_LA_17_CC	Yes	No
	FMC0_LA_18_CC	Yes	No
	FMC0_CLK0_M2C	Yes	Yes
	FMC0_CLK1_M2C	Yes	Yes
FMC1	FMC1_LA_0_CC	Yes	No
	FMC1_LA_1_CC	Yes	No
	FMC1_LA_17_CC	Yes	No
	FMC1_LA_18_CC	Yes	Yes
	FMC1_CLK0_M2C	Yes	Yes
	FMC1_CLK1_M2C	Yes	Yes
System Clock	SYS_CLK	Yes	No
PTP Clock	IEEE1588_CLK	No	-
Processor Bus	PROC_BUS_CLK	No	-
1Gbps Eth PHY	GMII_TX_CLK	No	-
	GMII_RX_CLK	No	-

GCLKs can be used at once. For example, by looking at Figure 4.5, it is clear that global clocks GCLK19 and GCLK11 can never be used at the same time, as they are both connected to the same horizontal clock lines, and hence are connected to the same input pins of the same BUFGMUXs. In fact, in turns out that no more than 8 differential clocks can be used at once, as the differential clocks are always routed using the "positive" GCLK pin, and there are always BUFGMUX conflicts if more than 8 "positive" GCLK pins are used. Hence Rhino only used 8 differential clocks; the rest are converted to single-ended.

Tables 4.8 and 4.9 shows how the clock sources on Rhino have been connected to the GCLK pins, so that there is no conflict in routing to the BUFGMUXs. Note that all 16 BUFGMUXs are used on Rhino. Also note that GCLK_20, in Table 4.9, has two clock sources: FMC1_CLK1_M2C or FMC1_LA_18_CC. Due to a lack of BUFGMUXs, only one of these clock sources can be used at a time. The choice is made by selectively installing 0 ohm resistors. However, this should not cause any problems, as it is highly unlikely that both FMC1_CLK1_M2C and FMC1_LA_18_CC will be used as clock lines simultaneously by the same FMC card.

All the information regarding the Spartan-6 clocks that was presented in this section is summarised in the block diagram in Figure 4.6. This diagram shows all the clock sources connected to the appropriate global clock (GCLK) pins on the FPGA.

The reader should now have a full understanding of the schematic-level design of Rhino. This chapter began by developing the detailed sub-system diagram for Rhino. This diagram explained the four major sub-systems within Rhino (FPGA, processor, debugging and power management), and how they fit together. This sub-system diagram was then transferred to the schematics, the details of which were also discussed. This chapter then ended off with an overview of the clocking infrastructure for the Spartan-6 on Rhino.

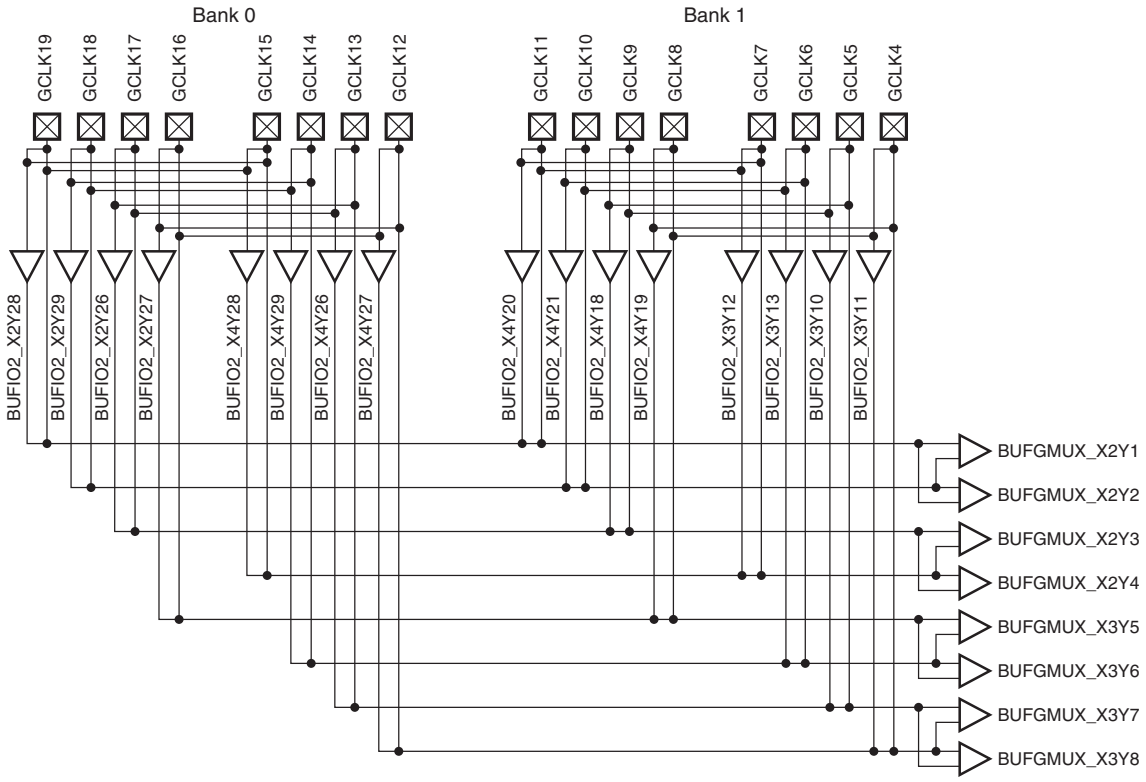


Figure 4.5: Internal routing of GCLKs to BUFGMUXs within the Spartan 6, for Bank 0 and Bank 1 (Image taken from the Xilinx Spartan-6 FPGA Clocking Resources Guide [49])

Table 4.8: Shared BUFGMUXs for Global Clocks on Banks 0 and 1 on Rhino

BUFGMUX	**Bank 0**	**Bank 1**
BUFGMUX_X2Y1	**GCLK_19**: FMC1_LA_0_CC	
BUFGMUX_X2Y2		**GCLK_10**: 1588_CLK
BUFGMUX_X2Y3	**GCLK_17**: FMC1_LA_1_CC	
BUFGMUX_X2Y4	**GCLK_15**: FMC1_LA_17_CC	
BUFGMUX_X3Y5		**GCLK_8**: PROC_CLK
BUFGMUX_X3Y6		**GCLK_6**: FMC0_CLK0_M2C
BUFGMUX_X3Y7	**GCLK_13**: SYS_CLK	
BUFGMUX_X3Y8		**GCLK_4**: FMC0_CLK1_M2C

Table 4.9: Shared BUFGMUXs for Global Clocks on Banks 2 and 3 on Rhino

BUFGMUX	**Bank 2**	**Bank 3**
BUFGMUX_X2Y9	**GCLK_3**: FMC0_LA_0_CC	
BUFGMUX_X2Y10		**GCLK_26**: GMII_TX_CLK
BUFGMUX_X2Y11	**GCLK_1**: FMC0_LA_1_CC	
BUFGMUX_X2Y12	**GCLK_31**: FMC0_LA_17_CC	
BUFGMUX_X3Y13		**GCLK_24**: GMII_RX_CLK
BUFGMUX_X3Y14		**GCLK_22**: FMC1_CLK0_M2C
BUFGMUX_X3Y15	**GCLK_29**: FMC0_LA_18_CC	
BUFGMUX_X3Y16		**GCLK_20**: FMC1_CLK1_M2C or FMC1_LA_18_CC

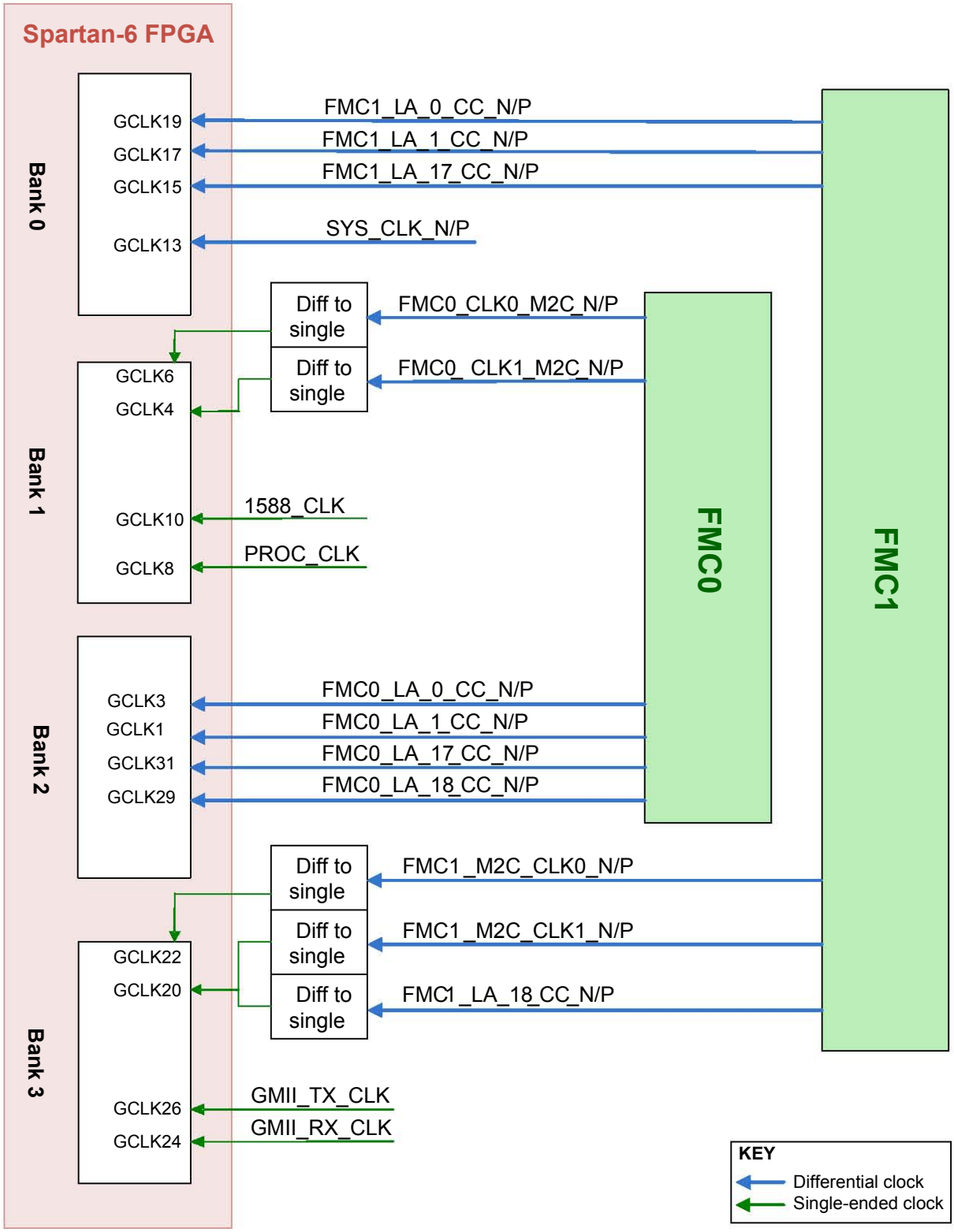


Figure 4.6: Rhino's clock sources, and how the connect to the FPGA

CHAPTER 5

RHINO PCB LAYOUT AND ROUTING

The previous chapter described the schematic-level design of the Rhino hardware. All the components, and the connections between them, were defined in the schematics, and the reasons for the main design decisions were explained in the text. However, this only constitutes half of the design of a PCB. The second half of the design process involves placing those components in appropriate places on a PCB, and routing the traces between them in such a manner that signal integrity is maximised and interference is minimised.

It should be mentioned at this point that the actual routing of the Rhino PCB was outsourced to SunCad Designs, a professional PCB design company [1]. The main reason for this was to accelerate the design process, as they are well experienced in PCB design and hence able to route a PCB very quickly. The result was that Rhino went from concept to working system in just 12 months.

Even though the actual routing of Rhino was outsourced, much other PCB-level work was required to ensure that the routing went as planned. The board stackup and routing rules had to be defined, and the major components had to be placed, before the design could be handed over to SunCad, who were responsible purely for the fairly mechanical job of routing the board. Furthermore, once the board was fully routed, signal integrity simulations had to be performed on all high-speed buses to ensure overshoot, undershoot and crosstalk were within limits. This chapter attempts to describe all these PCB design processes that took place before the design was handed over to SunCad, and after it was received back.

5.1 THE THEORY BEHIND ROUTING HIGH-SPEED DIGITAL SIGNALS

Rhino contains a number of high-speed buses, as listed in Table 5.1. Since one normally only considers transmission line effects when the PCB trace is a significant fraction of a wavelength, most of these buses (with the exception of the CX4 traces) do not seem fast enough to warrant concern over high-frequency effects. The second fastest bus is the DDR3 SDRAM data bus, which runs at 800MHz, giving a wavelength is 375mm, much longer than the PCB itself. However, one must remember that these are digital signals, and not analogue signals. Therefore, it is the switching times (also known as rise/fall times) that determine the high-frequency components of the signal, and not the square-wave frequency [50]. These switching times, and the corresponding effective frequencies and wavelengths, are shown in Table 5.1 too. Since these effective wavelengths are similar to the expected PCB trace lengths, transmission line effects must in fact be considered when routing Rhino.

[1] http://www.suncad.com

Table 5.1: Rhino High-Speed Digital Buses

Sub-System	Bus	Bus Frequency	Switching Time	Eff. Frequency	Eff. Wavelength
DDR3 SDRAM	Data	800MHz	0.3ns	3.3GHz	44mm
	Address	400MHz	0.3ns	3.3GHz	44mm
DDR2 SDRAM	Data	333MHz	0.9ns	1.1GHz	132mm
	Address	166MHz	1.8ns	555MHz	261mm
FMC	Data	400MHz	0.5ns	2GHz	72mm
1Gbps Eth PHY	GMII	125MHz	1ns	1GHz	144mm
CX4 10Gbps Eth	XAUI	3.125GHz	200ps	5GHz	29mm
Processor-FPGA Bus	All	83MHz	2ns	500MHz	289mm

5.1.1 What Happens at High Frequencies?

At high frequencies, traces on a PCB no longer act as simple conductors, but rather exhibit transmission line properties. According to Hall, Hall and McCall [50], this occurs if the signal trace is more than 10% of the wavelength of the signal. At an effective frequency of 1GHz, 10% of the wavelength is just 14mm, shorter than most PCB traces. If the PCB is not carefully designed, the high frequency effects can cause a number of problems which degrade the signal integrity, such as reflections, overshoot, cross-talk and switching noise.

While designing Rhino, much care was taken to match characteristic impedance of traces to input and output impedance of ICs. The characteristic impedance of a PCB trace is defined as the ratio of the voltage and current waves at any point along the trace [50], and is determined by the width and height of the trace and the distance to the nearest ground or power plane. If the characteristic impedance of the trace is not matched to the output impedance of the driver IC, or the input impedance of the receiver IC, reflections occur. The result is a distorted square wave, with overshoot, undershoot, stair-step or ringing characteristics. If the stair-step (seen in Figure 5.1) or ringing (shown in Figure 5.2) is large or continues for a long period of time, it may also cause the receiver to incorrectly detect the logic 1 or 0.

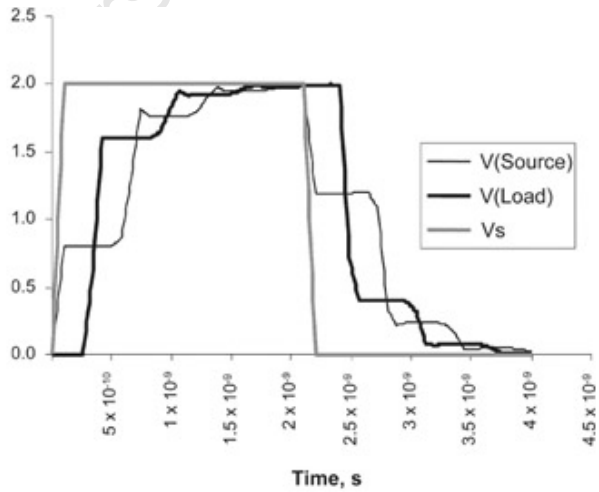


Figure 5.1: The stair-step waveform that occurs if the line impedance is less than the source impedance (Image taken from [50])

However, incorrect impedance matching is not the only cause of signal integrity degradation when dealing with high-speed digital signals. Crosstalk is the coupling of energy from one conductor to another, due to their electromagnetic fields interacting with one another [50]. Crosstalk has two detrimental effects on

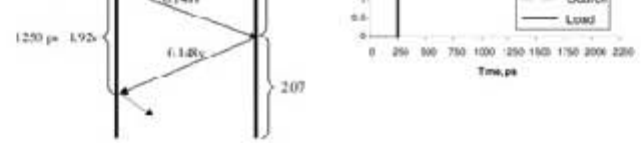


Figure 2.15: Example 2.3: Lattice diagram used to calculate multiple reflections for an overdriven transmission line.

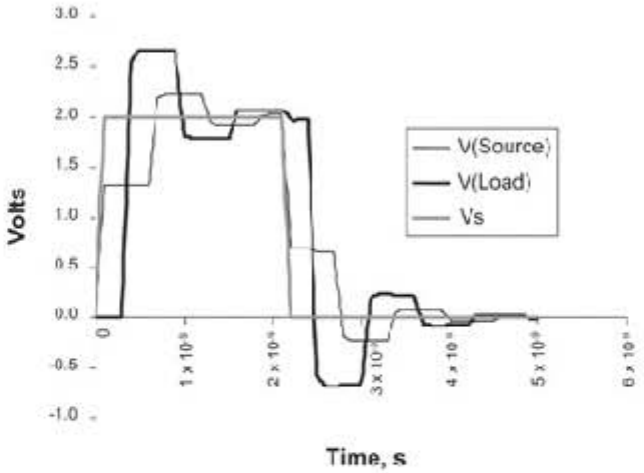


Figure 2.16: Simulation of transmission line system shown in Example 2.3 where the line impedance is greater than the source impedance (overdriven transmission line).

Figure 5.2: The ringing waveform that occurs if the line impedance is greater than the source impedance (Image taken from [50])

neighbouring traces: it induces noise onto those traces and can even change their characteristic impedance. However, these impedance changes are signal pattern dependent. Since the signal patterns are generally unknown, it is impossible to predict the impedance changes. However, one special case is differential signals, where the two signals always have opposite phase. In this case, the exact effective characteristic impedance (called the odd-mode impedance) can be calculated.

A final cause of signal integrity problems in high-speed circuits is split-planes. Split-planes occur when a PCB power layer is divided up into a number of copper fills (or planes), and each of these planes are connected to a different supply voltage. If this split-plane layer is used as a reference plane, and a trace on an adjacent layer traverses the split, then the characteristic impedance of the trace will be changed. Hall, Hall and McCall [50] explain that this is because the return current, which is induced in the power plane as the signal current travels down the trace, is unable to traverse the gap, and is forced to find an alternative path, changing the impedance in an unpredictable way. Therefore, split power planes are never used as reference planes on Rhino.

5.1.2 How can these Problems be Avoided?

There are a number of industry-recognised "good practices" that help to minimise reflections and crosstalk in high speed digital systems. The main rules of thumb that were followed when developing the routing rules for Rhino are described here. If the source of the guideline is not specified, then it was taken from Hall, Hall and McCall's book on high-speed digital design [50]. Due to the additional cost of manufacturing a PCB with impedance-controlled traces, these guidelines were only followed for the high-speed signals on Rhino (those listed in Table 5.1). Slow signals were routed as ordinary, non-impedance-controlled traces.

Match Trace Impedance to Termination Resistors

The output impedance of the driver IC must match the trace characteristic impedance to minimise reflections at the source. Similarly, the receiver termination impedance (either inside or outside the receiver IC package) must match the characteristic impedance to minimise reflections at the load.

In the case of differential pairs, the effective characteristic impedance (i.e. the odd-mode impedance) must be calculated, and the pair terminated with a single resistor between them, with impedance equal to twice the odd-mode impedance of each trace.

Reduce Drive Strength

In cases where exact impedance matching is difficult, source termination resistors can be used to reduce drive strength. This reduces the switching times of the signal, and hence reducing the high-frequency component of the signal. This helps to prevent overshoot, undershoot and crosstalk, but it is not always practical.

Use Differential Signals where Possible

Critical nets, such as clocks, should be differential signals. These nets must be routed close together to ensure strong coupling, which reduces EMI and improves noise immunity.

Avoid Split Planes

Impedance-controlled signals should never cross splits in the reference plane, as this breaks the current return path. If split planes are used, they should be arranged so that no signals on adjacent layers cross the splits. If this cannot be avoided, then capacitors must be used to stitch the split planes together, but this is not ideal.

Ensure Adequate Trace Separation

The general rule of thumb when routing single-ended signals is that the edge-to-edge spacing between the traces should be twice the trace width, in order to keep crosstalk down to an acceptable level [51]. This is commonly known as the *3w* rule, as the centre-to-centre spacing of the traces is three times the trace width (w).

Minimise Distance between Trace and Reference Plane

Another method of minimising crosstalk is to reduce the distance between the trace and the reference plane, by making the dielectric material as thin as possible [52]. This approach is however limited by manufacturing capabilities.

Reduce Coupling between Layers

To avoid coupling between traces on different layers, critical signal layers should be separated by ground or power planes. If this is not possible, then traces on adjacent signal layers should be routed orthogonally to each other.

5.1.3 Calculating Characteristic Impedance

In order to match the characteristic impedances of the traces to the source and load terminations, the impedance of the traces need to be calculated. Different formulas are used for microstrip traces (traces on external layers) and stripline traces (traces on internal layers). Furthermore, if a signal is routed as a differential pair, the coupling between the traces decreases the effective odd-mode impedance, which also needs to be calculated. The formulas shown here are taken from the IPC handbook *IPC-D-317A, Design Guidelines for Electronic Packaging Utilizing High-Speed Techniques* [53], although their derivation has not been shown. All images are courtesy of Mantaro Product Development Services [54].

Note that in all the formulas Z_0 is characteristic impedance; Z_d is differential impedance (which is twice odd-mode impedance); w is trace width; t is trace thickness; h is trace height above/below reference plane; d is distance between inner edges of a differential pair; and ε_r is relative dielectric constant (typically 4.3 for FR-4 PCB).

Impedance of a Microstrip

$$Z_0 = \frac{87}{\sqrt{\varepsilon_r + 1.41}} \ln\left(\frac{5.98h}{0.8w + t}\right)$$

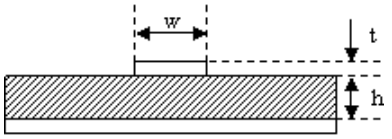


Figure 5.3: Parameters used when calculating the characteristic impedance of a microstrip

Impedance of a Differential Microstrip

$$Z_d = \frac{174}{\sqrt{\varepsilon_r + 1.41}} \ln\left(\frac{5.98h}{0.8w + t}\right)\left(1 - 0.48\exp\left(-0.96\frac{d}{h}\right)\right)$$

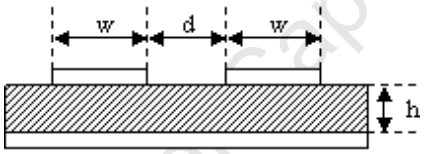


Figure 5.4: Parameters used when calculating the odd-mode impedance of a differential microstrip

Impedance of a Stripline

$$Z_0 = \frac{60}{\sqrt{\varepsilon_r}} \ln\left(\frac{1.9(2h + t)}{0.8w + t}\right)$$

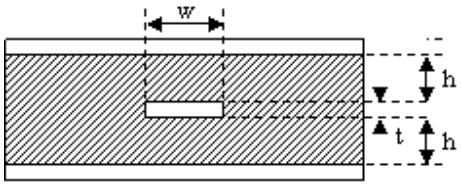


Figure 5.5: Parameters used when calculating the characteristic impedance of a stripline

Impedance of a Differential Stripline

$$Z_d = \frac{120}{\sqrt{\varepsilon_r}} \ln\left(\frac{1.9(2h + t)}{0.8w + t}\right)\left(1 - 0.347\exp\left(-2.9\frac{d}{2h + t}\right)\right)$$

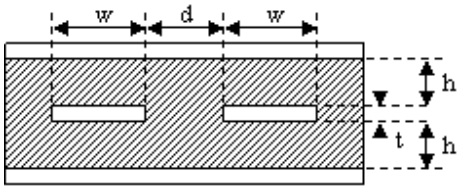


Figure 5.6: Parameters used when calculating the odd-mode impedance of a differential stripline

5.2 PLACEMENT OF MAJOR COMPONENTS

The first step in the PCB design process was to position the major components on the board. These included the FPGA, processor, SDRAM ICs and all the connectors. These components were placed in such a way as to minimise trace length and reduce the crossing-over of traces. Minimising trace length helps to reduce crosstalk on buses, while reducing the number of crossovers helps to lower the layer count, making the PCB cheaper to manufacture.

The PCB was designed to meet the microATX motherboard specification. As a result, the PCB measures 244mm by 244mm, with all the connectors placed along one edge, and the mounting holes in the appropriate positions. The placement of the major components is shown in Figure 5.7.

In this diagram, the PCB has been divided up into 5 different regions: the four edges and the centre. In each of the edge regions, the components may be placed in any order along that edge, but they *must* be placed along the specified edge. Similarly, the components in the centre region may be placed anywhere near the centre of the board. The placement of the major components was specified using regions, rather than absolute positions, in order to give SunCad as much flexibility as possible for the placement and routing, while still ensuring that the placement of connectors was compatible with the proposed enclosure design (which shall be discussed in Chapter 8). Note that all components and the microATX PCB have been drawn to scale, showing their correct relative sizes; however, the relative distances between components is meaningless in this diagram, as the final placement shall be decided by SunCad.

The advantage of this particular arrangement is that it minimises the crossing of PCB traces (which are called nets before they are actually routed). With the FPGA aligned with the centre of the two FMC connectors, the nets between the FMC connectors and the FPGA can be routed with minimal crossing. This is important, as the two FMC buses account for more PCB traces than any other bus. This particular placement of the FPGA (halfway between front and rear, and slightly to the left) also allows for easy routing to the rear connectors. Since all the FPGA connectors (1Gbps Ethernet and CX4) are on the left half of the PCB rear edge, traces can be routed between the FPGA and these connectors without crossing any processor traces.

The placement of the processor similarly has many advantages. By placing it near the FPGA, the processor-FPGA bus traces can be kept very short. Most of the processor connectors are on the right half of the rear edge (HDMI video, three audio jacks, USB host ports, USB On-the-Go, RS-232 header), allowing easy connection from the processor to these connectors without crossing any FPGA nets. Lastly, by placing the FPGA and processor SDRAM ICs along the outer edges of the FPGA and processor respectively, the SDRAM traces can be kept short, and away from FMC and other connector nets. For these exact reasons, SunCad decided to keep the relative positions of components exactly as was suggested, with only the distances between neighbouring components changing for the final layout.

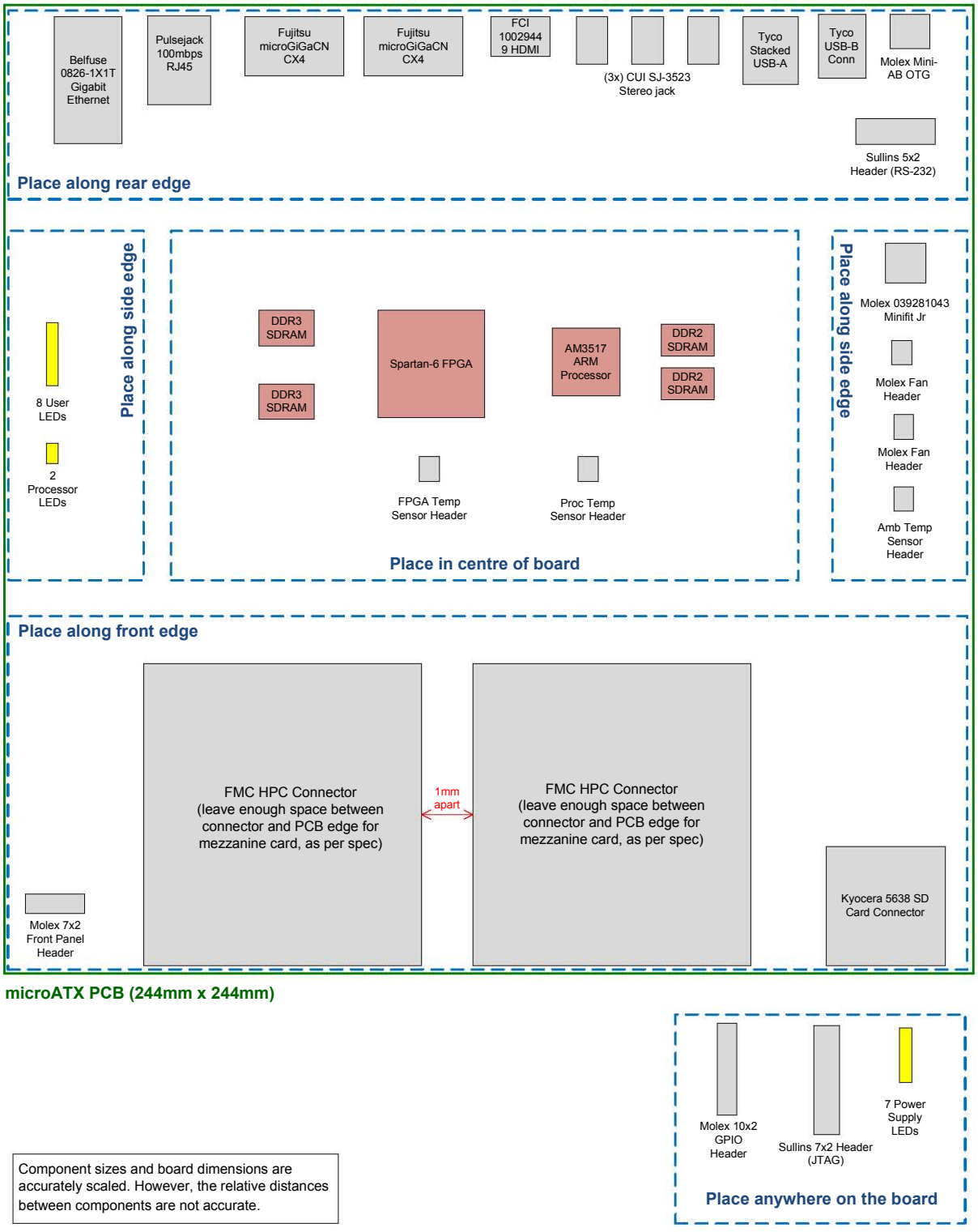


Figure 5.7: Diagram showing placement of major components on the Rhino PCB

5.3 THE PCB STACKUP

The Rhino PCB consists of 16 layers, of which half are ground and power planes. This was done to ensure that every high-speed trace was sandwiched between two ground planes, guaranteeing tight impedance control and

hence good signal integrity. The complete PCB stackup is shown in Figure 5.8. In the stackup diagram, the top and bottom layers are coloured orange, the internal high-speed signal layers are green and the non-critical layers (i.e. layers with no high-speed traces) are yellow. Ground and power planes have been left white. As per standard PCB construction methods, core and prepreg (both shown in grey in the diagram) are alternatively placed between the copper layers. Core is a solid sheet of FR-4 fibreglass with copper stuck to both sides, while prepreg (short for pre-impregnated) is a fibreglass weave with an uncured resin bonding agent that sticks the core layers together, both of which are dielectric materials.

Looking at the "Net Classes to Route on Layer" column in Figure 5.8, it is evident that a number of different types of impedance-controlled traces are used in the Rhino PCB, each with different impedance requirements. To make the design of the PCB simpler, the nets were arranged into different net classes, as shown in Table 5.2.

Table 5.2: Rhino Net Classes for PCB Routing

Class	**Impedance Requirement**	**Nets in the Class**
UNCONTROLLED	No impedance control.	All other nets
50_OHM	50 ohm single-ended impedance.	DDR3 SDRAM address and data. All DDR2 SDRAM nets. GMII bus to 1Gbps Ethernet PHY.
100R_DIFF	100 ohm differential impedance. No single-ended impedance requirement.	DDR3 SDRAM clocks and strobes. CX4 10Gbps Ethernet data traces. Traces to the RJ45 Ethernet connectors and HDMI video connector.
50R_SNGL_100R_DIFF	100 ohm differential impedance AND 50 ohm single-ended impedance. The differential requirement takes priority.	FMC connectors.
90R_DIFF	90 ohm differential impedance. No single-ended impedance requirement.	Data nets (DM and DP) to all USB connectors.

The *50R_SNGL_100R_DIFF* class used for the FMC connectors is an interesting one. Since the FMC data lines can be used as either single-ended or differential signals, the nets are required to have 50 ohm characteristic impedance when operating in single-ended mode and 100 ohm differential impedance when operating in differential mode. However, this is not actually possible. If the traces have 50 ohms impedance when operating in single-ended mode, then when they operate in differential mode, the odd-mode impedance will be less than 50 ohms (due to coupling between the pairs), resulting in a differential impedance of less than 100 ohms. To solve this, the pairs are moved slightly further apart so that the impedance change in differential mode is reduced, and the trace widths and heights are chosen to minimise the *combined* impedance error for *both* single-ended and differential modes.

As has been mentioned, all internal high-speed signal layers are sandwiched between two ground planes. Therefore, to calculate the trace width, spacing and dielectric thickness to give the required characteristic impedances shown in Table 5.2, the formulae given in Section 5.1.3 were used. Note that the microstrip formulae were used for the top and bottom layers, while the stripline formulae were used for the internal layers. The reference planes used in each calculation are shown in the "Reference Plane" column of the stackup diagram (Figure 5.8). Once the trace width and spacing were determined to the nearest mil, the *actual* characteristic impedance was calculated and given in the last column. For example, all differential pairs in the *100R_DIFF* class on layer 3 must be routed as 4mil wide traces with a 7mil gap between the traces. This will give a calculated differential impedance of 101.2 ohms.

er	Description	Copper Thickness	Material Thickness (mil)	Reference Plane	Nets Classes to Route on Layer	Track Width (mil)	I
	Components; 90 ohm pairs	1 oz Copper Foil	1.4	L2	90R_DIFF	4	
					50R_SNGL_100R_DIFF	4	
					100R_DIFF	4	
					50_OHM	5	
	Prepreg 3.5mil		3.5				
	Ground plane	1 oz	1.4				
	Core 6.5mil		6.5				
	100 ohm pairs; 50 ohm single ended	0.5 oz	0.7	L2, L4	50R_SNGL_100R_DIFF	4	
					CX4	4	
					100R_DIFF	4	
					50_OHM	5	
	Prepreg 6.5mil		6.5				
	Ground plane	1 oz	1.4				
	Core 6.5mil		6.5				
	100 ohm pairs; 50 ohm single ended	0.5 oz	0.7	L4, L6	50R_SNGL_100R_DIFF	4	
					100R_DIFF	4	
					50_OHM	5	
	Prepreg 6.5mil		6.5				
	Ground plane	1 oz	1.4				
	Core 3mil		3				
	Power plane	1 oz	1.4				
	Prepreg 3.5mil		3.5				
	Non-critical nets, power nets	1 oz	1.4		UNCONTROLLED	4	
	Core 3.5mil		3.5				
	Non-critical nets, power nets	1 oz	1.4		UNCONTROLLED	4	
	Prepreg 3.5mil		3.5				
	Power plane	1 oz	1.4				
	Core 3mil		3				
	Ground plane	1 oz	1.4				
	Prepreg 6.5mil		6.5				
	100 ohm pairs; 50 ohm single ended	0.5 oz	0.7	L9, L11	50R_SNGL_100R_DIFF	4	
					100R_DIFF	4	
					50_OHM	5	
	Core 6.5mil		6.5				
	Ground plane	1 oz	1.4				
	Prepreg 6.5mil		6.5				
	100 ohm pairs; 50 ohm single ended	0.5 oz	0.7	L11, L13	50R_SNGL_100R_DIFF	4	
					CX4	4	
					100R_DIFF	4	
					50_OHM	5	
	Core 6.5mil		6.5				
	Ground plane	1 oz	1.4				
	Prepreg 3.5mil		3.5				
	Components; solder side; 90 ohm pairs	1 oz Copper Foil	1.4	L13	90R_DIFF	4	
					50R_SNGL_100R_DIFF	4	
					100R_DIFF	4	
					50_OHM	5	
	Total board thickness (mils)		**95.1**				

Figure 5.8: Stackup of the Rhino PCB

There is however scope for cost reduction in this stackup. Layer 6, a ground plane, has been placed adjacent to layer 7, a power plane. The reason that these two reference planes were placed on adjacent layers is that layer 7 is a split power-plane. Since a number of the high-speed signals on layer 5 cross the split on the plane, there might have been signal integrity problems. To prevent this, a ground plane was inserted between layers 5 and 7. The same goes for the split power-plane on layer 10 and the high-speed signals on layer 12 (here, the ground plane was inserted on layer 11).

However, if some of the power planes on layers 7 and 10 were moved to non-critical layers 8 and 9, and one or two of the existing ground plane layers were converted to solid power plane layers, it would be possible to convert the split power-planes on layers 7 and 10 to solid power planes. This would allow layers 7 and 10 to be used as reference planes for signal layers 5 and 12, allowing layers 6 and 11 to be removed from the stackup. Based on the quotes that were received for the Rhino PCB, reducing the layer count from 16 down to 14 layers should reduce the PCB manufacturing cost by at least 20%.

5.4 Defining the Routing Rules for Rhino

Before the board was routed by SunCad Designs, a number of trace width, length and spacing rules were defined. This was done to ensure minimal crosstalk, minimal volt drop on the power lines and a no cutting-edge manufacturing requirements (which can be expensive).

5.4.1 Minimum Trace Width

All traces must be 4 mils or wider. If a trace is less than 4 mils wide, the manufacturing costs increase dramatically, due to the difficulty in manufacturing such thin traces. The exact width of impedance-controlled traces are specified in the stackup diagram, and the exact width of the power nets is discussed in the next section. For all non-impedance-controlled and non-power nets, 4 mils was chosen as the standard trace width.

5.4.2 Widths of Power-Carrying Traces

Although all the power supply rails were implemented as split planes on layers 7 and 10 of the PCB, sometimes standard nets are required to carry substantial amounts of current (e.g. the output of the fan controllers). Therefore, all nets in the Rhino PCB that carry more than 50mA of current were defined as power nets. The acceptable voltage drop for each power net was then determined (usually 10mV). From this, the required trace width for the power net could be calculated, by ensuring that:
a.) the temperature rise along the trace did not exceed 10°C
b.) the voltage drop along the trace was within the defined limits

The voltage drop and temperature rise along each trace were determined from the relevant plots in *IPC-2221A, Generic Standard on Printed Board Design* [55].

5.4.3 Inter-trace Spacing

Figure 5.9 shows a typical setup of traces on a PCB. The green lines indicate differential pairs, while the red lines indicate single-ended nets.

The dimensions in the diagram can be defined as follows:
A = Distance between the P and N traces in a single differential pair
B = Distance between adjacent differential pairs
C = Distance between a differential pair and a neighbouring single-ended net
D = Distance between adjacent single-ended nets

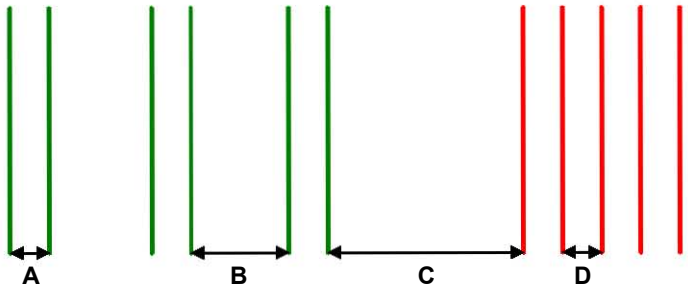


Figure 5.9: Definition of the inter-trace dimensions (A, B, C and D) on a PCB

Note that all these dimensions refer to inside-edge to inside-edge spacing between traces, and not centre to centre spacing. The following rules were defined for the inter-trace spacing on Rhino. These follow the industry "best practices" that were discussed earlier to minimise the crosstalk between nets:
B = 2 x A
C = 3 x A
D = 4mil (for slow net) or 8mil (for high-speed net)

The high-speed nets for dimension D are those that were identified in Table 5.1 at the beginning of this chapter.

5.4.4 **Trace Length Matching**

A number of length-matching rules were defined for synchronous busses to ensure that these buses met the necessary timing constraints. These rules stated that the average trace length of the data bus must match the length of the clock trace to within certain limits. Rules were also defined that limited the length difference between the longest and shortest data trace. These rules were defined for the DDR2 SDRAM, DDR3 SDRAM, 1Gbps Ethernet GMII and FMC buses. For brevity's sake, the actual rules will not be given here; however most rules required that the length of the data traces be within 300 mils of the length of their clock trace.

5.5 VERIFICATION OF PCB LAYOUT AND ROUTING

As has been mentioned, the placement of the majority of the components and the actual routing of the Rhino PCB was outsourced to SunCad Designs. The final routed PCB has been included on the attached CD. See Appendix D for instructions on how to view this file.

However, all work was checked by the author to ensure that it complied with the defined rules and specifications. Although most of this checking was done manually, two tools were used to automate part of the process. Firstly, 3D models of all components were created to check for footprint problems and collisions between components. Secondly, the HyperLynx simulation tool was used to perform signal integrity simulations on all high-speed traces. These automated PCB verification processes are described in more detail below.

5.5.1 **3D Models to Check Footprints**

Since all footprints were drawn by hand from the datasheet, the possibility of errors in the footprints always existed. In order to minimise this risk, Wiebke Toussaint from the UCT Mechanical Engineering Department was hired to create 3D models of all components used on the Rhino PCB, using Pro Engineer. Since Altium Designer allows one to import these 3D models directly into the footprint file, the 3D model of the component could be placed directly onto the footprint, ensuring that the pads and holes were all in the correct positions. An example of one such 3D model, the Fujitsu CX4 connector, is given in Figure 5.10. Note that the pins on the connector are silver in colour, while the pads on the PCB are copper in colour.

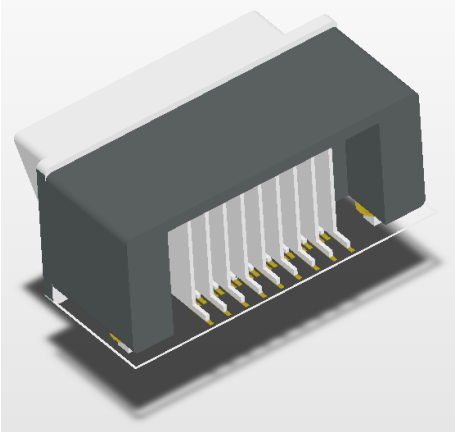


Figure 5.10: 3D model of the Fujitsu CX4 connector, as shown from the rear

5.5.2 Signal-Integrity Simulations

Mentor Graphics HyperLynx is a signal integrity simulation tool that allows users to check for signal integrity problems on a PCB before it is manufactured. It is able to calculate the reflections on a trace due to impedance mismatches, as well as determine the crosstalk from neighbouring traces by performing a field simulation. Furthermore, HyperLynx is able to plot the actual signal waveform as seen by the receiver, allowing users to easily measure overshoot, undershoot, ringing and crosstalk.

In order to check for signal integrity problems, the Rhino PCB file, with all the components placed and the traces routed, was imported into HyperLynx. IBIS models were used to describe the input and output buffers of each IC that was involved in the simulations. The IBIS models specify attributes such as high/low output levels, slew rate, input/output impedance, and input thresholds. Simple models were also used to describe any external termination resistors. By providing HyperLynx with the exact positions of components, the geometry of the traces, the board stackup and the input/output characteristics of the ICs, HyperLynx is able to calculate the impedance of each trace, the reflections on those traces, and the interference between traces.

In the case of Rhino, only the high-speed traces mentioned at the beginning of this chapter were simulated. This consists of all DDR2 SDRAM traces, all DDR3 SDRAM traces, the FMC data and clock traces, and the GMII traces for the 1Gbps Ethernet PHY. The CX4 traces (for 10Gbps Ethernet) could not be simulated, as standard IBIS models are not well suited for multi-gigahertz signals. The results of these signal integrity simulations are shown below. Note that all plots show the signal waveform at the *pin of the receiver IC*.

DDR2 SDRAM

All the DDR2 SDRAM traces, for both SDRAM ICs, were simulated together, as the two ICs share common address and control lines. The results of these simulations are shown in Table 5.3 below. Note that the safe overshoot and undershoot limits were taken from the datasheet for the Winbond DDR2 SDRAM IC.

Table 5.3: Signal Integrity Simulation Results for the DDR2 SDRAM

Signal Group	**Normal Voltage Range**	**Avg Simulated Overshoot**	**Max Simulated Overshoot**	**Max Acceptable Overshoot/Undershoot**
Address & Control	0V – 1.8V	20mV	72mV	900mV
Data	0V – 1.8V	60mV	169mV	900mV

As is quite clear, the maximum overshoot on any of the DDR2 SDRAM lines is 169mV, well below the 900mV specified limit. A full crosstalk analysis was also performed, the results of which are shown in Table 5.4. Note that for the crosstalk analysis, HyperLynx first does a quick analysis to determine if any nets have "noticeable" crosstalk. If so, HyperLynx will then perform a detailed simulation to quantify that crosstalk. The maximum crosstalk in this case is only 72mV, which is not nearly enough to cause a false signal detection.

Table 5.4: Crosstalk Simulation Results for the DDR2 SDRAM

Number of nets with "noticeable" crosstalk	38 out of 70
Average crosstalk of these "noticeable" nets	35mV
Maximum crosstalk of these nets	72mV

Average case and worst case plots for each of the signal groups that were simulated (Address & Control and Data) are given below. It is important to remember that these plots show the actual signal at the *receiver*. Figure 5.11 shows an average case for the address and control signals, while Figure 5.12 shows the signal with the worst overshoot. Similarly, Figure 5.13 shows an average case for the data bus, while Figure 5.14 shows the data signal with the worst overshoot/undershoot. Note that the dashed blue lines in each plot represent the receiver high/low input thresholds.

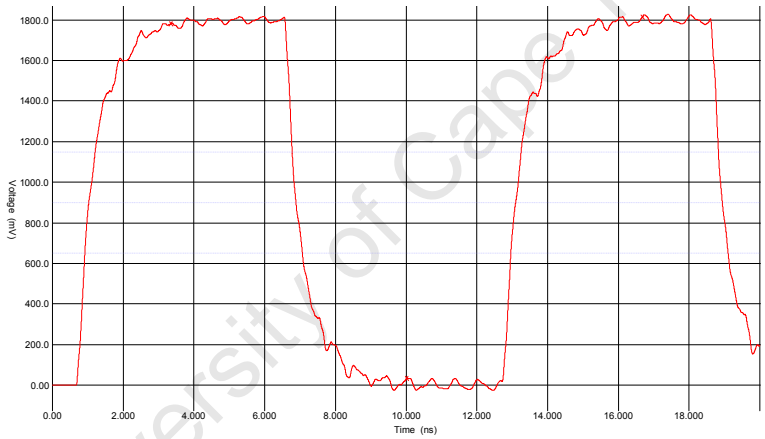


Figure 5.11: Signal integrity simulation for net A1 on the DDR2 bus. The line is being toggled at 166MHz, and exhibiting 16mV overshoot and 28mV undershoot (TYPICAL CASE).

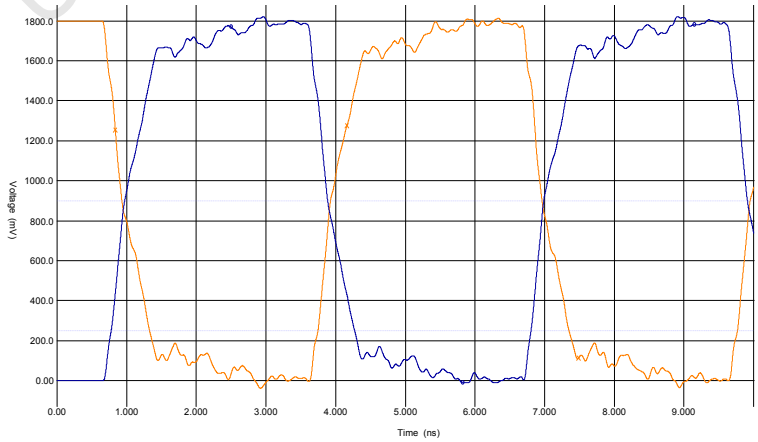


Figure 5.12: Signal integrity simulation for nets CK_P and CK_N on the DDR2 bus. The clocks are running at 166MHz, and exhibiting 72mV overshoot and 52mV undershoot (WORST CASE).

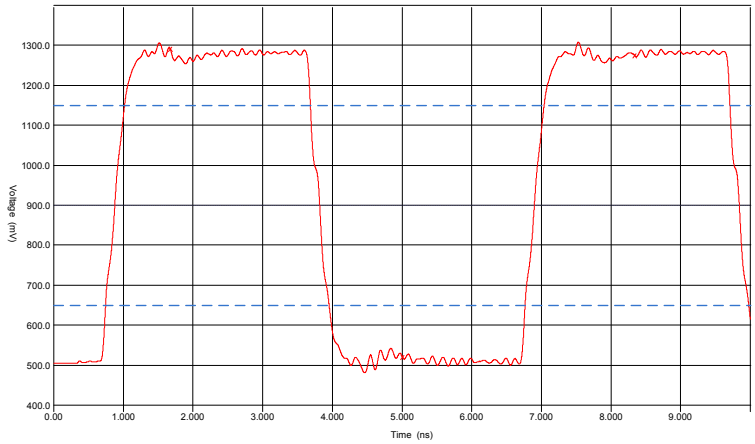


Figure 5.13: Signal integrity simulation for the DQ2 net on the DDR2_1 bus. The data line is toggling at 333MHz, and exhibiting 29mV overshoot and 37mV undershoot (TYPICAL CASE).

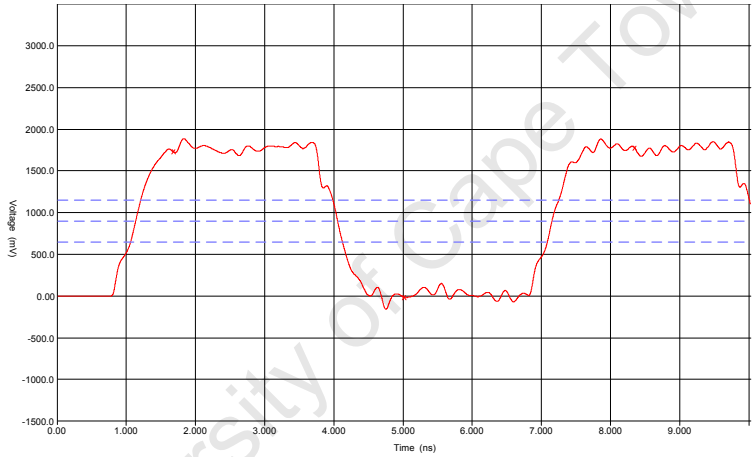


Figure 5.14: Signal integrity simulation for the DQ8 net on the DDR2_0 bus. The data line is toggling at 333MHz, and exhibiting 142mV overshoot and 169mV undershoot (WORST CASE).

In all these plots, the signals are switching cleanly with only minor ringing. The reason that signal DQ2 in Figure 5.13 does not reach the 0V and 1.8V rails is due to the volt drop across the 22 ohm series resistor. This resistor was recommended by the AM3517 datasheet to reduce EMI [37].

DDR3 SDRAM

The DDR3 SDRAM traces were simulated separately for each IC, as they are routed separately. The results of these simulations are shown in Table 5.5 below. Note that the safe overshoot and undershoot limits were taken from the datasheet for the Micron DDR3 SDRAM IC.

The maximum overshoot on any of the DDR3 SDRAM lines is 224mV, well below the 400mV specified limit. A full crosstalk analysis was also performed on both SDRAM ICs *together* and the results are shown in Table 5.6. The maximum crosstalk for the DDR3 SDRAM is only 57mV, which is not nearly enough to cause a false signal detection.

It is clear from Table 5.5 that the results are very similar for both SDRAM ICs. Therefore, plots are only

Table 5.5: Signal Integrity Simulation Results for the DDR3 SDRAM

Signal Group	**Normal Voltage Range**	**Avg Simulated Overshoot**	**Max Simulated Overshoot**	**Max Acceptable Overshoot/Undershoot**
DDR3_0 Address & Control	0V – 1.5V	150mV	220mV	400mV
DDR3_0 Data	0V – 1.5V	40mV	63mV	400mV
DDR3_1 Address & Control	0V – 1.5V	150mV	224mV	400mV
DDR3_1 Data	0V – 1.5V	60mV	122mV	400mV

Table 5.6: Crosstalk Simulation Results for the DDR3 SDRAM

Number of nets with “noticeable” crosstalk	46 out of 86
Average crosstalk of these “noticeable” nets	30mV
Maximum crosstalk of these nets	57mV

shown for DDR3_1. Again, average case and worst case plots are given for each of the signal groups that were simulated (Address & Control and Data). Figure 5.15 shows an average case for the address and control signals, while Figure 5.16 shows the signal with the worst overshoot. Similarly, Figure 5.17 shows an average case for the data bus, while Figure 5.18 shows the data signal with the worst overshoot/undershoot. Note that the dashed blue lines in each plot represent the receiver high/low input thresholds.

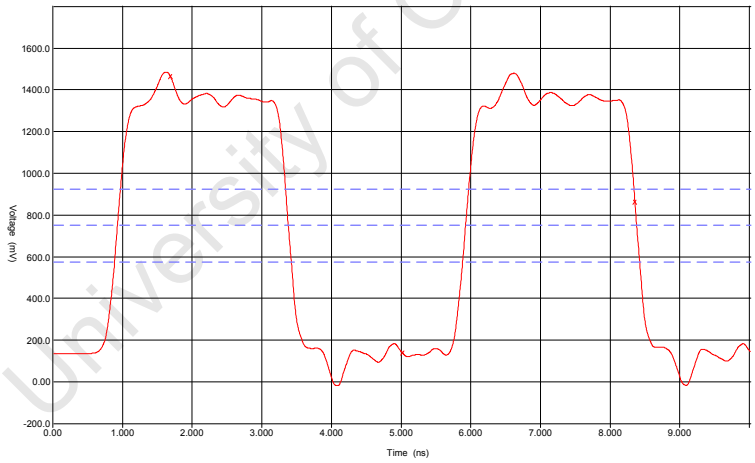


Figure 5.15: Signal integrity simulation for net A4 on the DDR3_1 bus. The line is being toggled at 400MHz, and exhibiting 139mV overshoot and 164mV undershoot (TYPICAL CASE).

In all these plots, the signals are switching cleanly with only minor ringing. As was explained earlier, this ringing is caused by the trace impedance being slightly larger than the source impedance.

FMC Connectors

In order to simulate the FMC traces, a “signal source” was required. In keeping with the use-case scenarios from Chapter 1, a Texas Instruments 500MS/s LVDS-out ADC (ADS5463) was used as the source. Therefore, the FMC connector was replaced in the simulations by a typical ADC that one can expect to find on an FMC card connected to Rhino. Both FMC connectors were simulated together, as their traces cross over, and the results are shown in Table 5.7. Note that on-die termination was enabled on the FPGA for all simulations.

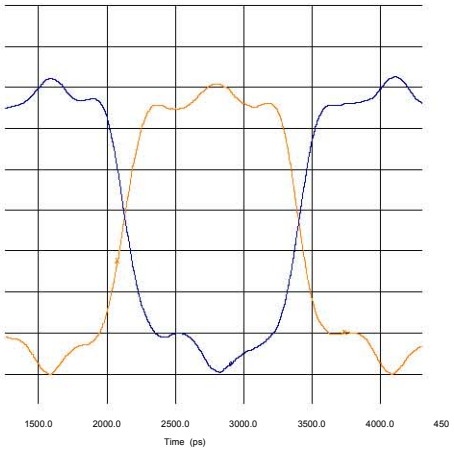


Figure 5.16: Signal integrity simulation for nets CK_P and CK_N on the DDR3_1 bus. The clocks are running at 400MHz, and exhibiting 224mV overshoot and 224mV undershoot (WORST CASE).

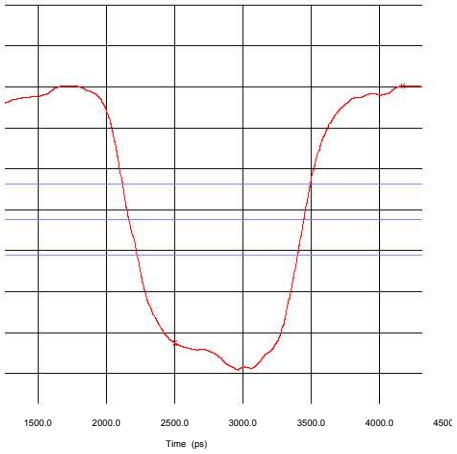


Figure 5.17: Signal integrity simulation for the DQ3 net on the DDR3_1 bus. The data line is toggling at 800MHz, and exhibiting 57mV overshoot and 77mV undershoot (TYPICAL CASE).

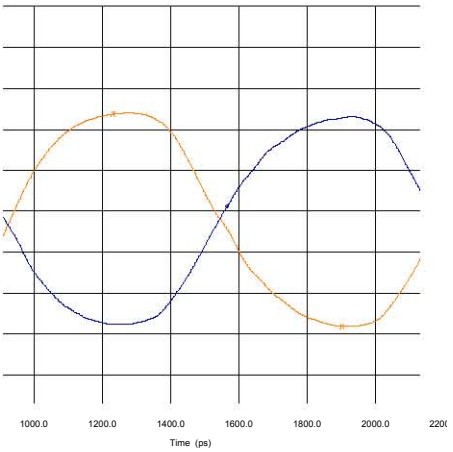


Figure 5.18: Signal integrity simulation for the LDQS_N and LDQS_P nets on the DDR3_1 bus. The data strobe is clocking at 800MHz, and exhibiting 120mV overshoot and 122mV undershoot (WORST CASE).

Table 5.7: Signal Integrity Simulation Results for the FMC Signals

Signal Group	Normal Voltage Range	Avg Simulated Overshoot	Max Simulated Overshoot	Max Acceptable Overshoot/Undershoot
FMCx_CLKx_M2C (single-ended)	0V – 3.3V	80mV	91mV	500mV
FMCx_LA_P/Nxx and FMCx_CLKx_M2C_P/N	1V – 1.6V	5mV	8mV	60mV
FMCx_ZDOK_P/Nxx	1V – 1.6V	6mV	37mV	60mV

The overshoot on the single-ended clock lines (FMCx_CLKx_M2C) was initially very high. The source series termination resistors on the output of each MAX9111 LVDS-to-CMOS converter was then changed from 22 ohms to 49.9 ohms, and the signal integrity improved drastically. One will also notice that the overshoot on the ZDOK differential lines is higher than the overshoot on the standard data (LA) differential lines. This is because the ZDOK lines are connected to an FPGA bank that does not support on-die termination; hence external 100 ohm termination resistors must be used, resulting in higher overshoot. However, the overshoot is still within specification.

A full crosstalk analysis was also performed on both connectors together, and the results are shown in Table 5.8. The cross talk for the data and ZDOK lines is extremely low, as these lines are routed as differential pairs, and hence have high crosstalk immunity.

Table 5.8: Crosstalk Simulation Results for the FMC Signals

Signal Group	Average Crosstalk	Maximum Crosstalk
FMCx_CLKx_M2C (single-ended)	20mV	56mV
FMCx_LA_P/Nxx and FMCx_CLKx_M2C_P/N	None	2mV
FMCx_ZDOK_P/Nxx	None	1mV

Since the average case and worst case are very similar for the FMC signal integrity simulations, plots will only be given of the "worst" quality signals. Figure 5.19 shows the single-ended FMC clock with the worst overshoot; Figure 5.20 shows the differential data signal with the worst overshoot; and Figure 5.21 shows the ZDOK data signal with the worst overshoot.

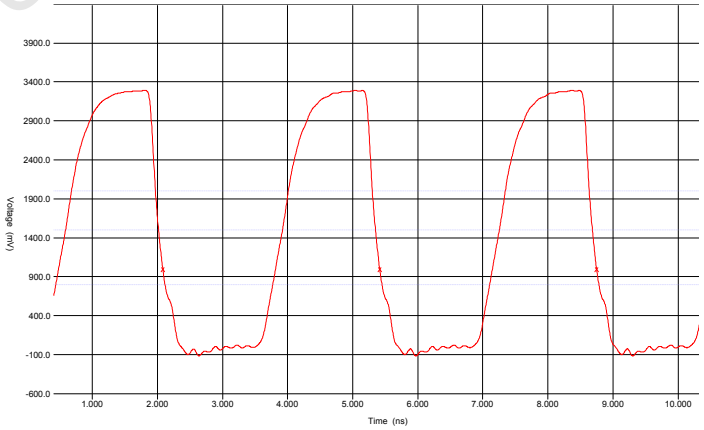


Figure 5.19: Signal integrity simulation for the FMC1_CLK0_M2C single-ended net. The clock is running at 300MHz, and experiencing 11mV overshoot and 91mV undershoot (WORST CASE).

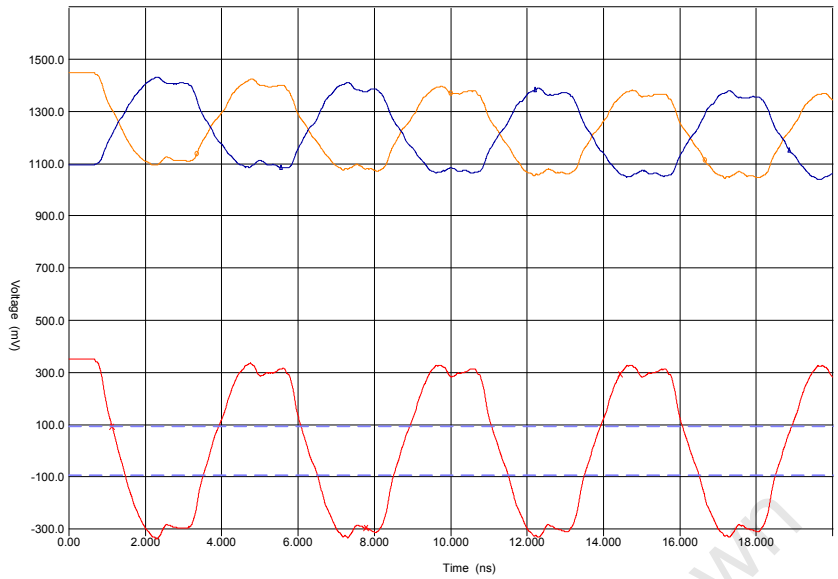


Figure 5.20: Signal integrity simulation for the FMC0_LA_P/N21 net. The positive and negative signals are shown (in orange and blue), as well as the difference between these two signals (in red). The signal is toggling at 400MHz, and experiencing 8mV overshoot and 8mV undershoot (WORST CASE).

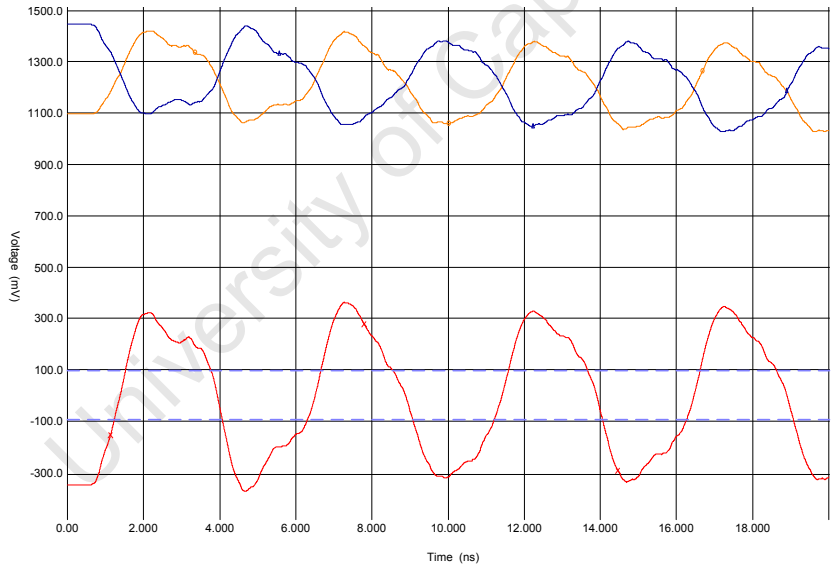


Figure 5.21: Signal integrity simulation for the FMC0_ZDOK_P/N0 net. The positive and negative signals are shown (in orange and blue), as well as the difference between these two signals (in red). The signal is toggling at 400MHz, and experiencing 37mV overshoot and 37mV undershoot (WORST CASE).

The LA and ZDOK traces are toggled at 400MHz, equivalent to alternating pattern of 1s and 0s being transmitted at 400Mbps. This was the maximum I/O data rate specified in Chapter 1. The single-ended clock signal, FMC1_CLK0_M2C, is running at 300MHz, allowing a maximum double datarate of 600Mbps.

The poor waveform shape of the ZDOK lines is due to *external* termination resistors being used. Fortunately, these lines are only used for the FMC-to-ZDOK+ adaptor, and hence will seldom be used. However, all traces simulated here do cross the receiver detection thresholds without oscillation, ensuring correct signal detection.

GMII Bus for 1Gbps Ethernet PHY

The traces for the GMII bus between the FPGA and 1Gbps Ethernet PHY were simulated. The results are shown in Table 5.9. Similarly, the results of the crosstalk simulation are shown in Table 5.10

Table 5.9: Signal Integrity Simulation Results for the GMII Signals

Signal Group	**Normal Voltage Range**	**Avg Simulated Overshoot**	**Max Simulated Overshoot**	**Max Acceptable Overshoot/Undershoot**
All GMII signals	0V – 2.5V	150mV	366mV	500mV

Table 5.10: Crosstalk Simulation Results for the GMII Signals

Signal Group	**Average Crosstalk**	**Maximum Crosstalk**
All GMII signals	110mV	157mV

The overshoot on all the GMII buses was initially very high (over 500mV). This was improved by running the simulations with a number of different source series termination resistors, to determine which resistor values provided the best impedance matching. However, even after tweaking the termination resistor values, the overshoot/undershoot is still higher than what one would expect. This is due to the very long traces (approximately 200mm) between the FPGA and the Ethernet PHY, which results in strong coupling between the parallel traces. Fortunately, the overshoot/undershoot and crosstalk is still within specification.

An average case GMII signal is shown in Figure 5.22, while the signal with the largest overshoot is shown in Figure 5.23. This worst-case signal (TXD2) is experiencing the "stair-step" effect that was described earlier in this chapter, due to reflections on the line.

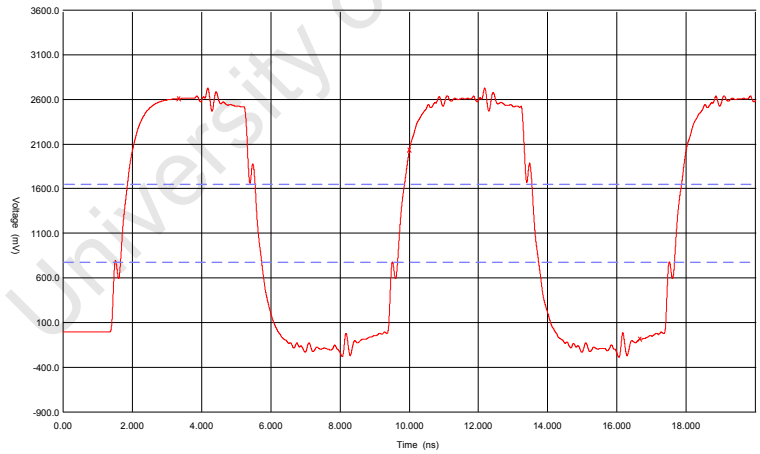


Figure 5.22: Signal integrity simulation for the GMII_RX_CLK trace. The clock is running at 125MHz, and experiencing 82mV overshoot and 145mV undershoot (TYPICAL CASE).

5.5.3 Full 3D Model of the Rhino PCB

Once the board had been fully routed by SunCad Designs and the high speed traces had been simulated, a complete 3D model of the board was built. This final verification step was completely automated by Altium Designer. Since component 3D models of components had already been imported into every footprint file, and all the component footprints had been placed on the PCB, Altium Designer had sufficient information to build a complete 3D model of the populated PCB. This 3D model is shown in Figure 5.24. This provided a simple mechanism to check for collisions between neighbouring components.

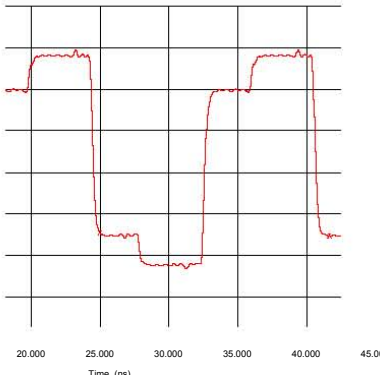


Figure 5.23: Signal integrity simulation for the TXD2 signal on the GMII bus. The signal is toggling at 125MHz, and experiencing 259mV overshoot and 366mV undershoot (WORST CASE).

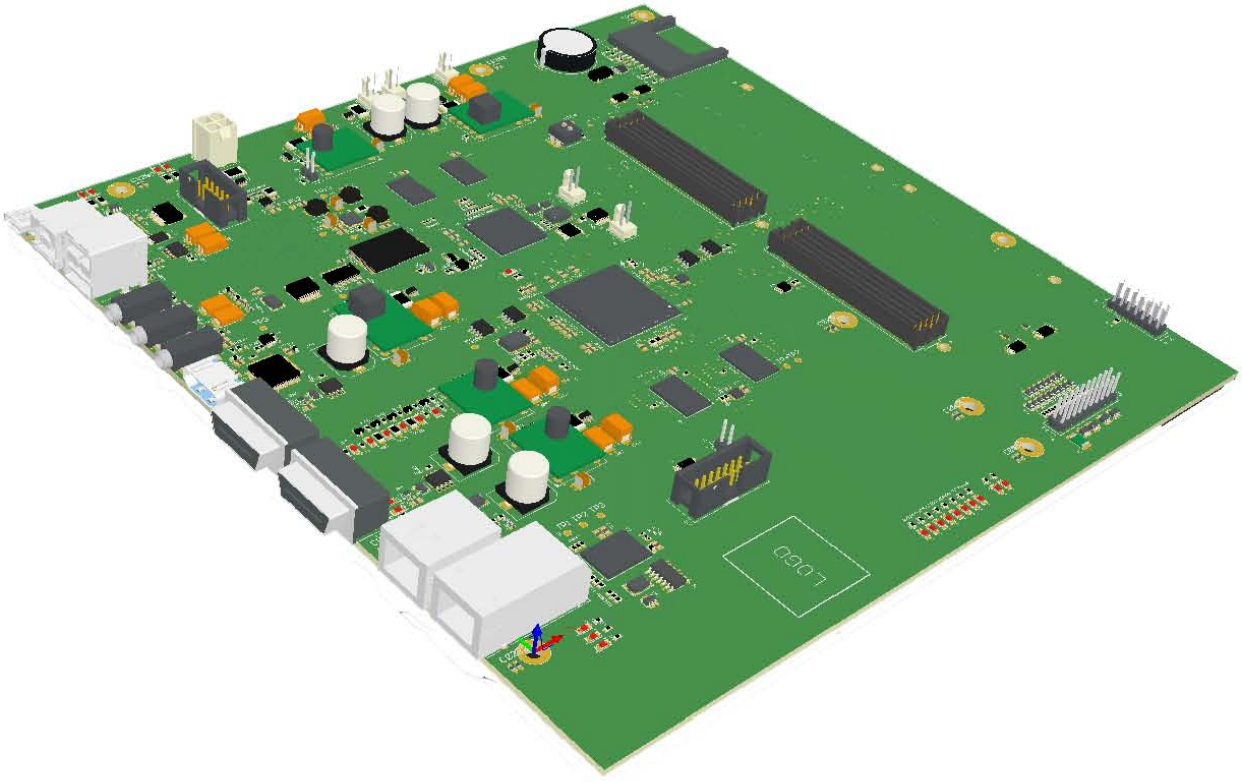


Figure 5.24: 3D model of the complete Rhino PCB

This concludes the design of the hardware for Rhino. Over the past few chapters, the requirements have been determined, the architecture defined, and the design fully specified at the schematic level. Furthermore, this chapter has described the PCB-level design of Rhino, which has involved defining routing rules and PCB stackup, placing the major components on the board, and verifying the final placement and routing using signal integrity simulations. Rhino is therefore now at a point where it is ready for manufacture.

The next chapter describes the software that was written to test the hardware once it was built, and the following chapter gives the results of these tests.

CHAPTER 6

DEVELOPMENT OF TOOLS FOR VERIFICATION OF HARDWARE DESIGN

As with any complex system, the functionality and performance of Rhino needs to be checked and verified at all stages of the design process. Therefore, even though the schematic-level design of Rhino was reviewed many times, and the PCB layout was simulated in HyperLynx, the complete system still needs to be functionally verified once it has been built. Both hardware and software tools were developed to aid this verification.

The hardware tools took the form of a test rig. This rig contained the basic hardware infrastructure needed to power up a Rhino board, peripherals that would typically be connected to Rhino in the field, and debugging equipment. The software tools, on the other hand, included the bootloaders required to boot the processor and correctly configure its peripherals, the application software that functionally tested the peripherals, and the gateware that was loaded onto the FPGA to check its peripherals. These hardware and software tools are described in this chapter.

6.1 THE RHINO TEST RIG

In order to power-up the first prototype Rhino boards, and fully test their hardware, a test rig was required. This rig needed to provide the following infrastructure:

- 12V DC power for the board
- fans and temperature sensors
- power button, reset button and power LED
- JTAG programmer for configuring the FPGA before BORPH is up and running
- a debug board that allows the FPGA to communicate with a PC over RS-232

Such a rig was therefore designed in Pro Engineer and manufactured out of perspex. See Figure 6.1 for the Pro Engineer 3D model of the rig. The test rig consists of a large base plate with standoffs for mounting the Rhino PCB, a front panel on which the fans and power/reset buttons are mounted, and a back panel with cutouts for the connectors. The top and sides are open to provide easy access to the board.

Figure 6.2 shows the test rig after it was built and all the fans, switches and debugging hardware had been mounted. The Rhino PCB sits in the open space in the middle of the test rig, on the brass standoffs. The

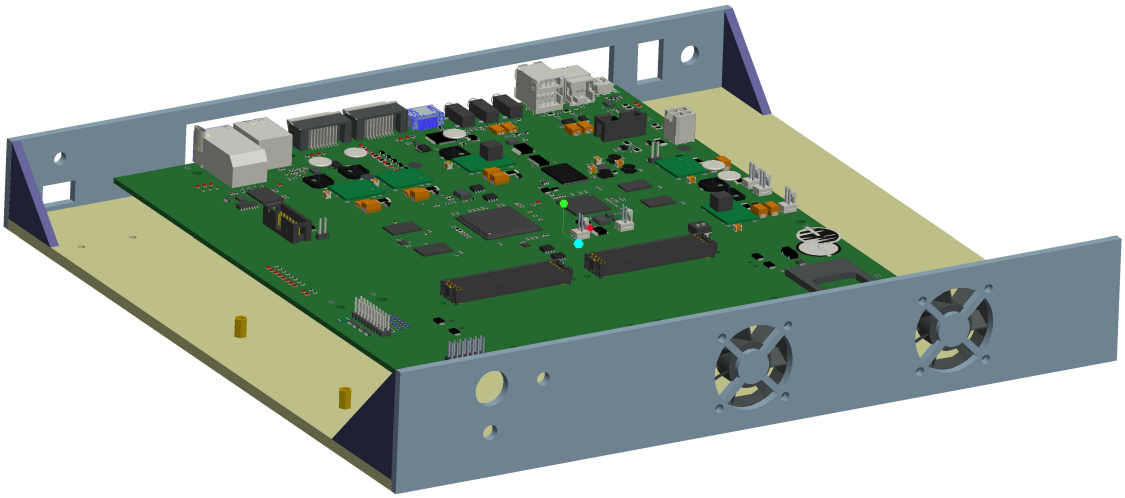


Figure 6.1: 3D model of the Rhino test rig

red power button, black reset button and green power LED are mounted on the left of the front panel, while the two 40mm fans are mounted on the right. The buttons and LED connect to the front panel header on the Rhino board, and the fans obviously connect to the two fan connectors. Power is supplied to the rig by a 230VAC-to-12VDC "power brick", which connects to a jack on the back panel.

A RS-232 debug board is mounted on the left of the test-rig. This board connects to the FPGA GPIO header via a ribbon cable. It contains toggle switches and an LED, connected to the GPIO lines, to aid debugging. It also contains a RS-232 level converter, whose CMOS-level receive/transmit lines connect to the GPIO header and whose RS-232-level lines are wired to a jack on the back panel. A serial cable connects this jack to a serial port on a PC, allowing serial communication between the FPGA and the PC.

Provision has also been made for a JTAG programmer to connect to the FPGA's JTAG port (the JTAG cables are shown in the top left corner of the photograph). This allows the FPGA to be programmed before BORPH is up and running on the processor. Although not shown here, a Xilinx FMC debug card plugs onto the Rhino PCB during testing, allowing the performance of the FMC interface to be verified.

6.2 Processor Bootloader Software

Booting the AM3517 processor is a multi-stage process, as shown in Figure 6.3. The first stage is performed by the ROM code on the processor itself, while the remaining three stages are carried out by the two bootloaders, X-Loader and U-Boot, and the operating system. Note that the boot ROM is stored on the processor die itself and cannot be changed, while the other three are stored on flash memory, an SD card or on the network.

The internal bootloader runs as soon as the processor is powered. It reads the boot-select switches to determine the required boot source. If the Ethernet port has been selected, the processor broadcasts boot image requests over the local network. If the SD card interface has been selected, the processor scans the first sector of the FAT table for a file named "MLO". Lastly, if the flash memory is the selected source, the processor scans the first four NAND memory blocks for the start of a valid boot image. In all these cases, once the ROM bootloader has found the boot image (i.e. the X-Loader file), it copies it into on-chip SRAM and executes it.

The main function of the X-Loader is to configure and initialise the DDR2 SDRAM. Once the DDR2 SDRAM has been initialised, the X-Loader reads the boot-select switches again, and searches either the SD card or the NAND flash for the U-Boot image. Once found, this file is copied into external DDR2 SDRAM and executed.

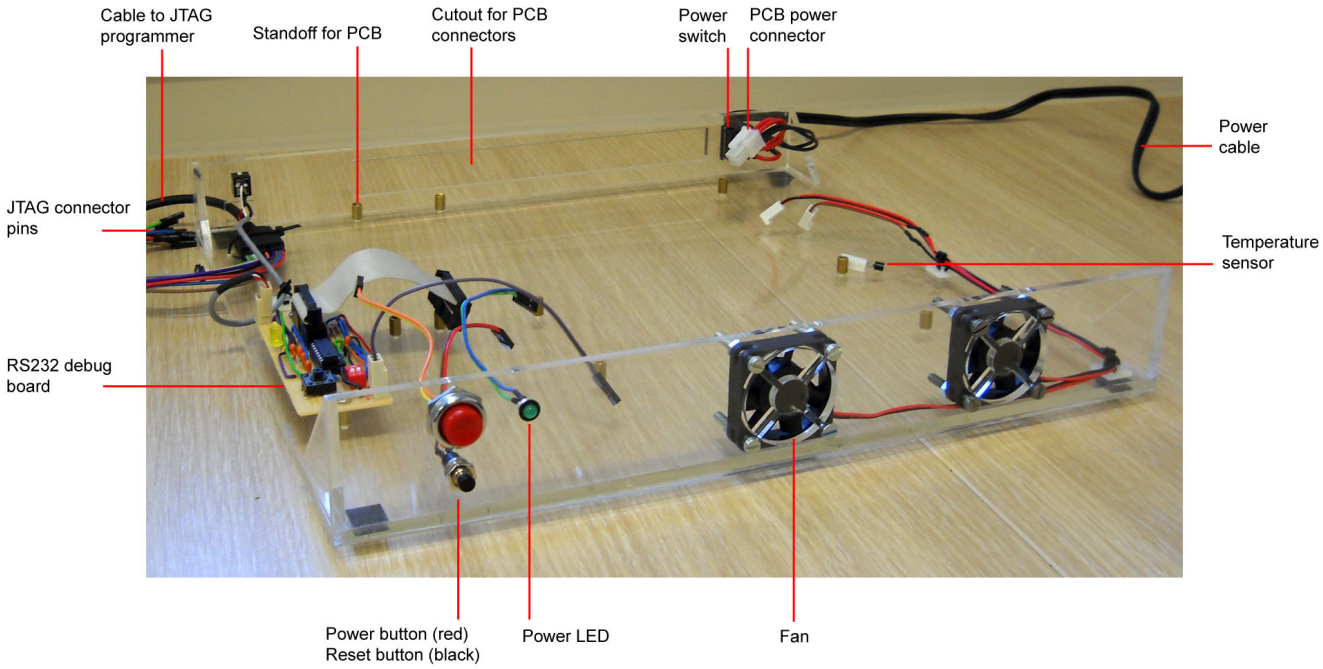


Figure 6.2: Photo of the completed Rhino test rig

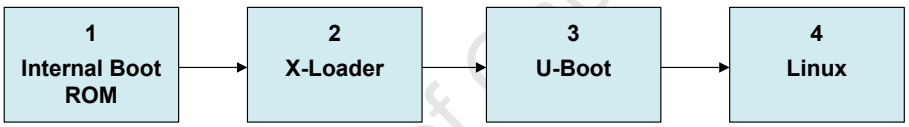


Figure 6.3: The AM3517 boot process

U-Boot then configures the peripheral sub-systems on the processor, such as the SPI and I^2C ports, the GPIO pins, and the video output interface. U-Boot uses environment variables to specify location of the operating system (SD card, NAND flash memory, USB flash drive, or FTP server), the Rhino board's network settings, and any command line parameters that should be passed to the operating system. The operating system is then copied into the SDRAM and executed. Although both Linux and Windows CE can run on the AM3517 processor, Rhino will always use Linux (BORPH).

The bootloaders are required as the internal boot ROM is unable to load the operating system directly. This is because the operating system image is typically quite a few megabytes, too large to fit into the processor's 64kB of on-chip SRAM. Since the boot ROM is fixed, it has no way of knowing the SDRAM settings for the particular board and hence cannot initialise the DDR2 SDRAM. Therefore bootloaders must be used to configure and initialise the SDRAM before the operating system is loaded. The reason that both X-Loader and U-Boot are required is that U-Boot is also too big to fit into SRAM (it is approximately 200kB); hence X-Loader must first run to initialise the DDR2 SDRAM, before calling U-Boot to initialise all the peripherals and load the operating system.

6.2.1 X-Loader

The X-Loader is the second-stage bootloader, and performs the following operations:

1. Initialise all the on-chip clocks and PLLs (phase-locked loops) to the correct frequencies.

2. Each of the processor I/O pins can perform a number of different functions. Each pin has a primary mode (such as a data line for the SDRAM), but can also have a number of secondary modes (such as a GPIO). The X-Loader therefore sets the SDRAM, GPMC (which is used for the NAND flash memory) and SD card pins to their correct primary mode.

3. Configure the SDRAM timing settings, and initialise the DDR2 SDRAM ICs.

4. Search the SD card or NAND flash memory for a file named "u-boot.bin", copy it into SDRAM, and execute it.

The X-Loader port for Rhino was based on the port for the AM3517 development board. This meant that the X-Loader did not need to be ported to a new processor, just to a new board. Therefore, in order to get this port running on Rhino, only the changes listed below needed to be made. The full X-Loader source code, with these changes, can be found on the attached CD. See Appendix D for details.

- Changed the terminal from UART3 to UART1.
- Added support for the second SD card interface (only MMC1 was supported, while Rhino uses MMC2).
- Updated the DDR2 SDRAM timings for the Winbond chip used on Rhino.

6.2.2 **U-Boot**

U-Boot is the third stage bootloader. However, it is more than just a simple bootloader. It provides a simple, terminal-like environment with a few basic commands for reading from and writing to NAND flash memory, copying files across the network and for simple communications with off-chip peripherals. The main functions carried out by U-Boot are:

1. Set *all* the I/O pins to their correct mode (the X-Loader only configured the SDRAM, GPMC and SD card pins).

2. Power-up all on-chip peripherals, and enable their clocks and interrupts.

3. Configure the GPMC correctly for interfacing to external memory devices.

4. Provide a command line environment for testing hardware peripherals and running standalone applications.

5. Provide drivers for peripherals (both on-chip and off), such as I^2C power management devices, network, real-time clocks on the SPI bus, video output interface, USB, and NAND flash memory. These drivers allow the peripherals to be accessed from both the U-Boot command line and standalone applications.

6. Provide a mechanism to easily copy standalone applications into RAM from a network server using TFTP, and then execute these applications.

7. Store settings (such as IP address, boot server IP address and booting options) as environment variables in NAND flash memory, so that they do not need to be re-entered each time the board is booted.

8. Load the operating system from either a network server, USB flash drive, SD card or NAND flash into RAM, and start it. The operating system command line arguments, which are stored as environment variables, are also passed to the operating system.

As with the X-Loader, the Rhino port of U-Boot was based on the port for the AM3517 development board. The full source code can be found on the attached CD. The following changes were made to the AM3517 development board port of U-Boot before it could run on Rhino:

- Changed the mode settings for the rest of the I/O pins so that they would interface correctly with Rhino's off-chip peripherals.
- Configured the GPMC correctly for interfacing to both the NAND flash memory and the FPGA on Rhino. This involved mapping regions of memory to the NAND flash and the FPGA, and configuring the GPMC timings and protocols to interface to each device correctly.
- Changed the sizes of the DDR2 SDRAM and NAND flash devices to match the Rhino hardware.
- Changed terminal from UART3 to UART1.
- Added hardware SPI support (U-Boot did not support SPI on the AM3517 processor).
- Extended the I^2C sub-system to include support for multiple I^2C busses and 16-bit data transfers.
- Exported the GPIO and I^2C functions so that they can be used in standalone applications.

By porting both X-Loader and U-Boot, Rhino is able to boot off either an SD card or NAND flash memory and then enter a terminal-like environment. This environment aids the hardware verification process, as it allows hardware peripherals to be tested and standalone applications to be executed.

6.3 CONFIGURING THE GPMC BUS

As mentioned above, the GPMC bus was configured in U-Boot to interface correctly to the FPGA. This section therefore discusses both the physical GPMC interface and how it was configured within U-Boot.

The GPMC contains 16 external data lines and 10 external address lines. With 10 address lines, it can only address 1024 locations, which is clearly insufficient if one wishes to access the FPGA DDR3 SDRAM from the processor. To increase the addressable space, the 16 data lines can be used as additional address lines. In this multiplexed mode, a total of 26 address lines are available, providing a total of 64 million addressable locations. Since data bus is two bytes wide (16 bits), this provides a total addressable space of 128MB. Furthermore, the GPMC bus has 8 chip select lines, allowing eight external memory devices, each 128MB, to be addressed. This provides a total addressable space of 1GB. The address lines, data lines, chip select lines and other control lines are all shown in Figure 6.4, which is taken from the Rhino schematics.

Looking at Figure 6.4, one can see that chip select (CS) 0 is wired to the NAND flash memory, while CS 1 to 7 are all connected to the FPGA. This means that 768MB of the processor's GPMC address space is allocated to the FPGA, and 128MB to the NAND flash memory. This may seem a bit strange, as the NAND flash memory is 256MB in capacity. Furthermore, looking at the diagram taken from the schematics, one can see that no address lines are connected to the NAND flash. This is because the NAND memory uses a unique addressing protocol, that multiplexes commands, addresses and data onto the 16 data lines. Therefore, the NAND flash memory is allocated only a single physical address, and software drivers are then used to correctly multiplex the commands, addresses and data, and hence create a virtual address space of 256MB.

The 1GB of GPMC address space is memory mapped onto the processor's address space, so that applications can access external devices by simply reading from or writing to appropriate memory addresses. The processor then handles the bus transactions in the background, completely invisibly to the user application. The memory

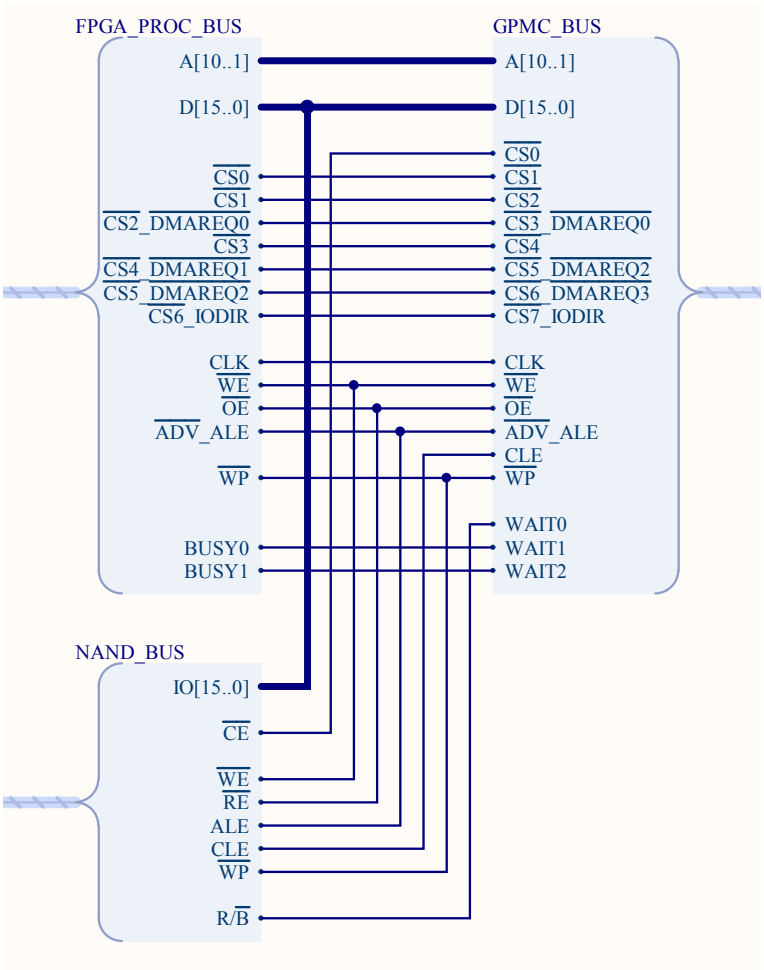


Figure 6.4: The GPMC bus in the Rhino schematics

mapping of the GPMC address space is fully configurable, and has been allocated as in Table 6.1. Note that the GPMC may only use addresses within the range 0x00000000 to 0x3FFFFFFF.

Table 6.1: Chip Select Regions for the GPMC Bus on Rhino

CS number	Connected to	Start address	End address	Region size
CS0	NAND flash	0x30000000	0x37FFFFFF	128MB
CS1	FPGA	0x08000000	0x0FFFFFFF	128MB
CS2	FPGA	0x10000000	0x17FFFFFF	128MB
CS3	FPGA	0x18000000	0x1FFFFFFF	128MB
CS4	FPGA	0x20000000	0x27FFFFFF	128MB
CS5	FPGA	0x28000000	0x2FFFFFFF	128MB
CS6	FPGA	0x38000000	0x3FFFFFFF	128MB
CS7	FPGA	–	–	Unused

Looking at Figure 6.4, one will notice that the GPMC contains a number of control lines. Understanding these control lines is essential to understanding the bus protocols of the GPMC. An explanation of each of the control lines is therefore given in Table 6.2. Although some of the CS lines can be used as DMA request lines (see Figure 6.4), all were used purely as chip selects to keep the testing simple.

Table 6.2: GPMC Control Lines

GPMC Control Line	Full name	Direction	Description
nCS0 to nCS7	Chip select 0 to 7	Output	Enables a specific external memory device.
CLK	Clock	Output	Bus clock
nWE	Write Enable	Output	Indicates valid data on bus during write
nOE	Output Enable	Output	Indicates peripheral can write to data bus.
nADV_ALE	Address Valid	Output	Indicates valid address present on bus.
CLE	Control Latch Enable	Output	Not used by FPGA.
nWP	Write Protect	Output	Enables write protect on memory device.
WAIT0 to WAIT2	Wait inputs	Input	Indicates that memory device still busy.

The bus protocols and timings of the GPMC are fully configurable through processor registers. These settings, as well as the CS memory mappings described earlier, are defined in the U-Boot Rhino board configuration file. The actual bus settings themselves are not given here; rather, they are explained using timing diagrams.

Figure 6.5 shows the bus cycles for a read operation, which takes 8 GPMC_FCLK cycles, while Figure 6.6 shows the write operation, which takes 6 GPMC_FCLK cycles. The GPMC_FCLK is the internal function clock for the GPMC bus, and runs at twice the rate of GPMC CLK. This allows events to take place on both the rising and falling edge of the CLK. Note that these diagrams are *specific* to the timing and protocol settings that have been chosen for Rhino. In both figures, pay careful attention to the bottom plot, which indicates the actions performed by the FPGA and processor on each bus cycle. The "ZZZ"s in the read operation diagram indicate that the data bus is undriven and thus in high-impedance state.

The objective of the Rhino test software is to test the hardware as thoroughly as possible, by the simplest means possible. Therefore, the FPGA-processor bus protocol was kept very simple so that the FPGA only needs to perform actions on the *rising* edge of the GPMC CLK. This does mean that the protocol is not as efficient as it could be; it is possible to achieve the same operations in fewer bus cycles, but with added complexity. However, the bus does still run at the full 83MHz, thereby testing the traces at full speed.

6.4 PROCESSOR TEST SOFTWARE

Standalone U-Boot applications were written to test Rhino's power management sub-system, to test the FPGA-processor bus and to test the real-time clock. The workings of these applications are discussed below.

6.4.1 Application: Rhino LEDs

Rhino's equivalent of "Hello World". The application switches the two processor user LEDs on or off, depending on the command line arguments.

6.4.2 Application: Rhino PSU Enable

This application switches the FPGA power supplies on and off. Since the FPGA power supplies are switched off at power up, this program must be run before the FPGA or any of its peripherals can be tested.

The command line usage for this application is: `go 0x80300000 <VCCINT> <VCCO_AUX> <VCCMGT>` where `0x80300000` is the address in SDRAM where all standalone applications are stored, and the remaining three parameters indicate whether the relevant FPGA power supply must be switched on or off. The program therefore merely reads the command line arguments and sets each of the three FPGA power supply enable lines (connected to GPIO pins on the processor) either high or low.

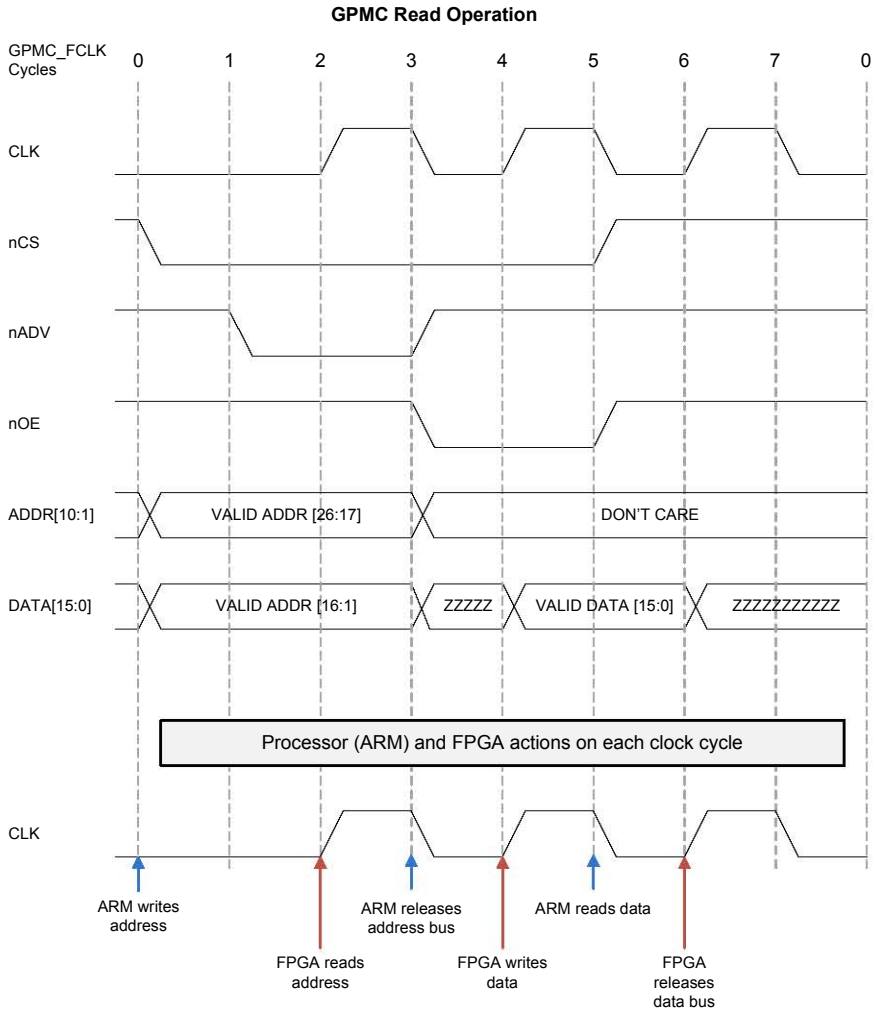


Figure 6.5: The GPMC read operation, as configured for Rhino

6.4.3 **Application: Rhino FMC Control**

The FMC interface is tested using a FPGA gateware design and a Xilinx XM105 FMC debug card. The XM105 card (shown in Figure 6.7) brings the data lines out onto headers, allowing the signals from one half of the bus to be looped back to the other half of the bus. The debug card also contains a Silicon Labs clock (Si570), whose frequency can be programmed over the FMC I^2C bus. However, before the FPGA can test the FMC interface, the FMC card must be powered up and the clock programmed to run at the correct frequency.

The Rhino FMC Control processor application therefore performs the following operations, in the given order. Other than the I^2C operations, which are carried out using the U-Boot I^2C drivers, the steps are performed by simply setting or reading GPIO pins.

1. Switches on all the FMC load switches, supplying 2.5V, 3.3V and 12V to both FMC cards.
2. Sets the PGOOD (power good) and GA (geographic address, which sets the EEPROM address on the I^2C bus) signals going to the FMC cards.
3. Indicates whether the debug card has been plugged into FMC_0 or FMC_1 connector, by reading the nFMC_PRSNT signals.

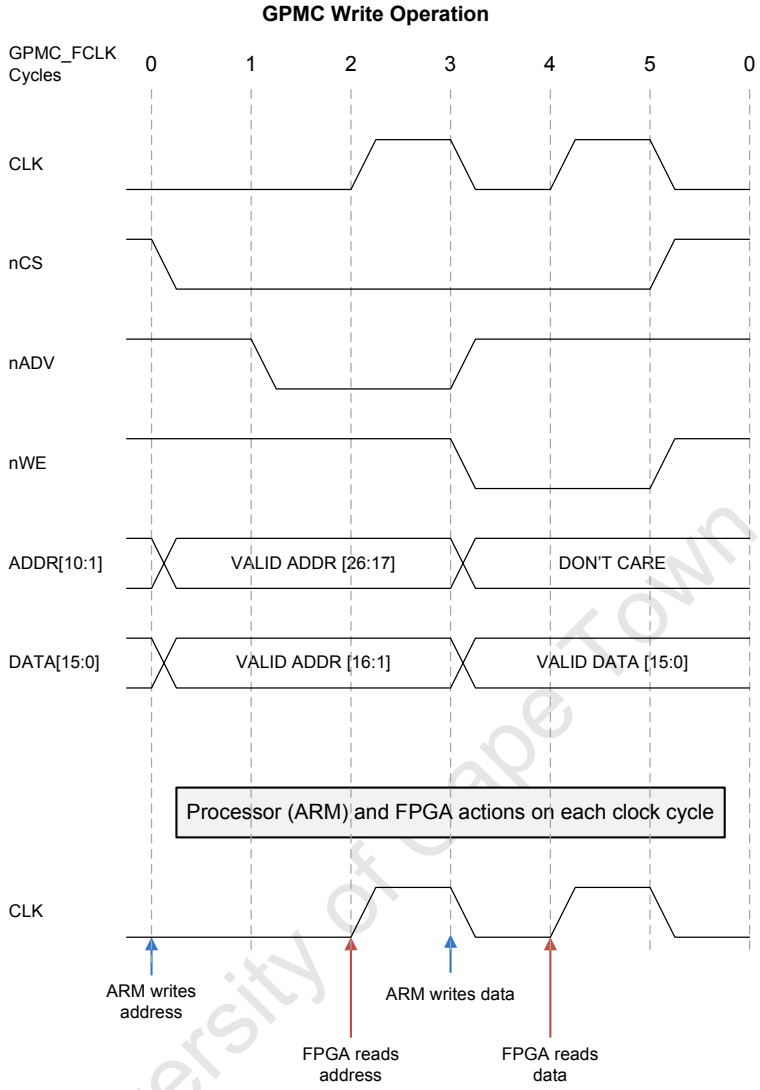


Figure 6.6: The GPMC write operation, as configured for Rhino

4. Reprograms the Silicon Labs clock on the FMC debug card, changing the frequency from 156.25MHz to 312.5MHz, by sending commands over the FMC I^2C bus.

6.4.4 **Application: Rhino GPMC Test**

This standalone application is used in conjunction with a gateware design on the FPGA to test the FPGA-processor bus. For this test, the FPGA is programmed to emulate a read-only memory. Rather than actually storing 768MB of data, the FPGA instead always returns (address[15:0] + 1) when read by the processor.

Simply returning the address as the data value (rather than the address + 1) would not have fully tested the FPGA-processor bus. Since the lower 16-bits of the address are multiplexed onto the data bus during the read operation, if the processor did not clear the data bus after writing the address, and the FPGA did not drive the bus at all, the lower 16-bits of the address would remain on the data bus. The processor would therefore incorrectly think that the FPGA returned the correct value. Hence, the FPGA returns (address[15:0] + 1). The full details of the FPGA gateware design are explained later in this chapter.

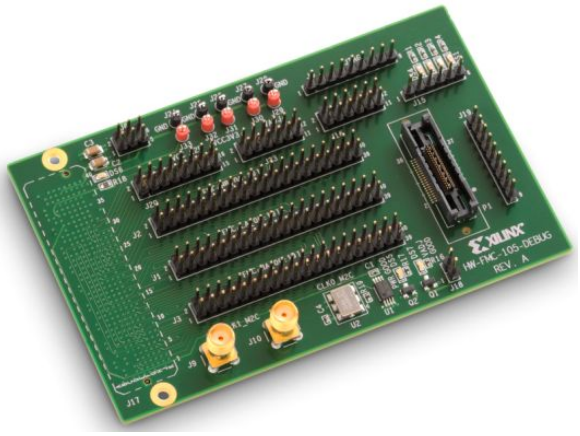


Figure 6.7: Xilinx XM105 FMC debug card
(Image taken from Xilinx website: http://www.xilinx.com)

In order to test the write operation, the FPGA is programmed to display the bottom four bits of the data word on four of the FPGA's LEDs, for any write request. This obviously does not test all the data and address lines, but that is not necessary; the write operation merely needs to test the write control lines.

The Rhino GPMC Test U-Boot application therefore reads a number of different address locations from the FPGA, and checks in each case that the returned values equals (address[15:0] + 1). The application also writes four values in sequence to an arbitrary address, and the user is required to check that the correct pattern appears on the LEDs. Note that in order to read or write to the FPGA, the application merely needs to read or write to an address that lies within one of the FPGA's six chip select regions. The processor handles the bus transactions in the background, completely invisible to the user application.

The application actually runs three separate tests, as described below:
Test 1: The processor sequentially reads all 768MB of the address space allocated to the FPGA, checking at each address that the returned value equals (address[15:0] + 1).
Test 2: The problem with test 1 is that most of the address lines change very slowly (especially those near the MSB). Therefore, to test all the address lines at high speed, Test 2 reads just two separate addresses that have inverse bit values. By alternately reading these two addresses 64 million times each, all the address lines are flipped on every read cycle, verifying their performance at high speed.
Test 3: To test the write operation, the processor writes the following four binary values in sequence: b0001, b0010, b0100, b1000. Since these writes occur 500ms apart, a "walking" effect is created on the FPGA LEDs.

For Test 1 and 2, any incorrect read responses are recorded and the total number of errors are displayed at the end of the test. For Test 3, the user must manually check that the correct pattern appears on the FPGA LEDs.

6.4.5 **Application: Rhino RTC Test**

This application tests the real-time clock that is connected to the processor via SPI. The application first sets the value of the *seconds* register to 1, by writing to the real-time clock via the SPI interface. The application then waits 4 seconds, using a delay loop, and then reads the new *seconds* value from the real-time clock, and prints it out to the terminal. If the real-time clock is working correctly, the new *seconds* value that is printed out to the terminal should be 5.

6.4.6 **Application: Rhino System Monitor**

The system monitor application reads the status of all the power, temperature and fan monitors on Rhino and prints the relevant voltages, currents, temperatures and fan speeds to the terminal. The application runs for a total of 6 minutes, taking readings every 2 seconds. The readings are printed to the terminal as comma-separated values (CSV), so that they can easily be saved as a CSV file and opened in a spreadsheet program.

The software consists of an initialisation stage and a 6 minute loop stage, described below. A number of events, such as powering up the FPGA or turning the fans on, also occur at pre-determined times (i.e. after a fixed number of loop iterations). All I^2C operations use the native U-Boot I^2C driver for the AM3517.

Initialisation:

Switch off fans
Calibrate current sensors (based on the value of the current sensing resistor)
Both of these operations are achieved by sending commands over the I^2C power management bus.

Loop (iterates every 2 seconds):

Print out all voltages, currents, temperatures and fan speeds to the terminal.
This is performed by reading all the monitors connected to the I^2C power management bus.

Events:

After 30 seconds: switch on the FPGA, by driving the FPGA power supply enable lines high.
After 1 minute: prompt user to program the FPGA using a JTAG programmer.
After 2.5 minutes: FPGA programming should now be complete.
After 3 minutes: the FMC load switches are enabled, switching the FMC cards on.
After 4 minutes: the fans are switched on at 90% speed.
After 6 minutes: test complete.

By switching on each component separately, the effect of that particular component on the board's power consumption and temperature can be easily determined.

6.5 **FPGA TEST GATEWARE**

The previous section described the standalone U-Boot applications that were written to test the processor and its peripherals. Similar test applications were written for the FPGA to verify the operation and performance of its peripherals. These *gateware designs* were written in VHDL and compiled into a FPGA configuration file using Xilinx ISE.

6.5.1 **Rhino Blinky**

Rhino Blinky is the most basic test application for the FPGA. The logic circuit is clocked by the 100MHz system clock, which is reduced in frequency using a PLL and an overflow counter within the FPGA. The circuit bit-shifts a 8-bit number every 100ms, which is displayed on the LEDs, creating a "walking" pattern.

The purpose of this design is to verify that the FPGA itself is operating correctly, that the 100MHz system clock is working, and that the LEDs have been correctly wired to the FPGA.

6.5.2 **Rhino DDR3 Memory Test**

The purpose of this gateware design is to test the DDR3 SDRAM ICs and their traces at high speed. In order to simplify the gateware required to interface to the DDR3 SDRAM ICs, Xilinx DDR3 SDRAM IP cores are used to perform the low-level interfacing operations. Each core interfaces to a single DDR3 SDRAM IC using the memory controller blocks on the Spartan-6. The cores are clocked by the 100MHz system clock, which is multiplied up by a PLL to generate the 333MHz clock for the address lines and the 667MHz clock for the data lines. The SDRAM devices are not run at the full 800MHz speed, as the current silicon version of the Spartan-6 contains a flaw that does not allow speeds higher than 667MHz, unless the core voltage is increased.

Two separate VHDL "processes" were written to test each of the two SDRAM ICs. Since these two processes work in exactly the same way, only the state chart for DDR3_0 has been shown in Figure 6.8. Each process is clocked by the output clock of the relevant memory core (called *clk_out* in the state chart). This output clock runs at $1/16^{th}$ of the 667MHz data clock, as the datapath on the fabric side of the core allows 16 bytes (128 bits) to be written every clock cycle.

Note that in the state chart, the boxes indicate states, with the name of the state shown in the top half, and the actions performed in that state in the bottom half. The arrows indicate transitions, with the text indicating the events that trigger that transition, and the text in square brackets explaining the guard condition that must be satisfied before the transition can occur.

To briefly explain the test gateware: incrementing values (1, 2, 3...) are written to the SDRAM. In order to accelerate the write operation, the FIFO queue in the memory core is used to queue 64 write values before the write operation is actually executed. This is repeated until the entire 256MB of SDRAM has been written.

The design then reads back the entire memory IC, requesting 64 addresses at a time (burst mode). The gateware reads each value in the read queue, checking that it matches the value that was originally written to SDRAM. This continues until the entire SDRAM has been read. At the end of the test, the LEDs are switched on to indicate that the test is complete, and whether any errors were detected.

6.5.3 **Rhino 1GBE Test**

The purpose of the Rhino 1GBE Test is to check the 1Gbps Ethernet PHY circuitry, as well as the traces between the FPGA and the PHY. The gateware design checks the management data interface, the GMII bus and the RJ45 analogue interface by enabling various loopback modes.

For the sake of clarity, the discussion of the 1Gbps Ethernet PHY circuitry that was given earlier in the schematics chapter is briefly recapped. Two separate interfaces connect the PHY and the FPGA. The GMII bus is an 8-bit transmit, 8-bit receive full-duplex parallel interface that is used for transferring data packets between the FPGA and the PHY. A separate, slower, serial interface, called the management data interface, is used for configuring the PHY. The FPGA uses this interface to modify registers on the PHY.

Using the management data interface, the FPGA can enable both GMII loopback and line loopback modes. In GMII loopback mode, the PHY simply connects the GMII receive lines to the GMII transmit lines, within the PHY. This allows the FPGA to verify the integrity of the GMII bus at high speed, without having to implement any network protocol. In line loopback mode, the PHY simply echoes any Ethernet packets that it receives on the Ethernet cable, back to device at the other end of the cable. This mode allows the analogue half of the PHY circuitry to be tested by using a PC to send Ethernet packets to Rhino, and then using a packet sniffing application to ensure that all the packets are correctly echoed back.

The gateware design uses a state machine with a large number of states. Since there is an almost completely linear progression from one state to the next, the states have been represented in tabular format in Table 6.3. Most of the states simply read or modify the registers on the PHY using the management data interface. With the exception of STATE 11, every state progresses to the following state once the read or write request has completed.

Table 6.3: Main State Machine for Rhino 1GBE Test

State	Actions within State
STATE 0	Wait 5 seconds for PHY to come out of reset.
STATE 1	Request PHY ID via MDIO.
STATE 2	Check that PHY ID = 0x0141. Request Status register.
STATE 3	Display link and auto-negotiation status. Write new PHY LED settings.
STATE 4	Request Control register via MDIO.
STATE 5	Modify Control register to enable GMII loopback, and write it back.
STATE 6	Wait for completion of Control register write. Then signal start of GMII loopback test.
STATE 7	Wait for GMII loopback test to finish. Request Control register via MDIO.
STATE 8	Modify Control register to disable GMII loopback, and write it back.
STATE 9	Wait for completion of Control register write. Request PHY-specific Control register.
STATE 10	Modify PHY-specific Control register to toggle line loopback, and write it back.
STATE 11	Wait 10 seconds and then go to STATE 9.

Of interest are states 9, 10 and 11. These states read and modify the PHY-specific control register every 10 seconds, and in doing so, enable and disable the line loopback mode every 10 seconds. This aids testing of the cable side of the PHY, as the operation can be verified when line loopback mode is both enabled and disabled.

Another two, much simpler, state machines were written that control the actual sending and receiving of data along the GMII bus. These two state machines are activated when the GMII loopback mode is enabled. The transmit state machine is clocked by the GTX_CLK. It first writes incrementing values from 0 to 255 to the GMII transmit bus. However, with this test, the most significant bits change slowly. The state machine therefore also writes just two values, with inverse bit values, 256 times. This causes the data traces to invert on every bus clock cycle.

The receive state is clocked by the RX_CLK, which is generated by the PHY. On the rising RX_CLK edge, the state machine checks the RX_DV signal from the PHY, to determine if valid data is present. If so, it reads the data bus and compares it to what was just transmitted. If it does not match, the error is recorded.

At the end of the test, the FPGA LEDs are switched on/off to indicate that the test has completed and whether it passed or not. Although not described here, an additional management data interface VHDL module was written to control the sending and receiving of 16-bit register values along the serial MDC and MDIO lines.

6.5.4 **Rhino 10GBE Test**

The Rhino 10GBE Test verifies the performance and functionality of the CX4 10Gbps Ethernet ports. It does this by sending 10 million 64-bit words out one CX4 port, through an external loopback cable to the second CX4 port, and back to the FPGA. The FPGA then inspects the received data to ensure that it matches the transmitted data.

All data is transmitted using the XAUI link-layer protocol; no attempt has been made to implement the Ethernet protocol, which is built on top of the XAUI protocol. The XAUI protocol uses four XAUI lanes, each

running at 3.125Gbps, connected in parallel to a single CX4 connector. These four lanes must be *bonded* on the FPGA, which means that their transmit clocks are synchronised. Furthermore, the XAUI lanes implement 8b/10b error correction. This means that each 8-bit data byte is encoded as a 10-bit symbol, allowing error correction to take place. As a result, the total throughput per CX4 connector is 10Gbps, and not 12.5Gbps.

The Rhino 10GBE Test logic design is implemented using two independent datapath processes, one per CX4 connector. The operation of each of these VHDL processes can be described as follows:

- Stage 1: wait for all the PLLs to achieve lock, and for the transmit clocks for the four XAUI lanes to become synchronised.
- Stage 2a: continuously send IDLE characters so that the receiver at the other end of the cable can detect these IDLE characters, lock onto the carrier clock signal, and align its own receive clocks.
- Stage 2b: At the same time as Stage 2a, wait until own receivers are aligned and synchronised. Once this occurs, clear any local receiver faults and proceed to Stage 3.
- Stage 3: repeat 1 million times:
  - Transmit a fixed 64-bit test sequence. The low-level transceiver logic, implemented in IP cores and transceiver hardware, converts the 64-bit word into four 16-bit words, and transmits each one serially down a XAUI lane.
  - Check that each received 64-bit word matches the test sequence that was transmitted by the transceivers at the other end of the CX4 cable. If not, record the error.

If any errors were detected after the 10 million transfers, the fault is indicated on FPGA LEDs. Note that since the 64-bit test sequence is transmitted serially, the test word doesn't need to be changed for each transfer. A single, fixed value will still cause a number of high-speed transitions on the data line.

Again, Xilinx IP cores were used to control the GTP transceivers and to implement the XAUI protocol. The transceiver IP cores contained a large number of user-defined settings, but most were left at their default value. One setting that is of particular interest is the pre-emphasis setting, which allows the high frequency components of the signal to be emphasised (by a gain ranging from 0dB to 7.6dB) before being transmitted [45], helping to maintain signal integrity. In order to determine the optimum pre-emphasis setting, a simple FPGA logic circuit was designed that cycled through 8 predefined pre-emphasis values every 30 seconds, allowing the signal waveform to be viewed on a high-speed digital oscilloscope at each setting.

6.5.5 **Rhino FMC Test**

The Rhino FMC Test verifies the integrity of FMC connector traces at high speed. This is achieved using the Xilinx XM105 FMC debug card that was described in Section 6.4.3. By placing jumpers on the headers on the FMC debug card, the one half of the FMC LA[33:0] data bus can be looped back to the other half. This allows the FPGA to write data to one half of the data bus, and then read that same data back on the other half. If, however, the data read back does not match what was written, then a hardware fault exists.

This gateware design is used in conjunction with the U-Boot Rhino FMC Control standalone application. The U-Boot application is run before the FPGA is programmed. It powers up the FMC card and programs the Silicon Labs clock to run at 312.5MHz. Two-pin jumpers are then placed on the LA[33:0] headers on the FMC card, creating loopback connections as shown in Figure 6.9. Data is written to the FMC data lines labelled "Outputs" and read back on the lines labelled "Inputs".

The Silicon Labs clock on the FMC card is connected to the FMCx_CLK0_M2C line. A PLL within the FPGA divides the clock frequency by four to obtain 75MHz, which in turn clocks the logic for the test. The design performs two types of tests: a counting test and an alternating bits test. The counting test writes incrementing values from 0 to 65 535 to the FMC output lines, testing for shorts and breaks in the data traces. An alternating bit test is also done which writes just two data values, with inverse bit values, on alternate clock cycles, testing the traces at high-speed. Both tests run for 65 536 clock cycles. The flow of logic on the FPGA for the FMC test can be described as follows:

Initialisation:
cycle_counter = 0
test_type = COUNTING

On falling clock edge:
Write data to output FMC pins

On rising clock edge:
Read FMC input pins. If this does not match most recently written data, record the error.
If test_type = COUNTING, increment write value.
Else test_type = ALTERNATING, and invert bits of write value.
Increment cycle_counter
If cycle_counter = 64k, set test_type = ALTERNATING
If cycle_counter = 128k, stop test

The FPGA LEDs are used to indicate whether the test has finished, and whether there were any errors. The status of the nPRSNT, GA0 and GA1 FMC control signals are also displayed on the LEDs. To aid debugging, a UART controller from OpenCores has been included in this design. The transmit and receive lines of the UART are wired to the GPIO pins that connect to the RS-232 debug board on the test rig. If an error is detected, both the transmitted data and the received data are sent to a PC via this UART for further analysis.

Another version of the test was written that runs at 150MHz, but as will be explained in the next chapter, this gave unsatisfactory results.

6.5.6 **Rhino Processor Interface Test**

This gateware design is used in conjunction with the U-Boot standalone application described earlier (Rhino GPMC Test) to test the FPGA-processor interface. It makes the FPGA appear as a read-only memory, that returns (address[15:0] + 1) whenever it is read. It also has a simple write function, that displays the bottom four bits of the data bus on the LEDs whenever a read occurs to any address.

The logic circuit is described by the flowchart in Figure 6.10. Note that this is a flowchart, and not a state diagram, and describes the main VHDL process. This process is executed on every rising edge of the GPMC CLK. If any of the FPGA CS lines are low, the logic design then samples the nADV, nOE and nWE lines. If nADV is low, then the processor has just written the address to the bus and the FPGA stores this address. If nOE is low, it is the second cycle of a read transaction, and the FPGA writes (address[15:0] + 1) to the data bus. If nWE is low, it is the second cycle of a write transaction, and the FPGA displays D[3:0] on the LEDs.

This concludes the discussion of the test hardware and software that was developed for Rhino. The design of construction of the test rig was described in the first half of the chapter, while the processor test software and FPGA test gateware was discussed in the second half. The next chapter gives the results of these tests.

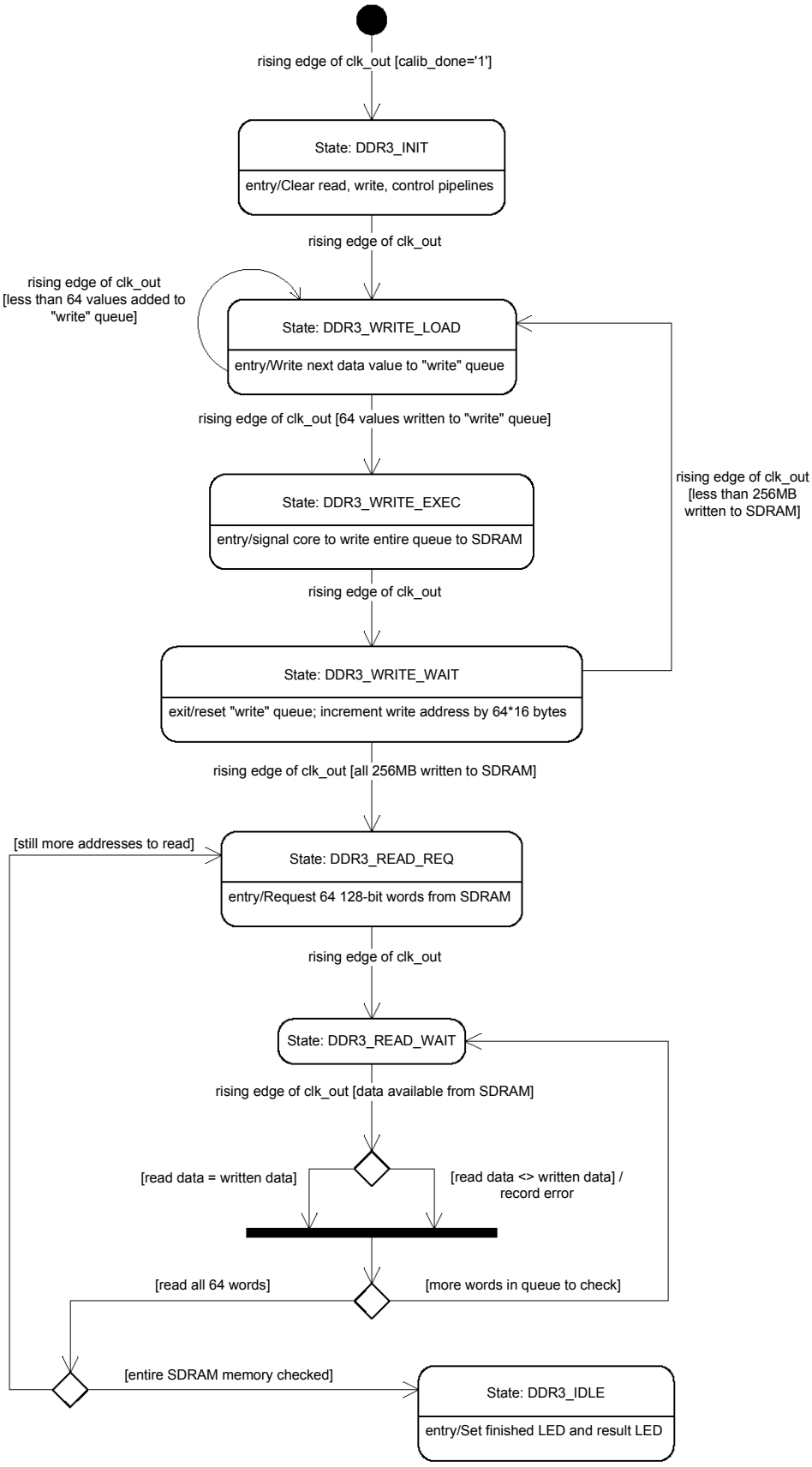


Figure 6.8: State chart for the DDR3_0 memory test

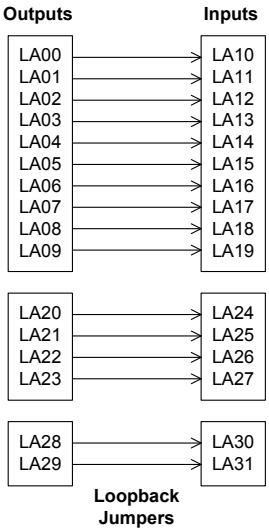


Figure 6.9: Loopback connections using jumpers on the FMC debug card

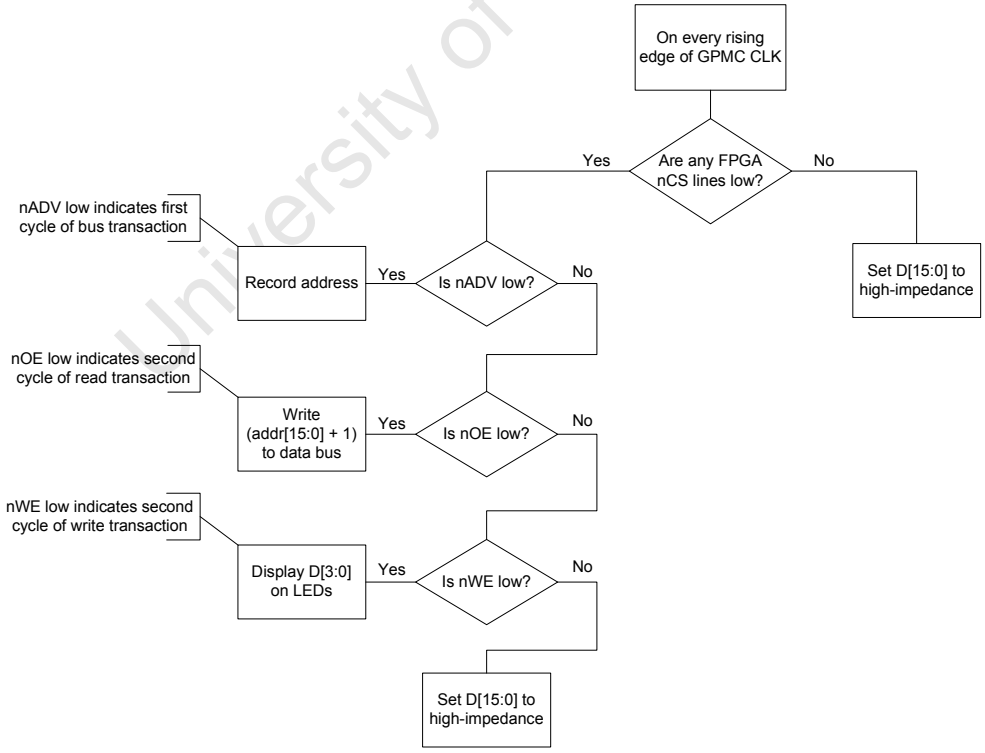


Figure 6.10: Flowchart describing the flow of logic in the Rhino Processor Interface Test gateware

CHAPTER 7

RESULTS OF HARDWARE VERIFICATION

While the test software was being developed, the first prototype Rhino PCBs were being manufactured. This chapter describes the tests that were carried out during manufacture to ensure that the boards were built according to specification, and the results of these tests. Photographs of the completed PCBs are also given. This is followed by a brief discussion of the hardware bugs that were picked up during the process of powering up the boards and running the bootloaders. Finally, the results of the software tests, which were described in the previous chapter, are given.

7.1 THE COMPLETED RHINO PCBS

The PCBs were manufactured in two stages. Firstly, the bare-board PCBs were manufactured by Vector Fabrication in Milpitas, California, USA. They were then shipped to South Africa where they were populated by Tellumat in Cape Town. Four boards were manufactured for the prototype run. Figure 7.1 shows the completed Rhino board, while Figure 7.2 shows the completed board from a different angle, with annotations.

Figure 7.1: The completed Rhino PCB

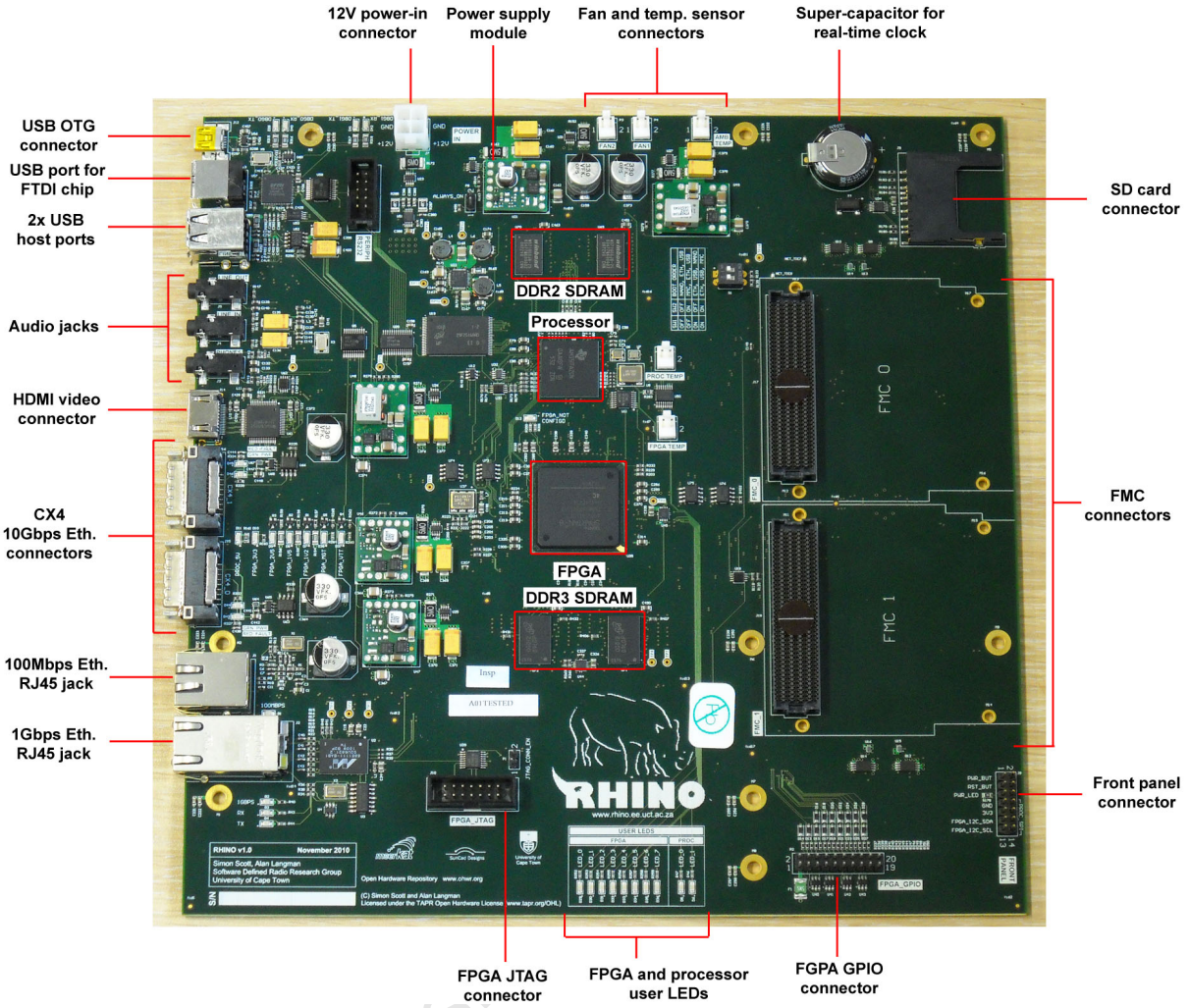


Figure 7.2: Another view of the completed Rhino PCB, with annotations

7.2 PCB Manufacturing Test Results

Two automated test processes were carried out during PCB manufacturing: impedance testing and X-raying. Impedance testing takes place during the manufacturing of the bare PCB and verifies that the impedance-controlled traces do in fact exhibit the correct AC impedance. X-raying, on the other hand, takes place after the board has been fully populated, and is used to check that the BGA packages have been correctly soldered.

Vector Fabrication used a digital sampling oscilloscope to measure the AC impedance of the impedance-controlled traces. The results of these measurements are given in Table 7.1. All measured impedances are within 6% of the target impedance, which is well within the 10% manufacturing tolerance specification.

All the PCBs were X-rayed after they had been assembled by Tellumat. The aim of this was to check for shorts and dry solder joints under the BGA packages, as these packages cannot be visually inspected. An X-ray of one of the DDR3 SDRAM ICs (U72) is given in Figure 7.3. The dark grey circles are the BGA balls, which are arranged in six columns on the SDRAM IC. The open circles between the balls are the vias. From this X-ray, one can see that all the balls are solid circles, indicating that there are no dry solder joints. One or two of the balls do have slightly lighter patches, called voids, but these are all within limits (as will be explained later). Furthermore, there are clearly no shorts between balls or vias.

Table 7.1: Impedance Measurement Results for the Rhino PCB

Layer	Single-ended Impedance			Differential Impedance		
	Target	Actual	Error	Target	Actual	Error
1	50	52.77	+5.54%	100	101.5	+1.5%
				90	91.23	+1.37%
3	50	52.04	+4.08%	100	105.3	+5.3%
5	50	51.93	+3.86%	100	101.5	+1.5%
12	50	51.11	+2.22%	100	103.4	+3.4%
14	50	51.21	+2.42%	100	105.3	+5.3%
16	50	51.21	+2.42%	100	103.4	+3.4%
				90	89.66	-0.38%

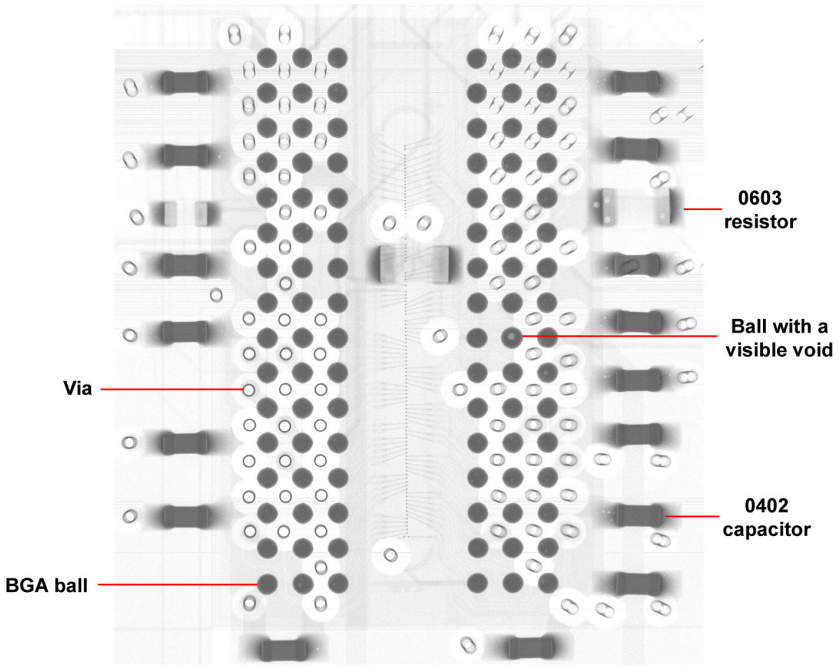


Figure 7.3: X-ray of one of the DDR3 SDRAM ICs on Rhino after production

A void analysis of the bottom-right six balls is given in Figure 7.4. The lighter grey areas on the circles (which have been highlighted with green in this image) indicate voids. A void is simply a cavity within the solder ball, caused by a trapped air bubble or vapourised flux [56]. The void percentage, which is the percentage of the cross-sectional area of the ball that is occupied by the void, is given next to each void. For these six balls, the void percentage varies from 0.6% to 4.8%. Since the IPC-7095 Class 2 specification for BGA void size is 12% of the ball area [57], all voids are within the recommended limit.

All the BGA packages on the Rhino boards were X-rayed and were checked with automated image recognition software. This software checks for shorts, dry joints and voids under the BGA packages. All the Rhino prototype PCBs passed these automated tests.

7.3 Hardware Bugs and Modifications

All four prototype Rhino PCBs were powered up and tested using the test rig described in the previous chapter. This test rig, with one of the Rhino boards, can be seen in Figure 7.5. Two FMC cards have been plugged

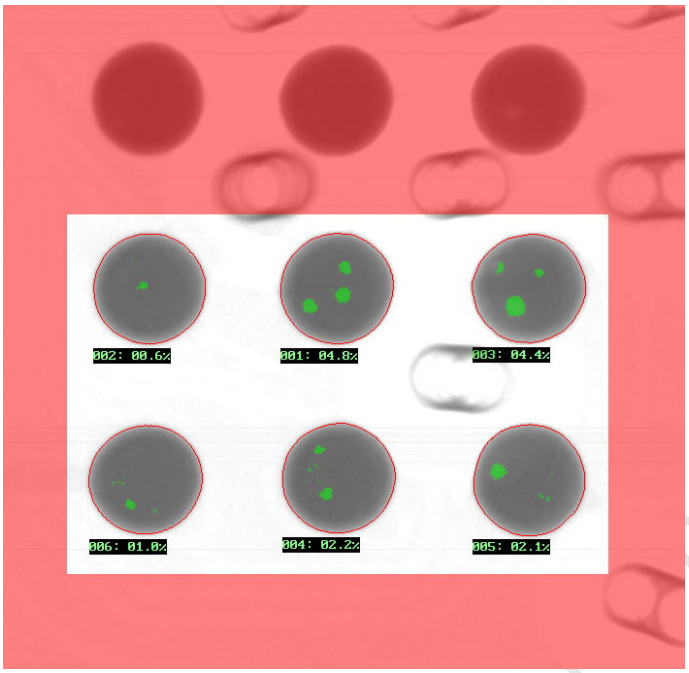


Figure 7.4: Void analysis of six of the balls on the DDR3 SDRAM IC

into this board: the red FMC card on the left is a quad-channel ADC card, while the board on the right is the Xilinx FMC debug card.

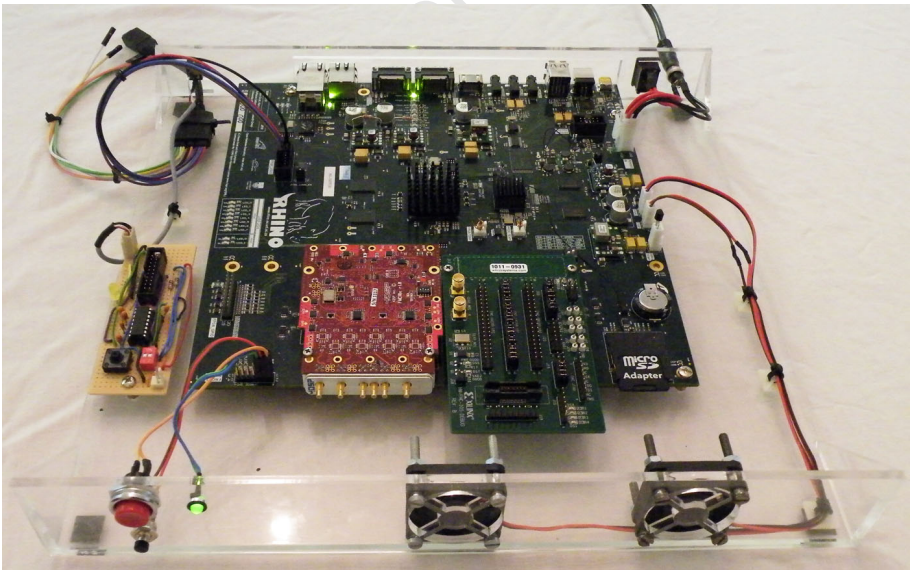


Figure 7.5: The Rhino test rig being used to power up a board

During the process of powering up the first prototype board and attempting to get X-Loader and U-Boot running, a number of hardware errors were picked up. All of these faults were design errors, and not manufacturing problems. Fortunately, most were relatively easy to solve with simple manual modifications. The details of these design errors, and the fixes, are detailed in Table 7.2. In the majority of these cases, the error was simply an omitted pull-up or pull-down resistor, which was easily corrected by hand-soldering a resistor between the appropriate vias or IC pins. Luckily, the four faults that could not be corrected with simple mod-

ifications (indicated by an asterisk next to the fault number) did not impede the operation of the board in any way. Note that the full schematics in Appendix B already include all these fixes.

Table 7.2: Rhino Hardware Design Errors

#	Fault	Fix
1	Processor power supply switches on as soon as DC power is connected.	10K pull-down resistor (R468) has been added to keep power supply enable line low at startup.
2	FPGA power supplies switch on as soon as DC power is connected	2k2 pull-down resistors (R466 and R467) added to keep supply EN pins low at startup.
3	Processor supply rails VCC_3V3_PROC and VCC_LDOs do not ever turn on, as the enable inputs are driven by open-collector outputs.	Added two 47k pull-up resistors (R463 and R464) to open-collector outputs, so that they can pull the enable pins high.
4	Boot buffer IC (U5) is incorrectly enabled at power up, causing incorrect detection of boot switches.	BOOT_BUF_EN net is now driven by GPIO11, and not GPIO115. Hence, the boot buffer is now disabled at startup, allowing correct detection of boot switches.
5	AM3517 boot ROM cannot boot from SD card, as SD clock (which is generated by processor) is not fed back to clock feedback input (MMC2_CLKIN).	Loopback connection added between MMC2_CLK and MMC2_CLKIN on processor.
6*	Possible overshoot on SD card traces. However, processor still able to read and write card without errors.	22 ohm series resistors (R38, R83, R85, R331, R338 and R462) have been added to all SD card traces.
7	Pressing the reset button powers down the processor, instead of resetting it.	10k pull-up resistor (R465) was added to keep PWR_KILL net high during the reset.
8*	The FPGA 3.3V power supply trips-out if a CX4 cable is plugged in before the FPGA is powered up. This is caused by a 2ms glitch in the optical cable detection circuitry at power-up.	A RC delay circuit (R469, C562) is added that only enables the CX4 optical power if the output of the detection circuit is high for at least 100ms.
9*	The traffic LED on the 1Gbps Ethernet RJ45 jack never turns on.	Previously, two parallel resistors were used as an OR-gate for switching the LED on. Instead, two parallel diodes (D47 and D48) are now used.
10*	The two USB UART serial ports have RX and TX traffic LEDs. The silkscreen label DBGx_RX was placed next to the TX LED, and vice versa.	The silkscreen labels were swapped around on the PCB design file.

After these modifications were made to the original design, the first prototype board was able to power up and run both X-Loader and U-Boot. A USB cable was used to connect the UART USB port to a PC, and the boot messages were monitored using a serial terminal program. The U-Boot command prompt was then used to run some rudimentary tests to verify that the processor, the power supplies and USB-to-UART converter were working correctly. The rest of the board was tested in more detail later, as is described in the next section.

Once the first board was powered up and running, the remaining three prototype boards were checked. Two of the three boards loaded U-Boot correctly, but one did not. After further investigation, it was discovered that the DDR2 SDRAM on this board was faulty, as the X-Loader would hang when trying to copy U-Boot into SDRAM. No shorts or large voids were present on the X-ray for that board. One must therefore conclude that either one of the DDR2 SDRAM ICs are faulty, or else there is a problem with a PCB trace. Since no flying-probe or bed-of-nails testing was performed on the bare boards, the second cause is more likely.

7.4 RESULTS OF THE SOFTWARE-BASED TESTS

Once the boards were powered up and running U-Boot, more detailed hardware tests were carried out. This involved using either the native U-Boot diagnostic tools, the standalone test applications described in the previous chapter, or the FPGA test gateware that was developed. The results of these tests are described here.

Oscilloscope plots have been given for a number of tests, which were obtained by probing a via on the relevant bus during the test. Since the via was typically in the middle of the trace, and the oscilloscope probes were unterminated, a number of the plots exhibit waveforms with poor signal integrity. Furthermore, 50 ohm probes could not be used, as the end of the trace was already terminated and, in most cases, the source was not able to drive two 50 ohm loads. Therefore, the oscilloscope plots shown here are not in anyway demonstrative of the integrity of the actual waveforms at the receiver. A Tektronix TDS5052B oscilloscope was used for all tests.

7.4.1 The Processor and its Peripherals

Since most of the tests for the processor and its peripherals were fairly trivial, they will all be discussed together in this section.

AM3517 ARM Processor

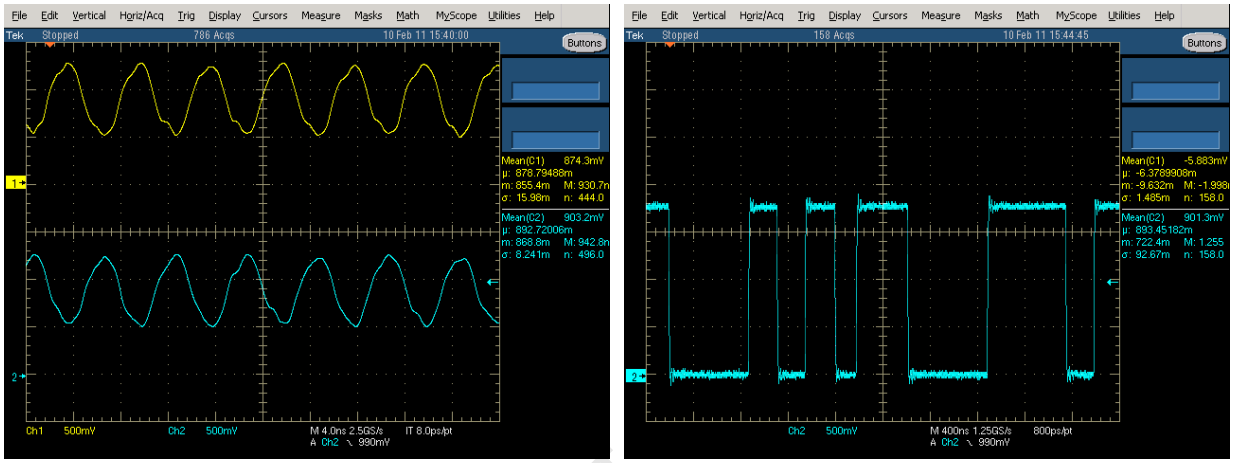
To verify that the processor itself was working correctly, X-Loader and U-Boot were run from an SD card, and the terminal output was monitored using a serial terminal application on a PC. The terminal output during a successful boot is given below:

```
Texas Instruments X-Loader 1.46 (Dec 23 2010 - 19:50:17)
Starting X-loader on MMC
Reading boot sector

220504 Bytes Read from MMC
Starting OS Bootloader from MMC...
Starting OS Bootloader...

U-Boot 2009.11-svn436 (Feb 08 2011 - 01:40:39)

OMAP34xx/35xx-GP ES1.0, CPU-OPP2 L3-165MHz
AM3517RHINO Board + LPDDR/NAND
I2C:   ready
DRAM:  256 MB
NAND:  256 MiB
In:    serial
Out:   serial
Err:   serial
Die ID #52020001000000000015da39601021017
Net:   davinci_emac_initialize
Ethernet PHY: GENERIC @ 0x01
DaVinci EMAC
Hit any key to stop autoboot: 10  9  8  7  6  5  4  3  2  1  0
No MMC card found
Booting from nand ...

NAND read: device 0 offset 0x280000, size 0x400000
 4194304 bytes read: OK
Wrong Image Format for bootm command
ERROR: can't get kernel image!
AM3517_RHINO #
```

Processor DDR2 SDRAM

U-Boot contains an integrated memory test application. This application checks both the integrity of the memory, as well as the integrity of the signal traces. It checks for stuck-high, stuck-low and shorted pins. The test was run five times, over a period of a few hours, and no errors were detected.

Figure 7.6a shows the oscilloscope traces for the two halves of the differential clock, while Figure 7.6b shows the trace for address line A1. While the address line is as clean as one would wish, the clock line does not exhibit fast rise and fall edges. This may be due to slow rise times at the output, or due to reflections caused by an incorrectly terminated oscilloscope probe. Nevertheless, the DDR2 still passes the memory tests.



(a) Clock CK_N and CK_P

(b) Address A1

Figure 7.6: DDR2 SDRAM oscilloscope traces

SD Card Connector

Since both the X-Loader and U-Boot were correctly read from the SD card and executed, one can only conclude that the SD card connector works correctly.

100Mbps Ethernet PHY

To test the Ethernet PHY circuitry, an Ethernet cable was used to connect the processor's Ethernet jack to a PC. The TFTP (trivial file transfer protocol) tool built into U-Boot was then used to copy a file from the PC to the board. The terminal output from this transfer is shown below. As can be seen, the entire file (rhino_psu_enable.bin) was successfully transferred.

```
AM3517_RHINO # tftp 0x80300000 rhino_psu_enable.bin

Using DaVinci EMAC device
TFTP from server 192.168.0.100; our IP address is 192.168.0.1
Filename 'rhino_psu_enable.bin'.
Load address: 0x80300000
Loading: **#
done
Bytes transferred = 642 (282 hex)
AM3517_RHINO #
```

NAND Flash

To test the NAND flash memory, the X-Loader and U-Boot images were copied from a PC, across the network, into the processor's SDRAM. These bootloader images were then written to the NAND flash using U-Boot's NAND drivers. The SD card was removed and the board power cycled. The board booted correctly into U-Boot, using the images stored on the NAND flash, proving that the flash memory works correctly.

Real-time Clock

The real-time clock was tested using the standalone application that was described in the previous chapter (Rhino RTC Test). This application set the *seconds* register of the real-time clock to 1, waited four seconds in a delay loop, and then read the *seconds* register back. The terminal output from this application is shown below. The program outputs the value 5, proving that the real-time clock did in fact increment its seconds register by 4 during the delay.

```
AM3517_RHINO # go 0x80300000

## Starting application at 0x80300000 ...
Rhino Real-time Clock Test Program
Seconds: 5.
## Application terminated, rc = 0x0
AM3517_RHINO #
```

USB Host Ports, Audio Codec, and Video Transmitter

None of these peripherals were tested, as no U-Boot drivers were available for any of them. While new drivers could have been written, this would have been both time consuming, as these are all complex peripherals, and an inefficient use of time, as working Linux drivers already exist for these peripherals. Since these are non-critical peripherals, they will only be tested once the Linux port is complete.

7.4.2 Spartan-6 FPGA and its User LEDs

The FPGA and its peripherals were tested by programming the FPGA with the gateware designs that were described in the previous chapter. The configuration files (bit files) for these designs were programmed onto the FPGA using a JTAG programmer that was connected to the FPGA's dedicated JTAG port. Neither the processor nor the USB-to-JTAG/I^2C/RS-232 converter could be used to program the FPGA, as the support software had not yet been developed for these interfaces.

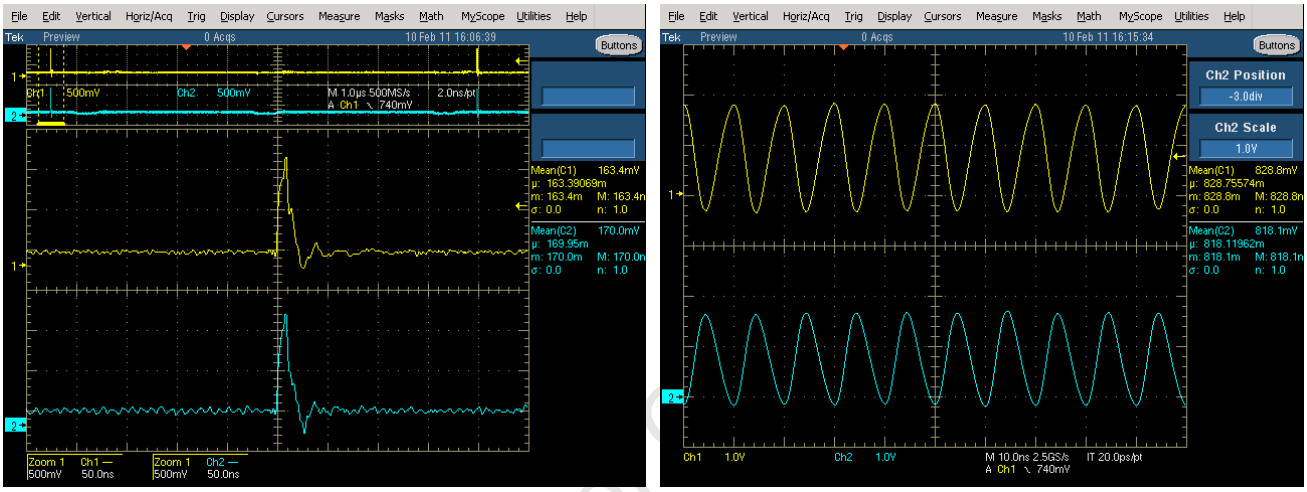
To test the Spartan-6 FPGA itself, the JTAG programmer was connected and the JTAG chain was scanned using a PC. The Spartan-6 was correctly detected, proving that the FPGA core works correctly. A basic configuration file, Rhino Blinky, was then downloaded onto the FPGA, which flashed the LEDs correctly.

7.4.3 DDR3 SDRAM

The operation of the DDR3 SDRAM was verified using the DDR3 SDRAM gateware. As has been described, this test wrote to both SDRAM ICs, and then read the data back. The entire test took 3 seconds to run, and the results are shown in Table 7.3, with no errors being detected. The signals on two of the traces were captured by an oscilloscope. Figure 7.7a shows address bits 0 and 1 for the DDR3_0 IC, while Figure 7.7b shows the LDQS differential data strobe. While the LDQS signal transitions cleanly without overshoot or ringing, the address signals are not ideal. The overshoot and undershoot, resulting from reflections, is clearly evident. However, this is most likely due to an unterminated oscilloscope, and not real signal integrity problems.

Table 7.3: Status of the FPGA LEDs during the DDR3 SDRAM Test

LED	Meaning	State
0	DDR3_0 calibration complete	On
1	DDR3_0 test complete	On (after 3 seconds)
2	DDR3_0 test passed (no errors)	On (after 3 seconds)
3	DDR3_1 calibration complete	On
4	DDR3_1 test complete	On (after 3 seconds)
5	DDR3_1 test passed (no errors)	On (after 3 seconds)
6	Heartbeat	Flashing



(a) Address lines A0 (top) and A1 (bottom)

(b) Data strobe lines LDQS_P and LDQS_N

Figure 7.7: DDR3_0 SDRAM oscilloscope traces

7.4.4 Processor-FPGA Interface

Both processor software and FPGA gateware was required to test the FPGA-processor interface. The Rhino GPMC Test application on the processor reads the entire address space, each time ensuring that the FPGA returns the correct value. The terminal output below shows the six FPGA chip-select regions being read, followed by an alternating address test (Test 2). No errors were detected. The entire test took 116 seconds, which gives a net read throughput of 7.7MB/s. The actual throughput in the final Rhino system should be much higher, as burst mode, more efficient bus transfers and DMA will be used.

```
AM3517_RHINO # go 0x80300000

## Starting application at 0x80300000 ...

== Rhino GPMC Test ==

Reading data from FPGA...
Read test 1: testing next CS region....
Read test 1: testing next CS region....
Read test 1: testing next CS region....
Read test 1: testing next CS region....
Read test 1: testing next CS region....
Read test 1: testing next CS region....
```

```
Read test 2: starting...
Finished reading from FPGA. Number of errors: 0
Writing data to FPGA to flash LEDs
GPMC test finished.
## Application terminated, rc = 0x0
AM3517_RHINO #
```

During the write test, the FPGA LEDs were observed. They turned on in the correct order, proving that the write operation works correctly. Oscilloscope traces were recorded for two of the signals during the read test: address valid signal nADV and data bit D0. The plot shows four "spikes" in the nADV signal, representing four bus transfers. In each "spike", the nADV signal goes low briefly near the beginning of the transaction, indicating that the address is valid. The signal stays high for the rest of the transfer while data is present on the bus. Since these plots were recorded during the read test, the data bit D0 is alternately high and low, representing incrementing data values. These plots show a fair amount of ringing, again due to the unterminated probe being connected to the middle of the trace.

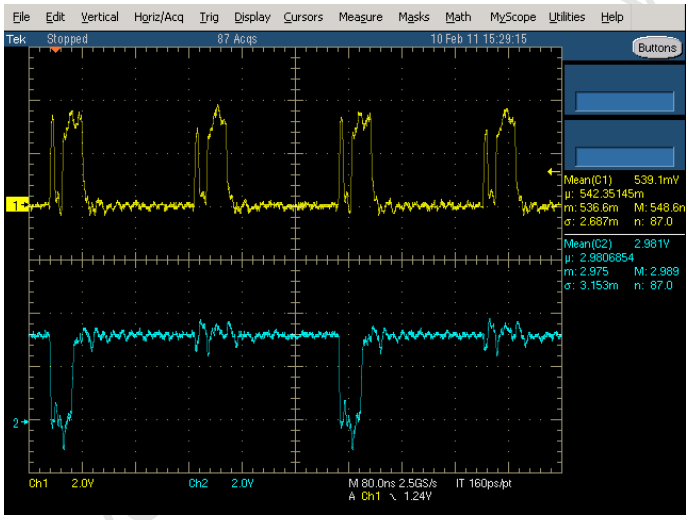


Figure 7.8: Oscilloscope traces for the processor-FPGA bus, showing signals nADV (top) and D0 (bottom)

7.4.5 **CX4 10Gbps Ethernet Ports**

The CX4 ports were tested by connecting a 0.5m CX4 cable from the one port to the other. 10 million packets were transmitted from each port, while the other port received and checked each packet. The table of status LEDs below show that no errors were detected on either port after receiving the 10 million packets.

One of the CX4 ports was then connected then connected to a 10Gbps Ethernet PCI-Express card on a PC, to check that the transceivers would align and synchronise with a different link partner. Figure 7.9 shows that Windows XP detects the link as being operational. Note that, due to a well-known bug in Windows XP, it cannot show link speeds higher than 1.4Gbps in the network status dialogue box [58].

Lastly, different length CX4 cables were used to connect one of the CX4 ports on Rhino to a Tektronix DPO72004 high-speed digital oscilloscope. This oscilloscope was used to produce eye-diagrams for each of the different cable lengths, with different pre-emphasis settings. Figure 7.10 shows eye diagrams for a 0.5m cable and a 2m cable, with a pre-emphasis setting of 3.5dB. In both plots, the eye is clearly open and only minor overshoot is present.

Table 7.4: Status of the FPGA LEDs during the CX4 Test

LED	Meaning	State after Loopback Test
0	CX4_0: link is up and all receivers aligned	On
1	CX4_0: test complete	On
2	CX4_0: errors were detected	Off
3	CX4_0: TX or RX local fault	Off
4	CX4_1: link is up and all receivers aligned	On
5	CX4_1: test complete	On
6	CX4_1: errors were detected	Off
7	CX4_1: TX or RX local fault	Off

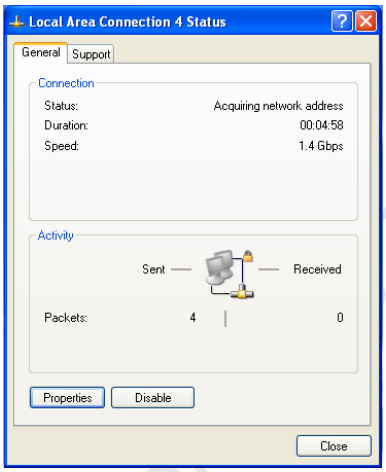
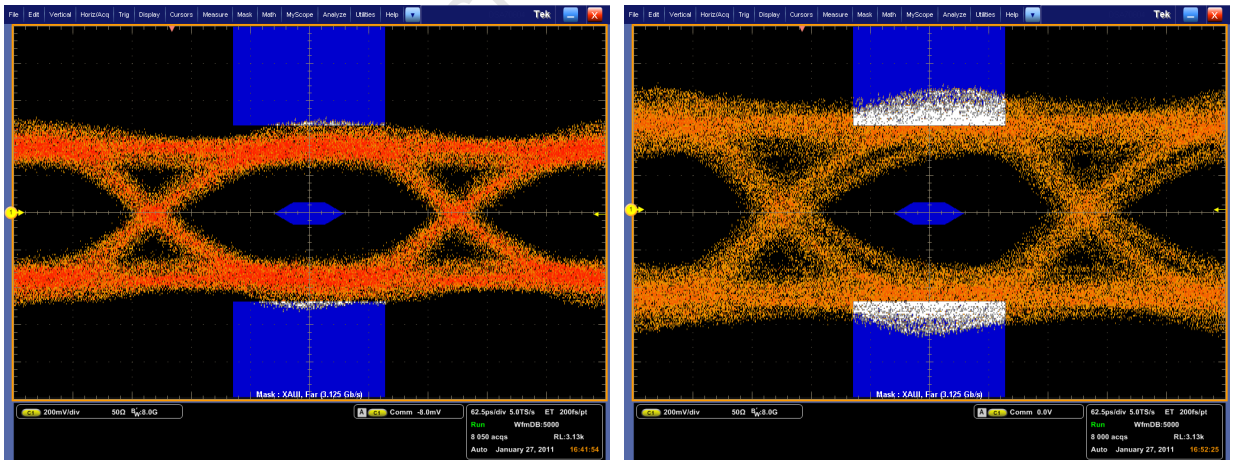


Figure 7.9: Windows XP network status when connected to Rhino via CX4 cable



(a) 0.5m cable

(b) 2m cable

Figure 7.10: Eye diagrams for the XAUI signals on different length CX4 cables

7.4.6 **1Gbps Ethernet**

The 1Gbps Ethernet gateware carried out three separate tests. The first test verified that the management data interface worked correctly by reading the Ethernet PHY ID. Table 7.5, which shows the status LEDs for the

test, shows that LED_3 was on, indicating that this test passed. The gateware then checked the GMII interface, by writing to it in loopback mode, and checking the data that was received back. LED_6 and LED_7 show that this test passed too.

Table 7.5: Status of the FPGA LEDs during the 1Gbps Ethernet Test

LED	Meaning	State after Test
0	Register reads (via MDIO) complete	On
1	GMII loopback test complete (all packets sent)	On
2	GMII loopback test complete (all packets received)	On
3	PHY ID correct	On
4	Ethernet link up	On
5	Link auto-negotiation complete	On
6	GMII loopback test passed (data bits)	On
7	GMII loopback test passed (control bits)	On
GPIO LED	Line loopback enabled	Flashing at 0.1Hz

Lastly, the Ethernet port was connected to a PC equipped with a Gigabit Ethernet card. The line loopback mode was then switched on and off every 10 seconds, as was indicated by an LED connected to one of the GPIO lines. A simple C application was written for the PC, which sent a UDP packet to the board once a second, via the Ethernet link. The traffic on the PC's Ethernet port was monitored using WireShark, a packet sniffing tool. When the line loopback was enabled on the Rhino board, Wireshark detected each packet being correctly echoed back to the PC. When line loopback mode was disabled, no packets were echoed.

Figure 7.11 gives a Windows XP network connection dialogue box, showing that the PC recognised the Rhino board as a valid Ethernet device, and correctly auto-negotiated the speed to 1Gbps.

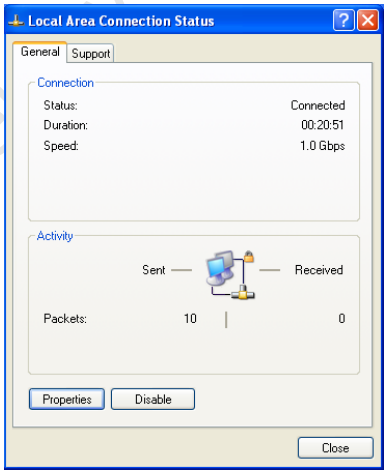


Figure 7.11: Windows XP network status when connected to Rhino via 1Gbps Ethernet cable

7.4.7 FMC Connectors

As was described in the previous chapter, the FMC connectors were tested using the Xilinx FMC debug card, with jumpers to loopback the data bus. However, before the FPGA gateware could run, the card needed to be powered up and the clock frequency reprogrammed. The output of the standalone U-Boot application that performed these operations is shown below. Note that the FMC0_nPRST_M2C is detected as being low, which means that an FMC card has been plugged into FMC connector 0.

```
AM3517_RHINO # go 0x80300000

## Starting application at 0x80300000 ...

== Rhino FMC Control Program ==

Switching on FMC supplies.
Setting FMC_PG_C2M = 1, FMC0_GA[1:0] = 00, FMC1_GA[1:0] = 01.
Done setting signals.
FMC0_nPRST_M2C> 0
FMC1_nPRST_M2C> 1
Register 7 of SiLab chip was 0. Setting to 0x00
Finished.
## Application terminated, rc = 0x0
AM3517_RHINO #
```

Once the card was powered up, the FPGA was programmed with the FMC test gateware. This gateware performed both a counting test (write incrementing values to the FMC) and an alternating test (write two data words with inverse bit values). Table 7.6 shows that both tests passed, for both FMC_0 and FMC_1.

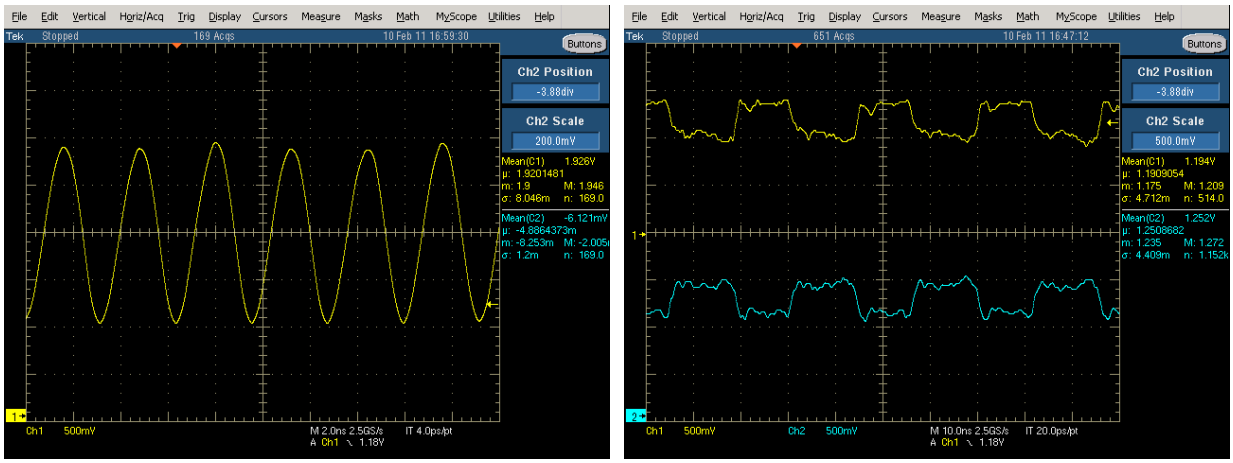
Table 7.6: Status of the FPGA LEDs after the FMC Tests

LED	Meaning	State for FMC_0 Test	State for FMC_1 Test
0	FMCx_PRSNT	On	On
1	FMCx_GA0	Off	On
2	FMCx_GA1	Off	Off
3	Unused	–	–
4	All tests finished	On	On
5	Counting test passed	On	On
6	Alternating test passed	On	On
7	Heartbeat	Flashing	Flashing

Oscilloscope plots were captured for the two of the signals. Figure 7.12a shows the clock signal after it has been converted from LVDS to single-ended. This signal does not reach either the ground or supply rails, as the slew rate is too slow. This is because the clock signal is running at 312.5MHz, but the LVDS converter is only rated for signals up to 250MHz. Although the FPGA does register the clock signals correctly, running the clock above 250MHz is not advisable for sensitive applications. Figure 7.12 shows the LA00 LVDS differential data signals for FMC connector 0. This signal, which has a data rate of 75Mbps, is fairly noisy and has ringing. This is because the Xilinx FMC debug card is unfortunately not designed for high speed: it has limited impedance control, the differential signals are not always routed together, and the jumpers are not designed for high-speed signals. Therefore, this particular test failed when the data was clocked at 150Mbps. However, if a proper high-speed FMC card is used, much better performance can be expected.

7.4.8 GPIO Connector

The GPIO connector was indirectly tested in two previous tests. The 1Gbps Ethernet test used an LED, connected to one of the GPIO lines, to indicate whether line loopback was enabled. The FMC test used the RS-232 debug board, which was connected to the GPIO connector, to send debug information to the PC over a serial cable. Since both of these tests worked correctly, one can assume that the GPIO connector performs as expected.



(a) FMC1_CLK0_M2C (single-ended), running at 312.5MHz (b) Differential data pair FMC0_LA00_N and FMC0_LA00_P

Figure 7.12: Oscilloscope plots of the FMC signals during testing

7.4.9 Power Supply and Management Sub-System

The Rhino System Monitor U-Boot application was developed to test the power supply and management sub-system. This application took readings from all the sensors every two seconds, and saved the results to a CSV file. The results of this file are plotted in Figures 7.13, 7.14 and 7.15. In each plot, the power-on and power-off events are marked with vertical dotted lines.

In Figure 7.13, the four FPGA power rails turn on correctly when the FPGA power supplies are switched on. The vertical axis of each plot is in mV, while the horizontal axis is in seconds.

In Figure 7.14, the current on the four FPGA rails ramps up when the FPGA is switched on. Once the FPGA programming has been completed, a significant increase of current is noticed on both the 1.2V FPGA core supply and the 1.5V FPGA DDR3 supply. This is because the FPGA was programmed with the DDR3 SDRAM memory test. When the FMC cards are eventually switched on, the current on the 3.3V and 12V FMC rails climbs, while the 2.5V FMC rail remains at zero current (it is not used on the FMC debug card). Just like the voltage plots, the vertical axis of these plots is in mA.

Lastly, Figure 7.15 shows the FPGA temperature rising once programming begins, and then rising at a faster rate once the DDR3 SDRAM test is running. However, once the fans are switched on, all three temperatures start dropping. The vertical axes of the temperature plots are in °C. The fan speeds rise correctly to 90% of their maximum speed when they are switched on, verifying that the fan speed controller works correctly.

Table 7.7: Comparison of Automated Measurements and Manual Measurements

Reading	Automated Measurement	Manual Measurement
12V In	12.16V	12.16V
5V Processor	5.02V	5.01V
1.2V FPGA	1.22V	1.21V
1.5V FGPA	1.52V	1.51V
2.5V FPGA	2.51V	2.50V
3.3V FPGA	3.31V	3.29V
Ambient Temperature	30°C	28.2°C

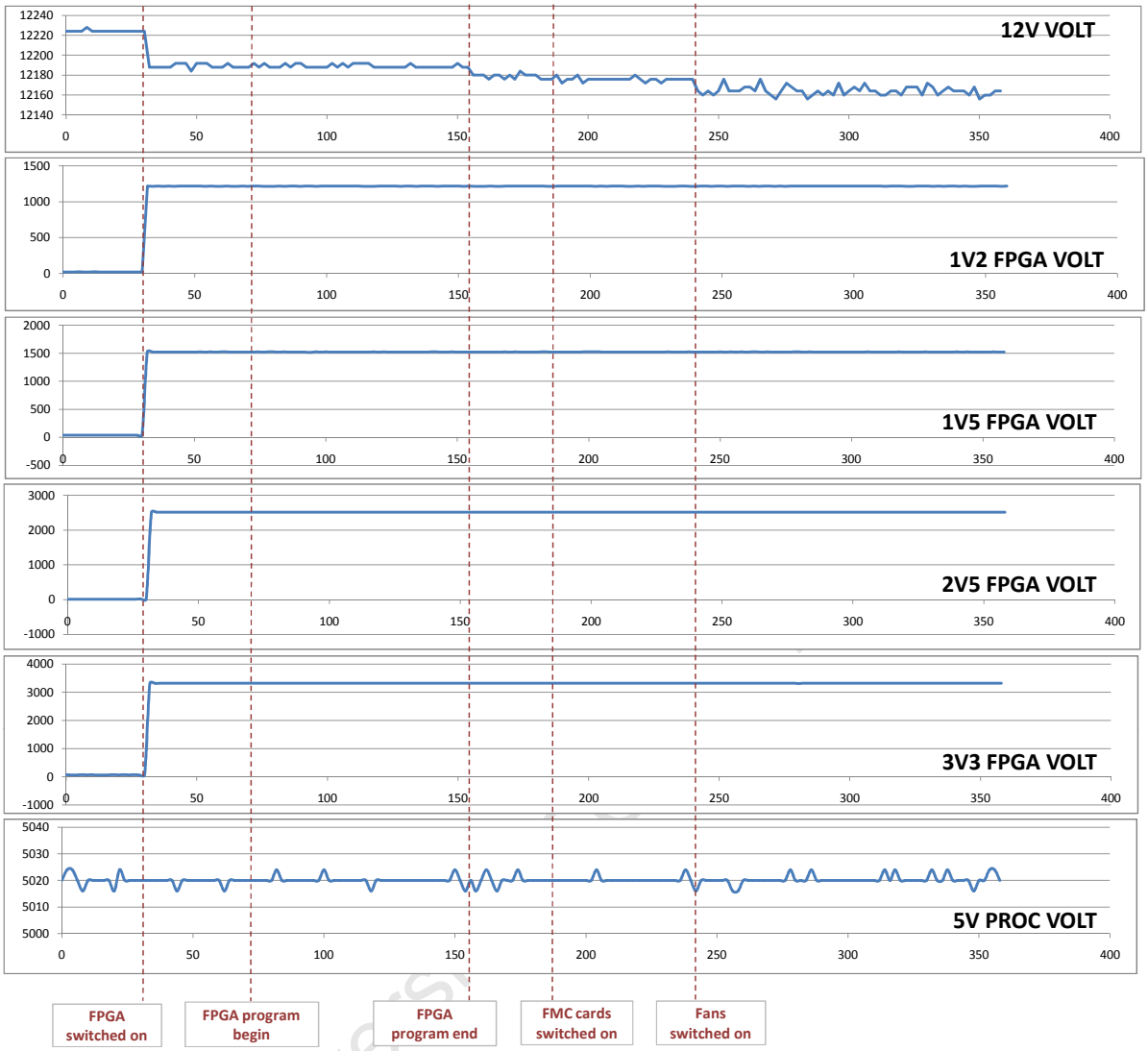


Figure 7.13: Plot of the voltage readings taken by the Rhino System Monitor

The validity of the voltage and temperature readings was checked by taking manual readings with a calibrated voltmeter and thermometer after the test. These readings were compared to the final system monitor readings and the results are shown in Table 7.7. All automated voltage measurements are sufficiently close to both the desired voltage and the manual measurement. The 1.8°C difference between the automated and manual temperature readings is due to the slow response time of the mercury thermometer used. Although the accuracy of the current measurements has not been determined at this stage, they are in line with predicted values.

7.4.10 Summary of Test Results

Since all the software-based tests passed for the processor, FPGA and monitoring sub-systems, one can conclude that Rhino was designed and built correctly. The one test, that although passed, did not perform quite as well as expected, was the FMC connector test. The fact that the test passed at 75MHz, and not 150MHz, is most probably due to the design of the Xilinx FMC debug card, and not Rhino itself. This will however require further testing with high-performance FMC ADC cards before Rhino enters production.

On the software front, other members of the UCT Software-Defined Radio Research Group have recently

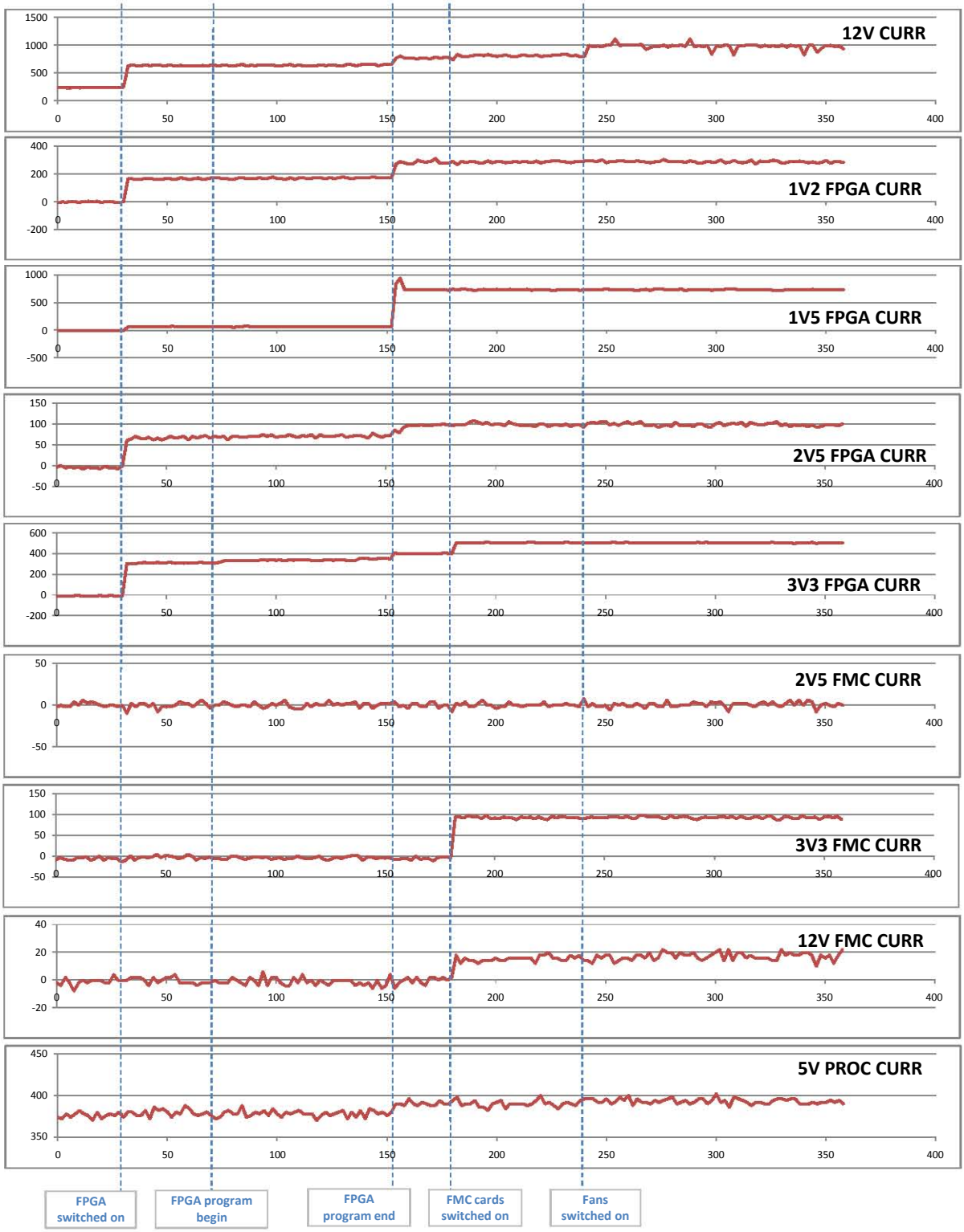


Figure 7.14: Plot of the current readings taken by the Rhino System Monitor

managed to get Linux running on the processor. A U-Boot application has also been developed that allows the processor to program the FPGA via the Serial Configuration Interface. These developments prove that Rhino meets the requirements of running Linux on the processor and having the processor configure the FPGA.

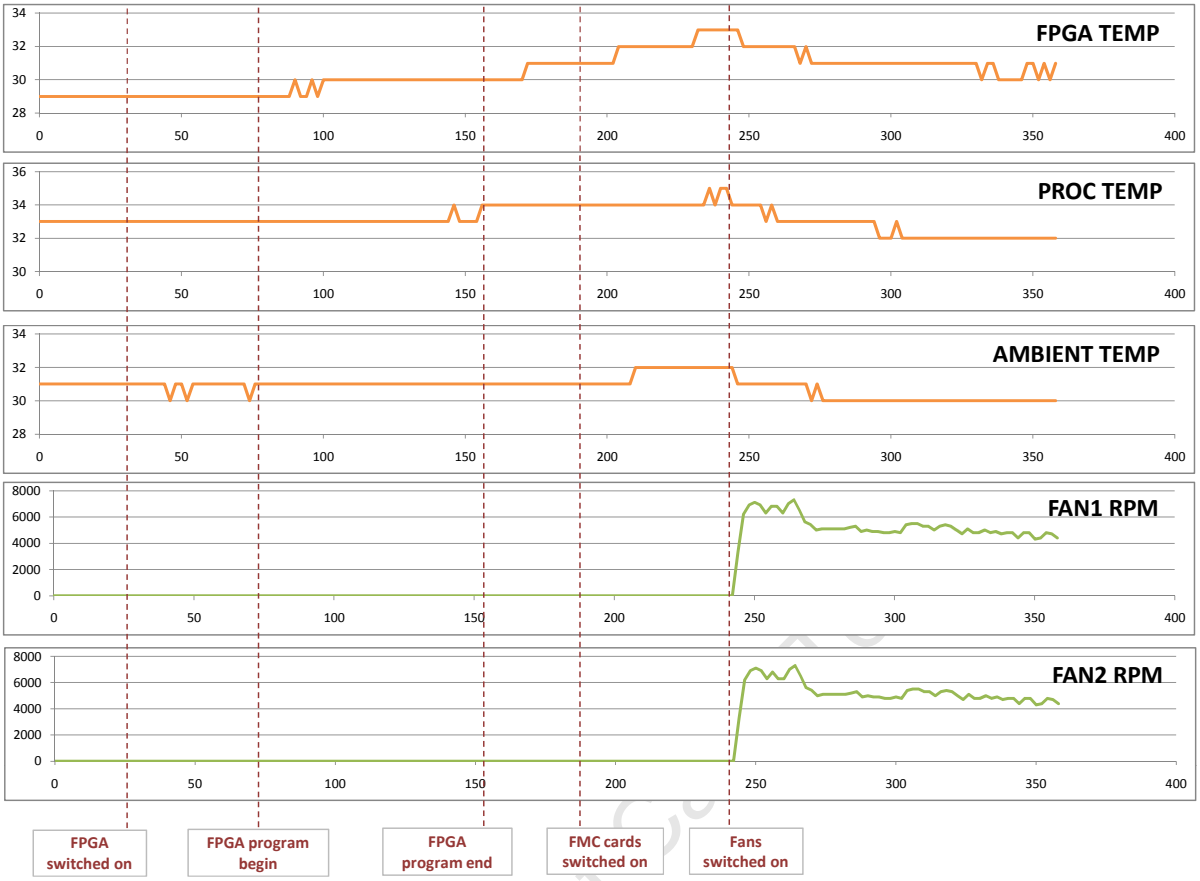


Figure 7.15: Plot of the temperature and fan speed readings taken by the Rhino System Monitor

7.5 Estimated Board Cost

Once the initial prototypes were working, quotes were obtained from manufacturers for producing larger volumes of the Rhino board. Below is the cost per board, if manufactured in quantities of 100. The cost of the enclosure (which has had some preliminary design and costing done), power supply and cabling has also been included. Therefore, the total below represents the total estimated cost price for producing a complete Rhino system. FMC mezzanine cards obviously need to be purchased separately at an additional cost.

Table 7.8: Estimated System Cost for Rhino

Components from Digi-Key	$905
PCB manufacturing by Vector Fabrication	$300
Assembly by Tellumat	$280
Power supply, cabling, etc	$70
Enclosure (estimate)	$80
Total	**$1635**

Since the intention of producing Rhino is not to make a profit, but rather to aid research and training, one should be able to sell a complete Rhino system for $1700. This takes into account price fluctuations and administrative costs. Interestingly enough, this is the same price as the USRP N200 series, but Rhino offers far higher performance. Since the requirements that were identified in Chapter 1 stated that the board should cost less the $1800, the cost shown above meets the requirements.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

With Rhino fully designed, the board manufactured and the system tested, conclusions can now be drawn regarding the success of this project. This chapter therefore begins with a brief summary of the design process that took Rhino from concept to working system. The system architecture, test results and estimated costs are then analysed to determine whether the final Rhino design meets the specification given in Chapter 3. Finally, the work still remaining for Rhino is discussed.

8.1 SUMMARY OF THE RHINO DESIGN PROCESS

The design process began by determining the basic requirements for a low-cost FPGA board for training and research, by obtaining input from specialists in the relevant target application fields. Existing FPGA boards were then reviewed to (a) determine if any of these boards met the requirements for the proposed low-cost FPGA board, and (b) identify the main pros and cons of each board. Although none of these boards met all the necessary requirements, the analysis of the relative strengths and weaknesses allowed guidelines to be developed for Rhino which built on the existing good ideas, and avoided the common pitfalls.

In Chapter 3, these guidelines were combined with the original customer specification and the requirements analysis to formulate a detailed specification for Rhino. This detailed specification guided all subsequent design decisions. The main electrical components were then selected to meet this specification.

With the detailed specification defined and the main components selected, the actual design could begin. This was started by expanding the high-level block diagram to obtain a sub-system diagram. This was then transferred to schematics, where each sub-system block became a schematic sheet. Once the schematics were complete, the PCB design began. The theory behind the routing of high-speed digital signals was first reviewed. This then lead to the development of the routing rules and PCB stackup that were used to avoid signal integrity problems. Although the actual routing itself was outsourced to SunCad Designs, all the high-speed traces were checked for signal integrity problems via simulation.

Both hardware and software tools were developed to verify that Rhino was designed and manufactured correctly. A test rig was built to provide the necessary support infrastructure (power, fans, etc.) and debugging hardware. Software was developed for the processor and gateware for the FPGA to test all the on-board components. Finally, these tests were run on a manufactured Rhino board, with the board passing all tests.

8.2 CONCLUSIONS

Chapter 7 showed that Rhino was successfully manufactured and passed all the software tests, proving that all the sub-systems operated correctly at full performance. However, one still needs to determine whether the completed Rhino system meets all the requirements that were identified in Chapter 1 and the detailed specification that was developed in Chapter 3. Table 8.1 is given below to help determine this. It gives each of the points in the detailed specification from Section 3.1 and indicates whether or not they were met.

Table 8.1: Comparison of the Detailed Rhino Specification and the Manufactured Board

Category	Specification	Met?	Comments
Architecture	Processor used to control and program FPGA.	Met	Processor-FPGA interface passed all tests. U-Boot standalone application able to program FPGA.
FPGA	Spartan-6 FPGA with sufficient logic resources and 64 LVDS pairs for FMC cards.	Met	Largest Spartan-6 FPGA used, with sufficient logic resources. The two FMC connectors together provide 68 LVDS data pairs.
Processor	Any TI ARM processor that is supported by Linux.	Met	The AM3517 is used on Rhino, with Linux successfully running on it.
Memory	1GB of SDRAM for FPGA, with throughput of 20Gbps. Sufficient SDRAM for processor to run Linux.	Partially met	FPGA has 512MB of DDR3 SDRAM, with 25.6Gbps throughput. Can be upgraded to 1GB once bigger ICs available. The processor has 256MB SDRAM, enough to run Linux. All memory ICs were successfully tested.
Flash memory	Processor has flash memory for storing operating system.	Met	Processor is supplied with 256MB NAND flash, which passed all U-Boot tests.
FPGA Networking	A high-speed 20Gbps connection and a slower 1Gbps Ethernet connection	Met	Both the CX4 and the 1Gbps Ethernet connections passed the software tests.
Processor Networking	Ethernet connection for control	Met	Processor has 100Mbps Ethernet link, which has successfully been used for TFTP.
I/O Interface	Two 32-bit LVDS interfaces with industry-standard connector.	Met	Two industry-standard FMC connectors are used, which provide 34 LVDS data pairs per connector. The FMC connectors passed testing at 75MHz.
Clock Synchronisation	Mechanism to synchronise clocks on different boards to within 10ns.	Met	The 100Mbps Ethernet PHY implements the Precision Time Protocol, allowing clock synchronisation to within 10ns.
System Monitoring	All voltages, currents, temperatures and fan speeds must be monitored.	Met	The Rhino System Monitor application provided all the required readings, with acceptable accuracy.
Cost	Rhino must cost less than $1800 to manufacture	Met	The estimated cost price for a complete Rhino system is $1635.

The only aspect where there is partial compliance is the FPGA SDRAM, which can be met as soon as larger RAM ICs are available. It is assumed that the FMC interface meets the performance requirement of 400Mbps per line, but this can only be tested once a high-speed FMC card is available. Therefore, one can conclude from the table above that Rhino does in fact comply with the detailed specification that was drawn up at the beginning of the project. This means that Rhino also meets the customer specification, the performance

requirements, and the guidelines based on the review of existing boards.

8.3 FUTURE WORK FOR RHINO

Much work still remains for Rhino, with the obvious tasks being the development of processor firmware, drivers, gateware libraries for the FPGA and a development toolchain. However, since this thesis focuses on the hardware of Rhino, only the remaining hardware items will be discussed here.

8.3.1 Manufacture Next Revision of Rhino with the Necessary Modifications

The first task is to spin a new version of the PCB that fixes the hardware bugs in the current revision. These bugs were identified in Table 7.2, as well as their fixes. These changes have already been made on the schematics; they still however need to be transferred to the PCB design file, and then manufactured.

Besides these bugs, there are also a few non-critical changes that would further enhance the Rhino board. These changes have not yet been implemented, but it is strongly recommended that they are included in the next revision of the board:

- Currently, the virtual serial port used for the processor terminal closes whenever the board is turned off, as the FTDI USB-to-JTAG/I^2C/RS-232 chip is powered off the processor power supply. This causes the serial terminal application on the PC to crash. Powering this IC off the USB port instead would allow the serial port to remain open while the board power is cycled.
- The current for all the power, status and GPIO LEDs should be reduced. At present, the current through each LED s 10mA, consuming a total of 0.9W if all LEDs are switched on. If the LED current is halved, 0.45W will be saved, and the LEDs will still be clearly visible.
- The boot switch table on the silkscreen on the PCB should be rotated by 180°, so that it has the same orientation as the boot switches.

The last change to the board is to replace the FPGA DDR3 SDRAM ICs with 4Gb devices, if they are available.

8.3.2 Further Reduce Board Cost

Although the Rhino board meets cost requirements, the cost could be further reduced if the PCB was reduced from 16 layers to 14 layers. The details of how this can be achieved were described in Section 5.3.

8.3.3 Upgrade to SFP+ Network Connectors

The Rhino design team only became aware of the low availability of new CX4 network switches at a late stage in the design process. As a result, Rhino was manufactured with CX4 network connectors. It may be advantageous to consider changing to SFP+ network connectors in future revisions. However, before this decision can be made, it will be necessary to take a survey of typical Rhino users to determine what percentage uses CX4 switches and what percentage uses SFP+ switches. If the majority of users only have CX4 hardware, then the CX4 connectors will probably remain, as purchasing new network switches is an expensive exercise.

8.3.4 Build the Rhino Hide

The last remaining hardware task is to build a rack-mount enclosure for Rhino. The specification for this enclosure, named the Rhino Hide, was developed as part of this thesis. The proposed layout for the Rhino Hide is given in Appendix C, while the full specification is given on the attached CD.

APPENDIX A

I/O INTERFACE REQUIREMENTS

The input-output requirements for each target application for Rhino (namely radar, radio astronomy and bioinformatics) were given in Table 1.1 in Chapter 1. However, in order to determine the exact number of I/O lines necessary to interface the ADCs and DACs with the FPGA, as well as the required data rate of each line, some calculations are required. These calculations are outlined in this appendix.

There are two concepts that must be understood before the calculations can make sense. These are complex (I/Q) sampling and differential signalling. In complex sampling, both the original signal (in-phase, I) and a 90° phase-shifted version of the signal (quadrature, Q) are sampled. This allows a bandwidth equal to the sampling rate to be sampled (i.e. a 500MHz bandwidth can be sampled at 500MS/s). The disadvantage is that the number of bits required to represent each sample is doubled, as both the I and Q signals must be sampled.

Digital differential signalling uses two separate signals to convey a single bit. The value of the bit is determined by the voltage *difference* between the two signals. Although this requires twice the number of signals as single-ended signalling, it does improve signal integrity and noise immunity, as we look only at the difference between two signals, and not the absolute value of each one. Because of this, low-voltage differential signalling (LVDS), one of the more popular differential signalling standards, is used for the interface between the ADCs/DACs and the FPGA.

A.1 I/O REQUIREMENTS FOR RADAR APPLICATIONS

For the radar I/O calculations, we assume that:

- There is one ADC and one DAC connected to the Rhino board.
- Both the ADC and DAC have 12-bit resolution, and support a 400MHz bandwidth (as per the requirements in Table 1.1).
- I/Q sampling is used.
- The ADCs and DACs use LVDS for the digital data lines.

Therefore:

- The sampling rate is 400MS/s.

- Each ADC/DAC has 24 LVDS pairs (12 bits for I and 12 bits for Q), each pair with a data rate of 400Mbps.
- Since the board needs to support both an ADC and a DAC, two separate 24-bit buses are required.

A.2 I/O Requirements for Radio Astronomy Applications

For the radio astronomy I/O calculations, we assume that:

- There are two ADCs connected to the Rhino board.
- The ADCs have 8-bit resolution, and support a 500MHz bandwidth (as per the requirements in Table 1.1).
- I/Q sampling is used.
- A demux factor of two is used in the ADC. This means that the ADC demultiplexes the digital output signals by a factor of two, resulting in twice as many digital outputs from the ADC, but each running at half the original data rate.
- The ADCs use LVDS for the digital data lines.

Therefore:

- The sampling rate is 500MS/s.
- Number of LVDS pairs is given by:

  (sampling resolution) * (demux factor) * 2 (for I and Q)

  = (8 bits) * 2 * 2

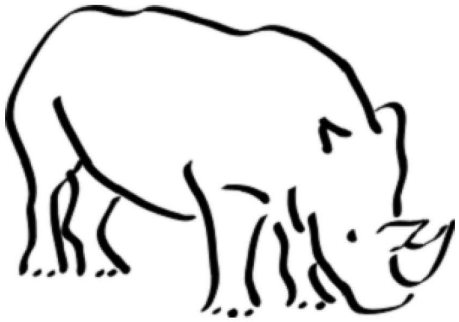
  = 32 LVDS pairs per ADC
- Since a demux factor of two is used, the data rate is half the sampling rate. Therefore, each LVDS pair runs at 250Mbps.
- Since the board needs to support two ADCs, two separate 32-bit LVDS buses are required.

APPENDIX B

FULL RHINO SCHEMATICS

The complete Rhino schematics can be found on the following pages. The contents page at the beginning of the schematics should be helpful when trying to navigate to a particular schematic.

It is strongly suggested that these schematics be read in conjunction with *Chapter 4: Schematic-level Hardware Design of Rhino*.



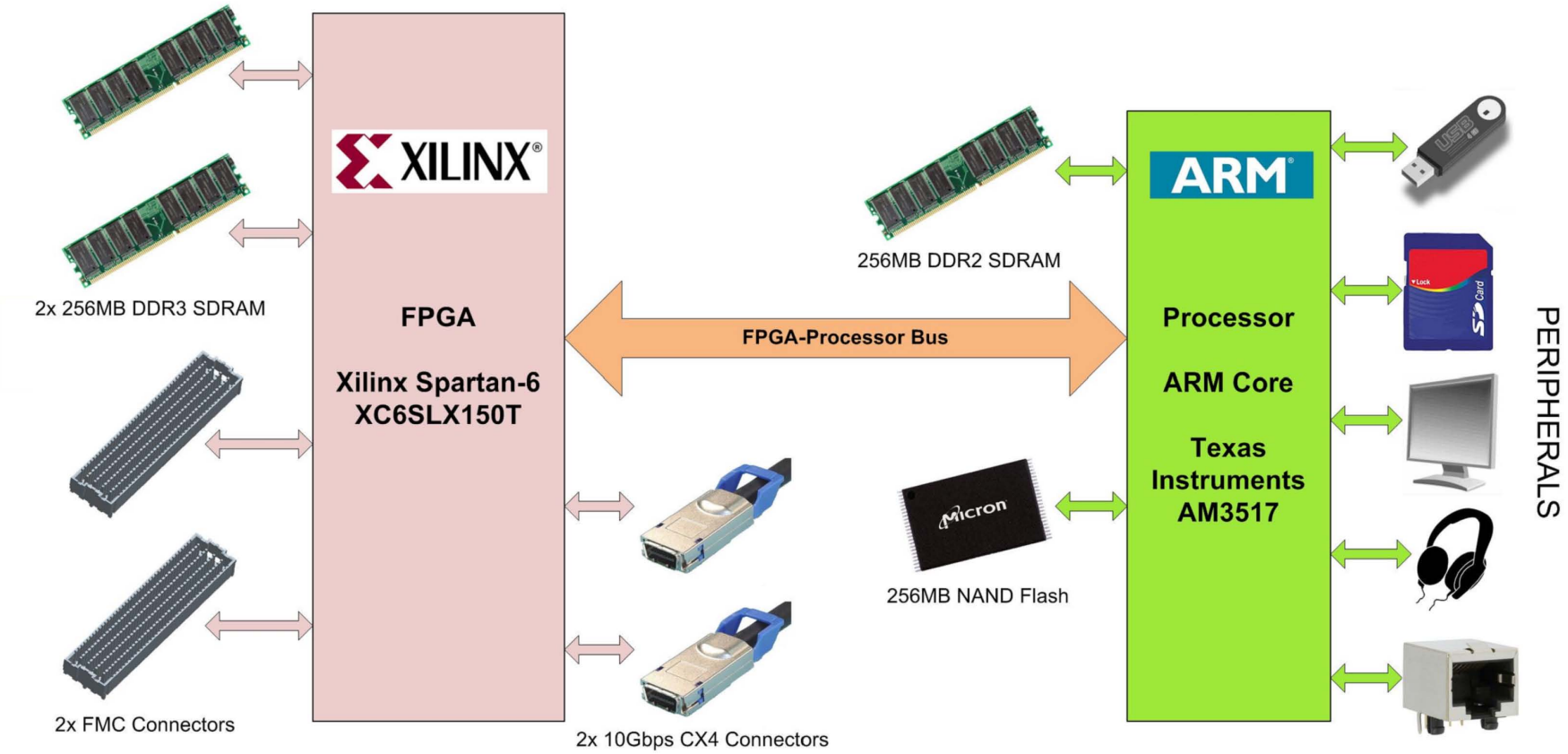
RHINO

Reconfigurable Hardware Interface for computatioN and radiO

UCT SDR Research Group

Published: 11/02/2011
Revision: 1.1

Schematics

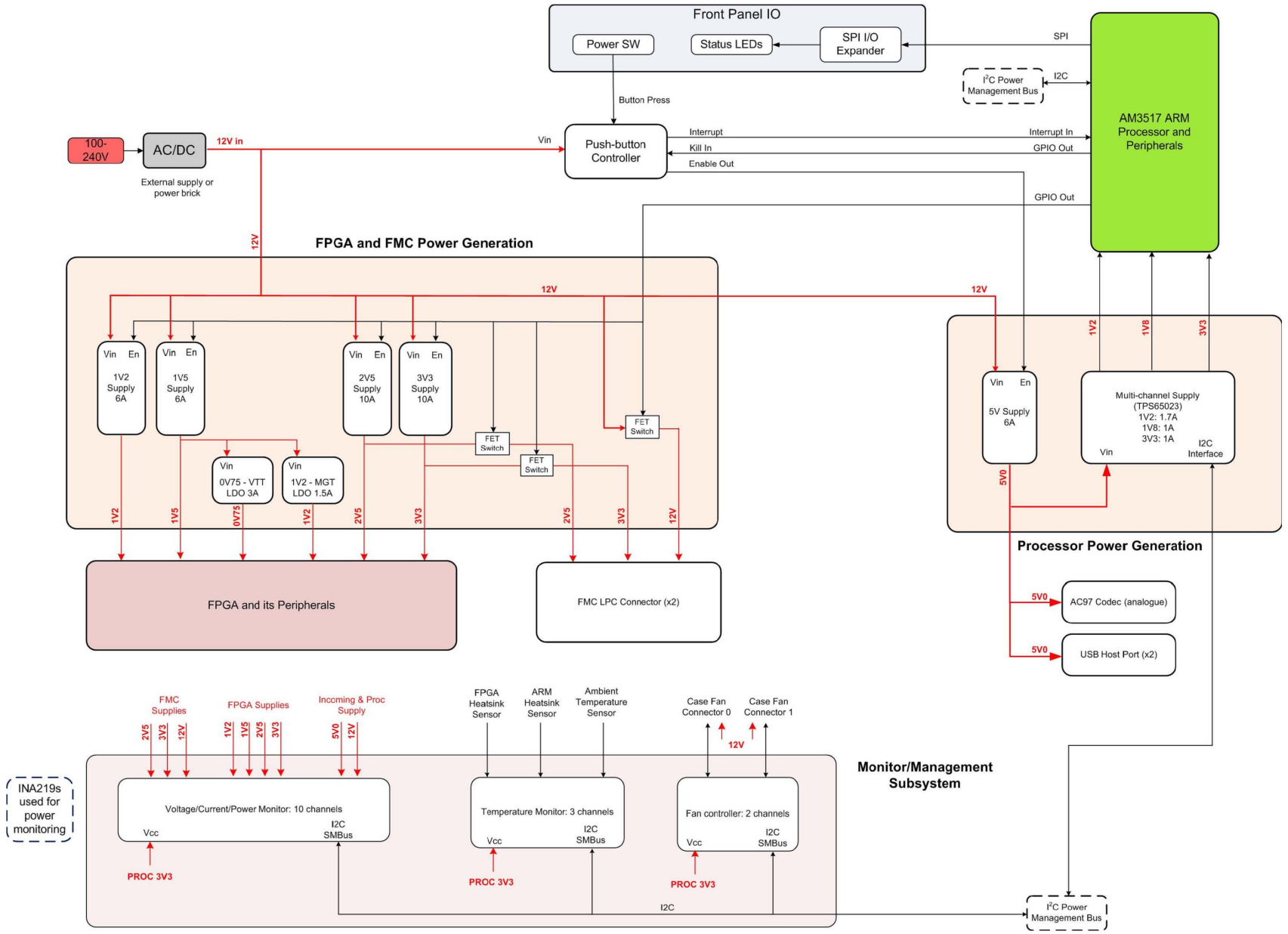


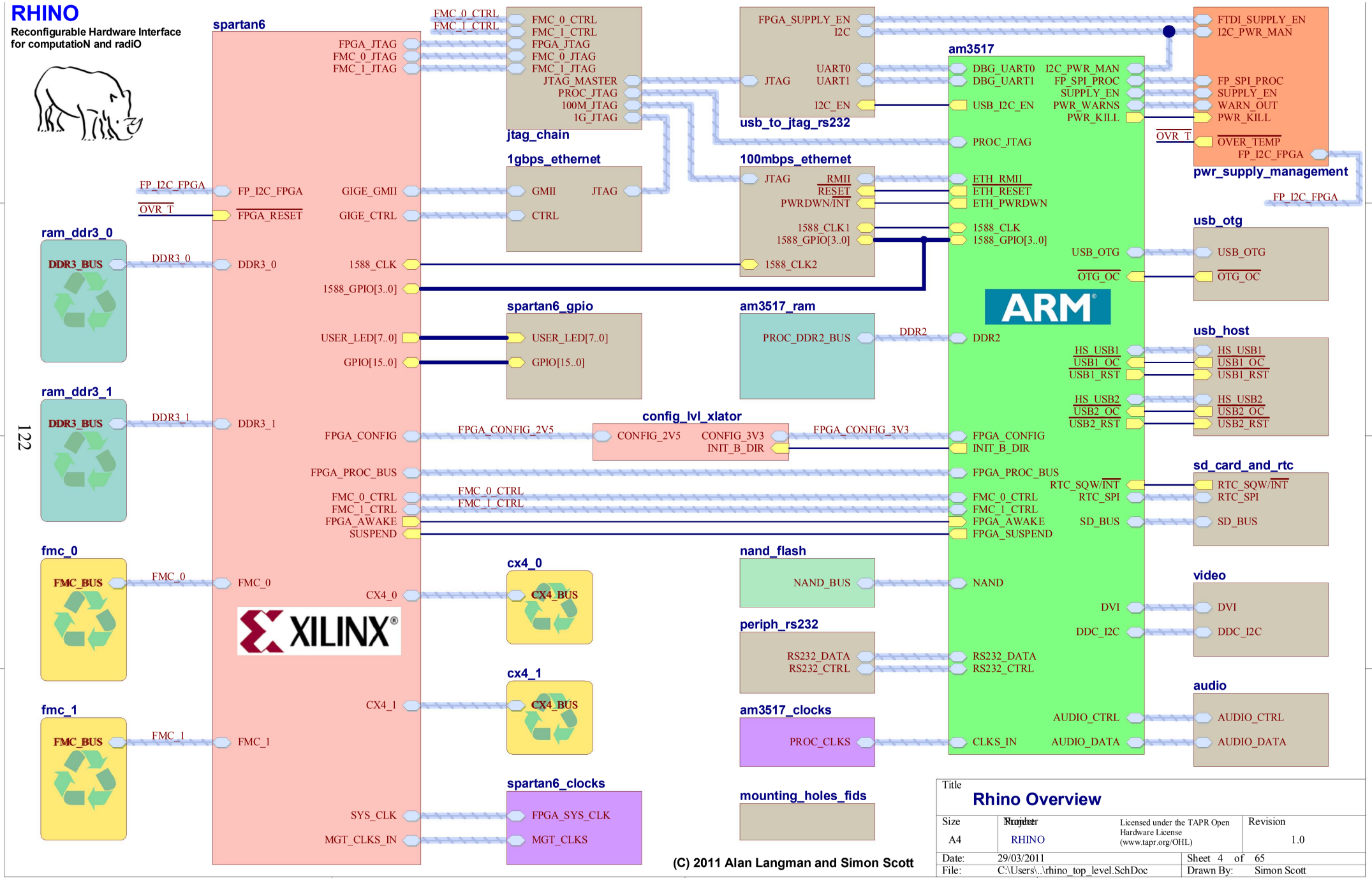
119

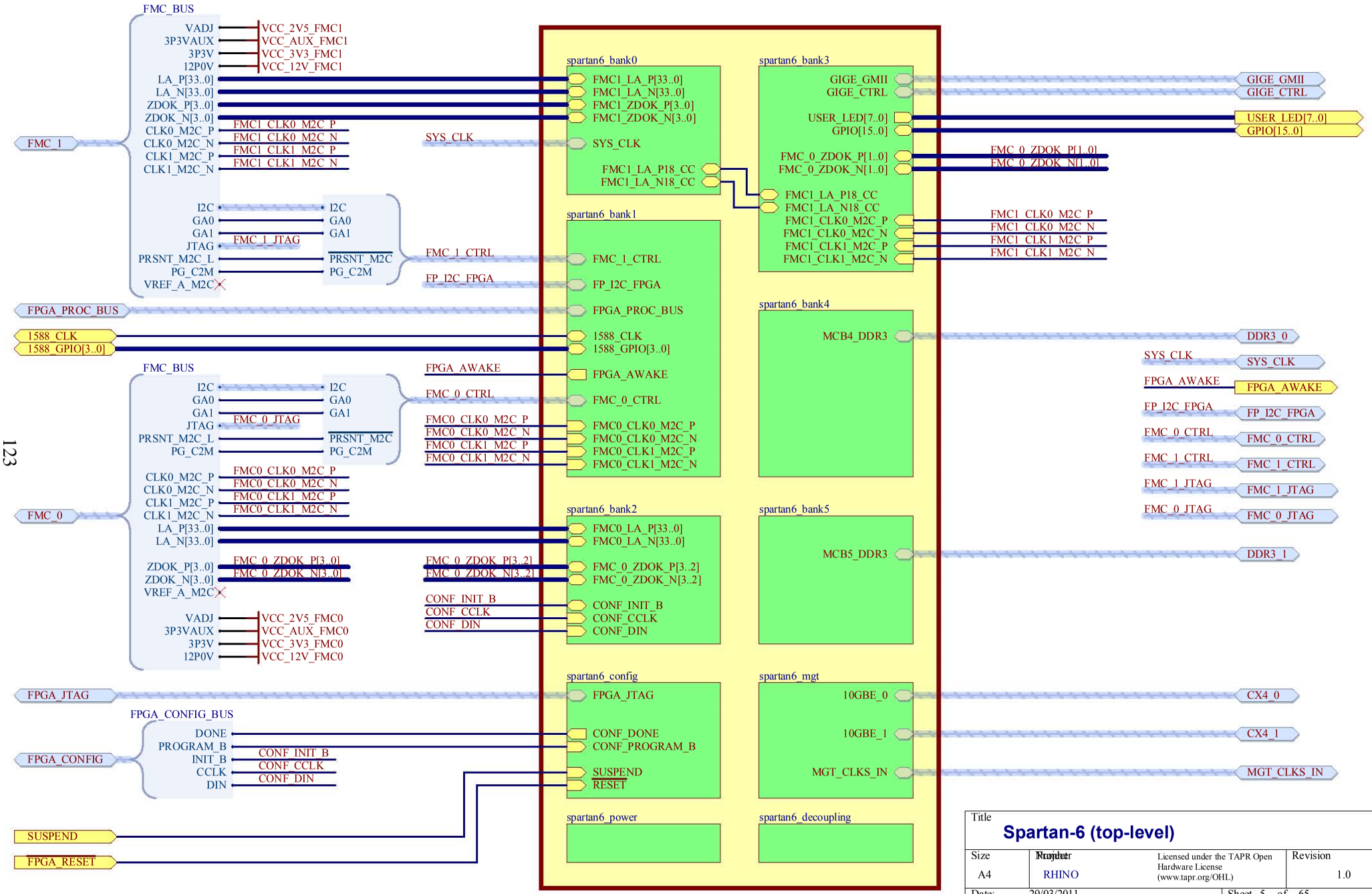
rhino_contents
rhino_power_distrib
rhino_top_level

Table of Contents

Rhino Power Distribution and Monitoring

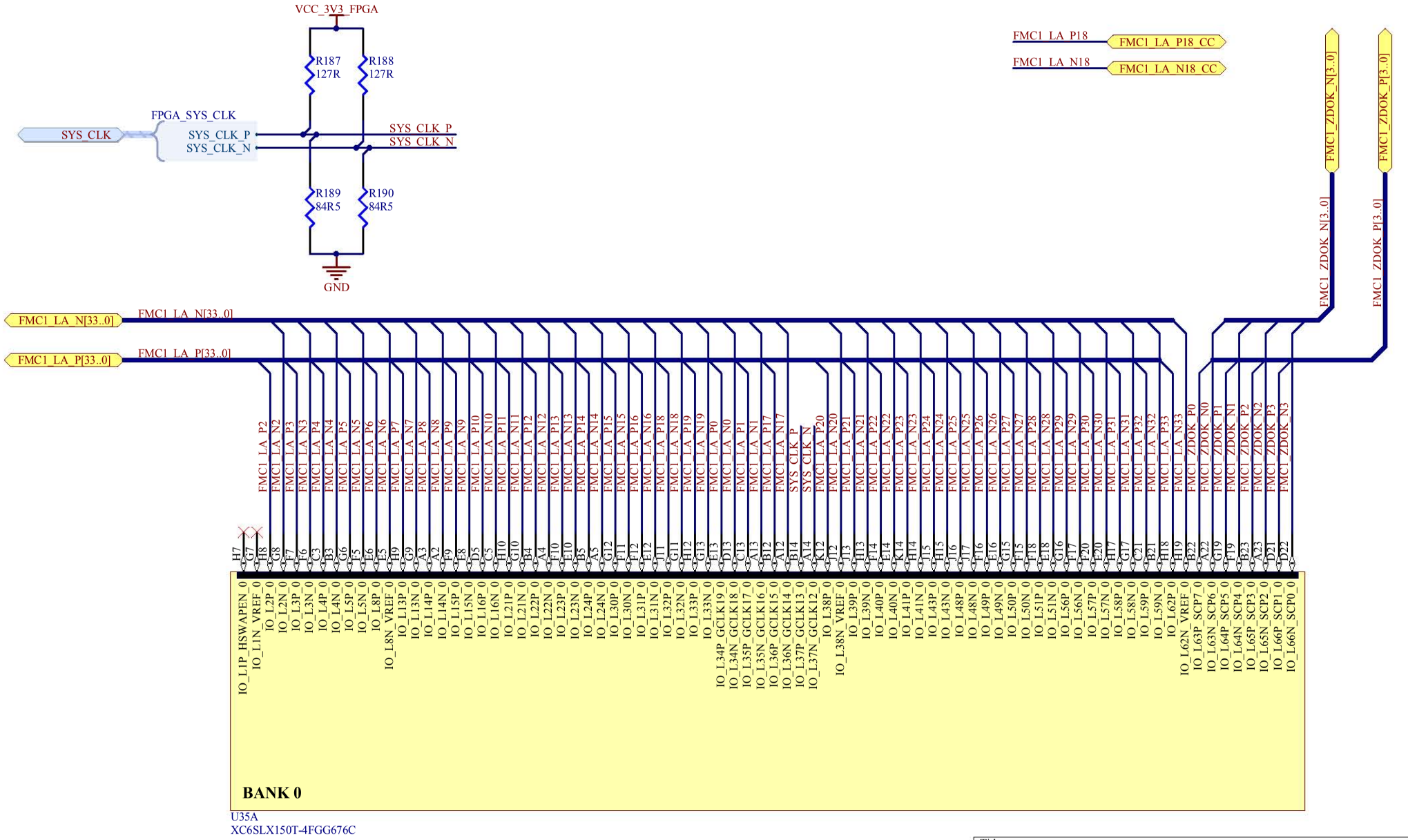



RHINO
Reconfigurable Hardware Interface for computatioN and radiO
spartan6
jtag_chain
usb_to_jtag_rs232
1gbps_ethernet
100mbps_ethernet
am3517
pwr_supply_management
ram_ddr3_0
ram_ddr3_1
spartan6_gpio
am3517_ram
usb_otg
usb_host
config_lvl_xlator
sd_card_and_rtc
fmc_0
fmc_1
cx4_0
cx4_1
nand_flash
periph_rs232
video
audio
am3517_clocks
spartan6_clocks
mounting_holes_fids
XILINX
ARM
122
(C) 2011 Alan Langman and Simon Scott
Rhino Overview
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Size A4
Project RHINO
Revision 1.0
Date: 29/03/2011
Sheet 4 of 65
File: C:\Users\..\rhino_top_level.SchDoc
Drawn By: Simon Scott



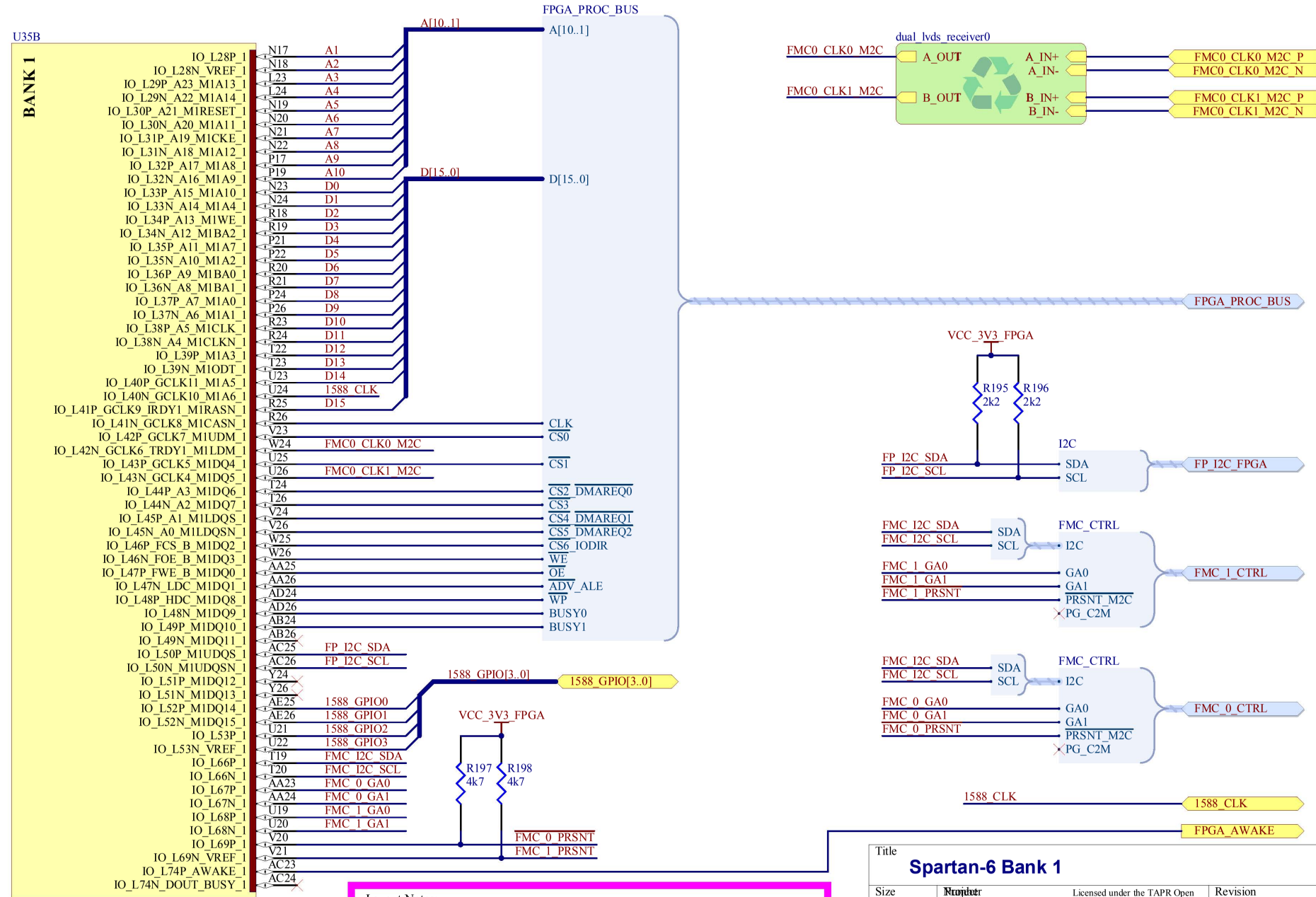
123

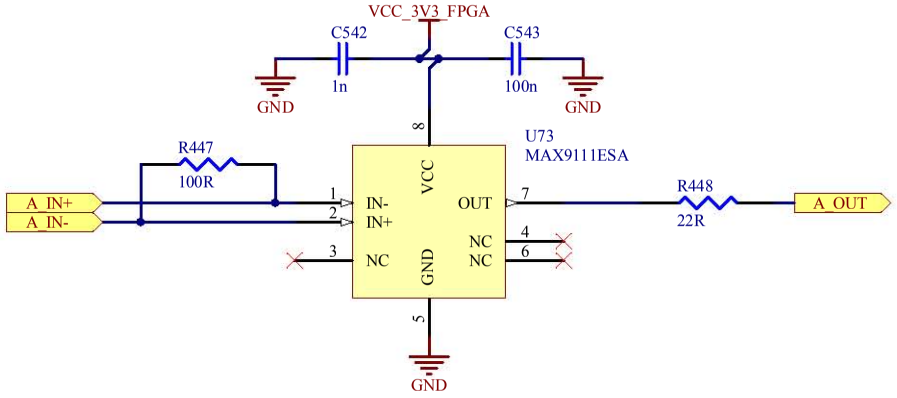
Title	**Spartan-6 (top-level)**
Size: A4	Project: RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011	Sheet 5 of 65
File: C:\Users\..\spartan6.SchDoc	Drawn By: Simon Scott

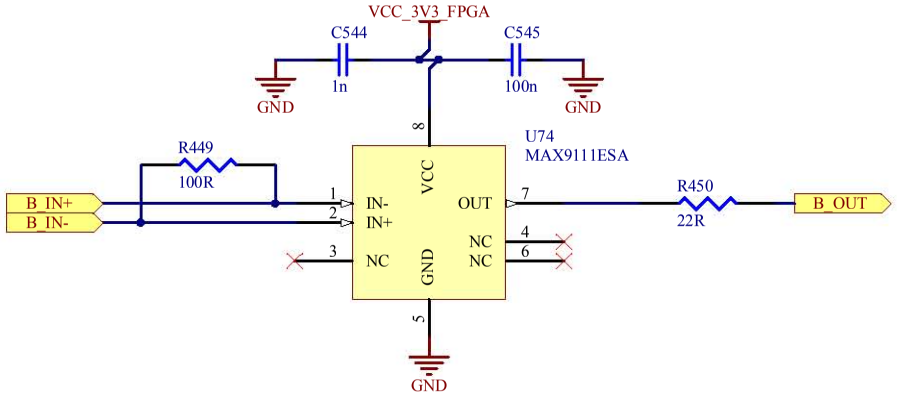


Note:
1.) FMC1_LA_0, FMC1_LA_1 and FMC1_LA_17 can all be used as mezzanine-to-carrier clocks
2.) Place the 127R and 84R5 termination resistors, on the SYS_CLK lines, as close to the Spartan-6 clock input pins as possible
3.) The SYS_CLK pins must be configured as LVPECL_33 inputs

Title	**Spartan-6 Bank 0**		
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 6 of 65	
File: C:\Users\..\spartan6_bank0.SchDoc		Drawn By: Simon Scott	

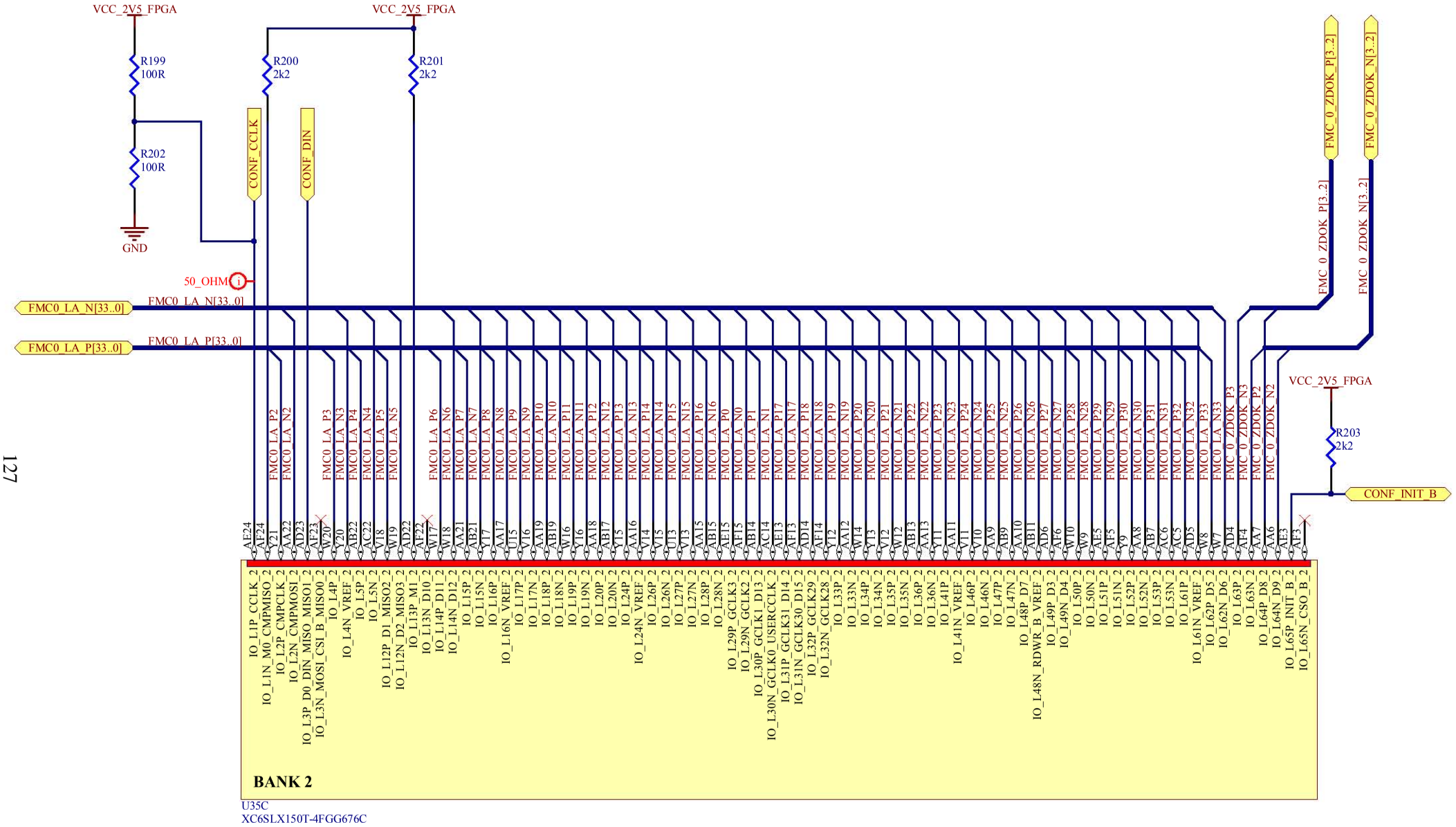

FPGA_PROC_BUS
U35B
BANK 1
A[10..1]
A[10..1]
D[15..0]
D[15..0]
IO_L28P_1
IO_L28N_VREF_1
IO_L29P_A23_M1A13_1
IO_L29N_A22_M1A14_1
IO_L30P_A21_M1RESET_1
IO_L30N_A20_M1A11_1
IO_L31P_A19_M1CKE_1
IO_L31N_A18_M1A12_1
IO_L32P_A17_M1A8_1
IO_L32N_A16_M1A9_1
IO_L33P_A15_M1A10_1
IO_L33N_A14_M1A4_1
IO_L34P_A13_M1WE_1
IO_L34N_A12_M1BA2_1
IO_L35P_A11_M1A7_1
IO_L35N_A10_M1A2_1
IO_L36P_A9_M1BA0_1
IO_L36N_A8_M1BA1_1
IO_L37P_A7_M1A0_1
IO_L37N_A6_M1A1_1
IO_L38P_A5_M1CLK_1
IO_L38N_A4_M1CLKN_1
IO_L39P_M1A3_1
IO_L39N_M1ODT_1
IO_L40P_GCLK11_M1A5_1
IO_L40N_GCLK10_M1A6_1
IO_L41P_GCLK9_IRDY1_M1RASN_1
IO_L41N_GCLK8_M1CASN_1
IO_L42P_GCLK7_M1UDM_1
IO_L42N_GCLK6_TRDY1_M1LDM_1
IO_L43P_GCLK5_M1DQ4_1
IO_L43N_GCLK4_M1DQ5_1
IO_L44P_A3_M1DQ6_1
IO_L44N_A2_M1DQ7_1
IO_L45P_A1_M1LDQS_1
IO_L45N_A0_M1LDQSN_1
IO_L46P_FCS_B_M1DQ2_1
IO_L46N_FOE_B_M1DQ3_1
IO_L47P_FWE_B_M1DQ0_1
IO_L47N_LDC_M1DQ1_1
IO_L48P_HDC_M1DQ8_1
IO_L48N_M1DQ9_1
IO_L49P_M1DQ10_1
IO_L49N_M1DQ11_1
IO_L50P_M1UDQS_1
IO_L50N_M1UDQSN_1
IO_L51P_M1DQ12_1
IO_L51N_M1DQ13_1
IO_L52P_M1DQ14_1
IO_L52N_M1DQ15_1
IO_L53P_1
IO_L53N_VREF_1
IO_L66P_1
IO_L66N_1
IO_L67P_1
IO_L67N_1
IO_L68P_1
IO_L68N_1
IO_L69P_1
IO_L69N_VREF_1
IO_L74P_AWAKE_1
IO_L74N_DOUT_BUSY_1
XC6SLX150T-4FGG676C
N17 A1
N18 A2
L23 A3
L24 A4
N19 A5
N20 A6
N21 A7
N22 A8
P17 A9
P19 A10
N23 D0
N24 D1
R18 D2
R19 D3
P21 D4
P22 D5
R20 D6
R21 D7
P24 D8
P26 D9
R23 D10
R24 D11
T22 D12
T23 D13
U23 D14
U24 1588_CLK
R25 D15
R26
V23 CLK
W24 CS0
U25
U26 CS1
T24
T26 CS2_DMAREQ0
V24 CS3
V26 CS4_DMAREQ1
W25 CS5_DMAREQ2
W26 CS6_IODIR
AA25 WE
AA26 OE
AD24 ADV_ALE
AD26 WP
AB24 BUSY0
AB26 BUSY1
AC25 FP_I2C_SDA
AC26 FP_I2C_SCL
Y24
Y26
AE25 1588_GPIO0
AE26 1588_GPIO1
U21 1588_GPIO2
U22 1588_GPIO3
T19 FMC_I2C_SDA
T20 FMC_I2C_SCL
AA23 FMC_0_GA0
AA24 FMC_0_GA1
U19 FMC_1_GA0
U20 FMC_1_GA1
V20
V21 FMC_0_PRSNT
AC23 FMC_1_PRSNT
AC24
1588_GPIO[3..0]
1588_GPIO[3..0]
VCC_3V3_FPGA
R197 4k7
R198 4k7
FMC0_CLK0_M2C
FMC0_CLK1_M2C
dual_lvds_receiver0
A_OUT
B_OUT
A_IN+
A_IN-
B_IN+
B_IN-
FMC0_CLK0_M2C_P
FMC0_CLK0_M2C_N
FMC0_CLK1_M2C_P
FMC0_CLK1_M2C_N
FPGA_PROC_BUS
VCC_3V3_FPGA
R195 2k2
R196 2k2
I2C
FP_I2C_SDA
FP_I2C_SCL
SDA
SCL
FP_I2C_FPGA
FMC_CTRL
FMC_I2C_SDA
FMC_I2C_SCL
FMC_1_GA0
FMC_1_GA1
FMC_1_PRSNT
GA0
GA1
PRSNT_M2C
PG_C2M
FMC_1_CTRL
FMC_CTRL
FMC_I2C_SDA
FMC_I2C_SCL
FMC_0_GA0
FMC_0_GA1
FMC_0_PRSNT
GA0
GA1
PRSNT_M2C
PG_C2M
FMC_0_CTRL
1588_CLK
1588_CLK
FPGA_AWAKE
Layout Notes:
1.) Place the MAX9111 LVDS receiver IC as close to the Spartan-6 as possible, to minimise noise on the clock lines
Title
Spartan-6 Bank 1
Size
A4
Number
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date:
29/03/2011
Sheet 7 of 65
File:
C:\Users\..\spartan6_bank1.SchDoc
Drawn By:
Simon Scott





Layout Notes:
1.) Place the 100R and 22R termination resistors as close to the pins of the MAX9111ESA IC as possible
2.) Place the 100nF and 1nF decoupling caps as close to the IC as possible

Title: **Dual LVDS Receiver**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 8 of 65	
File: C:\Users\..\dual_lvds_receiver.SchDoc		Drawn By: Simon Scott	

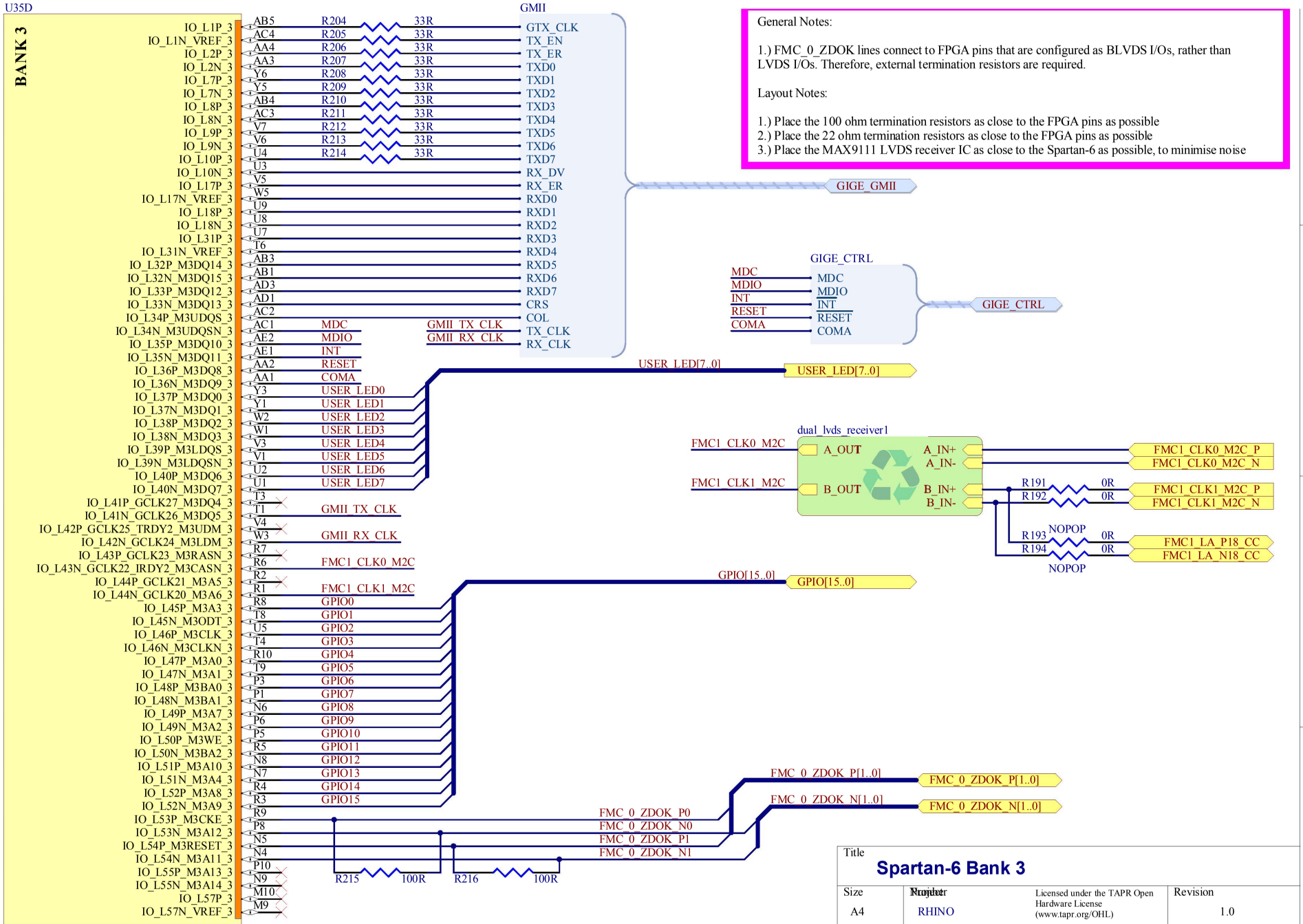


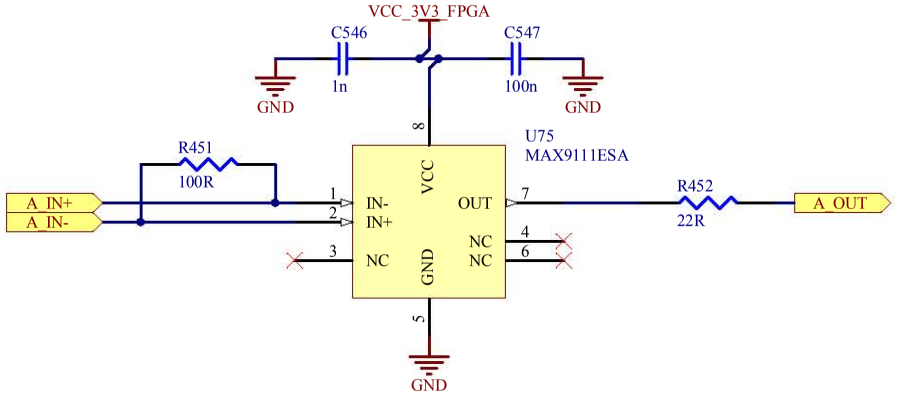
127

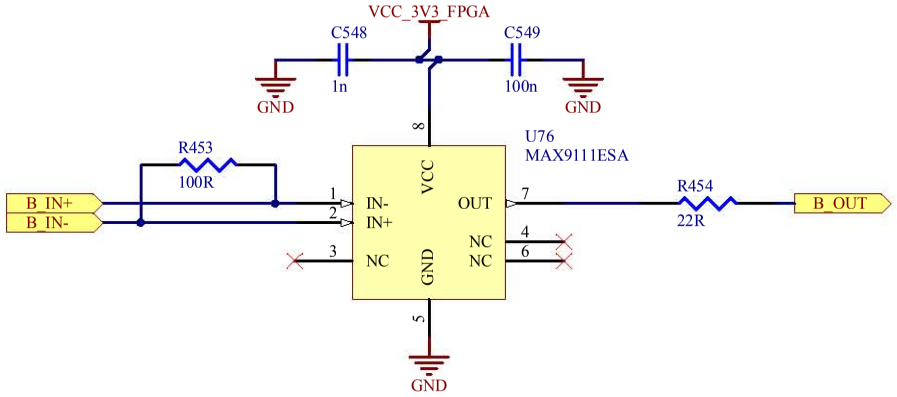
Layout Note:
1.) Route the CONF_CCLK line as a 50 Ohm controlled impedance transmission line
2.) Place the two 100 ohm resistors on the CONF_CCLK line as close to the FPGA pin as possible

General Note:
1.) FMC0_LA_0, 1, 17 and 18 can all be used as mezzanine-to-carrier clocks

Title: Spartan-6 Bank 2			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 9 of 65	
File: C:\Users\..\spartan6_bank2.SchDoc		Drawn By: Simon Scott	

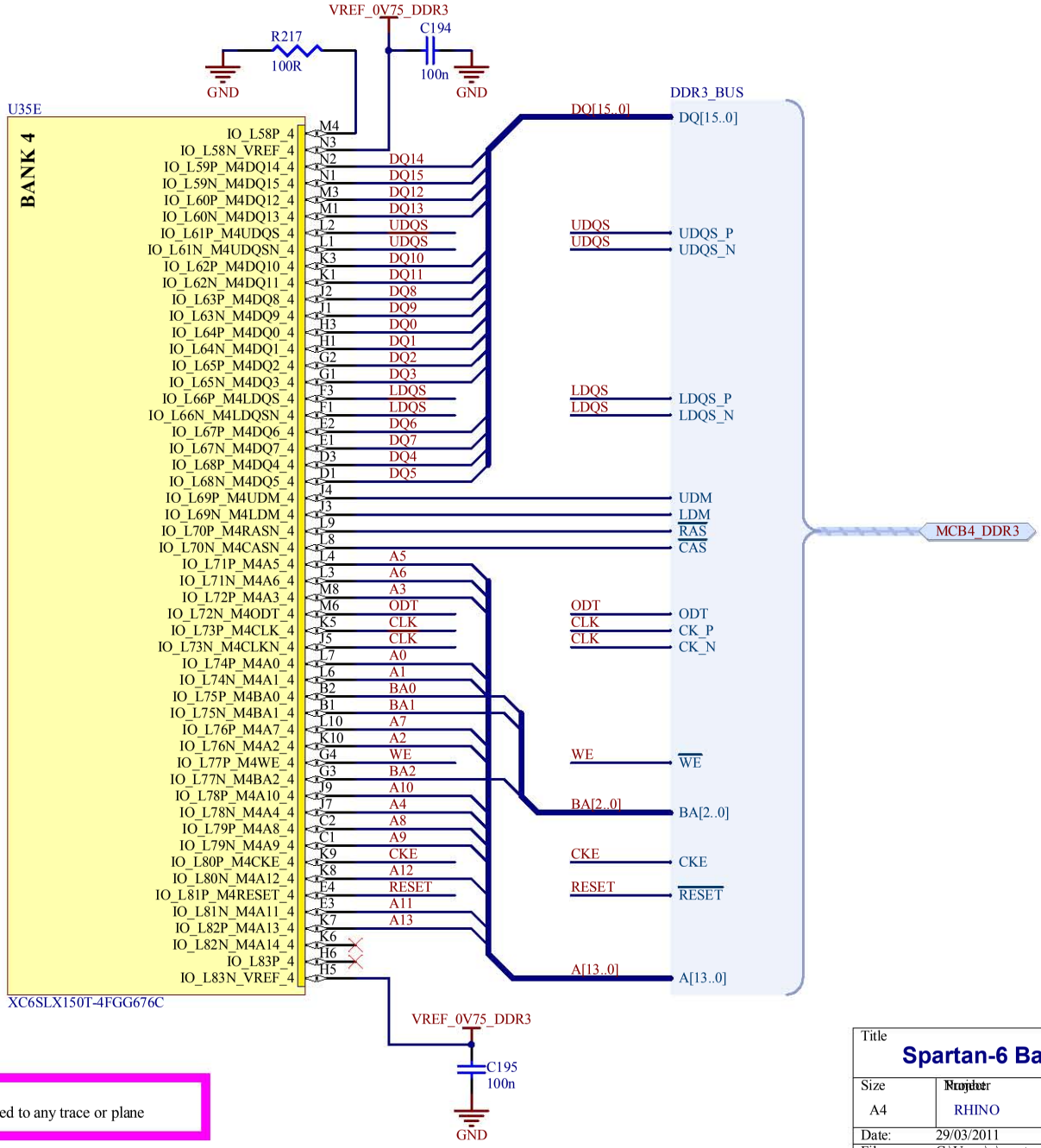

U35D
BANK 3
XC6SLX150T-4FGG676C
GMII
GTX_CLK
TX_EN
TX_ER
TXD0
TXD1
TXD2
TXD3
TXD4
TXD5
TXD6
TXD7
RX_DV
RX_ER
RXD0
RXD1
RXD2
RXD3
RXD4
RXD5
RXD6
RXD7
CRS
COL
TX_CLK
RX_CLK
GIGE_GMII
GIGE_CTRL
MDC
MDIO
INT
RESET
COMA
USER_LED[7..0]
USER_LED0
USER_LED1
USER_LED2
USER_LED3
USER_LED4
USER_LED5
USER_LED6
USER_LED7
GMII_TX_CLK
GMII_RX_CLK
FMC1_CLK0_M2C
FMC1_CLK1_M2C
GPIO[15..0]
GPIO0
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
GPIO8
GPIO9
GPIO10
GPIO11
GPIO12
GPIO13
GPIO14
GPIO15
R204 33R
R205 33R
R206 33R
R207 33R
R208 33R
R209 33R
R210 33R
R211 33R
R212 33R
R213 33R
R214 33R
dual_lvds_receiver1
A_OUT
B_OUT
A_IN+
A_IN-
B_IN+
B_IN-
FMC1_CLK0_M2C_P
FMC1_CLK0_M2C_N
FMC1_CLK1_M2C_P
FMC1_CLK1_M2C_N
R191 0R
R192 0R
R193 NOPOP 0R
R194 0R NOPOP
FMC1_LA_P18_CC
FMC1_LA_N18_CC
FMC_0_ZDOK_P0
FMC_0_ZDOK_N0
FMC_0_ZDOK_P1
FMC_0_ZDOK_N1
FMC_0_ZDOK_P[1..0]
FMC_0_ZDOK_N[1..0]
R215 100R
R216 100R
General Notes:
1.) FMC_0_ZDOK lines connect to FPGA pins that are configured as BLVDS I/Os, rather than LVDS I/Os. Therefore, external termination resistors are required.
Layout Notes:
1.) Place the 100 ohm termination resistors as close to the FPGA pins as possible
2.) Place the 22 ohm termination resistors as close to the FPGA pins as possible
3.) Place the MAX9111 LVDS receiver IC as close to the Spartan-6 as possible, to minimise noise
Title
Spartan-6 Bank 3
Size
A4
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date: 29/03/2011
Sheet 10 of 65
File: C:\Users\..\spartan6_bank3.SchDoc
Drawn By: Simon Scott





Layout Notes:
1.) Place the 100R and 22R termination resistors as close to the pins of the MAX9111ESA IC as possible
2.) Place the 100nF and 1nF decoupling caps as close to the IC as possible

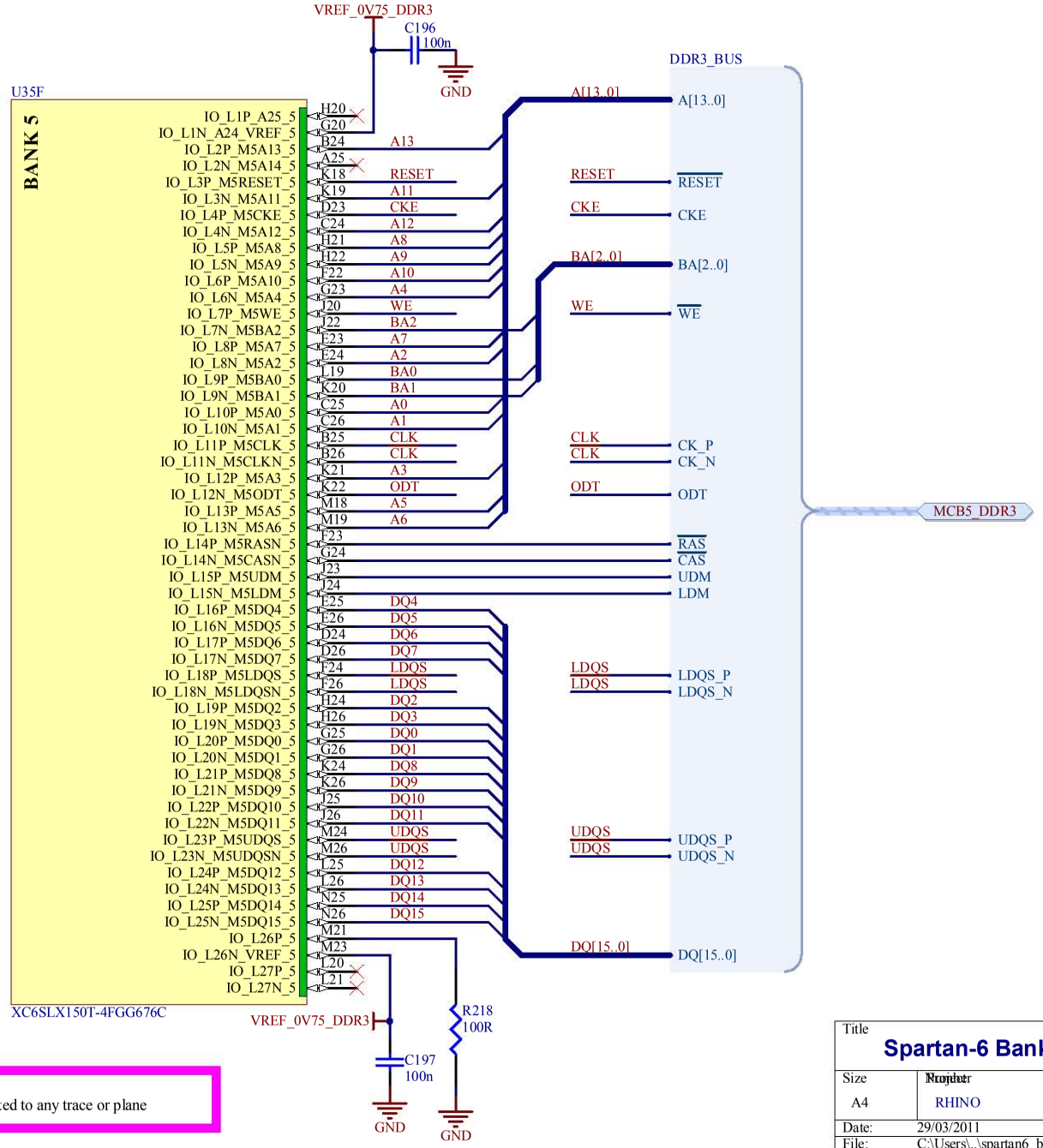
Title: **Dual LVDS Receiver**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 11 of 65	
File: C:\Users\..\dual_lvds_receiver.SchDoc		Drawn By: Simon Scott	



130

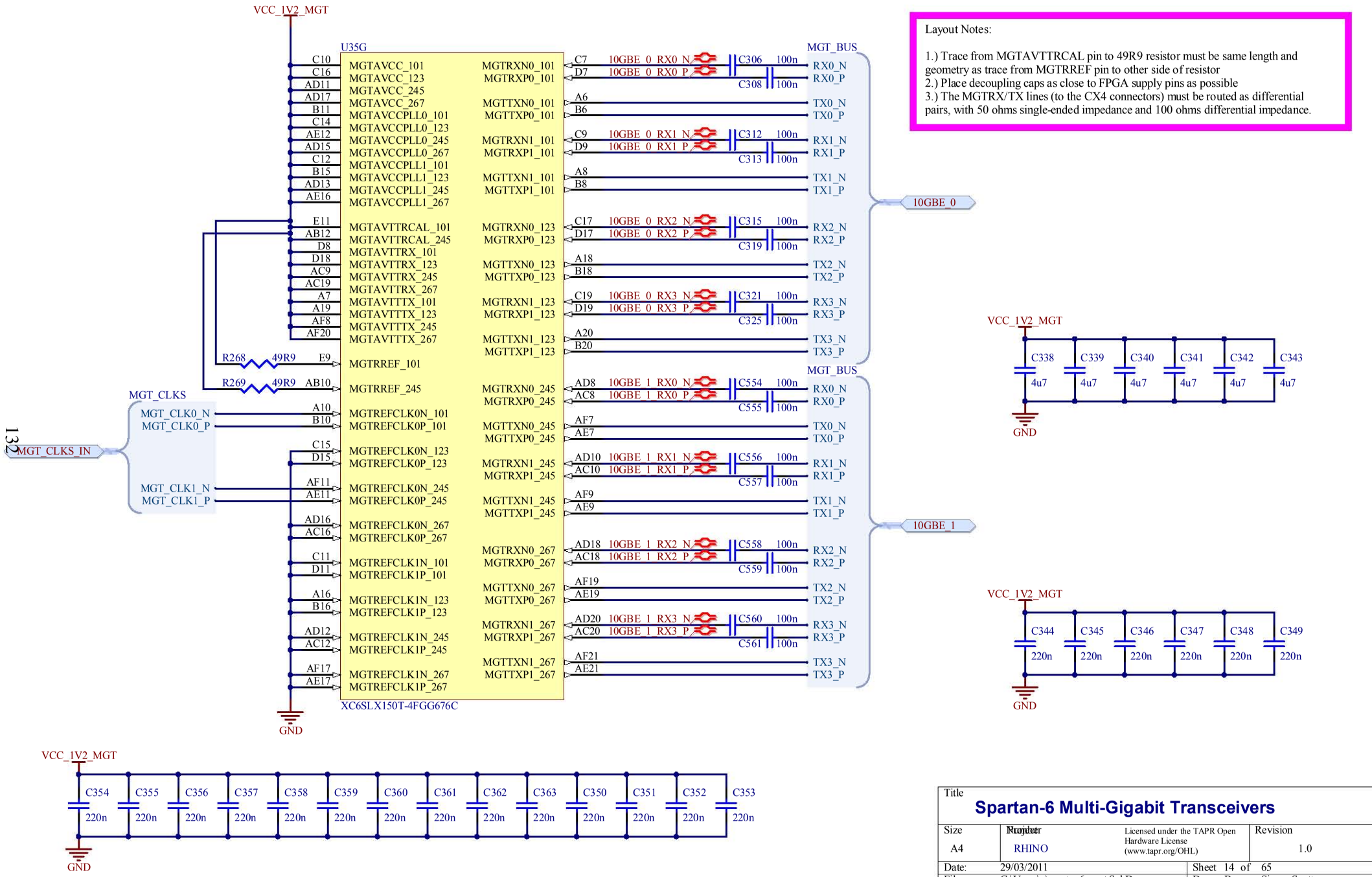
Layout Notes:
1.) Ensure that ball H6 (a NO CONNECT pin) is not connected to any trace or plane

Title: Spartan-6 Bank 4			
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 12 of 65	
File: C:\Users\..\spartan6_bank4.SchDoc		Drawn By: Simon Scott	



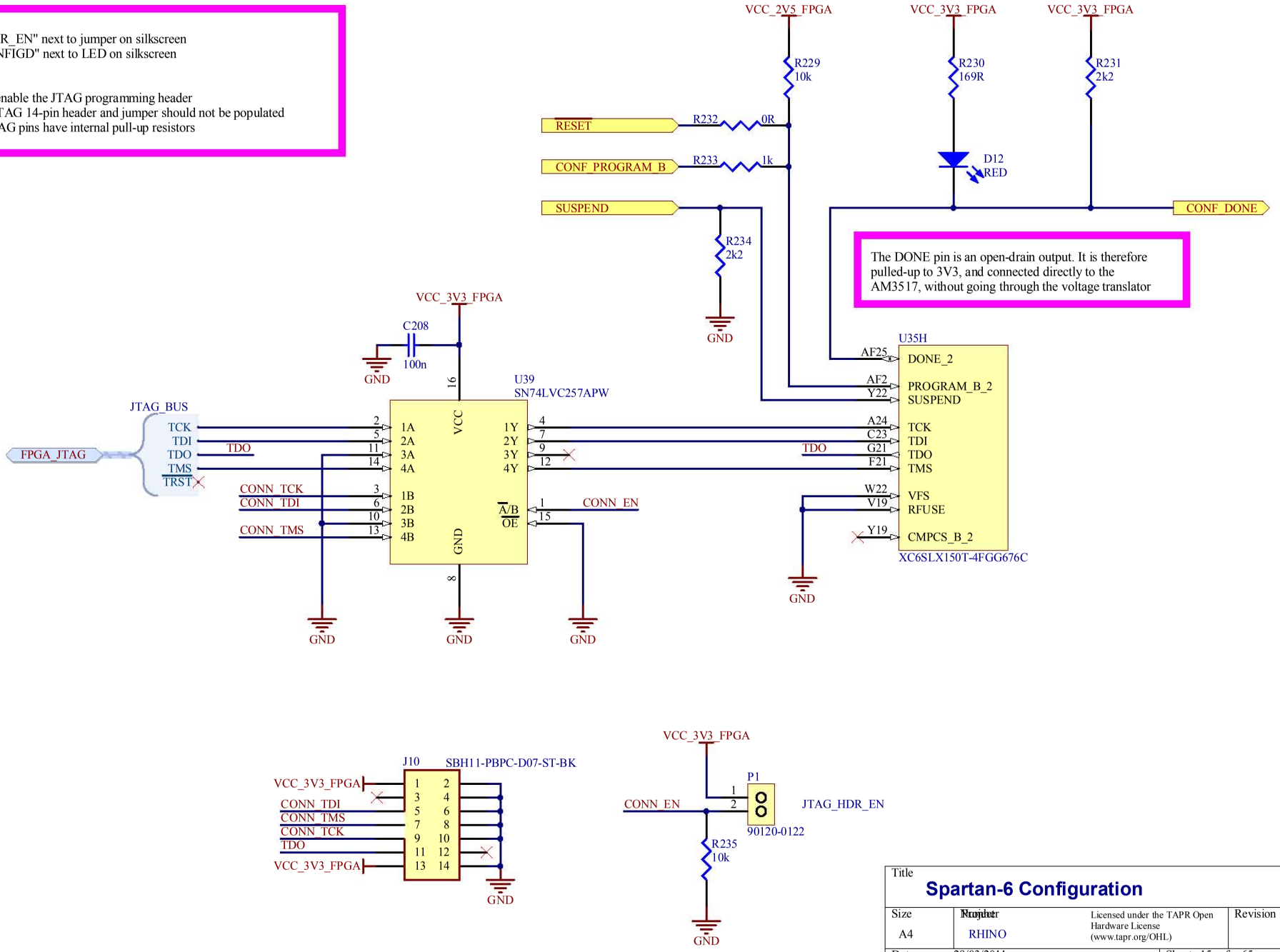
Layout Notes:
1.) Ensure that ball H20 (a NO CONNECT pin) is not connected to any trace or plane

Title	Spartan-6 Bank 5		
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 13 of 65	
File: C:\Users\..\spartan6_bank5.SchDoc		Drawn By: Simon Scott	

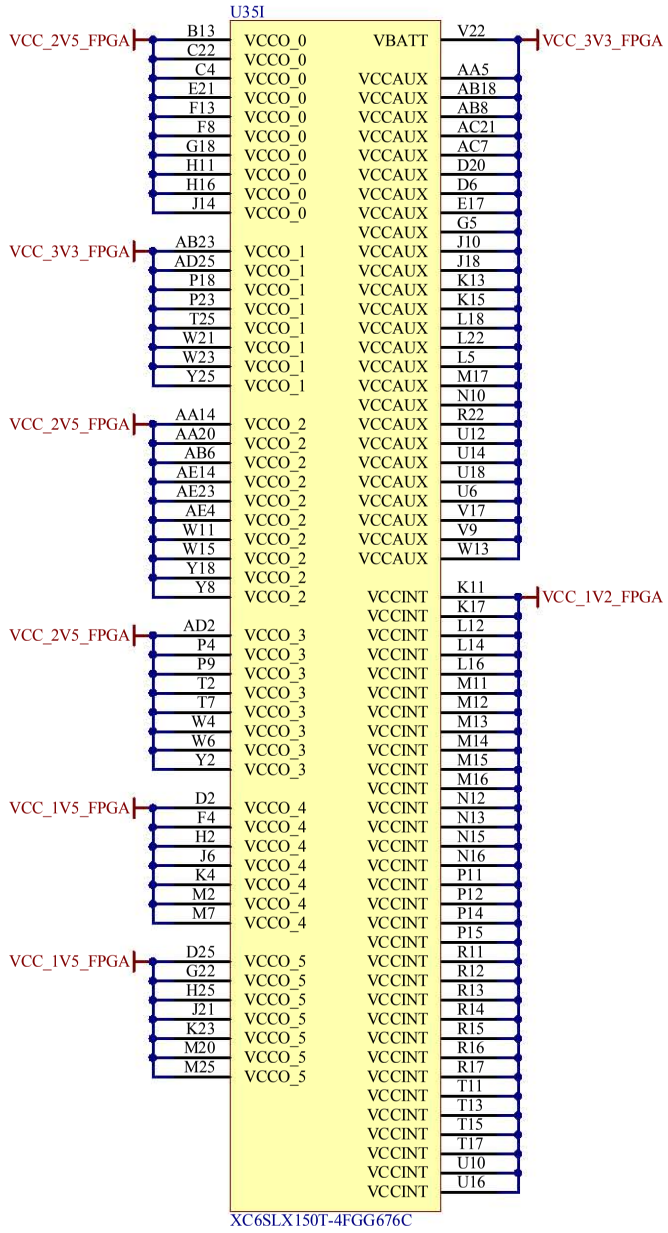

Layout Notes:
1.) Trace from MGTAVTTRCAL pin to 49R9 resistor must be same length and geometry as trace from MGTRREF pin to other side of resistor
2.) Place decoupling caps as close to FPGA supply pins as possible
3.) The MGTRX/TX lines (to the CX4 connectors) must be routed as differential pairs, with 50 ohms single-ended impedance and 100 ohms differential impedance.
VCC_1V2_MGT
U35G
MGT_BUS
10GBE_0
10GBE_1
MGT_CLKS
MGT_CLKS_IN
132
R268 49R9
R269 49R9
MGTRREF_101
MGTRREF_245
MGTAVCC_101
MGTAVCC_123
MGTAVCC_245
MGTAVCC_267
MGTAVCCPLL0_101
MGTAVCCPLL0_123
MGTAVCCPLL0_245
MGTAVCCPLL0_267
MGTAVCCPLL1_101
MGTAVCCPLL1_123
MGTAVCCPLL1_245
MGTAVCCPLL1_267
MGTAVTTRCAL_101
MGTAVTTRCAL_245
MGTAVTTRX_101
MGTAVTTRX_123
MGTAVTTRX_245
MGTAVTTRX_267
MGTAVTTTX_101
MGTAVTTTX_123
MGTAVTTTX_245
MGTAVTTTX_267
MGTREFCLK0N_101
MGTREFCLK0P_101
MGTREFCLK0N_123
MGTREFCLK0P_123
MGTREFCLK0N_245
MGTREFCLK0P_245
MGTREFCLK0N_267
MGTREFCLK0P_267
MGTREFCLK1N_101
MGTREFCLK1P_101
MGTREFCLK1N_123
MGTREFCLK1P_123
MGTREFCLK1N_245
MGTREFCLK1P_245
MGTREFCLK1N_267
MGTREFCLK1P_267
MGTRXN0_101
MGTRXP0_101
MGTTXN0_101
MGTTXP0_101
MGTRXN1_101
MGTRXP1_101
MGTTXN1_101
MGTTXP1_101
MGTRXN0_123
MGTRXP0_123
MGTTXN0_123
MGTTXP0_123
MGTRXN1_123
MGTRXP1_123
MGTTXN1_123
MGTTXP1_123
MGTRXN0_245
MGTRXP0_245
MGTTXN0_245
MGTTXP0_245
MGTRXN1_245
MGTRXP1_245
MGTTXN1_245
MGTTXP1_245
MGTRXN0_267
MGTRXP0_267
MGTTXN0_267
MGTTXP0_267
MGTRXN1_267
MGTRXP1_267
MGTTXN1_267
MGTTXP1_267
10GBE_0_RX0_N
10GBE_0_RX0_P
10GBE_0_RX1_N
10GBE_0_RX1_P
10GBE_0_RX2_N
10GBE_0_RX2_P
10GBE_0_RX3_N
10GBE_0_RX3_P
10GBE_1_RX0_N
10GBE_1_RX0_P
10GBE_1_RX1_N
10GBE_1_RX1_P
10GBE_1_RX2_N
10GBE_1_RX2_P
10GBE_1_RX3_N
10GBE_1_RX3_P
RX0_N
RX0_P
TX0_N
TX0_P
RX1_N
RX1_P
TX1_N
TX1_P
RX2_N
RX2_P
TX2_N
TX2_P
RX3_N
RX3_P
TX3_N
TX3_P
C306 100n
C308 100n
C312 100n
C313 100n
C315 100n
C319 100n
C321 100n
C325 100n
C554 100n
C555 100n
C556 100n
C557 100n
C558 100n
C559 100n
C560 100n
C561 100n
XC6SLX150T-4FGG676C
GND
C338 4u7
C339 4u7
C340 4u7
C341 4u7
C342 4u7
C343 4u7
C344 220n
C345 220n
C346 220n
C347 220n
C348 220n
C349 220n
C350 220n
C351 220n
C352 220n
C353 220n
C354 220n
C355 220n
C356 220n
C357 220n
C358 220n
C359 220n
C360 220n
C361 220n
C362 220n
C363 220n
Title
Spartan-6 Multi-Gigabit Transceivers
Size
A4
Number
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date: 29/03/2011
Sheet 14 of 65
File: C:\Users\..\spartan6_mgt.SchDoc
Drawn By: Simon Scott

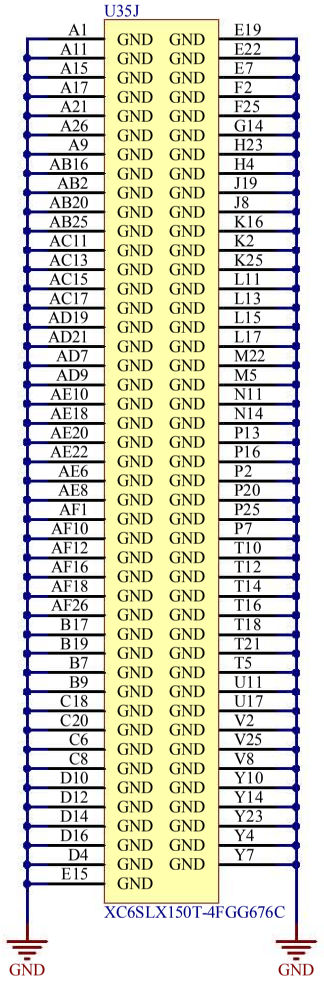
Layout Notes:
1.) Place "JTAG_HDR_EN" next to jumper on silkscreen
2.) Place "NOT_CONFIGD" next to LED on silkscreen

General Notes:
1.) Install jumper to enable the JTAG programming header
2.) For final board, JTAG 14-pin header and jumper should not be populated
3.) The Spartan-6 JTAG pins have internal pull-up resistors

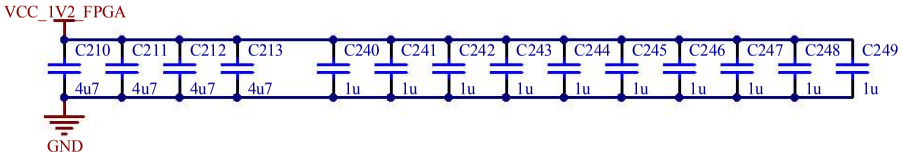


Title	**Spartan-6 Configuration**		
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 15 of 65	
File: C:\Users\..\spartan6_config.SchDoc		Drawn By: Simon Scott	

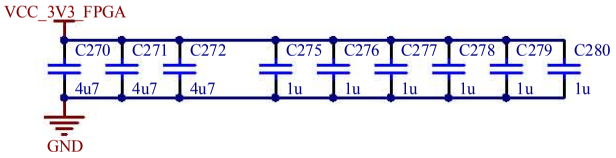




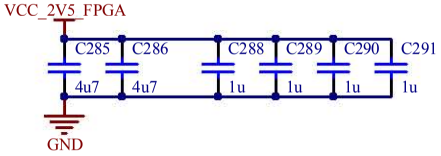
Title	**Spartan-6 Power**		
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 16 of 65	
File: C:\Users\..\spartan6_power.SchDoc		Drawn By: Simon Scott	



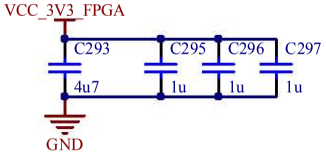
Decoupling caps for FPGA VCCINT



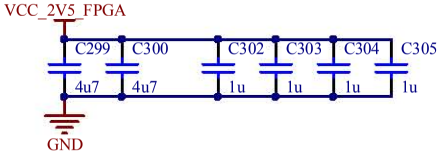
Decoupling caps for FPGA VCCAUX



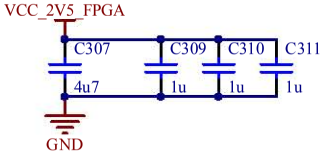
Decoupling caps for FPGA VCCO_0



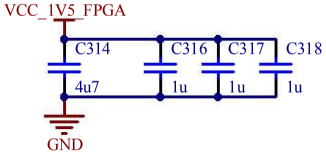
Decoupling caps for FPGA VCCO_1



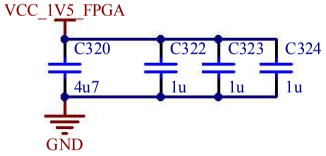
Decoupling caps for FPGA VCCO_2



Decoupling caps for FPGA VCCO_3



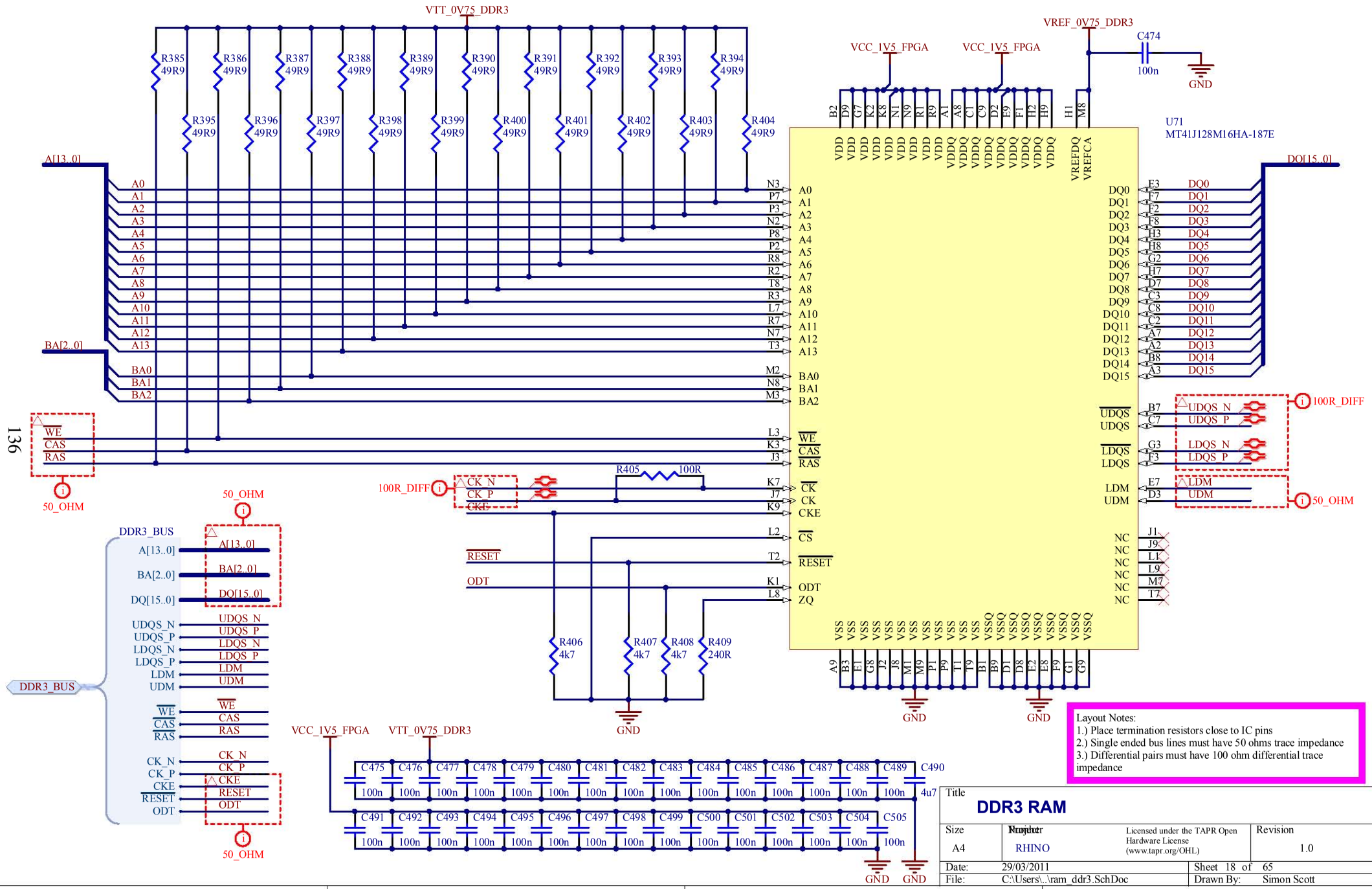
Decoupling caps for FPGA VCCO_4

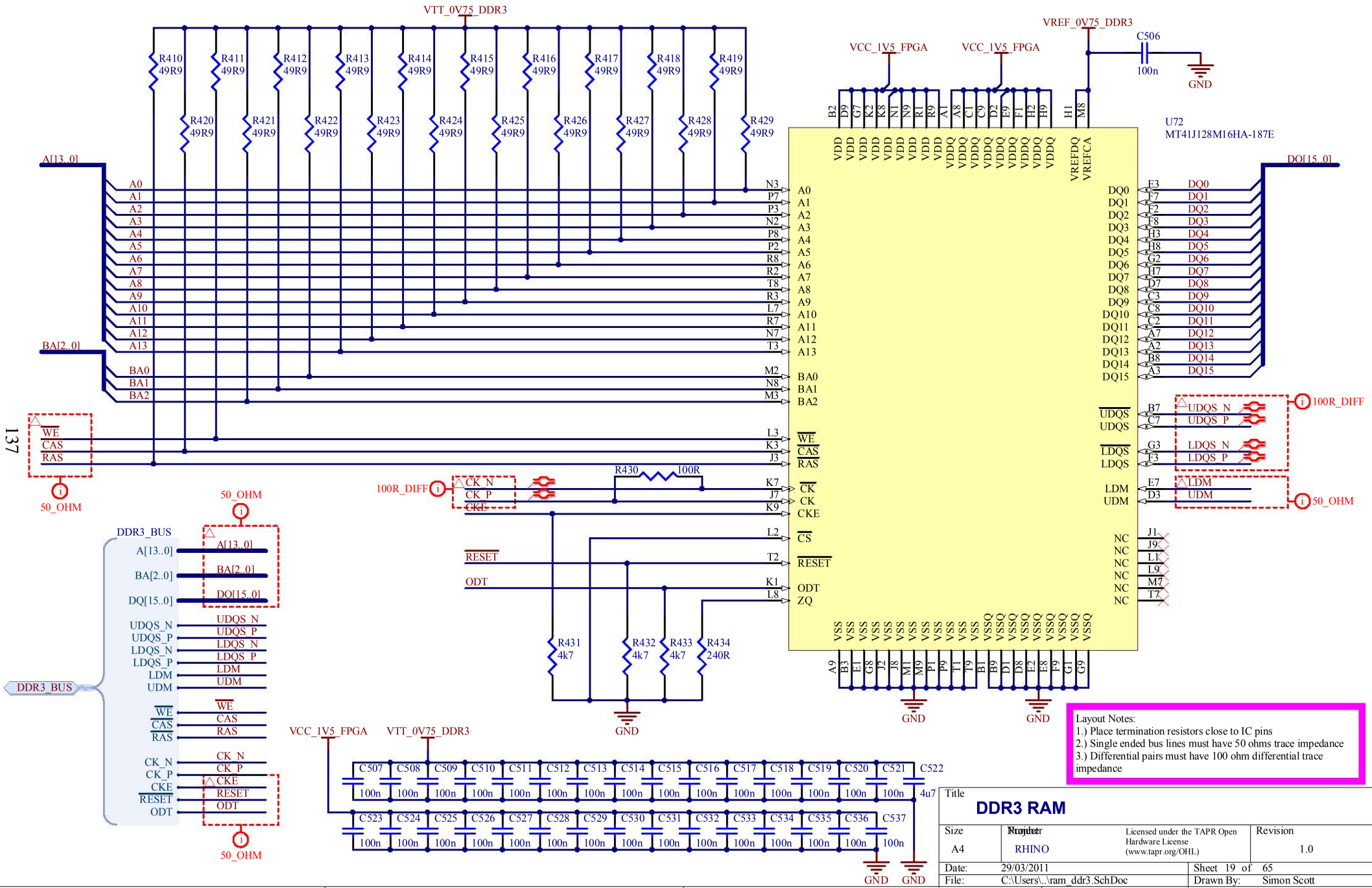


Decoupling caps for FPGA VCCO_5

Layout Notes:
1.) The 4u7 caps should be placed within 50mm of the outer edge of the FPGA
2.) The 1uF caps must be placed within 12mm of the outer edge of the FPGA. These caps must be placed as close to the FPGA as possible, preferably on the reverse sideo of the PCB, within the FPGA footprint.
3.) Capacitor mounting should be optimised for low inductance

Title: Spartan-6 Supply Decoupling Caps			
Size: A4	Project: RHINO		Revision: 1.0
Date: 29/03/2011		Sheet 17 of 65	
File: C:\Users\..\spartan6_decoupling.SchDoc		Drawn By: Simon Scott	

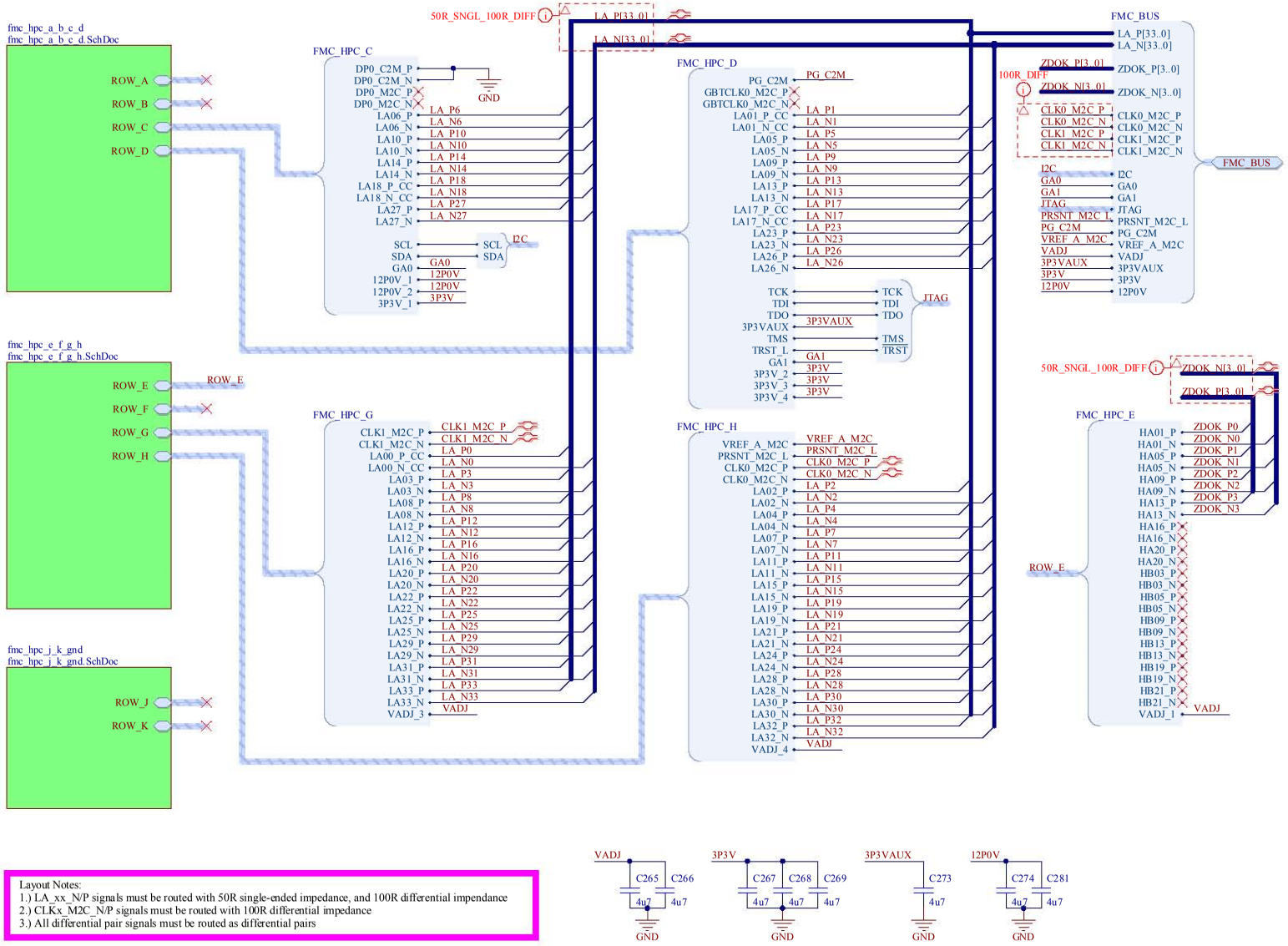

VTT_0V75_DDR3
VREF_0V75_DDR3
VCC_1V5_FPGA
VCC_1V5_FPGA
C474
100n
GND
U71
MT41J128M16HA-187E
R385 49R9
R386 49R9
R387 49R9
R388 49R9
R389 49R9
R390 49R9
R391 49R9
R392 49R9
R393 49R9
R394 49R9
R395 49R9
R396 49R9
R397 49R9
R398 49R9
R399 49R9
R400 49R9
R401 49R9
R402 49R9
R403 49R9
R404 49R9
A[13..0]
BA[2..0]
DQ[15..0]
WE
CAS
RAS
50_OHM
100R_DIFF
CK_N
CK_P
CKE
RESET
ODT
R405 100R
R406 4k7
R407 4k7
R408 4k7
R409 240R
UDQS_N
UDQS_P
LDQS_N
LDQS_P
LDM
UDM
DDR3_BUS
C475 C476 C477 C478 C479 C480 C481 C482 C483 C484 C485 C486 C487 C488 C489 C490 100n 4u7
C491 C492 C493 C494 C495 C496 C497 C498 C499 C500 C501 C502 C503 C504 C505 100n
Layout Notes:
1.) Place termination resistors close to IC pins
2.) Single ended bus lines must have 50 ohms trace impedance
3.) Differential pairs must have 100 ohm differential trace impedance
Title
DDR3 RAM
Size
A4
Number
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date: 29/03/2011
Sheet 18 of 65
File: C:\Users\..\ram_ddr3.SchDoc
Drawn By: Simon Scott



137

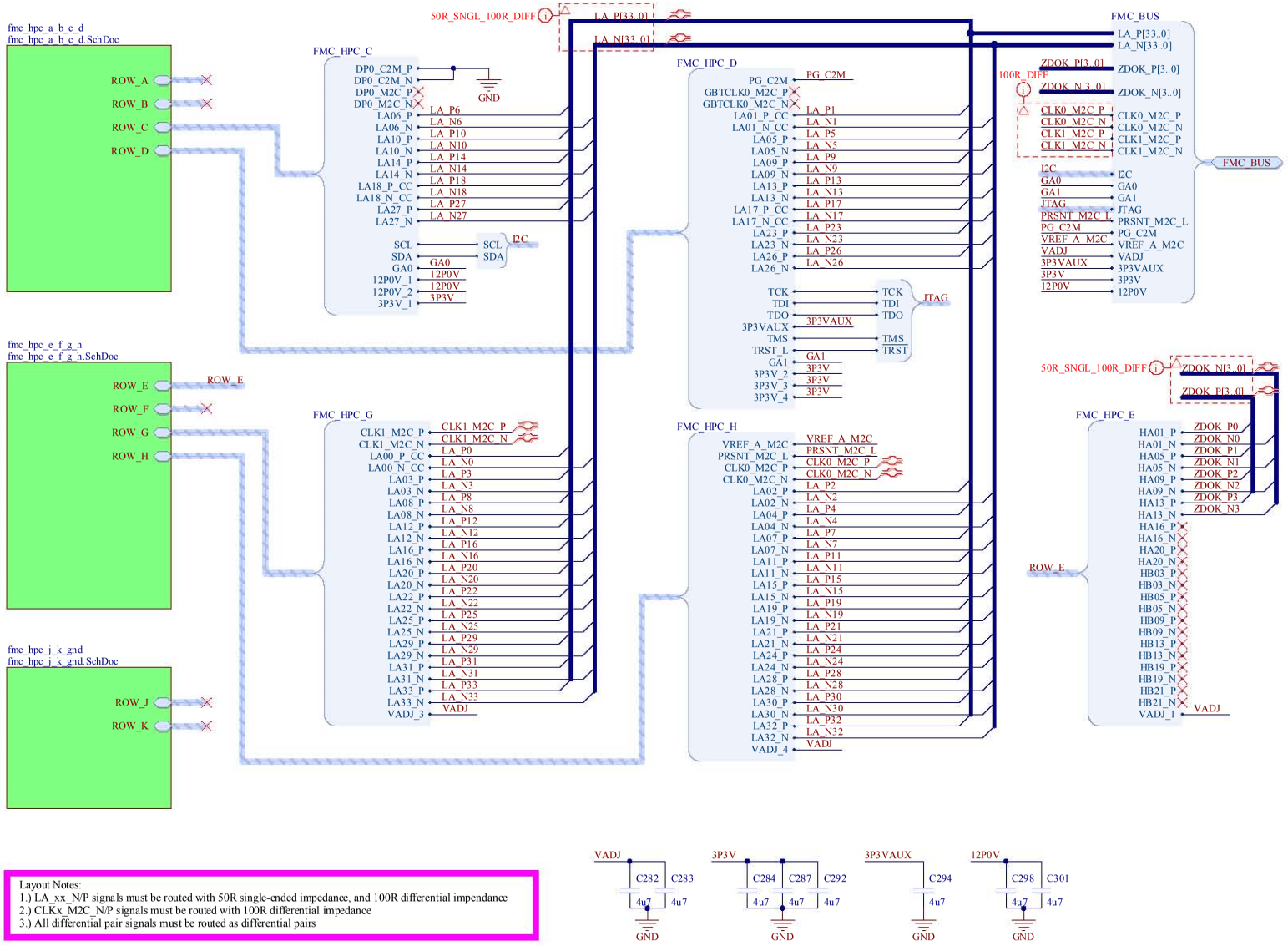
Layout Notes:
1.) Place termination resistors close to IC pins
2.) Single ended bus lines must have 50 ohms trace impedance
3.) Differential pairs must have 100 ohm differential trace impedance

Title	DDR3 RAM		
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011			Sheet 19 of 65
File: C:\Users\..\ram_ddr3.SchDoc			Drawn By: Simon Scott



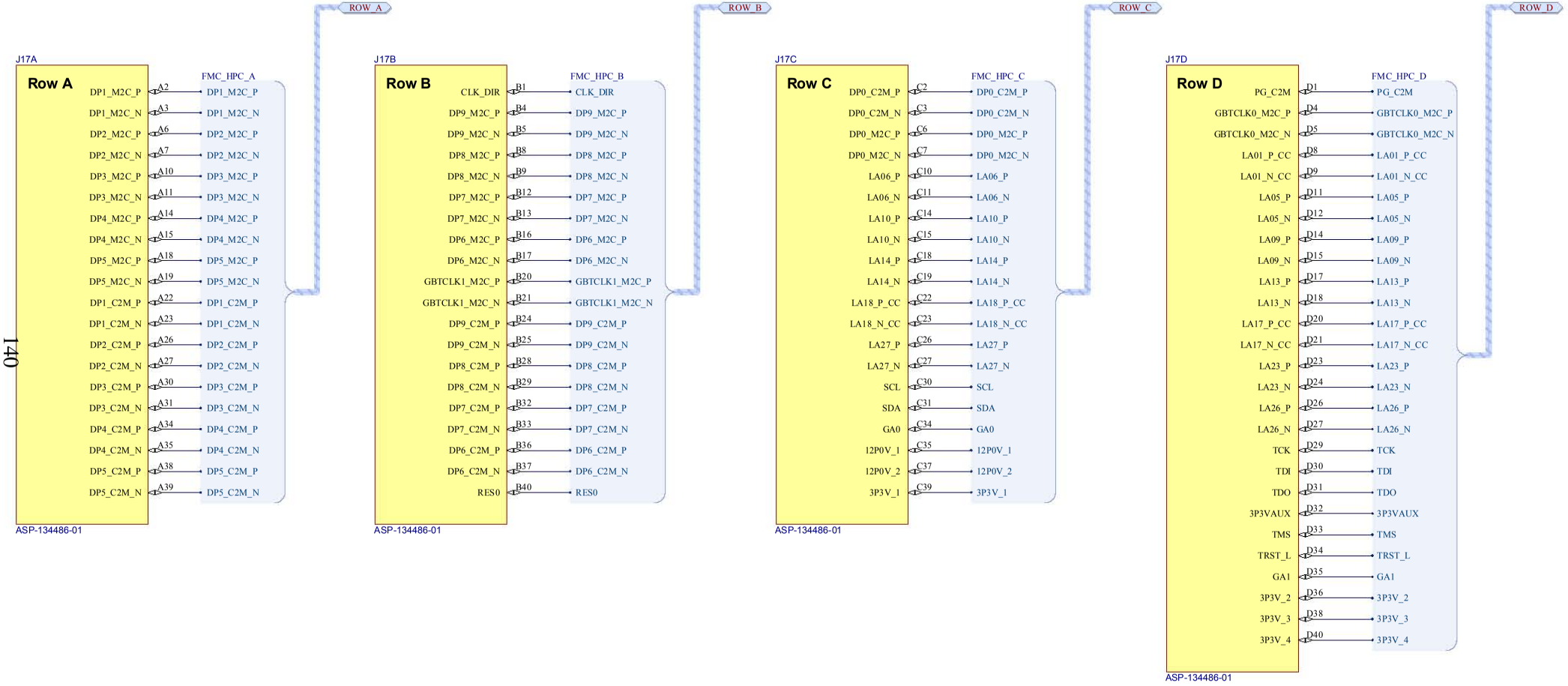
Layout Notes:
1.) LA_xx_N/P signals must be routed with 50R single-ended impedance, and 100R differential impendance
2.) CLKx_M2C_N/P signals must be routed with 100R differential impedance
3.) All differential pair signals must be routed as differential pairs

Title	FMC HPC Connector (top level)		
Size: A3	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 20 of 65	
File: C:\Users\...\fmc_hpc.SchDoc		Drawn By: Simon Scott	



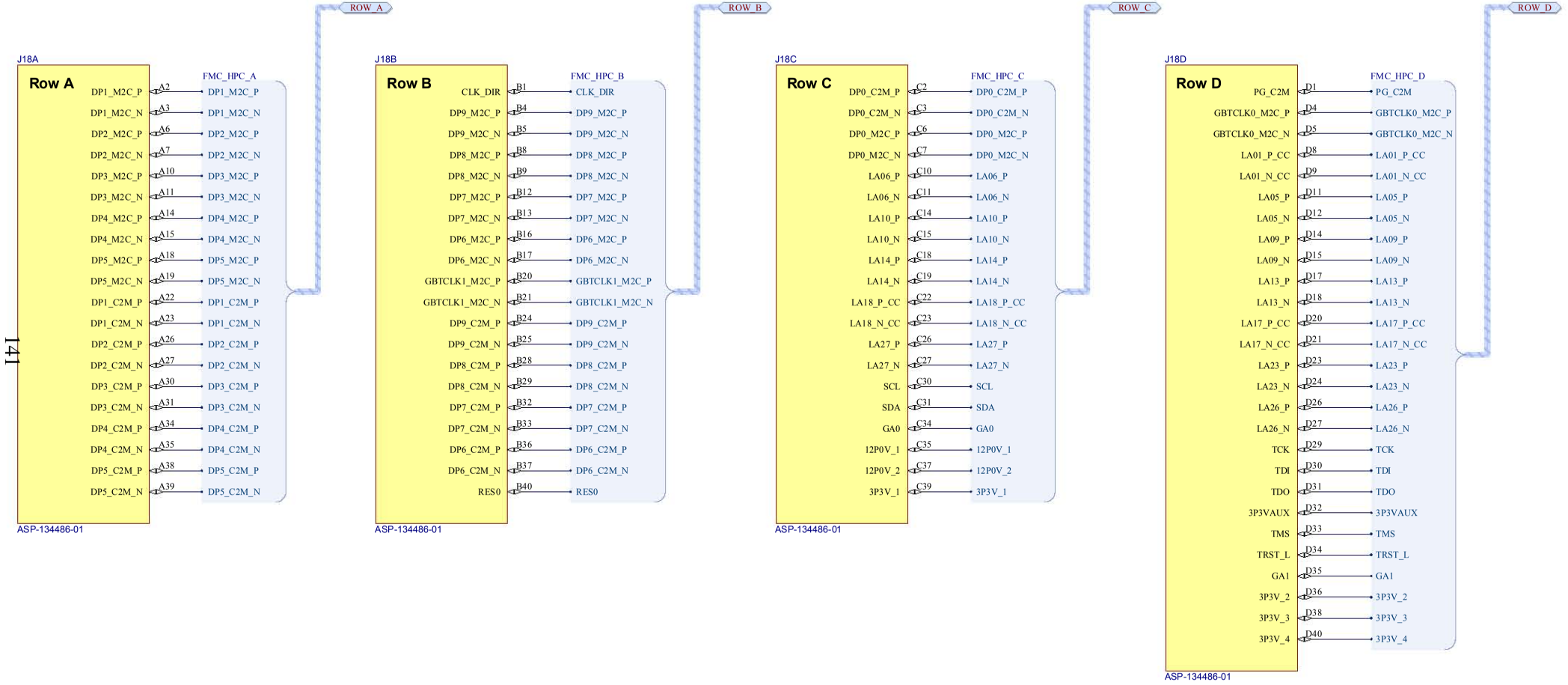
Layout Notes:
1.) LA_xx_N/P signals must be routed with 50R single-ended impedance, and 100R differential impendance
2.) CLKx_M2C_N/P signals must be routed with 100R differential impedance
3.) All differential pair signals must be routed as differential pairs

Title	FMC HPC Connector (top level)		
Size: A3	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 21 of 65	
File: C:\Users\...\fmc_hpc.SchDoc		Drawn By: Simon Scott	

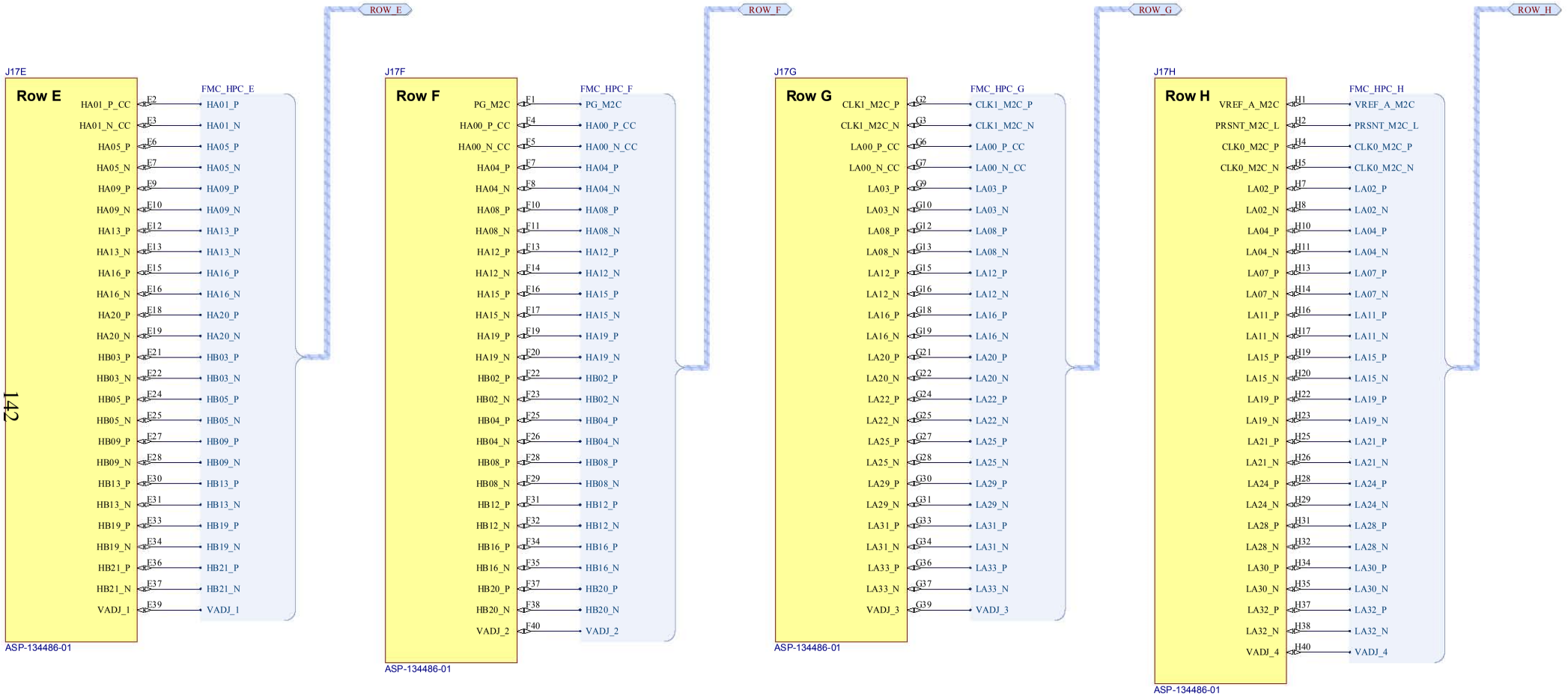


140

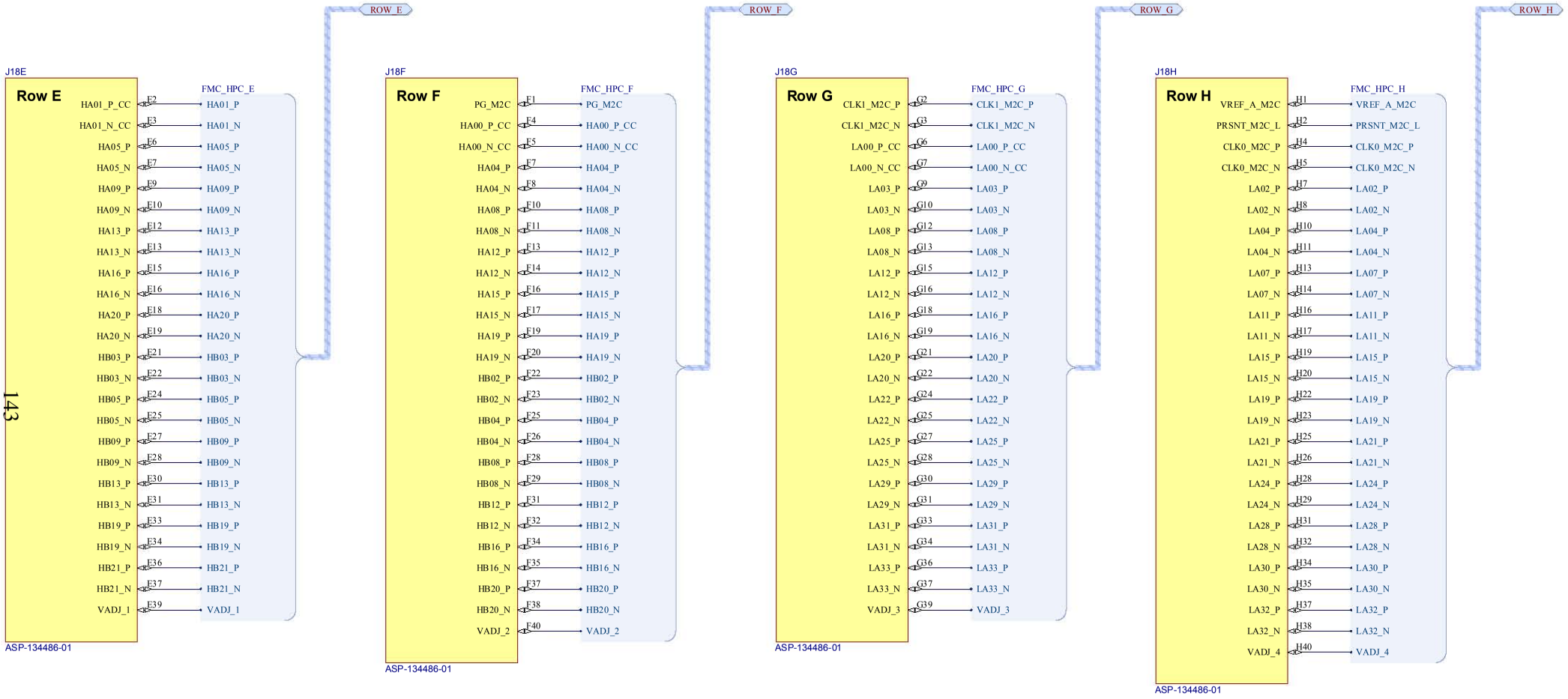
Title: FMC HPC Connector (Rows A, B, C, D)			
Size: A3	Project: RHINO		Revision: 1.0
Date: 29/03/2011		Sheet 22 of 65	
File: C:\Users\...\fmc_hpc_a_b_c_d.SchDoc		Drawn By: Simon Scott	



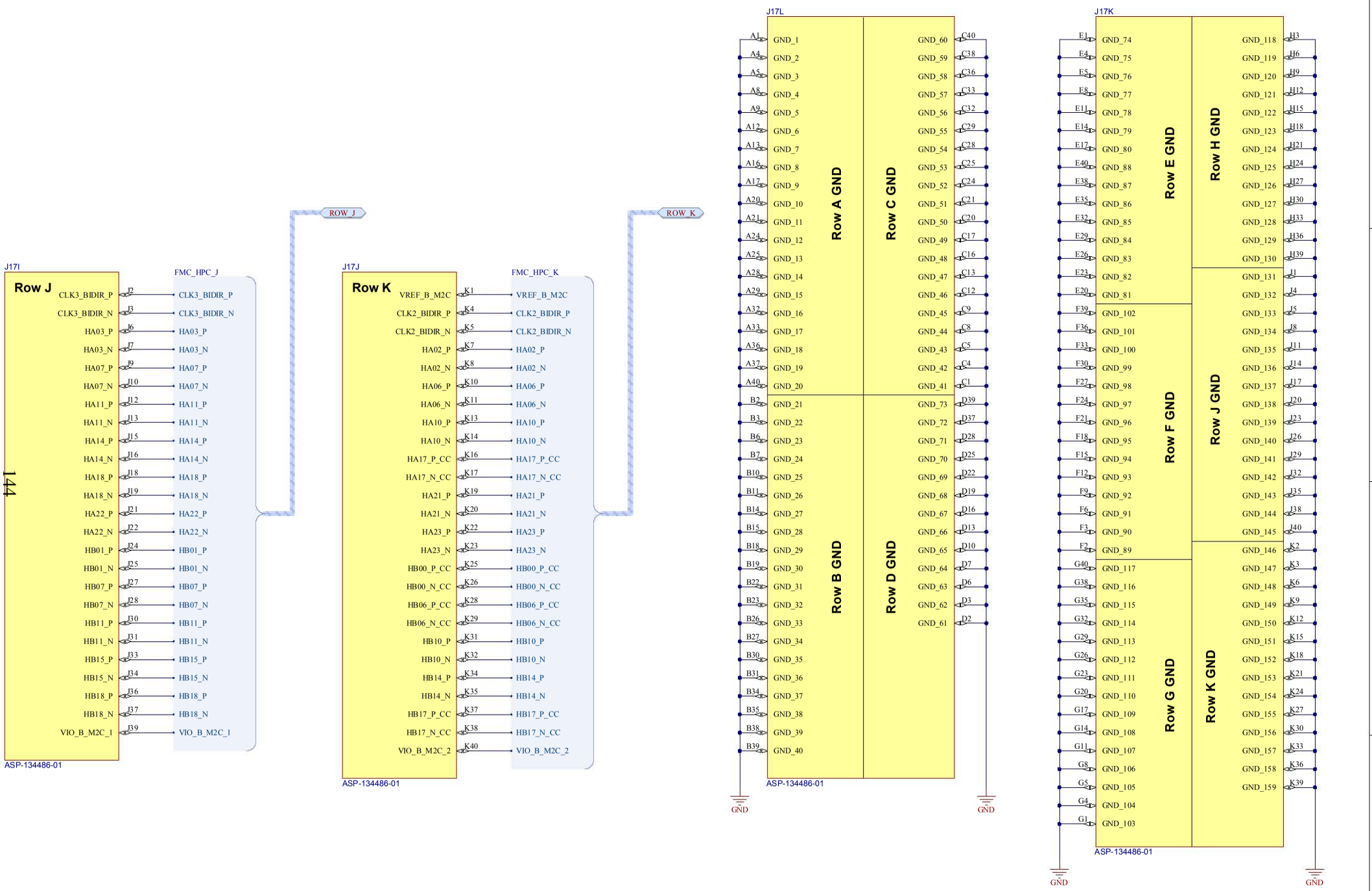
Title: FMC HPC Connector (Rows A, B, C, D)			
Size: A3	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 23 of 65	
File: C:\Users\...\fmc_hpc_a_b_c_d.SchDoc		Drawn By: Simon Scott	

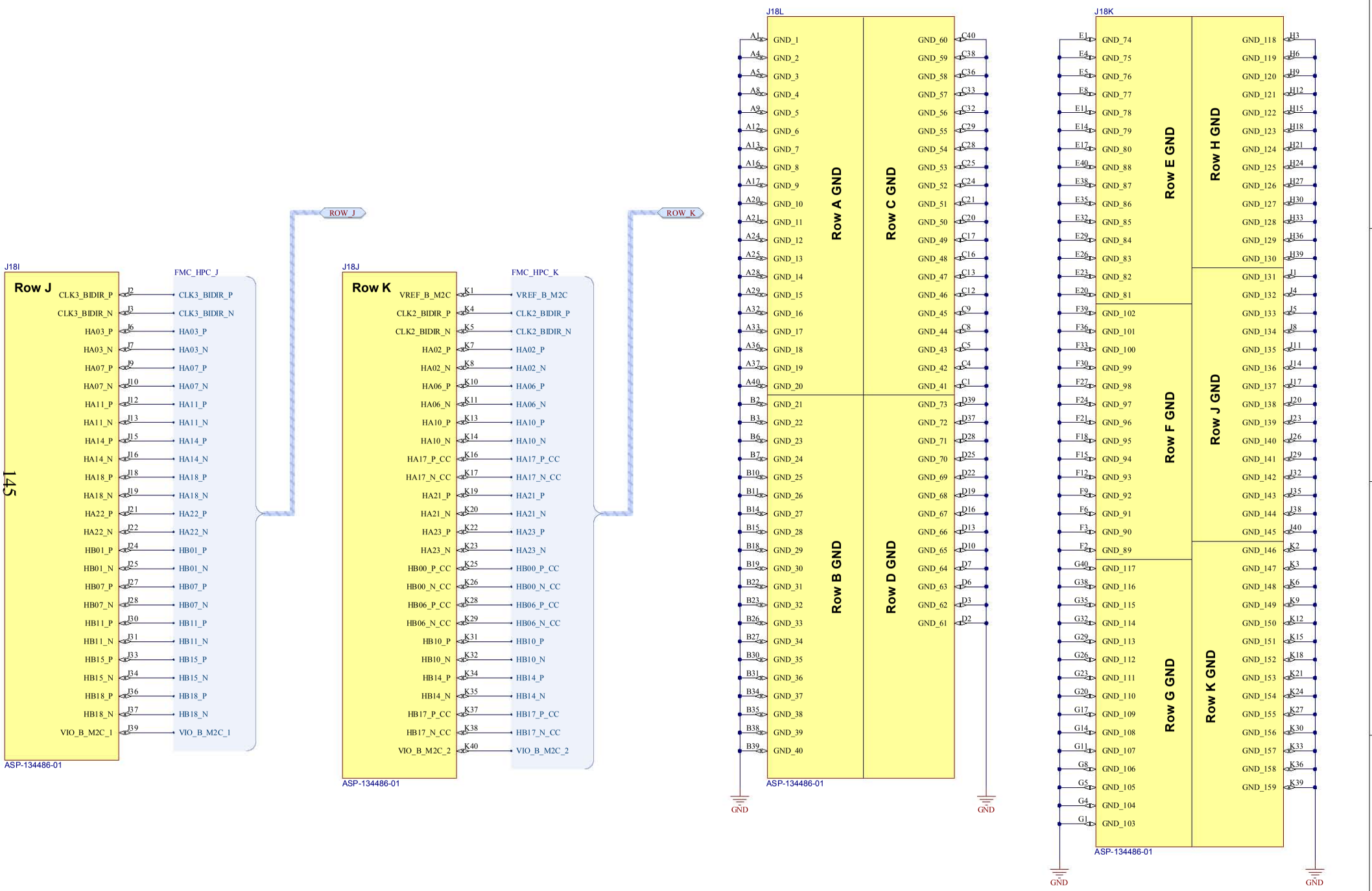


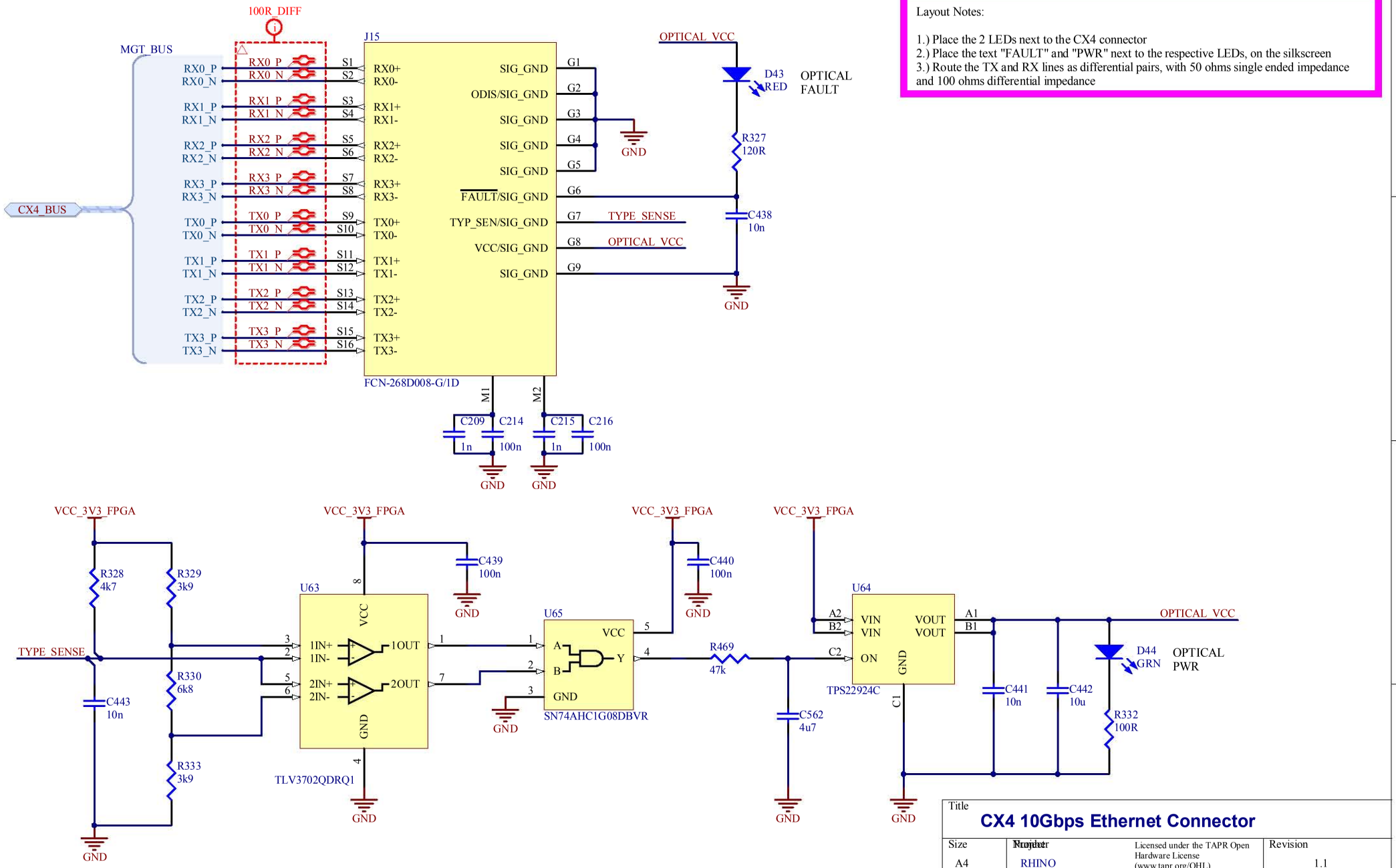
Title	FMC HPC Connector (Rows E, F, G, H)		
Size: A3	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 24 of 65	
File: C:\Users\...\fmc_hpc_e_f_g_h.SchDoc		Drawn By: Simon Scott	

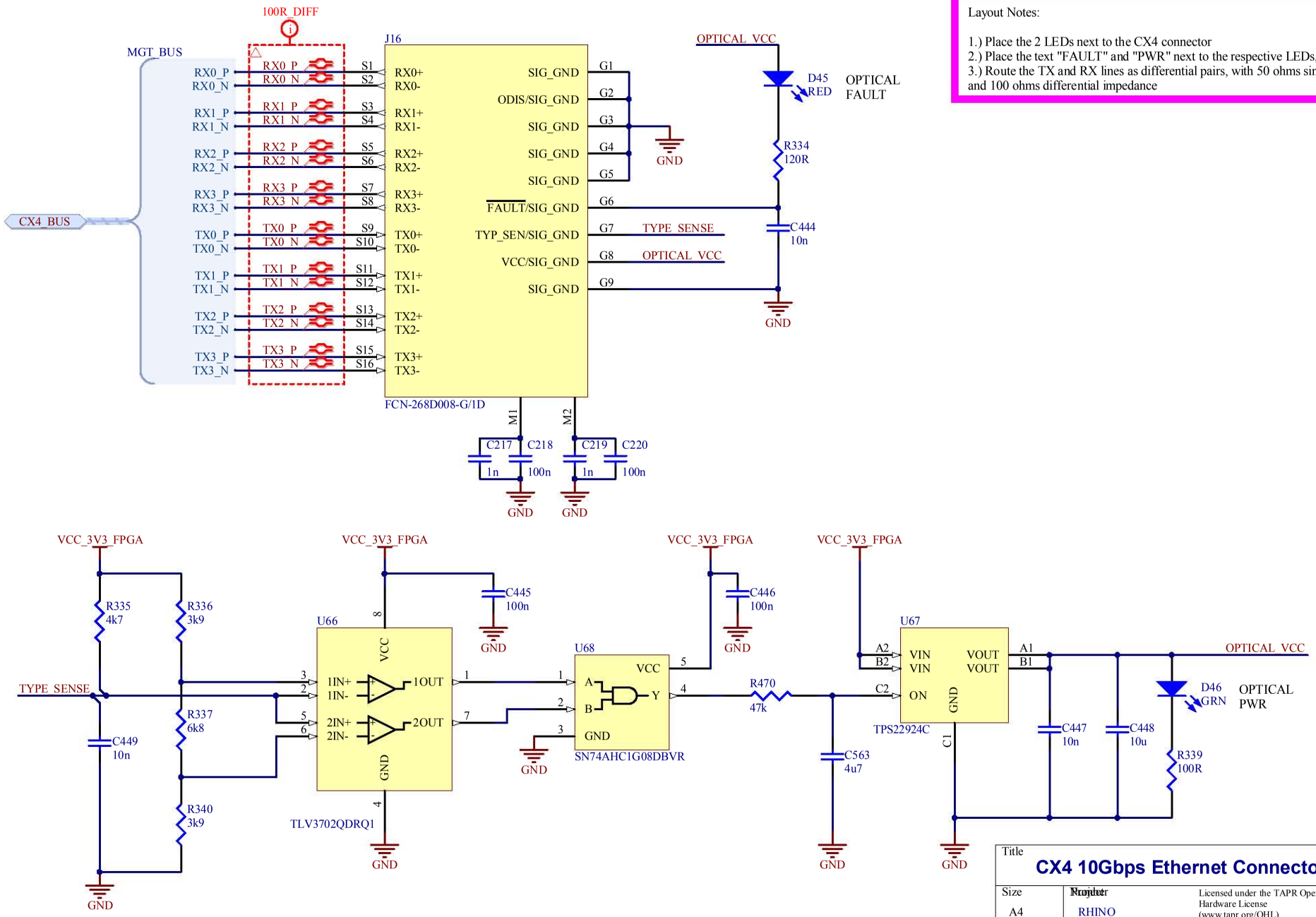


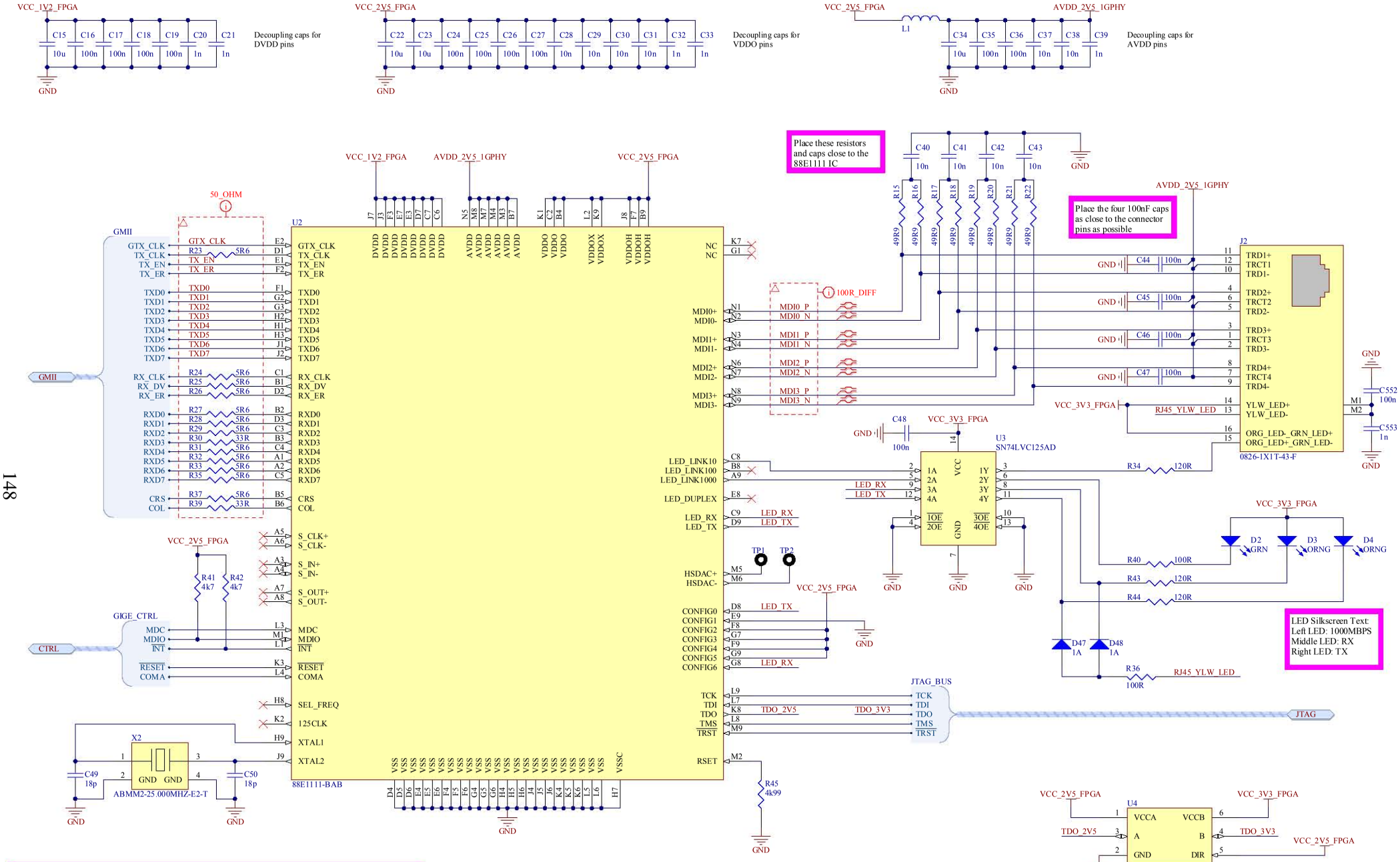
Title	FMC HPC Connector (Rows E, F, G, H)		
Size: A3	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 25 of 65	
File: C:\Users\...\fmc_hpc_e_f_g_h.SchDoc		Drawn By: Simon Scott	


J17I
Row J
CLK3_BIDIR_P J2
CLK3_BIDIR_N J3
HA03_P J6
HA03_N J7
HA07_P J9
HA07_N J10
HA11_P J12
HA11_N J13
HA14_P J15
HA14_N J16
HA18_P J18
HA18_N J19
HA22_P J21
HA22_N J22
HB01_P J24
HB01_N J25
HB07_P J27
HB07_N J28
HB11_P J30
HB11_N J31
HB15_P J33
HB15_N J34
HB18_P J36
HB18_N J37
VIO_B_M2C_1 J39
ASP-134486-01
FMC_HPC_J
CLK3_BIDIR_P
CLK3_BIDIR_N
HA03_P
HA03_N
HA07_P
HA07_N
HA11_P
HA11_N
HA14_P
HA14_N
HA18_P
HA18_N
HA22_P
HA22_N
HB01_P
HB01_N
HB07_P
HB07_N
HB11_P
HB11_N
HB15_P
HB15_N
HB18_P
HB18_N
VIO_B_M2C_1
ROW_J
144
J17J
Row K
VREF_B_M2C K1
CLK2_BIDIR_P K4
CLK2_BIDIR_N K5
HA02_P K7
HA02_N K8
HA06_P K10
HA06_N K11
HA10_P K13
HA10_N K14
HA17_P_CC K16
HA17_N_CC K17
HA21_P K19
HA21_N K20
HA23_P K22
HA23_N K23
HB00_P_CC K25
HB00_N_CC K26
HB06_P_CC K28
HB06_N_CC K29
HB10_P K31
HB10_N K32
HB14_P K34
HB14_N K35
HB17_P_CC K37
HB17_N_CC K38
VIO_B_M2C_2 K40
ASP-134486-01
FMC_HPC_K
VREF_B_M2C
CLK2_BIDIR_P
CLK2_BIDIR_N
HA02_P
HA02_N
HA06_P
HA06_N
HA10_P
HA10_N
HA17_P_CC
HA17_N_CC
HA21_P
HA21_N
HA23_P
HA23_N
HB00_P_CC
HB00_N_CC
HB06_P_CC
HB06_N_CC
HB10_P
HB10_N
HB14_P
HB14_N
HB17_P_CC
HB17_N_CC
VIO_B_M2C_2
ROW_K
J17L
Row A GND
GND_1 A1
GND_2 A4
GND_3 A5
GND_4 A8
GND_5 A9
GND_6 A12
GND_7 A13
GND_8 A16
GND_9 A17
GND_10 A20
GND_11 A21
GND_12 A24
GND_13 A25
GND_14 A28
GND_15 A29
GND_16 A32
GND_17 A33
GND_18 A36
GND_19 A37
GND_20 A40
Row B GND
GND_21 B2
GND_22 B3
GND_23 B6
GND_24 B7
GND_25 B10
GND_26 B11
GND_27 B14
GND_28 B15
GND_29 B18
GND_30 B19
GND_31 B22
GND_32 B23
GND_33 B26
GND_34 B27
GND_35 B30
GND_36 B31
GND_37 B34
GND_38 B35
GND_39 B38
GND_40 B39
Row C GND
GND_60 C40
GND_59 C38
GND_58 C36
GND_57 C33
GND_56 C32
GND_55 C29
GND_54 C28
GND_53 C25
GND_52 C24
GND_51 C21
GND_50 C20
GND_49 C17
GND_48 C16
GND_47 C13
GND_46 C12
GND_45 C9
GND_44 C8
GND_43 C5
GND_42 C4
GND_41 C1
Row D GND
GND_73 D39
GND_72 D37
GND_71 D28
GND_70 D25
GND_69 D22
GND_68 D19
GND_67 D16
GND_66 D13
GND_65 D10
GND_64 D7
GND_63 D6
GND_62 D3
GND_61 D2
ASP-134486-01
GND
GND
J17K
Row E GND
GND_74 E1
GND_75 E4
GND_76 E5
GND_77 E8
GND_78 E11
GND_79 E14
GND_80 E17
GND_88 E40
GND_87 E38
GND_86 E35
GND_85 E32
GND_84 E29
GND_83 E26
GND_82 E23
GND_81 E20
Row F GND
GND_102 F39
GND_101 F36
GND_100 F33
GND_99 F30
GND_98 F27
GND_97 F24
GND_96 F21
GND_95 F18
GND_94 F15
GND_93 F12
GND_92 F9
GND_91 F6
GND_90 F3
GND_89 F2
Row G GND
GND_117 G40
GND_116 G38
GND_115 G35
GND_114 G32
GND_113 G29
GND_112 G26
GND_111 G23
GND_110 G20
GND_109 G17
GND_108 G14
GND_107 G11
GND_106 G8
GND_105 G5
GND_104 G4
GND_103 G1
Row H GND
GND_118 H3
GND_119 H6
GND_120 H9
GND_121 H12
GND_122 H15
GND_123 H18
GND_124 H21
GND_125 H24
GND_126 H27
GND_127 H30
GND_128 H33
GND_129 H36
GND_130 H39
Row J GND
GND_131 J1
GND_132 J4
GND_133 J5
GND_134 J8
GND_135 J11
GND_136 J14
GND_137 J17
GND_138 J20
GND_139 J23
GND_140 J26
GND_141 J29
GND_142 J32
GND_143 J35
GND_144 J38
GND_145 J40
Row K GND
GND_146 K2
GND_147 K3
GND_148 K6
GND_149 K9
GND_150 K12
GND_151 K15
GND_152 K18
GND_153 K21
GND_154 K24
GND_155 K27
GND_156 K30
GND_157 K33
GND_158 K36
GND_159 K39
ASP-134486-01
GND
GND
Title
FMC HPC Connector (Rows J, K and Ground)
Size
A3
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date: 29/03/2011
Sheet 26 of 65
File: C:\Users\...\fmc_hpc_j_k_gnd.SchDoc
Drawn By: Simon Scott

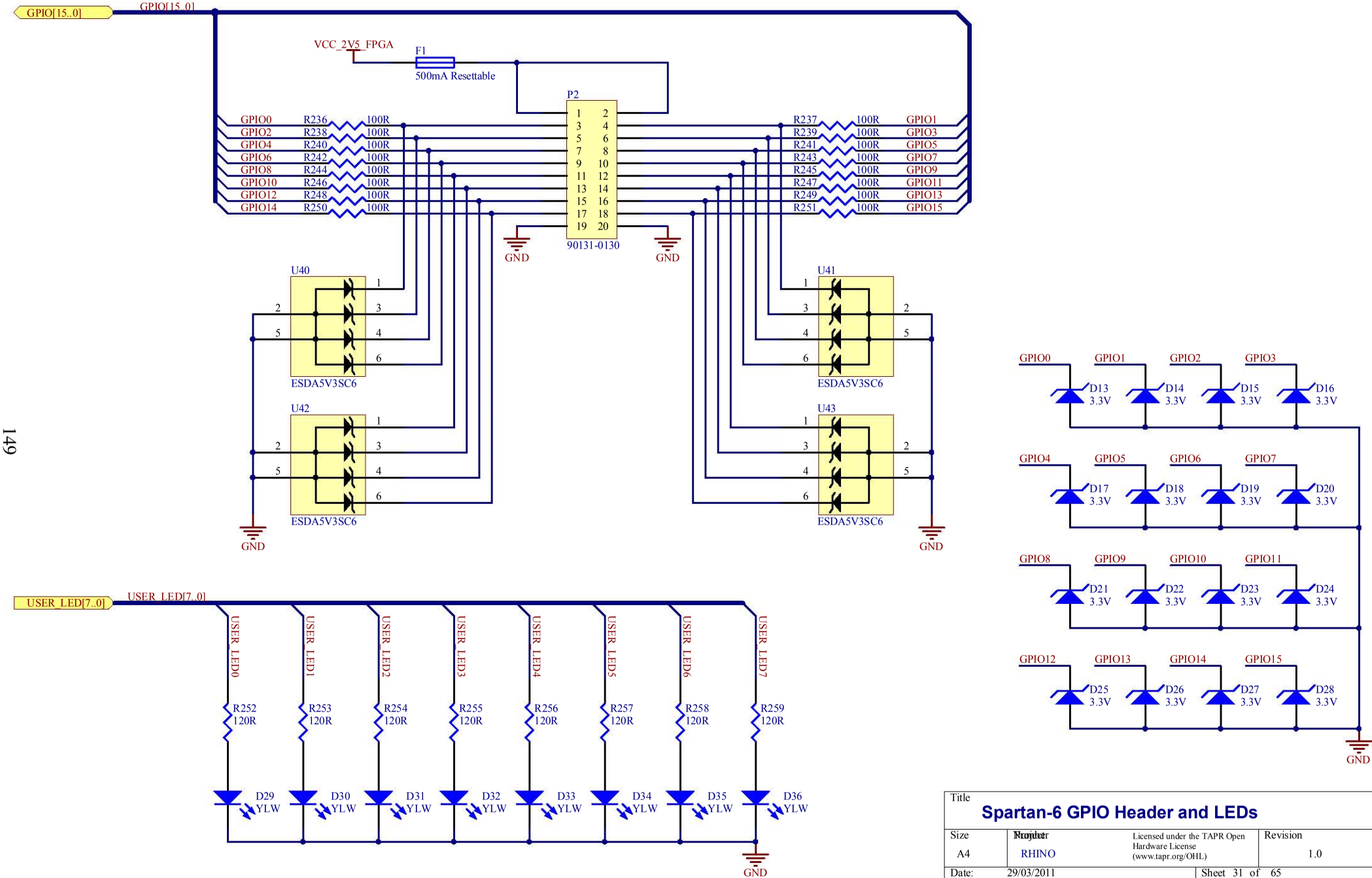

J18I
Row J
CLK3_BIDIR_P J2
CLK3_BIDIR_N J3
HA03_P J6
HA03_N J7
HA07_P J9
HA07_N J10
HA11_P J12
HA11_N J13
HA14_P J15
HA14_N J16
HA18_P J18
HA18_N J19
HA22_P J21
HA22_N J22
HB01_P J24
HB01_N J25
HB07_P J27
HB07_N J28
HB11_P J30
HB11_N J31
HB15_P J33
HB15_N J34
HB18_P J36
HB18_N J37
VIO_B_M2C_1 J39
ASP-134486-01
FMC_HPC_J
CLK3_BIDIR_P
CLK3_BIDIR_N
HA03_P
HA03_N
HA07_P
HA07_N
HA11_P
HA11_N
HA14_P
HA14_N
HA18_P
HA18_N
HA22_P
HA22_N
HB01_P
HB01_N
HB07_P
HB07_N
HB11_P
HB11_N
HB15_P
HB15_N
HB18_P
HB18_N
VIO_B_M2C_1
ROW_J
J18J
Row K
VREF_B_M2C K1
CLK2_BIDIR_P K4
CLK2_BIDIR_N K5
HA02_P K7
HA02_N K8
HA06_P K10
HA06_N K11
HA10_P K13
HA10_N K14
HA17_P_CC K16
HA17_N_CC K17
HA21_P K19
HA21_N K20
HA23_P K22
HA23_N K23
HB00_P_CC K25
HB00_N_CC K26
HB06_P_CC K28
HB06_N_CC K29
HB10_P K31
HB10_N K32
HB14_P K34
HB14_N K35
HB17_P_CC K37
HB17_N_CC K38
VIO_B_M2C_2 K40
ASP-134486-01
FMC_HPC_K
VREF_B_M2C
CLK2_BIDIR_P
CLK2_BIDIR_N
HA02_P
HA02_N
HA06_P
HA06_N
HA10_P
HA10_N
HA17_P_CC
HA17_N_CC
HA21_P
HA21_N
HA23_P
HA23_N
HB00_P_CC
HB00_N_CC
HB06_P_CC
HB06_N_CC
HB10_P
HB10_N
HB14_P
HB14_N
HB17_P_CC
HB17_N_CC
VIO_B_M2C_2
ROW_K
J18L
Row A GND
GND_1 A1
GND_2 A4
GND_3 A5
GND_4 A8
GND_5 A9
GND_6 A12
GND_7 A13
GND_8 A16
GND_9 A17
GND_10 A20
GND_11 A21
GND_12 A24
GND_13 A25
GND_14 A28
GND_15 A29
GND_16 A32
GND_17 A33
GND_18 A36
GND_19 A37
GND_20 A40
Row B GND
GND_21 B2
GND_22 B3
GND_23 B6
GND_24 B7
GND_25 B10
GND_26 B11
GND_27 B14
GND_28 B15
GND_29 B18
GND_30 B19
GND_31 B22
GND_32 B23
GND_33 B26
GND_34 B27
GND_35 B30
GND_36 B31
GND_37 B34
GND_38 B35
GND_39 B38
GND_40 B39
Row C GND
GND_60 C40
GND_59 C38
GND_58 C36
GND_57 C33
GND_56 C32
GND_55 C29
GND_54 C28
GND_53 C25
GND_52 C24
GND_51 C21
GND_50 C20
GND_49 C17
GND_48 C16
GND_47 C13
GND_46 C12
GND_45 C9
GND_44 C8
GND_43 C5
GND_42 C4
GND_41 C1
Row D GND
GND_73 D39
GND_72 D37
GND_71 D28
GND_70 D25
GND_69 D22
GND_68 D19
GND_67 D16
GND_66 D13
GND_65 D10
GND_64 D7
GND_63 D6
GND_62 D3
GND_61 D2
ASP-134486-01
GND
GND
J18K
Row E GND
GND_74 E1
GND_75 E4
GND_76 E5
GND_77 E8
GND_78 E11
GND_79 E14
GND_80 E17
GND_88 E40
GND_87 E38
GND_86 E35
GND_85 E32
GND_84 E29
GND_83 E26
GND_82 E23
GND_81 E20
Row F GND
GND_102 F39
GND_101 F36
GND_100 F33
GND_99 F30
GND_98 F27
GND_97 F24
GND_96 F21
GND_95 F18
GND_94 F15
GND_93 F12
GND_92 F9
GND_91 F6
GND_90 F3
GND_89 F2
Row G GND
GND_117 G40
GND_116 G38
GND_115 G35
GND_114 G32
GND_113 G29
GND_112 G26
GND_111 G23
GND_110 G20
GND_109 G17
GND_108 G14
GND_107 G11
GND_106 G8
GND_105 G5
GND_104 G4
GND_103 G1
Row H GND
GND_118 H3
GND_119 H6
GND_120 H9
GND_121 H12
GND_122 H15
GND_123 H18
GND_124 H21
GND_125 H24
GND_126 H27
GND_127 H30
GND_128 H33
GND_129 H36
GND_130 H39
Row J GND
GND_131 J1
GND_132 J4
GND_133 J5
GND_134 J8
GND_135 J11
GND_136 J14
GND_137 J17
GND_138 J20
GND_139 J23
GND_140 J26
GND_141 J29
GND_142 J32
GND_143 J35
GND_144 J38
GND_145 J40
Row K GND
GND_146 K2
GND_147 K3
GND_148 K6
GND_149 K9
GND_150 K12
GND_151 K15
GND_152 K18
GND_153 K21
GND_154 K24
GND_155 K27
GND_156 K30
GND_157 K33
GND_158 K36
GND_159 K39
ASP-134486-01
GND
GND
145
Title
FMC HPC Connector (Rows J, K and Ground)
Size
A3
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date: 29/03/2011
Sheet 27 of 65
File: C:\Users\...\fmc_hpc_j_k_gnd.SchDoc
Drawn By: Simon Scott


Layout Notes:
1.) Place the 2 LEDs next to the CX4 connector
2.) Place the text "FAULT" and "PWR" next to the respective LEDs, on the silkscreen
3.) Route the TX and RX lines as differential pairs, with 50 ohms single ended impedance and 100 ohms differential impedance
100R_DIFF
MGT_BUS
CX4_BUS
J15
FCN-268D008-G/1D
OPTICAL_VCC
D43
RED
OPTICAL FAULT
R327
120R
C438
10n
GND
TYPE_SENSE
C209
1n
C214
100n
C215
1n
C216
100n
M1
M2
VCC_3V3_FPGA
R328
4k7
R329
3k9
R330
6k8
R333
3k9
C443
10n
U63
TLV3702QDRQ1
C439
100n
U65
SN74AHC1G08DBVR
C440
100n
R469
47k
C562
4u7
U64
TPS22924C
C441
10n
C442
10u
D44
GRN
OPTICAL PWR
R332
100R
146
Title
CX4 10Gbps Ethernet Connector
Size
A4
Number
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.1
Date:
29/03/2011
Sheet 28 of 65
File:
C:\Users\..\cx4.SchDoc
Drawn By:
Simon Scott

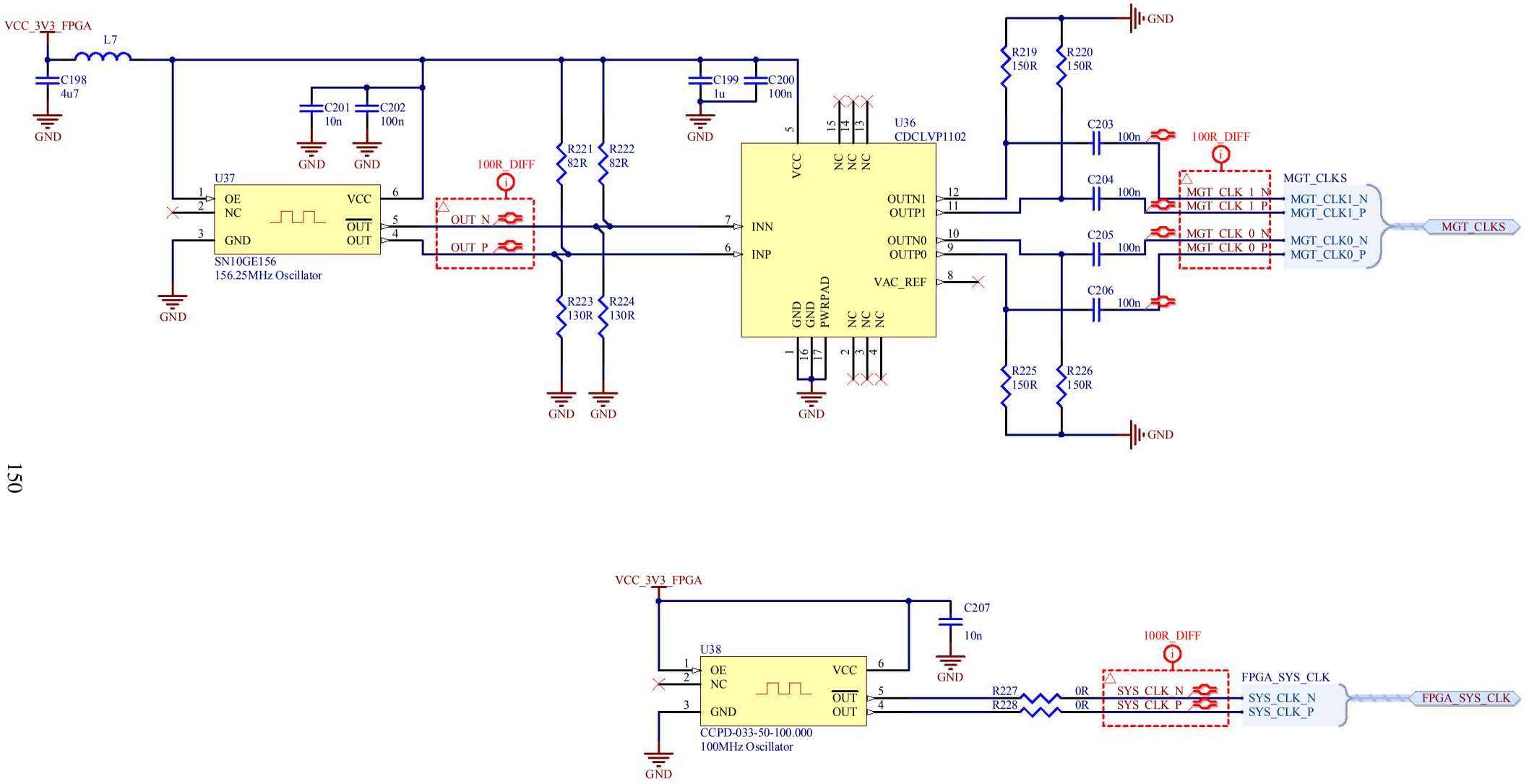

Layout Notes:
1.) Place the 2 LEDs next to the CX4 connector
2.) Place the text "FAULT" and "PWR" next to the respective LEDs, on the silkscreen
3.) Route the TX and RX lines as differential pairs, with 50 ohms single ended impedance and 100 ohms differential impedance
100R_DIFF
MGT_BUS
CX4_BUS
J16
FCN-268D008-G/1D
OPTICAL_VCC
D45
RED
OPTICAL FAULT
R334
120R
C444
10n
GND
TYPE_SENSE
VCC_3V3_FPGA
R335
4k7
R336
3k9
R337
6k8
R340
3k9
C449
10n
U66
TLV3702QDRQ1
C445
100n
C446
100n
U68
SN74AHC1G08DBVR
R470
47k
C563
4u7
U67
TPS22924C
C447
10n
C448
10u
D46
GRN
OPTICAL PWR
R339
100R
C217
1n
C218
100n
C219
1n
C220
100n
M1
M2
147
Title
CX4 10Gbps Ethernet Connector
Size
A4
Number
RHINO
Revision
1.1
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Date: 29/03/2011
Sheet 29 of 65
File: C:\Users\..\cx4.SchDoc
Drawn By: Simon Scott


VCC_1V2_FPGA
C15 C16 C17 C18 C19 C20 C21
10u 100n 100n 100n 100n 1n 1n
GND
Decoupling caps for DVDD pins
VCC_2V5_FPGA
C22 C23 C24 C25 C26 C27 C28 C29 C30 C31 C32 C33
10u 10u 100n 100n 100n 100n 10n 10n 10n 10n 1n 1n
GND
Decoupling caps for VDDO pins
VCC_2V5_FPGA
L1
AVDD_2V5_1GPHY
C34 C35 C36 C37 C38 C39
10u 100n 100n 10n 10n 1n
GND
Decoupling caps for AVDD pins
Place these resistors and caps close to the 88E1111 IC
Place the four 100nF caps as close to the connector pins as possible
50_OHM
100R_DIFF
U2
88E1111-BAB
GMII
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
MDI0+
MDI0-
MDI1+
MDI1-
MDI2+
MDI2-
MDI3+
MDI3-
LED_LINK10
LED_LINK100
LED_LINK1000
LED_DUPLEX
LED_RX
LED_TX
HSDAC+
HSDAC-
CONFIG0
CONFIG1
CONFIG2
CONFIG3
CONFIG4
CONFIG5
CONFIG6
S_CLK+
S_CLK-
S_IN+
S_IN-
S_OUT+
S_OUT-
MDC
MDIO
INT
RESET
COMA
SEL_FREQ
125CLK
XTAL1
XTAL2
RSET
TCK
TDI
TDO
TMS
TRST
R15 R16 R17 R18 R19 R20 R21 R22
49R9
C40 C41 C42 C43
10n
J2
0826-1X1T-43-F
TRD1+
TRCT1
TRD1-
TRD2+
TRCT2
TRD2-
TRD3+
TRCT3
TRD3-
TRD4+
TRCT4
TRD4-
YLW_LED+
YLW_LED-
ORG_LED-_GRN_LED+
ORG_LED+_GRN_LED-
C44 C45 C46 C47
100n
C552 100n
C553 1n
RJ45_YLW_LED
U3
SN74LVC125AD
R34 120R
R40 100R
R43 120R
R44 120R
D2 GRN
D3 ORNG
D4 ORNG
D47 1A
D48 1A
R36 100R
LED Silkscreen Text:
Left LED: 1000MBPS
Middle LED: RX
Right LED: TX
JTAG_BUS
JTAG
TDO_2V5
TDO_3V3
U4
SN74LVC1T45DBVR
R41 4k7
R42 4k7
R45 4k99
X2
ABMM2-25.000MHZ-E2-T
C49 18p
C50 18p
TP1
TP2
Layout Notes:
1.) HSDAC+ and HSDAC- pins must be left unconnected, but should be attached to test points
2.) The traces for the GMII bus must have 50 ohms impedance
3.) The 22 ohm termination resistors must be placed as close to the 88E1111 IC as possible
4.) The MDIx+/- traces must be routed as differential pairs, with 100 ohms diff impedance
5.) Place the 3 LEDs next to the RJ45 connector, with the silkscreen labels alongside
6.) Bypass capacitors should be distributed amongst the IC supply pins, and placed as close to the pins as possible. Preference should be given to the placement of low value caps
General Notes:
1.) PHY address is 0x01
2.) Config settings: pause disabled, full auto negotiation (slave preferred), crossover enabled, 125MHz clock output disabled, set to GMII copper mode, sleep disabled, MDIO interface enabled
3.) Although the JTAG pins are powered off VDDOX (2.5V), they are 3.3V tolerant
4.) LEDs on RJ45 connector: Green LED indicates link up, Yellow LED indicates traffic
Title
1 Gbps Ethernet PHY
Size
A3
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.1
Date:
29/03/2011
Sheet 30 of 65
File:
C:\Users\..\1gbps_ethernet.SchDoc
Drawn By:
Simon Scott

148



Title	**Spartan-6 GPIO Header and LEDs**		
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 31 of 65	
File: C:\Users\..\spartan6_gpio.SchDoc		Drawn By: Simon Scott	

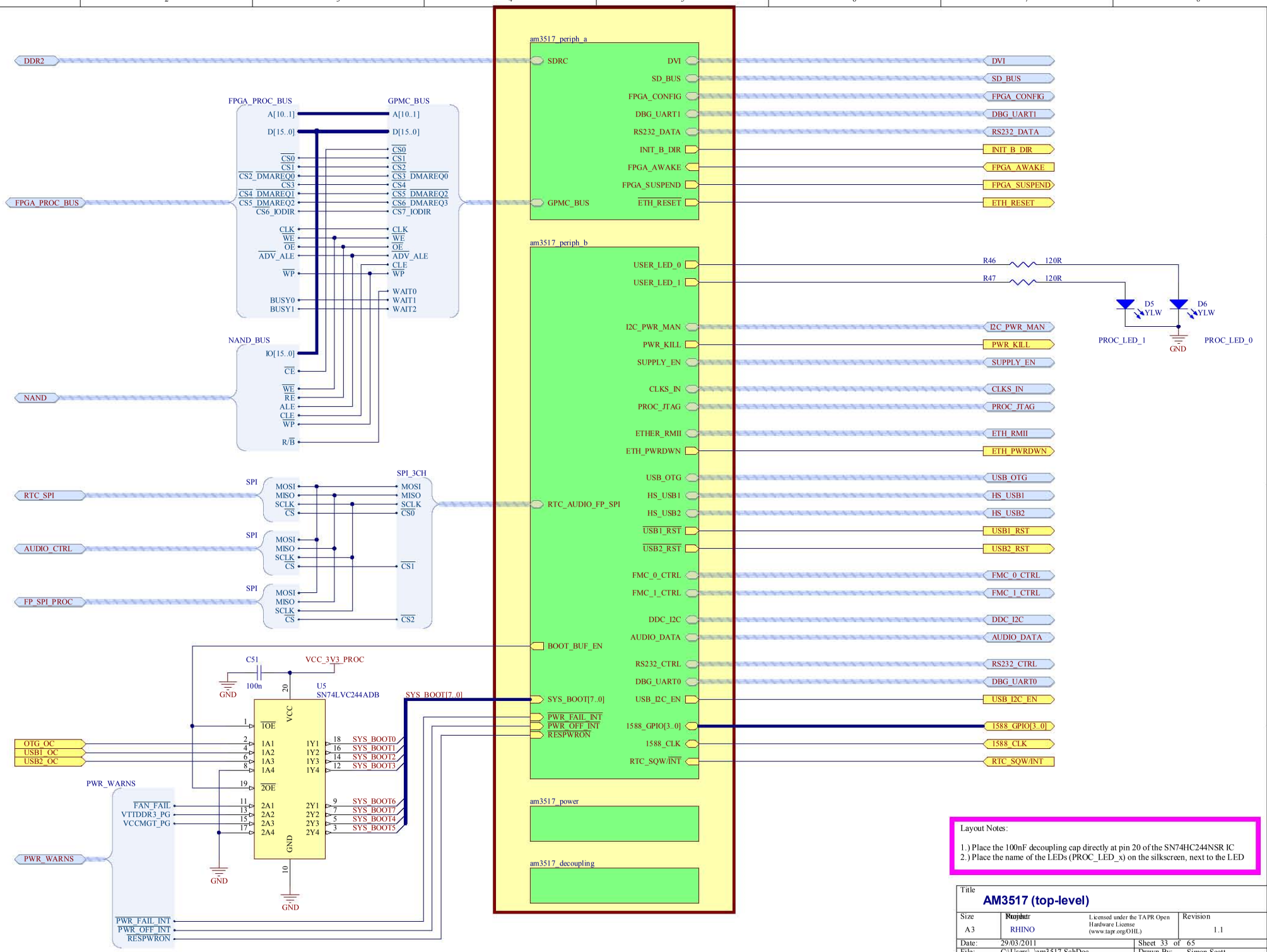


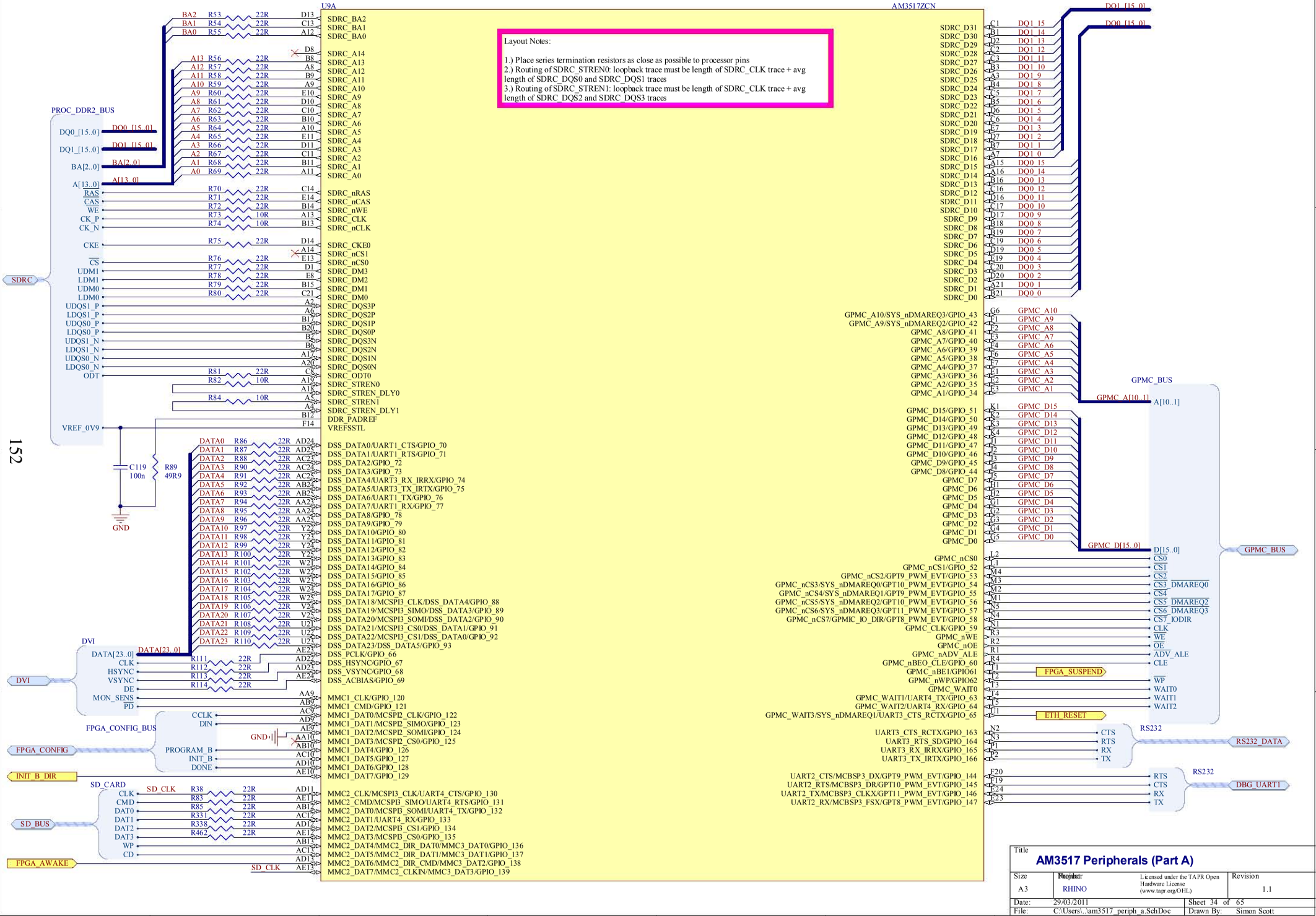
150

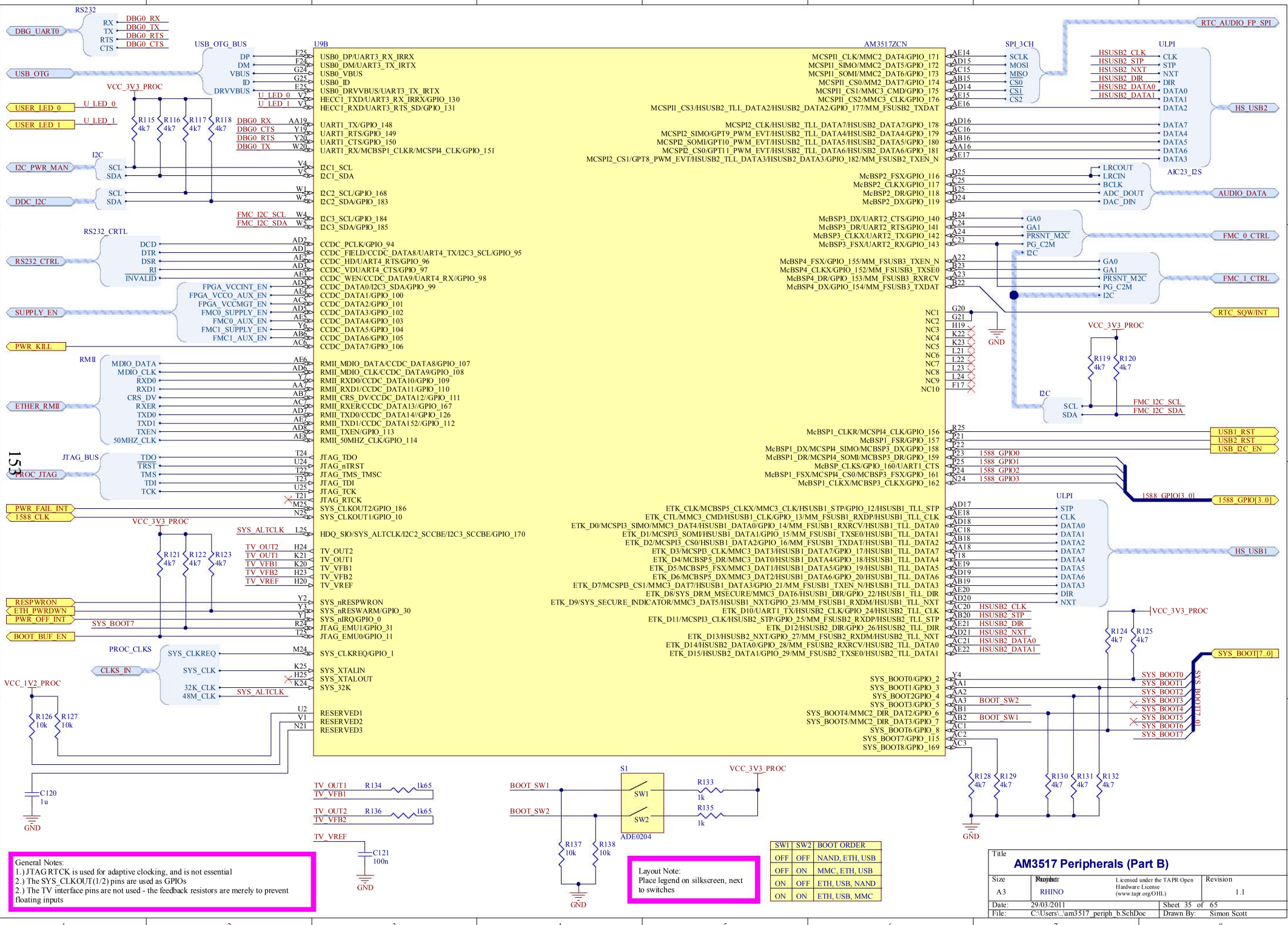
Layout Notes:

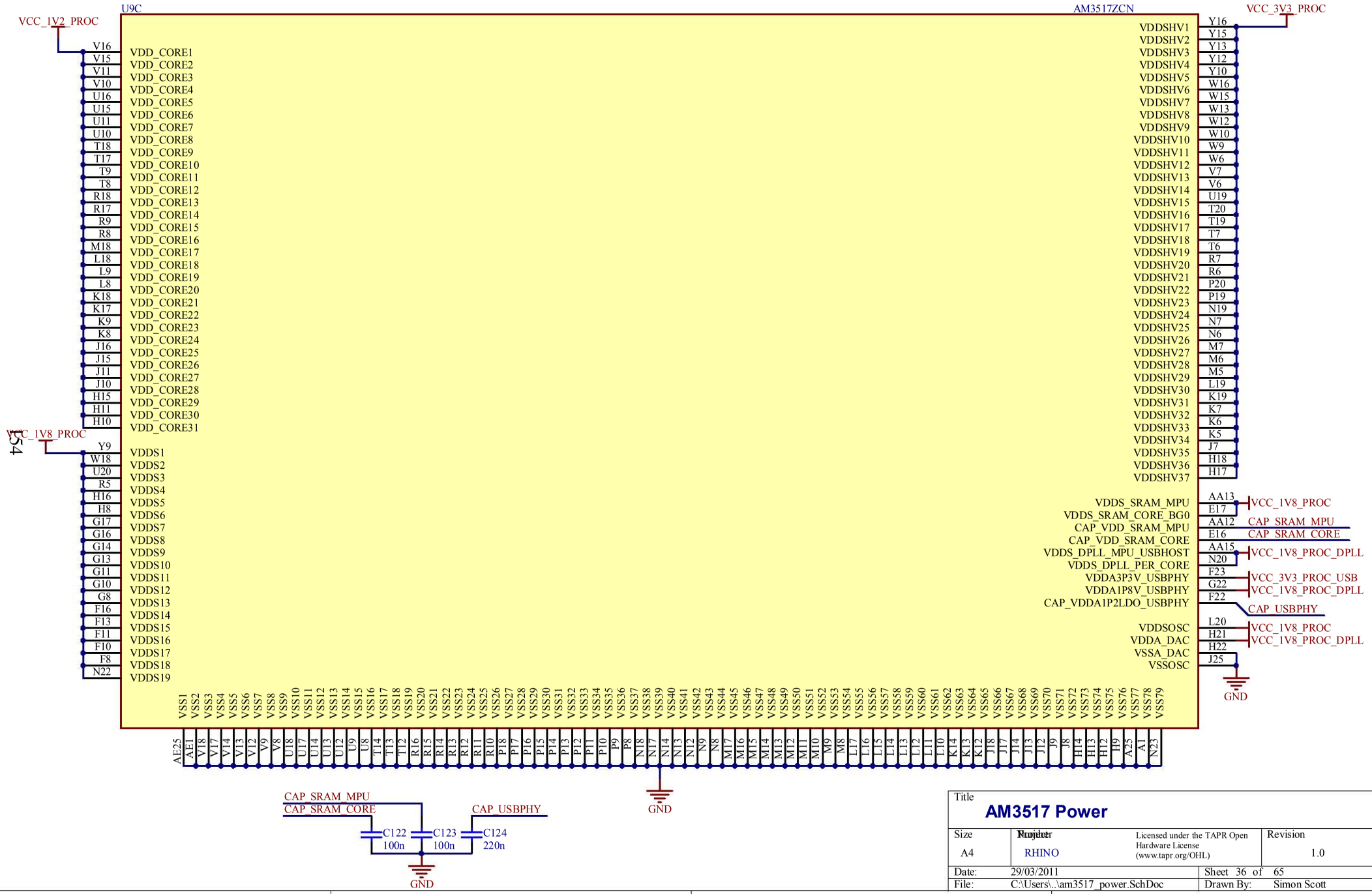
1.) Place decoupling caps as close to the power pins of each clock IC as possible
2.) Place 82R and 130R termination resistors, on pins 6 and 7 of CDCLVP1102, as close to the input pins as possible
3.) Place the 150R and 100nF termination resistors and caps as close to the output pins of the CDCLVP1102 device as possible
4.) Place the 0R resistors as close as possible to the pins of the CCPD-033 IC

Title: Spartan-6 Clocks			
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 32 of 65	
File: C:\Users\..\spartan6_clocks.SchDoc		Drawn By: Simon Scott	

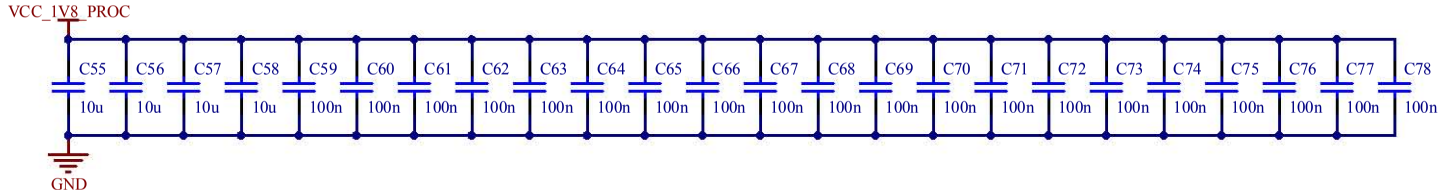

am3517_periph_a
SDRC
GPMC_BUS
DVI
SD_BUS
FPGA_CONFIG
DBG_UART1
RS232_DATA
INIT_B_DIR
FPGA_AWAKE
FPGA_SUSPEND
ETH_RESET
am3517_periph_b
USER_LED_0
USER_LED_1
I2C_PWR_MAN
PWR_KILL
SUPPLY_EN
CLKS_IN
PROC_JTAG
ETHER_RMII
ETH_PWRDWN
USB_OTG
HS_USB1
HS_USB2
USB1_RST
USB2_RST
FMC_0_CTRL
FMC_1_CTRL
DDC_I2C
AUDIO_DATA
RS232_CTRL
DBG_UART0
USB_I2C_EN
1588_GPIO[3..0]
1588_CLK
RTC_SQW/INT
RTC_AUDIO_FP_SPI
BOOT_BUF_EN
SYS_BOOT[7..0]
PWR_FAIL_INT
PWR_OFF_INT
RESPWRON
am3517_power
am3517_decoupling
DDR2
FPGA_PROC_BUS
NAND
RTC_SPI
AUDIO_CTRL
FP_SPI_PROC
PWR_WARNS
OTG_OC
USB1_OC
USB2_OC
FAN_FAIL
VTTDDR3_PG
VCCMGT_PG
R46 120R
R47 120R
D5 YLW
D6 YLW
PROC_LED_1
PROC_LED_0
GND
C51 100n
VCC_3V3_PROC
U5 SN74LVC244ADB
1OE
2OE
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
VCC
GND
SYS_BOOT0
SYS_BOOT1
SYS_BOOT2
SYS_BOOT3
SYS_BOOT4
SYS_BOOT5
SYS_BOOT6
SYS_BOOT7
FPGA_PROC_BUS
A[10..1]
D[15..0]
CS0
CS1
CS2_DMAREQ0
CS3
CS4_DMAREQ1
CS5_DMAREQ2
CS6_IODIR
CLK
WE
OE
ADV_ALE
WP
BUSY0
BUSY1
CS3_DMAREQ0
CS5_DMAREQ2
CS6_DMAREQ3
CS7_IODIR
CLE
WAIT0
WAIT1
WAIT2
NAND_BUS
IO[15..0]
CE
RE
ALE
R/B
SPI_3CH
SPI
MOSI
MISO
SCLK
CS
Layout Notes:
1.) Place the 100nF decoupling cap directly at pin 20 of the SN74HC244NSR IC
2.) Place the name of the LEDs (PROC_LED_x) on the silkscreen, next to the LED
Title
AM3517 (top-level)
Size
A3
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.1
Date:
29/03/2011
Sheet 33 of 65
File:
C:\Users\..\am3517.SchDoc
Drawn By:
Simon Scott



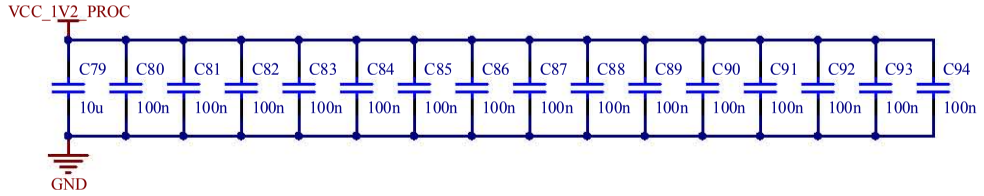

RS232
DBG_UART0
RX
TX
RTS
CTS
DBG0_RX
DBG0_TX
DBG0_RTS
DBG0_CTS
USB_OTG_BUS
USB_OTG
DP
DM
VBUS
ID
DRVVBUS
VCC_3V3_PROC
USER_LED_0
USER_LED_1
U_LED_0
U_LED_1
R115 4k7
R116 4k7
R117 4k7
R118 4k7
I2C
I2C_PWR_MAN
DDC_I2C
SCL
SDA
RS232_CRTL
RS232_CTRL
DCD
DTR
DSR
RI
INVALID
SUPPLY_EN
FPGA_VCCINT_EN
FPGA_VCCO_AUX_EN
FPGA_VCCMGT_EN
FMC0_SUPPLY_EN
FMC0_AUX_EN
FMC1_SUPPLY_EN
FMC1_AUX_EN
PWR_KILL
RMII
ETHER_RMII
MDIO_DATA
MDIO_CLK
RXD0
RXD1
CRS_DV
RXER
TXD0
TXD1
TXEN
50MHZ_CLK
JTAG_BUS
PROC_JTAG
TDO
TRST
TMS
TDI
TCK
PWR_FAIL_INT
1588_CLK
VCC_3V3_PROC
R121 4k7
R122 4k7
R123 4k7
RESPWRON
ETH_PWRDWN
PWR_OFF_INT
BOOT_BUF_EN
SYS_BOOT7
PROC_CLKS
CLKS_IN
SYS_CLKREQ
SYS_CLK
32K_CLK
48M_CLK
SYS_ALTCLK
VCC_1V2_PROC
R126 10k
R127 10k
C120 1u
GND
U9B
USB0_DP/UART3_RX_IRRX
USB0_DM/UART3_TX_IRTX
USB0_VBUS
USB0_ID
USB0_DRVVBUS/UART3_TX_IRTX
HECC1_TXD/UART3_RX_IRRX/GPIO_130
HECC1_RXD/UART3_RTS_SD/GPIO_131
UART1_TX/GPIO_148
UART1_RTS/GPIO_149
UART1_CTS/GPIO_150
UART1_RX/MCBSP1_CLKR/MCSPI4_CLK/GPIO_151
I2C1_SCL
I2C1_SDA
I2C2_SCL/GPIO_168
I2C2_SDA/GPIO_183
I2C3_SCL/GPIO_184
I2C3_SDA/GPIO_185
CCDC_PCLK/GPIO_94
CCDC_FIELD/CCDC_DATA8/UART4_TX/I2C3_SCL/GPIO_95
CCDC_HD/UART4_RTS/GPIO_96
CCDC_VDUART4_CTS/GPIO_97
CCDC_WEN/CCDC_DATA9/UART4_RX//GPIO_98
CCDC_DATA0/I2C3_SDA/GPIO_99
CCDC_DATA1/GPIO_100
CCDC_DATA2/GPIO_101
CCDC_DATA3/GPIO_102
CCDC_DATA4/GPIO_103
CCDC_DATA5/GPIO_104
CCDC_DATA6/GPIO_105
CCDC_DATA7/GPIO_106
RMII_MDIO_DATA/CCDC_DATA8/GPIO_107
RMII_MDIO_CLK/CCDC_DATA9/GPIO_108
RMII_RXD0/CCDC_DATA10/GPIO_109
RMII_RXD1/CCDC_DATA11/GPIO_110
RMII_CRS_DV/CCDC_DATA12/GPIO_111
RMII_RXER/CCDC_DATA13//GPIO_167
RMII_TXD0/CCDC_DATA14//GPIO_126
RMII_TXD1/CCDC_DATA152//GPIO_112
RMII_TXEN/GPIO_113
RMII_50MHZ_CLK/GPIO_114
JTAG_TDO
JTAG_nTRST
JTAG_TMS_TMSC
JTAG_TDI
JTAG_TCK
JTAG_RTCK
SYS_CLKOUT2/GPIO_186
SYS_CLKOUT1/GPIO_10
HDQ_SIO/SYS_ALTCLK/I2C2_SCCBE/I2C3_SCCBE/GPIO_170
TV_OUT2
TV_OUT1
TV_VFB1
TV_VFB2
TV_VREF
SYS_nRESPWRON
SYS_nRESWARM/GPIO_30
SYS_nIRQ/GPIO_0
JTAG_EMU1/GPIO_31
JTAG_EMU0/GPIO_11
SYS_CLKREQ/GPIO_1
SYS_XTALIN
SYS_XTALOUT
SYS_32K
RESERVED1
RESERVED2
RESERVED3
AM3517ZCN
MCSPI1_CLK/MMC2_DAT4/GPIO_171
MCSPI1_SIMO/MMC2_DAT5/GPIO_172
MCSPI1_SOMI/MMC2_DAT6/GPIO_173
MCSPI1_CS0/MM2_DAT7/GPIO_174
MCSPI1_CS1/MMC3_CMD/GPIO_175
MCSPI1_CS2/MMC3_CLK/GPIO_176
MCSPI1_CS3/HSUSB2_TLL_DATA2/HSUSB2_DATA2/GPIO_177/MM_FSUSB2_TXDAT
MCSPI2_CLK/HSUSB2_TLL_DATA7/HSUSB2_DATA7/GPIO_178
MCSPI2_SIMO/GPT9_PWM_EVT/HSUSB2_TLL_DATA4/HSUSB2_DATA4/GPIO_179
MCSPI2_SOMI/GPT10_PWM_EVT/HSUSB2_TLL_DATA5/HSUSB2_DATA5/GPIO_180
MCSPI2_CS0/GPT11_PWM_EVT/HSUSB2_TLL_DATA6/HSUSB2_DATA6/GPIO_181
MCSPI2_CS1/GPT8_PWM_EVT/HSUSB2_TLL_DATA3/HSUSB2_DATA3/GPIO_182/MM_FSUSB2_TXEN_N
McBSP2_FSX/GPIO_116
McBSP2_CLKX/GPIO_117
McBSP2_DR/GPIO_118
McBSP2_DX/GPIO_119
McBSP3_DX/UART2_CTS/GPIO_140
McBSP3_DR/UART2_RTS/GPIO_141
McBSP3_CLKX/UART2_TX/GPIO_142
McBSP3_FSX/UART2_RX/GPIO_143
McBSP4_FSX/GPIO_155/MM_FSUSB3_TXEN_N
McBSP4_CLKX/GPIO_152/MM_FSUSB3_TXSE0
McBSP4_DR/GPIO_153/MM_FSUSB3_RXRCV
McBSP4_DX/GPIO_154/MM_FSUSB3_TXDAT
NC1
NC2
NC3
NC4
NC5
NC6
NC7
NC8
NC9
NC10
McBSP1_CLKR/MCSPI4_CLK/GPIO_156
McBSP1_FSR/GPIO_157
McBSP1_DX/MCSPI4_SIMO/MCBSP3_DX/GPIO_158
McBSP1_DR/MCSPI4_SOMI/MCBSP3_DR/GPIO_159
McBSP_CLKS/GPIO_160/UART1_CTS
McBSP1_FSX/MCSPI4_CS0/MCBSP3_FSX/GPIO_161
McBSP1_CLKX/MCBSP3_CLKX/GPIO_162
ETK_CLK/MCBSP5_CLKX/MMC3_CLK/HSUSB1_STP/GPIO_12/HSUSB1_TLL_STP
ETK_CTL/MMC3_CMD/HSUSB1_CLK/GPIO_13/MM_FSUSB1_RXDP/HSUSB1_TLL_CLK
ETK_D0/MCSPI3_SIMO/MMC3_DAT4/HSUSB1_DATA0/GPIO_14/MM_FSUSB1_RXRCV/HSUSB1_TLL_DATA0
ETK_D1/MCSPI3_SOMI/HSUSB1_DATA1/GPIO_15/MM_FSUSB1_TXSE0/HSUSB1_TLL_DATA1
ETK_D2/MCSPI3_CS0/HSUSB1_DATA2/GPIO_16/MM_FSUSB1_TXDAT/HSUSB1_TLL_DATA2
ETK_D3/MCSPI3_CLK/MMC3_DAT3/HSUSB1_DATA7/GPIO_17/HSUSB1_TLL_DATA7
ETK_D4/MCBSP5_DR/MMC3_DAT0/HSUSB1_DATA4/GPIO_18/HSUSB1_TLL_DATA4
ETK_D5/MCBSP5_FSX/MMC3_DAT1/HSUSB1_DATA5/GPIO_19/HSUSB1_TLL_DATA5
ETK_D6/MCBSP5_DX/MMC3_DAT2/HSUSB1_DATA6/GPIO_20/HSUSB1_TLL_DATA6
ETK_D7/MCSPI3_CS1/MMC3_DAT7/HSUSB1_DATA3/GPIO_21/MM_FSUSB1_TXEN_N/HSUSB1_TLL_DATA3
ETK_D8/SYS_DRM_MSECURE/MMC3_DAT6/HSUSB1_DIR/GPIO_22/HSUSB1_TLL_DIR
ETK_D9/SYS_SECURE_INDICATOR/MMC3_DAT5/HSUSB1_NXT/GPIO_23/MM_FSUSB1_RXDM/HSUSB1_TLL_NXT
ETK_D10/UART1_TX/HSUSB2_CLK/GPIO_24/HSUSB2_TLL_CLK
ETK_D11/MCSPI3_CLK/HSUSB2_STP/GPIO_25/MM_FSUSB2_RXDP/HSUSB2_TLL_STP
ETK_D12/HSUSB2_DIR/GPIO_26/HSUSB2_TLL_DIR
ETK_D13/HSUSB2_NXT/GPIO_27/MM_FSUSB2_RXDM/HSUSB2_TLL_NXT
ETK_D14/HSUSB2_DATA0/GPIO_28/MM_FSUSB2_RXRCV/HSUSB2_TLL_DATA0
ETK_D15/HSUSB2_DATA1/GPIO_29/MM_FSUSB2_TXSE0/HSUSB2_TLL_DATA1
SYS_BOOT0/GPIO_2
SYS_BOOT1/GPIO_3
SYS_BOOT2GPIO_4
SYS_BOOT3/GPIO_5
SYS_BOOT4/MMC2_DIR_DAT2/GPIO_6
SYS_BOOT5/MMC2_DIR_DAT3/GPIO_7
SYS_BOOT6/GPIO_8
SYS_BOOT7/GPIO_115
SYS_BOOT8/GPIO_169
SPI_3CH
SCLK
MOSI
MISO
CS0
CS1
CS2
ULPI
HSUSB2_CLK
HSUSB2_STP
HSUSB2_NXT
HSUSB2_DIR
HSUSB2_DATA0
HSUSB2_DATA1
CLK
STP
NXT
DIR
DATA0
DATA1
DATA2
DATA3
DATA4
DATA5
DATA6
DATA7
HS_USB2
RTC_AUDIO_FP_SPI
AIC23_I2S
LRCOUT
LRCIN
BCLK
ADC_DOUT
DAC_DIN
AUDIO_DATA
FMC_0_CTRL
FMC_1_CTRL
GA0
GA1
PRSNT_M2C
PG_C2M
I2C
RTC_SQW/INT
VCC_3V3_PROC
R119 4k7
R120 4k7
FMC_I2C_SCL
FMC_I2C_SDA
USB1_RST
USB2_RST
USB_I2C_EN
1588_GPIO0
1588_GPIO1
1588_GPIO2
1588_GPIO3
1588_GPIO[3..0]
HS_USB1
VCC_3V3_PROC
R124 4k7
R125 4k7
SYS_BOOT[7..0]
BOOT_SW2
BOOT_SW1
R128 4k7
R129 4k7
R130 4k7
R131 4k7
R132 4k7
GND
TV_OUT1
TV_VFB1
R134 1k65
TV_OUT2
TV_VFB2
R136 1k65
TV_VREF
C121 100n
S1
SW1
SW2
ADE0204
BOOT_SW1
BOOT_SW2
R133 1k
R135 1k
R137 10k
R138 10k
SW1 SW2 BOOT ORDER
OFF OFF NAND, ETH, USB
OFF ON MMC, ETH, USB
ON OFF ETH, USB, NAND
ON ON ETH, USB, MMC
General Notes:
1.) JTAG RTCK is used for adaptive clocking, and is not essential
2.) The SYS_CLKOUT(1/2) pins are used as GPIOs
2.) The TV interface pins are not used - the feedback resistors are merely to prevent floating inputs
Layout Note:
Place legend on silkscreen, next to switches
Title
AM3517 Peripherals (Part B)
Size
A3
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.1
Date:
29/03/2011
Sheet 35 of 65
File:
C:\Users\..\am3517_periph_b.SchDoc
Drawn By:
Simon Scott



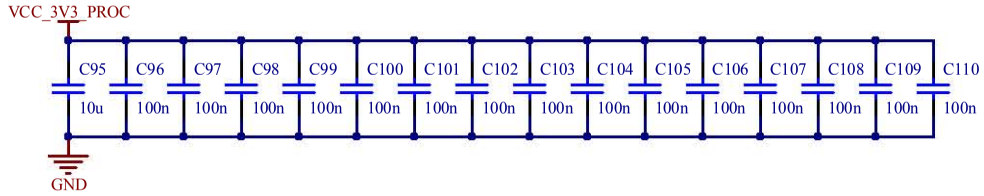
Title	AM3517 Power
Size	A4
Number	RHINO
Revision	1.0
Date:	29/03/2011
File:	C:\Users\..\am3517_power.SchDoc
Sheet	36 of 65
Drawn By:	Simon Scott



Bypass capacitors for AM3517 VDDS pins:
1.) Place caps as close to the VDDS pins as possible (prefereably within 7mm)
2.) Preference should be given to the placement of the 100nF caps over the 10uF caps

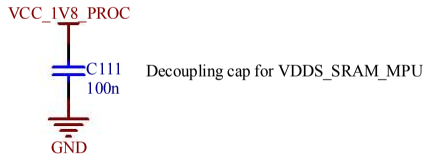


Bypass capacitors for AM3517 VDD_CORE pins:
1.) Place caps close to VDD_CORE pins
2.) Preference should be given to the placement of the 100nF caps over the 10uF cap

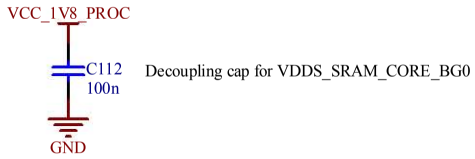


Bypass capacitors for AM3517 VDDSHV pins:
1.) Place caps close to VDDS_HV pins
2.) Preference should be given to the placement of the 100nF caps over the 10uF cap

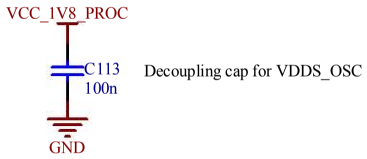
155



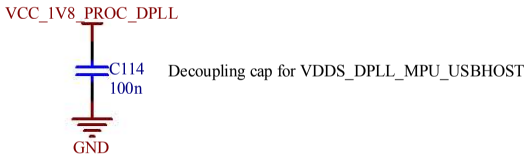
Decoupling cap for VDDS_SRAM_MPU



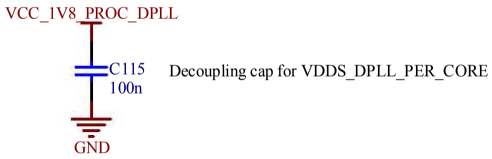
Decoupling cap for VDDS_SRAM_CORE_BG0



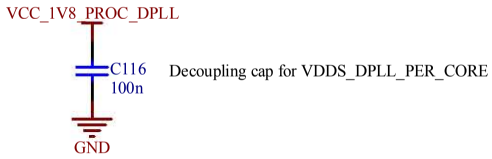
Decoupling cap for VDDS_OSC



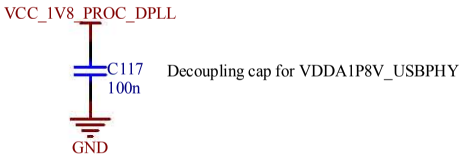
Decoupling cap for VDDS_DPLL_MPU_USBHOST



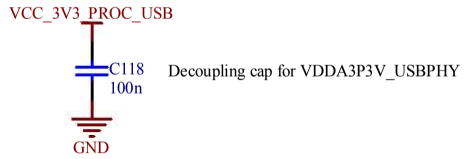
Decoupling cap for VDDS_DPLL_PER_CORE



Decoupling cap for VDDS_DPLL_PER_CORE



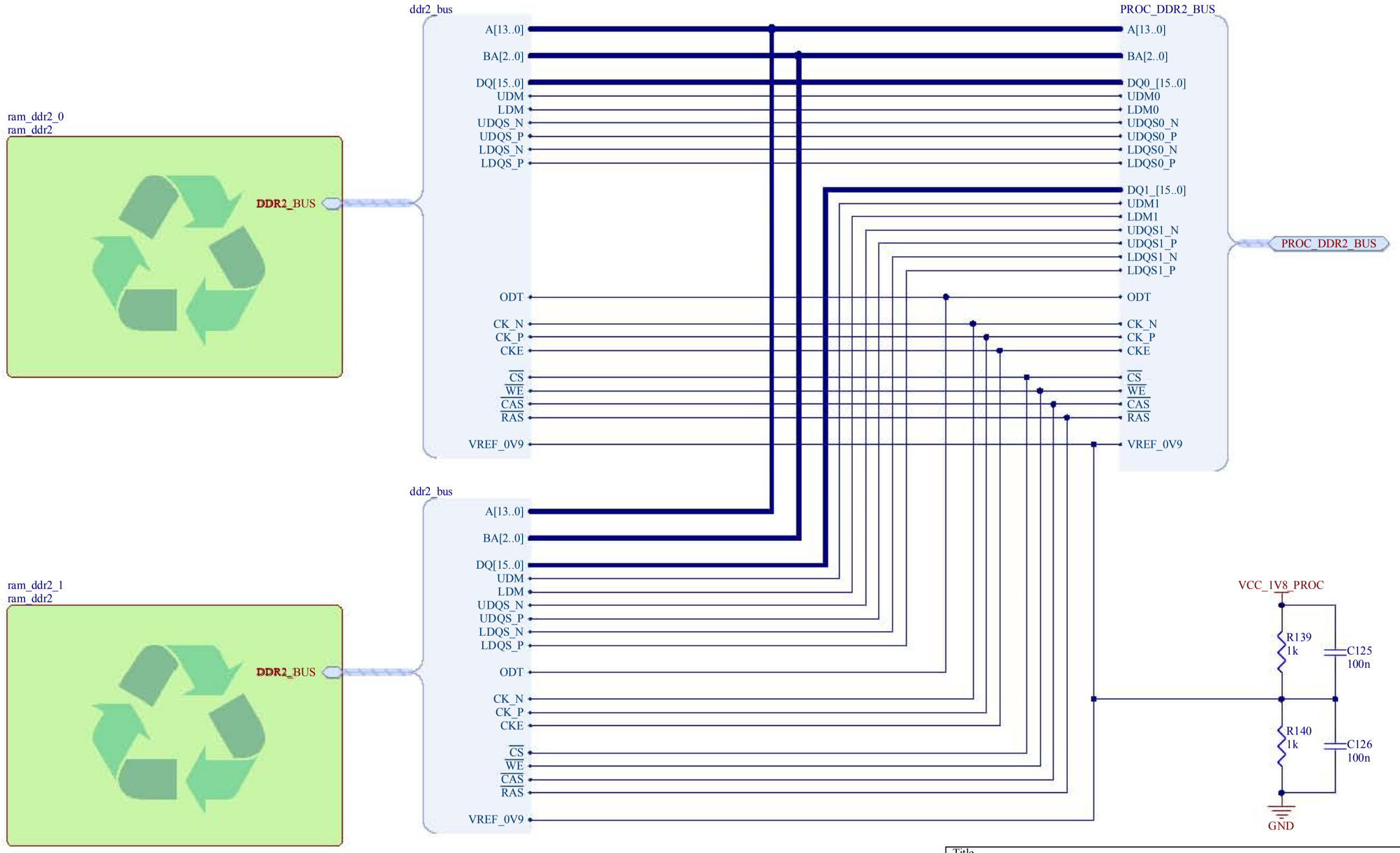
Decoupling cap for VDDA1P8V_USBPHY



Decoupling cap for VDDA3P3V_USBPHY

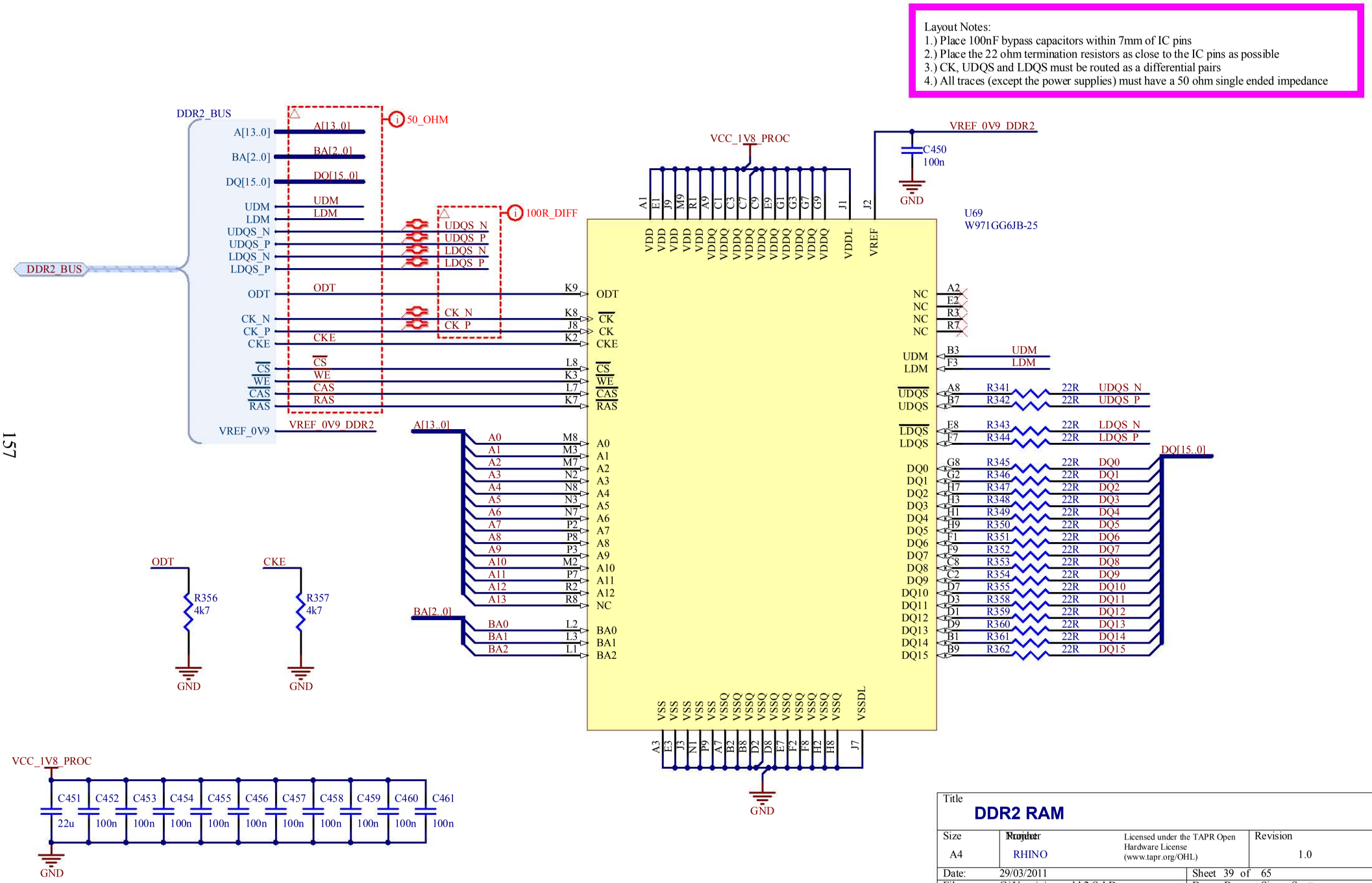
Layout Notes:
1.) Each decoupling capacitor should be connected to a group of 1, 2 or 3 adjacent power balls, and then to the closest ground ball.
2.) In the case of interconnected power pins, first connect the decoupling cap, then interconnect the pins.

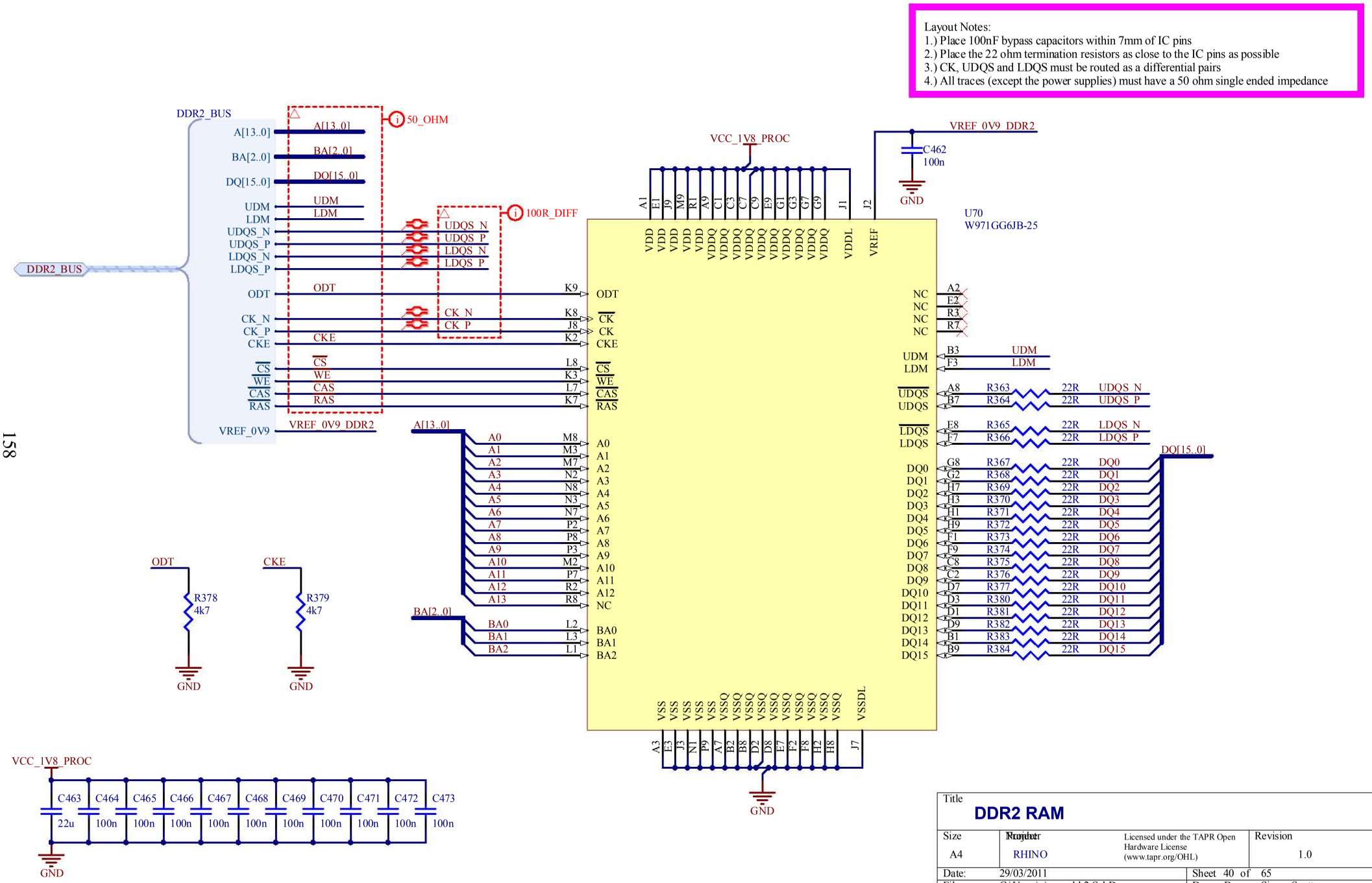
Title: AM3517 Supply Decoupling Caps			
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 37 of 65	
File: C:\Users\..\am3517_decoupling.SchDoc		Drawn By: Simon Scott	

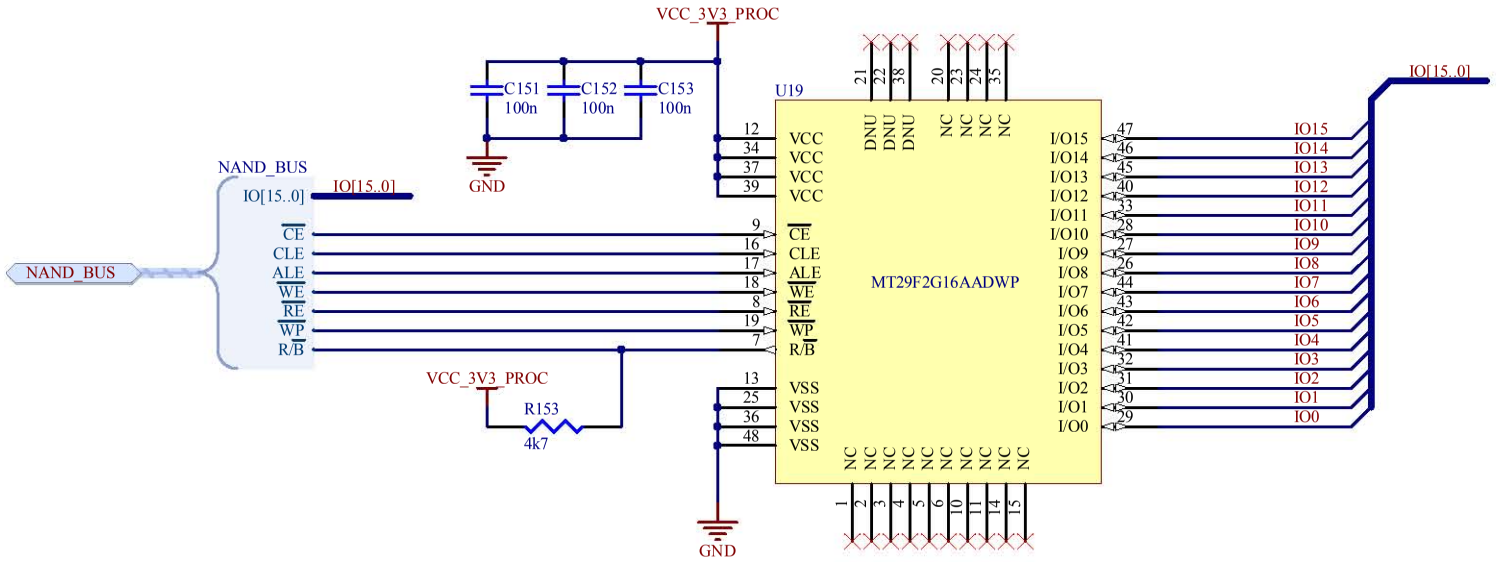


156

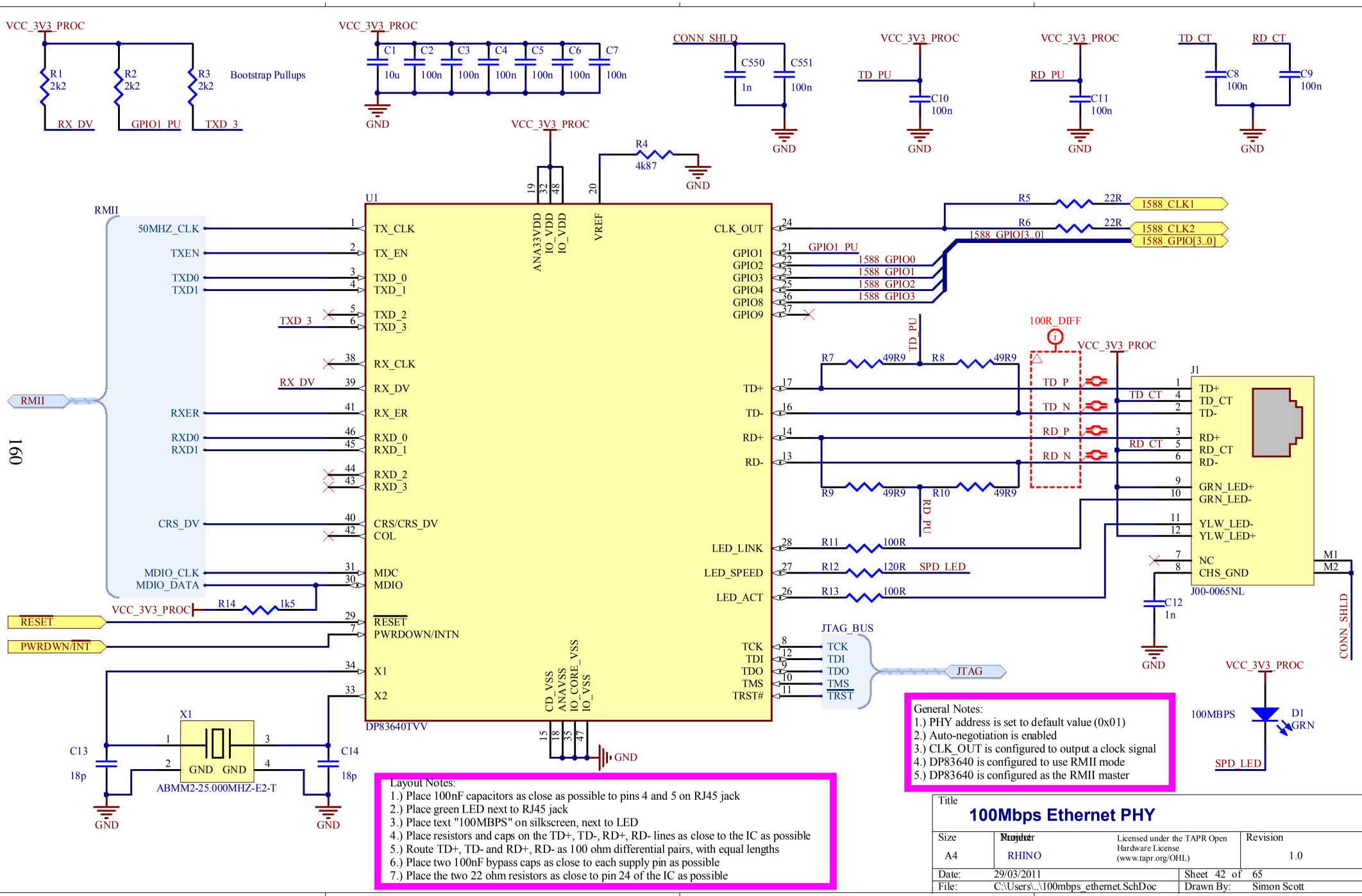
Title	**AM3517 DDR2 RAM (top level)**		
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 38 of 65	
File: C:\Users\..\am3517_ram.SchDoc		Drawn By: Simon Scott	


Layout Notes:
1.) Place 100nF bypass capacitors within 7mm of IC pins
2.) Place the 22 ohm termination resistors as close to the IC pins as possible
3.) CK, UDQS and LDQS must be routed as a differential pairs
4.) All traces (except the power supplies) must have a 50 ohm single ended impedance
DDR2_BUS
A[13..0]
BA[2..0]
DQ[15..0]
UDM
LDM
UDQS_N
UDQS_P
LDQS_N
LDQS_P
ODT
CK_N
CK_P
CKE
CS
WE
CAS
RAS
VREF_0V9
50_OHM
100R_DIFF
VCC_1V8_PROC
VREF_0V9_DDR2
C450
100n
GND
U69
W971GG6JB-25
R356
4k7
R357
4k7
C451
C452
C453
C454
C455
C456
C457
C458
C459
C460
C461
22u
100n
Title
DDR2 RAM
Size
A4
Number
RHINO
Revision
1.0
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Date:
29/03/2011
Sheet 39 of 65
File:
C:\Users\..\ram_ddr2.SchDoc
Drawn By:
Simon Scott
157


Layout Notes:
1.) Place 100nF bypass capacitors within 7mm of IC pins
2.) Place the 22 ohm termination resistors as close to the IC pins as possible
3.) CK, UDQS and LDQS must be routed as a differential pairs
4.) All traces (except the power supplies) must have a 50 ohm single ended impedance
DDR2_BUS
50_OHM
100R_DIFF
VCC_1V8_PROC
VREF_0V9_DDR2
C462
100n
GND
U70
W971GG6JB-25
ODT
CKE
R378
4k7
R379
4k7
C463 C464 C465 C466 C467 C468 C469 C470 C471 C472 C473
22u 100n 100n 100n 100n 100n 100n 100n 100n 100n 100n
Title
DDR2 RAM
Size
A4
Number
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date: 29/03/2011
Sheet 40 of 65
File: C:\Users\..\ram_ddr2.SchDoc
Drawn By: Simon Scott
158



Title	**NAND Flash**		
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 41 of 65	
File: C:\Users\..\nand_flash.SchDoc		Drawn By: Simon Scott	

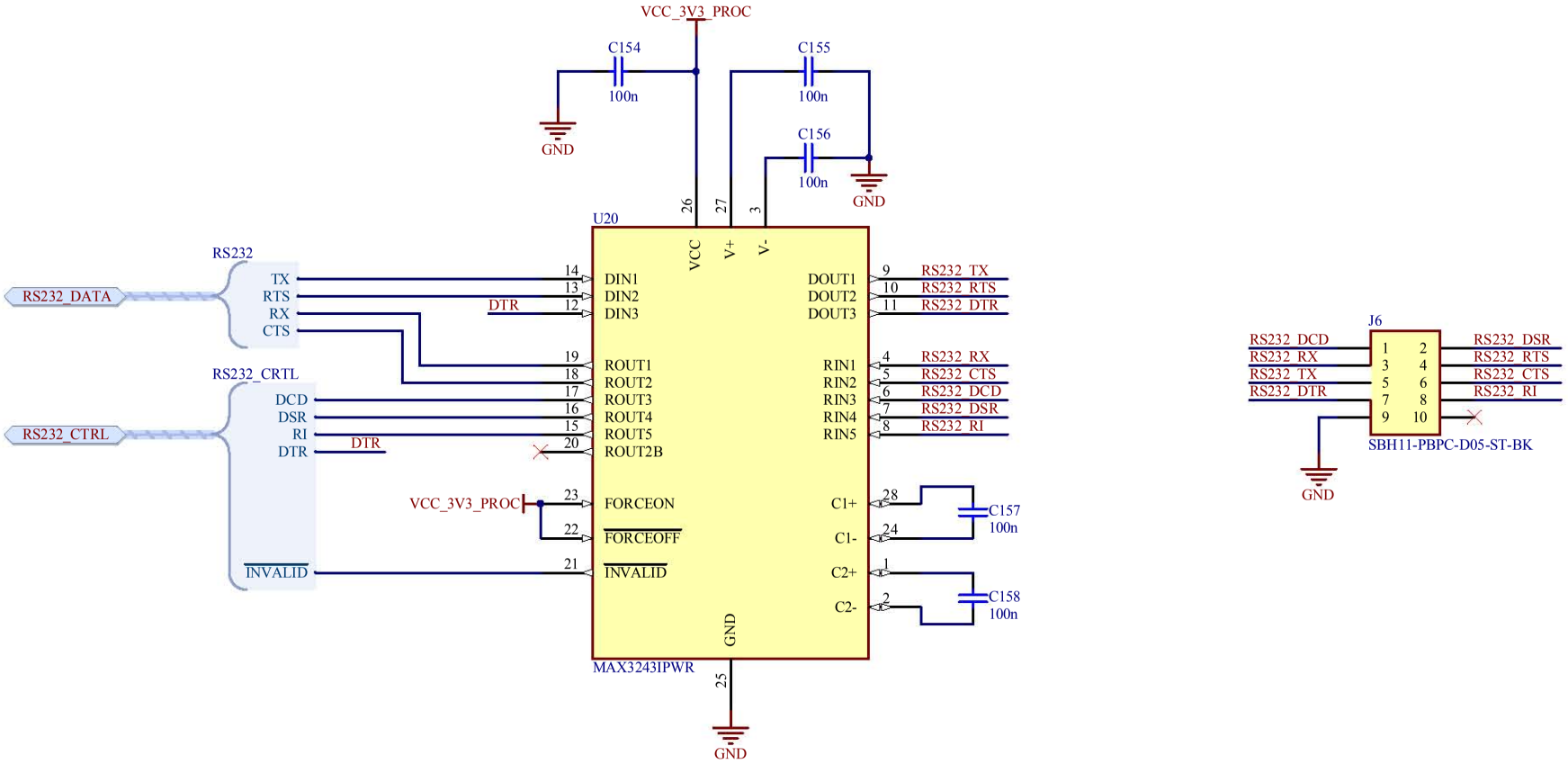


160

Layout Notes:
1.) Place 100nF capacitors as close as possible to pins 4 and 5 on RJ45 jack
2.) Place green LED next to RJ45 jack
3.) Place text "100MBPS" on silkscreen, next to LED
4.) Place resistors and caps on the TD+, TD-, RD+, RD- lines as close to the IC as possible
5.) Route TD+, TD- and RD+, RD- as 100 ohm differential pairs, with equal lengths
6.) Place two 100nF bypass caps as close to each supply pin as possible
7.) Place the two 22 ohm resistors as close to pin 24 of the IC as possible

General Notes:
1.) PHY address is set to default value (0x01)
2.) Auto-negotiation is enabled
3.) CLK_OUT is configured to output a clock signal
4.) DP83640 is configured to use RMII mode
5.) DP83640 is configured as the RMII master

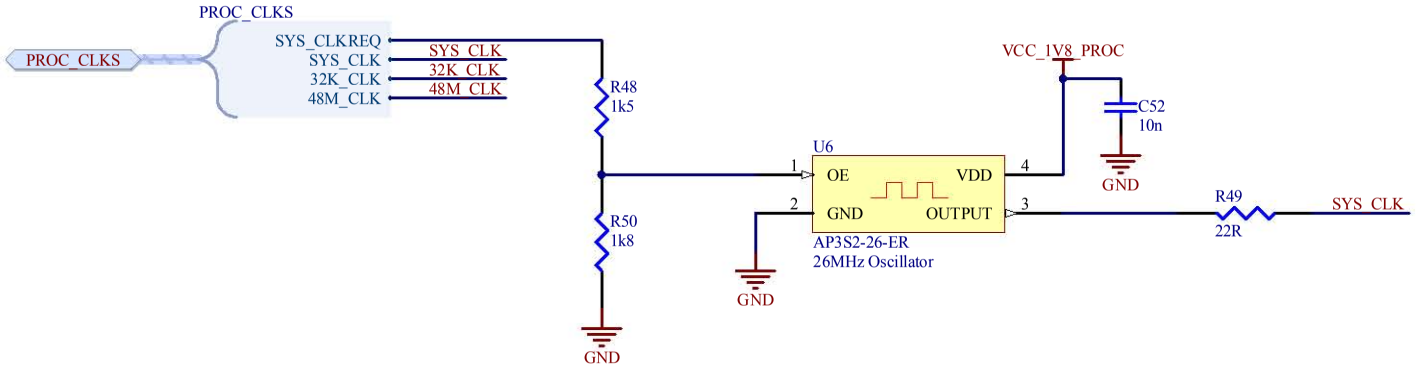
Title	100Mbps Ethernet PHY		
Size	Number	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision
A4	RHINO		1.0
Date:	29/03/2011		Sheet 42 of 65
File:	C:\Users\..\100mbps_ethernet.SchDoc		Drawn By: Simon Scott

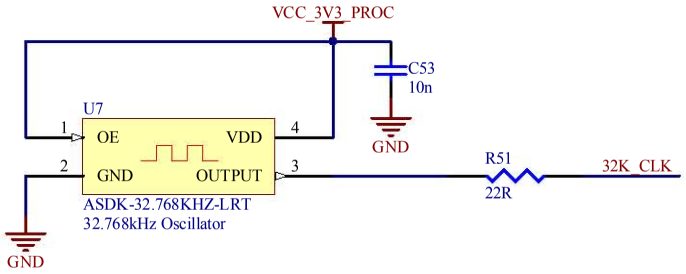


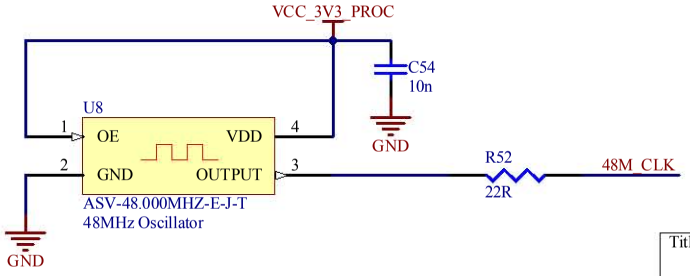
161

Layout Notes:
1.) Place the 100nF decoupling caps as close to the power pins of the IC as possible

Title	**RS-232 Header for Peripherals**		
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 43 of 65	
File: C:\Users\..\periph_rs232.SchDoc		Drawn By: Simon Scott	

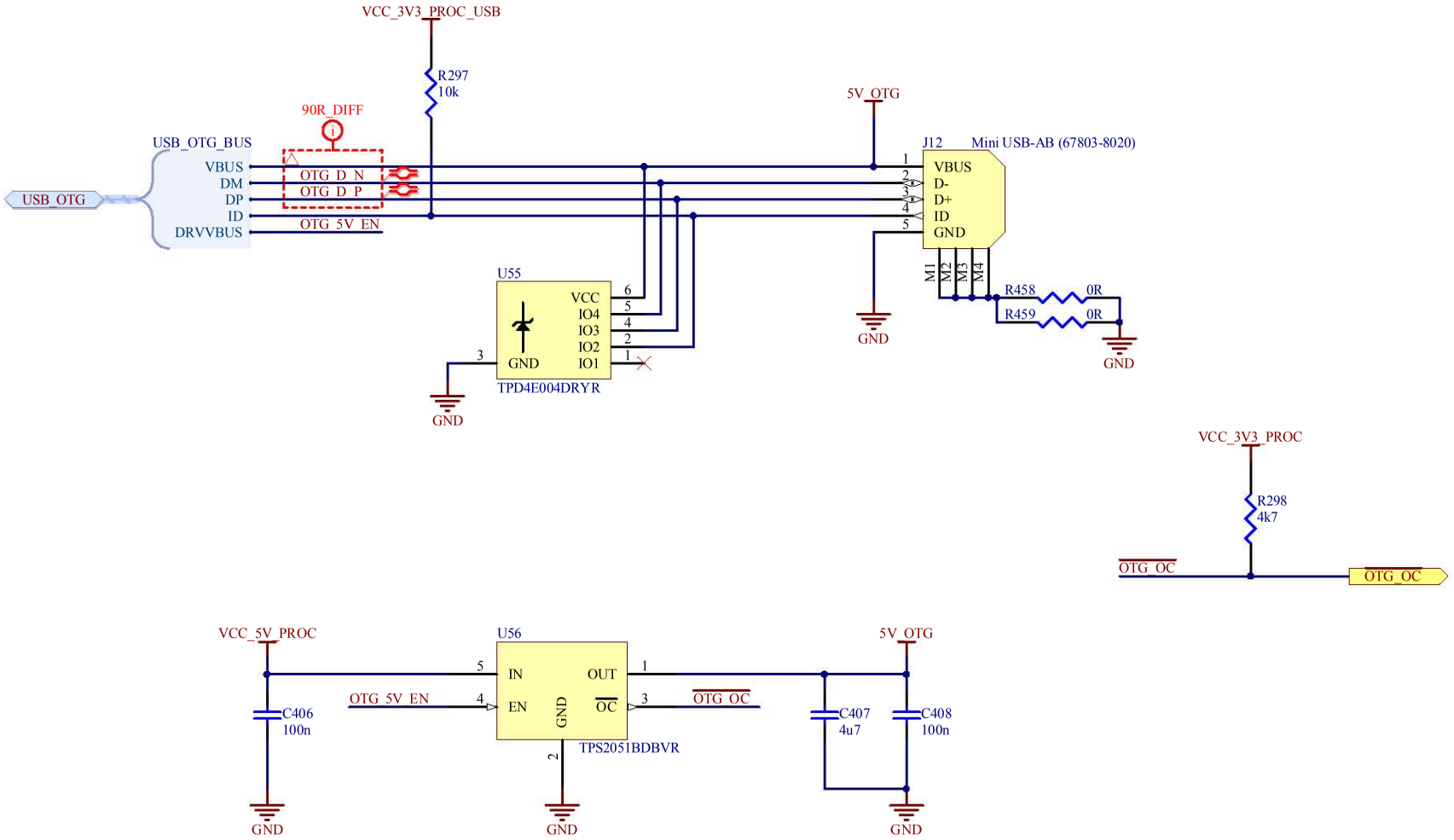






Layout Notes:
1.) Place decoupling caps as close to the power pins of each clock IC as possible

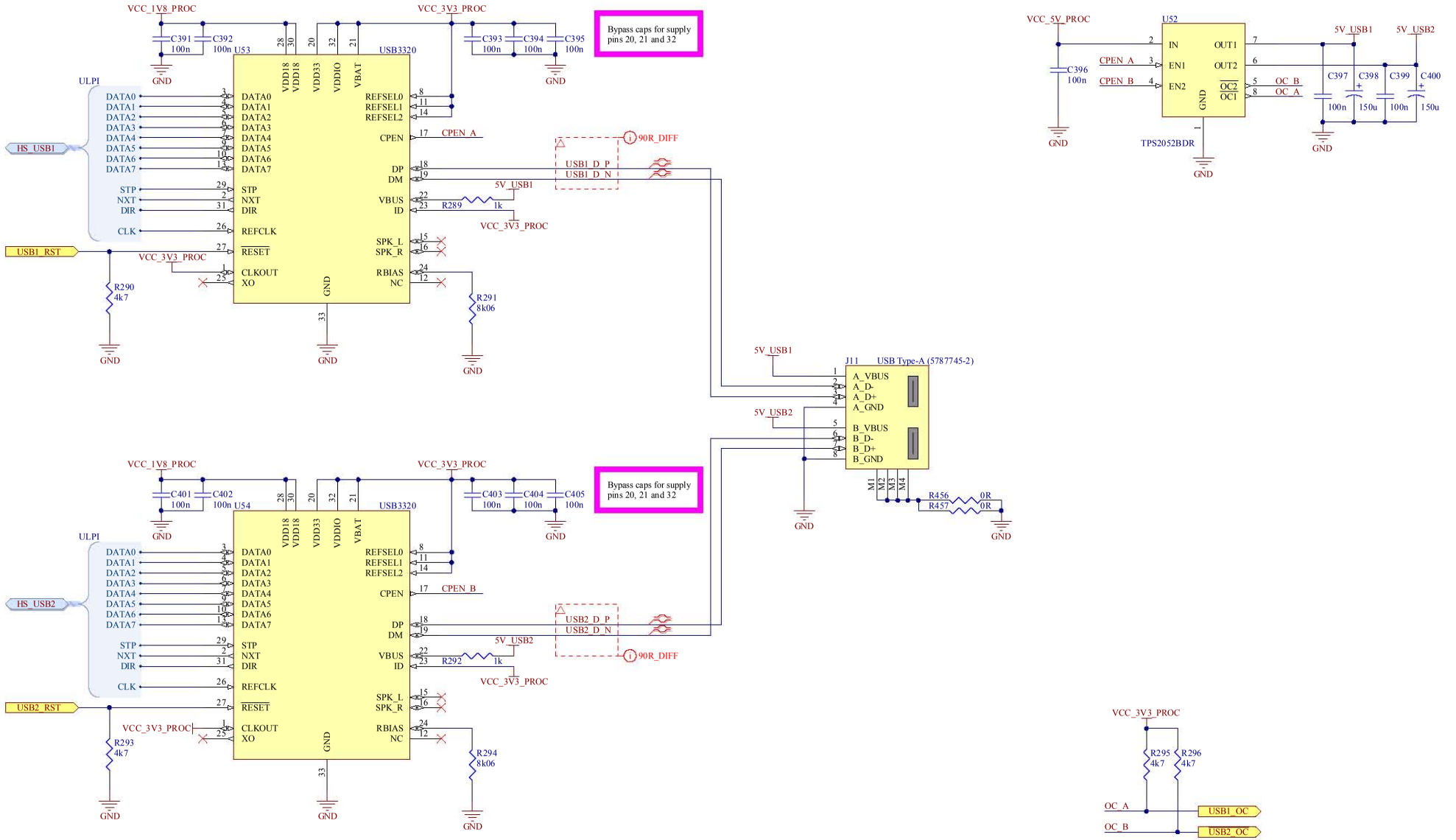
Title: AM3517 Clocks			
Size: A4	Number: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 44 of 65	
File: C:\Users\..\am3517_clocks.SchDoc		Drawn By: Simon Scott	



Layout Notes:

1.) USB data pairs (DM and DP) must be routed as differential pairs, with 90 ohms differential impedance and equal length
2.) The TPD4E004DRYR IC must be placed directly on the DM and DP lines, and not connected via stubs
3.) The 4.7uF and 100nF capacitors should be placed as close as possible to the USB connector
4.) The TPD4E004DRYR should also be placed close to the USB connector, directly after the above mentioned caps

Title	**USB On-the-Go**		
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 45 of 65	
File: C:\Users\..\usb_otg.SchDoc		Drawn By: Simon Scott	



164

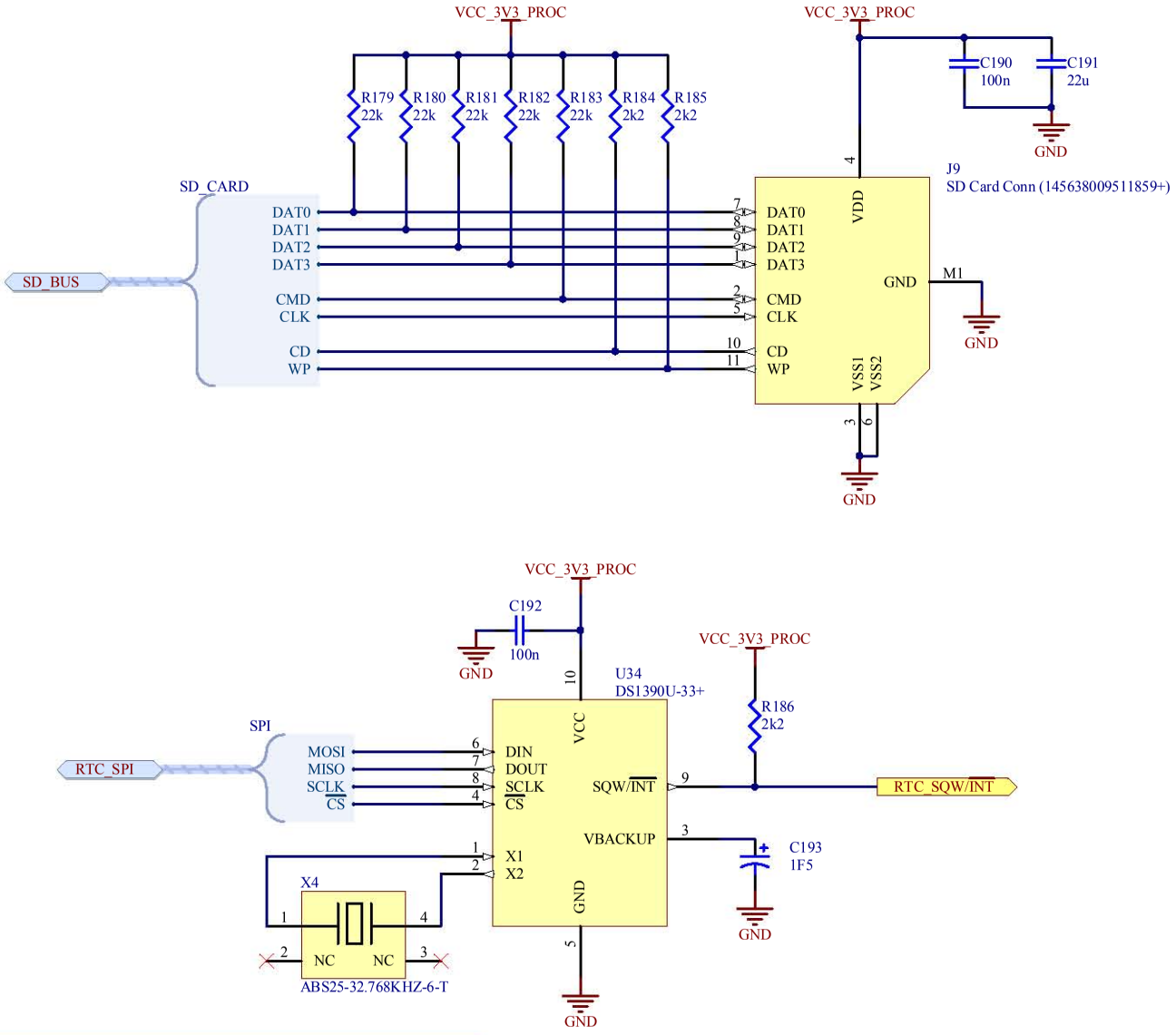
Layout Notes:

1.) All USB data pairs (DM and DP) must be routed as differential pairs, with 90 ohms differential impedance
2.) The 100nf and 100uF capacitors on the VUSB pins of the USB connectors, must be placed as close to the connector as possible
3.) The decoupling caps on IC power supply pins, must be placed as close to pins as possible
4.) The 8k06 resistor must be placed as close as possible to pin 24 of the IC

General Notes:

1.) The USB3320 devices are configured for HOST-only mode

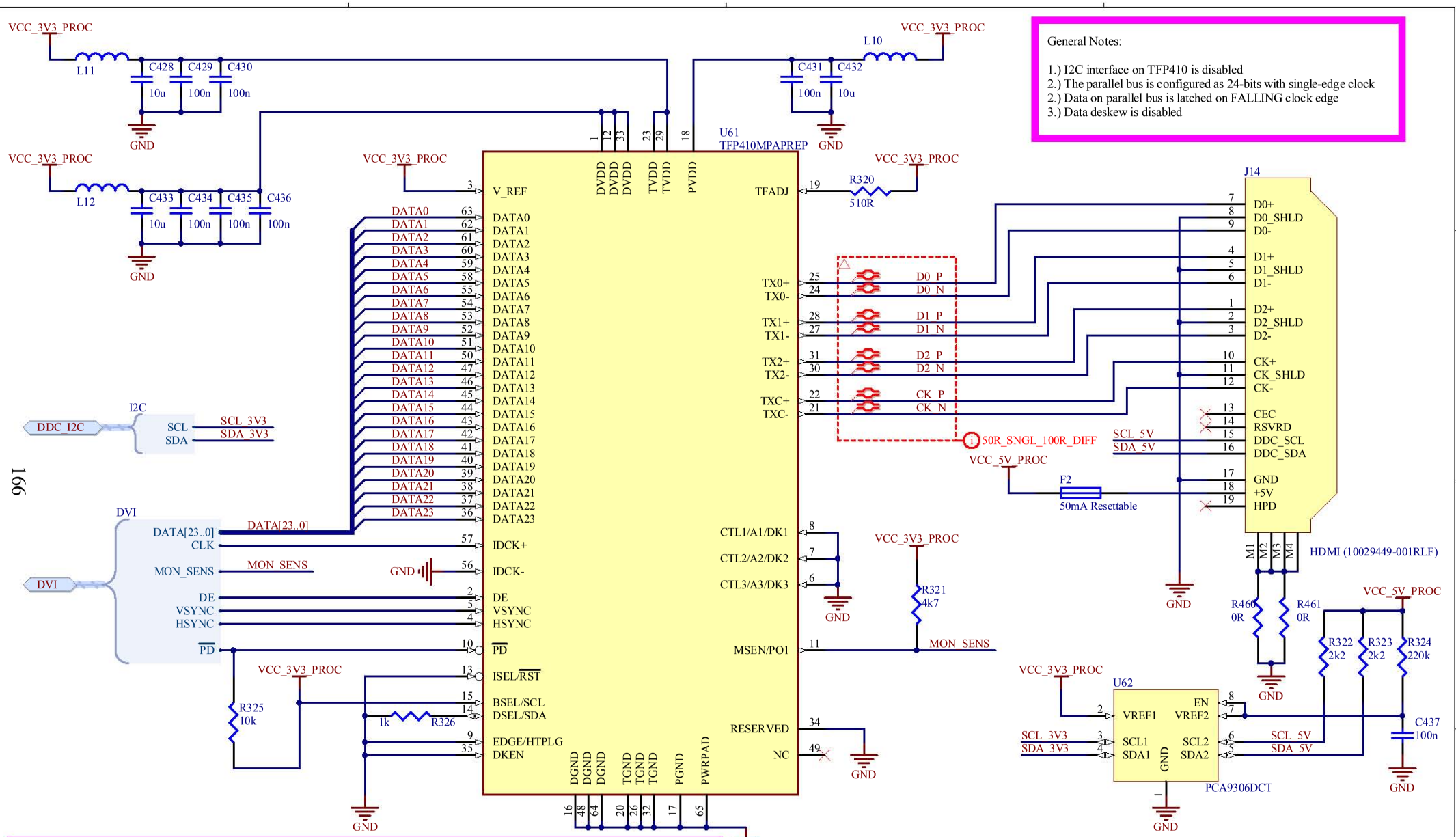
Title: USB Host Transceivers			
Size: A3	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 46 of 65	
File: C:\Users\...\usb_host.SchDoc		Drawn By: Simon Scott	

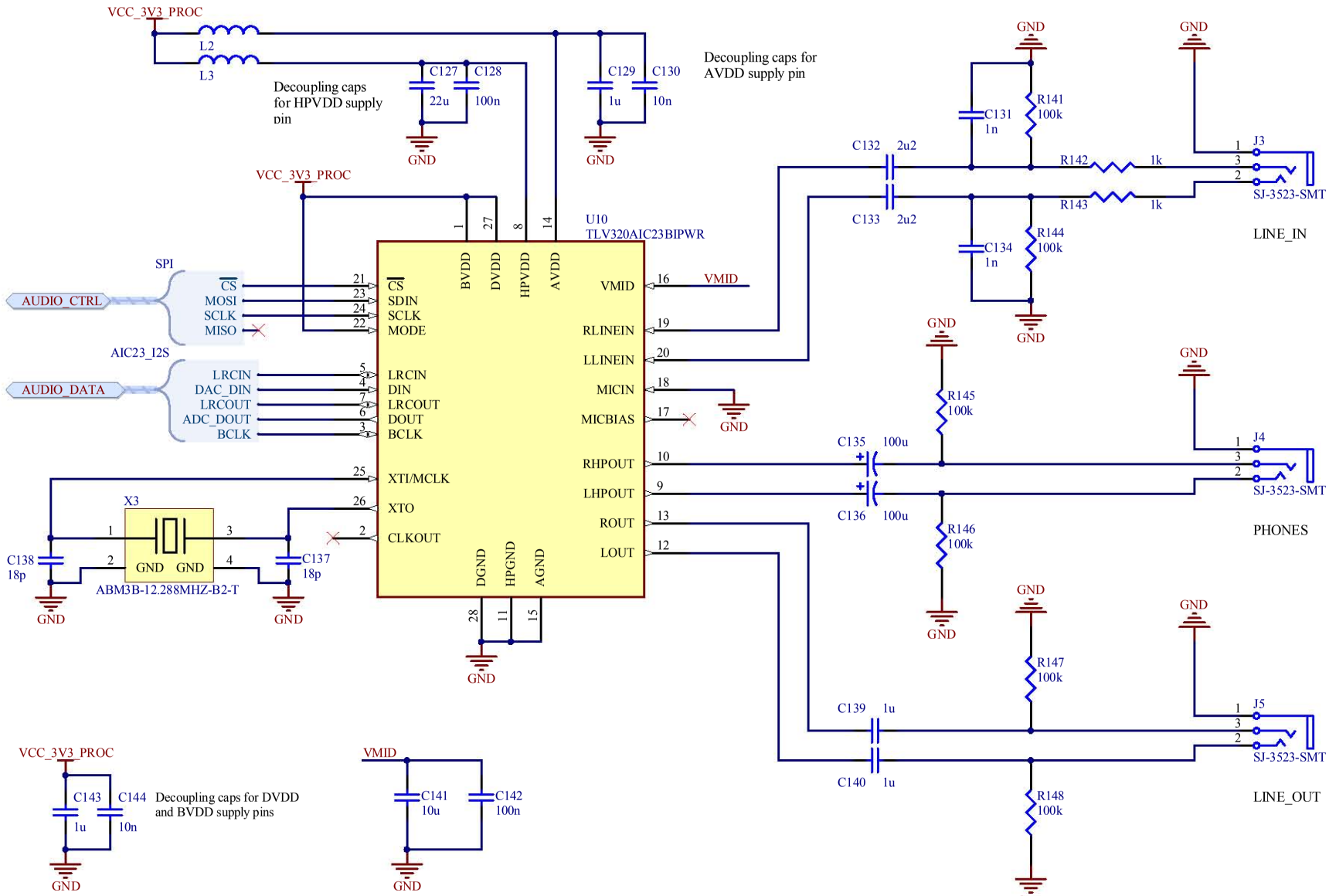


Layout Notes:

1.) Place 100nF and 22uF decoupling caps as close to pin 4 of SD card connector as possible
2.) Place 100nF decoupling capacitor as close to pin 10 of DS1390 IC as possible
3.) Trace lengths between DS1390 and the crystal should be minimised, and should be as straight as possible

Title	SD Card and Real-time Clock		
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 47 of 65	
File: C:\Users\..\sd_card_and_rtc.SchDoc		Drawn By: Simon Scott	


General Notes:
1.) I2C interface on TFP410 is disabled
2.) The parallel bus is configured as 24-bits with single-edge clock
2.) Data on parallel bus is latched on FALLING clock edge
3.) Data deskew is disabled
U61
TFP410MPAPREP
V_REF
TFADJ
R320
510R
DATA0
DATA1
DATA2
DATA3
DATA4
DATA5
DATA6
DATA7
DATA8
DATA9
DATA10
DATA11
DATA12
DATA13
DATA14
DATA15
DATA16
DATA17
DATA18
DATA19
DATA20
DATA21
DATA22
DATA23
TX0+
TX0-
TX1+
TX1-
TX2+
TX2-
TXC+
TXC-
D0_P
D0_N
D1_P
D1_N
D2_P
D2_N
CK_P
CK_N
50R_SNGL_100R_DIFF
J14
D0+
D0_SHLD
D0-
D1+
D1_SHLD
D1-
D2+
D2_SHLD
D2-
CK+
CK_SHLD
CK-
CEC
RSVRD
DDC_SCL
DDC_SDA
GND
+5V
HPD
HDMI (10029449-001RLF)
SCL_5V
SDA_5V
VCC_5V_PROC
F2
50mA Resettable
I2C
DDC_I2C
SCL
SDA
SCL_3V3
SDA_3V3
DVI
DATA[23..0]
CLK
MON_SENS
DE
VSYNC
HSYNC
PD
IDCK+
IDCK-
CTL1/A1/DK1
CTL2/A2/DK2
CTL3/A3/DK3
MSEN/PO1
ISEL/RST
BSEL/SCL
DSEL/SDA
EDGE/HTPLG
DKEN
RESERVED
NC
R321
4k7
R325
10k
R326
1k
R460
0R
R461
0R
R322
2k2
R323
2k2
R324
220k
U62
VREF1
VREF2
EN
SCL1
SDA1
SCL2
SDA2
PCA9306DCT
C437
100n
L10
L11
L12
C428
C429
C430
C431
C432
C433
C434
C435
C436
10u
100n
VCC_3V3_PROC
GND
Layout Notes:
1.) The data inputs to the TFP410 IC (DATAx) must have equal lengths. Trace separation should be approx 5x height
2.) The data pairs to the HDMI connector (D0+, D0-, etc) must be routed as diff pairs, with the lengths matched (length < 50mm)
3.) These data differential pairs must have single-ended impedance of 50 ohms and differential impedance of 100 ohms
4.) Trace separation for the diff pairs should be 2x to 4x height
5.) Place each 100nF supply decoupling cap directly at each supply pin
6.) The PowerPad on the TFP410 must be connected to ground with vias (as per footprint)
Title
HDMI Video Transmitter
Size
A4
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.0
Date:
29/03/2011
Sheet 48 of 65
File:
C:\Users\..\video.SchDoc
Drawn By:
Simon Scott



General Notes:

1.) The control interface is configured for SPI mode
2.) The sampling rate is set to multiple of 8kHz, and therefore cannot do 44.1kHz

Layout Notes:

1.) Place label for each connector (LINE_IN, etc) on the sikscreen, next to each connector
2.) Place the supply decoupling caps as close to the supply pins as possible

Title: Audio			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 49 of 65	
File: C:\Users\..\audio.SchDoc		Drawn By: Simon Scott	

Layout Notes:

1.) D_P and D_N must be routed as differential pair, with 90 ohms differential impedance, and equal lengths
2.) The 4 traffic LEDs must be placed in a line, with their silkscreen labels next to them

General Notes:

1.) The FT4232 USB IC is the JTAG master in the chain
2.) The current through the traffic LEDs is 3.9mA

I2C_EN

VCC_3V3_PROC L8 L9 C410 4u7 C411 4u7 C412 100n C413 100n GND

VCC_1V8_USBREG VCC_3V3_PROC

U57 FT4232HQ

VREGIN VREGOUT VPHY VPLL VCORE VCORE VCORE VCCIO VCCIO VCCIO VCCIO

ADBUS0 TCK
ADBUS1 TDO_UNBUF
ADBUS2 TDI_BUF
ADBUS3 TMS
ADBUS4 TRST
ADBUS5 JTAG_EN
ADBUS6 FPGA_VCCINT_EN
ADBUS7 FPGA_VCCO_AUX_EN

BDBUS0 SCL
BDBUS1 SDA
BDBUS2
BDBUS3
BDBUS4
BDBUS5 I2C_EN
BDBUS6
BDBUS7

CDBUS0 U0_TX
CDBUS1 U0_RX
CDBUS2
CDBUS3
CDBUS4
CDBUS5
CDBUS6
CDBUS7

DDBUS0 U1_TX
DDBUS1 U1_RX
DDBUS2
DDBUS3
DDBUS4
DDBUS5
DDBUS6
DDBUS7

PWREN SUSPEND

DM DP RESET REF EECS EECLK EEDATA OSCI OSCO TEST AGND

C414 4u7

J13 USB Type-B (292304-1) VBUS D- D+ GND M2 M1 C415 100n

90R_DIFF D_N D_P R304 4k7 R305 1k R306 1k R307 10k R308 12k

RS232 TX RX RTS CTS UART0

RS232 TX RX RTS CTS UART1

EEPROM_CS EEPROM_CLK EEPROM_DAT

X5 ABM3B-12.000MHZ-10-1-U-T C417 10p C418 10p

VCC_3V3_PROC C409 100n R299 2k2

U58 SN74LVC244ADB

JTAG_EN TCK TDI TDO_UNBUF TMS TRST TDI_BUF

R300 10k R301 10k R302 10k R303 10k

JTAG_BUS TCK TDI TDO TMS TRST JTAG

VCC_3V3_PROC C416 100n R309 4k7 R310 4k7

U59 PCA9515APWR SCL0 SDA0 SCL1 SDA1 EN NC R311 10k

I2C SCL SDA I2C

LED Silkscreen Labels
(left to right):
DGB0_RX
DBG0_TX
DGB1_RX
DBG1_TX

VCC_3V3_PROC D39 ORNG D40 ORNG D41 ORNG D42 ORNG

R315 330R R316 330R R317 330R R318 330R

U0_TX U0_RX U1_TX U1_RX

FPGA_VCCINT_EN FPGA_VCCO_AUX_EN FTDI_SUPPLY_EN_BUS FPGA_SUPPLY_EN

VCC_1V8_USBREG C419 100n C420 100n C421 100n GND

Decoupling Caps for FT4232HQ:
Place one cap at each VCORE pin

VCC_3V3_PROC C423 100n C424 100n C425 100n C426 100n C427 100n GND

Decoupling Caps for FT4232HQ:
Place one cap at each VCCIO and VREGIN pin

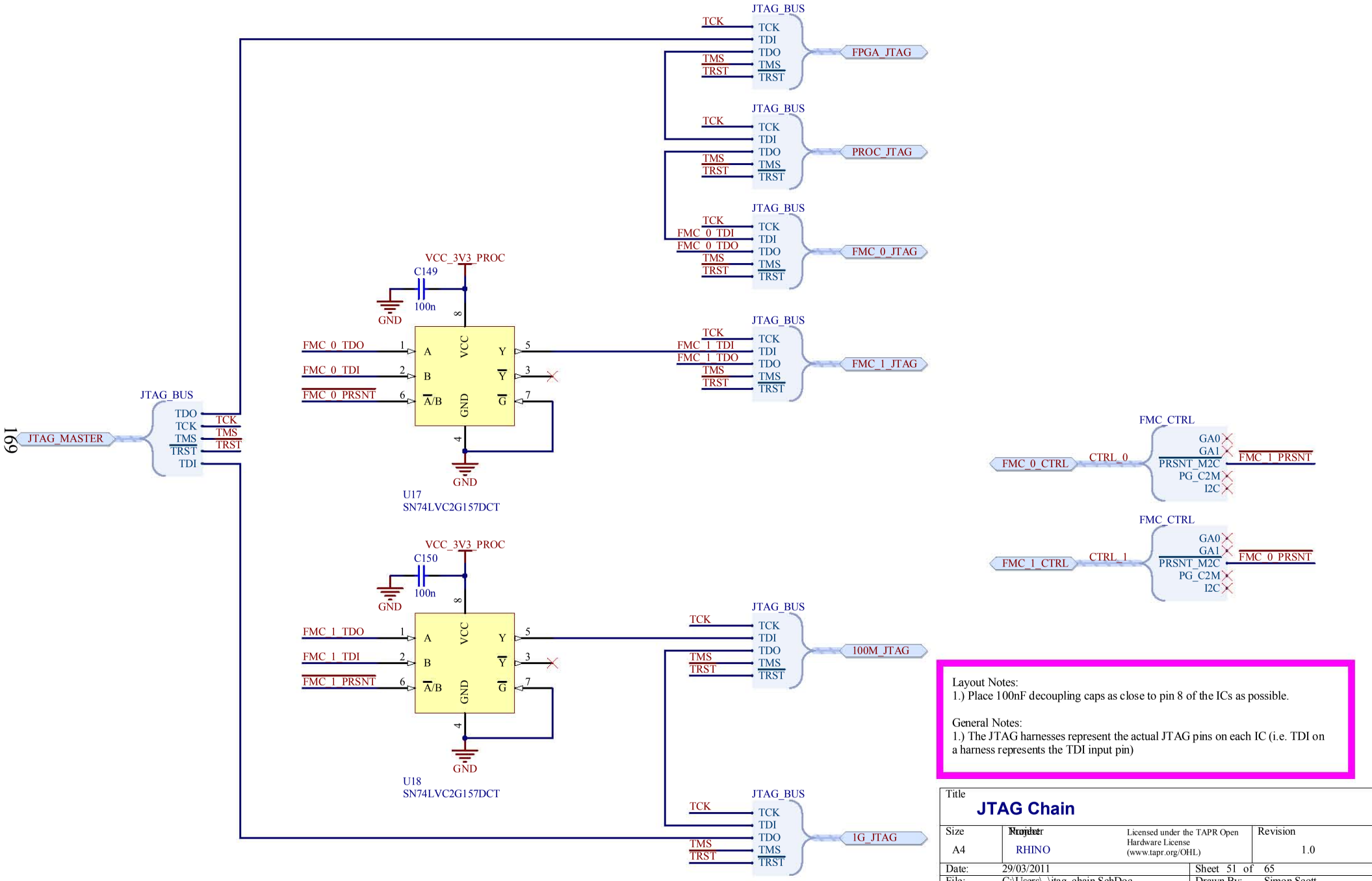
VCC_3V3_PROC C422 100n R312 10k R313 10k R314 10k

U60 93LC56BT-I/OT CS CLK DI DO VCC VSS R319 2k2

EEPROM_CS EEPROM_CLK EEPROM_DAT

Title	USB to JTAG/I2C/RS-232		
Size: A3	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.1
Date: 29/03/2011		Sheet 50 of 65	
File: C:\Users\...\usb_to_jtag_rs232.SchDoc		Drawn By: Simon Scott	

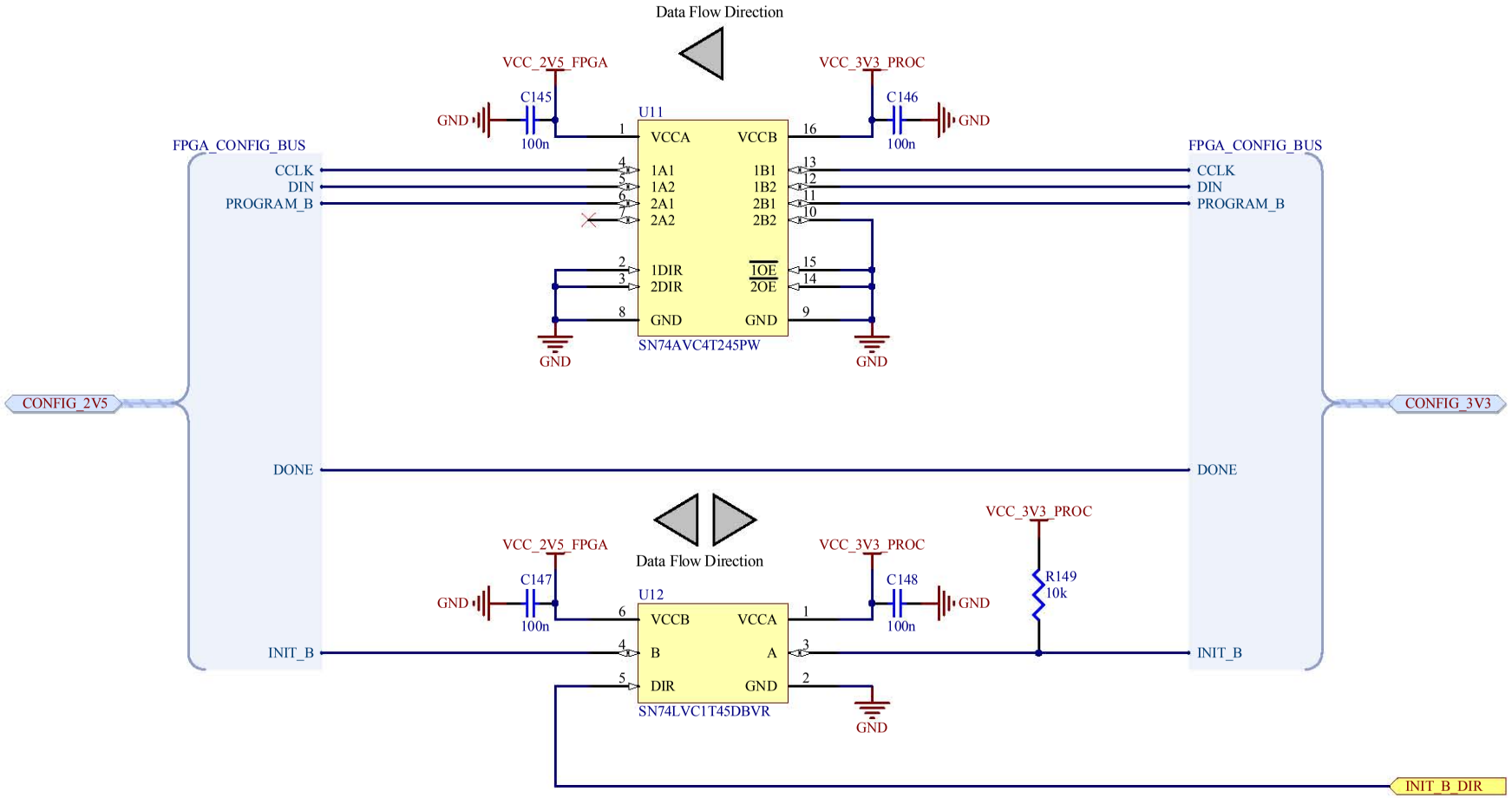
168



Layout Notes:
1.) Place 100nF decoupling caps as close to pin 8 of the ICs as possible.

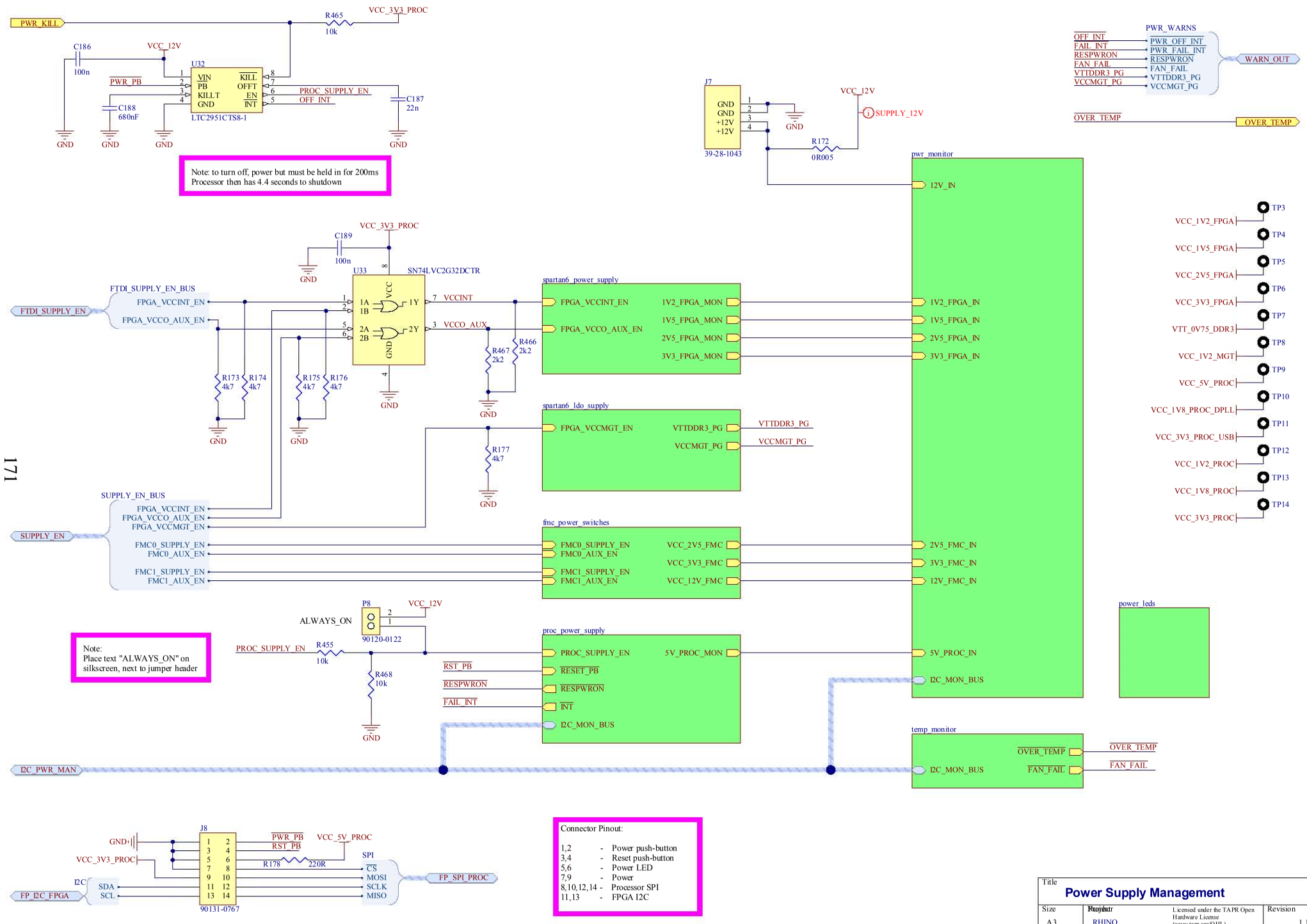
General Notes:
1.) The JTAG harnesses represent the actual JTAG pins on each IC (i.e. TDI on a harness represents the TDI input pin)

Title: **JTAG Chain**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 51 of 65	
File: C:\Users\..\jtag_chain.SchDoc		Drawn By: Simon Scott	

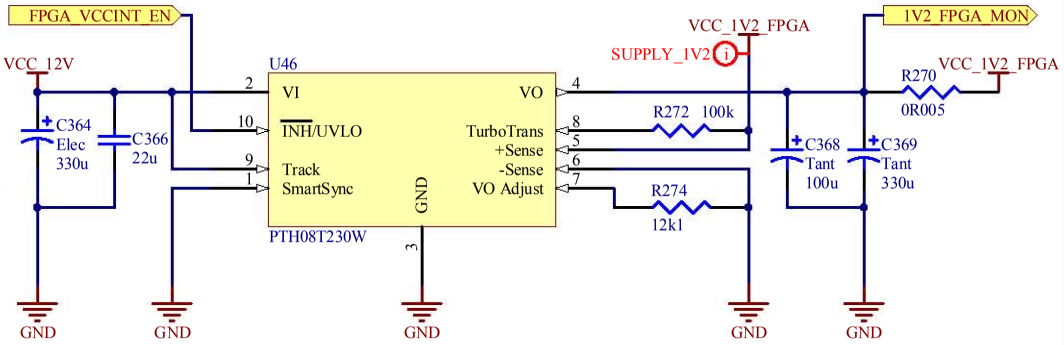


170

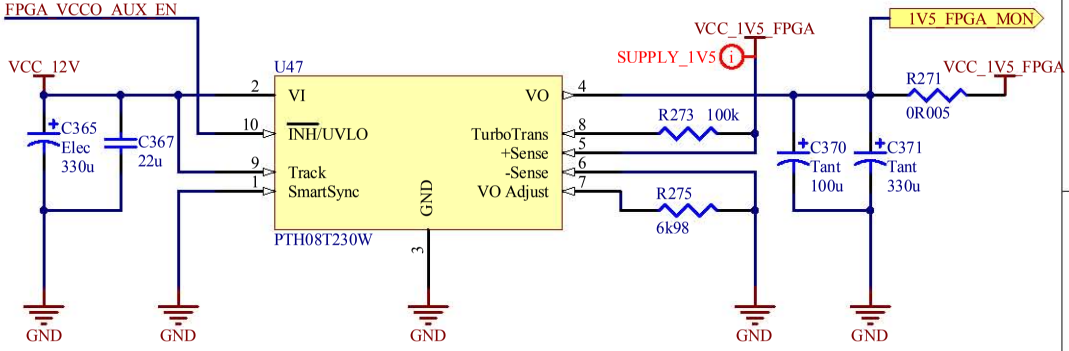
Title	Configuration Interface Level Translator
Size: A4	Number: RHINO
Revision: 1.0	
Date: 29/03/2011	Sheet 52 of 65
File: C:\Users\..\config_lvl_xlator.SchDoc	Drawn By: Simon Scott


PWR_KILL
R465
10k
VCC_3V3_PROC
C186
100n
VCC_12V
U32
VIN
PB
KILLT
GND
KILL
OFFT
EN
INT
LTC2951CTS8-1
PWR_PB
C188
680nF
PROC_SUPPLY_EN
OFF_INT
C187
22n
GND
Note: to turn off, power but must be held in for 200ms
Processor then has 4.4 seconds to shutdown
J7
GND
GND
+12V
+12V
39-28-1043
VCC_12V
SUPPLY_12V
R172
0R005
PWR_WARNS
OFF_INT
FAIL_INT
RESPWRON
FAN_FAIL
VTTDDR3_PG
VCCMGT_PG
PWR_OFF_INT
PWR_FAIL_INT
RESPWRON
FAN_FAIL
VTTDDR3_PG
VCCMGT_PG
WARN_OUT
OVER_TEMP
OVER_TEMP
pwr_monitor
12V_IN
1V2_FPGA_IN
1V5_FPGA_IN
2V5_FPGA_IN
3V3_FPGA_IN
2V5_FMC_IN
3V3_FMC_IN
12V_FMC_IN
5V_PROC_IN
I2C_MON_BUS
VCC_3V3_PROC
C189
100n
U33
SN74LVC2G32DCTR
FTDI_SUPPLY_EN_BUS
FTDI_SUPPLY_EN
FPGA_VCCINT_EN
FPGA_VCCO_AUX_EN
1A
1B
2A
2B
1Y
2Y
VCC
GND
VCCINT
VCCO_AUX
R173
4k7
R174
4k7
R175
4k7
R176
4k7
R467
2k2
R466
2k2
spartan6_power_supply
FPGA_VCCINT_EN
FPGA_VCCO_AUX_EN
1V2_FPGA_MON
1V5_FPGA_MON
2V5_FPGA_MON
3V3_FPGA_MON
spartan6_ldo_supply
FPGA_VCCMGT_EN
VTTDDR3_PG
VCCMGT_PG
VTTDDR3_PG
VCCMGT_PG
R177
4k7
SUPPLY_EN_BUS
SUPPLY_EN
FPGA_VCCINT_EN
FPGA_VCCO_AUX_EN
FPGA_VCCMGT_EN
FMC0_SUPPLY_EN
FMC0_AUX_EN
FMC1_SUPPLY_EN
FMC1_AUX_EN
fmc_power_switches
FMC0_SUPPLY_EN
FMC0_AUX_EN
FMC1_SUPPLY_EN
FMC1_AUX_EN
VCC_2V5_FMC
VCC_3V3_FMC
VCC_12V_FMC
P8
90120-0122
ALWAYS_ON
VCC_12V
Note:
Place text "ALWAYS_ON" on
silkscreen, next to jumper header
PROC_SUPPLY_EN
R455
10k
R468
10k
proc_power_supply
PROC_SUPPLY_EN
RESET_PB
RESPWRON
INT
I2C_MON_BUS
5V_PROC_MON
RST_PB
RESPWRON
FAIL_INT
power_leds
temp_monitor
I2C_MON_BUS
OVER_TEMP
FAN_FAIL
OVER_TEMP
FAN_FAIL
I2C_PWR_MAN
TP3
VCC_1V2_FPGA
TP4
VCC_1V5_FPGA
TP5
VCC_2V5_FPGA
TP6
VCC_3V3_FPGA
TP7
VTT_0V75_DDR3
TP8
VCC_1V2_MGT
TP9
VCC_5V_PROC
TP10
VCC_1V8_PROC_DPLL
TP11
VCC_3V3_PROC_USB
TP12
VCC_1V2_PROC
TP13
VCC_1V8_PROC
TP14
VCC_3V3_PROC
J8
GND
VCC_3V3_PROC
I2C
SDA
SCL
FP_I2C_FPGA
90131-0767
PWR_PB
RST_PB
VCC_5V_PROC
R178
220R
SPI
CS
MOSI
SCLK
MISO
FP_SPI_PROC
Connector Pinout:
1,2 - Power push-button
3,4 - Reset push-button
5,6 - Power LED
7,9 - Power
8,10,12,14 - Processor SPI
11,13 - FPGA I2C
171
Title
Power Supply Management
Size
A3
Project
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.1
Date:
29/03/2011
Sheet 53 of 65
File:
C:\Users\...\pwr_supply_management.SchDoc
Drawn By:
Simon Scott

1V2 6A FPGA VCCINT Supply

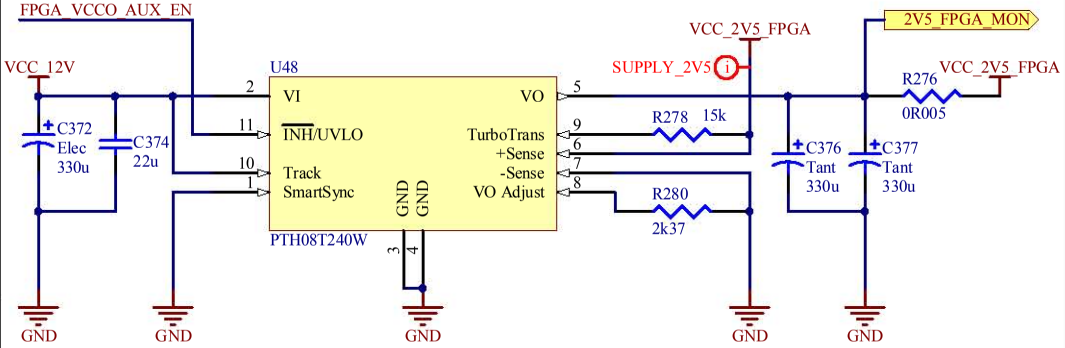


1V5 6A FPGA + DRAM Supply

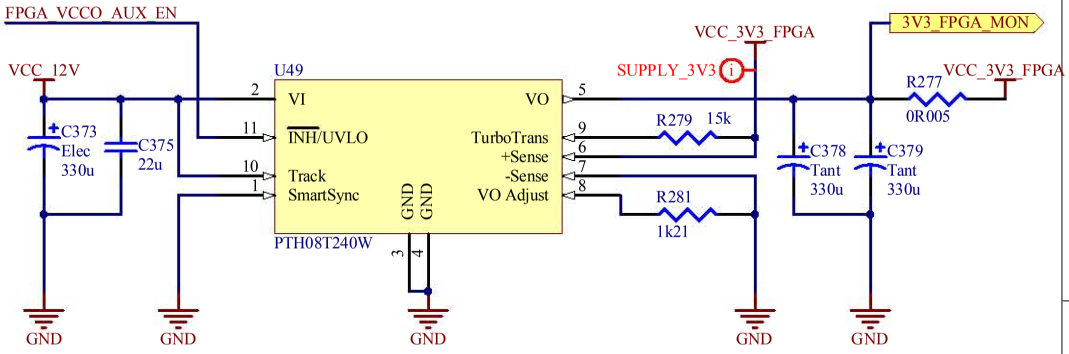


172

2V5 10A FPGA + FMC Supply



3V3 10A FPGA + FMC Supply



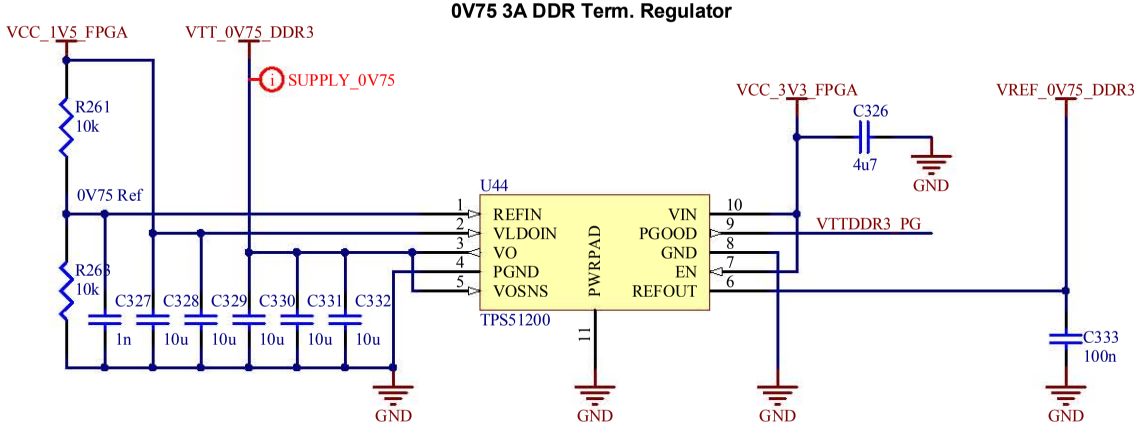
Layout Notes for Supplies:

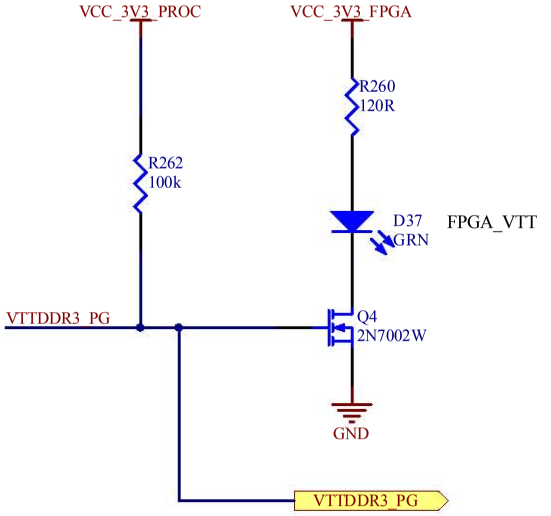
1.) 22uF input caps must be placed within 1.3cm of pin 2
2.) Resistor between "-Sense" and "VO Adjust" must be placed directly at the pins

FPGA_VCCO_AUX_EN FPGA_VCCO_AUX_EN

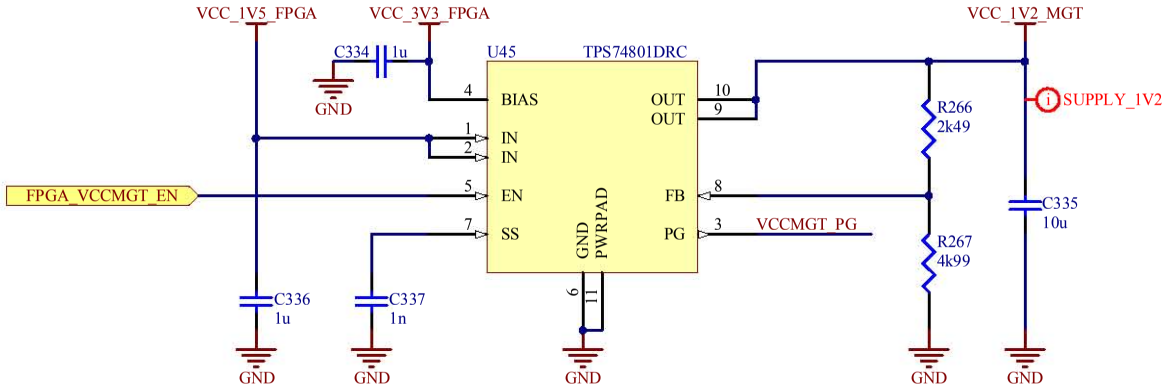
Title: **Spartan-6 Power Supplies**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 54 of 65	
File: C:\Users\..\spartan6_power_supply.SchDoc		Drawn By: Simon Scott	

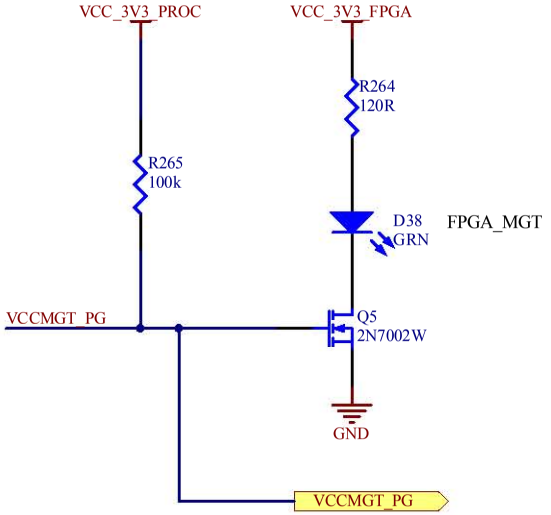
0V75 3A DDR Term. Regulator





1V2 1.5A FPGA MGT Regulator

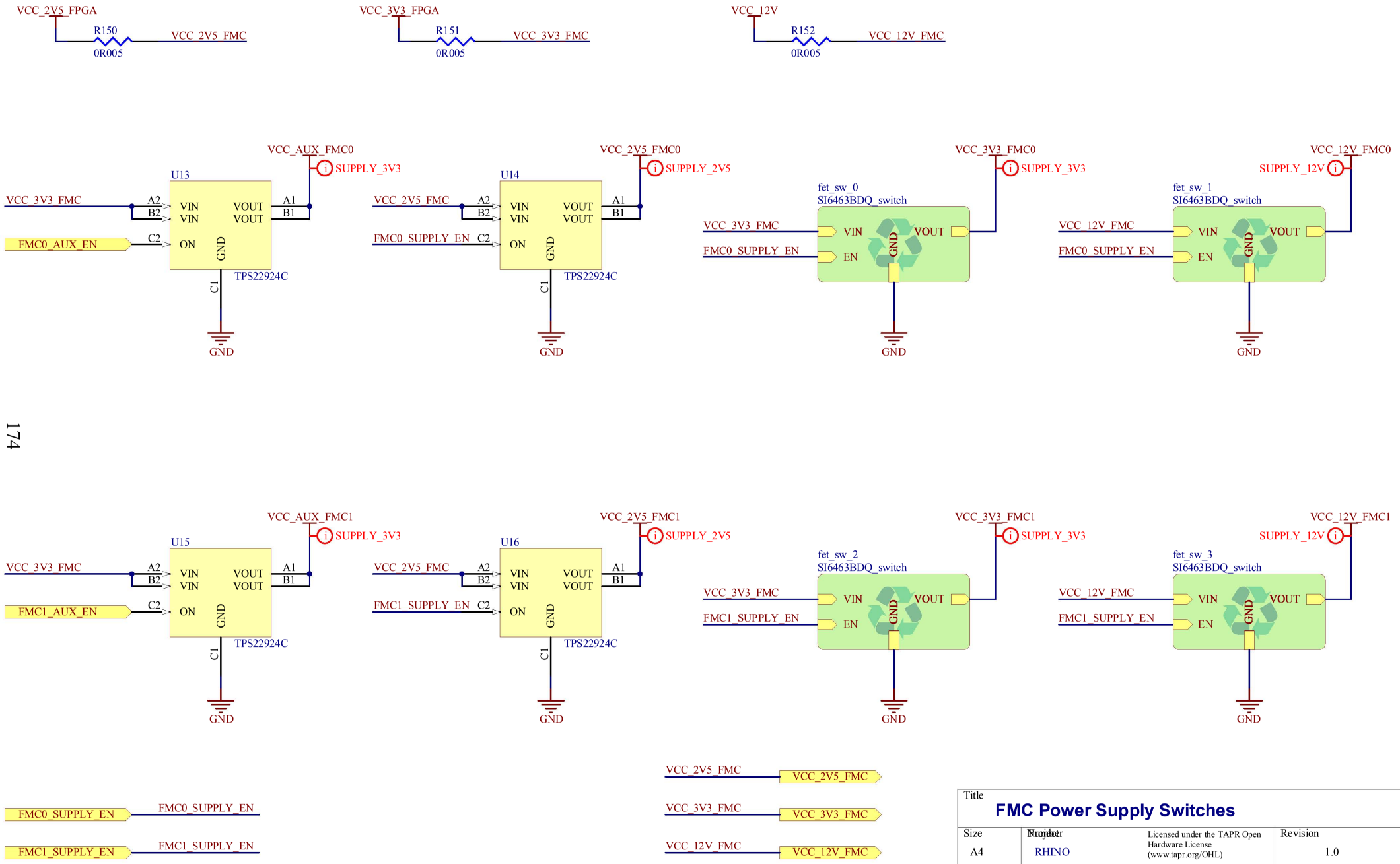




Layout Notes for Supplies:

1.) For TPS51200, place caps at VLDOIN and VO close to the pins, via short+wide traces
2.) Connect the PowerPad of both ICs directly to the ground plane with vias, to help heat dissipation
3.) Place the name of each LED (e.g. FPGA_VTT) on the silkscreen, next to the LED

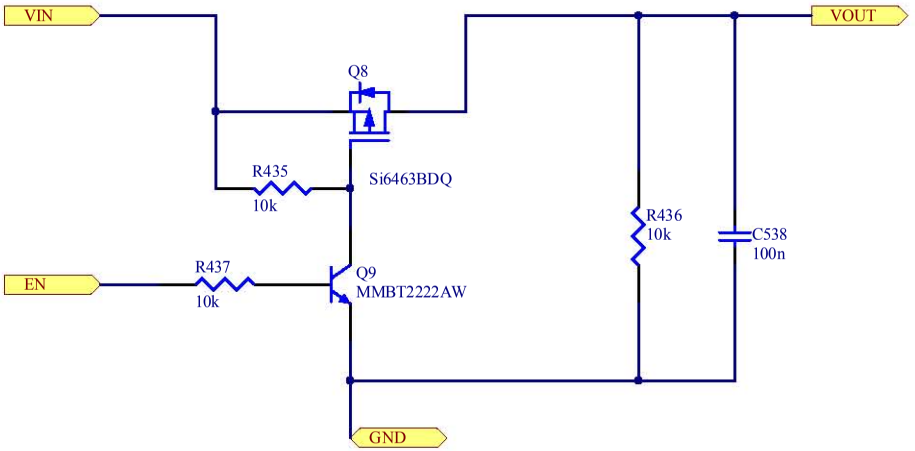
Title: **Spartan-6 LDO Power Supplies**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 55 of 65	
File: C:\Users\..\spartan6_ldo_supply.SchDoc		Drawn By: Simon Scott	



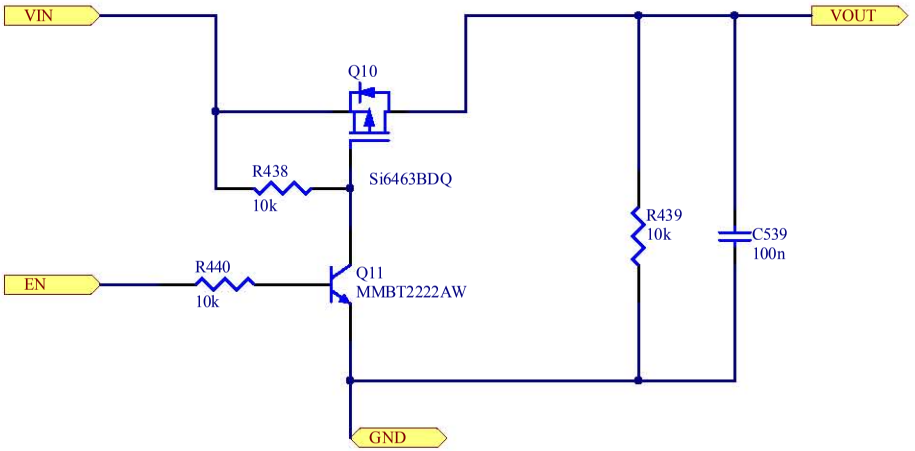
Title
FMC Power Supply Switches

Size	Number	Revision
A4	RHINO	1.0

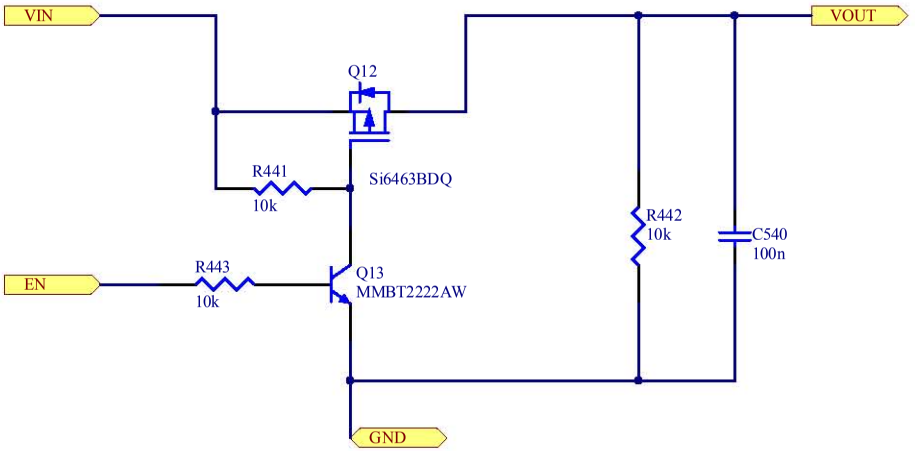
Date:	29/03/2011	Sheet 56 of 65	
File:	C:\Users\..\fmc_power_switches.SchDoc	Drawn By:	Simon Scott



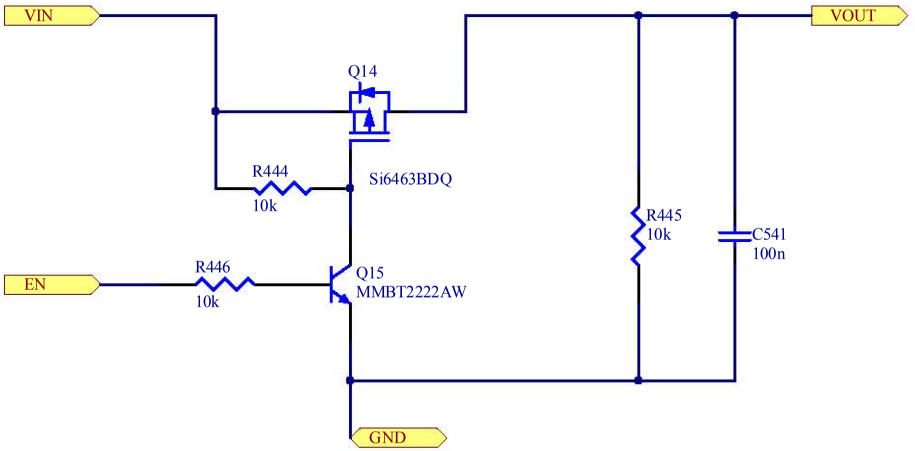
Title: **Si6463BDQ FET Load Switch**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 57 of 65	
File: C:\Users\..\SI6463BDQ_switch.SchDoc		Drawn By: Simon Scott	



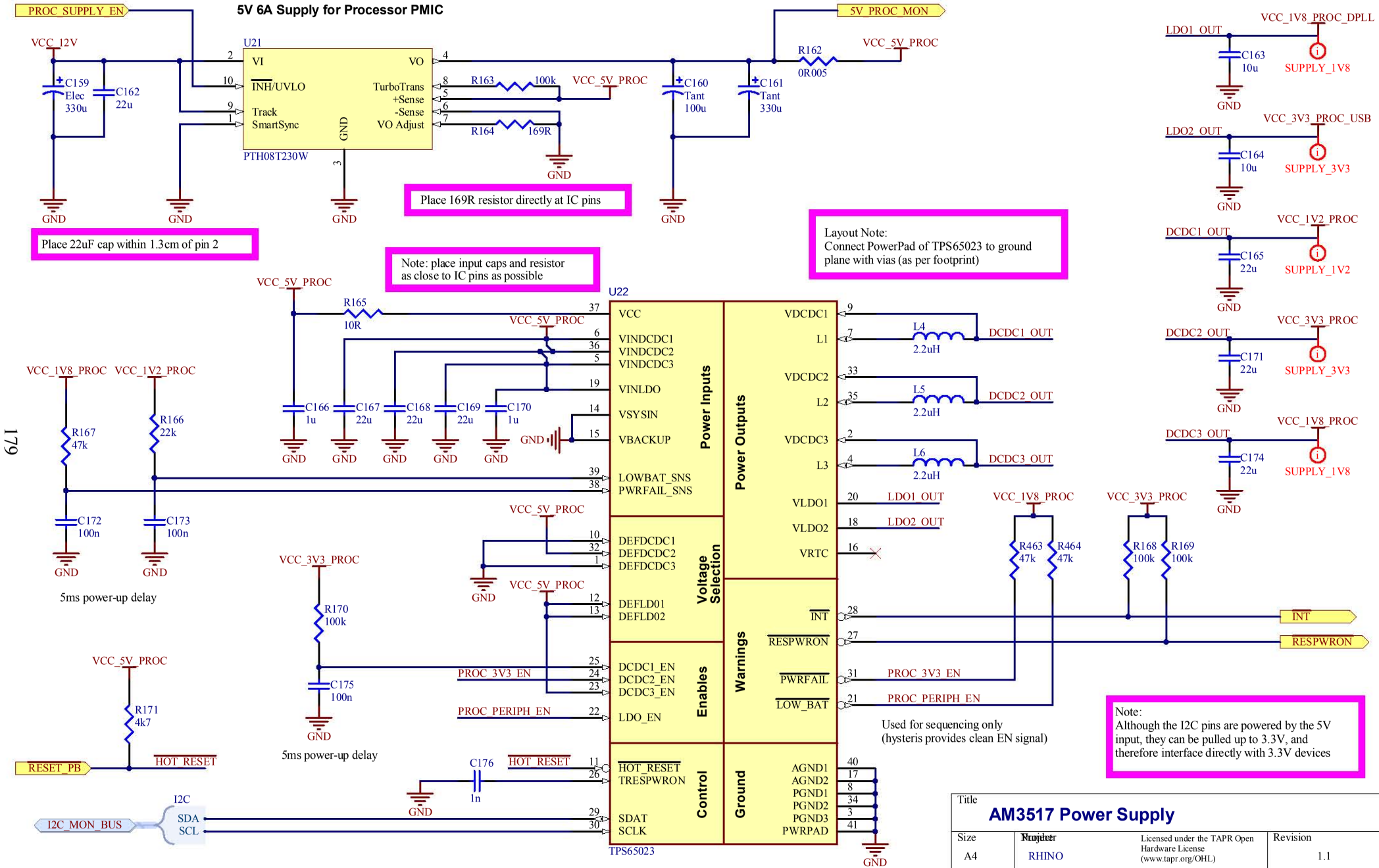
Title: **Si6463BDQ FET Load Switch**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 58 of 65	
File: C:\Users\..\SI6463BDQ_switch.SchDoc		Drawn By: Simon Scott	

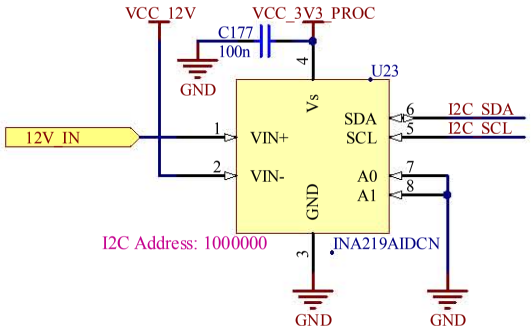


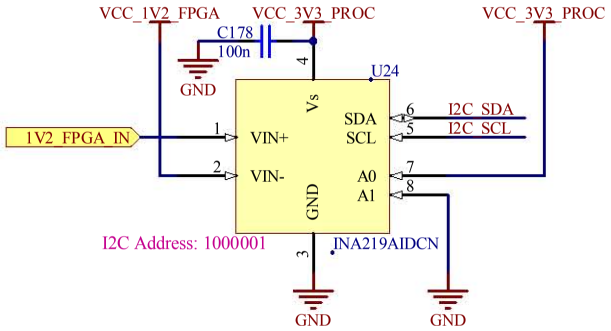
Title: **Si6463BDQ FET Load Switch**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 59 of 65	
File: C:\Users\..\SI6463BDQ_switch.SchDoc		Drawn By: Simon Scott	

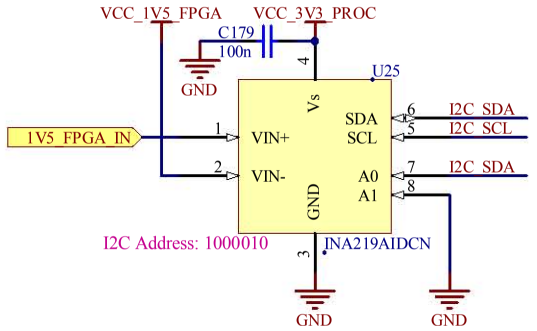


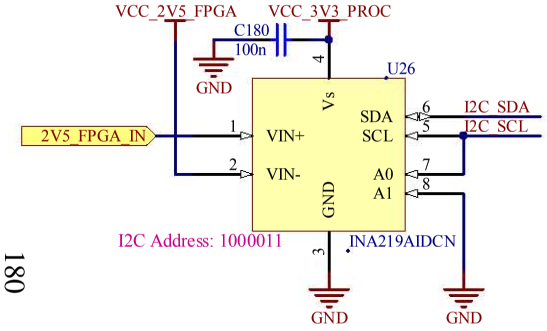
Title: **Si6463BDQ FET Load Switch**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 60 of 65	
File: C:\Users\..\SI6463BDQ_switch.SchDoc		Drawn By: Simon Scott	

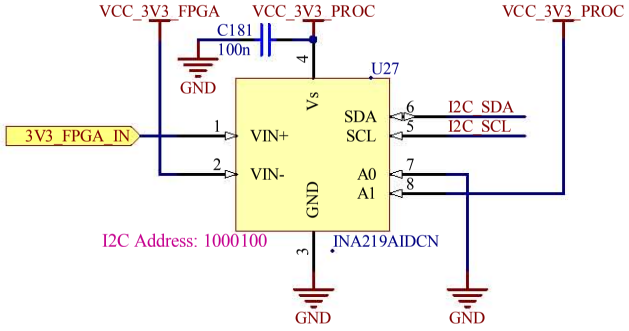

5V 6A Supply for Processor PMIC
PROC_SUPPLY_EN
VCC_12V
U21
VI
INH/UVLO
Track
SmartSync
GND
VO
TurboTrans
+Sense
-Sense
VO Adjust
PTH08T230W
C159 Elec 330u
C162 22u
R163 100k
R164 169R
R162 0R005
5V_PROC_MON
VCC_5V_PROC
C160 Tant 100u
C161 Tant 330u
Place 169R resistor directly at IC pins
Place 22uF cap within 1.3cm of pin 2
Note: place input caps and resistor as close to IC pins as possible
Layout Note:
Connect PowerPad of TPS65023 to ground plane with vias (as per footprint)
U22
VCC
VINDCDC1
VINDCDC2
VINDCDC3
VINLDO
VSYSIN
VBACKUP
LOWBAT_SNS
PWRFAIL_SNS
Power Inputs
DEFDCDC1
DEFDCDC2
DEFDCDC3
DEFLD01
DEFLD02
Voltage Selection
DCDC1_EN
DCDC2_EN
DCDC3_EN
LDO_EN
Enables
HOT_RESET
TRESPWRON
Control
SDAT
SCLK
VDCDC1
L1
VDCDC2
L2
VDCDC3
L3
VLDO1
VLDO2
VRTC
Power Outputs
INT
RESPWRON
PWRFAIL
LOW_BAT
Warnings
AGND1
AGND2
PGND1
PGND2
PGND3
PWRPAD
Ground
TPS65023
R165 10R
C166 1u
C167 22u
C168 22u
C169 22u
C170 1u
L4 2.2uH
L5 2.2uH
L6 2.2uH
DCDC1_OUT
DCDC2_OUT
DCDC3_OUT
LDO1_OUT
LDO2_OUT
R167 47k
R166 22k
C172 100n
C173 100n
5ms power-up delay
R170 100k
C175 100n
5ms power-up delay
R171 4k7
RESET_PB
HOT_RESET
C176 1n
I2C
I2C_MON_BUS
SDA
SCL
PROC_3V3_EN
PROC_PERIPH_EN
R463 47k
R464 47k
R168 100k
R169 100k
Used for sequencing only
(hysteris provides clean EN signal)
Note:
Although the I2C pins are powered by the 5V input, they can be pulled up to 3.3V, and therefore interface directly with 3.3V devices
VCC_1V8_PROC_DPLL
LDO1_OUT
C163 10u
SUPPLY_1V8
VCC_3V3_PROC_USB
LDO2_OUT
C164 10u
SUPPLY_3V3
VCC_1V2_PROC
DCDC1_OUT
C165 22u
SUPPLY_1V2
VCC_3V3_PROC
DCDC2_OUT
C171 22u
SUPPLY_3V3
VCC_1V8_PROC
DCDC3_OUT
C174 22u
SUPPLY_1V8
Title
AM3517 Power Supply
Size
A4
Number
RHINO
Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)
Revision
1.1
Date:
29/03/2011
Sheet 61 of 65
File:
C:\Users\..\proc_power_supply.SchDoc
Drawn By:
Simon Scott

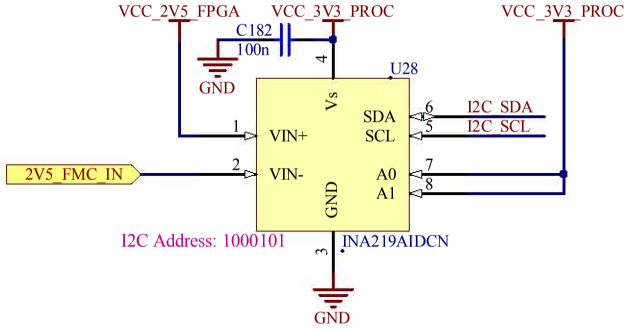








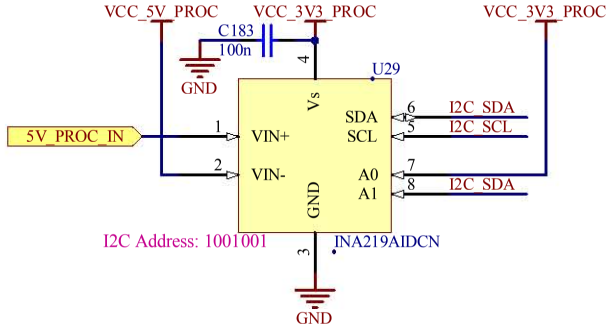


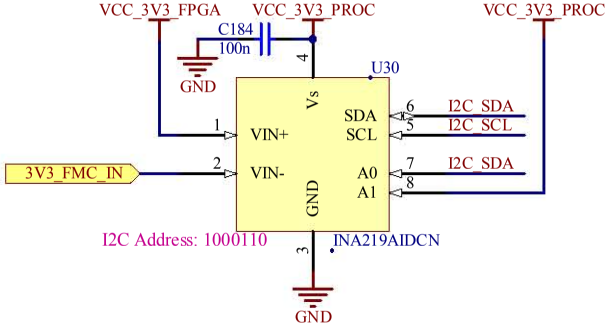


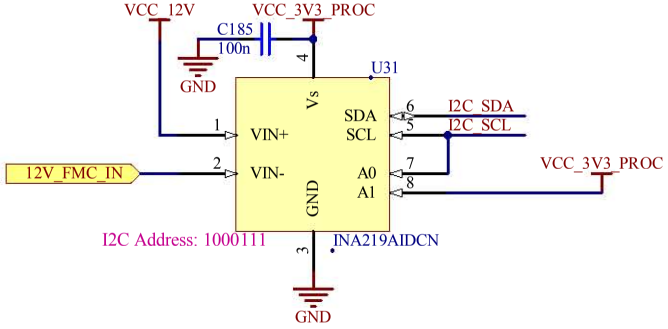
180

Layout Notes:

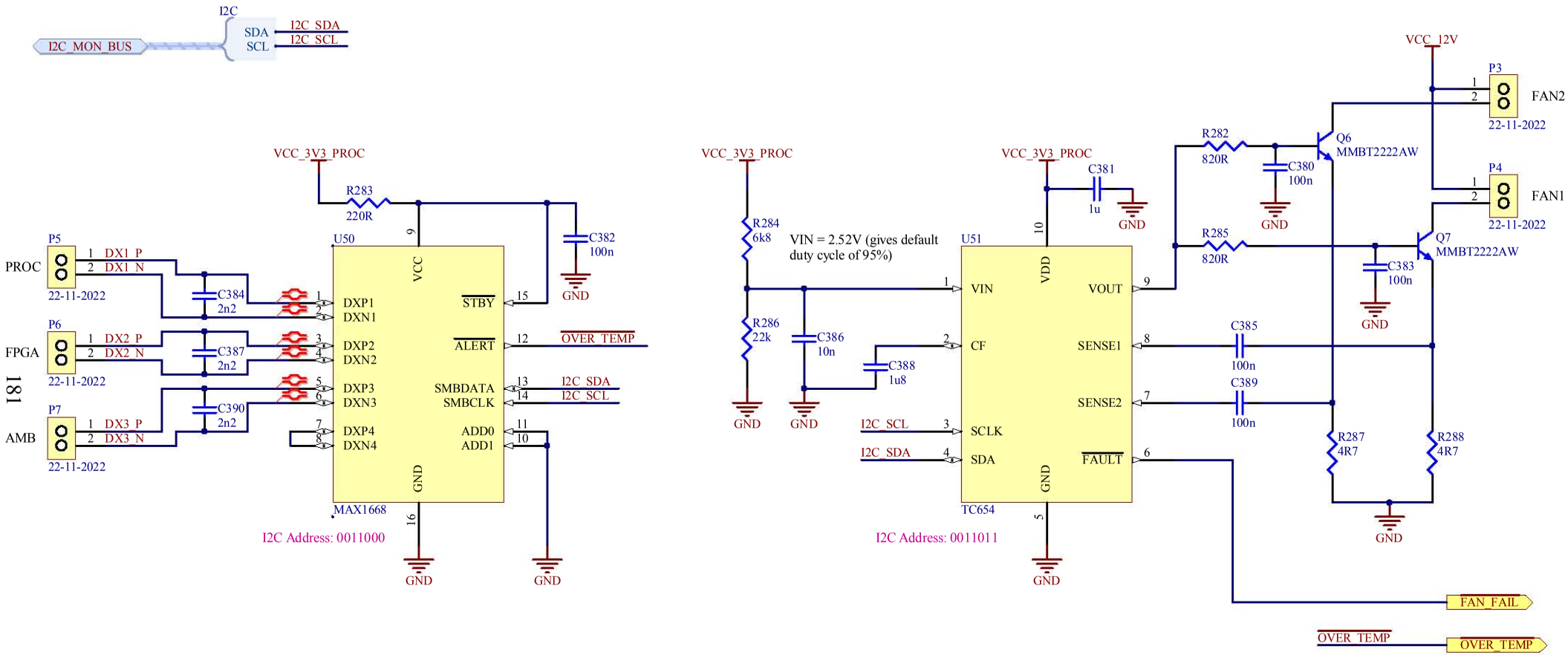
1.) Place each INA219 device as close to its current-sensing resistor (5mR 1W) as possible
2.) Place the 100nF decoupling capacitor as close to the IC pin as possible







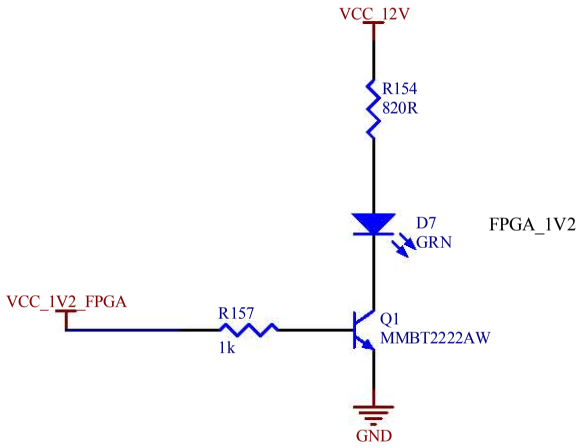
Title: **Power Monitor**			
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 62 of 65	
File: C:\Users\..\pwr_monitor.SchDoc		Drawn By: Simon Scott	

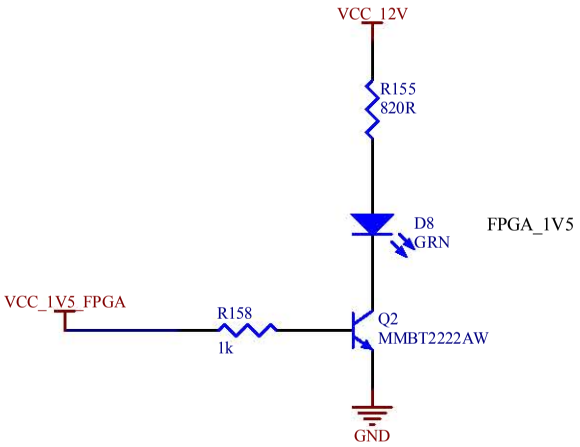


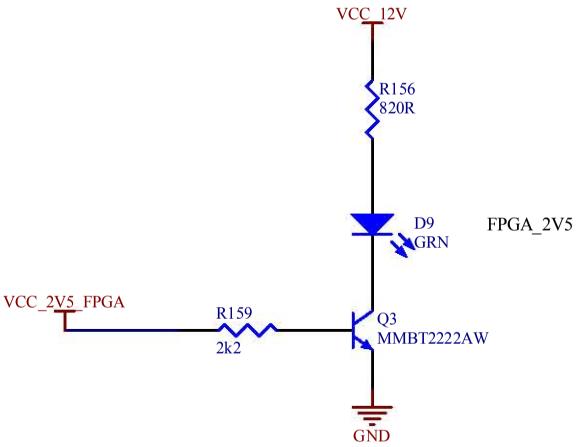
Routing notes for MAX1668:

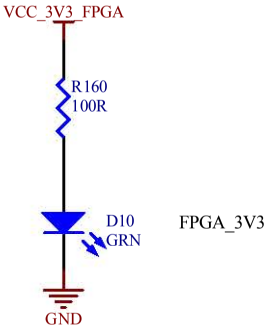
1.) Place each 2-pin connector within 10cm of MAX1668
2.) Also place the FPGA and PROC 2-pin connectors within 10cm of the respective devices
3.) Place the AMB 2-pin connector near a board edge
4.) Route each DXN and DXP pair parallel to each other, with guard GND traces on either side
5.) Keep DXN and DXP pairs away from high-speed busses

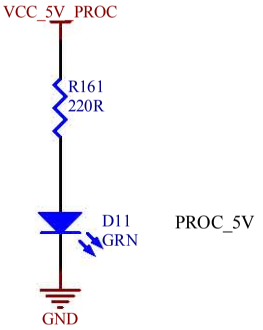
Title	**Temperature Monitor and Fan Controller**		
Size: A4	Project: RHINO	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision: 1.0
Date: 29/03/2011		Sheet 63 of 65	
File: C:\Users\..\temp_monitor.SchDoc		Drawn By: Simon Scott	









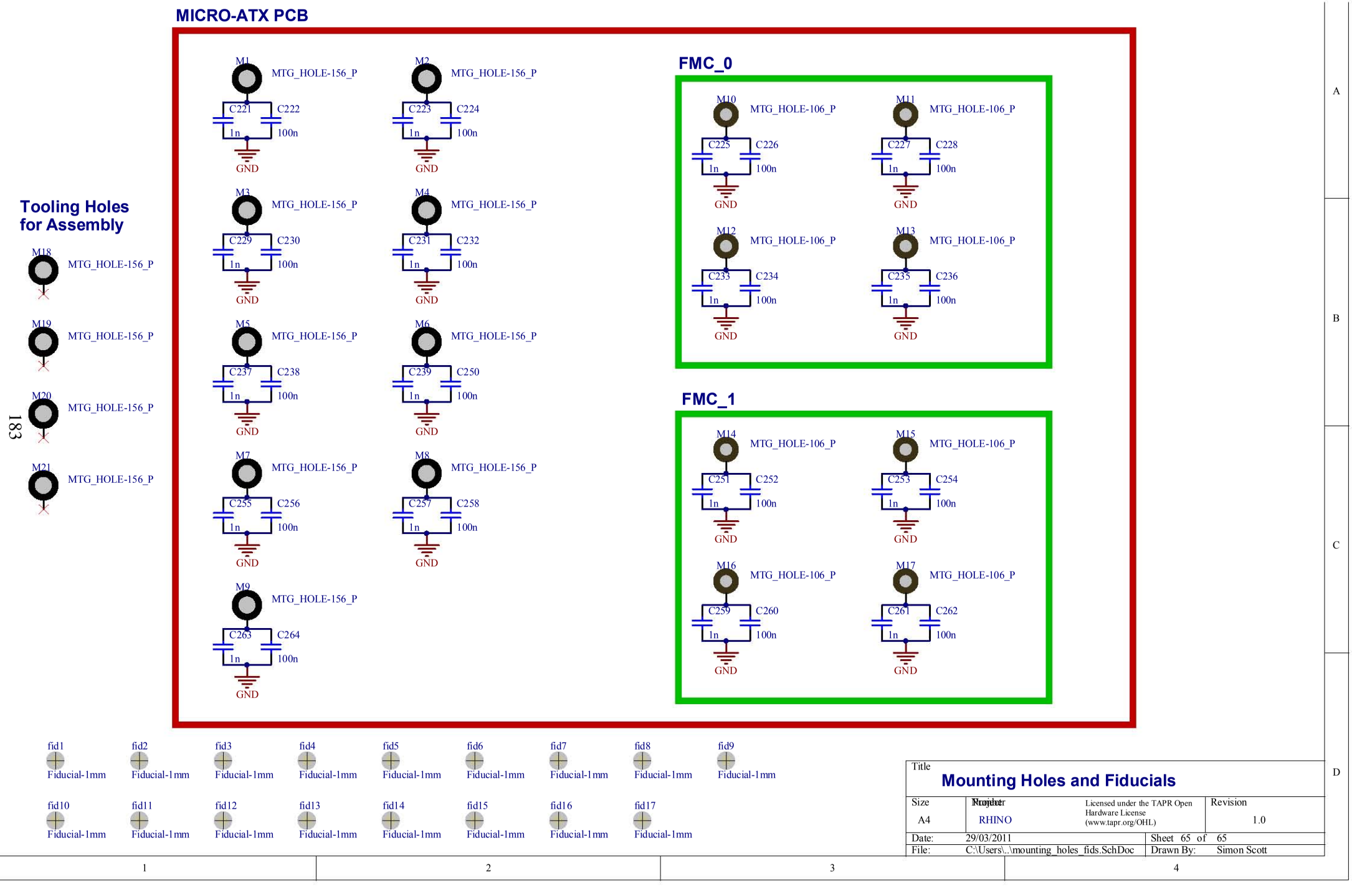


182

Layout Note:

1.) Place these 5 LEDs in a line, next to the Power Good LEDs for the LDO supplies
2.) Place the name of each LED (as indicated on the schematic, next to each LED symbol, e.g. FPGA_1V2) on the silkscreen, alongside each LED

Title: **Power LEDs**			
Size: A4	Project: RHINO		Revision: 1.0
Date: 29/03/2011		Sheet 64 of 65	
File: C:\Users\..\power_leds.SchDoc		Drawn By: Simon Scott	



183

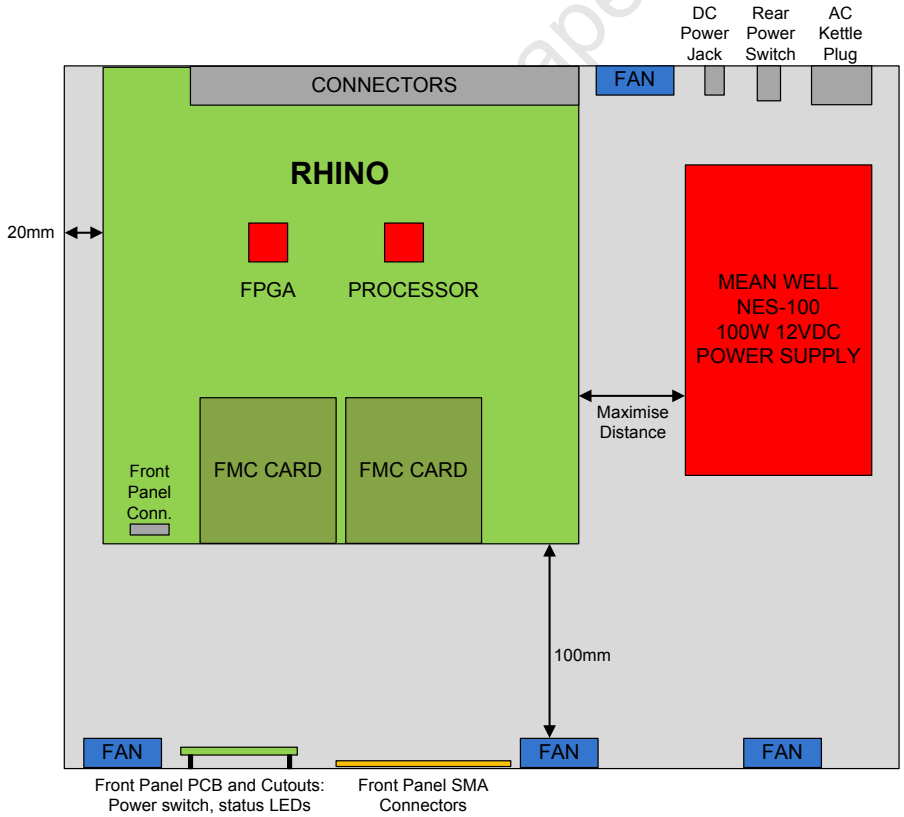
Title
Mounting Holes and Fiducials

Size	Number	Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)	Revision
A4	RHINO		1.0
Date:	29/03/2011	Sheet 65 of 65	
File:	C:\Users\..\mounting_holes_fids.SchDoc	Drawn By:	Simon Scott

APPENDIX C

LAYOUT OF THE RHINO HIDE

The Rhino Hide is the proposed enclosure for Rhino. It is a 1U rack-mount box that houses the Rhino PCB, the power supply, the FMC cards and the RF cards. It also contains a front-panel PCB for power and reset switches and status LEDs. The proposed layout for the Rhino Hide is given below, while the full specification can be found on the attached CD.



Notes:
1.) Red blocks indicate heat sources
2.) Fan blocks indicate possible fan locations. Only 2 fans (at most) should be required. Designer should pick the two best locations to maximise air flow over heat sources
3.) Components have been drawn to scale. However, the positions of components are not fixed, and should be determined by designer.

APPENDIX D

LISTING OF FILES ON ATTACHED CD

A CD has been included with this thesis, containing the PCB design files and test software. The contents of this CD are shown below:

Root Directory	Sub-directories	Description
dissertation	–	This document in PDF format
fpga_gateware	rhino_1gbe_test	Gateware for 1Gbps Ethernet PHY test
	rhino_10gbe_test	Gateware for CX4 test
	rhino_blinky	Gateware to flash FPGA LEDs
	rhino_ddr3_memtest	Gateware for DDR3 SDRAM test
	rhino_fmc0_test_75mhz	Gateware to test FMC connector 0
	rhino_fmc1_test_75mhz	Gateware to test FMC connector 1
	rhino_proc_intrfc_test	Gateware for FPGA-processor bus test
processor_software	u-boot	Source code for the U-Boot bootloader
	u-boot_standalone_apps	Source code for U-Boot standalone test apps
	x-loader	Source code for the X-Loader bootloader
rhino_hide	–	Specification for the Rhino Hide
schematics_and_pcb	altium_rhino_project	The Altium Designer project for the Rhino board
	altium_viewer	Installation file for the free Altium Designer Viewer
	pcb	Altium PcbDoc file and Gerbers
	pcb_3d_model	The STEP model of the Rhino board
	pcb_xrays	The X-rays for one of the Rhino prototype boards
	photos	Photographs of the Rhino board
	schematics	The Rhino schematics in PDF format

Further explanation is now given on how best to view these files. The gateware designs were developed in Xilinx ISE 12, and are hence best viewed in this program. If, however, Xilinx ISE is not available, the .VHD files in each gateware directory can be opened in a text editor.

The schematics and PCB files can be viewed in a number of different ways. The schematics are available in PDF format, which is sufficient for most viewing purposes. The easiest method to view the PCB design file is by using the free Altium Designer Viewer. This viewer, which has been included on the CD, must first be installed. The *Rhino51b.PcbDoc* file in the */schematics_and_pcb/pcb* directory can then be opened directly in the Altium viewer. Alternatively, the Gerber files, which can be found in the same directory, can be opened in a Gerber viewer program.

For the very interested reader, the complete Rhino PCB project can be opened in the Altium Designer Viewer. The project file to open is */schematics_and_pcb/altium_rhino_project/rhino.PrjPCB*. Note that one might need to set the Device Sheets directory in Altium Viewer to */schematics_and_pcb/altium_rhino_project/device_sheets* for this to work correctly.

The 3D model of the Rhino PCB has also been included. This 3D model is in the STEP file format, and can be opened using a number of 3D CAD tools, such as AutoDesk Inventor, Pro Engineer and SolidWorks.

Lastly, the X-rays for one of the PCBs can be found in the */schematics_and_pcb/pcb_xrays* directly. It is suggested that these X-rays are viewed using the HTML document in this directory.

BIBLIOGRAPHY

[1] K. Asanovic et al., "The Landscape of Parallel Computing Research: A View from Berkeley," Tech. Rep. UCB/EECS-2006-183, University of California at Berkeley, Electrical Engineering and Computer Science Department, December 2006.

[2] K. Asanovic et al., "The Parallel Computing Laboratory at U.C. Berkeley: A Research Agenda Based on the Berkeley View," Tech. Rep. UCB/EECS-2008-23, University of California at Berkeley, Electrical Engineering and Computer Science Department, March 2008.

[3] NVIDIA, "TESLA C2050 / C2070 GPU Computing Processor," July 2010.

[4] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fourth Edition*. Morgan Kaufmann, 2009.

[5] B. Cope, P. Cheung, W. Luk, and S. Witt, "Have GPUs made FPGAs redundant in the field of video processing?," in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, pp. 111 –118, Dec. 2005.

[6] Berkeley Design Technology Inc., "BDTI DSP Dictionary," Jan. 2011. [Online]. Available: `http://www.bdti.com/Resources/DSPDictionary`.

[7] C. Yu, K. Kwong, K. Lee, and P. Leong, "A Smith-Waterman Systolic Cell," in *New Algorithms, Architectures and Applications for Reconfigurable Computing* (P. Lysaght and W. Rosenstiel, eds.), pp. 291–300, Springer US, 2005.

[8] Center for Astronomy Signal Processing and Electronics Research, "ROACH," Jan. 2011. [Online]. Available: `http://casper.berkeley.edu/wiki/ROACH`.

[9] Ettus Research LLC, "Ettus Research Website," Jan. 2011. [Online]. Available: `http://www.ettus.com`.

[10] Xilinx, "Design Tools," Jan. 2011. [Online]. Available: `http://www.xilinx.com/onlinestore/design_resources.htm`.

[11] The MathWorks, Inc., "MathWorks - MATLAB and Simulink for Technical Computing," Jan. 2011. [Online]. Available: `http://www.mathworks.com`.

[12] H. K.-H. So and R. Brodersen, "Improving Usability of FPGA-Based Reconfigurable Computers Through Operating System Support," in *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, pp. 1 –6, Aug. 2006.

[13] R. Andraka and A. Berkun, "FPGAs Make a Radar Signal Processor on a Chip a Reality," in *IEEE Proceedings of the Asilomar Conference on Signals, Systems and Computers*, pp. 559–563, 1999.

[14] A. Parsons, D. Backer, C. Chang, D. Chapman, H. Chen, P. Droz, C. de Jesus, D. MacMahon, A. Siemion, D. Wertheimer, and M. Wright, "A New Approach to Radio Astronomy Signal Processing," *General Assembly of the International Union of Radio Science*, October 2005.

[15] C. Chang, J. Wawrzynek, and R. Brodersen, "BEE2: a High-End Reconfigurable Computing System," *Design and Test of Computers, IEEE*, vol. 22, pp. 114 – 125, Mar. 2005.

[16] A. Parsons, D. Backer, C. Chang, D. Chapman, H. Chen, P. Crescini, C. de Jesus, C. Dick, P. Droz, D. MacMahon, K. Meder, J. Mock, V. Nagpal, B. Nikolic, A. Parsa, B. Richards, A. Siemion, J. Wawrzynek, D. Werthimer, and M. Wright, "PetaOp/Second FPGA Signal Processing for SETI and Radio Astronomy," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pp. 2031 –2035, Nov. 2006.

[17] M. Zvelebil and J. Baum, *Understanding Bioinformatics*. Garland Science, 2008.

[18] E. Sotiriades and A. Dollas, "A General Reconfigurable Architecture for the BLAST Algorithm ," *The Journal of VLSI Signal Processing*, vol. 48, no. 3, pp. 189 – 208, 2007.

[19] S. Derrien and P. Quinton, "Hardware Acceleration of HMMER on FPGAs," *Journal of Signal Processing Systems*, vol. 58, no. 1, pp. 53 – 67, 2010.

[20] BEEcube, "BEEcube Inc. - Products," Feb. 2011. [Online]. Available: `http://beecube.com/products`.

[21] Center for Astronomy Signal Processing and Electronics Research, "CASPER - Center for Astronomy Signal Processing and Electronics Research," Feb. 2011. [Online]. Available: `http://casper.berkeley.edu`.

[22] A. Parsons et al., "A Scalable Correlator Architecture Based on Modular FPGA Hardware, Reuseable Gateware, and Data Packetization," *Publications of the Astronomy Society of the Pacific*, vol. 120, pp. 1207 – 1221, 2008.

[23] Ettus Research, "USRP N200 Series Datasheet," Nov. 2010.

[24] R. C. Qiu, Z. Chen, N. Guo, Y. Song, P. Zhang, H. Li, and L. Lai, "Towards a Real-Time Cognitive Radio Network Testbed: Architecture, Hardware Platform, and Application to Smart Grid," in *Networking Technologies for Software Defined Radio (SDR) Networks, 2010 Fifth IEEE Workshop on*, pp. 1 –6, June 2010.

[25] J.-P. Lang, "GNU Radio - WikiStart - gnuradio.org," Feb. 2011. [Online]. Available: `http://gnuradio.org/redmine/wiki/gnuradio`.

[26] J. Davis, C. Thacker, and C. Chang, "BEE3: Revitalizing Computer Architecture Research," Tech. Rep. MSR-TR-2009-45, Microsoft Research, 2009.

[27] 4DSP, LLC, "FMC110 - 4DSP, LLC," Feb. 2011. [Online]. Available: `http://www.4dsp.com/FMC110.php`.

[28] BEEcube, "BEE4 Hardware Platform Brochure," Nov. 2010.

[29] BEEcube, "BEEcube Platform Studio," Sept. 2010.

[30] Cisco Systems, "End-of-Sale and End-of-Life Announcement for the Cisco SFS Infiniband DDR Family Switches [Cisco SFS 7000 Series Infiniband Server Switches] - Cisco Systems," Jan. 2009. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/ps6418/ps6419/ps6421/eol%5Fc51%5F516729.html.

[31] Xilinx, "UG526 (v1.1.1) SP605 Hardware User Guide," Feb. 2010.

[32] Xilinx, "DS160 (v1.3) Spartan-6 Family Overview," Nov. 2009.

[33] Xilinx, "XAPP 059 (Version 1.1) Gate Count Capacity Metrics for FPGAs," Feb. 1997.

[34] Texas Instruments, "SPRT596 Selection Guide: TI processors based on ARM technology," 2011.

[35] Y. Min and C. Ming, "X86 Platform: The Best Infrastructure for MID," in *Management and Service Science, 2009. MASS '09. International Conference on*, pp. 1 –4, Sept. 2009.

[36] D. Marsh, "ARM targets automotive and industrial dominance," *EDN Europe*, pp. 24 – 38, December 2005.

[37] Texas Instruments, "SPRS550B AM3517/05 ARM Microprocessor," July 2010.

[38] CASPER, "ZDOK Pin Numbering - Casper," Sept. 2009. [Online]. Available: http://casper.berkeley.edu/wiki/ZDOK_Pin_Numbering.

[39] VMEbus International Trade Association (VITA), *ANSI/VITA 57.1 FPGA Mezzanine Card (FMC) Standard*, Feb. 2010.

[40] Spansion, "S30ML-P_00 S30ML-P ORNAND Flash Family Datasheet (Revision 03)," Apr. 2008.

[41] Texas Instruments, "SLVS670H TPS65023 Datasheet," Dec. 2009.

[42] Texas Instruments, "SBOS448C INA219 Datasheet," Mar. 2009.

[43] Fujitsu, "o-microGiGaCN Data Sheet (Revision 4.2)," June 2005.

[44] Marvell, "88E1111 Datasheet MV-S100649-00, Rev. I," Mar. 2010.

[45] Xilinx, "UG386 Spartan-6 FPGA GTP Transceivers," Apr. 2010.

[46] National Semiconductor, "AN-1728 IEEE 1588 Precision Time Protocol Time Synchronization Performance," Oct. 2007.

[47] Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, Renesas Electronics Corporation, ST-Ericsson, *On-The-Go and Embedded Host Supplement to the USB Revision 2.0 Specification*, June 2010.

[48] Microchip, "DS21734A TC654/TC655 Datasheet," May 2002.

[49] Xilinx, "UG382 Spartan-6 FPGA Clocking Resources," Jan. 2010.

[50] S. H. Hall, G. W. Hall, and J. A. McCall, *High-Speed Digital System Design–A Handbook of Interconnect Theory and Design Practices*. Wiley-Interscience, 2000.

[51] K. Mitzne, *Complete PCB design using OrCad Capture and Layout*. Newnes, 2007.

[52] D. Brooks, "Crosstalk Coupling: Single-Ended vs. Differential," Tech. Rep. TECH6650-W, Mentor Graphics, 2005.

[53] IPC, *IPC-D-317A, Design Guidelines for Electronic Packaging Utilizing High-Speed Techniques*, Jan. 1995.

[54] Mantaro Product Development Services, "Impedance Calculators," Jan. 2011. [Online]. Available: http://www.mantaro.com/resources/impedance_calculator.htm.

[55] IPC, *IPC-2221A, Generic Standard on Printed Board Design*, May 2003.

[56] R. Prasad, "Voids in BGAs in SMT Assemblies," *SMT Magazine*, May 2001.

[57] IPC, *IPC-7095A, Design and Assembly Process Implementation for BGAs*, Oct. 2004.

[58] Intel, "Intel(R) PRO Network Connections Software Version 12.1 Release Notes," Apr. 2007. [Online]. Available: http://downloadmirror.intel.com/15228/eng/LAN_allOS_12_1_PV_Intel_141678.txt.