RICE UNIVERSITY

Design of Prototyping Platforms for Multiple Antenna Wireless Communications

by

Patrick O. Murphy

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Approved, Thesis Committee:

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

Edward W. Knightly
Associate Professor of Electrical and Computer Engineering and Computer Science

Ashutosh Sabharwal
Faculty Fellow in Electrical and Computer Engineering

Houston, Texas

April 2005

ABSTRACT

Design of Prototyping Platforms for Multiple Antenna Wireless Communications

by

Patrick O. Murphy

As the demand for higher performance wireless communications continues to grow, novel algorithms have been developed which provide increased performance and efficiency. One such class of algorithms involves the use of multiple antennas on either end of a wireless link. Many of these multiple input multiple output (MIMO) algorithms offer impressive performance gains over their single antenna counterparts. The practicality of implementing such algorithms in a real system, however, has received far less attention. A primary reason for this is the scarcity of hardware platforms suitable for implementing and evaluating complex wireless communications algorithms. We present in this thesis two such platforms designed specifically to fill this void. The first platform is constructed from commercial off the shelf hardware, including equipment for baseband processing, RF up/downconversion and wireless channel emulation. The second, more ambitious platform, is built from custom hardware designed specifically for flexible MIMO prototyping.

Acknowledgments

I would like to thank my mentors, colleagues, family and friends for their unending support and encouragement. I would also like to thank our friends in industry, especially at Xilinx and Maxim. Their design advice and generous component donations made this work possible.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

As the demand for higher performance wireless communications continues to grow, novel algorithms have been developed which provide increased performance and efficiency. One such class of algorithms involves the use of multiple antennas on either end of a wireless link. Many of these multiple input multiple output (MIMO) algorithms offer impressive performance gains over their single antenna counterparts.

While the development of MIMO algorithms has received substantial attention by the research community, much less time has been spent studying the practicality of implementing such algorithms and evaluating their real-world performance. This neglect is due largely to the substantial processing resources required to implement wideband MIMO algorithms in real-time. Outside of corporate research labs, very few testbeds exist which provide processing resources of sufficient size and flexibility to develop, prototype and evaluate MIMO algorithms.

The need for platforms for prototyping wireless communications sytems is not new. A variety of other testbeds have been constructed in recent years for exploring multiple-antenna algorithms. Two of these testbeds are briefly described here to provide some background on our motivations for constructing our own.

One such testbed was designed at the University of Texas' Wireless Network and

Communications group[1, 2]. This testbed is built entirely from commercial off-the-shelf hardware, mostly from National Instruments. It provides resources for half-duplex operation at RF up to 2.7 GHz with two transmit and two receive antennas. Embedded PCs are used for baseband processing with algorithms implemented in LabVIEW. Processing and memory limitations prevent real-time operation; data bursts are limited to around 200ms in length with each burst requiring four seconds of processing at the receiver. This duty cycle permits fairly long transmissions (relative to packet sizes in wireless LAN standards), but clearly falls far short of real-time.

Another testbed was built at UCLA and has been used to implement and evaluate a wide variety of MIMO algorithms[3]. This testbed supports up to three transmit and four receive antennas. It operates in real-time at an RF of 220 MHz with a bandwidth of 4kHz. The testbed hardware is a combination of custom RF hardware and commercial equipment for baseband processing. This testbed has been used to evaluate the performance of a large number of space-time algorithms in real wireless environments[4].

Each of these testbeds, and others like them, provide some of the resources we view as necessary for prototyping wideband MIMO algorithms. Neither one, nor any other testbed we could find, addresses all of the requirements we identified for highly flexible, real-time, wideband MIMO prototyping. This thesis presents two testbeds designed to address all of these needs. The first testbed, discussed in chapter 2 was constructed first and is built entirely from commercial off-the-shelf hardware. The second testbed, discussed in chapter 3, consists entirely of custom hardware designed as part of the Transit Access Point project[5, 6]. The design, hardware choices, capabilities and application examples for both testbeds are described in detail.

Chapter 2
Off-the-shelf MIMO Testbed

2.1 Introduction

We present in this chapter a testbed designed for the development, implementation and evaluation of MIMO wireless algorithms. In the interest of building the testbed rapidly, it is constructed entirely from commercial off-the-shelf hardware. The testbed provides hardware for baseband processing, up and downconversion to RF and emulation of multiple wireless channels. It was designed to provide sufficient flexibility to implement a wide range of algorithms while preserving the ability to evaluate an algorithm's resource and power requirements for real-time operation. The testbed hardware and configuration options are discussed here, along with a demonstration of its functionality.

2.2 Testbed Design

The testbed consists of hardware which performs three major functions: baseband processing, conversion to and from radio frequencies and emulation of realistic wireless channels. The components we chose to fill each role are discussed below.

2.2.1 Baseband Hardware

One of the major challenges in designing a testbed capable of real-time wideband wireless communication is choosing baseband hardware. We first identified the major requirements the baseband hardware must meet. First, it must possess sufficient processing power to implement a wide variety of complex algorithms. Reliable wireless communication alone is a computationally intensive task; the added complexity of multiple transmit and receive antennas significantly increases the computational requirements. The hardware must also allow reliable measurement of resource utilization and power consumption on a per-algorithm basis. This is an important requirement as the computational requirements of a particular algorithm play a large role in gauging its suitability for real-world use.

We found early on that of all the options for baseband processing, FPGAs provided the greatest flexibility in algorithm design and visibility of resource utilization. We acknowledge that FPGAs may not be ideal for deployment in large scale wireless systems. They are, however, very well suited for use in a rapid prototyping and research environment such as this testbed.

We decided on FPGA-based development systems designed by Nallatech of Glasgow, Scotland as the baseband processing platform. As shown in Figure 2.1, each board houses a multi-million gate Xilinx Virtex-II FPGA and two channels each of analog-to-digital and digital-to-analog conversion. The board's ADCs and DACs are fast enough to connect directly to the intermediate frequency interfaces on the testbed's RF hardware. These connections allow each board to potentially act as a dual-antenna transceiver. Each board also contains a smaller FPGA dedicated to providing clocks to the larger FPGA and the external analog interfaces. This feature provides a great deal of flexibility in connecting to hardware from different designers with various interface requirements. Finally, the Nallatech board is also a PCI card, providing a high throughput connection over which a host PC can provide and receive data and monitor system performance.

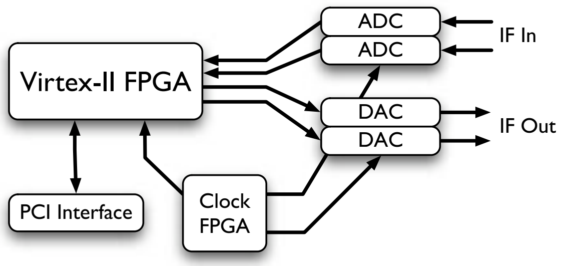


Figure 2.1 FPGA-based processing board

2.2.2 RF Hardware

This testbed is designed for wideband wireless communication in the 2.4 GHz ISM band. The RF hardware must not be tied to any particular standard and must have intermediate frequencies within the sampling capability of the FPGA baseband platform. We also desired control over the synchronization of the frequency translation stages of the radios. Such control allows variation in the carrier frequency offsets between antennas and will prove useful in characterizing the performance of multi-antenna carrier recovery schemes. It also allows offsets between the carriers of multiple transmit or receive antennas to be eliminated entirely, a useful option when testing algorithms lacking carrier recovery systems.

We decided on 2.4 GHz RF hardware from National Instruments. NI has only recently ventured into the RF hardware market, but we found that their hardware fit the needs of the testbed very well. These devices operate with an intermediate frequency of just 15 MHz, easily within range of the FPGA baseband's sampling capabilities. They also support signal bandwidths up to 20 MHz, accommodating a large variety of wideband schemes.

In order to maximize flexibility in choosing multiple antenna configurations, the testbed also uses a number of RF multiplexers. These devices form the interconnect between the RF transceivers and the channel emulators, described in section 2.2.3. The multiplexers allow most configurations to be implemented programmatically without any changes to the equipment's physical connections. This both minimizes the risk of damage to the equipment and allows for a great deal of automation in switching between interconnection configurations.

2.2.3 Wireless Channel Emulation

Another significant challenge in designing this wireless testbed was identifying some means by which we could provide a realistic wireless channel in a laboratory setting. We required the ability to simulate a wide variety of channel conditions, including fading, multi-path effects and delay spread. We also needed to assure repeatability of experimental conditions, a virtually impossible requirement when using real wireless channels.

To fill this need, we chose RF channel emulators from Spirent Communications. Each device is capable of emulating two wireless channels, each with six independently configured paths. Figure 2.2 illustrates the capabilities of each emulated wireless channel.

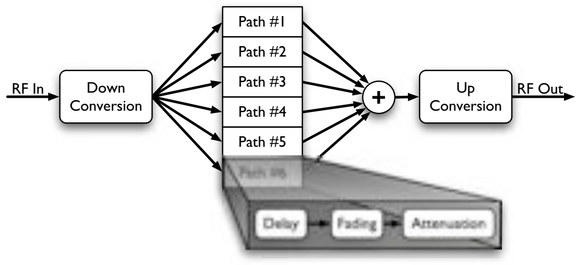


Figure 2.2 Emulated wireless channel

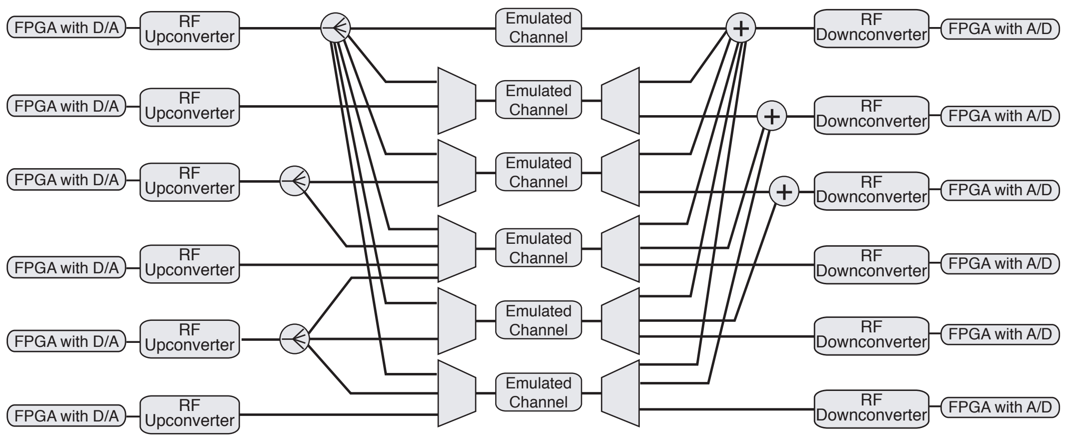


Figure 2.3 Testbed connectivity for half-duplex testing

The emulator can apply one of many fading models, including Rayleigh, Rician and Nakagami, plus programmed delay and attenuation to each path. This flexibility allows the emulation of a large number of realistic line-of-sight and multi-path environments. The emulator can also introduce arbitrary frequency shifts to each path, providing a reliable means by which to simulate the Doppler effects of various mobile scenarios.

The testbed currently houses three such emulators, providing a total of six independent emulated channels. This fairly large number of channels provides significant flexibility in designing test configurations. In terms of transmit-to-receive antennas, the testbed can implement half-duplex systems of 6-to-1, 3-to-2, 2-to-3, 1-to-6 or any smaller configuration requiring six or fewer channels. Figure 2.3 illustrates the full connectivity for multiple-antenna half-duplex testing.

2.3 Application Example: Alamouti Space Time Block Code

Alamouti's transmit diversity scheme was the first example of a space-time code which requires only linear processing at the receiver. Previous space-time coding schemes used trellis based processing. While they provided substantial gains in a wireless communications system, these trellis-based coding schemes were much more complicated to implement than the scheme proposed by Alamouti. This lower complexity makes Alamouti's scheme an ideal candidate for real-world implementation.

The simplest case of Alamouti's scheme utilizes two transmit antennas and one receive antenna. Alamouti included a generalization of his scheme to an arbitrary number of receive antennas, and others later extended his work to include an arbitrary number of transmit antennas [7]. While we present an implementation of the 2-to-1 scheme here, many of the challenges we faced and the solutions we devised would be applicable in more complex systems.

The remainder of this chapter is organized as follows. Section 2.3.1 describes some of the general challenges in implementing a multiple-antenna communications system in real hardware. Section 2.3.2 discusses in detail the development and design of the system. Finally, the verification of the full system using the testbed equipment is described in Section 2.3.3.

2.3.1 Implementation Challenges

Transmitter

The inclusion of an Alamouti encoder in a transmitter design does not significantly increase its complexity. In fact, the hardware realization differs very little from the implementation of two standard wireless transmitters. The only operation the Alamouti encoder performs on modulated symbols is the negation of either the real or imaginary part of a symbol. For most constellations, this processes is analogous to mapping one symbol to another valid symbol. The output of the encoding process is two streams of modulated symbols. Each stream can be fed to identical transmit chains each driving a separate antenna.

Receiver

The implementation of an Alamouti receiver is somewhat more challenging. Some of these challenges stem from assumptions made in the original development of this space-time coding scheme. First, the receiver is assumed to have perfect knowledge of every channel between its antennas and those of the transmitter. Such perfect channel knowledge is often assumed in the process of developing wireless algorithms but is rarely available in practice. As a result, an Alamouti receiver must include some mechanism for channel estimation. Further, this coding scheme requires that the channels remain static for the entire duration of two symbol transmissions. This is a fair assumption in some wireless scenarios but is unlikely in many others. Finally, even if the channel conditions change only every other symbol period, it is difficult for the receiver to have knowledge of the changes at such a rate.

Another challenge in implementing the receiver is compensating for carrier frequency offsets. This is a well understood problem in single antenna systems, and many schemes have been developed to deal with such offsets. In a multiple antenna system, however, carrier offset recovery can be much more complicated. First, many carrier recovery schemes are built around phase detectors which estimate the difference in phase between a received signal and a valid modulation symbol. This approach breaks down in the case of transmit diversity systems as multiple symbols arrive at the receiver simultaneously, significantly increasing the number of valid received symbols. As a result, the carrier recovery system must detect phase errors relative to this expanded set of received symbols. The carrier recovery scheme must also compensate for the potential ambiguity resulting from the system locking to the phases of any of the valid received symbols.

A final challenge in the receiver not described in Alamouti's original paper[8] is the recovery of symbol timing. The receiver must determine the frequency and phase of the transmitter's symbol clock based only on the received signal. Like carrier recovery, this is a well understood problem in the single antenna case. Fortunately, most of these approaches are applicable in the case of multiple antennas as well. One complication not addressed by single antenna timing recovery schemes is the potential for different delays experienced by signals traversing separate channels from independent transmit antennas to the same receiver. Rather than modify the timing recovery system to address this, the difference in delays can be modeled as a phase offset imposed by the channel and dealt with in the process of channel estimation.

2.3.2 Implementation Results

As described above, the implementation of a transmitter with Alamouti encoding does not present any significant challenges. The transmitter combines differentially encoded QPSK modulation with Alamouti encoding. This system outputs two streams of symbols which are fed to identical transmit chains. Each transmit chain includes a raised cosine pulse shaping filter, upsampling and upconversion to an intermediate frequency of 25 MHz. This IF was chosen for compatibility with the National Instruments NI5610 RF Upconverter, the transmit radio we chose for testing the system.

The transmitter design was targeted to a three million gate Xilinx Virtex-II FPGA. The design consumes about 20% of the logic resources in the FPGA and runs with a system clock of 100 MHz. It should be noted that there are variety of optimizations that could significantly reduce the resource requirements of the transmitter design and that 20% of a three million gate FPGA should be considered an upper bound. A block diagram of the transmitter is shown in Figure 2.4.

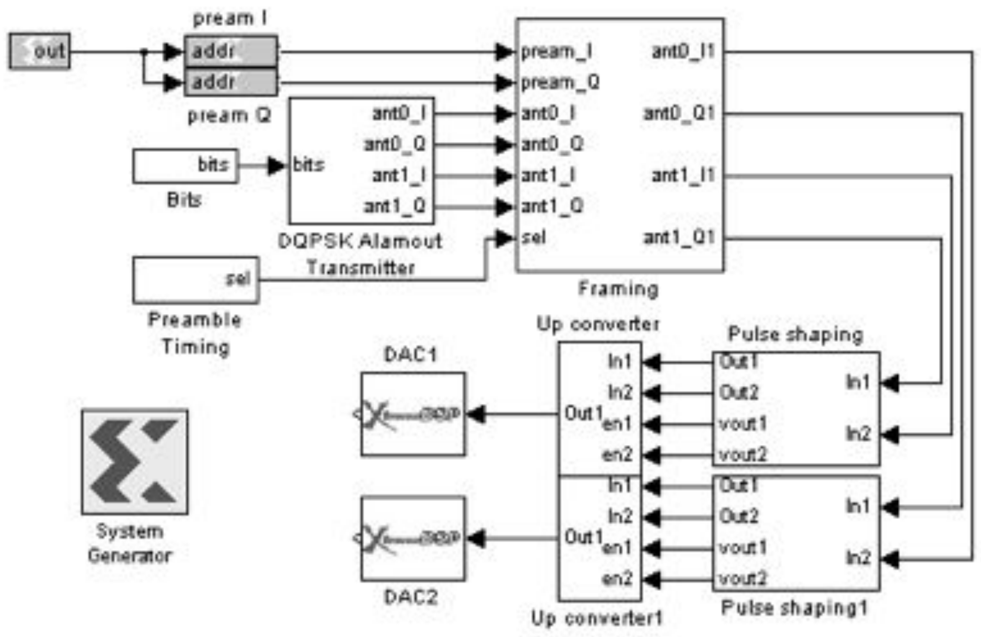


Figure 2.4 System Generator model of Alamouti transmitter

The implementation of an Alamouti receiver was far more challenging. The receiver was designed to interface to an NI5600 RF Downconverter, the device used to translate down from RF to an IF of 15 MHz. Samples of this signal are fed to the FPGA at 60 MHz. The FPGA implements a quadrature downconverter to convert the received signal to baseband. This signal is then downsampled to two samples per symbol and fed to the remaining receiver logic. A block diagram of the complete receiver is shown in Figure 2.5. The functionality and design of each block is described below.

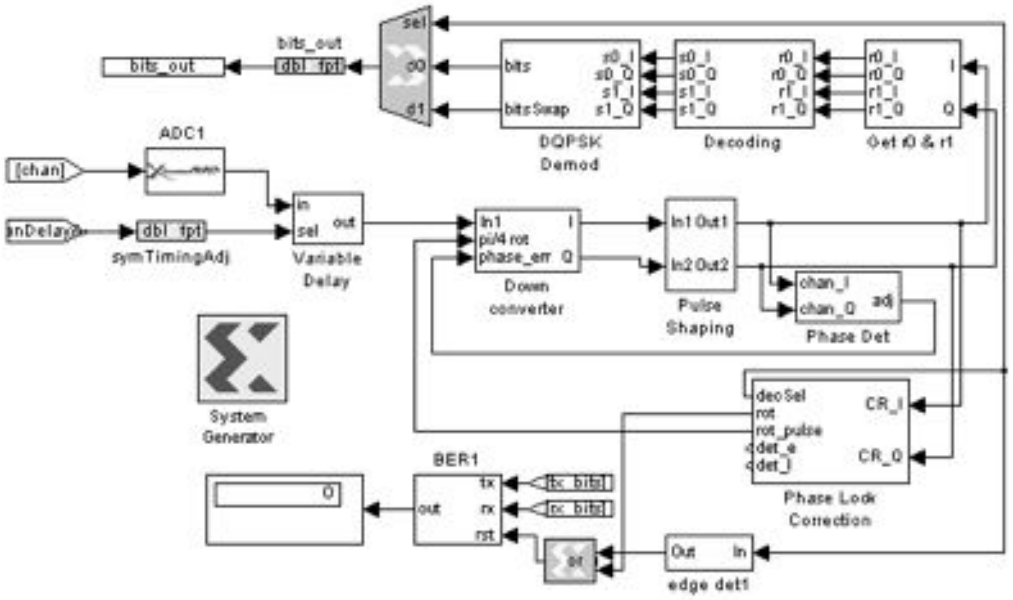


Figure 2.5 System Generator model of Alamouti receiver

Symbol Timing Recovery

The first receiver block following downsampling is a system for recovering symbol timing as described above. This system must choose the optimal time to sample the incoming symbol stream based only on a local estimate to the transmitter's symbol clock and the received data itself. The FPGA cannot modify the frequency of the oscillator providing its clock signal, so any offset relative to the transmitter's clock must be addressed digitally in the FPGA itself.

We chose to use the symbol timing recovery scheme described in [9]. Specifically, the system utilizes an eight branch interpolating polyphase matched filter. Each of the branches represents a different delay ranging from 0 to 7/8 of a sample period. For each received sample, the timing recovery system must decide which of the eight branches provides the best estimate of the transmitted symbol. This decision is based on an error value calculated by a second polyphase structure implementing the derivative of the matched filter. The optimal sampling time is derived by choosing the polyphase branch which minimizes the magnitude of the derivative filter's output. The details of this approach are discussed in [9] and [10].

The primary reason we chose the timing recovery scheme based on a polyphase filter is the efficiency with which it can be realized in hardware. Most illustrations of this scheme show discrete filter blocks for each polyphase branch. In practice, only one of these branches is used during any one sample period. As a result, the entire polyphase filter can be implemented using a single filter block whose coefficients are reloaded in response to changes in the desired polyphase branch. The only complication to this approach is the requirement that the coefficients be reloaded fast enough that a stream of samples can be processed without interruption.

The receiver implements these filters using multiply-accumulate structures. Each filter uses a single multiplier, accumulator and memory block for coefficient storage. Each polyphase branch has eight filter taps, requiring the filter block to run at eight times the input sample rate. With each input sample, the timing recovery control logic specifies which of eight sets of coefficients should be used to process it. The coefficients for all eight sets are stored in the memory block, so the task of reloading coefficients becomes the much simpler one of indexing into the memory. This approach eliminates any latency in switching branches.

Once the timing recovery system has locked and is tracking the frequency and phase of the transmitter's clock, it will eventually traverse all eight branches of the polyphase matched filter. When the polyphase branch index rolls over from seven to zero (or zero to seven, depending on the sign of the frequency offset), it has effectively repeated or skipped an input sample. Many receiver designs operate at N samples per symbol to accommodate some algorithms, frequently equalizers, which require the extra information. Eventually the receiver must choose which of the multiple samples to use for detection and decoding. This selection process is complicated by this stuffing or skipping of samples as it must account for the missing or extra sample. Further, every Nth skipping or stuffing of a sample effectively ignores or repeats a symbol. This complicates the Alamouti decoding process as it expects to operate in parallel on a pair of symbols received in series, and the elimination or repetition of a symbol upsets this paring.

The receiver, which operates at two samples per symbol, includes logic to detect when the polyphase branch index rolls over. With each roll over, this logic directs the downstream sample selector to swap its parity. With every other roll over, the Alamouti decoder is directed to jump forward or back one symbol in order to keep transmitted symbol pairs together. This process inevitably introduces errors to the received bits, either by repeating bits or skipping them altogether. Fortunately, this happens with such infrequency that its effects are minimal. With separate FPGAs implementing the transmitter and receiver, each driven by an independent 60 MHz oscillator, we observed the stuffing or skipping of symbols no more frequently than four times per second, or once per million transmitted symbols.

Carrier Recovery

The next major function performed by the receiver is to compensate for an offset in the carrier frequencies at the transmitter and receiver. We assume that the carriers from the two transmit antennas have exactly the same frequency and are in perfect phase at the transmitter. In real hardware, this assumption translates to multiple radios sharing a common reference clock. Fortunately, this is realistic in most systems, including the one in which we tested this design. Given this assumption, the receiver needs to compensate for just the offset between the transmitted carrier and the local estimate used for downconversion.

Like timing recovery, carrier recovery is a well studied problem in single antenna systems. As discussed above, the use of multiple transmit antennas complicates the process somewhat. The system uses two transmit antennas and QPSK modulation. As a result, the receiver must recognize one of nine valid sums of two QPSK symbols. Eight of these valid sums are non-zero, meaning the carrier recovery system must measure the phase error to one of eight valid phases.

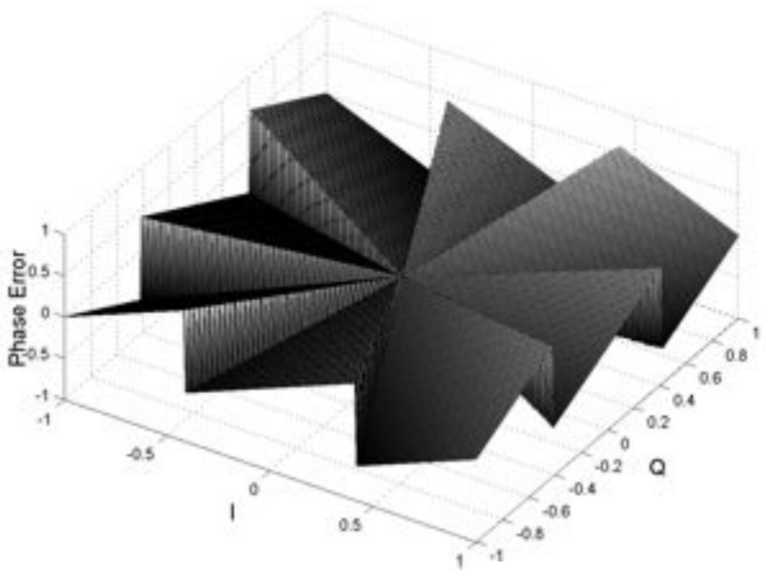


Figure 2.6 Carrier recovery phase error estimation function

The most obvious method for measuring this phase error is to calculate the arctangent of the received symbol. While the CORDIC algorithm provides a method to implement such a function in hardware, such implementations tend to be resource intensive and suffer high latencies in computing results. Instead, we extended the must simpler phase error estimator used in a Costas loop to accommodate the expanded selection of valid phases the receiver must recognize. This new detector calculates a value proportional to both the phase error of a received symbol and the symbol's magnitude. This latter proportionality is important because of the potential for the transmitter to send two symbols which sum to zero. In this case, the only value seen at the receiver will be noise. Any phase error estimate in this case should not be allowed to significantly impact the carrier recovery process. A plot of the function which the phase error estimator implements is shown in Figure 2.6.

The output of this new phase detector feeds a standard 2nd order loop filter which can track both phase and frequency offsets. The output of the loop filter is used to adjust the phase increment of the front-end IF downconverter, eliminating the need for a second digital synthesizer and complex multiplier. This carrier recovery scheme will lock to one of eight phases, offset from the transmitted phase by a multiple of $\frac{\pi}{4}$. The orientations which are rotated a multiple of $\frac{\pi}{2}$ will produce constellations identical to the case of no phase difference. These four orientations will produce symbols which, by means of differential encoding, can be successfully decoded and demodulated. The remaining four orientations, each offset $\frac{\pi}{4}$ from one of the previous four, will produce a constellation which appears rotated $\frac{\pi}{4}$ from the valid nine point constellation resulting from two transmitted QPSK symbols. Symbols received in this orientation cannot be successfully decoded and demodulated. Instead, the receiver detects whether it has locked in one of these states and forces a $\frac{\pi}{4}$ rotation of its phase lock.

Parts of this carrier recovery scheme would break down in cases where higher order modulation or additional transmit antennas were used. In such cases, the direct calculation of the received symbol's phase by means of a CORDIC arctangent would likely be justified. Alternatively, a frequency-locked loop could be used at the receiver's front end to eliminate an offset in carrier frequency, but not phase. Because such loops are not data-driven, they can be designed independent of modulation schemes or the number of antennas at the transmitter. The receiver would then only need to compensate for a nearly constant phase offset.

Channel Estimation

The next block in the design is channel estimation. Alamouti's original algorithm assumed the receiver had perfect knowledge of the channels to each transmit antenna. In reality, the receiver can only estimate the channel conditions. While there has been a great deal of work in designing advanced channel estimation algorithms, we chose to implement a fairly simple scheme in the first revision. The receiver relies on a periodic training sequence that must be inserted into the symbol stream at the transmitter.

The receiver detects this training sequence and correlates the received version against a local copy. This correlation process provides estimates of both channels; these estimates are then used to weight the received symbols and decode the transmitted data.

The channel estimator must listen for the expected training sequence and for one in which the order of symbols in each pair are swapped. This swapping results from the potential ambiguity introduced by the carrier recovery system. If the system locks $\frac{n\pi}{2}$ out of phase from the transmitter, the order of decoded symbols will be effectively swapped. The channel estimator must detect either the standard or swapped training sequence and include the necessary correction factor in the channel coefficients it provides to the decoder.

Decoding & Demodulation

Given two received symbols and estimates of both channels, the Alamouti decoding process is actually very simple. It produces output symbols by using the channel estimates as weights in combining the received symbols. The only complication is the required compensation for skipped or repeated symbols resulting from the timing synchronization process. This compensation is implemented as a three stage shift register which stores incoming symbols. The decoder nominally uses the value stored in the second register but can switch to either the first or third stage as directed by the timing recovery system. This switching allows the decoder to stay aligned to the proper symbol pairing as produced by the transmitter.

The output of the Alamouti decoder is a pair of symbols which began as valid QPSK constellation points. The demodulator first implements the inverse of the transmitter's differential encoding, then demodulates the data by simply thresholding the real and imaginary parts of the complex symbols.

Resource Consumption

The receiver was also implemented in an XC2V3000 FPGA. The full receiver requires just 25% of the logic resources in the FPGA and runs at 60 MHz. At this clock frequency, the receiver achieves a throughput of 7.5 Mb/s.

2.3.3 Verification

Both the transmitter and receiver designs have been synthesized and tested in FPGA hardware. In order to test their functionality, we used the models to establish an end-to-end wireless link. The hardware used for this verification is part of the multiple antenna wireless testbed described in [11]. This tested consists of FPGA hardware for implementing baseband algorithms, wideband radio hardware for translation to RF and channel emulators for simulating various wireless channel conditions. All stages of the testbed are designed to operate in real-time, allowing the design and evaluation of high-throughput, wideband communications schemes.

The FPGA resource utilization for the two models are discussed above. We were able to establish a connection with a sustained throughput of 7.5 Mb/s operating at 2.4 GHz. Separate FPGA boards were used for the transmitter and receiver, and no connections existed between the boards aside from the wireless link. With no common reference clock, there was a definite offset in the symbol timing and generated carriers between the two systems. This offset allowed the verification of the synchronization and carrier recovery loops, both of which locked and tracked for extended periods.

Chapter 3
Transit Access Point Platform

3.1 Introduction

This chapter presents the custom hardware platform designed for the transit access points project. The objective of the TAPs project is to design, analyze, prototype and experimentally study the theoretical underpinnings for a wireless internet that simultaneously achieves deployability, scalability, high performance and a cost-effective model. A core building block will be what we call wireless transit access points. A transit access point is a wireless base station with two major features. First, like any standard base station, it provides wireless data services to mobile users. Second, and more importantly, a TAP is capable of high speed wireless links to other TAPs. These connections utilize multiple antennas at each end to dramatically increase the spectral efficiency and throughput of the link. Such TAP-to-TAP links are designed to supplement, or even replace, the wired network infrastructure usually required when deploying wireless data systems.

The ambitious goals of the TAPs project impose certian requirements on the hardware design. These requirements, discussed in the next section, motivated nearly all the design decisions as described in sections 3.3 and 3.4. And although the hardware was designed to prototype and deploy the algorithmic underpinnings of the TAPs project, the platform is flexible and capable enough for use in a wide variety of applications. A few such applications are discussed in section 3.5.

3.2 Platform Requirements

The design of a TAP must support its use in a variety of situations in which it must provide a variety of services. This is clear in the network illustrated in Figure 3.1. In this example, most TAPs provide connectivity to mobile users in its vicinity. Some TAPs rely on high-speed wireless links for their own connectivity (B & C, for example). Other TAPs have wired connections to a larger network infrastructure (A & D) but must share this connectivity with non-wired TAPs. Finally, some TAPs must act as wireless routers (C), providing network connectivity by multi-hop networking to other access points which cannot directly communicate with a wired node. The hardware design of a TAP must provide all of these capabilities.

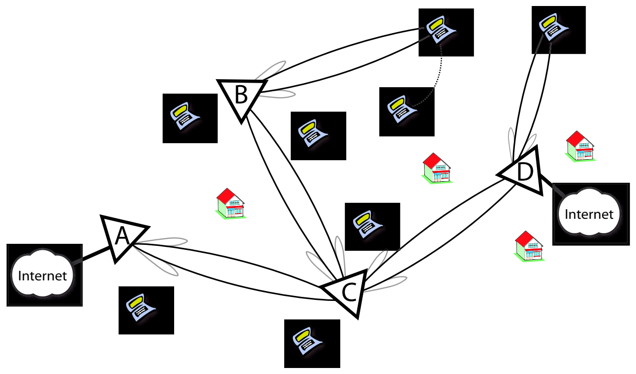


Figure 3.1 Example transit access point network

3.2.1 Wireless Interfaces

One of the fundamental premises of the TAP design is its being equipped with multiple radios and antennas which can be used in unison for spectrally efficient links at very high data rates. This requires that the hardware design provide a means for multiple radios to be driven by a common baseband processor. Further, the physical layer design for TAP-to-TAP links will be entirely custom and will likely not be interoperable with any existing system. Thus, a TAP's radios must be capable of wideband operation in order to support the spectral requirements of these high throughput links but cannot not be tied to any particular standard.

Another requirement imposed by the capabilities described above is the need for multiple air interfaces. An air interface is defined here as the logical abstraction of multiple radios and antennas which act together to communicate over a single link. The number of air interfaces required will vary depending on what services a particular TAP is expected to provide. For example, a TAP which provides connectivity to mobile users and uses a high speed wireless link for its own network connection would need at least two interfaces. A TAP at the core of a network which shares its wired network connection with many other non-wired nodes would need many more.

3.2.2 Processing Resources

The other fundamental requirement of the TAP hardware is the need for sufficient processing power to implement advanced multiple antenna wireless communications algorithms. This requirement poses a kind of cart-before-the-horse problem in that many of these algorithms are still being researched. This is especially true for TAP-to-TAP links, whose physical layer design has only recently been started. Consequently, the TAP hardware design should provide as much processing power in as flexible way as possible. Further, it should support some means of supplementing these processing resources should the base design prove insufficient.

3.2.3 Flexibility

A final requirement of the TAP platform is flexibility. This requirement is imposed by the wide variety of scenarios in which a TAP must operate. This is well illustrated in the example TAP network shown in Figure 3.1. In this example, only a few TAPs have wired network connections. Some TAPs act as routers for neighboring access points while others provide access only to mobile users. Finally, some TAPs utilize multiple TAP-to-TAP links while others require just one. Clearly the TAP platform must be very flexible, able to provide whichever capabilities are required at a particular node in the network.

3.3 High-level Design Decisions

3.3.1 Custom Hardware

One of the earliest and most basic decisions was the choice to design custom hardware for all the major components of the platform. This may seem an odd decision, given the existence of the earlier MIMO testbed described in chapter 2. However, in the early stages of defining the requirements of the TAP platform, it became clear that neither the earlier testbed nor any commercial platform we knew to exist would be suitable. The TAP platform requires substantial baseband processing resources which are tightly coupled with multiple radios, repliacted many times to form a TAP network. The bulk, cost and limited extensibility of the original testbed equipment meant it simply could not be used to construct TAPs.

The primary drawback to the custom hardware approach is the enormity of the design effort. We had never attempted to design boards as complex as the two described later in this chapter. However, it was clear that the benefits of a design customized to the exact needs of the TAP platform justified the effort and risk involved.

3.3.2 Core Components

As mentioned above, the TAP baseband algorithms are not yet defined. Dedicated baseband processors are available for a wide variety of wireless networking standards, but such devices are not suitable for implementing the custom algorithms in a TAP. The device that provides the baseband processing must therefore be very flexible. Further, these baseband algorithms are expected to be very complex, requiring significant processing resources. The baseband processor must be tailored to the DSP-intensive operations, such as filtering and correlation, which are common in communications algorithms. Finally, the processor must be capable of highly parallel operation in order to realize the bandwidth and throughput goals for a TAP.

Next, the TAP baseband processor connects to multiple radios, each of which communicates via a wideband analog interface. Once digitized, each analog interface will require a high throughput, high precision, digital connection to the baseband processor. Consider a TAP's baseband processor controling four radios and each radio has a analog complex baseband interface (e.g. separate I/Q analog signals) with a bandwidth of 20 MHz. The resulting sustained throughput requirements for the baseband processor exceed 300 MB/s in each direction. This is a substantial requirement, generally exceeding the capabilities of processors whose primary off-chip interfaces are standard memory busses.

It became clear very early in the design process that FPGAs were the only devices which could practically meet all of these requirements. Large FPGAs provide tremendous amounts of processing power, all of which, by definition, operates in parallel. All of an FPGAs interfaces to external devices also operate in parallel, significantly easing the aggregate throughput requirement discussed above. FPGAs are also extremely well suited for DSP-intensive operations. For example, large devices include more than 300 dedicated multiplier blocks, all of which can be used simultaneously.

Once FPGAs were chosen as the baseband processor, a radio had to be selected. Two options were considered here. First, it would be possible to design a radio from discrete components, implementing all of the necessary mixing, filtering and amplification in a custom circuit. Such a design would provide a completely generic analog interface and could be tailored to the desired bandwidth and radio frequency specifications. Unfortunately, the design of such a system is a very challenging undertaking which falls well outside the expertise of anyone involved with this project. Instead, a third-party radio transceiver was identified which meets the TAP radio requirements, the specifications of which are discussed below.

3.3.3 Hardware Partitioning

A final high-level design decision involves the partitioning of a TAP into multiple boards. In general, a TAP will have a minimum of three air interfaces, each equipped with four radios. Some TAPs will also have wired network connections in addition to their wireless interfaces. Assuming each wireless interface will require at least one FPGA, a TAP will consist of at least three large FPGAs plus 12 radios. Designing a single board with all of these components would be risky and very expensive. Instead, a TAP's functionality is divided across three boards. The first is a simple radio board, containing a single RF transceiver and the necessary analog-digital conversion. The second board hosts the large baseband FPGA and has slots for four radio boards. This board provides the TAP with a single air interface. Finally, a third board contains a smaller FPGA with a wired network interface. This division of hardware seems natural but poses the significant problem of designing some means to interconnect the boards. The wireless interfaces in a TAP will need to communicate with both the wired and other wireless interfaces. Given the high data rates these interfaces are expected to support, this board-to-board communication must be fast and have low latency. Further, an average TAP will consist of four boards, each of which must be able to communicate with every other. As more boards are added to a TAP (e.g. should additional wireless interfaces be required), this problem of interconnection becomes substantially more complicated.

A possible solution for the board-to-board interconnect is a traditional backplane architecture where each board connects to a common, parallel bus. There are a wide variety of standards for such busses, including the various flavors of PCI, which are easily implemented in an FPGA. The problems which plague such architectures, however, would prove especially troublesome in a TAP. For example, the maximum number of boards which can access the bus must be pre-defined when designing the backplane. Additionally, because boards are connected in parallel, they must contend for bus resources. In the worst case, a pair of boards could monopolize the bus, severely limiting the rate at which any other boards could communicate. Finally, the maximum communication rate in most parallel busses is simply too slow for the high-throughput connections which are required between boards in a TAP.

The solution which was selected for this design is, in many ways, the exact opposite of the bus architecture described above. Instead of having a connections to a common backplane, every board in a TAP is directly connected to every other. This point-to-point topology enables communication between any pair of boards regardless of what resources any other boards are consuming. A fully-interconnected design, however, somewhat complicates the architecture of a TAP. Each board must be equipped with a dedicated connection for every other board in a TAP. Adding a new wireless interface, for example, would consume an additional connection on every existing board. At first a glance, it seems problematic that the resource requirements increase with the number of boards in a TAP. However, if each board could be efficiently equipped with a large number of identical, high speed connections, this architecture is justified. Fortunately, this requirement is easily met in the TAP design, as detailed in the discussion of component selection below.

3.4 Hardware Design

This section discusses the low-level hardware design of the TAP platform. The actual design includes a large number of parts, including analog converters, memory and power regulators. Rather than list all the various parts and their individual functions here, the full schematics are included in Appendix A and are available online [12]. Instead, the primary components and key design features of each of the three TAP boards are discussed.

3.4.1 Wireless Interface Board

Figure 3.2 shows a block diagram of the TAP wireless interface board. Each of its major features is discussed in detail below.

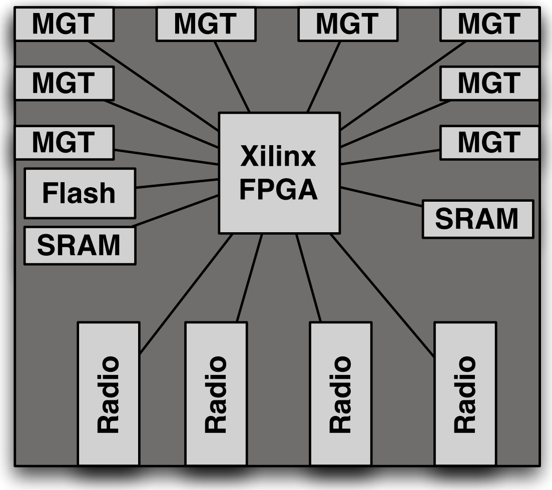


Figure 3.2 TAP wireless interface board

FPGA

The most important component on the board is the FPGA. The choice of this FPGA for baseband processing is actually fairly straightforward. As described above, this design must provide sufficient resources to implement baseband algorithms which, though not yet defined, are expected to be very complex. As a result, the chosen FPGA should be as large as is reasonably possible. The most advanced FPGAs available at the time of this design are those in Xilinx's Virtex-II Pro line [13]. The part chosen for the TAP baseband processor is the XC2VP70, the second-largest FPGA in the Virtex-II Pro family.

Table 3.1 summarizes the resources provided by the TAP baseband FPGA. It also lists the resources for a slightly less capable chip, the XC2VP50. This FPGA is pin-compatible with the larger chip; either can be mounted on the wireless interface board as the TAP baseband processor.

	XC2VP50	*XC2VP70*
User I/O	852	964
Gates	≈5 million	≈7 million
Integrated RAM	4.2 Mb	5.9Mb
18x18 Multipliers	232	328
PowerPC Cores	2	
Rocket I/O	16	
Price Each	≈US$1700	≈US$2500

Table 3.1 TAP baseband FPGA resources

In addition to providing generous logic resources, the Virtex-II Pro baseband FPGA includes a number of additional features which will be critical in the operation of a TAP. The first is the inclusion of multiple PowerPC cores embedded in the logic of the FPGA. These cores are full 32-bit RISC processors whose external interfaces are tied to the device's programmable logic fabric. The TAP baseband FPGA has two such cores which operate independently. These cores are meant to compliment the operation of the FPGA by providing a means to execute pre-existing software code or to perform other processing which is not well suited for implementation in general logic. It is even possible to run a full operating system in one of the processor cores. Xilinx provides a version of Linux which will boot in the embedded PowerPC and can interact with the custom design implemented in the surrounding logic. This offers the interesting possibility of a TAP being, in essence, a PC with many wireless network interfaces. This abstraction could prove very useful when it comes time to implement higher-layer protocols (MAC, routing, etc.) in a TAP.

Multi-gigabit Transceivers

Another feature of Virtex-II Pro FPGAs which is key to the TAP architecture is RocketIO, Xilinx's name for its high-speed serial transceiver technology. These transceivers, generically known as MGTs (**m**ulti-**g**igabit **t**ransceivers), enable very high throughput, full duplex connections for chip-to-chip or board-to-board communication. Each transceiver is capable of communicating at 3.125 Gb/s over a four wire serial link. Multiple transceivers can be bonded together to achieve even faster aggregate throughput. The TAP baseband FPGA is equipped with 16 such transceivers, providing significant resources for off-chip communications. Eight MGTs are wired to off-board connectors on the wireless interface board, allowing up to eight boards to be fully interconnected with multi-gigabit point-to-point links when constructing a TAP. These eight MGTs are terminated with Infiniband-keyed HSSDC2 jacks. These jacks are a standard high-speed serial connector, allowing the use of commercially available cables for connecting TAP boards.

Daughtercard Slots

The wireless interface board has four daughtercard slots. Each slot uses two high density, low profile connectors providing a total of 124 digital signals routed to dedicated pins on the FPGA. Each slot is routed to a separate I/O bank on the FPGA, facilitating the efficient allocation of logic resources to each card. In a TAP these slots will be occupied by radio boards (see section 3.4.2). However any board designed with the same interface can fit in these slots. This provides tremendous flexibility in connecting other peripherals and interfaces to the FPGA. One such board, providing fast Ethernet, USB and serial interfaces, has already been designed.

FPGA Configuration

FPGAs store their configuration in volatile memory which requires it be downloaded every time power is cycled. Further, the FPGA is not active during this download, so the configuration file must be downloaded on startup as quickly as possible. The size of the configuration file grows with the logic resources provided by an FPGA. For the XC2VP70, this configuration file requires 3.2MB of storage. Thus, the TAP configuration system must provide both high-speed, automatic programming and high storage density for configuration files.

A configuration solution which addresses all of these needs was designed using the

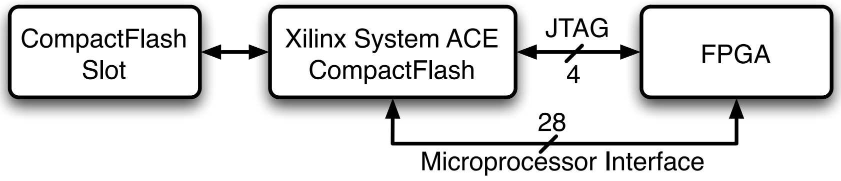


Figure 3.3 TAP FPGA configuration system

Xilinx System ACE CompactFlash chip. This chip is intended specifically for systems requiring rapid in-system configuration and high storage density for configuration files. It has three primary interfaces as shown in Figure 3.3. First, it connects to a standard CompactFlash slot, accommodating the same removable flash cards which are widely available for use in consumer electronics. The CompactFlash card is loaded with configuration files on a PC which, when plugged into the board, are downloaded to the FPGA by the System Ace chip. This download occurs over the System Ace's second interface, a standard JTAG port for target devices. The third interface is a standard microprocessor memory bus between the FPGA and System ACE chip. This interface allows the FPGA to access registers in the System Ace to monitor status or trigger re-configuration. More importantly, this interface provides the FPGA read/write access to the CompactFlash card. This enables two important features. First, the FPGA can overwrite or add new configuration files received over one of its network interfaces, then trigger re-configuration using the new file. In other words, a TAP can be reprogrammed remotely without any physical access. Second, the FPGA can use empty space on the flash card to store any information it requires between power cycles. This includes both board-specific parameters, like MAC addresses and serial numbers, and state information like log files. The CompactFlash card can be removed without interrupting the FPGA, allowing a card filled with log files to be swapped for an empty one on the fly.

The FPGA's JTAG interface can be isolated from the System ACE by removing four jumpers. This is intended to facilitate both hardware debugging and the creation of one long JTAG chain containing every FPGA in a multi-board TAP. When these jumpers are replaced with cables connecting the boards' JTAG chains together, a single System ACE will be able to configure every FPGA in a TAP. This will simplify the process of remotely reprogramming a TAP.

SRAM

The wireless interface board includes two static RAMs. Each is a 100 MHz 9Mb (36x256k) SRAM wired to dedicated pins in independent I/O banks on the FPGA. The two banks can be used separately or together as a single 72-bit wide memory. These RAMs were included in anticipation of memory requirements beyond what is provided by the FPGA's internal block RAMs. Potential uses include program and data storage for the PPC cores or as packet buffers for the MAC and PHY algorithms. Both RAMs use zero bus turnaround (ZBT) interfaces, allowing one read or write access per cycle without any wait states.

User I/O

In order to speed hardware debugging and algorithmic development, the board includes a number of components dedicated to user interface. Three push buttons, four LEDs, two seven segment displays and 16 general I/O are all available for users to interact with designs implemented in the FPGA. Once a TAP is deployed, these components will serve little function and can easily be omitted during assembly. Experience has shown, however, that such user I/O are invaluable during the incremental development and implementation of algorithms on the FPGA.

Power Supplies

Large FPGAs are capable of consuming very significant amounts of power. The actual power requirements depend heavily on the design implemented in the FPGA. The number of I/O buffers, block RAMs, multipliers, logic slices, processor cores and MGTs used in a design all significant affect the total power consumption. A conservative estimate of the power consumption for a moderately full XC2VP70 exceeds 20 watts[14]. The short term consumption can be even higher, depending on the number of transistors switching in a given clock cycle. These requirements are further complicated by the three different voltage supplies used by the FPGA: 1.5 v for its core, 2.5 v for auxiliary functions and 3.3 v for I/O buffers. The four daughtercard slots also have the potential to consume significant amounts of power. This is especially true for radio daughtercards whose analog converters, transceivers and power amplifiers all draw significant currents.

The wireless interface board's power system addresses all of these requirements using three power modules from Texas Instruments. The first supplies up to 10 A to each of the three voltages required by the FPGA. The other two are dedicated supplies for the four daughtercard slots, providing +12 v at 30 watts and -12 v and 7 watts. The outputs of all three modules are isolated from the shared 48 v input. The daughtercards do not share the FPGAs supplies. Instead, a daughtercard must regulate its own power from the ±12 v supplied through their connectors. As described in section 3.4.2, the analog circuity on the radio board requires very low-noise power supplied by a linear regulator near the circuit. This requirement would be impossible to meet if the power regulators were external to the radio board. The two daughtercard power modules are through-hole components, facilitating their mounting or removal after initial assembly. In some cases, a wireless interface board will be used without daughtercards, utilizing only the FPGA for extra processing resources. The daughtercard power modules can be removed in such scenarios to reduce costs and power consumption.

PCB Details

The TAP wireless interface board is a 16 layer 8" x 8" x 0.092" printed circuit board. The board was routed entirely by hand in order to maximize signal integrity by minimizing layer transitions by signal traces. The flexible pinout of the FPGA allowed every signal to be routed using no more than two vias. In other words, every trace on the board is routed on just one or two layers, passing through no more than two layer transitions (generally from the top to an inner routing layer and back). Ten layers are dedicated to signal routing; the remaining six are used for power distribution and ground planes. Controlled impedance is required during fabrication on layer 6 in order to assure the $100\Omega \pm 5\%$ differential impedance of the MGT traces. The board is manufactured using FR-4 dielectric with 0.005" minimum trace width and spacing. The routing was completed using only through-hole vias in order to minimize fabrication costs*. There is an extensive network of bypass capacitors, more than 300 in all, mounted on the back of the board. These capacitors provide both paths for signal return currents and local power storage for the FPGA when its short term current requirements vary faster than the power regulators can adapt. The quantity and distribution of values for these capacitors are designed according to recommendations from Xilinx[15]. See Appendix **??** for plots of each layer.

The first revision of this board has been manufactured, assembled, tested and is fully functional. The only error discovered so far is a missing pull-up resistor on one signal. A second revision is planned which fixes this error and rearranges components

*A version of the board was actually completed using just 10 layers and blind vias (holes which do not extend all the way through the board). This version was scrapped when the fabrication costs were quoted at $6,000 per board, 10x the cost of the final 16 layer version.

for more flexibility in designing daughtercards. Another future revision will likely be designed to take advantage of Xilinx's newer family of Virtex-4 FPGAs.

3.4.2 Radio Board

The second core hardware design is the TAP radio board. This board is designed as a daughtercard for the wireless interface board described above. It implements full transmit and receive chains, all the way from the FPGA's digital interface to analog RF. A block diagram of the board is shown in figure 3.4 and its major features are discussed below.

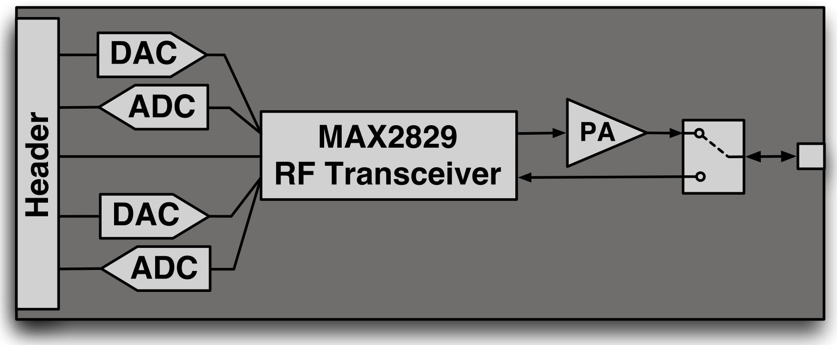


Figure 3.4 TAP radio board

RF Transceiver

A radio transceiver suitable for use in this design was to be the most difficult part to choose. The difficulty stems from the scarcity of wideband radio chips which are not tied to a particular baseband processor. A vast majority of wireless networking designs are built to comply with one or more of the IEEE 802.11 standards and provide little flexibility beyond those specifications. A TAP's wireless interfaces, on the other hand, will not use these standards and must be designed to support a multiple antenna physical layer which is still under active development.

Fortunately, a suitable radio chip was recently released which exceeds all of the requirements discussed above. This radio, the MAX2829 from Maxim Integrated Products, is a direct conversion radio transceiver which supports both the 2.4 GHz and 5 GHz ISM bands [16]. Although it is intended for use in 802.11a/b/g/n designs, this transceiver provides a flexible analog baseband interface. This interface allows the translation of any waveform, with a bandwidth up to 40 MHz, between baseband and RF, regardless of the waveform's adherence to an 802.11 standard. Further, when driven by a common reference clock, the phase coherency of the local oscillators in multiple MAX2829 transceivers is guaranteed. This feature is critical in MIMO applications as many algorithms require carefully controlled phase relationships and accurate measurements of phase among multiple antennas.

Data Converters

The wireless interface board daughtercard connectors provide a strictly digital interface to the FPGA. The MAX2829, however, has an analog baseband interface, consisting of four differential analog signals (I and Q for transmit and receive). The radio board includes the necessary A/D and D/A converters to complete these interfaces. The receive path uses the AD9248 A/D converter. This part is a dual-channel 14-bit converter which samples at up to 65MS/sec. The sampling rate and input bandwidth both exceed the requirements for operating the MAX2829 in 40 MHz bandwidth mode. The transmit path uses the AD9777 D/A converter. This is a dual-channel 16-bit converter which samples at up to 160MS/sec. The use of dual-channel converters is intended to minimize gain and delay errors which would manifest themselves as I/Q gain and phase imbalances in the transmit and receive paths. A separate 10-bit 20MS/sec A/D converter is used to convert the MAX2829's received signal strength indicator (RSSI) signal. All three converters have parallel digital interfaces connected to dedicated pins on the daughtercard connectors.

RF Front End

The TAP radio board's RF front end includes all the components required to interface the MAX2829's RF transmit and receive signals with off-board antenna connectors. The transmit chain primarily consists of passive components for impedance matching and a dual-band power amplifier from Sharp Microelectronics. The amplifier has an output power of +18dBm in both the 2.4 GHz and 5 GHz bands. The receive chain consists of bandpass filters for both RF bands and passives for impedance matching. The two chains are joined by diplexers and a double pole double throw (DPDT) antenna switch. This switch has four ports: a transmit input, driven by the combined 2.4 GHz and 5 GHz RF signals, a receive output, driving a combined 2.4 GHz and 5 GHz signal and two bi-directional antenna ports. This switch is controlled by the FPGA and allows a variety of configurations for the two antenna connectors. One antenna can be bi-directional at 2.4 GHz with the other used at 5 GHz. Alternatively one antenna can be used in both bands for transmission only with the other antenna for used receiving. Since both the antenna switch and the MAX2829 are controlled by the FPGA, these modes can be selected on the fly without any hardware changes.

The RF front end section of the TAP radio board is based on a reference design provided by Maxim. Part of this reference design is discussed in an application note authored by Maxim engineers[17]; the schematics and layout files are not available for general distribution.

Clocking

A variety of clocks are required by the components on this board, all of which are very sensitive to the quality of the clock signals. The data converters require a clock with a strict 50% duty cycle and very low jitter. Further, the clock signals driving each channel in the dual-channel converters must be perfectly synchronized in order to minimize any offsets in the I and Q samples. The reference clock signal provided to the MAX2829 transceiver is used to generate the 2.4 GHz and 5 GHz carriers for conversion to and from RF. Any phase noise in this clock signal will directly translate to the RF signal, potentially reducing noise margins and significantly decreasing performance. Finally, all of these clock signals must be perfectly synchronized across all the radio boards connected to a common wireless interface board. This is a strict requirement of MIMO algorithms which may operate by applying minor phase offsets between antennas, possible only when a common reference clock is shared by all components.

This complicated clock distribution problem is solved by using a pair of off-board master oscillators and clock distribution chips, one each for the converter and MAX2829 reference clocks. The radio board has two MMCX jacks for receiving these clock signals by coaxial cable. The cables used to route clocks to each radio board can be matched in length, minimizing any skew between the boards. This approach also isolates the clock signals from any digital signals routed on the wireless interface or radio boards, minimizing the potential for coupled noise.

Power Supplies

Analog circuits, and RF circuits in particular, require very low-noise power supplies in order to maximize performance. One key way to reduce the noise in analog power supplies is to isolate the supplies for the digital and analog sections of mixed-signal chips. This approach is used on the TAP radio board. A dedicated voltage regulator is used for the digital interfaces on the fast data converters. A separate regulator drives the analog supplies on the converter. A third regulator is used for the MAX2829's supply in order to further isolate the RF chain from potential supply noise. A fourth regulator is dedicated to the supply pins of the transceiver's voltage-controlled oscillator (VCO). Any noise present in the VCO supply can directly impact the noise floor at RF; the dedicated regulator is used to mitigate this as much as possible. Finally, multiple bypass capacitors of various values are placed adjacent to every power pin on every component to minimize coupled and emitted interference from suboptimal return current paths.

PCB Details

The TAP radio board is a 6 layer 2.9" x 1.7" x 0.062" printed circuit board. The board was routed entirely by hand in order to maximize signal integrity and adhere to the strict impedance requirements of the RF circuits. It was manufactured with FR-4 dielectric and 0.005" minimum trace width and spacing. Layers 1, 3, 4 and 6 are used for routing signals; layers 2 and 5 are used for power and ground planes. Where not otherwise occupied, all six layers are flooded with copper tied to ground. All analog and RF signals are routed on the top layer where an impedance of 50Ω $\pm$5% is guaranteed by the board manufacturer. See Appendix **??** for plots of each layer.

3.4.3 Wired Network Board

The final board in the TAP platform provides a wired network connection and will generally be used by TAPs which have direct access to a internet connection. This board is based on a Virtex-II Pro FPGA smaller than the one mounted on the wireless interface board. This FPGA provides eight multi-gigabit transceivers (MGTs), six of which are dedicated to communicating directly with wireless interfaces in a TAP. The remaining two MGTs are used as gigabit Ethernet connections. The FPGA also provides two PowerPC cores, an ideal resource for implementing the various networking algorithms operating above the MAC and PHY.

The design of this board is currently underway.

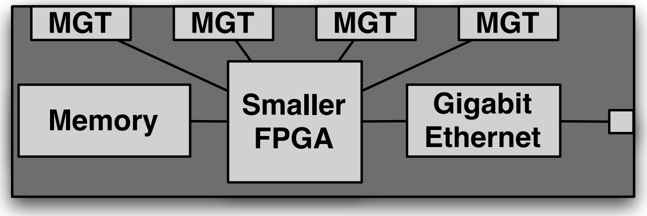


Figure 3.5 TAP wired network board

3.5 Applications

This section describes some applications for the TAP hardware platform. Though the platform was designed to meet the requirements of a TAP network, it provides sufficient resources and flexibility to be used in many other settings.

3.5.1 Multi-hop Fixed Wireless Networks

This platform was designed to meet the requirements presented in the original TAP proposal[5]. This section describes how the custom hardware designs discussed above can be assembled and utilized to meet this requirements.

Figure 3.6 shows the typical TAP network discussed in section 3.2. In this network the five TAP nodes each perform different functions which require different resources. Node A, for example, has a wired network interface connected to the Internet, a wireless TAP-to-TAP link with node C and provides wireless access to nearby mobile users. Contrast this with node C which acts only as wireless router using three TAP-to-TAP links.

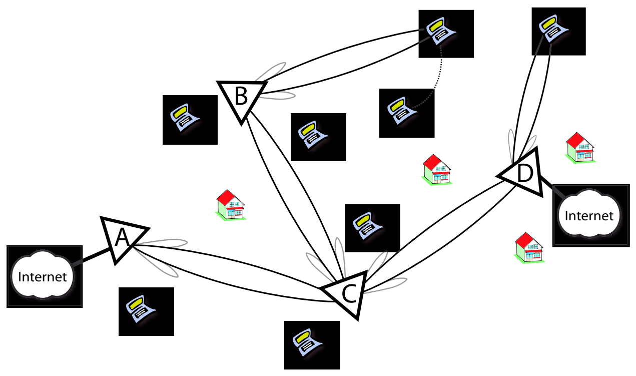
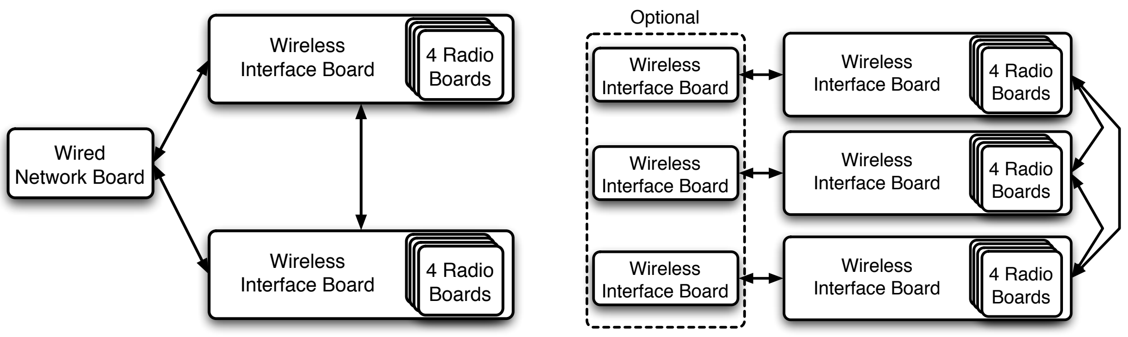


Figure 3.6 TAP network



(a) Hardware for node A (b) Hardware for node C

Figure 3.7 Building TAPs

Though they serve very different functions, both of these nodes can be constructed from the custom TAP hardware. Node A can be built from a wired network board and two wireless interface boards, one for the point-to-point link, the other for mobile user access. Each wireless interface board would have four radio boards connected in order to provide MIMO capabilities on both wireless interfaces. The three TAP boards would be interconnected by MGTs. This configuraiton is shown in figure 3.7(a). Node *C* would be built entirely from wireless interface and radio boards. Each of the three wireless interfaces would require its own board, each equipped with four radios. These three boards would be fully interconnected by MGTs. It is likely that the implementation of a full four antenna MIMO transceiver will require more than a single FPGA. Node *C* would likely need extra processing resources for each wireless interface. The TAP wireless interface board can easily fill this role by simply using it without radios. This scenario is shown in figure 3.7(b). Each secondary processing board would be connected by another MGT to its primary wireless interface board. Should even more processing resources be required, this approach could be extended using additional wireless interface boards interconnected by MGTs.

3.5.2 Real-time Networking Experiments

One of the most promising alternate uses for the TAP platform is in the construction of testbeds for experiments in wireless network algorithms above the physical layer. There has been significant interested in recent years in the empirical evaluation of MAC and routing algorithms in real-world wireless environments. Part of this interest stems from the difficulty in accurately simulating both wireless physical layer algorithms and real wireless channel effects. A number of testbeds have been constructed using standard PCs and 802.11b wireless networking interfaces for the purpose of performing such evaluation[18, 19]. A major drawback to this approach, however, is the limited degree to which the physical and MAC layer implementations can be observed and modified in standard wireless network interfaces. For example, a number of changes to the IEEE 802.11 MAC protocol have been proposed that aim to reduce its overhead and improve its performance[20, 21]. Many of these proposals are supported by simulation results demonstrating performance improvements, but none which requires substantial changes to the MAC is supported by results from evaluation in real wireless environments. This is due largely to academic researchers not having access to a platform with a standard wireless physical layer and programmable MAC. Commercial wireless networking products implement the MAC protocol in an ASIC which, in order to maintain standards compliance certification, must not be modifiable by the end-user. This results in the inability of networking researchers to evaluate the real-world performance of alternate MAC protocols in testbeds built from commercial hardware.

The TAP platform does not suffer this restriction. A TAP wireless interface board with one radio board combined with a standards-compliant physical layer implementation forms a wireless network node with a fully programmable MAC. The MAC could be implemented either in the FPGA's logic resources or in one of its PowerPC cores. This latter option is particularly useful given that many MAC protocols are already implemented in C for simulation purposes. With some modifications, this code can be targeted to one of the embedded PPC cores where it can interface with the physical layer implemented in the surrounding logic.

Other features of the TAP platform further enhance its use in networking experiments. The nonvolatile storage space provided by the SysateACE CompactFlash interface can be used to store log files and packet traces at each node for offline analysis. Additional wireless interface or wired network boards can be used to provide remote access to a node via an interface independent from the wireless interface under test. And additional radio boards can be added to enable testing of MAC protocols for upcoming MIMO wireless network standards like IEEE 802.11n and 802.20.

Chapter 4
Conclusions and Future Work

The most significant contribution of this work is the design of the TAP wireless interface and radio boards. These boards are the building blocks for the TAP hardware platform and will play an essential role in the continuing development and eventual deployment of TAP networks. However, a substantial amount of work remains in developing and implementing the algorithms required to achieve the performance goals of the TAPs project. These efforts are already underway, both at the physical and MAC layers. A single antenna OFDM transceiver has been implemented and verified which will form the basis for the full MIMO physical layer. A IEEE 802.11 based MAC implementation targeted to one of the FPGA's embedded PowerPC cores has also been started. These efforts will eventually culminate in the use of a four antenna transceiver, built entirely from the custom hardware described in this thesis, to achieve the aggressive spectral efficiency and throughput goals of the TAPs project.

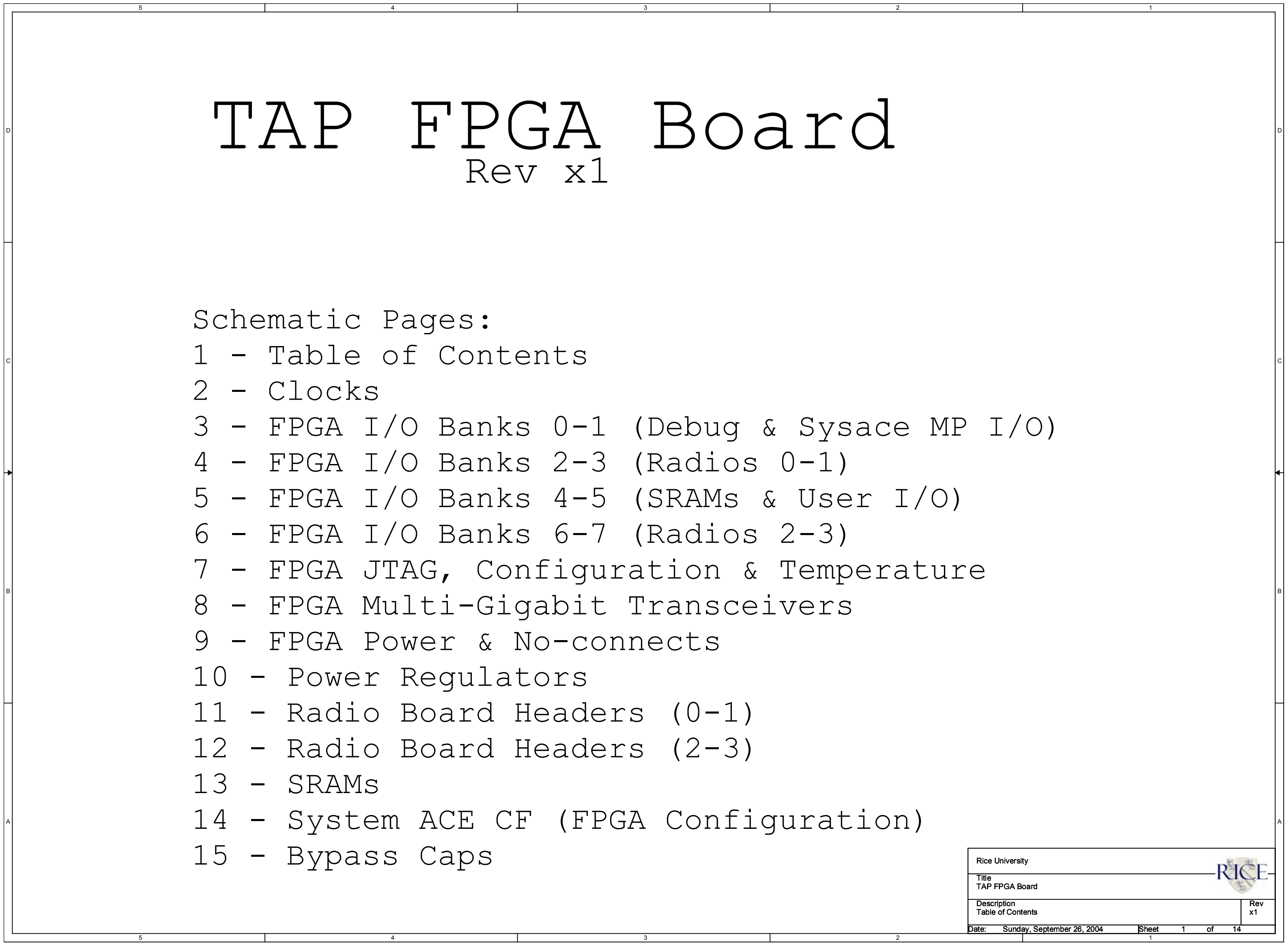
We fully expect that these algorithmic implementation efforts will expose various strengths and weaknesses of the hardware platform. This, in turn, will drive the development of future revisions of the hardware. We also expect that, in parallel to the TAP implementation efforts, other research groups will begin using this hardware for their own projects. Networking experiments will likely be among the first such outside applications for the reasons discussed in section 3.5.2. These, and other outside uses of the hardware, will provide valuable feedback for future revisions and extensions of the platform.

Appendix A
PCB Schematics

This appendix includes the complete schematics for the TAP wireless interface and TAP radio boards. These schematics were prepared in Cadence OrCAD Capture CIS v10.1. The Capture project files are available from http://taps.rice.edu/hardware/. Each page includes a brief description of its contents in the bottom right corner. Wires terminated with a « symbol indicate connections across pages with the corresponding schematic page number appended to the signal name.

A.1 TAP Wireless Interface Board Schematics

Following are the 15 pages of schematics for the TAP wireless interface board.



TAP FPGA Board

Rev x1

Schematic Pages:

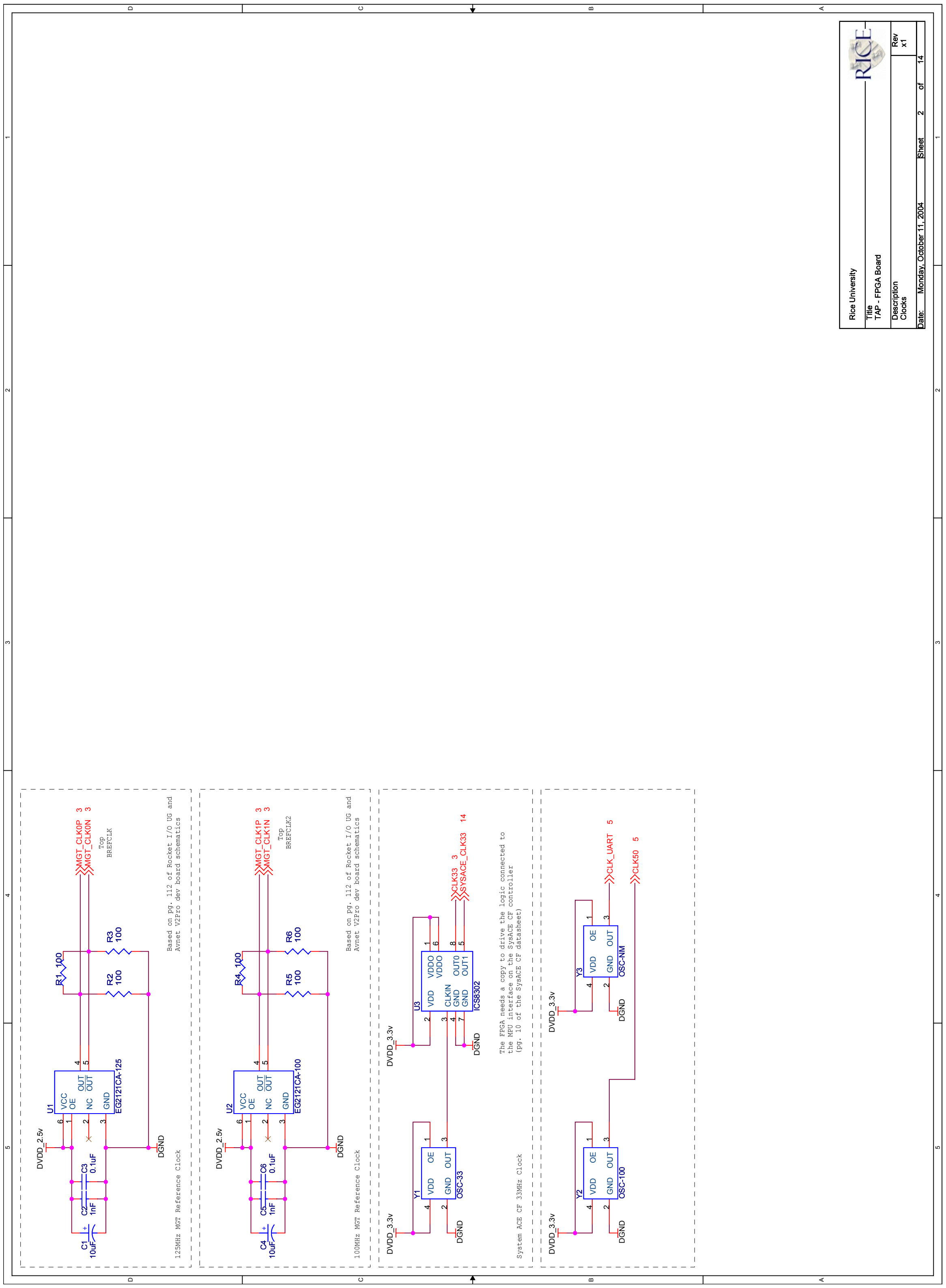
Rice University

Title
TAP FPGA Board

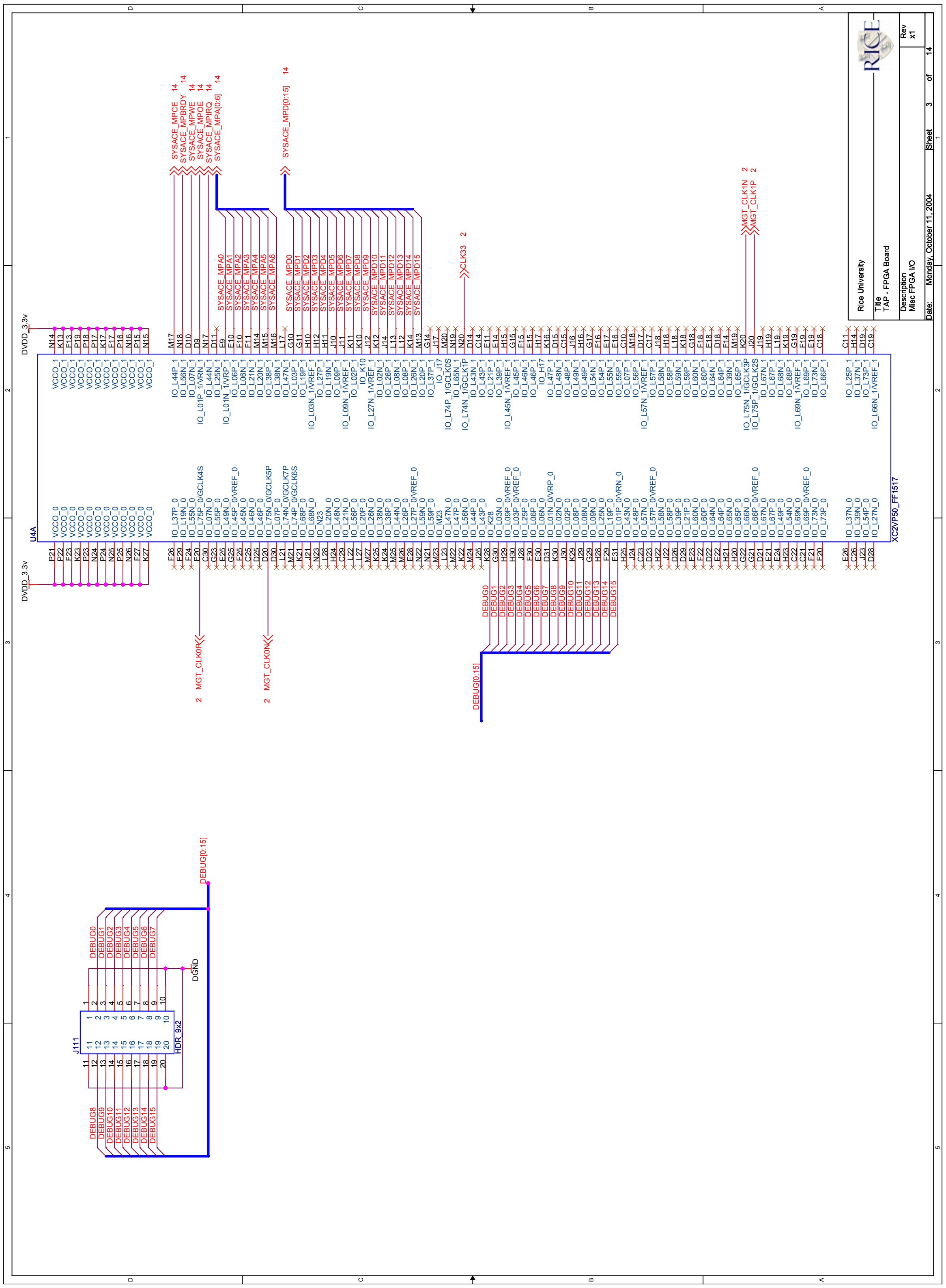
Description
Table of Contents

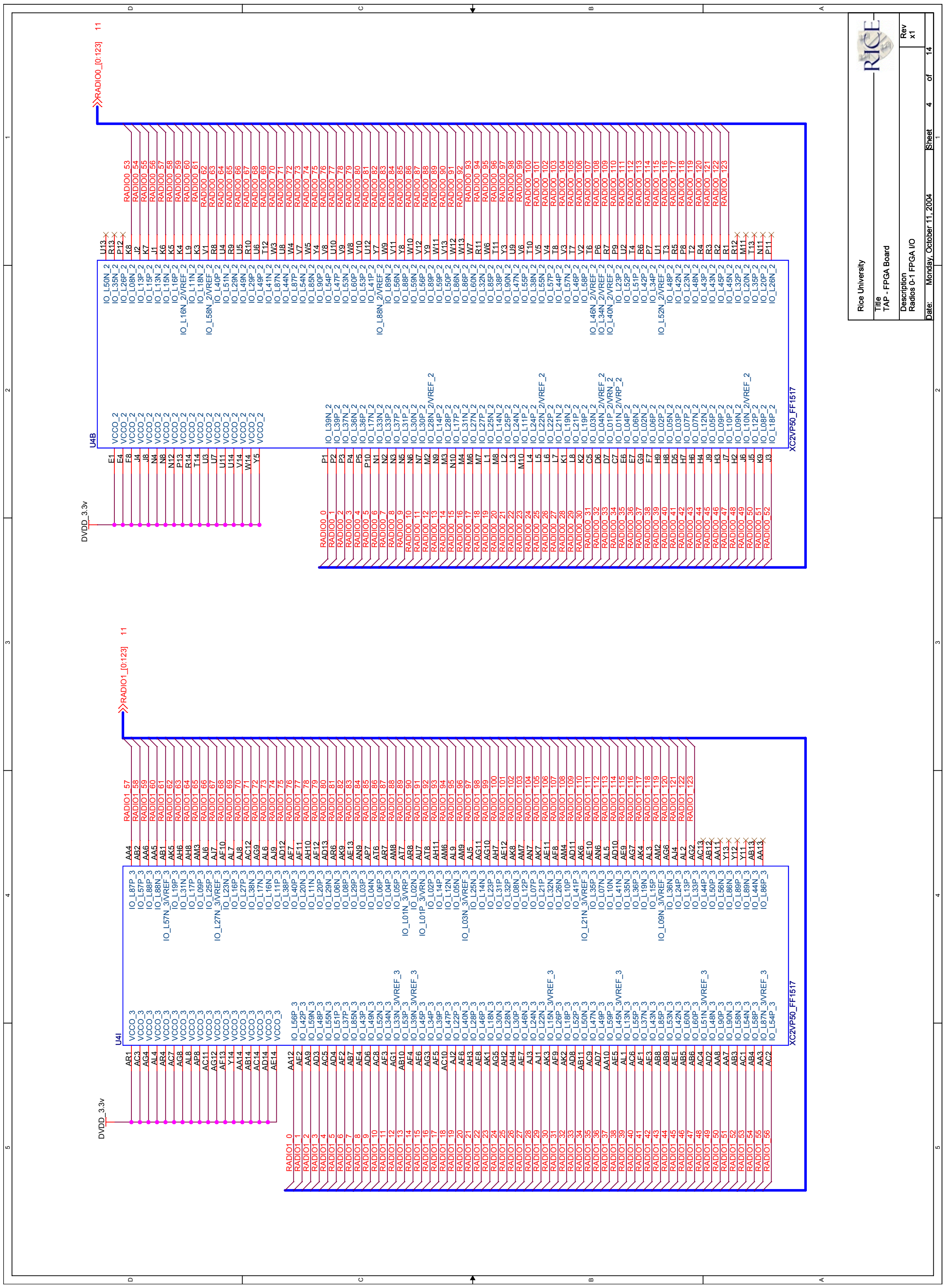
Rev
x1

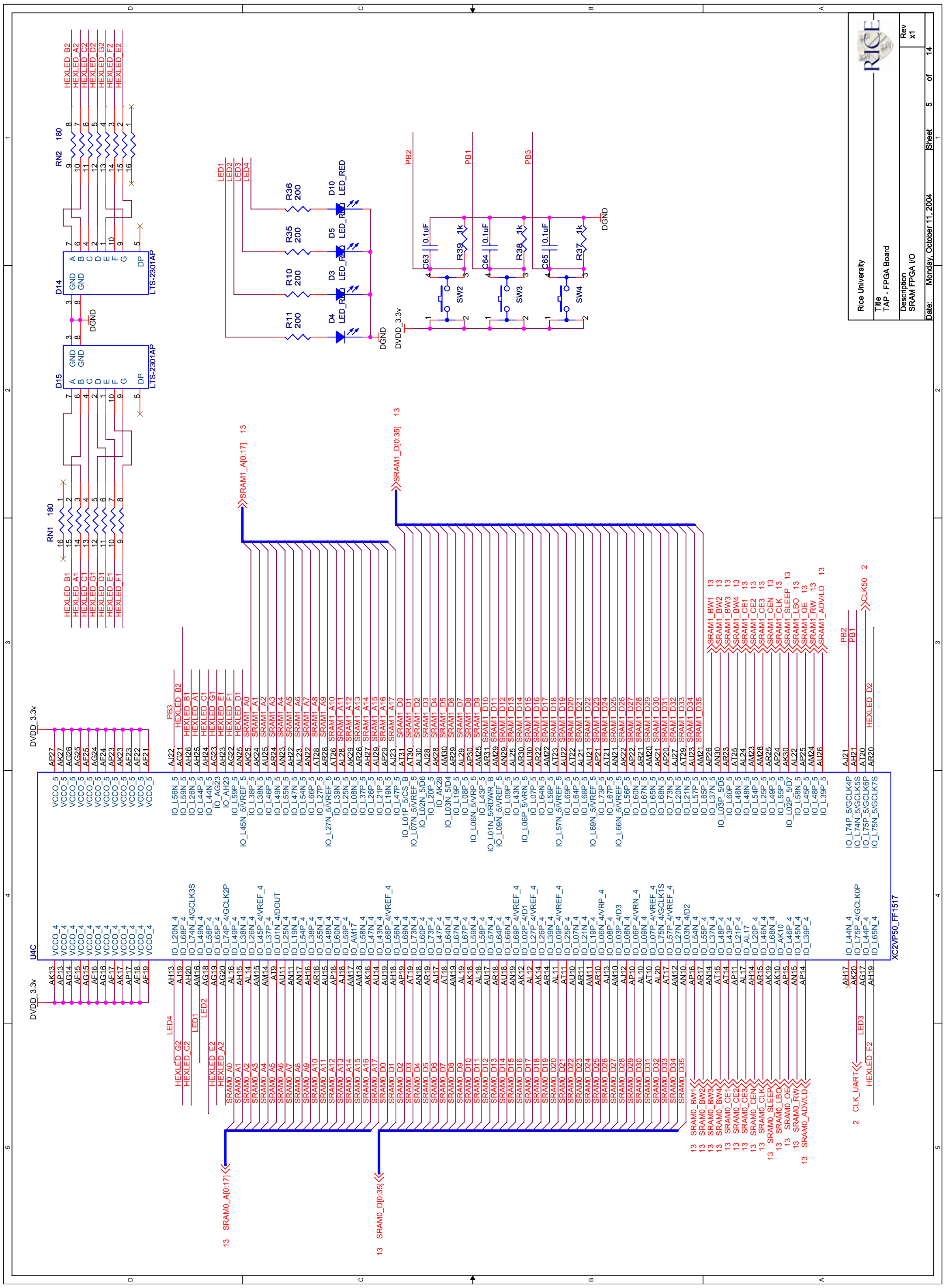
Date: Sunday, September 26, 2004 Sheet 1 of 14

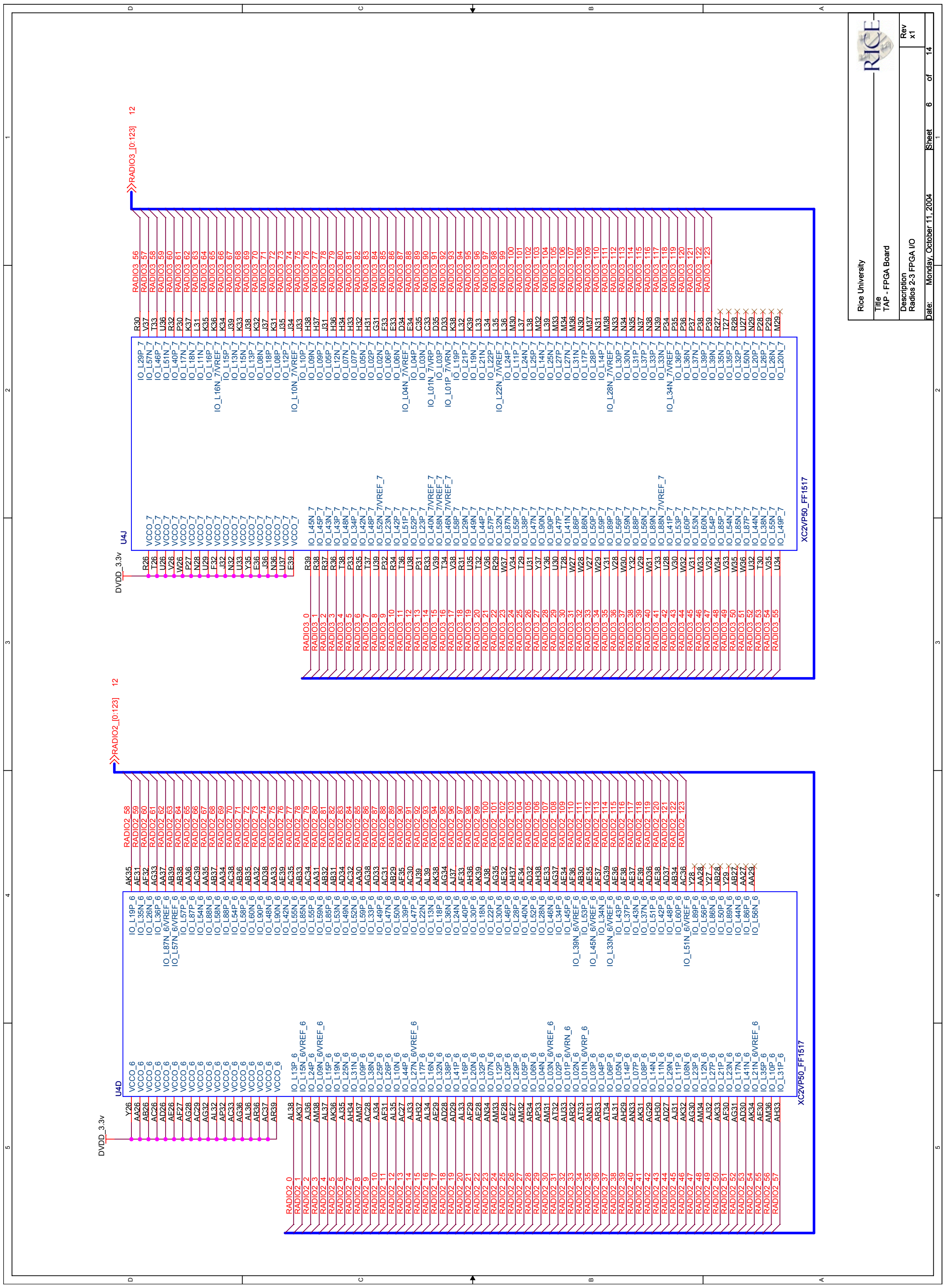


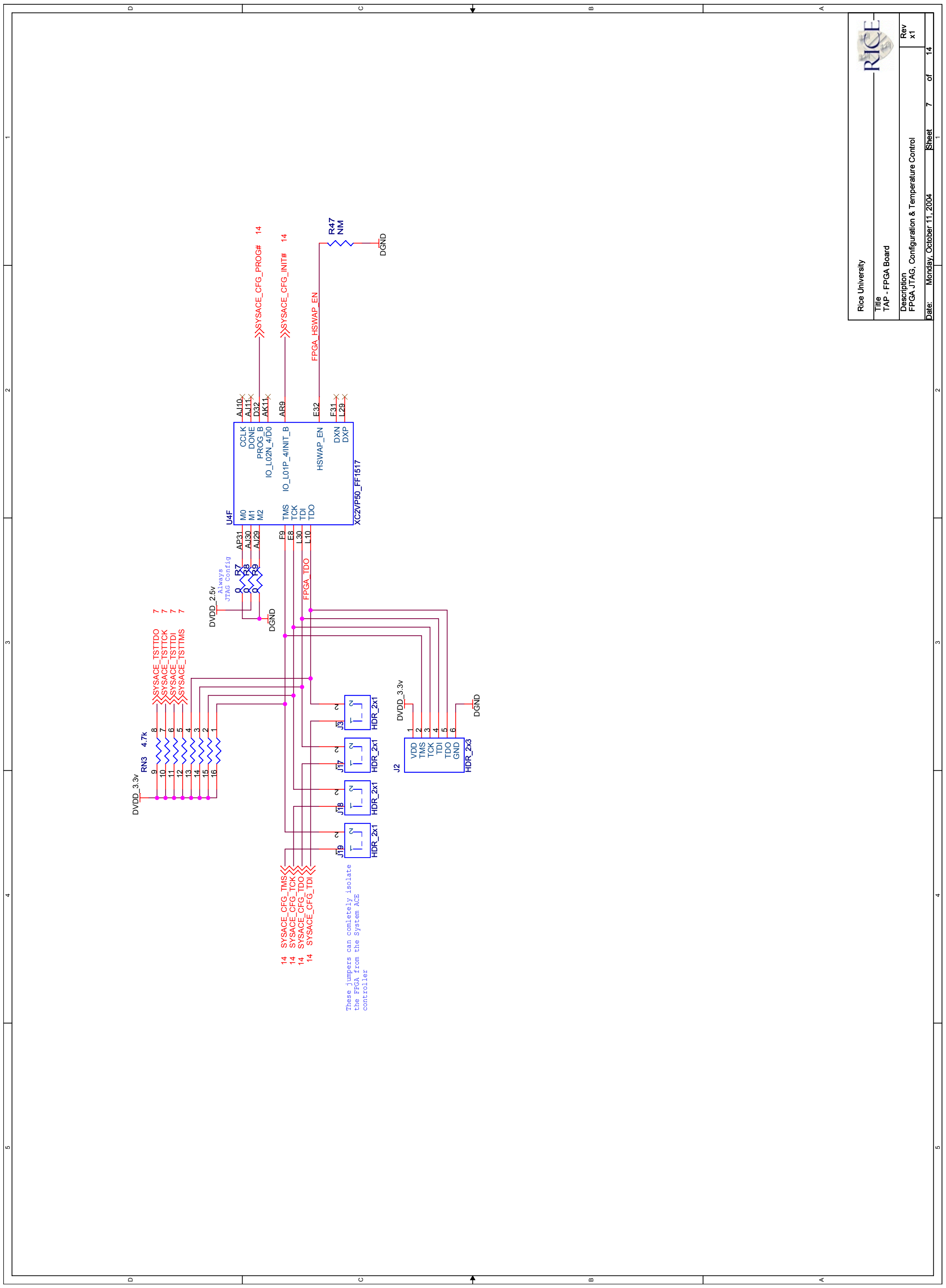
Rice University
Title
TAP - FPGA Board
Description
Clocks
Date: Monday, October 11, 2004
Sheet 2 of 14
Rev x1
125MHz MGT Reference Clock
Based on pg. 112 of Rocket I/O UG and Avnet V2Pro dev board schematics
DVDD_2.5v
DGND
U1
EG2121CA-125
VCC
OE
NC
GND
OUT
OUT
C1 10uF
C2 1nF
C3 0.1uF
R1 100
R2 100
R3 100
MGT_CLK0P 3
MGT_CLK0N 3
Top BREFCLK
100MHz MGT Reference Clock
U2
EG2121CA-100
C4 10uF
C5 1nF
C6 0.1uF
R4 100
R5 100
R6 100
MGT_CLK1P 3
MGT_CLK1N 3
Top BREFCLK2
System ACE CF 33MHz Clock
DVDD_3.3v
Y1
OSC-33
VDD
OE
GND
OUT
U3
ICS8302
VDDO
CLKIN
OUT0
OUT1
CLK33 3
SYSACE_CLK33 14
The FPGA needs a copy to drive the logic connected to the MPU interface on the SysACE CF controller (pg. 10 of the SysACE CF datasheet)
Y2
OSC-100
Y3
OSC-NM
CLK_UART 5
CLK50 5


U4A
XC2VP50_FF1517
DVDD_3.3v
SYSACE_MPD[0:15]
SYSACE_MPA[0:6]
CLK33
MGT_CLK1N
MGT_CLK1P
MGT_CLK0P
MGT_CLK0N
DEBUG[0:15]
J111
HDR_9x2
DGND
Rice University
Title
TAP - FPGA Board
Description
Misc FPGA I/O
Date: Monday, October 11, 2004
Sheet 3 of 14
Rev x1

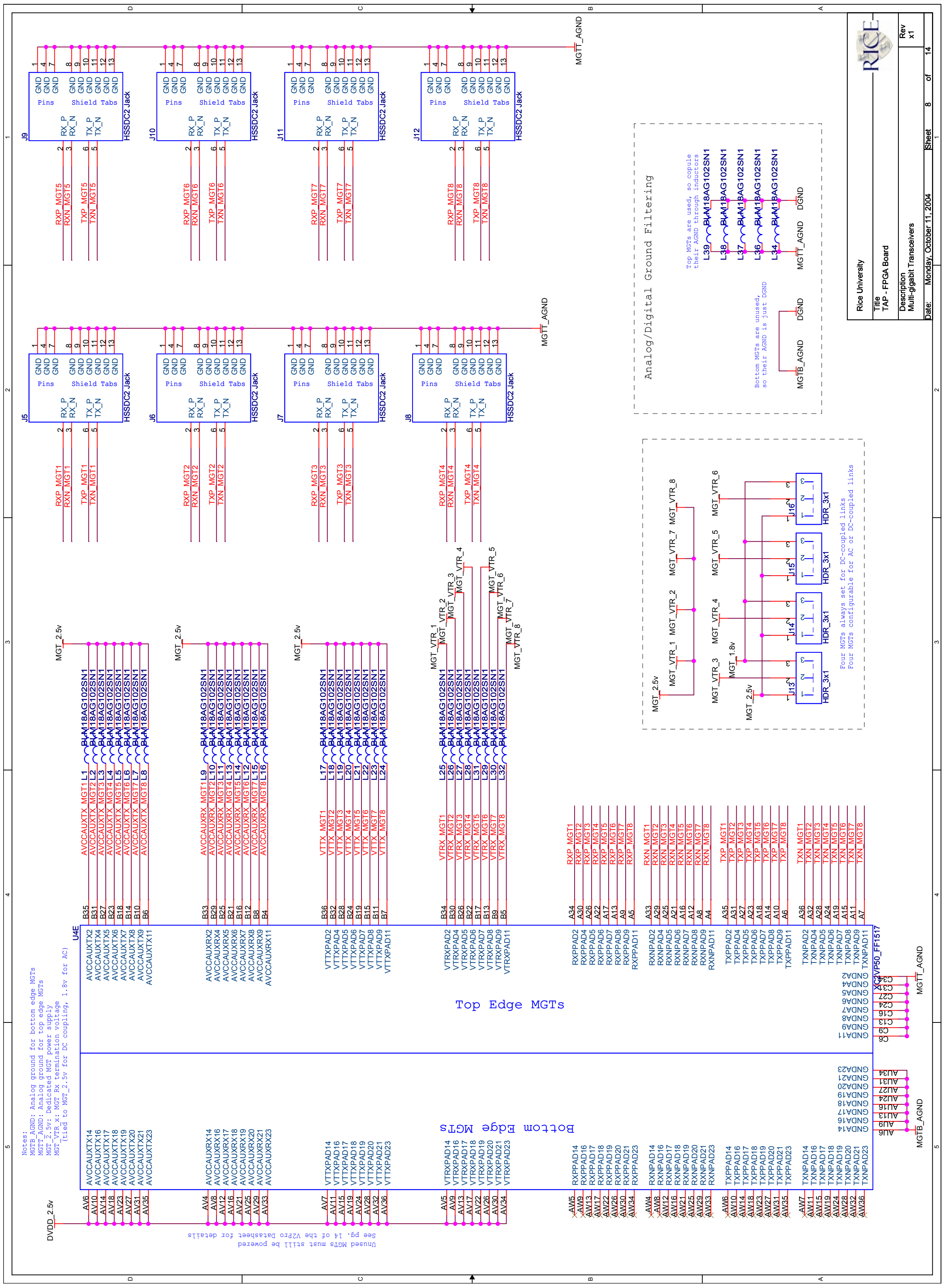

Rice University
Title
TAP - FPGA Board
Description
Radios 0-1 FPGA I/O
Date: Monday, October 11, 2004
Sheet 4 of 14
Rev x1
U4B
U4I
XC2VP50_FF1517
RADIO0_[0:123]
RADIO1_[0:123]
DVDD_3.3v

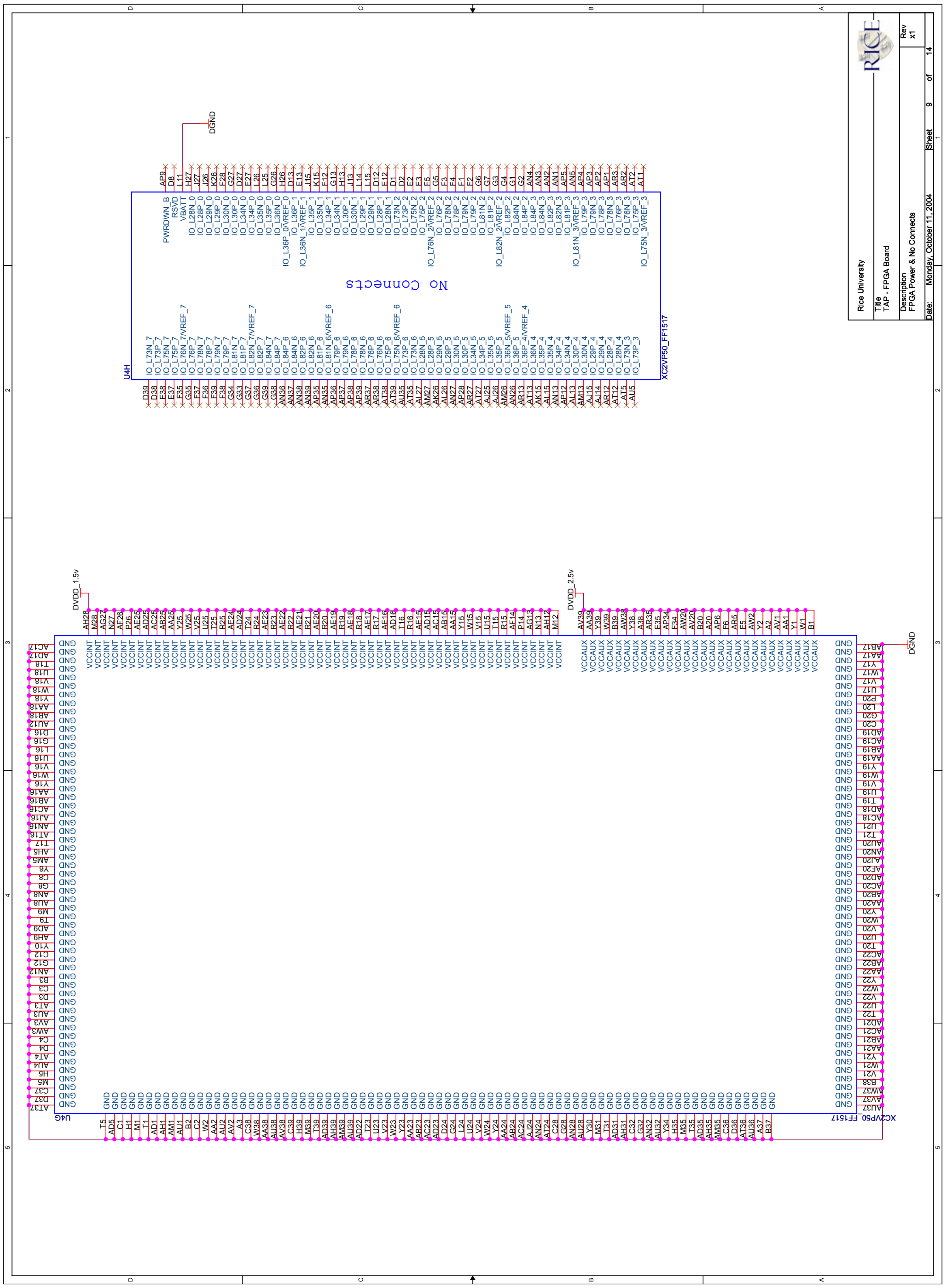

Rice University
Title
TAP - FPGA Board
Description
SRAM FPGA I/O
Date: Monday, October 11, 2004
Sheet 5 of 14
Rev
x1
U4C
XC2VP50_FF1517
DVDD_3.3v
DGND
D14
D15
LTS-2301AP
RN1 180
RN2 180
R36 200
R35 200
R10 200
R11 200
D10 LED_RED
D5
D3
D4
SW2
SW3
SW4
C63 0.1uF
C64 0.1uF
C65 0.1uF
R39 1k
R38 1k
R37 1k
PB1
PB2
PB3
LED1
LED2
LED3
LED4
SRAM1_A[0:17]
SRAM1_D[0:35]
SRAM0_A[0:17]
SRAM0_D[0:35]
CLK50
CLK_UART

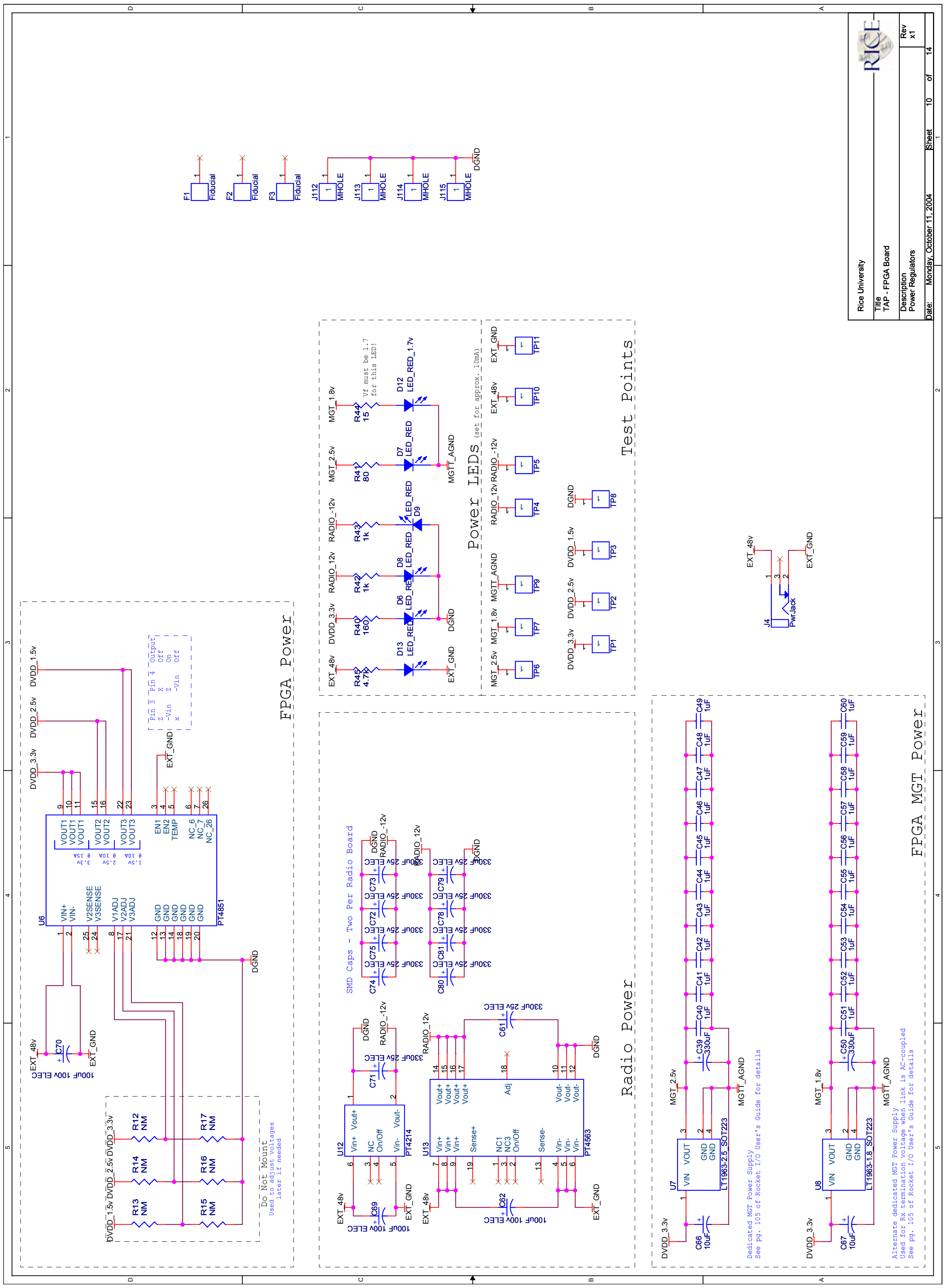

RADIO3_[0:123] 12
RADIO2_[0:123] 12
DVDD_3.3v
U4J
XC2VP50_FF1517
U4D
XC2VP50_FF1517
Rice University
Title
TAP - FPGA Board
Description
Radios 2-3 FPGA I/O
Date: Monday, October 11, 2004
Sheet 6 of 14
Rev x1

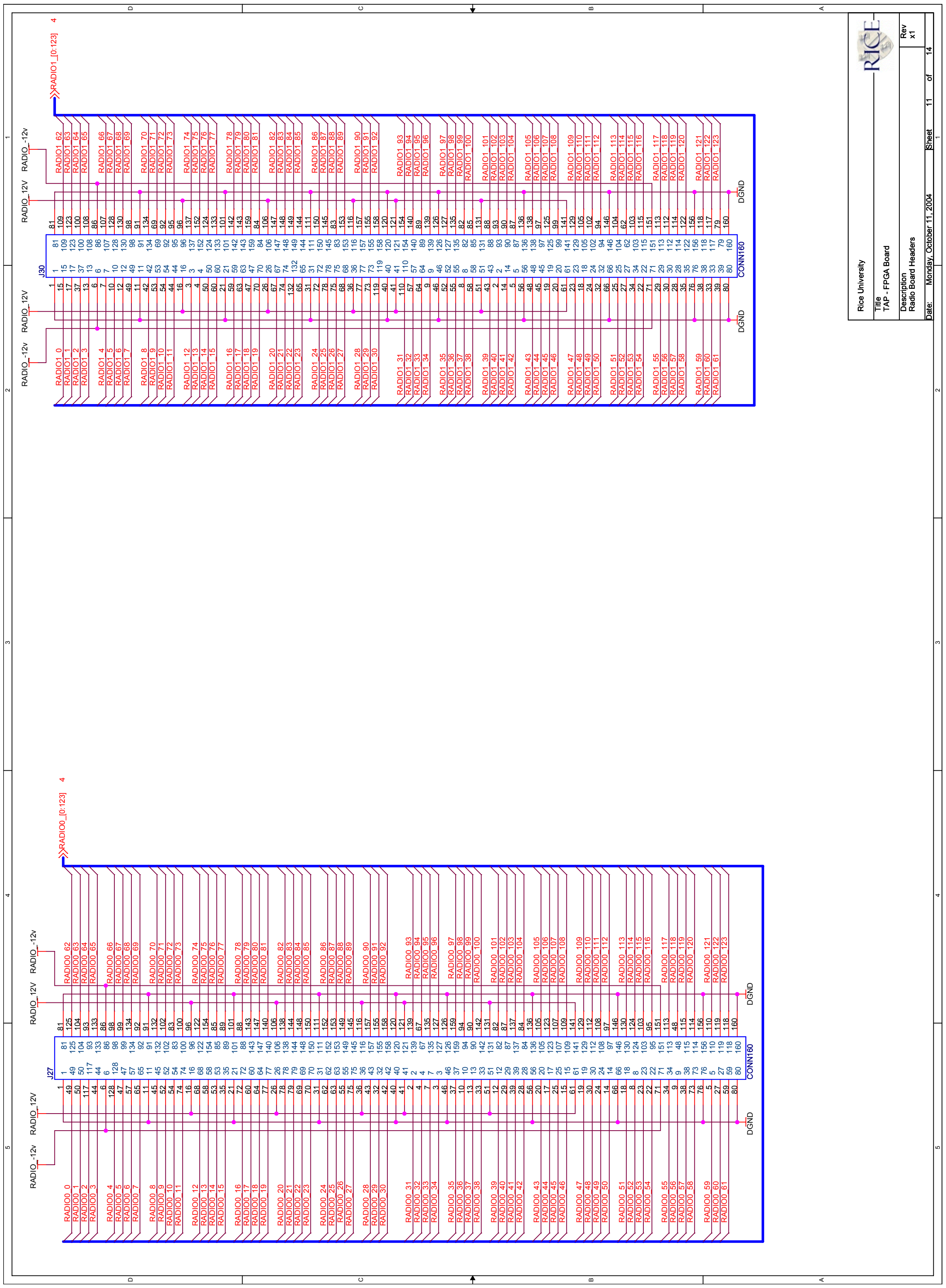


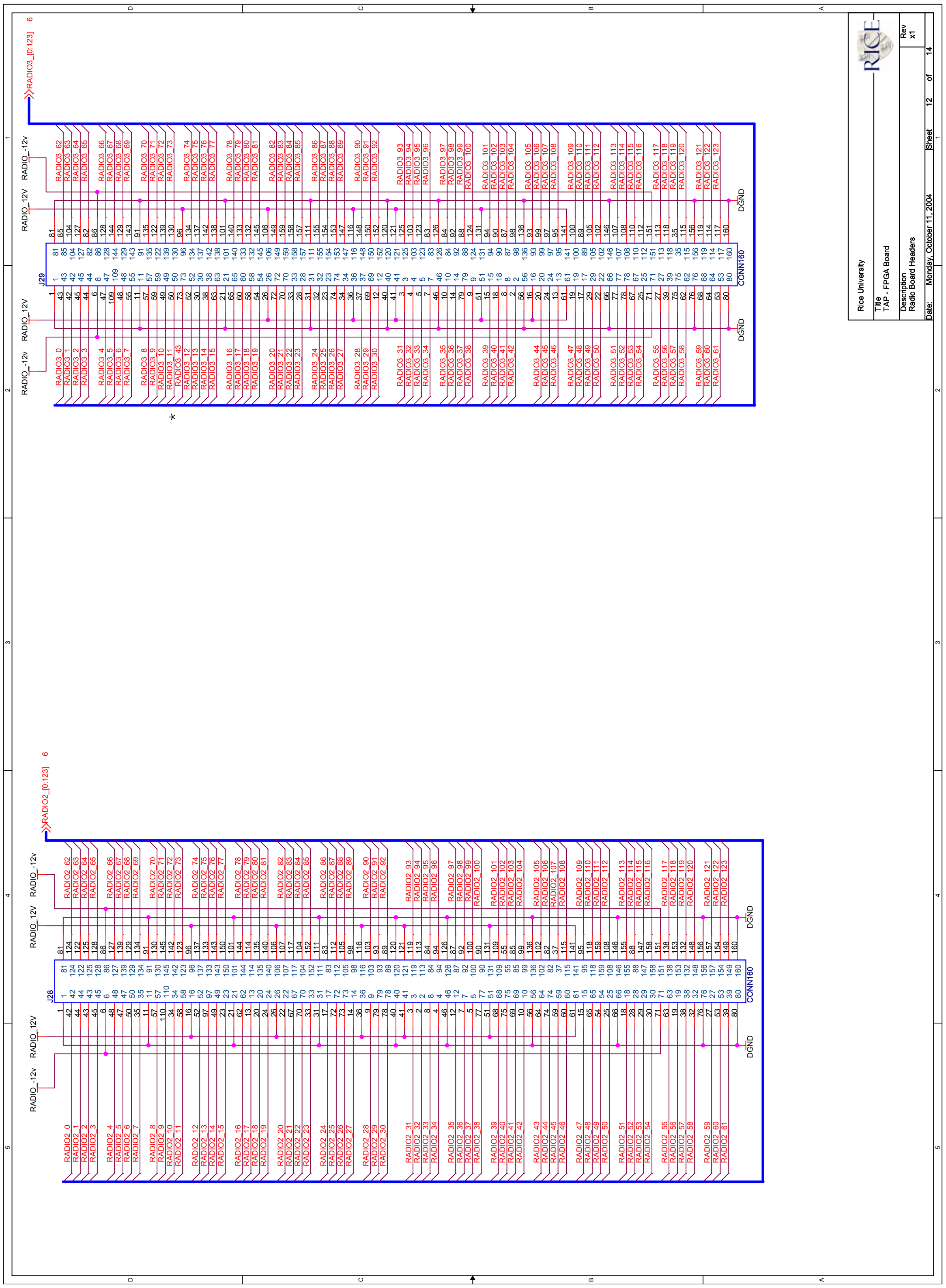
U4F
XC2VP50_FF1517
CCLK
DONE
PROG_B
IO_L02N_4/D0
IO_L01P_4/INIT_B
HSWAP_EN
DXN
DXP
M0
M1
M2
TMS
TCK
TDI
TDO
AJ10
AJ11
D32
AK11
AR9
E32
F31
L29
AP31
AJ30
AJ29
F9
E8
L30
L10
SYSACE_CFG_PROG# 14
SYSACE_CFG_INIT# 14
FPGA_HSWAP_EN
R47
NM
DGND
R7
R8
R9
0
DVDD_2.5v
Always JTAG Config
FPGA_TDO
SYSACE_TSTTDO 7
SYSACE_TSTTCK 7
SYSACE_TSTTDI 7
SYSACE_TSTTMS 7
RN3 4.7k
DVDD_3.3v
J3
J17
J18
J19
HDR_2x1
J2
HDR_2x3
VDD
TMS
TCK
TDI
TDO
GND
14 SYSACE_CFG_TMS
14 SYSACE_CFG_TCK
14 SYSACE_CFG_TDO
14 SYSACE_CFG_TDI
These jumpers can comletely isolate the FPGA from the System ACE controller
Rice University
RICE
Title
TAP - FPGA Board
Description
FPGA JTAG, Configuration & Temperature Control
Rev
x1
Date: Monday, October 11, 2004
Sheet 7 of 14

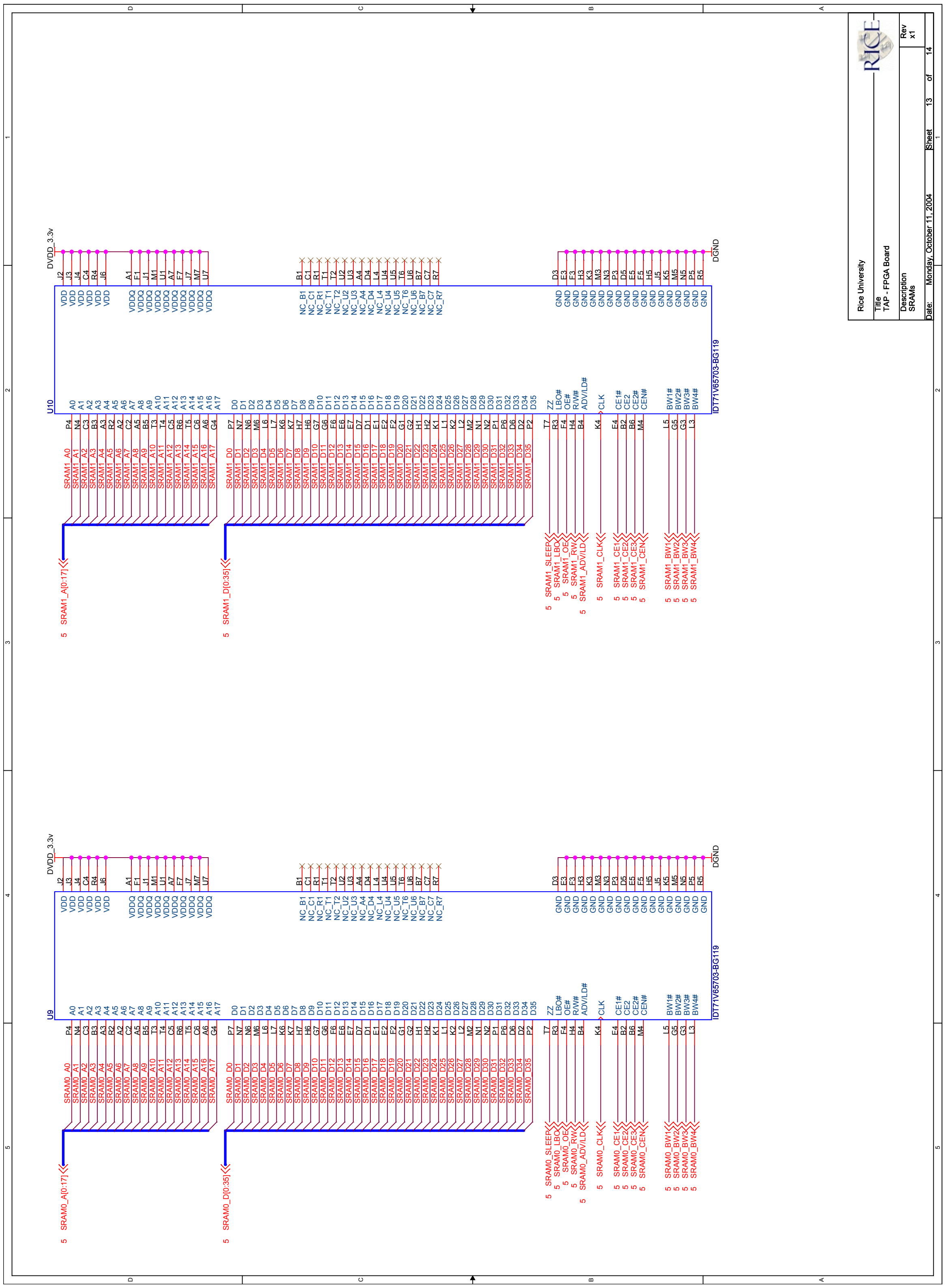

Rice University
Title
TAP - FPGA Board
Description
Multi-gigabit Transceivers
Date: Monday, October 11, 2004
Sheet 8 of 14
Rev x1
Top Edge MGTs
Bottom Edge MGTs
Analog/Digital Ground Filtering
MGTT_AGND
MGTB_AGND
DGND
MGT_2.5v
DVDD_2.5v
MGT_1.8v
HSSDC2 Jack
HDR_3x1
XC2VP50_FF1517
Notes:
MGTB_AGND: Analog ground for bottom edge MGTs
MGTT_AGND: Analog ground for top edge MGTs
MGT_2.5v: Dedicated MGT power supply
MGT_VTR_x: MGT Rx termination voltage
(tied to MGT_2.5v for DC coupling, 1.8v for AC)
Top MGTs are used, so copule their AGND through inductors
Bottom MGTs are unused, so their AGND is just DGND
Four MGTs always set for DC-coupled links
Four MGTs configurable for AC or DC-coupled links
Unused MGTs must still be powered
See pg. 14 of the V2Pro Dataset for details

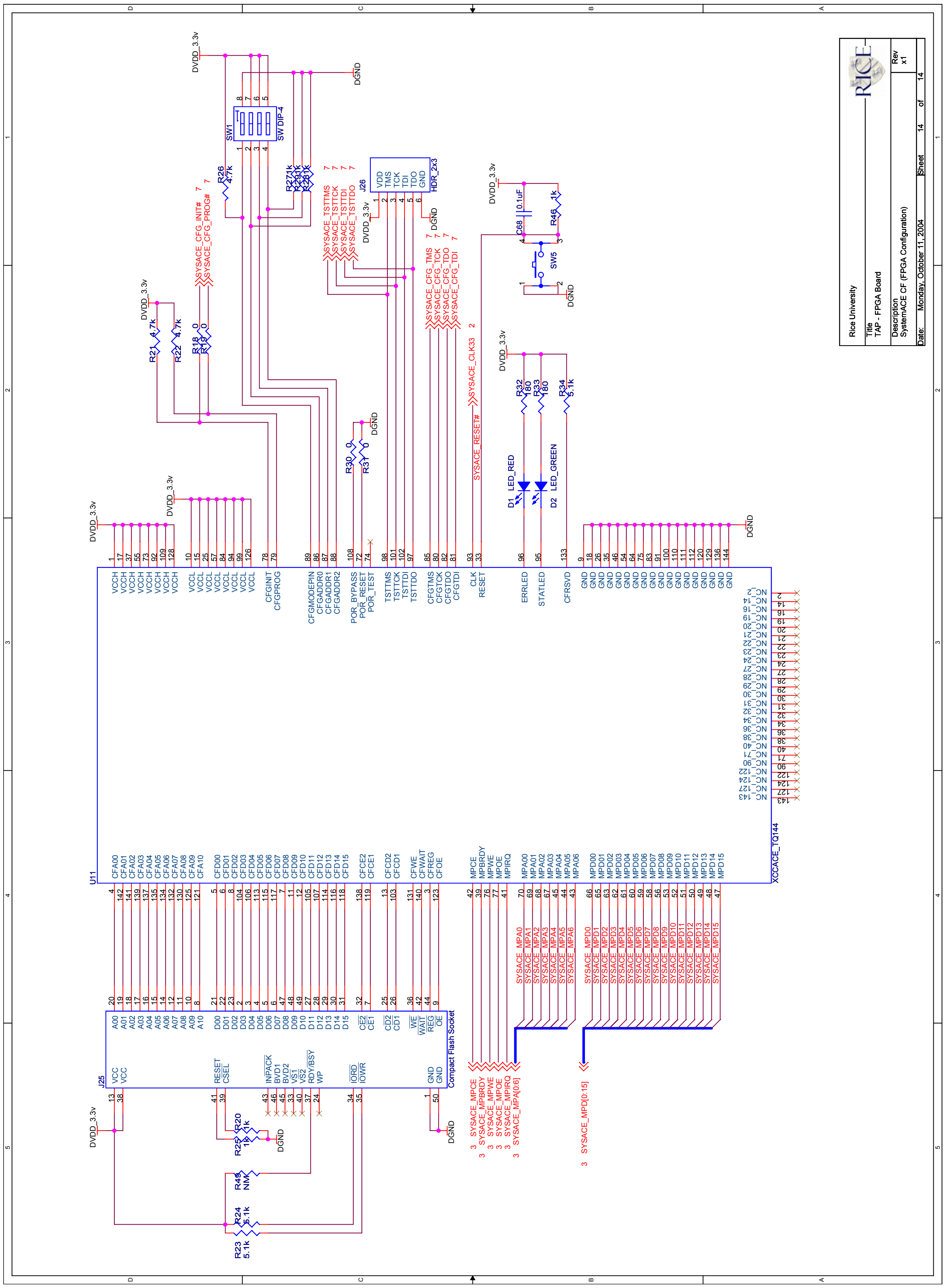

Rice University
Title
TAP - FPGA Board
Description
FPGA Power & No Connects
Date: Monday, October 11, 2004
Sheet 9 of 14
Rev
x1
No Connects
U4H
XC2VP50_FF1517
U4G
XC2VP50_FF1517
DVDD_1.5v
DVDD_2.5v
DGND
VCCINT
VCCAUX
GND
PWRDWN_B
RSVD
VBATT

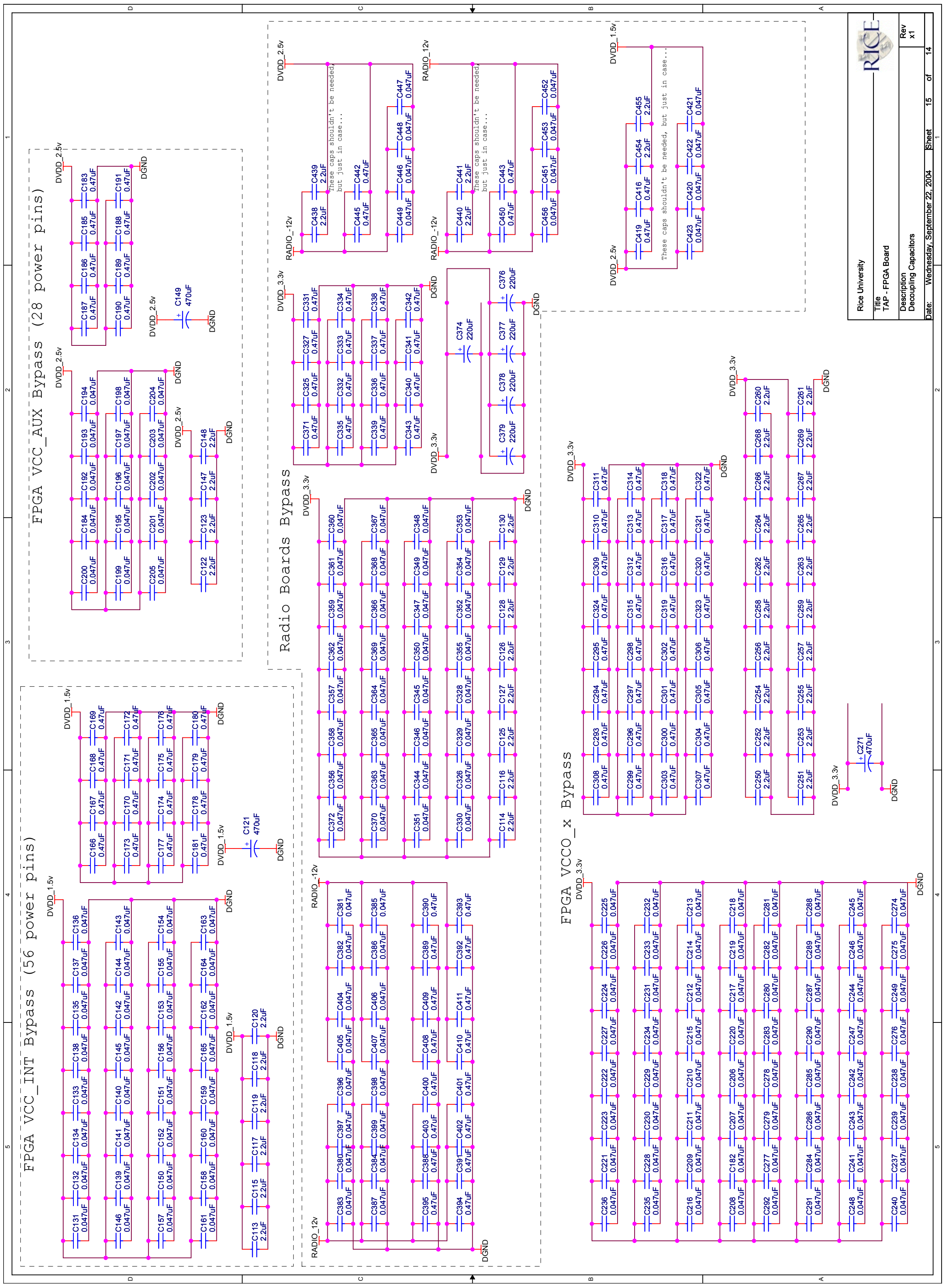

FPGA Power
Power LEDs
Test Points
Radio Power
FPGA MGT Power
SMD Caps - Two Per Radio Board
Do Not Mount Used to adjust voltages later if needed
Dedicated MGT Power Supply See pg. 105 of Rocket I/O User's Guide for details
Alternate dedicated MGT Power Supply Used for Rx termination voltage when link is AC-coupled See pg. 105 of Rocket I/O User's Guide for details
U6 PT4851
U12 PT4214
U13 PT4563
U7 LT1963-2.5_SOT223
U8 LT1963-1.8_SOT223
J4 PwrJack
Rice University
Title TAP - FPGA Board
Description Power Regulators
Date: Monday, October 11, 2004
Sheet 10 of 14
Rev x1


RADIO1_[0:123]
RADIO0_[0:123]
J30
J27
CONN160
RADIO_12V
RADIO_-12v
DGND
Rice University
Title
TAP - FPGA Board
Description
Radio Board Headers
Date: Monday, October 11, 2004
Sheet 11 of 14
Rev x1


RADIO3_[0:123]
RADIO2_[0:123]
RADIO_12V
RADIO_-12v
DGND
J29
J28
CONN160
Rice University
Title
TAP - FPGA Board
Description
Radio Board Headers
Date: Monday, October 11, 2004
Sheet 12 of 14
Rev x1

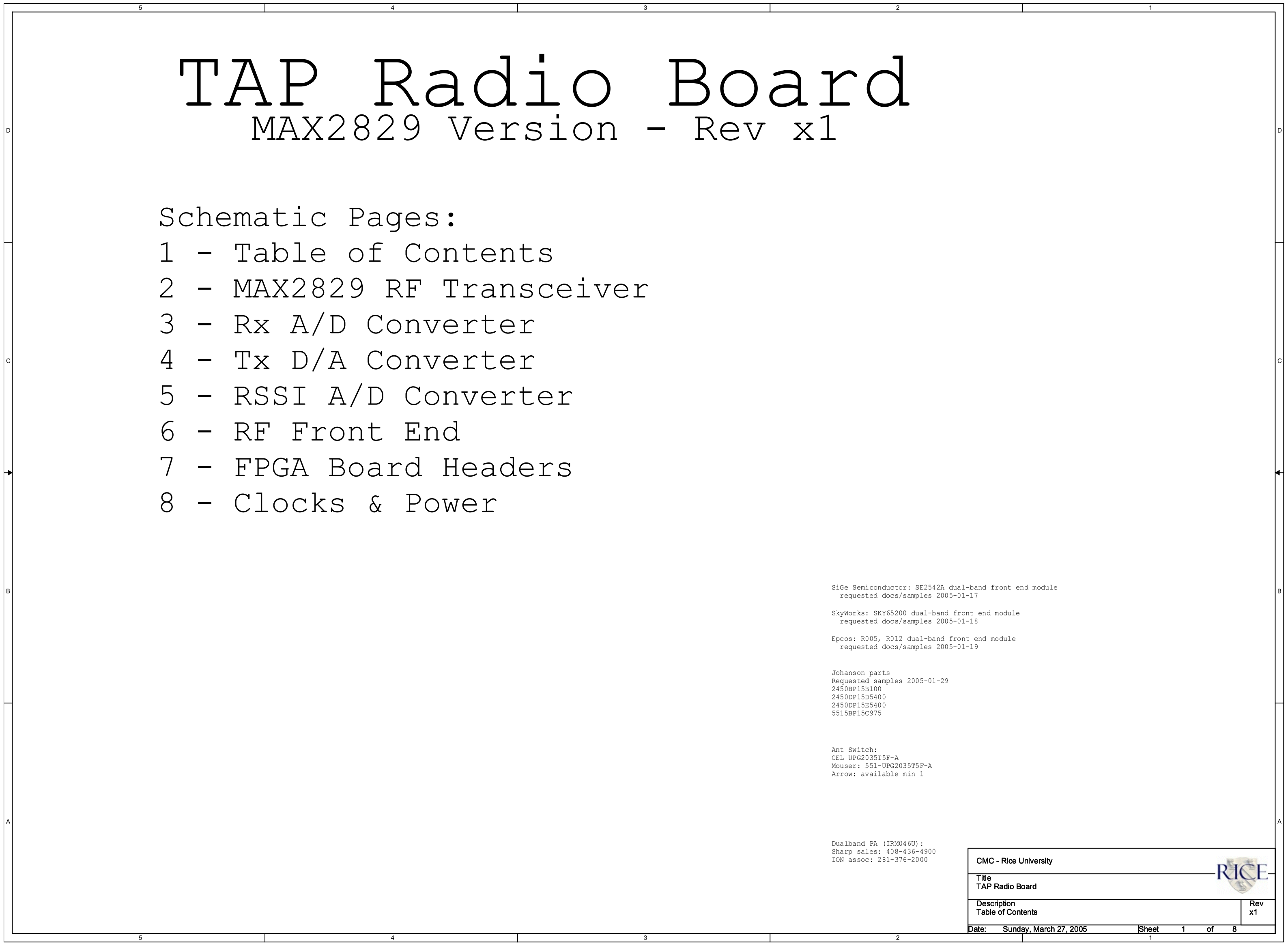

Rice University
Title
TAP - FPGA Board
Description
SRAMs
Date: Monday, October 11, 2004
Sheet 13 of 14
Rev
x1
U10
U9
IDT71V65703-BG119
DVDD_3.3v
DGND
SRAM1_A[0:17]
SRAM1_D[0:35]
SRAM1_SLEEP
SRAM1_LBO
SRAM1_OE
SRAM1_RW
SRAM1_ADV/LD
SRAM1_CLK
SRAM1_CE1
SRAM1_CE2
SRAM1_CE3
SRAM1_CEN
SRAM1_BW1
SRAM1_BW2
SRAM1_BW3
SRAM1_BW4
SRAM0_A[0:17]
SRAM0_D[0:35]
SRAM0_SLEEP
SRAM0_LBO
SRAM0_OE
SRAM0_RW
SRAM0_ADV/LD
SRAM0_CLK
SRAM0_CE1
SRAM0_CE2
SRAM0_CE3
SRAM0_CEN
SRAM0_BW1
SRAM0_BW2
SRAM0_BW3
SRAM0_BW4
ZZ
LBO#
OE#
R/W#
ADV/LD#
CLK
CE1#
CE2
CE2#
CEN#
BW1#
BW2#
BW3#
BW4#
VDD
VDDQ
GND


Rice University
Title
TAP - FPGA Board
Description
SystemACE CF (FPGA Configuration)
Date: Monday, October 11, 2004
Sheet 14 of 14
Rev x1
U11
XCCACE_TQ144
J25
Compact Flash Socket
SW1
SW DIP-4
J26
HDR_2x3
SW5
D1 LED_RED
D2 LED_GREEN
DVDD_3.3v
DGND


FPGA VCC_AUX Bypass (28 power pins)
FPGA VCC_INT Bypass (56 power pins)
Radio Boards Bypass
FPGA VCCO_x Bypass
These caps shouldn't be needed, but just in case...
DVDD_2.5v
DVDD_1.5v
DVDD_3.3v
RADIO_12v
RADIO_-12v
DGND
Rice University
Title
TAP - FPGA Board
Description
Decoupling Capacitors
Date: Wednesday, September 22, 2004
Sheet 15 of 14
Rev x1

A.2 TAP Radio Board Schematics

Following are the 8 pages of schematics for the TAP radio board.



TAP Radio Board

MAX2829 Version - Rev x1

Schematic Pages:

```
SiGe Semiconductor: SE2542A dual-band front end module
  requested docs/samples 2005-01-17

SkyWorks: SKY65200 dual-band front end module
  requested docs/samples 2005-01-18

Epcos: R005, R012 dual-band front end module
  requested docs/samples 2005-01-19


Johanson parts
Requested samples 2005-01-29
2450BP15B100
2450DP15D5400
2450DP15E5400
5515BP15C975


Ant Switch:
CEL UPG2035T5F-A
Mouser: 551-UPG2035T5F-A
Arrow: available min 1


Dualband PA (IRM046U):
Sharp sales: 408-436-4900
ION assoc: 281-376-2000
```

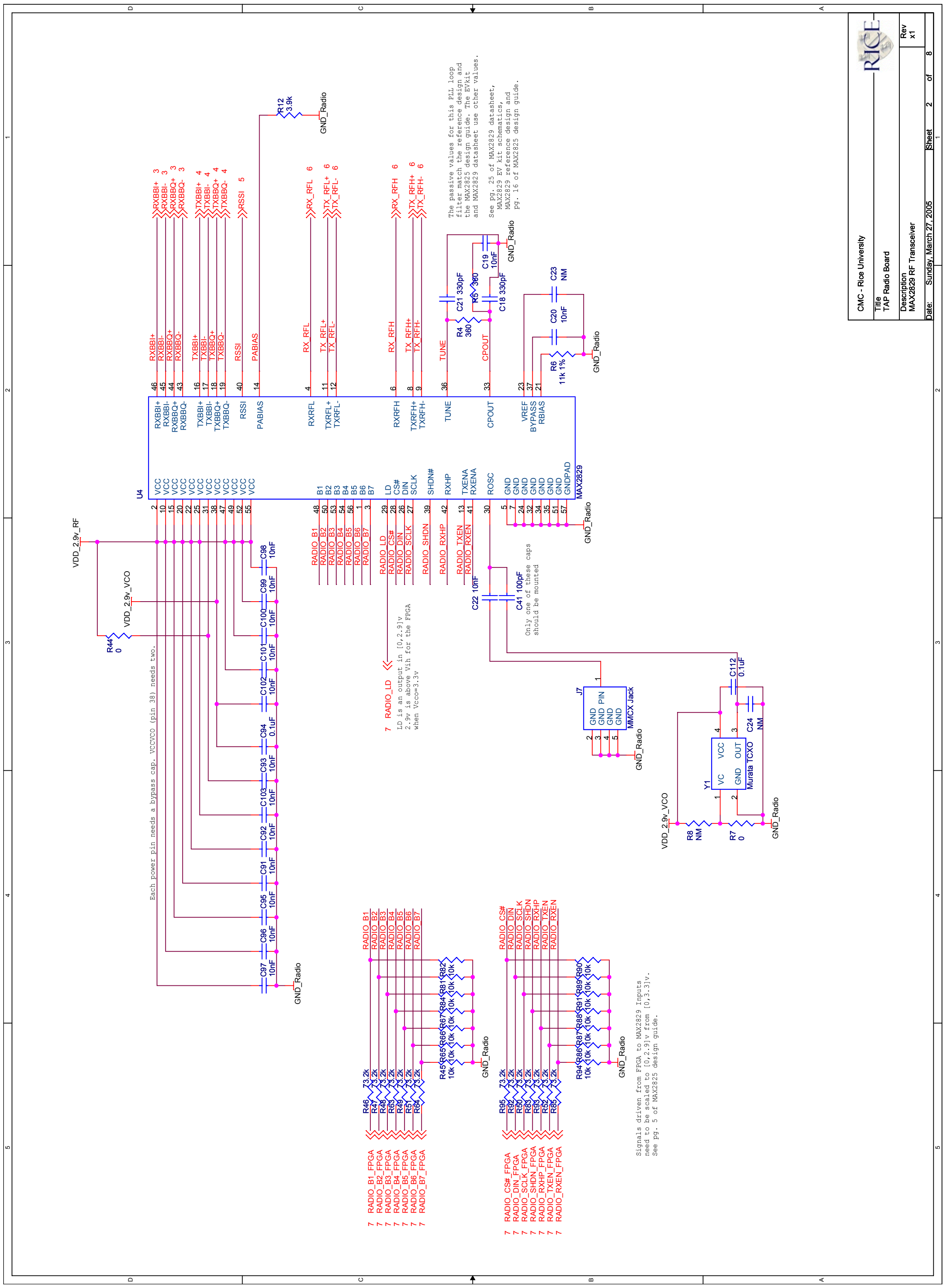
CMC - Rice University

Title
TAP Radio Board

Description
Table of Contents

Rev
x1

Date: Sunday, March 27, 2005 Sheet 1 of 8



The passive values for this PLL loop filter match the reference design and the MAX2825 design guide. The EVkit and MAX2829 datasheet use other values.

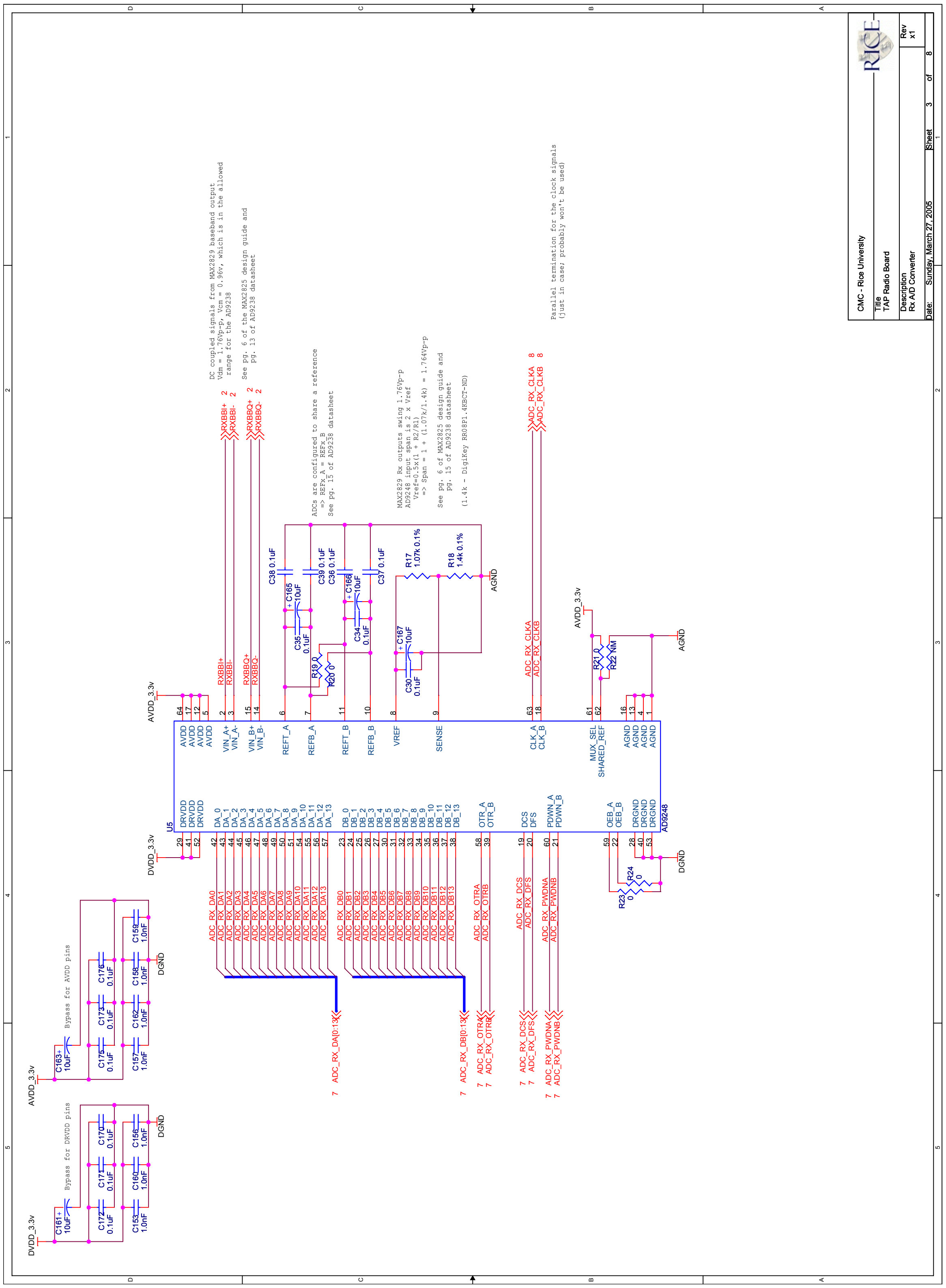
See pg. 25 of MAX2829 datasheet, MAX2829 EV kit schematics, MAX2829 reference design and pg. 16 of MAX2825 design guide.

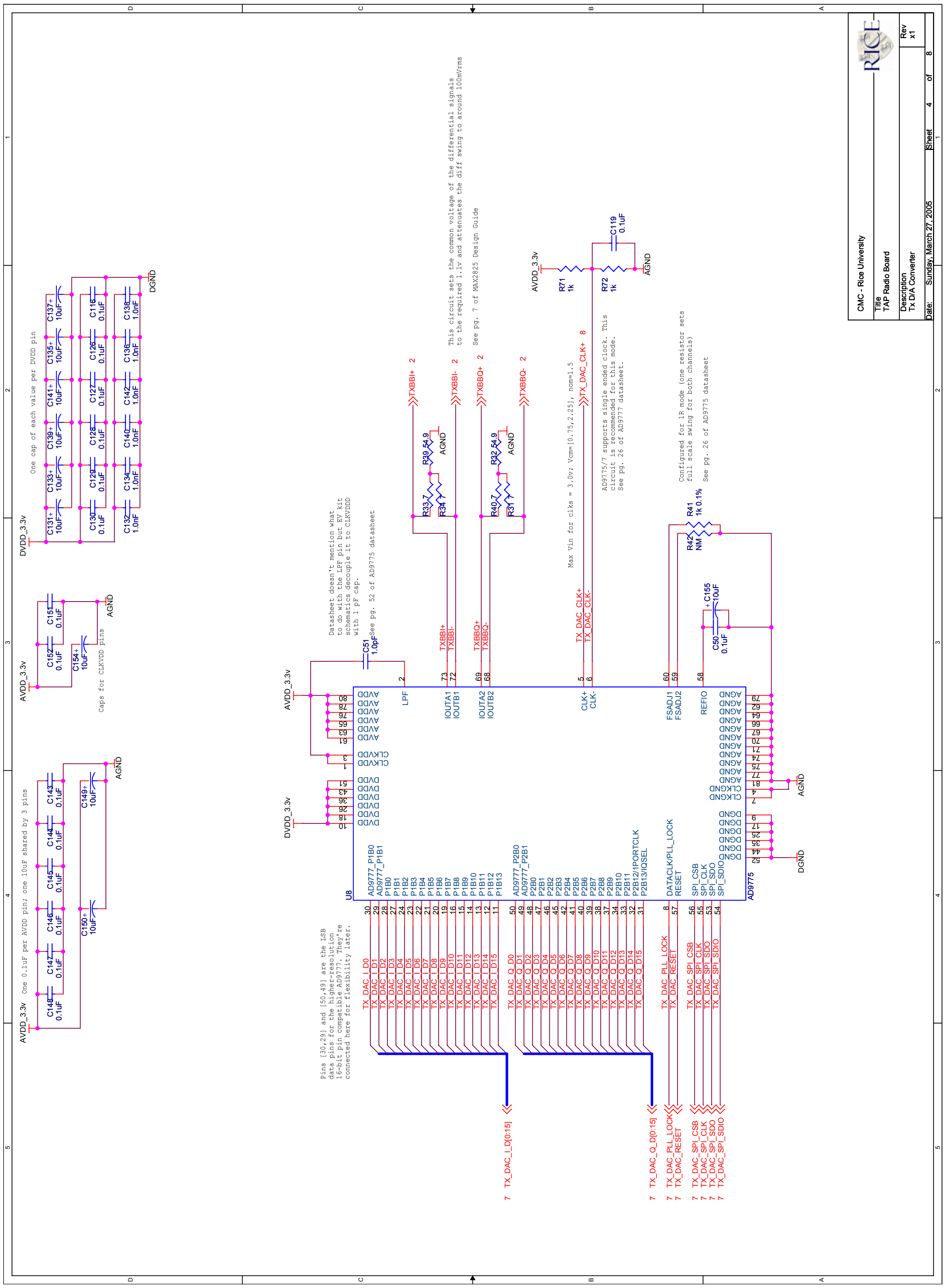
Only one of these caps should be mounted

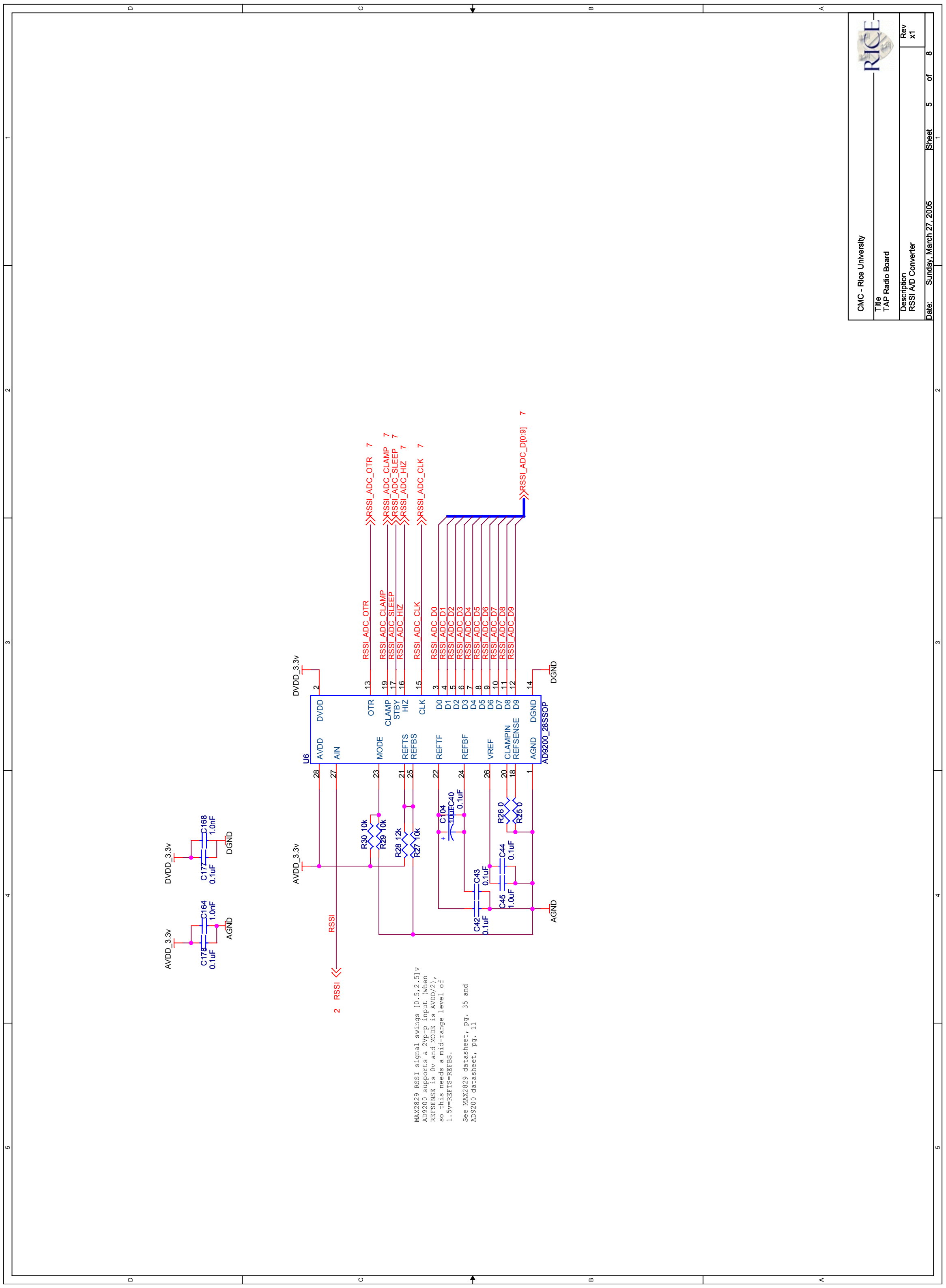
LD is an output in [0,2.9]v. 2.9v is above Vih for the FPGA when Vcco=3.3v

Each power pin needs a bypass cap. VCCVCO (pin 38) needs two.

Signals driven from FPGA to MAX2829 inputs need to be scaled to [0,2.9]v from [0,3.3]v. See pg. 5 of MAX2825 design guide.

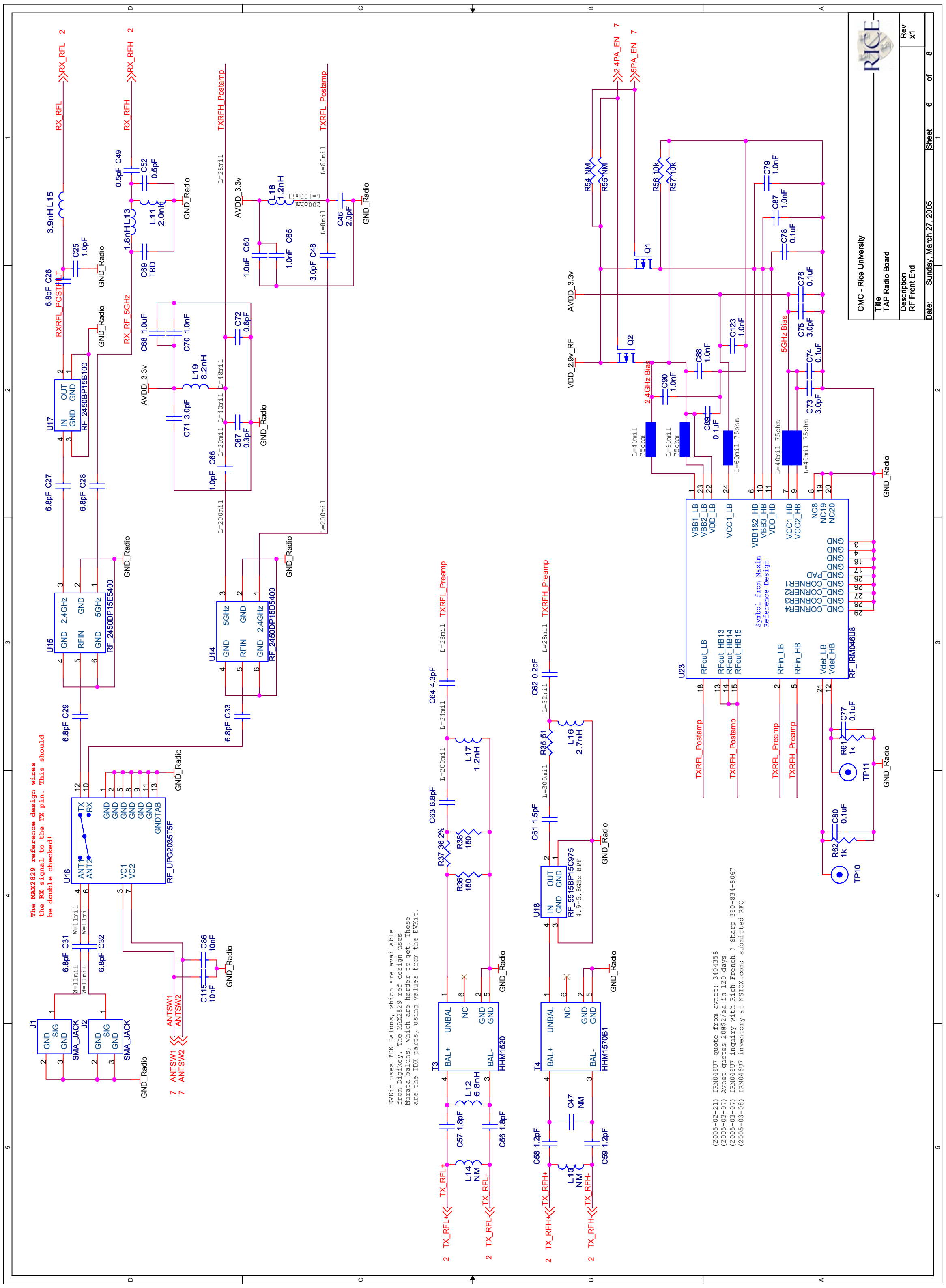

CMC - Rice University
Title
TAP Radio Board
Description
Rx A/D Converter
Date: Sunday, March 27, 2005
Sheet 3 of 8
Rev x1
U5
AD9248
DC coupled signals from MAX2829 baseband output
Vdm = 1.76Vp-p, Vcm = 0.96v, which is in the allowed range for the AD9238
See pg. 6 of the MAX2825 design guide and pg. 13 of AD9238 datasheet
ADCs are configured to share a reference => REFx_A = REFx_B
See pg. 15 of AD9238 datasheet
MAX2829 Rx outputs swing 1.76Vp-p
AD9248 input span is 2 x Vref
Vref=0.5x(1 + R2/R1)
=> Span = 1 + (1.07k/1.4k) = 1.764Vp-p
See pg. 6 of MAX2825 design guide and pg. 15 of AD9238 datasheet
(1.4k - DigiKey RR08P1.4KBCT-ND)
Parallel termination for the clock signals (just in case; probably won't be used)
Bypass for AVDD pins
Bypass for DRVDD pins
AVDD_3.3v
DVDD_3.3v
AGND
DGND
RXBBI+ 2
RXBBI- 2
RXBBQ+ 2
RXBBQ- 2
ADC_RX_CLKA 8
ADC_RX_CLKB 8
7 ADC_RX_DA[0:13]
7 ADC_RX_DB[0:13]
7 ADC_RX_OTRA
7 ADC_RX_OTRB
7 ADC_RX_DCS
7 ADC_RX_DFS
7 ADC_RX_PWDNA
7 ADC_RX_PWDNB
R17 1.07k 0.1%
R18 1.4k 0.1%
R19 0
R20 0
R21 0
R22 NM
R23 0
R24 0
C38 0.1uF
C39 0.1uF
C36 0.1uF
C37 0.1uF
C165 10uF
C166 10uF
C167 10uF
C35 0.1uF
C34 0.1uF
C30 0.1uF
C163 10uF
C161 10uF
C175 0.1uF
C173 0.1uF
C176 0.1uF
C172 0.1uF
C174 0.1uF
C170 0.1uF
C157 1.0nF
C162 1.0nF
C158 1.0nF
C159 1.0nF
C153 1.0nF
C160 1.0nF
C156 1.0nF
AVDD
DRVDD
VIN_A+
VIN_A-
VIN_B+
VIN_B-
REFT_A
REFB_A
REFT_B
REFB_B
VREF
SENSE
CLK_A
CLK_B
MUX_SEL
SHARED_REF
AGND
DA_0 … DA_13
DB_0 … DB_13
OTR_A
OTR_B
DCS
DFS
PDWN_A
PDWN_B
OEB_A
OEB_B
DRGND

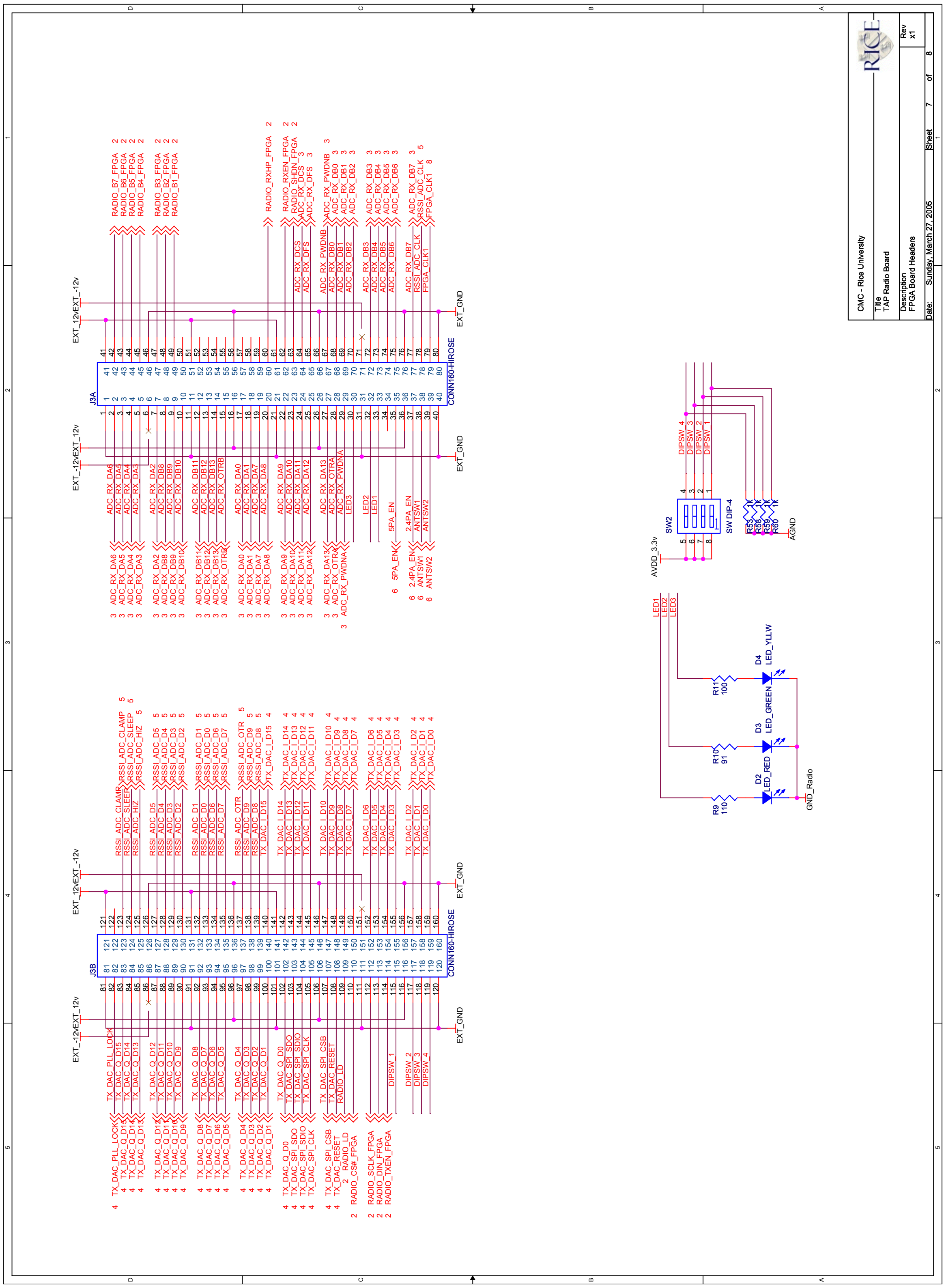

CMC - Rice University
Title
TAP Radio Board
Description
Tx D/A Converter
Date: Sunday, March 27, 2005
Sheet 4 of 8
Rev x1
U8
AD9775
One 0.1uF per AVDD pin; one 10uF shared by 3 pins
Caps for CLKVDD pins
One cap of each value per DVDD pin
Datasheet doesn't mention what to do with the LPF pin but EV kit schematics decouple it to CLKVDD with 1 pF cap.
See pg. 52 of AD9775 datasheet
This circuit sets the common voltage of the differential signals to the required 1.1v and attenuates the diff swing to around 100mVrms
See pg. 7 of MAX2825 Design Guide
Max Vin for clks = 3.0v; Vcm=[0.75,2.25], nom=1.5
AD9775/7 supports single ended clock. This circuit is recommended for this mode. See pg. 26 of AD9777 datasheet.
Configured for 1R mode (one resistor sets full scale swing for both channels)
See pg. 26 of AD9775 datasheet
Pins [30,29] and [50,49] are the LSB data pins for the higher-resolution 16-bit pin compatible AD9777. They're connected here for flexibility later.
AVDD_3.3v
DVDD_3.3v
AGND
DGND
TXBBI+
TXBBI-
TXBBQ+
TXBBQ-
TX_DAC_CLK+
TX_DAC_CLK-
TX_DAC_I_D[0:15]
TX_DAC_Q_D[0:15]
TX_DAC_PLL_LOCK
TX_DAC_RESET
TX_DAC_SPI_CSB
TX_DAC_SPI_CLK
TX_DAC_SPI_SDO
TX_DAC_SPI_SDIO



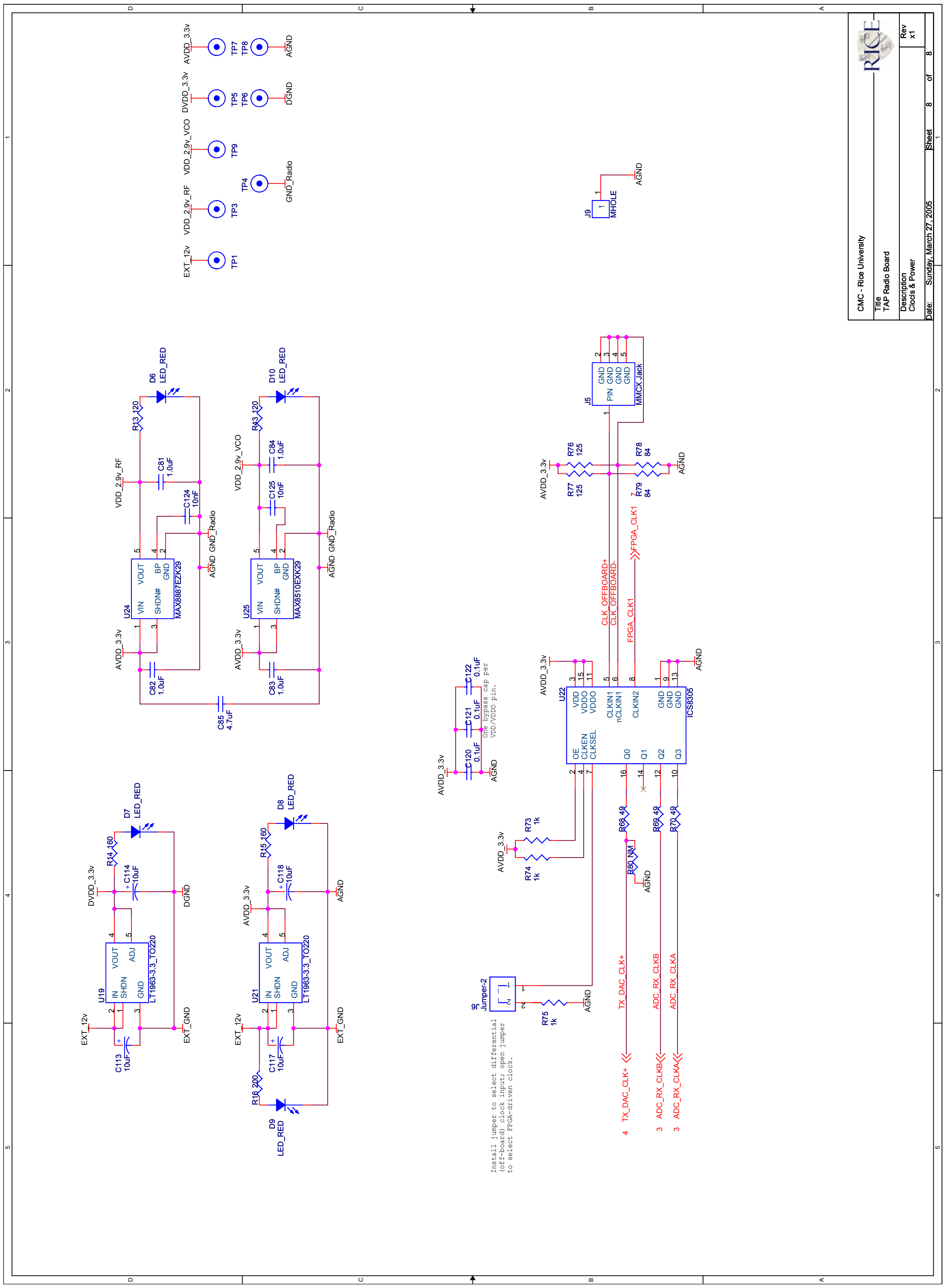
MAX2829 RSSI signal swings [0.5,2.5]v
AD9200 supports a 2Vp-p input (when
REFSENSE is 0v and MODE is AVDD/2),
so this needs a mid-range level of
1.5v=REFTS=REFBS.

See MAX2829 datasheet, pg. 35 and
AD9200 datasheet, pg. 11


CMC - Rice University
Title
TAP Radio Board
Description
RF Front End
Date: Sunday, March 27, 2005
Sheet 6 of 8
Rev x1
U23 RF_IRM046U8
Symbol from Maxim Reference Design
U16 RF_UPG2035T5F
U15 RF_2450DP15E5400
U14 RF_2450DP15D5400
U17 RF_2450BP15B100
U18 RF_5515BP15C975
T3 HHM1520
T4 HHM1570B1
J1 SMA_JACK
J2 SMA_JACK
The MAX2829 reference design wires the RX signal to the TX pin. This should be double checked!
EVKit uses TDK Baluns, which are available from Digikey. The MAX2829 ref design uses Murata baluns, which are harder to get. These are the TDK parts, using values from the EVKit.
(2005-02-21) IRM046U7 quote from avnet: 3404358
(2005-03-07) Avnet quotes 20$@2/ea in 120 days
(2005-03-07) IRM046U7 inquiry with Rich French @ Sharp 360-834-8067
(2005-03-08) IRM046U7 inventory at NSICX.com; submitted RFQ
RX_RFL
RX_RFH
TXRFH_Postamp
TXRFL_Postamp
TXRFL_Preamp
TXRFH_Preamp
2.4PA_EN
5PA_EN
AVDD_3.3v
VDD_2.9v_RF
2.4GHz Bias
5GHz Bias
GND_Radio
ANTSW1
ANTSW2
TX_RFL+
TX_RFL-
TX_RFH+
TX_RFH-
TP10
TP11



CMC - Rice University
Title
TAP Radio Board
Description
FPGA Board Headers
Date: Sunday, March 27, 2005
Sheet 7 of 8
Rev x1
J3A
J3B
CONN160-HIROSE
EXT_GND
EXT_-12v
EXT_12v
SW2
SW DIP-4
AVDD_3.3v
AGND
GND_Radio
D2 LED_RED
D3 LED_GREEN
D4 LED_YLLW
R9 110
R10 91
R11 100
LED1
LED2
LED3
DIPSW_1
DIPSW_2
DIPSW_3
DIPSW_4


CMC - Rice University
Title
TAP Radio Board
Description
Clocks & Power
Date: Sunday, March 27, 2005
Sheet 8 of 8
Rev x1
U22 ICS8305
U24 MAX8887EZK29
U25 MAX8510EXK29
U19 LT1963-3.3_TO220
U21 LT1963-3.3_TO220
J5 MMCX_Jack
J9 MHOLE
J8 Jumper-2
CLK_OFFBOARD+
CLK_OFFBOARD-
FPGA_CLK1
TX_DAC_CLK+
ADC_RX_CLKB
ADC_RX_CLKA
One bypass cap per VDD/VDDO pin.
Install jumper to select differential (off-board) clock input; open jumper to select FPGA-driven clock.
EXT_12v
VDD_2.9v_RF
VDD_2.9v_VCO
DVDD_3.3v
AVDD_3.3v
AGND
DGND
GND_Radio
EXT_GND

Appendix B
TAP Wireless Interface Board Users Guide

B.1 Board Overview

The TAP wireless interface board is a 8"x8" PCB built around a Xilinx XC2VP50 or XC2VP70 Virtex-II Pro FPGA. Figure B.1 shows a fully assembled board. The following sections detail the various components and configuration options on the board. For details on the design of the board, including details on features and possible applications, please see chapter 3.

B.1.1 SystemACE and FPGA Configuration

The TAP wireless interface board uses Xilinx's SystemACE CompactFlash solution for managing the configuration process of the FPGA. The SystemACE chip acts as an interface between the FPGA and a standard CompactFlash slot. The SystemACE automatically configures the FPGA on powerup or when the manual reset button (SW5) is pushed. Multiple configuration files can be stored on the CF card. The SystemACE chip reads a 3-bit address from a DIP switch (SW1) to choose one of eight configuration files. The fourth position on the DIP switch, labeled CFGMODE, should generally be set to 1 (slid to the right).

The SystemACE chip has two JTAG ports. The first, labeled TSTJTAG, is used to access both internal registers on the chip itself and the target FPGA. The second,

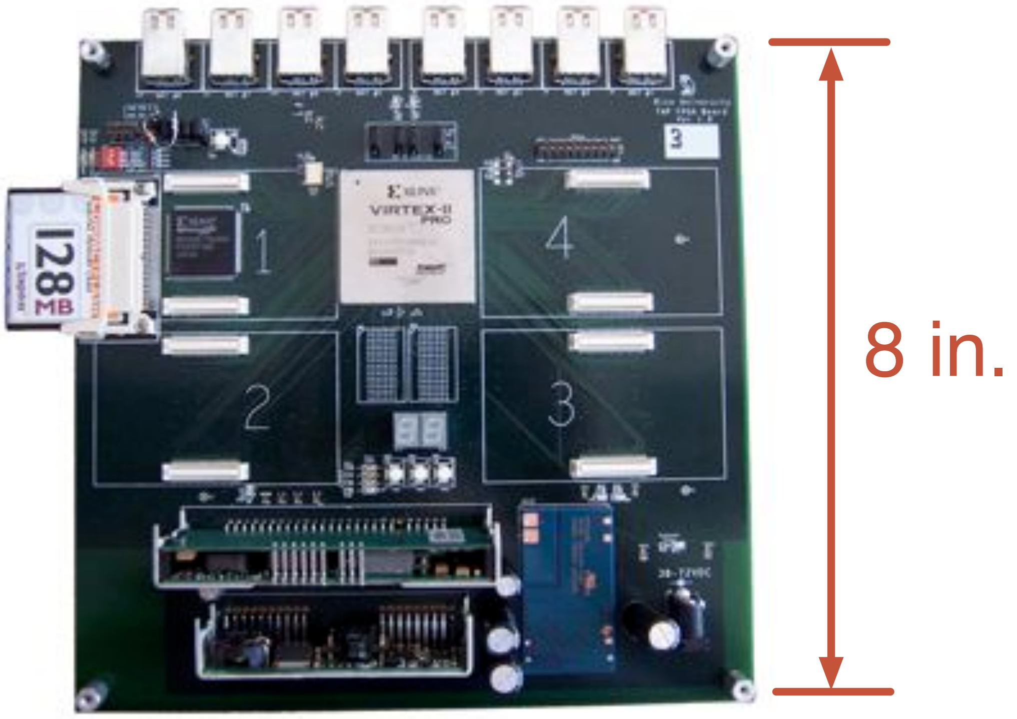


Figure B.1 TAP wireless interface board photo

labeled CFGJTAG, is the port used by the SystemACE to configure the target FPGA. Both ports are routed to 6-pin headers on the board, suitable for connection to an external programming cable. Adjacent to these headers are four jumpers. When mounted, these jumpers connect the FPGA to the SystemACE's CFGJTAG port. Removing these jumpers isolates the FPGA from the SystemACE controller, allowing use of the FPGA in cases where the SystemACE is not available or not desired.

Two LEDs are driven by the SystemACE controller to display configuration status and error state. The green LED blinks during configuration and glows steadily

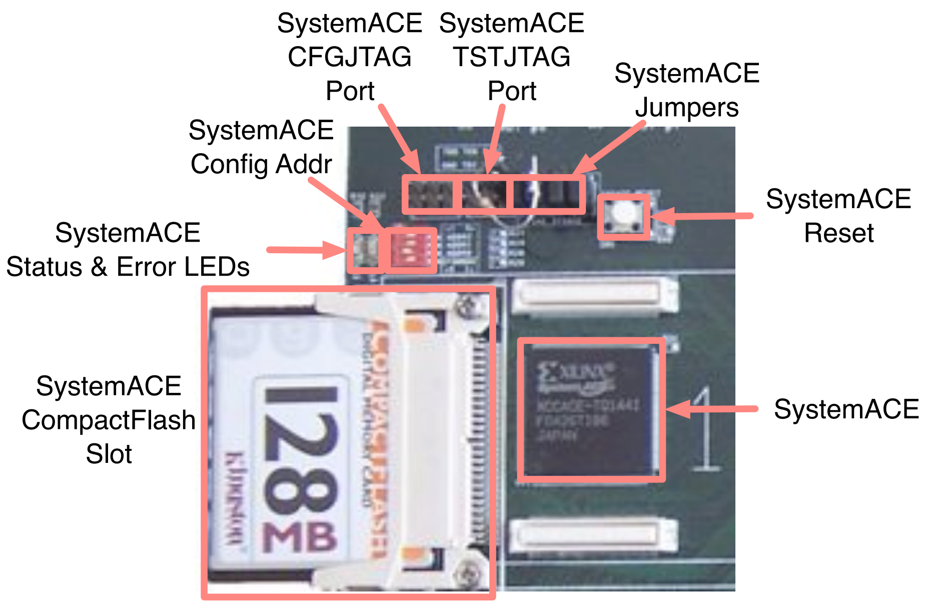


Figure B.2 SystemACE and FPGA configuration

afterwards. The red LED blinks when no CF card is found or when some other error occurs.

On the first revision of the board, a discrete pull-up resistor is required on the FPGA TDO signal. This is accomplished by connecting the jumper at J3 to 3.3 v through a 1 kΩ resistor. This resistor is shown in figure B.2.

B.1.2 Power

Three Texas Instruments PowerTrends power modules are used to regulate the voltages required on the board. One module supplies the three voltages required by the FPGA; the other two supply ±12 v for the daughtercard slots. The inputs of all three modules are tied to the power jack shown in figure B.3. This jack must be driven by an external DC voltage between 36 and 72 volts. The external regulator should be capable of supplying at least 20 watts. The jack accepts a standard barrel connector with a positive tip and grounded sleeve. There are two linear regulators included for powering the MGTs. These regulators are required to meet the tight noise tolerances of the MGT power supply pins.

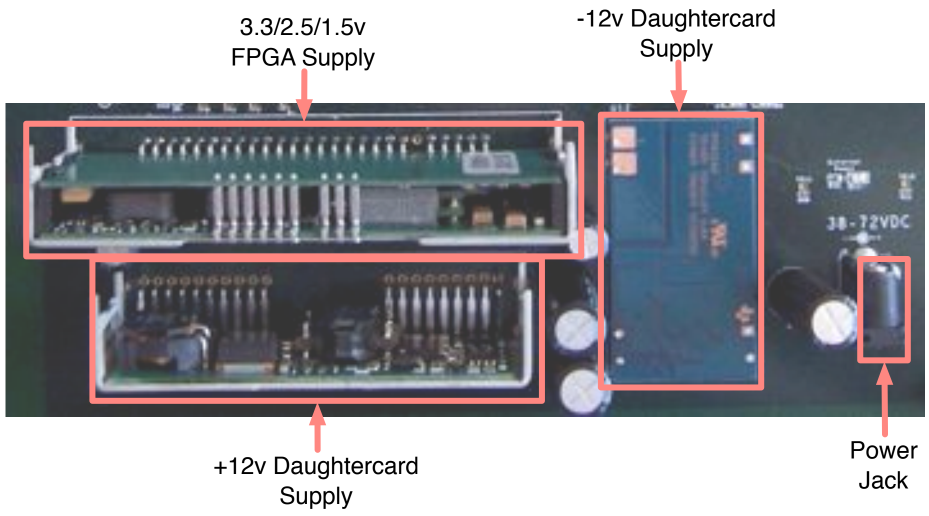


Figure B.3 External power and voltage regulators

The board includes test points and LEDs for every voltage plane in order to quickly identify any power problems. Each LED and test point is labeled on the board's silkscreen. Every power LED should illuminate when external power is connected. If any fails to do so, immediately disconnect external power to avoid damaging any improperly power components.

B.1.3 MGTs and Digital I/O

The eight off-board MGTs are routed to HSSDC2 jacks along the top of the board. The jacks are labeled on the board as “MGT #1” though “MGT #8”. Depending on the speed grade of FPGA mounted, each MGT can communicate at up to 3.125 Gb/s. Most builds of the board will use the slowest speed grade (-5), limiting each MGT to 2.5 Gb/s.

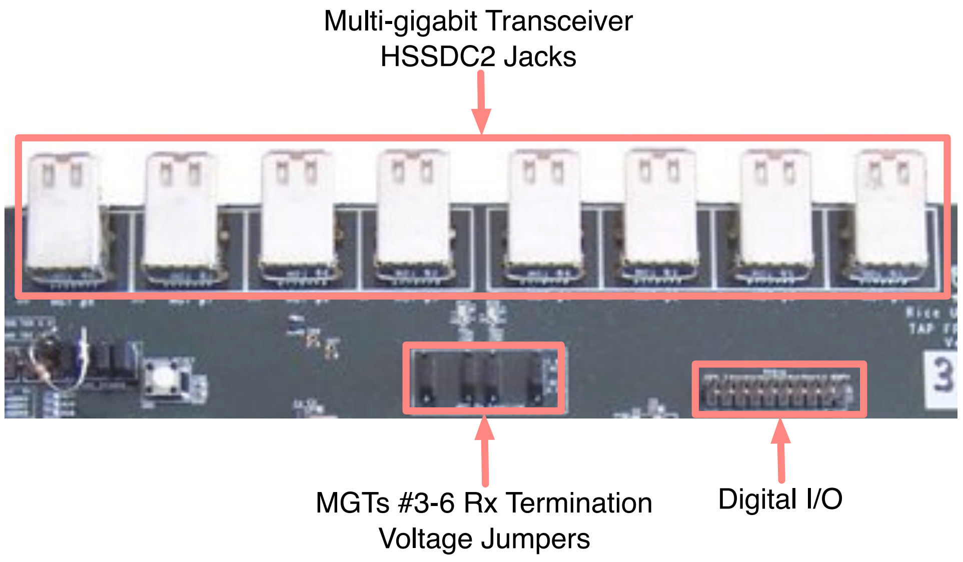


Figure B.4 MGTs and digital I/O

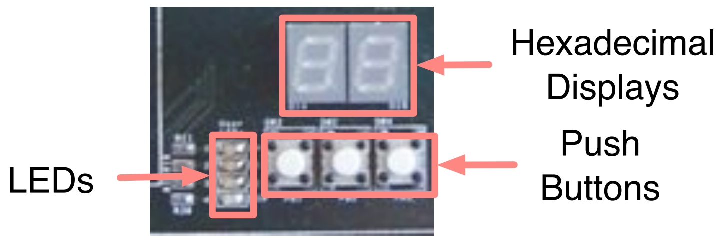
MGTs are designed for either DC-coupled or AC-coupled connects. In the case of DC-coupling, an MGT’s Tx and Rx termination voltages are both tied to 2.5 v. For AC-coupling, the Rx termination voltage must be 1.8 v. On the TAP wireless interface board, MGTs 1, 2, 7 and 8 are configured only for DC-coupled connections to other TAP boards. MGTs 3-6 can be individually configured for AC- or DC-coupling, allowing their use with other non-TAP hardware. The termination voltages are set by the jumpers shown in figure B.4, labeled J13-J16 on the board. A jumper **must** be installed for every MGT, regardless of whether it is being used. Mount a jumper on the upper pair of pins for AC-coupling (VTRx=1.8 v). Otherwise mount a jumper on the lower pair of pins for DC-coupling (VTRx=2.5 v). The board has labels printed in silkscreen to aid in choosing jumper positions.

The board also includes 16 bits of digital I/O routed to 0.1" header (J111). The header has four ground pins, tied to the FPGA's DGND, at either end of each row of pins. This header is routed directly to FPGA pins; the direction of each pin should be set in the FPGA configuration. The user must be careful to avoid drive fights between external sources and FPGA pins configured as outputs. See table B.2 for the FPGA pin assignments for these signals.

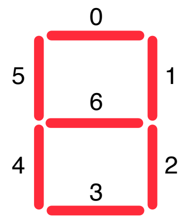
B.1.4 User I/O

The board includes a number of interactive I/O features intended to speed algorithm development debugging in hardware. Four LEDs, three push buttons and two hexadecimal seven-segment displays are provided. All are routed to dedicated FPGA pins. The push buttons include a RC circuit to debounce the mechanical switch. The hex displays use the signal mapping shown in figure B.5(b). The FPGA pin assignments for these components are listed in table B.2.

The hex display segments and LEDs illuminate when the corresponding FPGA



(a) User I/O Components



(b) Hex Display Signal Assignments

Figure B.5 User I/O

pins are driven high. The buttons normally pull the FPGA inputs low; pushing a button drives the corresponding FPGA input high. The Verilog code below can be used to map a 4-bit value to a 7-bit value which displays the corresponding hexadecimal value on a display. This code is adapted from a synthesis template provided in Xilinx ISE.

```
output  [6:0] hexDisplay;
reg     [6:0] hexDisplay;

wire    [3:0] fourBitInput;

always @(fourBitInput[3:0])
case (fourBitInput[3:0])
    4'b0001 : hexDisplay = ~(7'b1111001);   // 1
    4'b0010 : hexDisplay = ~(7'b0100100);   // 2
    4'b0011 : hexDisplay = ~(7'b0110000);   // 3
    4'b0100 : hexDisplay = ~(7'b0011001);   // 4
    4'b0101 : hexDisplay = ~(7'b0010010);   // 5
    4'b0110 : hexDisplay = ~(7'b0000010);   // 6
    4'b0111 : hexDisplay = ~(7'b1111000);   // 7
    4'b1000 : hexDisplay = ~(7'b0000000);   // 8
    4'b1001 : hexDisplay = ~(7'b0010000);   // 9
    4'b1010 : hexDisplay = ~(7'b0001000);   // A
    4'b1011 : hexDisplay = ~(7'b0000011);   // b
    4'b1100 : hexDisplay = ~(7'b1000110);   // C
    4'b1101 : hexDisplay = ~(7'b0100001);   // d
```

```
        4'b1110 : hexDisplay = ~(7'b0000110);   // E
        4'b1111 : hexDisplay = ~(7'b0001110);   // F
        default : hexDisplay = ~(7'b1000000);   // 0
    endcase
```

B.1.5 Daughtercard Slots

The four daughtercards slots each consist of two 80-pin connectors. Of the 160 pins, 124 are digital signals routed to dedicated pins on the FPGA. Each slot is routed to a different FPGA I/O bank, facilitating efficient allocation of logic resources. Table B.1 lists the pin assignments for the 124 digital signals for each daughtercard slot. The first column is the pin number of the daughtercard connector. Power and ground pins are included in this list; the corresponding FPGA pins are listed as ”-”. This table also includes the signals routed to each daughtercard pin on the TAP radio board.

The four slots are functionally identical, providing the same power and digital I/O. In the first revision, however, slot #1 can only accommodate daughtercards with no components mounted on the bottom in the area beyond the connectors. This is due to the height of the CompactFlash slot which sits beneath the daughtercard mounted in slot #1. The CF slot will be moved to the bottom of the wireless interface board in a future revision, eliminating this constraint.

Each daughtercard slot uses two 80-pin connectors from Hirose. The wireless interface board uses a receptacle with reinforced metal tabs for mechanical stability.

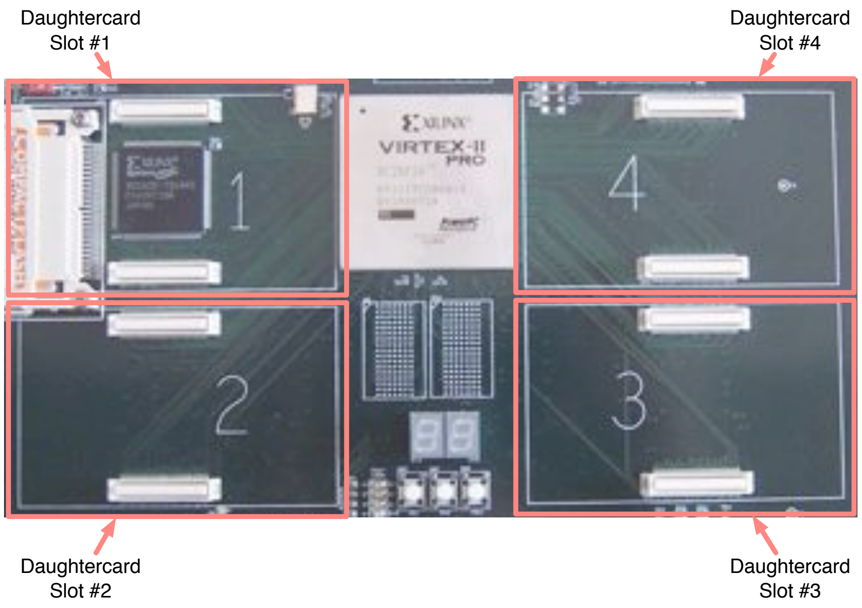


Figure B.6 Daughtercard slots

The daughtercard uses the matching header. Both receptacle and header are 4 mm tall. The receptacle is Hirose part number DF17(4.0)-80DS-0.5V(51) (Digikey part number H2379CT-ND). The header is Hirose part DF17(4.0)-80DP-0.5V(51) (Digikey part H2407CT-ND).

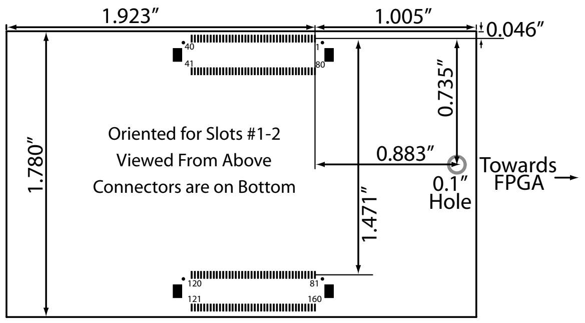


Figure B.7 Daughtercard dimensions and orientation

Figure B.7 is a scale drawing of the nominal size of a daughtercard and the location and orientation of the connectors. The figure is oriented as a daughtercard would be in slots #1-2. The connector pin numbers match those used in the schematics for both the wireless interface board and the radio board. Distances in the figure are measured relative to the center of pin 1 on the upper connector. The distance from pin 1 to the right board edge must not exceed 1.005"; any excess would cause a daughtercard to overlap the FPGA's heatsink. The vertical size of 1.78" is also a hard limit in order to accommodate daughtercards in adjacent slots. The size of the board left of the connectors is flexible; any excess will simply hang over the edge of the wireless interface board. The 1.923" dimension show in figure B.7 reflects the size of the TAP radio board. A 0.1" mounting hole is located in the right half of each daughter card slot. A matching hole can be included in a daughtercard to provide extra mechanical support.

Table B.1 lists which pins are connected to ground, +12 v and -12 v on the wireless interface board. Ground pins are connected directly to the ground plane used by the FPGA and other digital circuits. The ± 12 v pins are driven by dedicated supplies, isolated from anything on the wireless interface board. The digital signals connect directly to FPGA pins configured for 3.3 v (LVTTL) signaling.

B.2 SRAM

Two banks of static RAM are included on the wireless interface board. Each chip is a 36-bit wide memory with 256k words. Each bank is an IDT 100 MHz synchronous ZBT memory in a BG119 package, part number IDT71V65703-BG119. Zero Bus Turnaround (ZBT) is a bus interface which allows consecutive read and write commands without any stall cycles. The two RAM chips are connected to separate I/O banks on the FPGA. This facilitates using the chips separately or together, depending on the needs of a particular algorithm. An application node and core are available from Xilinx detailing how to interface the FPGA to these memories[22]. See tables B.3 and B.4 for the FPGA pin mappings for the two SRAM chips.

B.3 FPGA Pin Mappings

The following tables list the mapping of signals to FPGA pins on the TAP wireless interface board. The FPGA pin numbers included here use the same naming convention required by Xilinx's tools. A Xilinx ISE UCF formatted version of the complete pin mapping is available from http://taps.rice.edu/hardware.

Connector Pin	Radio Board Signal	Slot 1 FPGA Pin	Slot 2 FPGA Pin	Slot 3 FPGA Pin	Slot 4 FPGA Pin
1	EXT_GND	-	-	-	-
2	ADC_RX_DA6	C5	AC6	AL34	U31
3	ADC_RX_DA5	C7	AG1	AH32	V39
4	ADC_RX_DA4	D6	AB10	AD28	T34
5	ADC_RX_DA3	K4	AE3	AL33	V38
6	-	-	-	-	-
7	ADC_RX_DA2	D7	AC5	AJ36	R31
8	ADC_RX_DB8	J3	AA10	AE29	T29
9	ADC_RX_DB9	J1	AB11	AB34	V36
10	ADC_RX_DB10	E7	AD4	AE27	U35
11	EXT_GND	-	-	-	-
12	ADC_RX_DB11	H9	AF2	AD29	R33
13	ADC_RX_DB12	G9	AD3	AB30	R36
14	ADC_RX_DB13	J5	AF1	AD37	R37
15	ADC_RX_OTRB	H3	AA12	AM31	R29
16	EXT_12v	-	-	-	-
17	ADC_RX_DA0	H4	AE2	AD36	W27
18	ADC_RX_DA1	K9	AC4	AN31	V34
19	ADC_RX_DA7	J7	AE1	AL37	T28
20	ADC_RX_DA8	H6	AB5	AE35	Y36
21	EXT_GND	-	-	-	-
22	ADC_RX_DA9	J2	AB4	AE36	W29
23	ADC_RX_DA10	K8	AB6	AF31	U38
24	ADC_RX_DA11	J6	AD2	AF37	U30
25	ADC_RX_DA12	J9	AA7	AT33	Y32
26	EXT_12v	-	-	-	-
27	ADC_RX_DA13	L9	AB3	AG29	V29
28	ADC_RX_OTRA	H7	AA4	AR33	P34
29	ADC_RX_PWDNA	H8	AA3	AT34	W28
30	LED3	H2	AC2	AL31	N30
31	EXT_GND	-	-	-	-
32	LED2	L8	AA8	AK31	P35
33	LED1	F7	AA5	AF39	N39
34	-	K7	AC1	AR36	P37
35	5PA_EN	N10	AB2	AD30	L38
36	EXT_12v	-	-	-	-
37	2.4PA_EN	E6	AA9	AG28	P38
38	ANTSW1	K6	AA6	AN33	M37
39	ANTSW2	D5	AB1	AJ31	T38
40	EXT_GND	-	-	-	-
41	EXT_12v	-	-	-	-
42	RADIO_B7_FPGA	K2	AE4	AR39	U37
43	RADIO_B6_FPGA	K1	AL1	AG39	E39
44	RADIO_B5_FPGA	P4	AF3	AC37	P31

Connector Pin	Radio Board Signal	Slot 1 FPGA Pin	Slot 2 FPGA Pin	Slot 3 FPGA Pin	Slot 4 FPGA Pin
45	RADIO_B4_FPGA	N3	AB9	AC27	N35
46	EXT_GND	-	-	-	-
47	RADIO_B3_FPGA	P10	AC10	AR32	W37
48	RADIO_B2_FPGA	P8	AB8	AN34	V30
49	RADIO_B1_FPGA	P1	AB7	AE34	N36
50	-	P2	AF4	AH30	W26
51	EXT_12v	-	-	-	-
52	-	N5	AC9	AH37	M32
53	-	M3	AD6	AD27	V31
54	-	N6	AC8	AU33	N34
55	-	L6	AD7	AC33	T30
56	EXT_GND	-	-	-	-
57	-	N1	AK2	AB39	L31
58	-	N9	AE5	AD26	N33
59	-	K3	AG3	AP33	J35
60	RADIO_RXHP_FPGA	M6	AE6	AM38	M38
61	EXT_12v	-	-	-	-
62	RADIO_RXEN_FPGA	L4	AD10	AF36	Y33
63	RADIO_SHDN_FPGA	L5	AF5	AH29	T36
64	ADC_RX_DCS	M7	AD8	AM32	W32
65	ADC_RX_DFS	N2	AK1	AT32	N31
66	EXT_GND	-	-	-	-
67	ADC_RX_PWDNB	R11	AF6	AF38	W30
68	ADC_RX_DB0	M2	AE7	AE28	U28
69	ADC_RX_DB1	L3	AC12	AF28	P39
70	ADC_RX_DB2	M10	AJ2	AE37	N38
71	-	-	-	-	-
72	ADC_RX_DB3	M4	AG5	AL35	N37
73	ADC_RX_DB4	K5	AJ1	AE38	Y37
74	ADC_RX_DB5	N7	AH3	AR34	P36
75	ADC_RX_DB6	L7	AH4	AM33	W31
76	EXT_GND	-	-	-	-
77	ADC_RX_DB7	L1	AJ3	AF29	Y31
78	RSSI_ADC_CLK	M8	AH2	AJ33	V28
79	FPGA_CLK1	L2	AG2	AC36	T32
80	EXT_GND	-	-	-	-
81	EXT_GND	-	-	-	-
82	TX_DAC_PLL_LOCK	V5	AG10	AC29	V33
83	TX_DAC_Q_D15	W4	AM8	AH34	U39
84	TX_DAC_Q_D14	V3	AR6	AC34	H34
85	TX_DAC_Q_D13	V8	AH7	AP32	W33
86	-	-	-	-	-
87	TX_DAC_Q_D12	V4	AN7	AA31	U33
88	TX_DAC_Q_D11	V9	AE12	AL38	H32

Connector Pin	Radio Board Signal	Slot 1 FPGA Pin	Slot 2 FPGA Pin	Slot 3 FPGA Pin	Slot 4 FPGA Pin
89	TX_DAC_Q_D10	U10	AL9	AE39	V26
90	TX_DAC_Q_D9	V6	AM7	AL36	Y35
91	EXT_GND	-	-	-	-
92	TX_DAC_Q_D8	T12	AG9	AB32	H33
93	TX_DAC_Q_D7	U4	AK8	AA33	J32
94	TX_DAC_Q_D6	U9	AN6	AM37	E36
95	TX_DAC_Q_D5	T3	AL6	AE26	N28
96	EXT_12v	-	-	-	-
97	TX_DAC_Q_D4	T4	AE11	AG37	U29
98	TX_DAC_Q_D3	U5	AL7	AB35	N32
99	TX_DAC_Q_D2	R10	AM4	AL32	F32
100	TX_DAC_Q_D1	V7	AH8	AB31	P27
101	EXT_GND	-	-	-	-
102	TX_DAC_Q_D0	U8	AE10	AG32	T26
103	TX_DAC_SPI_SDO	U1	AE9	AD38	J31
104	TX_DAC_SPI_SDIO	R8	AL5	AB37	V32
105	TX_DAC_SPI_CLK	T7	AK6	AB36	U26
106	EXT_12v	-	-	-	-
107	TX_DAC_SPI_CSB	T6	AJ6	AC39	R26
108	TX_DAC_RESET	U2	AM3	AA26	R39
109	RADIO_LD	P6	AK5	AG36	V27
110	RADIO_CS#_FPGA	R3	AF9	AA32	R38
111	EXT_GND	-	-	-	-
112	RADIO_SCLK_FPGA	P9	AL3	AC38	R34
113	RADIO_DIN_FPGA	R5	AK4	AB33	M30
114	RADIO_TXEN_FPGA	R4	AM2	AA37	M34
115	DIPSW_1	T2	AG7	AF27	L39
116	EXT_12v	-	-	-	-
117	DIPSW_2	P3	AL2	AA35	M36
118	DIPSW_3	R1	AJ4	AC26	L37
119	DIPSW_4	R2	AK3	AC35	M33
120	EXT_GND	-	-	-	-
121	EXT_12v	-	-	-	-
122	-	W5	AG6	AG30	U34
123	RSSI_ADC_CLAMP	V2	AH6	AH33	H36
124	RSSI_ADC_SLEEP	P7	AF7	AK32	J36
125	RSSI_ADC_HIZ	V1	AF8	AM34	H37
126	EXT_GND	-	-	-	-
127	RSSI_ADC_D5	T11	AJ5	AK36	W34
128	RSSI_ADC_D4	P5	AJ7	AJ32	P33
129	RSSI_ADC_D3	R7	AD11	AF30	W36
130	RSSI_ADC_D2	R6	AF10	AK34	V37
131	EXT_12v	-	-	-	-
132	RSSI_ADC_D1	W3	AE8	AE32	K35

Connector Pin	Radio Board Signal	Slot 1 FPGA Pin	Slot 2 FPGA Pin	Slot 3 FPGA Pin	Slot 4 FPGA Pin
133	RSSI_ADC_D0	R9	AF11	AE31	K37
134	RSSI_ADC_D6	U6	AJ8	AG31	T33
135	RSSI_ADC_D7	W6	AG11	AB38	V35
136	EXT_GND	-	-	-	-
137	RSSI_ADC_OTR	T8	AJ9	AK35	U36
138	RSSI_ADC_D9	Y7	AK7	AJ38	R32
139	RSSI_ADC_D8	W7	AM9	AK33	R30
140	TX_DAC_I_D15	U12	AM6	AA36	P30
141	EXT_12v	-	-	-	-
142	TX_DAC_I_D14	T10	AH10	AM36	R35
143	TX_DAC_I_D13	W8	AF12	AJ35	U32
144	TX_DAC_I_D12	W9	AP7	AG33	W35
145	TX_DAC_I_D11	W11	AR7	AE30	K36
146	EXT_GND	-	-	-	-
147	TX_DAC_I_D10	V10	AK9	AK37	K31
148	TX_DAC_I_D9	V11	AE13	AF34	J34
149	TX_DAC_I_D8	Y9	AN9	AE33	K34
150	TX_DAC_I_D7	Y8	AT6	AF32	J33
151	-	-	-	-	-
152	TX_DAC_I_D6	W10	AD12	AA34	H38
153	TX_DAC_I_D5	V12	AT7	AG35	J37
154	TX_DAC_I_D4	Y4	AH11	AH38	K32
155	TX_DAC_I_D3	W12	AU7	Y26	T37
156	EXT_GND	-	-	-	-
157	TX_DAC_I_D2	V13	AR8	AD32	J38
158	TX_DAC_I_D1	W13	AT8	AJ34	K33
159	TX_DAC_I_D0	Y3	AD13	AB26	J39
160	EXT_GND	-	-	-	-

Signal Name	FPGA Pin		Signal Name	FPGA Pin
User I/O			**SystemACE Interface**	
Hex Display 1[0]	AH25		MPCE	M17
Hex Display 1[1]	AH26		MPBRDY	N18
Hex Display 1[2]	AH24		MPWE	D10
Hex Display 1[3]	AN25		MPOE	D9
Hex Display 1[4]	AH23		MPIRQ	N17
Hex Display 1[5]	AG22		MPA[0]	E9
Hex Display 1[6]	AG23		MPA[1]	E10
Hex Display 2[0]	AG20		MPA[2]	F10
Hex Display 2[1]	AG21		MPA[3]	F11
Hex Display 2[2]	AH20		MPA[4]	M14
Hex Display 2[3]	AR20		MPA[5]	M15
Hex Display 2[4]	AG19		MPA[6]	M16
Hex Display 2[5]	AH19		MPD[0]	G10
Hex Display 2[6]	AJ19		MPD[1]	G11
Led 1	AM16		MPD[2]	H10
Led 2	AG18		MPD[3]	H12
Led 3	AG17		MPD[4]	H11
Led 4	AH13		MPD[5]	J10
Button 1	AH21		MPD[6]	J11
Button 2	AJ21		MPD[7]	K11
Button 2	AJ22		MPD[8]	K10
Digital I/O [0]	K28		MPD[9]	J12
Digital I/O [1]	G30		MPD[10]	K12
Digital I/O [2]	H29		MPD[11]	J14
Digital I/O [3]	H30		MPD[12]	L13
Digital I/O [4]	J28		MPD[13]	L12
Digital I/O [5]	F30		MPD[14]	K14
Digital I/O [6]	E30		MPD[15]	M13
Digital I/O [7]	D31			
Digital I/O [8]	K30		**Clocks**	
Digital I/O [9]	J30		MGT BREFCLK+	J20
Digital I/O [10]	K29		MGT BREFCLK-	K20
Digital I/O [11]	J29		MGT BREFCLK2+	E20
Digital I/O [12]	G29		MGT BREFCLK2-	D20
Digital I/O [13]	H28		Oscillator Y3 (50MHz)	AK20
Digital I/O [14]	F29		Oscillator Y2 (100MHz)	AT20
Digital I/O [15]	E31		SystemACE Clk (33MHz)	N20

Signal Name	**FPGA Pin**		**Signal Name**	**FPGA Pin**
		SRAM 1		
SRAM1 A[0]	AL16		SRAM1 D[0]	AU19
SRAM1 A[1]	AH15		SRAM1 D[1]	AH18
SRAM1 A[2]	AL14		SRAM1 D[2]	AP19
SRAM1 A[3]	AM15		SRAM1 D[3]	AT19
SRAM1 A[4]	AM14		SRAM1 D[4]	AN18
SRAM1 A[5]	AT9		SRAM1 D[5]	AR19
SRAM1 A[6]	AU11		SRAM1 D[6]	AJ17
SRAM1 A[7]	AN11		SRAM1 D[7]	AT18
SRAM1 A[8]	AN17		SRAM1 D[8]	AM19
SRAM1 A[9]	AH16		SRAM1 D[9]	AL19
SRAM1 A[10]	AR16		SRAM1 D[10]	AK18
SRAM1 A[11]	AU15		SRAM1 D[11]	AL18
SRAM1 A[12]	AP18		SRAM1 D[12]	AU17
SRAM1 A[13]	AJ18		SRAM1 D[13]	AR18
SRAM1 A[14]	AM17		SRAM1 D[14]	AU18
SRAM1 A[15]	AM18		SRAM1 D[15]	AN19
SRAM1 A[16]	AK16		SRAM1 D[16]	AK12
SRAM1 A[17]	AU14		SRAM1 D[17]	AL12
SRAM1 ADV/LD	AP14		SRAM1 D[18]	AK14
SRAM1 BW[1]	AP16		SRAM1 D[19]	AR14
SRAM1 BW[2]	AR17		SRAM1 D[20]	AL11
SRAM1 BW[3]	AN14		SRAM1 D[21]	AT11
SRAM1 BW[4]	AT15		SRAM1 D[22]	AU10
SRAM1 CE[1]	AT14		SRAM1 D[23]	AR11
SRAM1 CE[2]	AP11		SRAM1 D[24]	AM11
SRAM1 CE[3]	AL17		SRAM1 D[25]	AR10
SRAM1 CEN	AH14		SRAM1 D[26]	AJ13
SRAM1 CLK	AR15		SRAM1 D[27]	AM10
SRAM1 LBO	AK10		SRAM1 D[28]	AJ12
SRAM1 OE	AP15		SRAM1 D[29]	AP10
SRAM1 RW	AN15		SRAM1 D[30]	AL10
SRAM1 SLEEP	AK19		SRAM1 D[31]	AT10
			SRAM1 D[32]	AL20
			SRAM1 D[33]	AT17
			SRAM1 D[34]	AM12
			SRAM1 D[35]	AN10

Signal Name	**FPGA Pin**		**Signal Name**	**FPGA Pin**
		SRAM 2		
SRAM2 A[0]	AK25		SRAM2 D[0]	AT31
SRAM2 A[1]	AK24		SRAM2 D[1]	AT30
SRAM2 A[2]	AU25		SRAM2 D[2]	AL30
SRAM2 A[3]	AR24		SRAM2 D[3]	AJ28
SRAM2 A[4]	AN23		SRAM2 D[4]	AK28
SRAM2 A[5]	AH22		SRAM2 D[5]	AM30
SRAM2 A[6]	AL23		SRAM2 D[6]	AR29
SRAM2 A[7]	AN22		SRAM2 D[7]	AL29
SRAM2 A[8]	AT28		SRAM2 D[8]	AP30
SRAM2 A[9]	AR28		SRAM2 D[9]	AM25
SRAM2 A[10]	AT26		SRAM2 D[10]	AR31
SRAM2 A[11]	AL28		SRAM2 D[11]	AM29
SRAM2 A[12]	AK29		SRAM2 D[12]	AN29
SRAM2 A[13]	AR26		SRAM2 D[13]	AL25
SRAM2 A[14]	AH27		SRAM2 D[14]	AR30
SRAM2 A[15]	AU29		SRAM2 D[15]	AU30
SRAM2 A[16]	AP29		SRAM2 D[16]	AR22
SRAM2 A[17]	AJ23		SRAM2 D[17]	AM22
SRAM2 ADV/LD	AU26		SRAM2 D[18]	AT23
SRAM2 BW[1]	AP26		SRAM2 D[19]	AU22
SRAM2 BW[2]	AN30		SRAM2 D[20]	AT22
SRAM2 BW[3]	AR23		SRAM2 D[21]	AL21
SRAM2 BW[4]	AT25		SRAM2 D[22]	AU21
SRAM2 CE[1]	AL24		SRAM2 D[23]	AP21
SRAM2 CE[2]	AM23		SRAM2 D[24]	AT21
SRAM2 CE[3]	AM28		SRAM2 D[25]	AN21
SRAM2 CEN	AR25		SRAM2 D[26]	AK22
SRAM2 CLK	AP24		SRAM2 D[27]	AP22
SRAM2 LBO	AL22		SRAM2 D[28]	AR21
SRAM2 OE	AP25		SRAM2 D[29]	AM20
SRAM2 RW	AM24		SRAM2 D[30]	AK21
SRAM2 SLEEP	AK30		SRAM2 D[31]	AP20
			SRAM2 D[32]	AJ27
			SRAM2 D[33]	AT29
			SRAM2 D[34]	AU23
			SRAM2 D[35]	AM21

References

1. A. Gupta, A. Forenza, and R.W. Heath, Jr., "Rapid MIMO-OFDM software defined radio system prototyping," in *IEEE Workshop on Signal Processing Systems*, 2004, pp. 182–187.

2. *http://www.ece.utexas.edu/∼rheath/research/mimo/proto/.*

3. R. Rao, W. Zhu, S. Lang, C. Oberli, D. Browne, J. Bhatia, J. Frigon, J. Wang, P. Gupta, H. Lee, D. Liu, S. Wong, M. Fitz, B. Daneshrad, and O. Takeshita, "Multi-antenna testbeds for research and education in wireless communications," in *IEEE Communications Magazine*, vol. 42, 2004, pp. 72–81.

4. P. Gupta, W. Zhu, M. Fitz, H. Lee, D. Liu, and S. Wong, "Field test results for space-time coding," in *Asilomar Conference on Signals, Systems and Computers*, 2003, pp. 243–247.

5. E. Knightly, B. Aazhang, J. P. Frantz, D. B. Johnson, and A. Sabharwal, *ITR/ANIR: Wireless Transit Access Points - New Foundations for a Scalable, Deployable, High Performance Wireless Internet*, NSF ITR Proposal.

6. *http://taps.rice.edu/.*

7. V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Transactions on Information Theory*, vol. 45, pp. 1456–1467, July 1999.

8. S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Select Areas in Communications*, vol. 16, pp. 1451–1458, October 1998.

9. F. Harris and M. Rice, "Multirate digital filters for symbol timing synchronization in software defined radios," *IEEE Journal on Select Areas in Communications*, vol. 19, pp. 2346–2357, December 2001.

10. C. Dick, F. Harris, and M. Rice, "Synchronization in software radios- carrier and timing recovery using FPGAs," in *Proceedings of 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2000, pp. 195–204.

11. P. Murphy, F. Lou, and J. P. Frantz, "A hardware testbed for the implementation and evaluation of MIMO algorithms," in *Proceedings of the 2003 Conference on Mobile and Wireless Communications Networks*, October 2003.

12. *http://taps.rice.edu/hardware/.*

13. *http://xilinx.com/products/tables/fpga.htm#v2p.*

14. *http://xilinx.com/cgi-bin/power_tool/power_Virtex2p.*

15. *http://xilinx.com/bvdocs/appnotes/xapp623.pdf.*

16. *http://maxim-ic.com/MAX2829.*

17. *http://rfdesign.com/mag/411rfdf1.pdf.*

18. D. Raychaudhuri, M. Ott, and I. Secker, "ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," in *Conference on testbeds and research infrastructures for the development of networks and communities*, 2005.

19. B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc. of the Fifth Symposium on Operating Systems Design and Implementation.* Boston, MA: USENIX Association, 2002, pp. 255–270.

20. B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media access for multirate ad hoc networks," in *Proceedings of ACM MOBICOM 2002*, 2002.

21. G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multihop wireless networks," in *Proceedings of ACM MOBICOM 2001*, 2001.

22. *http://xilinx.com/bvdocs/appnotes/xapp136.pdf.*