PROJECT #: AEE MQP NIX7

Egenera Debug Board

A Major Qualifying Project Report

Sponsored By Egenera:

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Nixon Mathew

4/24/2008

Approved:

Professor Alexander E. Emanuel, Advisor

Abstract

The Egenera debug board connects to the latest Egenera's processing blade servers and conducts specific debug operations and bios updates. The board integrates USB, SATA, Ethernet, Mouse, Keyboard, Video, PCI and PCI-E capabilities to the processing blades. There are four dual LED segments that will display debug data when the board is attached to the processing blades. The debug data, as well as the BIOS updates are stored in a SRAM so it could be accessed again in the future.

Acknowledgement

Special thanks to Professor Emanuel and Neil Haley for advising me in completing my Major Qualifying Project requirements. I am also very much grateful to Egenera for sponsoring the project. My faculty and company advisors and my sponsor company brought my project to be where it is today.

Executive Summary

The Egenera Debug board is an extension boards for the newest Egenera processing blade server that are going to be in the market in the near future. Since the architecture of these processing blades is different from the older processing blades, the older debug cards are not capable of performing any of the debug operation on the new processing blade servers. A new daughter board is required to conduct debug operations. Thus the purpose of this project is to design and develop a board that implements all the older features of the debug board to the newer one. Moreover, the newer debug board entails many other features that the older board didn't support. Due to the outdated design and parts of the older debug board, the new debug board is almost completely revolutionized, and many powerful and efficient parts are employed in the current design.

The schematic works for the design is carried out in ORCAD and I had to familiarize myself with this schematic capture tool instead of the Multisim tool which is widely used in WPI. Overall the ORCAD performed better than Multisim.

Originally a Philips NXP ARM7 microcontroller was supposed to be used for the design, but when the design was mostly completed, a better MCU solution was found that would simplify the coding process and supports a full blown Linux OS. Thus the design was significantly changed by implementing the ATMEL MCU and adding extra memories as well as configuring the MCU and other components differently. Even though this was superior design, this caused a set back by having a delay in sending the board out to the manufacturing company. Thus no measurements were able to take from the debug board. However, the design has been through several design reviews at Egenera and different depopulated components were added to ensure the functionality of the board.

The CPLD programming is carried out in Quartus because the CPLD family was not available in Xilinx. The simulations however were done in Xilinx since the VDHL file could be interchanged in both Xilinx and Quartus.

Acronyms

Table 1 provides the acronyms that are used in the document.

Table 1: Acronyms

Acronyms	Definition
BMC	Baseboard Management Controller
cBlade	Control Blade Server
CAS	Column Address Strobe
CD	Compact Disc
CLK	Clock
CPLD	Complex Programmable Logic Device
CS	Chip Select
DIP	Dual Inline Package
EEPROM	Electrically Erasable Programmable Read-Only Memory
FWH	Firmware Hub
I/O	Input/Output
I^2C	Inter Integrated Circuit
LPC	Low Pin Count
MCU	Micro Controller Unit
MII	Media Independent Interface
NMI	Non Maskable Interrupt
OS	Operating System
OE	Output Enable
pBlade	Processing Blade Server
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCI-E	Peripheral Component Interconnect Express
PHY	Physical Layer
RAS	Row Address Strobe
RMII	Reduced Media Independent Interface
SATA	Serial Advance Technology Attachment
SD	Secure Digital
SDRAM	Synchronous Dynamic Random Access Memory
SRAM	Static Random Access Memory
TCK	Test Clock
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
TRST	Test Reset
VGA	Video Graphics Array
USB	Universal Serial Bus
WE	Write Enable
WP	Write Protect

Definitions

Table 1 provides the definitions for specific terms that was used throughout the document.

Table 2: Definitions

Terms	Definition
BA	Bank Select
DQML	Data Write/Output Lower Byte Enable/Disable
DQMU	Data Write/Output Upper Byte Enable /Disable
RY	Ready/Busy Output

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The design challenge of the Egenera Debug Board project is to design a daughter board that will connect to either Egenera's Processing Blade (pBlade) or Control Blade (cBlade) and enables the user to either troubleshoot the product or conduct bios updates to the host server through Ethernet. The debug board integrates PC like features to the blade servers such as mouse, keyboards, monitors, CD drives, and USB connectivity. Four seven segment display will show the debug data of the host server and this data is also stored in memory.

This paper provides a design specific block diagram and detailed explanation of each of the design blocks. The debug board specifications and requirements are also provided in this document. Some preliminary research on product parts was also conducted and different part choices have been narrowed down for the main components.

Furthermore, different design approaches for hardware integration and programming is discussed in this document.

2. Debug Board Requirements

The desired and minimum requirements for the debug board are explained in the following sections.

2.1 Minimum Requirements

The following are the minimum requirements that the debug board should entail upon completion of this project:

- Designed for video connectivity for Pilot II based video solutions, where the video signals will be driven from the Pilot II and is connected to the VGA connector
- Design supports SATA, PS/2 Mouse and Keyboard ports
- Supports four USB Type A ports
- Design supports a Low Pin Count (LPC) bus that intakes Port 80-83 data and display it on a seven segment LED
- Requires redirection and buffering to an external host through Ethernet for remote access capability.
- Low cost and high performance Rom Emulator that supports LPC and Firmware Hub (FWH)
- Supports eight settings of general purpose signals to the motherboard through DIP switches
- Four Push Buttons for Power Control, Reset, NMI, and BMC Reset
- Revision Identification through LED interface and remote interface

2.2 Additional Desired Requirements

- Low cost Connector for the mother board connection
- One additional PCI expansion for addition debug connections
- Remotely control the status of the GPIOs and the push buttons

3. Block Diagram

Given the requirements a block diagram was created to illustrate the design flow of the debug board. This is illustrated in Figure 1. The different functionalities of the blocks are discussed further in the following sections.

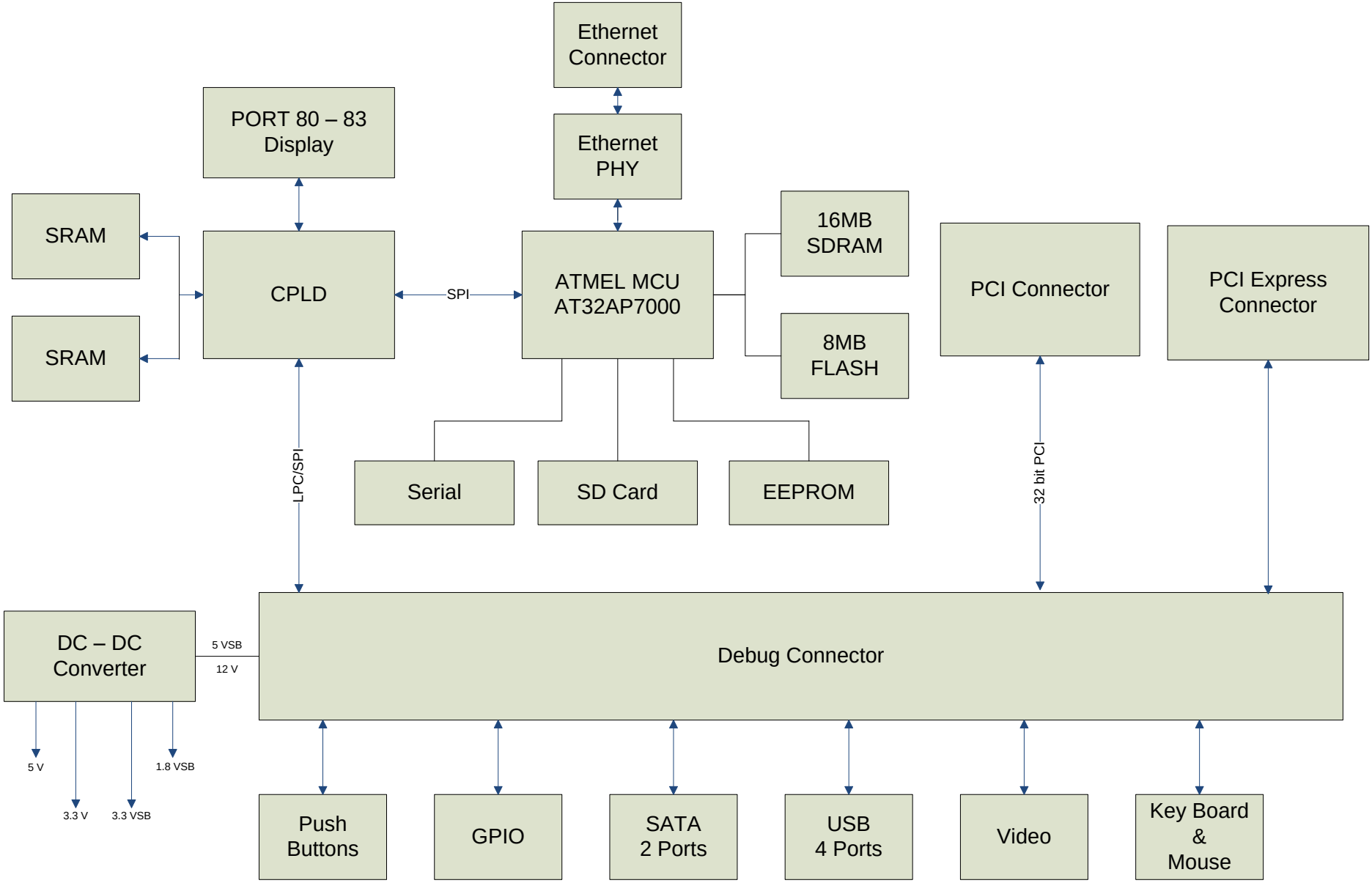


Figure 1: Block Diagram

4. ROM Emulator

The ROM emulator processes and executes various instructions sets and delegates numerous operations. The microcontroller uses an Ethernet port to communicate with a foreign device. Furthermore the ROM emulator is connected to the CPLD through SPI bus for displaying, redirecting and storing data. The microcontroller unit (MCU) also enables the user to control the GPIOs and the push buttons remotely. The MCU is programmed through a JTAG interface and requires an external clock oscillating within the frequency range of 1 MHz to 32 MHz and a 33 KHz signal. The schematic for the ROM Emulator is demonstrated in Appendix A: MCU Schematic Part .

Atmel AT32AP7000 ARM7 based microcontroller is utilized for the debug board. Another alternative to the Atmel processor was the NXP ARM7 and ARM9 based processors. Even though NXP processors supports all the features required for the debug board, the Atmel processor's ability of running Linux gives it a huge advantage over the NXP processors. The Linux environment simplifies the integration of Ethernet connectivity. The following specifications illustrate the features of the microcontroller that going to be implemented in the project [3]:

- External Memory Interface (EBI) with various memory supports
    - SRAM
    - SDRAM
    - FLASH
    - Secure Digital (SD)
- Two Wire Interface that follows Philips I^2C compliance
- Synchronous Protocol Controllers
- Universal Synchronous\Asynchronous Receive/Transmitters
- Ethernet MAC 10/100 Mbps interface that supports MII and RMII

The block diagram for the MCU is illustrated in Figure 2. Further discussion of this microcontroller's desirable features is addressed in the preceding sections.

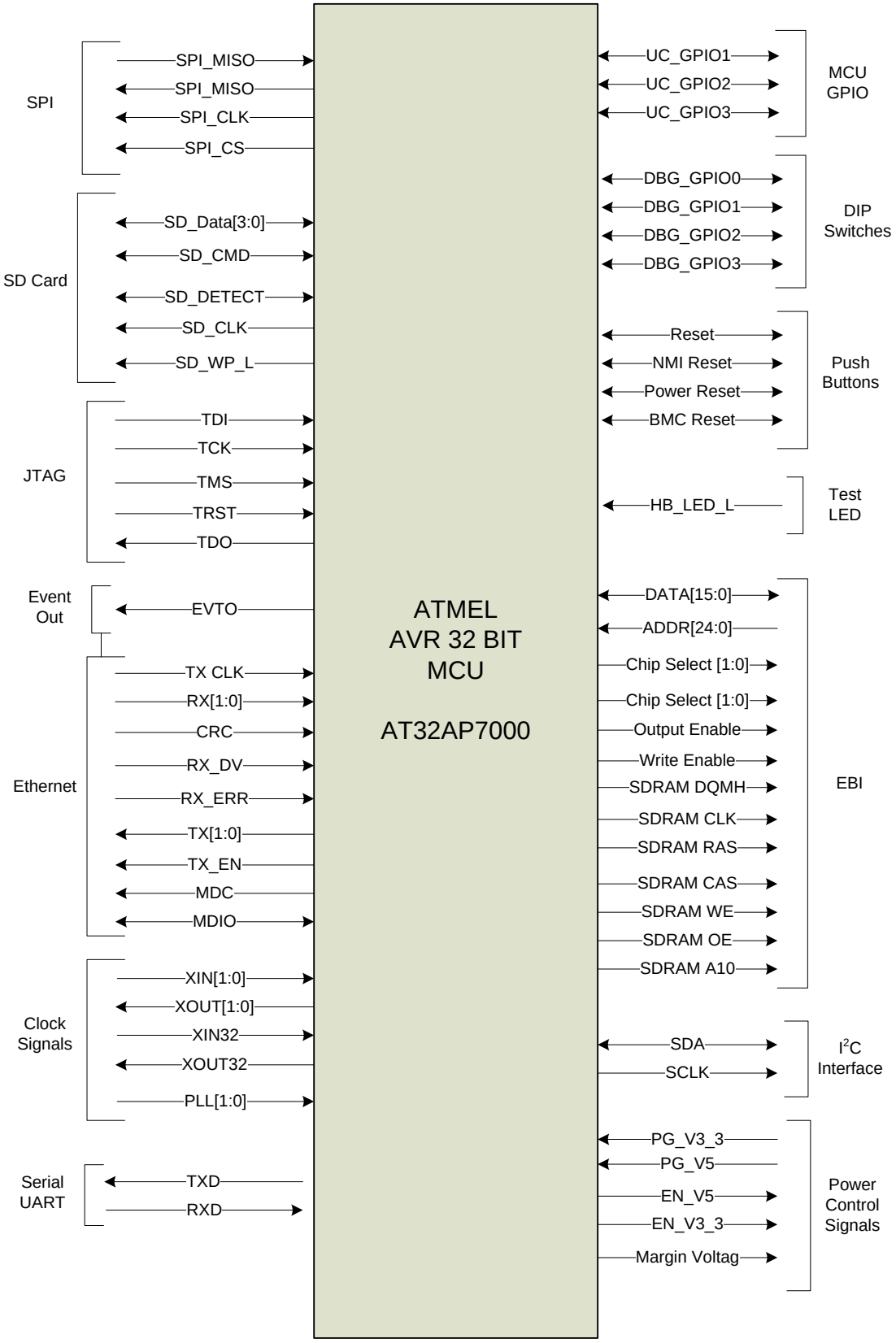


Figure 2: MCU Block Diagram

4.1 External Bus Interface

The external bus interface (EBI) enables the microcontroller to connect to various different memories. The MCU requires memory for BIOS and the operating system (OS) environment. A 16 MB of SDRAM is used for the BIOS and a 8 MB of FLASH is used for the OS. A Linux based OS is expected to be running on the MCU. The next section explains the design integration of these memories [3].

4.1.1 SDRAM

The block diagram for the SDRAM is shown in Figure 3

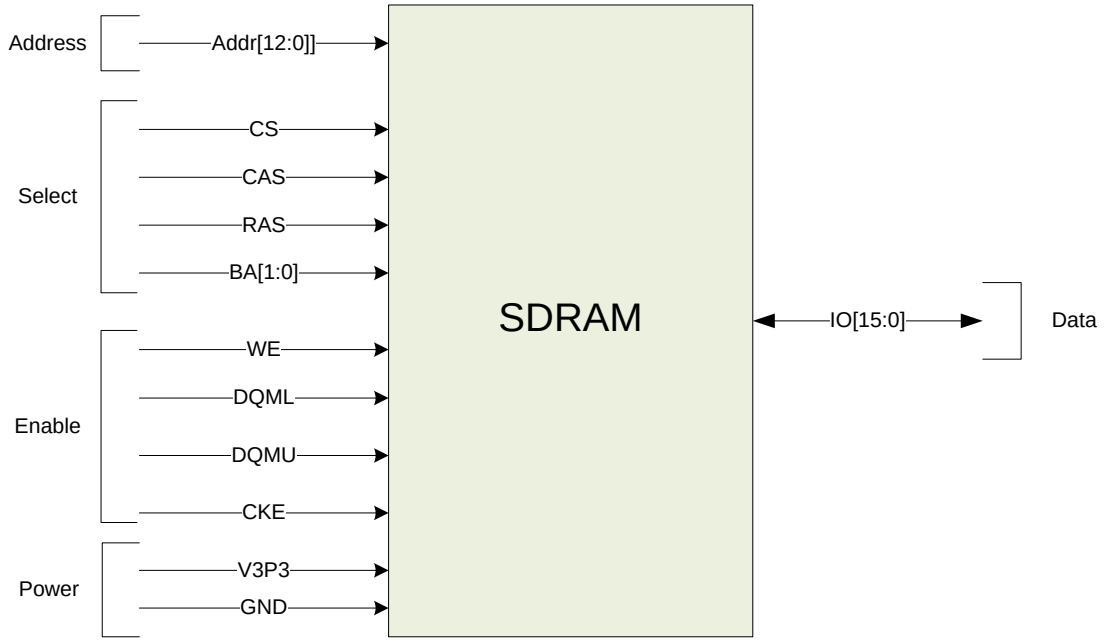


Figure 3: SDRAM Block Diagram

As shown in Figure 3, the SDRAM has 13 address pins and 16 data pins. The clock enable pin needs to be high in order to read or write to the SDRAM. If the write enable is low then the data that appears on the data pins will be written to the specified address, otherwise the user can read from the SDRAM. The only exception is that DQML and the DQMU can mask the output when it is active. The MQML is designated for the lower byte, first eight bits and the MQMU is for the higher byte, the remaining 8 bits [1].

The addresses are governed by the CAS, RAS and BA0, BA1 pins. There are two available memory banks. The MCU have to select between these two banks then drive either the RAS or CAS low to pick either the row addresses or the column addresses respectively [1][3]. The schematic for the SDRAM configuration is illustrated in Figure 4.

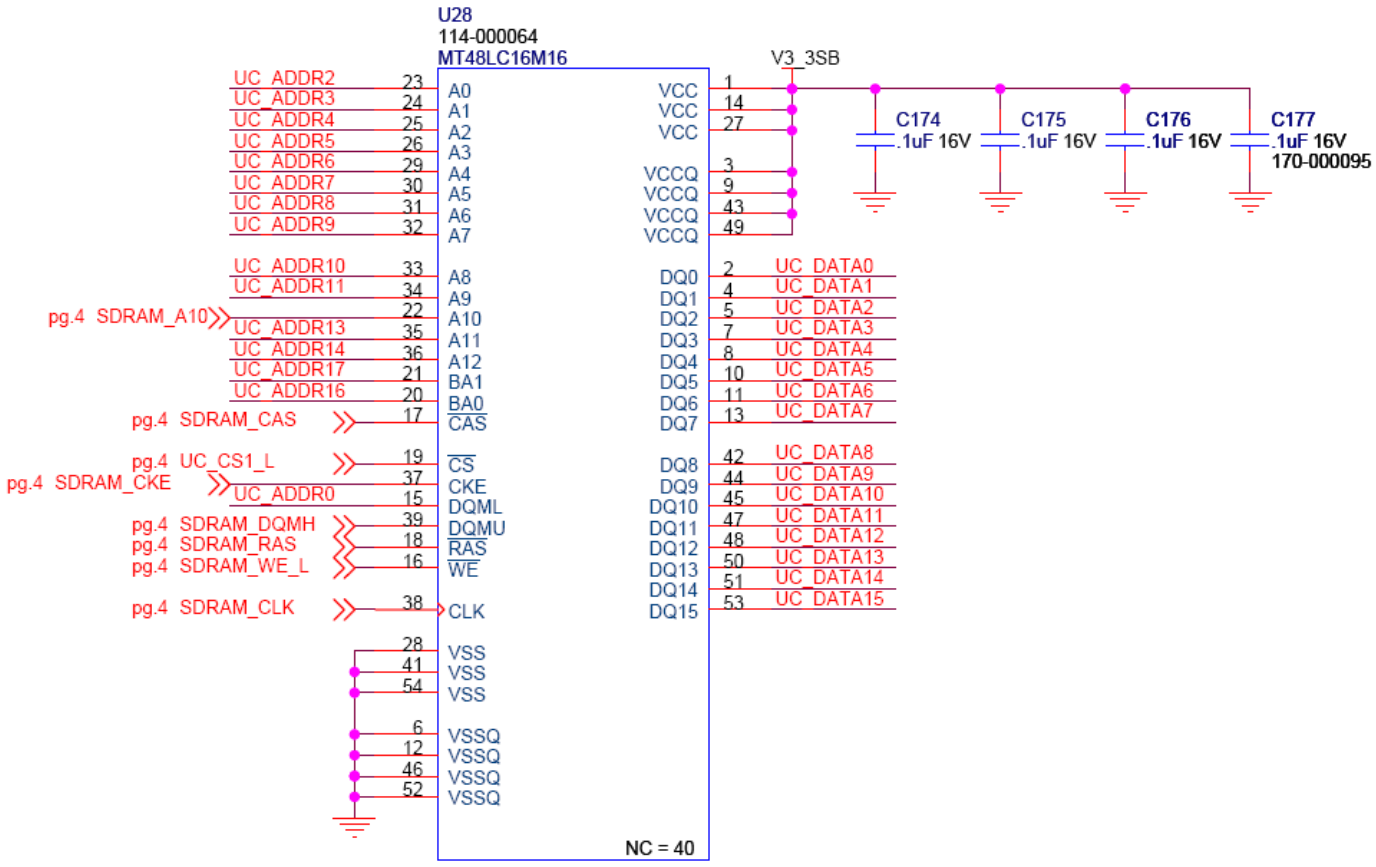


Figure 4: SDRAM Schematic

4.1.2 FLASH

The flash memory is little bit simpler than the SDRAM. The block diagram for the FLASH memory is demonstrated in Figure 5.

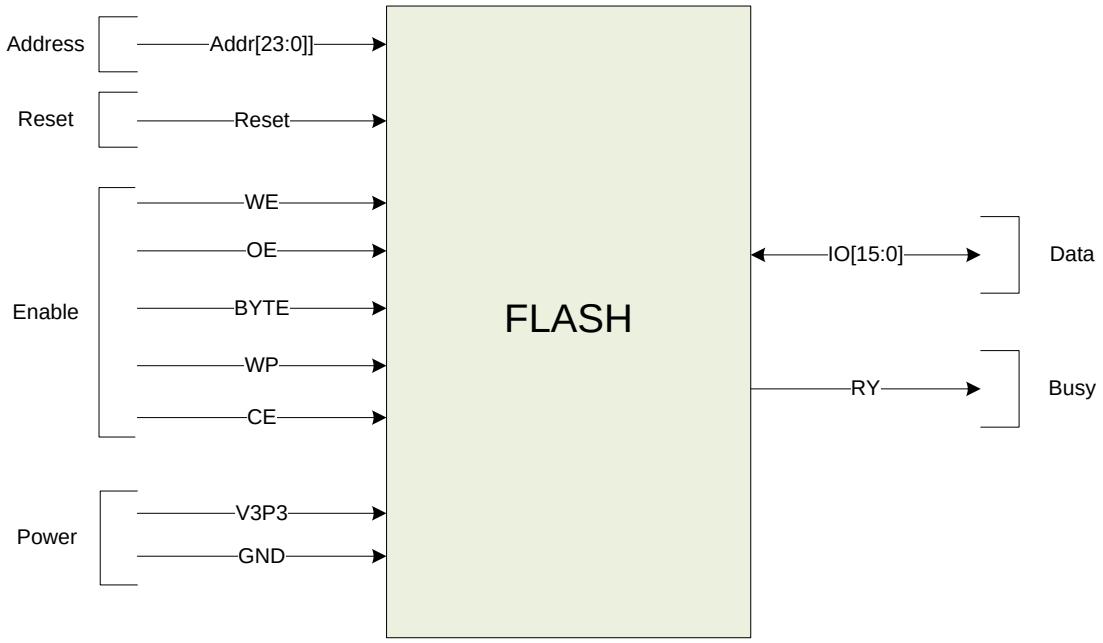


Figure 5: FLASH Block Diagram

Figure 5 shows that the FLASH memory has 24 address lines and 16 data pins which are connected to the MCU's EBI bus. The reset pins allows the flash to be in reset. While the flash device is in reset it will ignore all read and write commands. Driving the WE pin low enable the user to write to the flash memory and the OE enables read functions. The CE permits the functionality of the flash memory. The Byte pin allows the chip to read/write a word (16 bits) or half a word. In the debug board design it is connected to Vcc so all 16 data pins are being used. The WP signal is used to protect the data, so while WP is active the data in flash cannot be modified. The RY signals the MCU if it's busy. Pull-up resistors are used for the WP and RY signals, so by default the write protect is disabled and the content of the flash can be modified. However, the MCU can drive the WP pin low to enable write protection [12]. The schematic for the flash device is shown Figure 6.

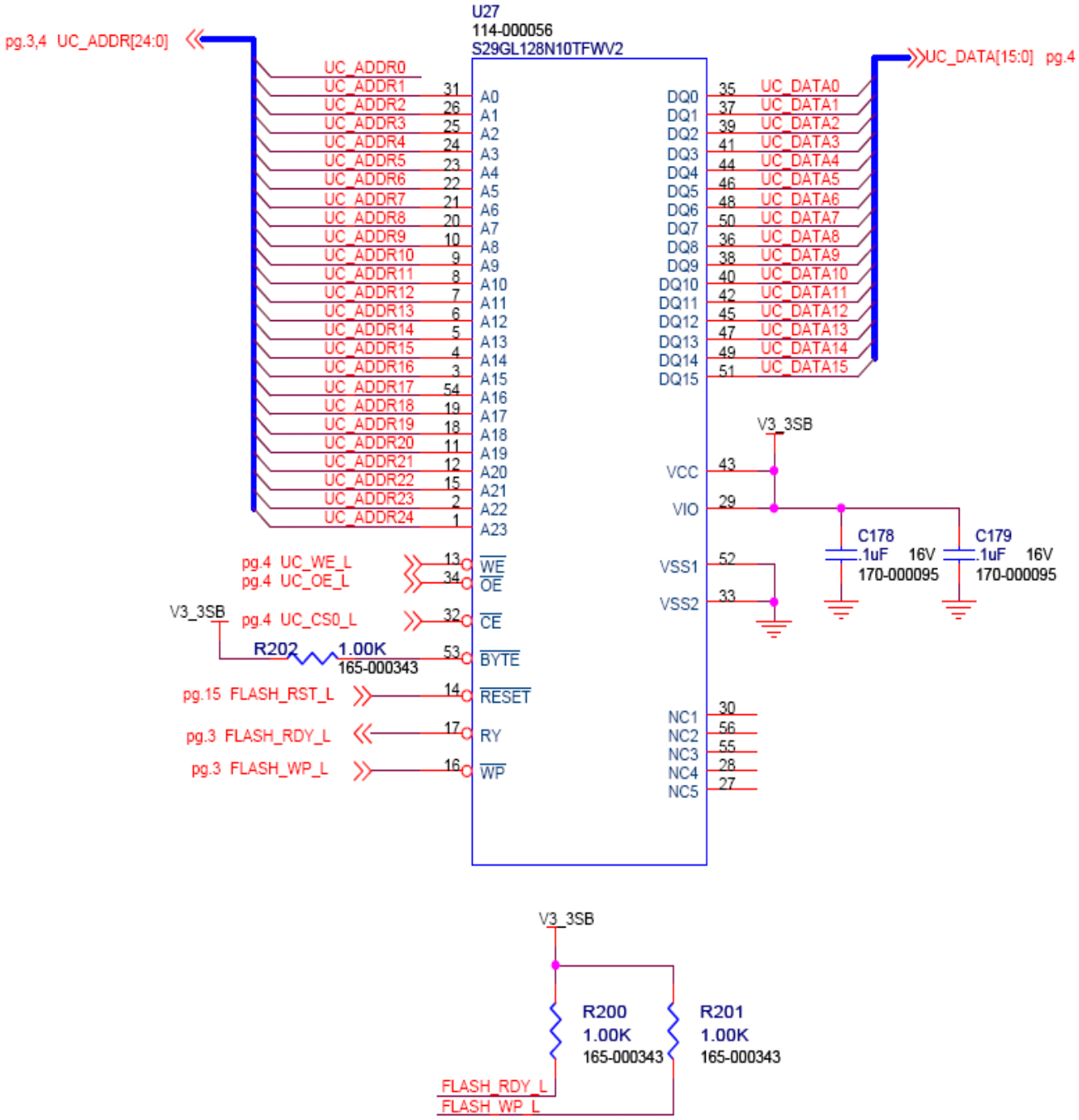
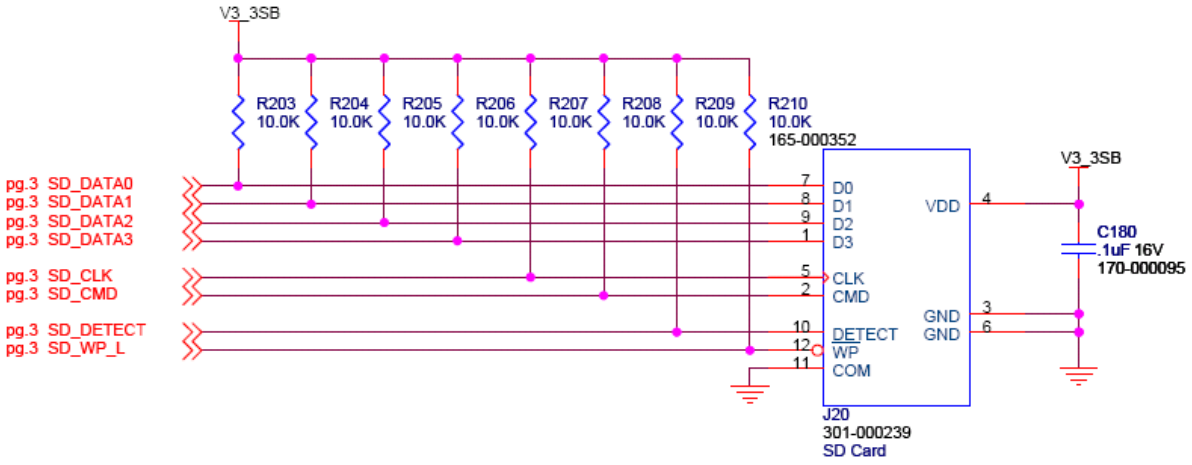


Figure 6: FLASH Schematic

4.2 SD CARD

The SD card is integrated to the design so if the newer revisions of the Linux or a different OS that are going to be implemented in the future requires more memory.



4.5 MCU GPIOs

There are three GPIO pins that are connected to the CPLD. The functionalities of these GPIO registers are at the user's discretion because these pin can be assigned to different applications on the CPLD.

4.7 SPI Interface

The SPI interface is used to channel data between the CPLD and the MCU. The four SPI pins are connected directly to the IO pins and the clock pin of the CPLD. It is disused further in section 5.1 CPLD to MCU Connection.

4.6 I2C Interface

The I2C interface is similar to the SPI interface, but it only has two pins which are SDA and SCL. The SDA pin is the data pin and the SCL is the serial clock input. The I2C implementation is discussed further in 7.3 External MAC Memory.

4.8 JTAG Connector

The ROM Emulator is programmed through JTAG interface. The debug card utilizes a ten pin header for the JTAG Connector, which consists of the typical JTAG pins: TMS, TCK, TDI, TDO and TRST and additional input pin: MCU reset and an output pin: event out (EVTO) [3][8].

Table 3: JTAG Pin Out

Pin Name	Type	MCU Pin Number
TMS	Input	J2
TCK	Input	J4
TDI	Input	J3
TDO	Output	J5
TRST	Input	J1
EVTO	Output	D13
RESET_N	In	K5
VCC	Power	Not Applicable
GND	Power	Not Applicable

The schematic for the JTAG Connector is illustrated in Figure 7.

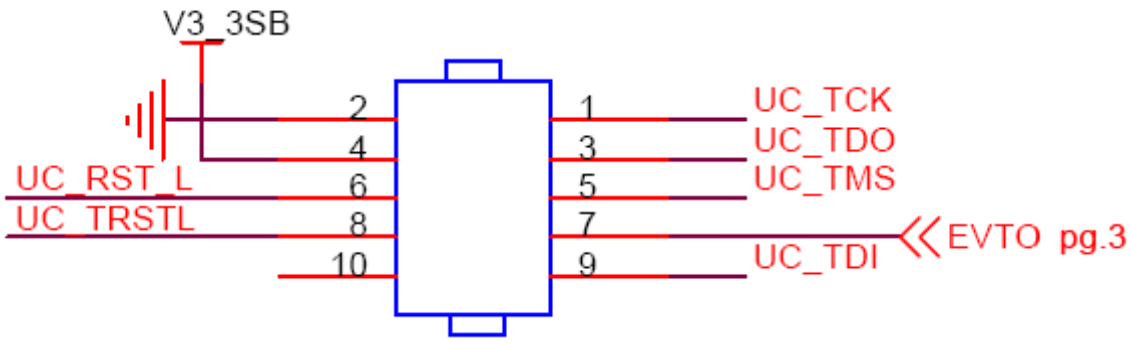


Figure 7: JTAG Connector

4.9 UART Serial Connector

The ROM emulator supports UART serial data transferring capabilities. The pins dedicated for UART are listed in Table 4.

Table 4: Serial Connector Pins

PIN	TYPE	DESCRIPTION
RXDO, RXD2, RXD3	Input	Serial Receive Data
TXD0, TXD1, TXD3	Output	Serial Transmit Data

Only one UART serial link connection is implemented in this design, so only one pair of receive and transmit pins are needed. These pins can't be connected directly to a serial connector. An external transceiver chip is required to transfer data from the ROM emulator to the serial connector [3][11]. A MAX3221 transceiver chip is used for this design. The schematic for the transceiver is shown in Figure 8.

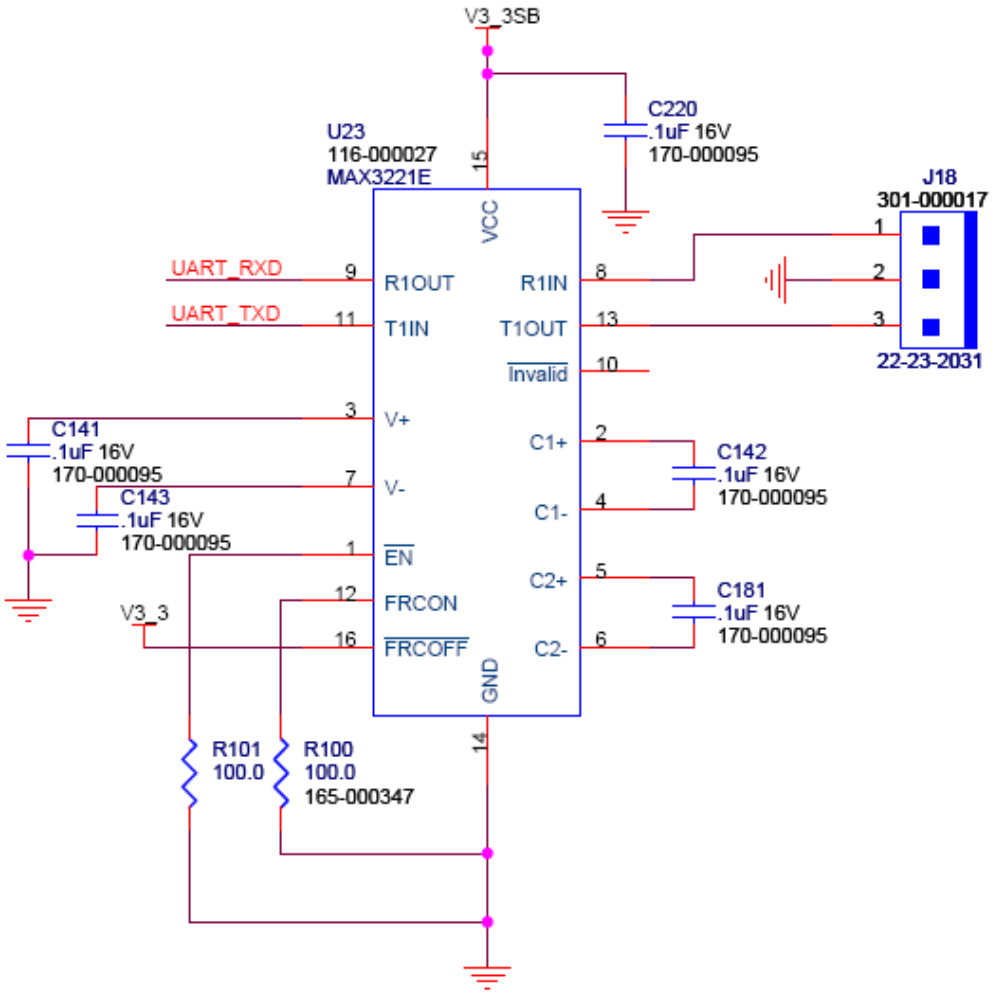


Figure 8: Transceiver Schematic

The transceiver internal power supply has dual charge pumps that outputs 5.5V and -5.5V. If the output voltage exceeds 5.5V then charge pumps are disabled. Each of the charge pumps requires a flying capacitor. The first capacitor is connected in between C1+ and C1- and the second capacitor is placed between C2+ and C2-. In addition to the flying capacitor for the charge pumps, reservoir capacitors are also needed for V+ and V- supply. Three 0.01uF capacitors would be sufficient for both the flying capacitor and the reservoir, but 0.33uF are used because it reduces the ripple on the transmitter output and it is safer to use higher capacitor value than the required minimal capacitance of 0.01uF. The capacitor values may degrade due to temperature; therefore 0.33uF capacitors are a better fit. Pin 16 or the FORCEOFF pin is tied to Vcc and the pin 1 or the EN pin is

pulled down to the ground. The logic displayed in Table 5 denotes that when FORCEOFF and EN are logic high and low respectively, the receiver out (R_OUT) and transmitter out (T_OUT) are active [11].

Table 5: Transceiver Logic

FORCEOFF	EN	T_OUT	R_OUT
0	0	High-Z	Active
0	1	High-Z	High-Z
1	0	Active	Active
1	1	Active	High -Z

The R_OUT and T_OUT pins connect to a DB9 serial connector. The pin out for the connector is shown in Figure 9: DB9 Serial Connector. Only the signal ground, receive data and transmit data pins are being used, thus the DB9 header shown in Figure 8 only has three pins [13].

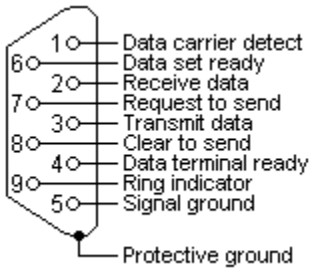


Figure 9: DB9 Serial Connector

4.10 External Clock

The MCU requires two clock signals. One of them has to be in between 1 MHz and 32 MHz. This design utilizes a 25 MHz crystal with 14pF load capacitance. The load capacitance of a crystal is specified by the manufacture and it can vary from 5pF to 50pF. The low load capacitance has a greater loop gain, thus less prone to start up problems. However, the tradeoff is that the frequency is less stable when a lower load capacitance is used when compared to a higher load capacitance. Since the required clock signal range is reasonably far from the crystal frequency that is being utilized, 14pF

load capacitance should generate a desired clock signal with minimal startup problems [3]. The schematic for the external clock is demonstrated in Figure 10

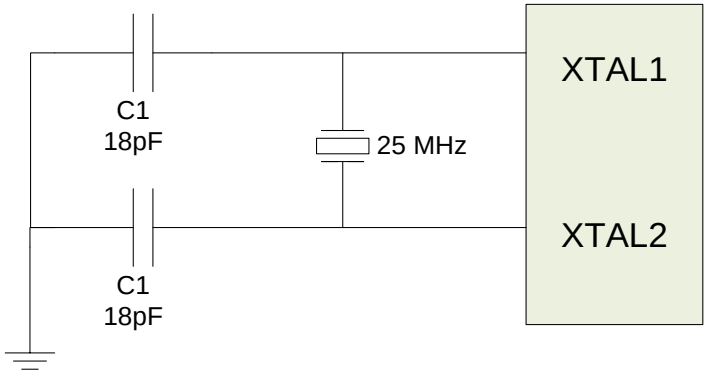


Figure 10: External Clock Schematic

The capacitor values for the external clock network are calculated with the following equation, where C_s is the stray capacitance of the PCB, which is typically 5pF [5].

$$C_L = \frac{C_1 * C_2}{C_1 + C_2} + C_s$$

Equation 1: Load Capacitance Equation

It is recommended that C_1 and C_2 should be approximately equal, thus the above equation simplifies to the expression shown below.

$$C_L = \frac{C_{eq}}{2} + C_s$$

By substituting 5 pF for the C_s and 14 pF for C_L the value of C_1 and C_2 are calculated to be 18 uF.

The second clock input is a 32 MHz clock and it is configured the same way as the 25 MHz clock. Additionally, a depopulated 25 MHz clock is added to the second clock input of the MCU so if the first 25 MHz is not working properly then the second clock input could be used. The clock inputs are shown in Appendix A: MCU Schematic Part B.

5. Complex Programmable Logic Device

The CPLD block performs all the logical and arithmetical operations. Egenera prefers an ATMEL chip since the company has all the tools to program the ATMEL chip. An Altera MAX300A CPLD is ideal for the debug board because it has four IO ports (port 1-4) and a large number of I/O pins dedicated to each port [10]. The CPLD plays a major role in this design because it lowers cost, simplifies design and any software updates could be easily facilitated. The major connections to the CPLD are the MCU, seven segment displays, JTAG connector and the memory. The block diagram for the CPLD is demonstrated in Figure 11.

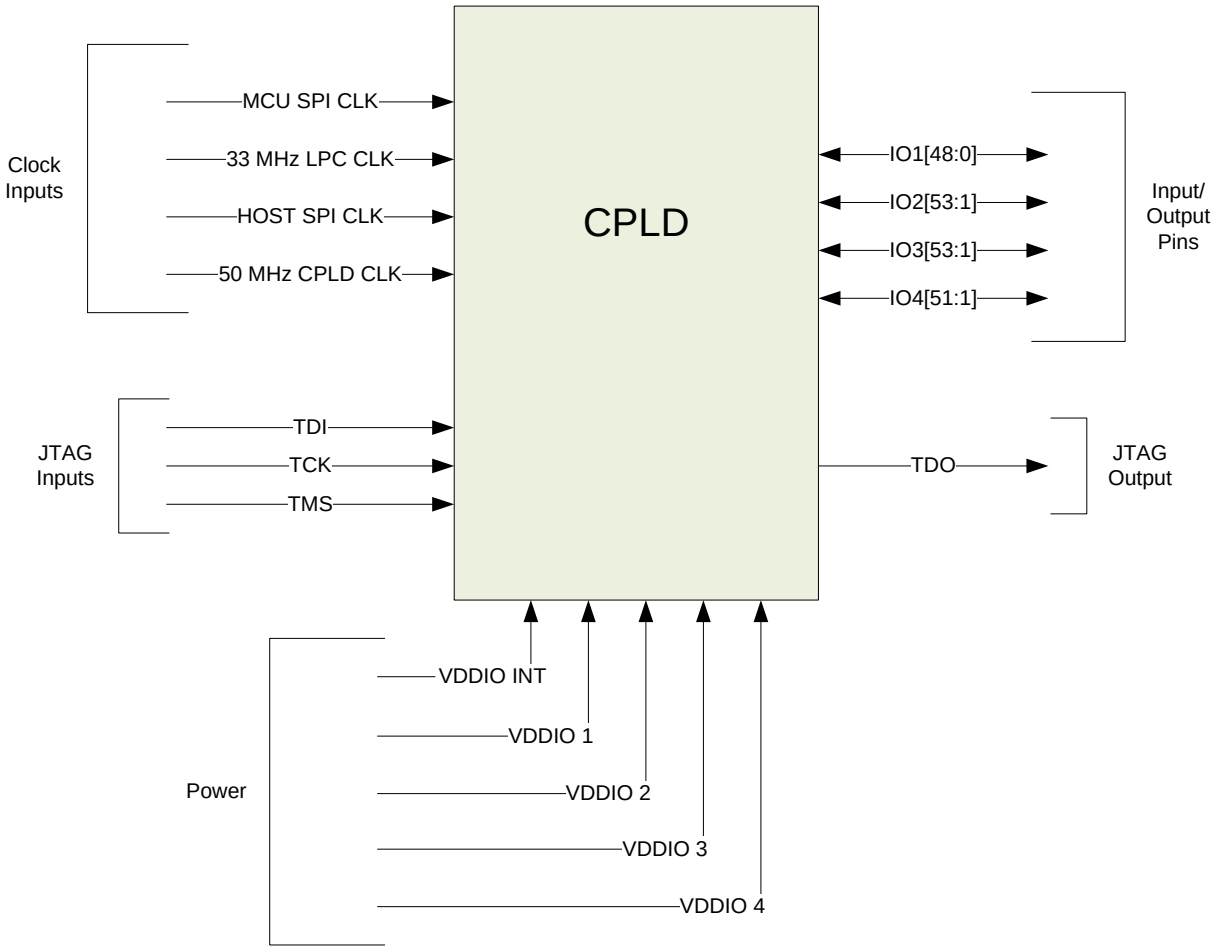


Figure 11: CPLD Block Diagram

The different signal descriptions for the CPLD are discussed further in the following section.

5.1 CPLD to MCU Connection

The CPLD is connected to the microcontroller through a SPI bus. The MCU is set as the master device and the CPLD act as a slave. The connection between the MCU and the CPLD is shown below.

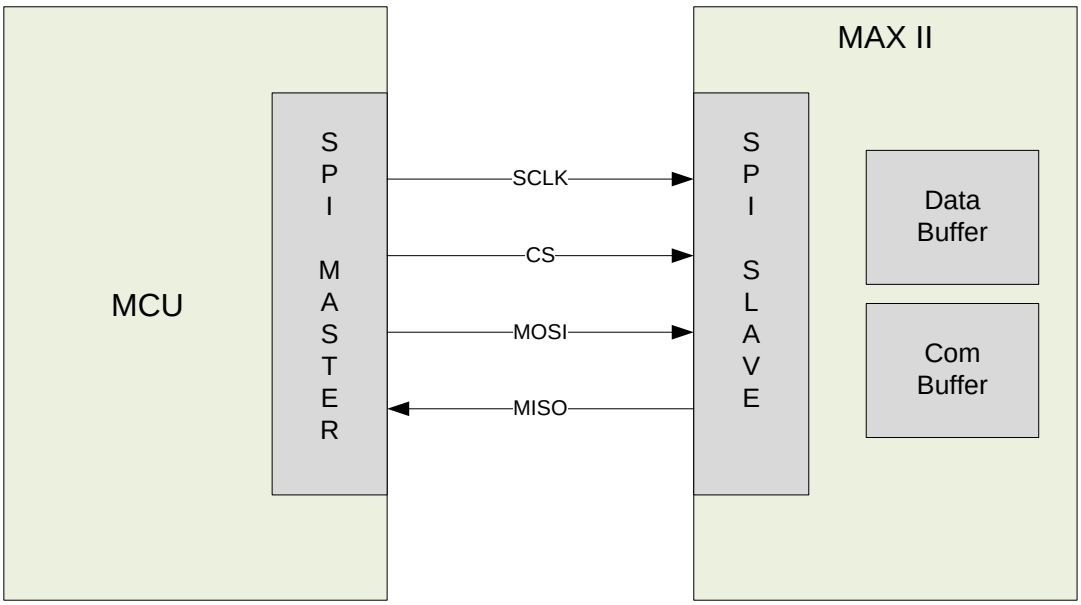


Figure 12: SPI Connection diagram

Figure 12 shows that there are four connections from MCU and the CPLD, which are SCLK, CS, MOSI and MISO. The CPLD has the capability of being either the Master or Slave. Since the MCU delegates different tasks the CPLD is run in slave mode.

Table 6 describes the pin out for the CPLD SPI interface.

Table 6: SPI CPLD Pin Out

Pin Name	Type	Pin Description	MAXII Pin #
SCLK	Input	Serial Clock	P12
NCS0	Input	Peripheral Chip Selects	T15

MOSI	Input	Master Out Slave In	P13
MISO	Output	Master In Slave Out	R16

The table above describes the required pins for SPI operations. The serial clock and the peripheral chip select pins are driven from the MCU. The MOSI pin is configured as the receiver input and the MISO is configured as transmitter output. The SCLK is generated when NCS0 goes active low and the transmission or receiving of data shortly follows [14]. The pins described in

Table 6 can be connected to the appropriate I/O pins in the CPLD, which is also given in the last column of the table.

5.2 CPLD to Seven Segment Displays

The seven segment displays are for the port 80-83 data and each of the packages contains two seven segment displays. Since there are four ports outputting data, four display package components are needed. The schematic for the display is shown Figure 13: 7-Segment Display and the rest of the three display packages are wired in the same manner. All the pins of the seven segment displays are connected to the CPLD's I/O ports and it is shown in Figure 13. The 100 Ω resistors limit the current and keep the LEDs from blowing out. The AN0 and AN1 are for selecting between the two of the seven segments display. The other pins select which led segments to be active. This component requires a LED driver, which is programmed in the CPLD. The full schematic of the LED display is given in Appendix E: Port 80-83 Display Schematic.

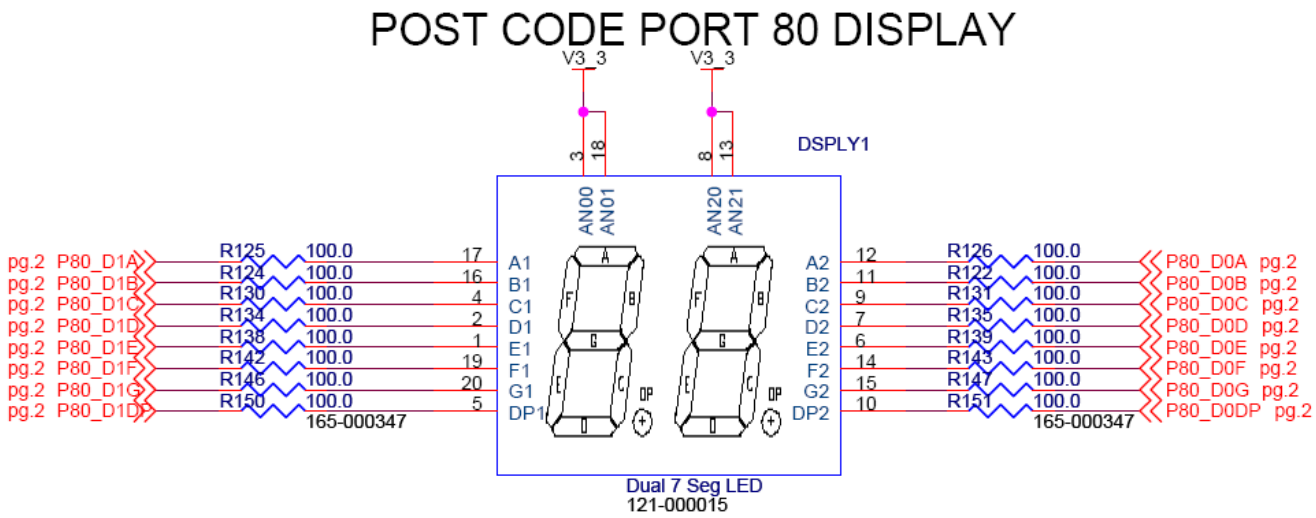


Figure 13: 7-Segment Display

5.2 CPLD JTAG Header

The CPLD is programmed through a JTAG connector. An eight pin JTAG header is commonly used for CPLDs. The pin-out for the JTAG connector is listed in Table 7: CPLD JTAG Header Pin-out and these pins correspond to the JTAG pins in the CPLD [8].

Table 7: CPLD JTAG Header Pin-out

Pin #	Pin Name	MAXII Pin #
1	Vddh (3.3V)	Not Applicable
2	TDO	M6
3	TDI	L6
4	nTRST	Not Applicable
5	Not Connected	Not Applicable
6	TMS	N4
7	Vss	Not Applicable
8	TCK	P3

5.3 CPLD Code and Simulation

The CPLD has to be programmed through the Quartus environment since the debug board uses an Altera CPLD. VHDL language is used to code the CPLD and since VHDL is compatible in Xilinx, the codes are simulated in Xilinx. The following section explains the design flow and the simulation process.

5.3.1 CPLD Coding

The code for channeling data from the MCU to the SRAM is given in Appendix F: CPLD SPI Interface Code.

The flowchart for the SPI interface for writing data from the MCU to the SRAM is shown in Figure 14. As illustrated in Figure 14, the main concept of the SPI interface is to function on every rising edge of the serial clock input. Thus, after the entity and the architecture are defined, the code will run infinitely as long as there is a clock input. Since the data is coming into CPLD serially, it has to break up the data into parallel outputs for the SRAM. Therefore a 35 bit array is created. The first bit is designated for Output enable and Write enable, the 19 bits afterwards are the address bits and the remaining 16 are the data bits.

The SRAM is only enabled when all the data bus and the address bus is full because this ensure that no unwanted data will be stored in an arbitrary location. The SRAM chip enable is active low, so a logic low has to be applied to the CE pin to enable SRAM operations. A counter is used to track the number of bits that are stored in the array and when the counter reaches 36 the CPLD will output logic low for the CE. The counter counts up to 36 instead of 35 because the counter variable is defined as a *variable* instead of a *signal* and the *variable* will takes on each value in the same clock cycle. This means that the CE goes low before the last data bit is loaded.

Once the data has been send to the SRAM, the CE is once again set to logic high and the counter is reset to zero.

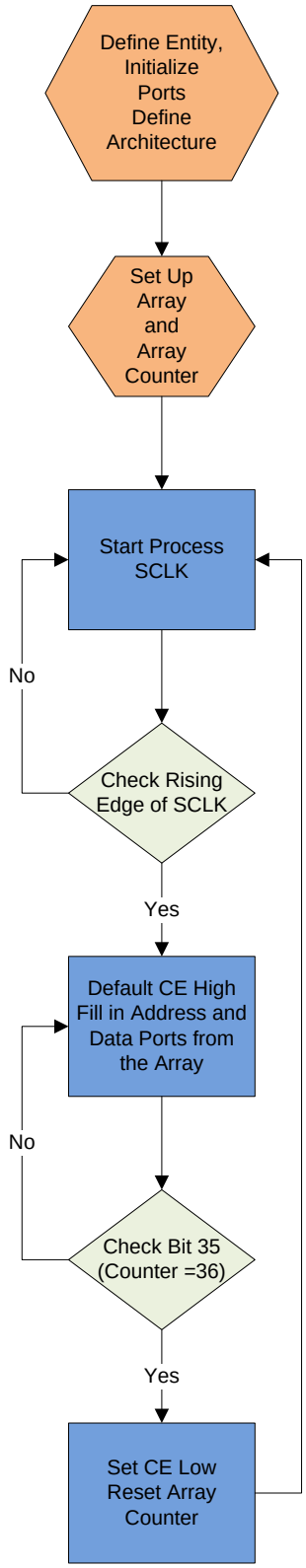


Figure 14: SPI CPLD Code Flow Chart

5.3.2 CPLD Simulation

For simulation purposes the data bits and the address bits are narrowed down to three bits each, thus it will be easier to view them. The simulation is shown in Appendix G: CPLD Simulation Part A and Appendix G: CPLD Simulation Part B. The input data for the simulation is MOSI_SPI and the serial clock is SCLK. These simulations show that as the clock ticks the address bus and the data bus gets filled. Initially the CE is held high, but once the counter reaches 9 the CE goes low, which means that data can be written to the specified address. Afterwards, the CE switches to high state and the counter is reset to zero on the next clock tick.

6. BIOS and Port Memory

Two Cyprus CY7C1051DV33 eight mega-bit SRAM is used for both Bios and Port memory. Even though SRAMs are more expensive than other types of memory, an SRAM is preferred for both buffering the Port80 –Port83 data and storing bios updates because it is faster than all other types of memory. The data bus for the SRAM is 16 bits wide and it is organized into 512K words [2]. The block diagram for the SRAM is shown in Figure 15.

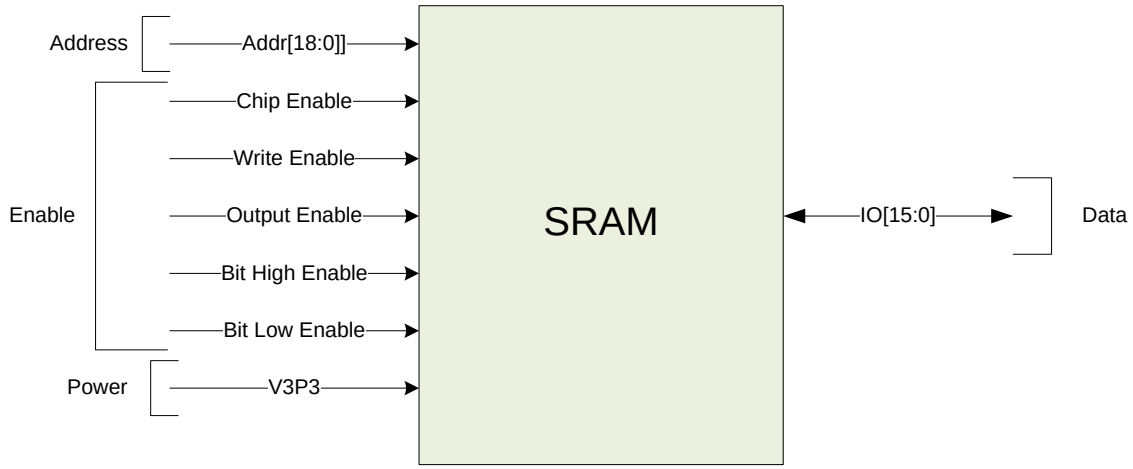


Figure 15: SRAM Block

In Figure 15, the inputs to SRAM are on the left side and the IO[15:0] is bidirectional, so it could be either inputs or outputs. A detailed description of the different inputs and outputs and the functionality of the SRAM memories are provided in the following sections.

6.1 BUFFRAM

BUFFRAM is used for buffering Port 80 – Port 83 data. It is connected to port 4 of the CPLD. The schematic for the BUFFRAM is demonstrated in Figure 16. The schematic also includes the two 0.1uF capacitors for the power rails.

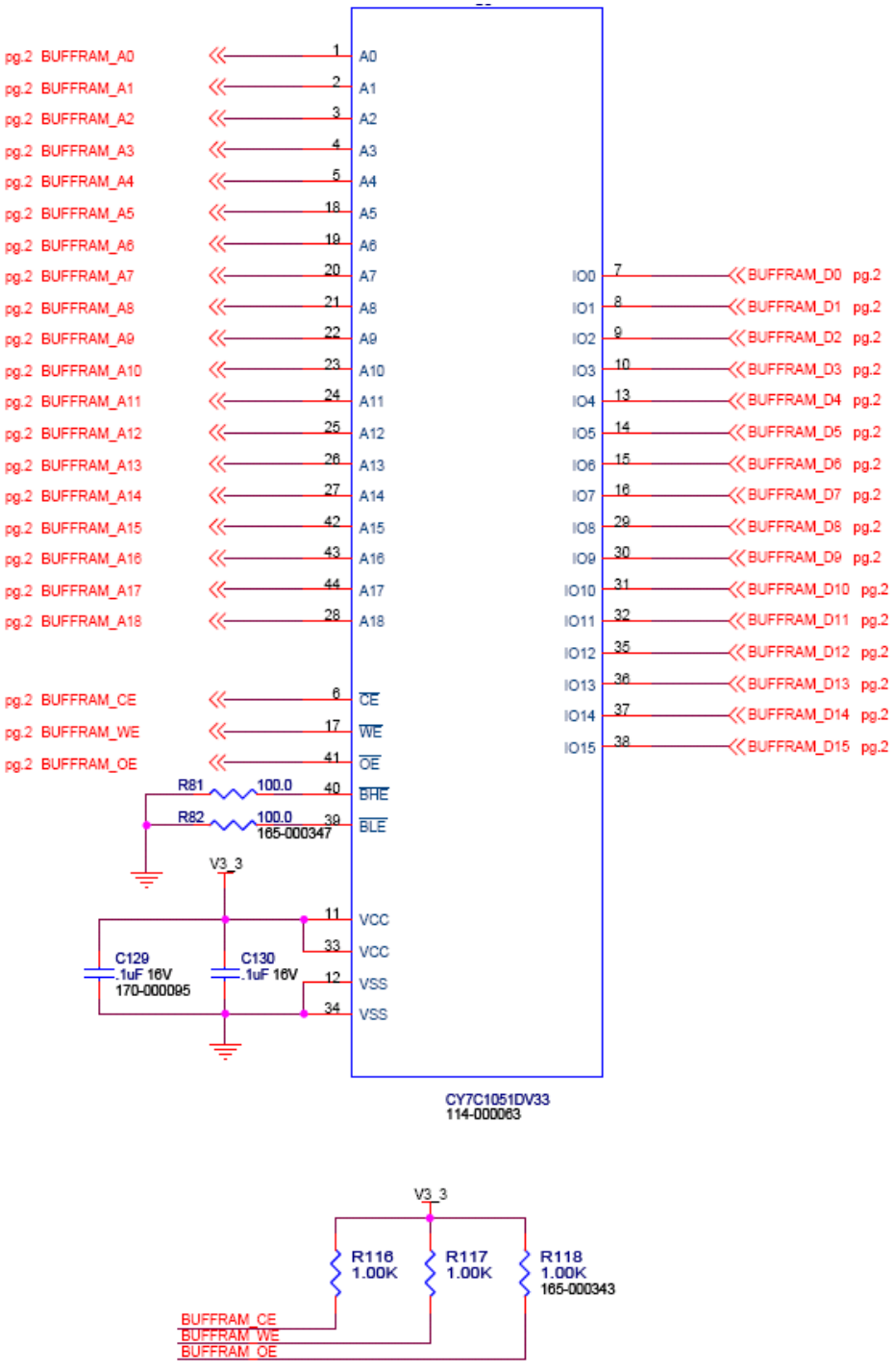


Figure 16: BUFFRAM Schematic

The table shown in Table 8 describes the pin-out of the SRAM.

Table 8: SRAM Pin-Out

Pin Name	Type	Pin Description
A0-A18	Input	Address Pins
IO0-IO16	Bidirectional	Data Pins
CE	Input	Chip Enable (Active Low)
WE	Input	Write Enable (Active Low)
OE	Input	Output Enable (Active Low)
BHE	Input	Bit High Enable (Active Low)
BLE	Input	Bit Low Enable (Active Low)
Vcc	Power	3.3V
Vss	Power	GND

The address, the data, the CE, the WE, the OE, the BHE and the BLE pins are connected to the I/O pins of the CPLD. CE pin enables the functionality of the SRAM and it requires an active low input to have the SRAM operational. The WE pin enables the user to write to the memory and the OE pin allows the SRAM to output the data. The BHE and the BLE pins will determine which I/O pin the data from the memory will appear on [2].

Table 9 shows the logic for BHE and BLE and which I/O pins are being used.

Table 9: BLE and BHE Truth Table

BLE Input	BHE Input	Data Pins Used
1	0	IO0-IO7
0	1	IO8-IO15
1	1	IO0-IO15

The debug board design uses all the I/O pins so both BLE and BHE is grounded through a 1KΩ resistor. The CE, the WE and the OE pins needed to be active low as well and it is controlled by the CPLD. By default the CE, the WE, and the OE pins are disabled or pulled up to Vcc, but CPLD will set these pins active low when the SRAM is being utilized.

6.2 ICERAM

ICERAM stores the bios updates from the MCU and it is connected to port 1 of the CPLD. For the most part ICERAM is configured the same manner as the BUFFRAM. The main difference is that it has a 1.5F super capacitor that provides the SRAM with 3.3V for a short duration of the time if the debug card needs to me removed from the motherboard and the main power is cut off. The design for this is shown in Figure 17.

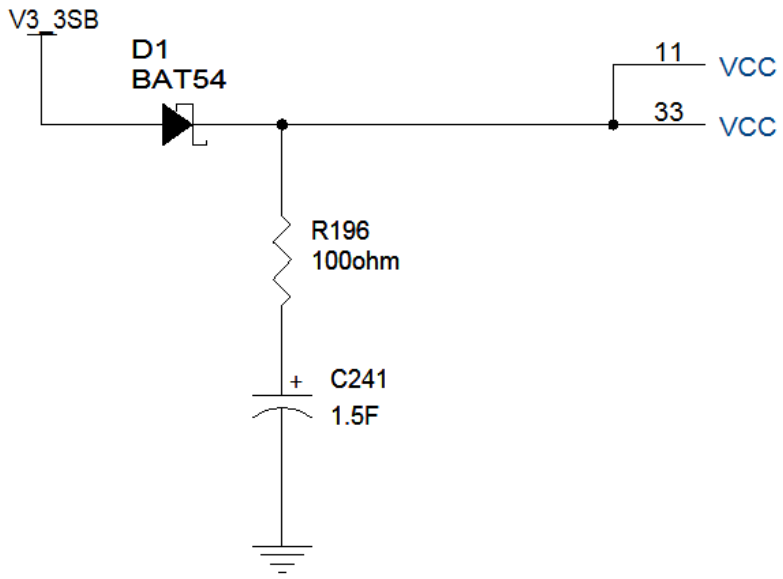


Figure 17: Super Capacitor Circuit

The capacitor charges up and holds the charge when the ICERAM is powered by 3.3V. When the power from the motherboard is cutoff, the super capacitor will discharge through the resistor and provides enough power for the SRAM. The 100Ω resistor limits the current going into the power pins and the Schottky diode prevents the current going into the positive power rail.

The schematic for the ICERAM is illustrated in Figure 18.

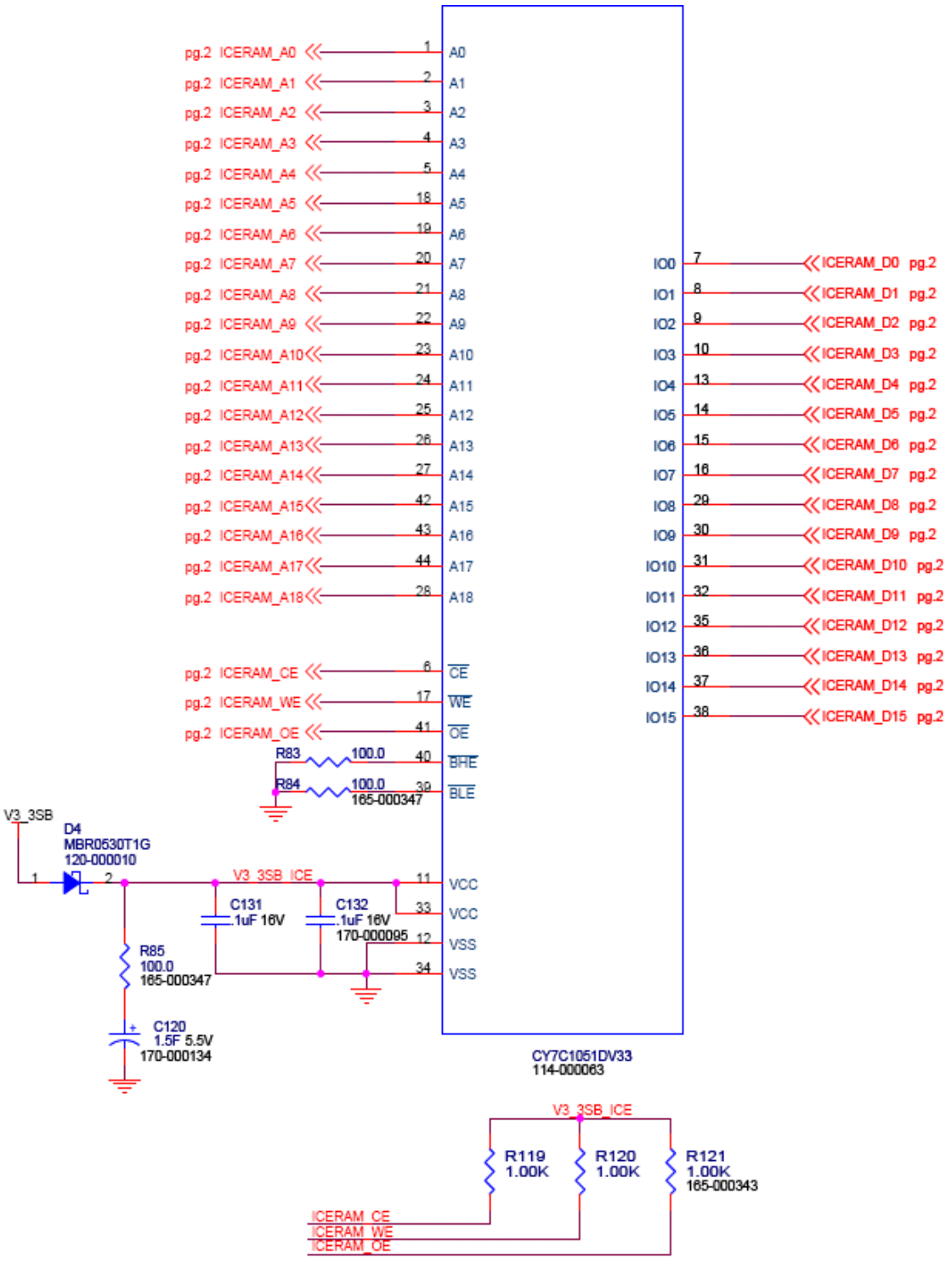


Figure 18: ICERAM Schematic

7. Ethernet

Implementing Ethernet communications enables the user to remotely connect to the server blade and perform the same tasks from a distant location as it where in front of a work bench. These tasks include updating BIOS and accessing memory for port 80-83 data. The Ethernet interface consists of four major parts, which are the Media Access Controller (MAC), the Ethernet PHY, the Magnetics, and the connector. The ATMEL microcontroller has an internal 10/100 Ethernet MAC interface with DMA and most of the RJ45 connectors integrate magnetic to its connectors. Therefore the Ethernet interface only requires an external PHY and an appropriate RJ-45 connector with magnetic [3]. The pins that are dedicated for Ethernet capabilities are shown in Table 10.

Table 10: Ethernet Pin out

Pin Name	Type	Pin Description
ENET_TX_EN	Output	Transmit Data Enable
ENET_TXD[1:0]	Output	Transmit Data (2 bits)
ENET_RXD[1:0]	Input	Receive Data (2 bits)
ENET_RX_ERR	Input	Receive Error
ENET_RXDV	Input	Carrier Sense/Receive Data Valid
ENET_REF_CLK	Input	Reference Clock
ENET_MDC	Output	MII Management Data Pin
ENET_MDIO	Bidirectional	MII Management Clock Pin

A more detailed explanation of the Ethernet PHY and the RJ45 connector is provided in the following sections.

7.1 Ethernet PHY

The ARM7 microcontroller can interface with Ethernet PHY through either Media Independent Interface (MII) or Reduced Media Independent Interface (RMII). RMII connection is preferred in this design since RMII support low pin counts and 100

Mb/s data rate. RMII connection requires a 50 MHz reference clock compared to a 25 MHz clock for the MII connection.

A Micrel KSZ8001L PHY chip is used in this design. The schematic for the Ethernet PHY is provided in Appendix C: Ethernet PHY Schematic.

The block diagram for the chip is obtained from the data sheet and is illustrated in Figure 19 [9].

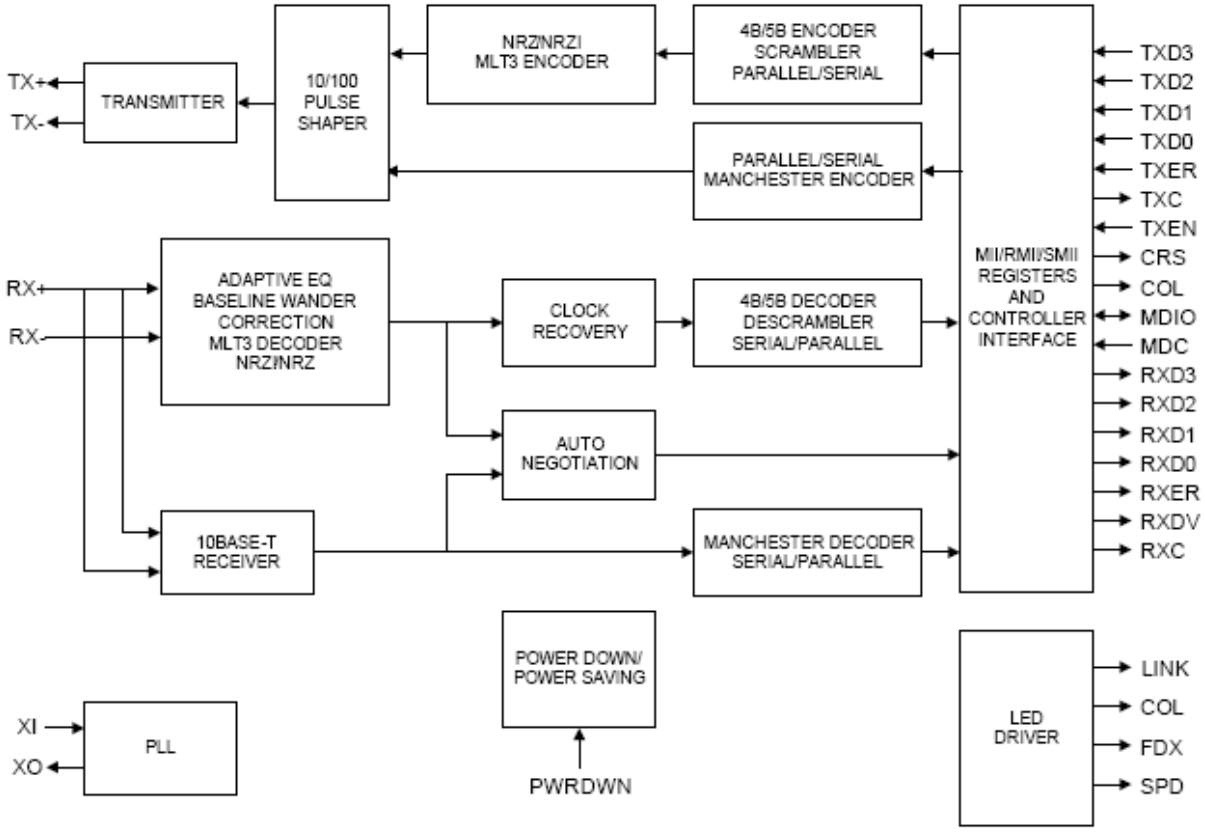


Figure 19: Micrel Block Diagram

The block diagram above shows all the input and output pins. Most of the pins in the MII/RMII/SMII REGISTERS AND CONTROLLER INTERFACE block correspond to the pins described in Table 10, with exception of TXD[2:3], RXD[2:3], TXER, COL, RXER and RXC. The ROM emulator is only transmitting and receiving a nibble or two bits of data at a time; thus TXD[2:3] and RXD[2:3] are not connected. Since the design implements RMII connection, TXER, RXC, and COL are not connected because these pins are associated with MII connection [3][9].

The TXC pin is where the reference clock needs to be inputted, which requires a 50 MHz clock signal. The 50 MHz signal must be inputted to the ENET_REF_CLK pin as well. The design employs a 50 MHz oscillator to generate the clock signal. A voltage divider resistor network is used to provide 50 MHz signal to both the TXC and the ENET_REF_CLK pins [3][9]. The schematic for this is shown in Figure 20.

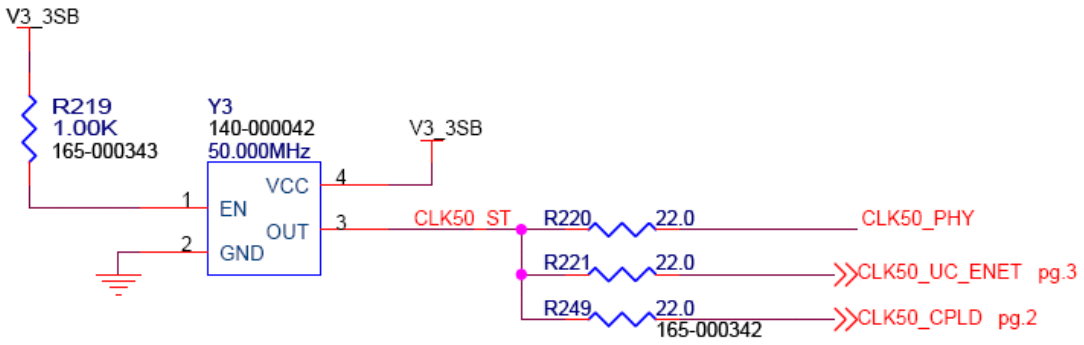


Figure 20: 50 MHz Oscillator

If the pin 1 of the oscillator is either left open or pulled high, then the output is set to oscillation. However, if pin 1 is pulled down then pin 3 will be in a high impedance state. Pin 1 has an internal pull up resistor if it is left unconnected, but it is good design practice to use an external pull resistor to input logic high for pin 1. The Vcc pin is connected to 3.3V and a decoupling capacitor is added to reduce noise. The output of the Oscillator is split into two signals through a pair of 1kΩ resistors.

The PLL block requires a 25 MHz clock input, which can be generated through either an external 25 MHz oscillator, 25 MHz Crystal, or use the 50 MHz reference clock signal. Since the first two methods require additional part, the third method is more efficient. The Micrel data sheet recommends the PLL configuration shown in Figure 21 for the third method [9].

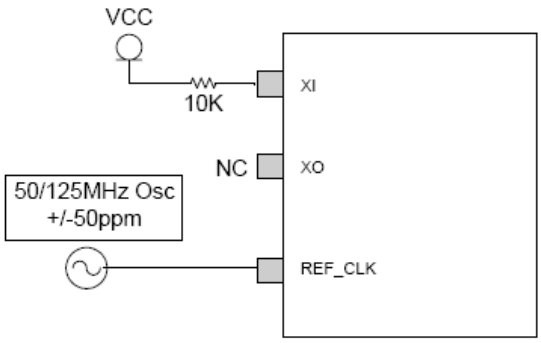


Figure 21: Micrel PLL Configuration

The Figure 19: Micrel Block Diagram does not show the power pins, but the PHY chip requires 3.3V and 1.8V supply. The Micrel chip will internally regulate the 1.8V from the 3.3V supply. The recommend power configuration is provided by the Micrel datasheet and is employed in this design. The recommended power configuration is shown in Figure 22.

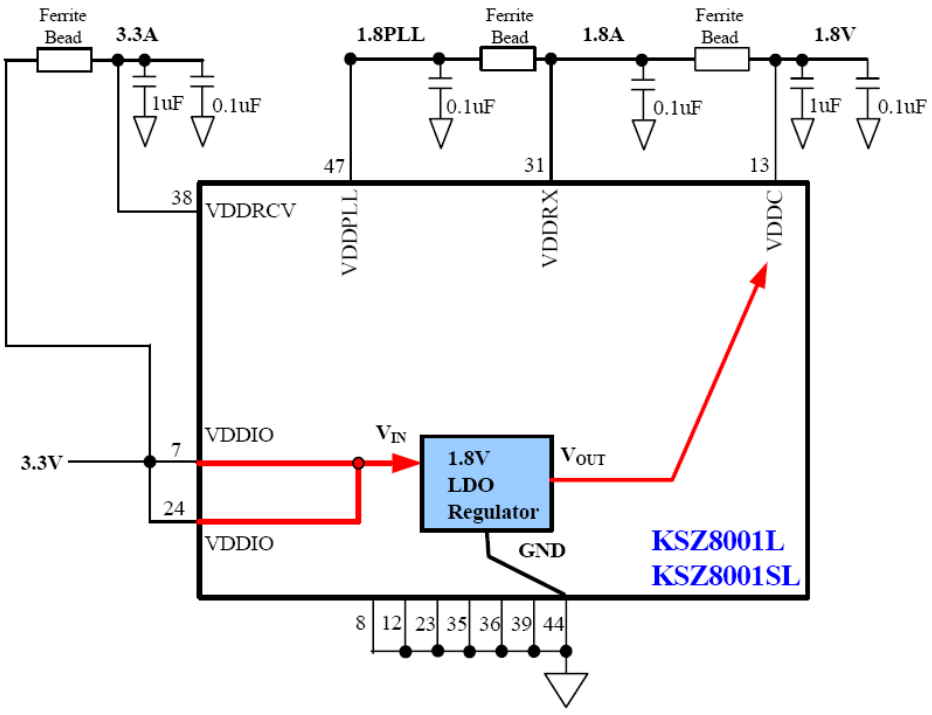


Figure 22: Micrel Power Configuration

The remaining Micrel pins are dedicated to Receiving, Transmitting, and LED Driver blocks, which will be discussed further in the next section [9].

7.2 Ethernet Connector

The Ethernet Connector section consists of RJ45 Connector and the USB connector because it is manufactured in the same package. Currently the parts are separate but the final part will be one unit. The reason for choosing one package is to conserve board space. Even though the part numbers will be different in the final design, the pin connections are essentially the same. The RJ45 Connector and the USB Connector is discussed further in the following sections.

7.2.1 RJ45

The pins from the receiver and transmitter block in Figure 19 are directly connected to the RJ45 connector. The differential transmit signals from the Ethernet PHY will connect to the receive pins of the RJ45 and the differential transmit signals from the RJ45 will connect to the receive pin on the Ethernet PHY.

The RJ45 connector has three internal LEDs that are green, yellow and orange. The yellow LED identifies if the connection is 10BASE-T or 100BASE-T. The yellow LED will be on for the 100BASE-T connection. The green LED turns on when an Ethernet connection is established. The LED block in the PHY block diagram controls the LEDs functions. The third LED in the RJ45 connector is on when there are no Ethernet connections. The schematic for the RJ45 connector is shown in Figure 23.

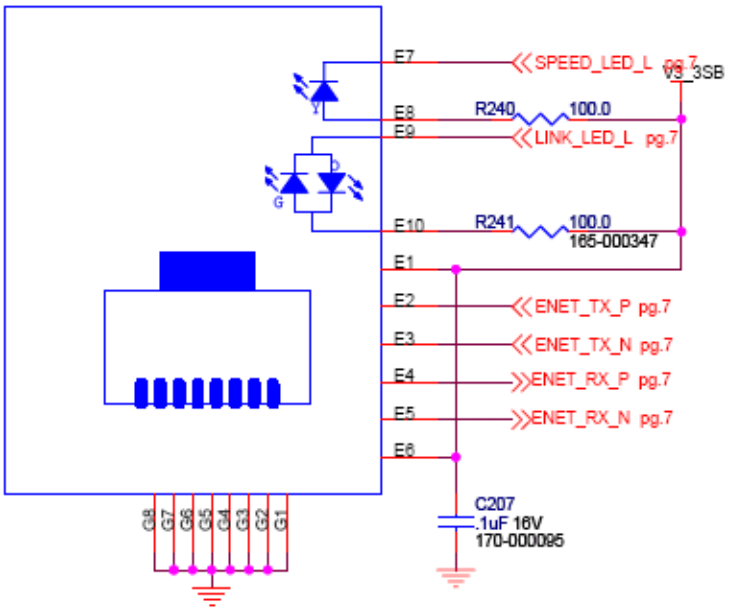


Figure 23: RJ45 Connector Schematics

The transmit/receive signal pairs require proper termination. Conventionally a 49.9 Ω pull-up resistor is used for Ethernet termination.

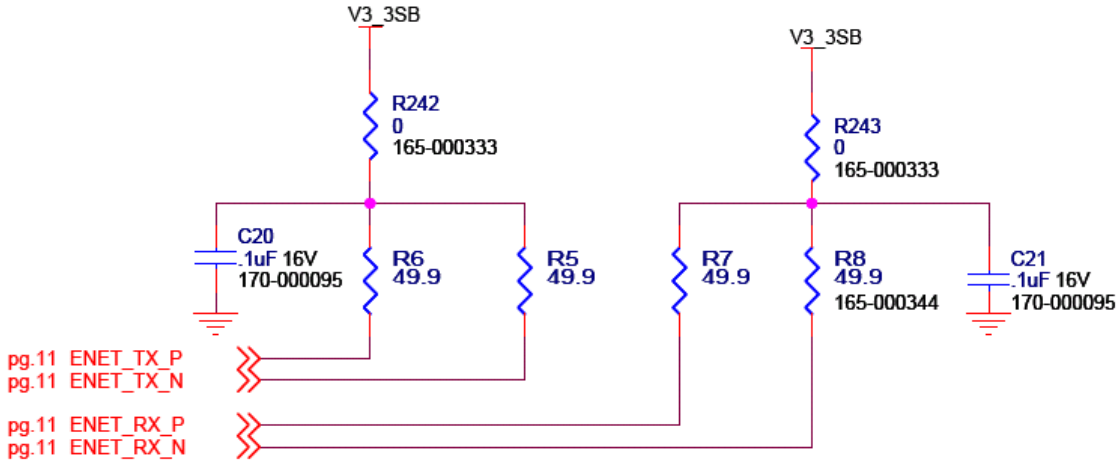


Figure 24: Ethernet Termination

7.3 External MAC Memory

Two kilobyte of external EEPROM memory is added to the MCU for storing the MAC address. The ROM emulator is connected to the EEPROM through an I^2C bus [15]. The following the approach shown in Figure 25 is used to access the data in EEPROM.

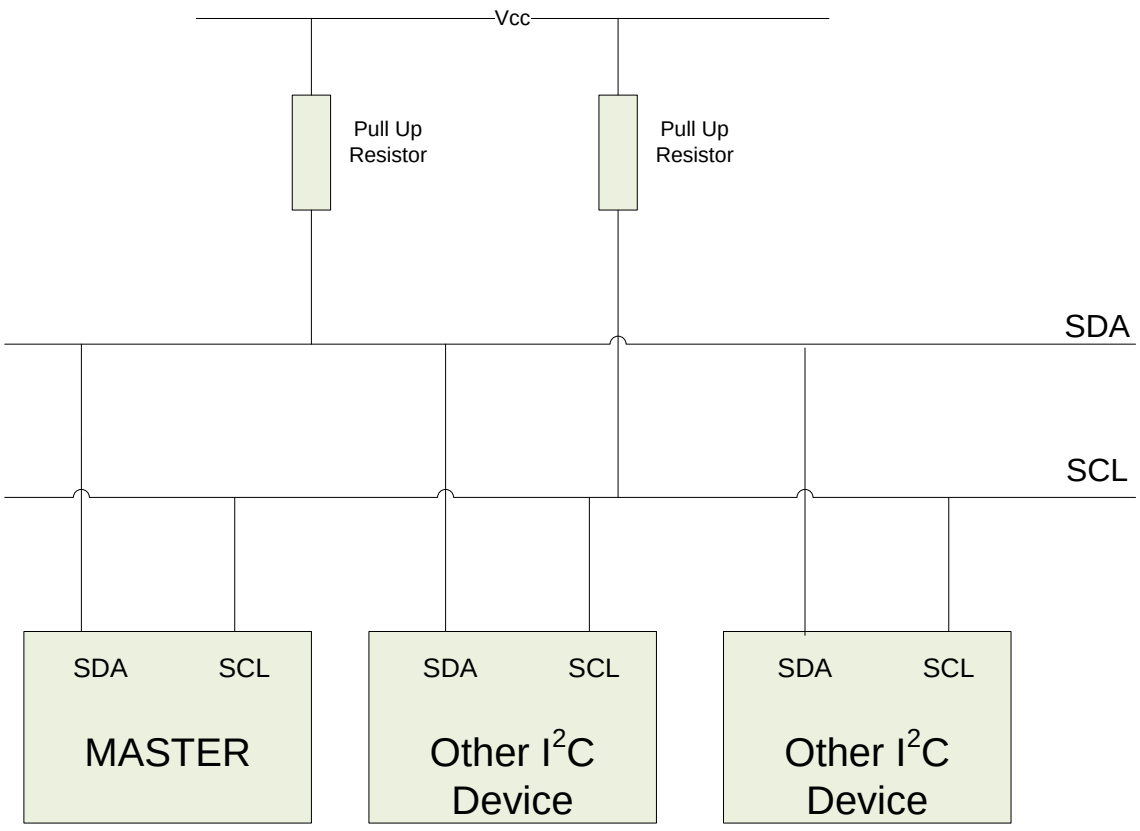


Figure 25: I2C Bus Interface

An Atmel AT24C02A EPPROM memory is used in this design. In addition to the SDA and SCL pins, the EEPROM have three address pins (A3-A0), a write protection pin (WP), a power pin and a ground pin. The address pins, A3 through A0 are hardwired to Vcc. The WP pin is tied to ground to enable read/write, and the power pin is connected to 3.3V [15]. The schematic for the external EEPROM is displayed in Figure 26.

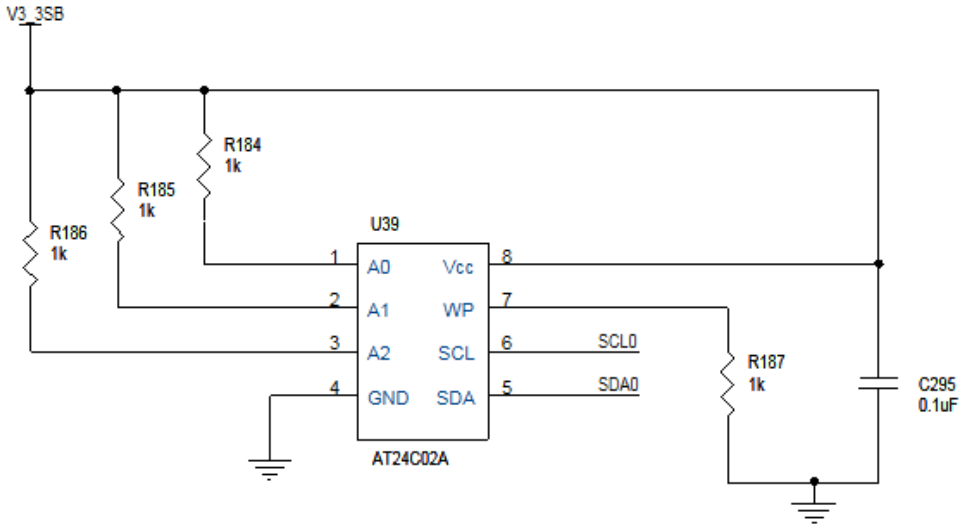


Figure 26: EEPROM Memory Circuit

8. Debug Connector

The debug Connector is an array of pins that connects to the host server blades. Table 11 shows the different signals and the functions that are associated with the signals.

Table 11: Debug Connector Signals

Function	Signal
PCI	PCI_AD[31:0]
	PCI_CBEL[3:0]
	PCI_GNTL
	PCI_REQL
	PCI_IRQL
	PCI_RSTL
	PCI_PAR
	PCI_PERRL
	PCI_SERRL
	PCI_DEVSELL
	PCI_FRAMEL
	PCI_IRDYL
	PCI_TRDYL
	PCI_STOPL
PCI-E	PCIE_TXP
	PCIE_TXN
	PCI_RXP

	PCI_RXN
	CLK100_P
	CLK100_N
SATA	SATA0_TXP
	SATA0_TXN
	SATA0_RXP
	SATA0_RXN
	SATA1_TXP
	SATA1_TXN
	SATA1_RXP
	SATA1_RXN
VIDEO	V_BLUE
	V_RED
	V_GREEN
	V_DDCDAT
	V_DDCCLK
	V_HSYNC
	V_SYNC
I2C	I2C0_SCL
	I2C0_SDA
	I2C1_SCL
	I2C1_SDA
	I2C2_SCL
	I2C2_SDA
	FRU_SCL
	FRU_SDA
USB	USB_OC0_L
	USB_OC1_L
	USB_OC2_L
	USB_OC3_L
	USB_P0P
	USB_P0N
	USB_P1P
	USB_P1N
	USB_P3P
	USB_P3N
	USB_P2P
	USB_P2N
PUSH BUTTONS	PB_NMIL
	PB_PWRL
	PB_RSTL

	PB_BMC_RSTL
DIP SWTICHES	DBG_GPIO0
	DBG_GPIO1
	DBG_GPIO2
	DBG_GPIO3
	DBG_GPIO4
	DBG_GPIO5
	DBG_GPIO6
	DBG_GPIO7
KEYBOARD	KBCLK
	KBDATA
MOUSE	MSCLK
	MSDATA
SPI INTERFACE HOST TO CPLD	HST_SPI_CLK
	HST_SPI_CS
	HST_SPI_MISO
	HST_SPI_MOSI
LPC INTERFACE HOST TO CPLD	LPC_AD0
	LPC_AD1
	LPC_AD2
	LPC_AD3
	LPC_FRAMEL
	LPC_INITL

The schematic for the debug connector is shown in Appendix D: Debug Connector Schematic Part A, Appendix D: Debug Connector Schematic Part B and Appendix D: Debug Connector Schematic Part C

9. USB Connector

USB Type A connections are required for the debug board. The USB ports enable the user to connect hard drives, thumb drives and other peripherals to the server blades. The differential signal for the USB is coming from a USB hub inside the blade servers. Thus, the connections directly go to the debug connector.

There are only four pins for each of the USB ports. Two of the pins are for Vcc and ground. The remaining two are the differential pair signals used for data transfers. The circuitry for the USB connections is based off the prior debug board design, since the USB connections are configured the same way. The schematic is shown in Figure 27 for the USB connections.

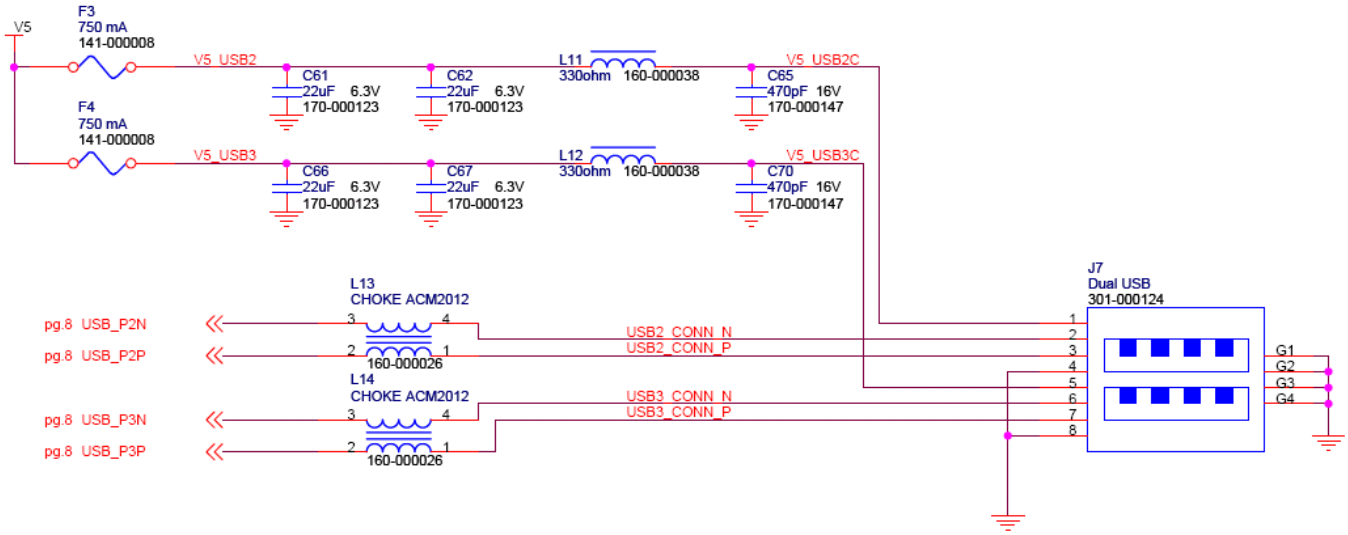


Figure 27: USB Connector Schematic

The numerous bypass capacitors and the small ferrite beads are place between the power rails to reduce noise spikes and to prevent any damage to the component. A choke inductor is also added to the differential pairs signal for the data to block any AC noise.

10. GPIO Switches and Push Buttons

The GPIO switches and the push buttons are designed to either to initiate a specific operation or to program for a desirable operation. They are discussed in further details in the following section.

10.1 Push Button Controls

There are four push buttons controls, which triggers system reset, BMC Reset, administers Power Control, and conduct NMI. The System Reset button simply reset the server. Similarly the BMC Reset resets the BMC chip. The power control button turns the blades on or off and the NMI button forces a Non-Maskable Interrupt to be run on the system to fix faulty code. When a NMI is initiated, it will execute an interrupt handler to redirect the controls to a system monitor program, where the user can inspect the system memory. These controls can also be remotely controlled through the MCU. The schematic for the push button is shown in Figure 28. Since the inputs need to be active low, a pull up resistor network is used. Thus, when the buttons are pushed or a logic low is inputted to the appropriate GPIO pins through the MCU, the desired push button controls are triggered.

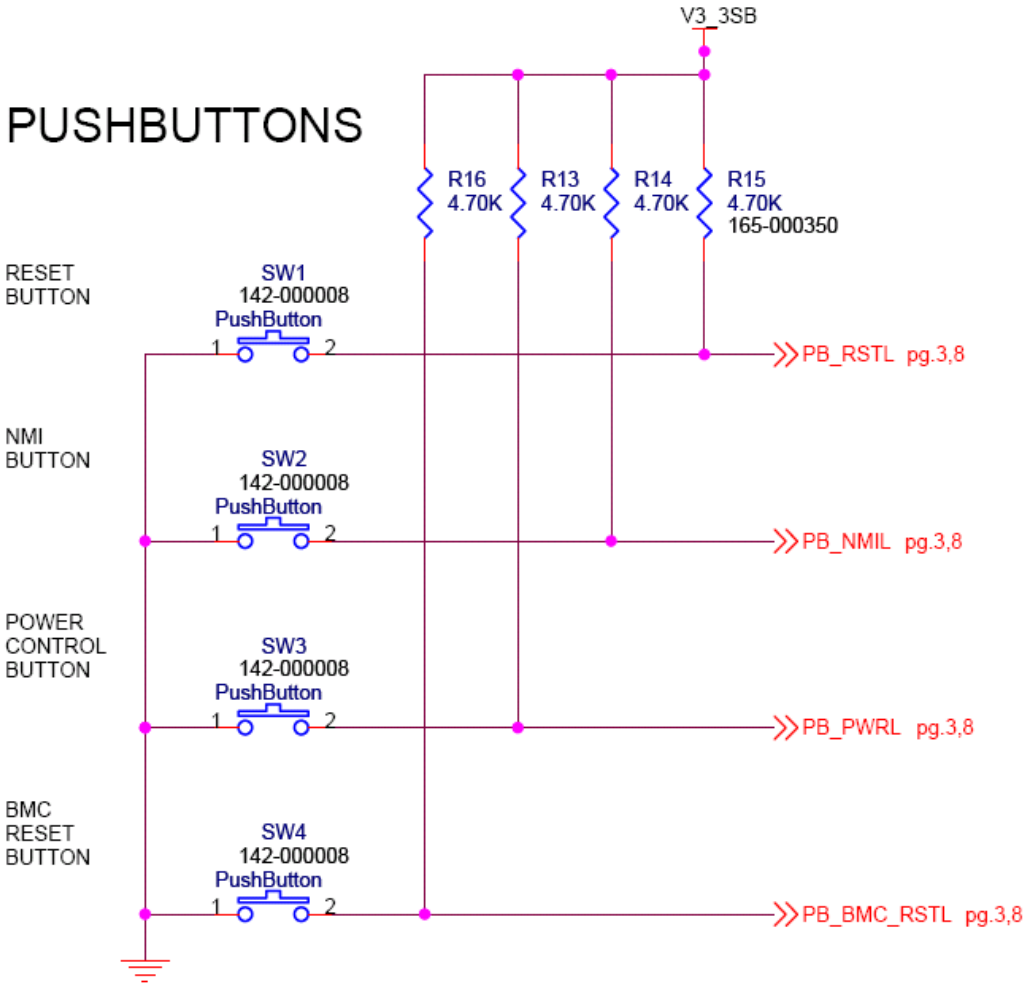


Figure 28: Push Button Schematic

10.2 GPIO Switches

The General Purpose Input/Output (GPIO) switches are implemented to be used at user's consent. The switches are connected to the debug connector and the MCU. Thus, the user can physically control the state of the input by toggling the DIP switches or remotely forcing logic low through the GPIO of the MCU. Similar to the push buttons, pull up resistors are used to default to a logic high state when the switches are closed. The schematic for the GPIO DIPs are given in Figure 29.

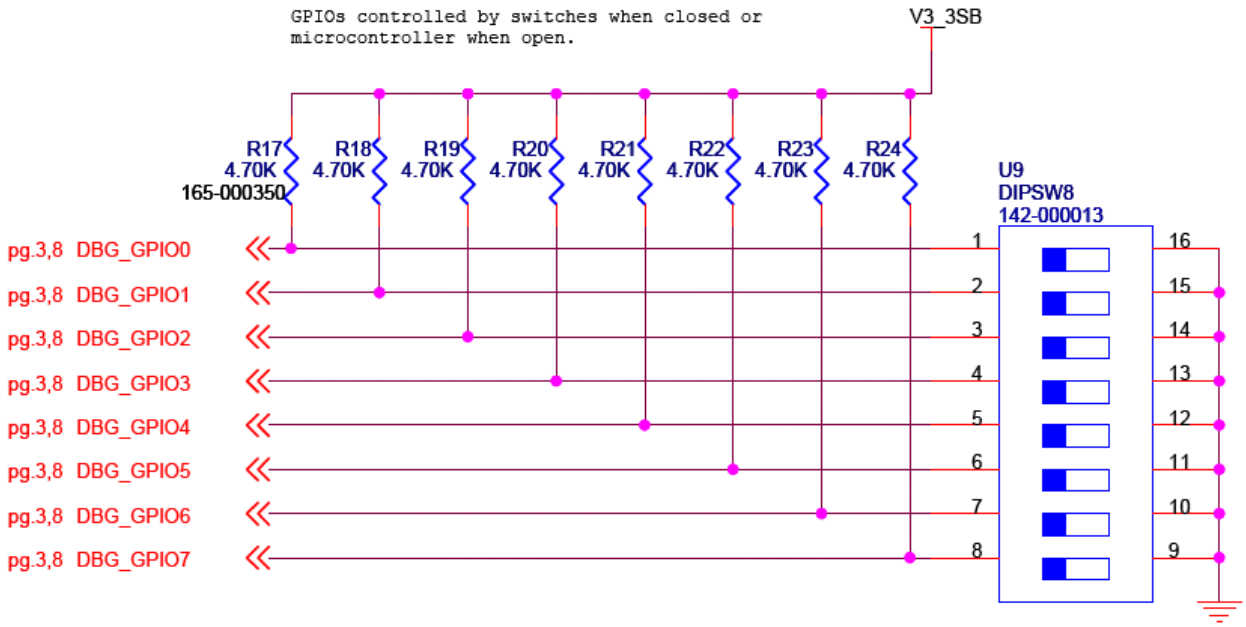


Figure 29: GPIO Schematic

11. PCI Connector

The PCI Connector is configured by integrating the signals under PCI Connector in Table 11 with the standard 32 bit PCI connector pin out. The PCI connector is divided into power signals, JTAG signals, Address/Data signals, Interrupt Signal, System Bus, Command/Byte Enable signals, Error Bus, Reserved Signals, 64 MHz Control Signals, Parity Bus signals and other Miscellaneous Signals. The power signals consist of 5V, 3.3V, 3.3V standby (3.3VSB) and -12V. The JTAG signals are disabled by biasing signals through a 2.49 KΩ to either the 5V power rail or the GND. The PCI standard specifies to use a 2.49 KΩ resistor for either pulling up or pulling down a signal.

Since this is a 32 bit PCI connector it has 32 bit Address/Data pins. The PCI connector has four interrupt pins, but all fours are connected to one interrupt event from the debug connector. The system bus entails of the 33 MHz clock signal and the reset signal. There are four Command/Byte Enable signals and two Error Bus signals [7].

Some of the pins are reserved so there are not connected and all the 64 MHz Control signals are disabled because those signals are dedicated for 64 bit PCI connector. The Table 12 matches up the 32 Bit PCI connector pin out with the signals drawn from the debug connector. Different color codes are assigned to different types of signals.

Table 12: PCI Connector Pins

Power	Command/Byte Enable
JTAG	Error Bus
Address/Data	Reserved/NC
Interrupt	64 MHz Control
System Bus	Partity Bus
Misc Signals	

PIN #	Pin Name	Pin Description	Signal From Debug Connector
A1	TRST	Test Logic Reset	GND[1]
A2	+12V	+12 VDC	V12
A3	TMS	Test Mode Select	V5[2]
A4	TDI	Test Data Input	V5[2]
A5	+5V	+5 VDC	V5
A6	INTA	Interrupt A	PCI_IRQL
A7	INTC	Interrupt C	PCI_IRQL
A8	+5V	+5 VDC	V5
A9	-----	Reserved	NC
A10	+5V	Power (+5 V or +3.3 V)	V5
A11	-----	Reserved	NC
A12	GND03	Ground or Keyway for 3.3/Universal PWB	GND
A13	GND05	Ground or Keyway for 3.3/Universal PWB	GND
A14	3.3Vaux	-----	V3_3SD
A15	RESET	Reset	PCI_RSTL
A16	+5V	Power (+5 V or +3.3 V)	V5
A17	GNT	Grant PCI use	PCI_GNTL
A18	GND08	Ground	GND
A19	PME#	Power Managment Event	V3_3SB[1]
A20	AD30	Address/Data 30	PCI_AD30
A21	+3.3V01	+3.3 VDC	V3_3
A22	AD28	Address/Data 28	PCI_AD28
A23	AD26	Address/Data 26	PCI_AD26
A24	GND10	Ground	GND
A25	AD24	Address/Data 24	PCI_AD24
A26	IDSEL	Initialization Device Select	PCI_IDSEL[3]
A27	+3.3V03	+3.3 VDC	V3_3
A28	AD22	Address/Data 22	PCI_ADD22
A29	AD20	Address/Data 20	PCI_ADD20
A30	GND12	Ground	GND
A31	AD18	Address/Data 18	PCI_ADD18
A32	AD16	Address/Data 16	PCI_ADD16
A33	+3.3V05	+3.3 VDC	V3_3
A34	FRAME	Address or Data phase	PCI_FRAMEL

1. Pull-down Resistor is used to disable the pin
2. Pull-up Resistor is used to disable the pin
3. DIP switches are used to select a specific address

A35	GND14	Ground	GND
A36	TRDY#	Target Ready	PCI_TRDY
A37	GND15	Ground	GND
A38	STOP	Stop Transfer Cycle	PCI_STOPL
A39	+3.3V07	+3.3 VDC	V3_3
A40	-----	Reserved	NC
A41	-----	Reserved	NC
A42	GND17	Ground	GND
A43	PAR	Parity	PCI_PAR
A44	AD15	Address/Data 15	PCI_AD15
A45	+3.3V10	+3.3 VDC	V3_3
A46	AD13	Address/Data 13	PCI_AD13
A47	AD11	Address/Data 11	PCI_AD11
A48	GND19	Ground	GND
A49	AD9	Address/Data 9	PCI_AD9
A50	Keyway	Open or Ground for 3.3V PWB	NC
A51	Keyway	Open or Ground for 3.3V PWB	NC
A52	C/BE0	Command, Byte Enable 0	PCI_CBEL0
A53	+3.3V11	+3.3 VDC	V3_3
A54	AD6	Address/Data 6	PCI_AD6
A55	AD4	Address/Data 4	PCI_AD4
A56	GND21	Ground	GND
A57	AD2	Address/Data 2	PCI_AD2
A58	AD0	Address/Data 0	PCI_AD0
A59	+5V	Power (+5 V or +3.3 V)	V5
A60	REQ64	Request 64 bit	V5[2]
A61	VCC11	+5 VDC	V5
A62	VCC13	+5 VDC	V5
B1	-12V	-12 VDC	VNEG12
B2	TCK	Test Clock	GND[1]
B3	GND	Ground	GND
B4	TDO	Test Data Output	NC
B5	+5V	+5 VDC	V5
B6	+5V	+5 VDC	V5
B7	INTB	Interrupt B	PCI_IRQL
B8	INTD	Interrupt D	PCI_IRQL
B9	PRSNT1	Present	GND
B10	-----	Reserved	NC
B11	PRSNT2	Present	GND
B12	GND	Ground or Keyway for 3.3/Universal PWB	GND
B13	GND	Ground or Open (Key) for 3.3/Universal PWB	GND

B14	RES	Reserved	NC
B15	GND	Ground	GND
B16	CLK	Clock	CLK33_PCI
B17	GND	Ground	GND
B18	REQ	Request	PCI_REQL
B19	+5V	Power (+5 V or +3.3 V)	V5
B20	AD31	Address/Data 31	PCI_AD31
B21	AD29	Address/Data 29	PCI_AD29
B22	GND	Ground	GND
B23	AD27	Address/Data 27	PCI_AD27
B24	AD25	Address/Data 25	PCI_AD25
B25	+3.3V	+3.3VDC	V3_3
B26	C/BE3	Command, Byte Enable 3	PCI_CBEL3
B27	AD23	Address/Data 23	PCI_AD_23
B28	GND	Ground	GND
B29	AD21	Address/Data 21	PCI_AD21
B30	AD19	Address/Data 19	PCI_AD19
B31	+3.3V	+3.3 VDC	V3_3
B32	AD17	Address/Data 17	PCI_AD17
B33	C/BE2	Command, Byte Enable 2	PCI_CBEL2
B34	GND13	Ground	GND
B35	IRDY#	Initiator Ready	PCI_IRDYL
B36	+3.3V06	+3.3 VDC	V3_3
B37	DEVSEL	Device Select	PCI_DEVSELL
B38	GND16	Ground	GND
B39	LOCK#	Lock bus	V5[2]
B40	PERR#	Parity Error	PCI_PERRL
B41	+3.3V08	+3.3 VDC	V3_3
B42	SERR#	System Error	PCI_SERRL
B43	+3.3V09	+3.3 VDC	V3_3
B44	C/BE1	Command, Byte Enable 1	PCI_CBEL1
B45	AD14	Address/Data 14	PCI_AD14
B46	GND18	Ground	GND
B47	AD12	Address/Data 12	PCI_AD12
B48	AD10	Address/Data 10	PCI_AD10
B49	GND20	Ground	GND
B50	Keyway	Open or Ground for 3.3V PWB	NC
B51	Keyway	Open or Ground for 3.3V PWB	NC
B52	AD8	Address/Data 8	PCI_AD8
B53	AD7	Address/Data 7	PCI_AD7
B54	+3.3V12	+3.3 VDC	V3_3

B55	AD5	Address/Data 5	PCI_AD5
B56	AD3	Address/Data 3	PCI_AD3
B57	GND22	Ground	GND
B58	AD1	Address/Data 1	PCI_AD1
B59	VCC08	Power (+5 V or +3.3 V)	V5
B60	ACK64	Acknowledge 64 bit	V5[2]
B61	VCC10	+5 VDC	V5
B62	VCC12	+5 VDC	V5

The schematic for the PCI connector is shown in Figure 30. There are several 0.1uF bypass capacitors used to reduce noise.

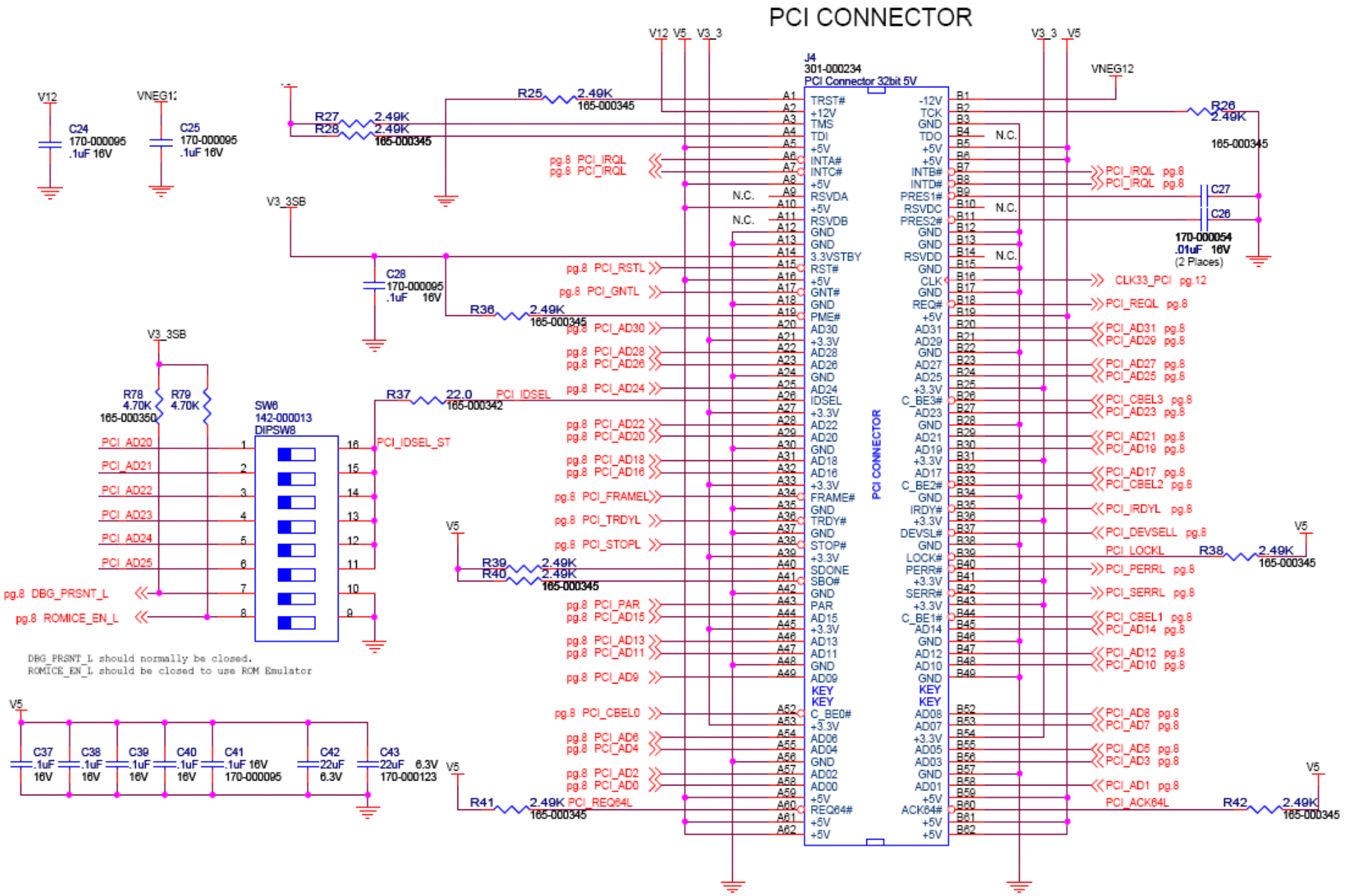
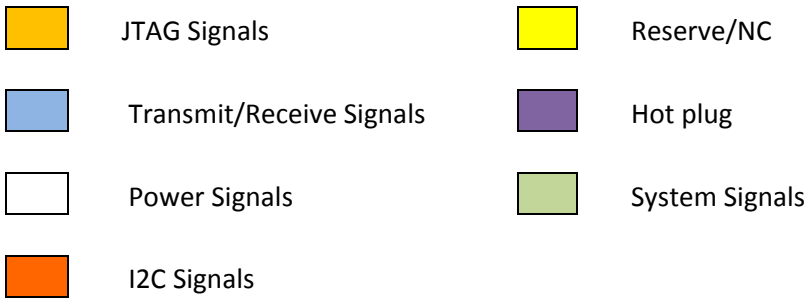


Figure 30: PCI Connector

12. PCI-E Connector

The PCI-E is a one lane point to point header. The PCI-E has unidirectional transmit and receive pair signals. Since the PCI_E is only one lane the connector is compact and the number of signals need is fewer than the PCI connector. The PCI –E connector consist of JTAG signals, one Transmit/Receive Signal pairs, Power Signals, I^2C signals, Reserve Signals, Hot plug signals and the System signals [6]. The JTAG signals and the I^2C signals are disabled by biasing them to the power rails. The PCI-E standards require a 4.7 KΩ for biasing the signals. Table 13 explains the pin-outs for the PCI-E connector.

Table 13: PCI-E Connector Pin Outs



Pin #	Pin Name	Description	Signal from Debug Connector
A1	PRSNT#1	Hot plug presence detect	PCIE_PRST_L
A2	+12v	+12 VDC	V12
A3	+12v	+12 VDC	V12
A4	GND	Ground	GND
A5	JTAG2	TCK	GND[1]
A6	JTAG3	TDI	V3_3[2]
A7	JTAG4	TDO	V3_3[2]
A8	JTAG5	TMS	V3_3[2]
A9	+3.3v	+3.3 VDC	V3_3
A10	+3.3v	+3.3 VDC	V3_3
A11	PWRGD	Power Good	PWRGD_HOST
A12	GND	Ground	GND
A13	REFCLK+	Reference Clock	CLK100_P
A14	REFCLK-	Differential pair	CLK100_N

A15	GND	Ground	GND
A16	HSIp(0)	Receiver Lane 0,	PCIE_RXP
A17	HSIn(0)	Differential pair	PCIE_RXN
A18	GND	Ground	GND
B1	+12V	+12VDC	V12
B2	+12V	+12VDC	V12
B3	RSVD	Reserved	NC
B4	GND	Ground	GND
B5	SMCLK	SMBus clock	V3_3[2]
B6	SMDAT	SMBus data	V3_3[2]
B7	GND	Ground	GND
B8	+3.3V	+3.3VDC	V3_3
B9	JTAG1	+TRST#	GND2[1]
B10	3.3Vaux	----------	V3_3SB
B11	WAKE#	Link Reactivation	NC
B12	RSVD	Reserved	NC
B13	GND	Ground	GND
B14	HSOp(0)	Transmitter Lane 0,	PCIE_TXP
B15	HSOn(0)	Differential pair	PCIE_TXN
B16	GND	Ground	GND
B17	PRSNT#2	Hot plug detect	GND
B18	GND	Ground	GND

1. Pull down resistor is used to disable the pin
2. Pull up resistor is used to disable the pin

13. SATA

There are two SATA headers on the debug board. The SATA connector consists of a differential transmit and receive signals. These signals are drawn from the debug connector. The schematic for the SATA connector is shown in Figure 31.

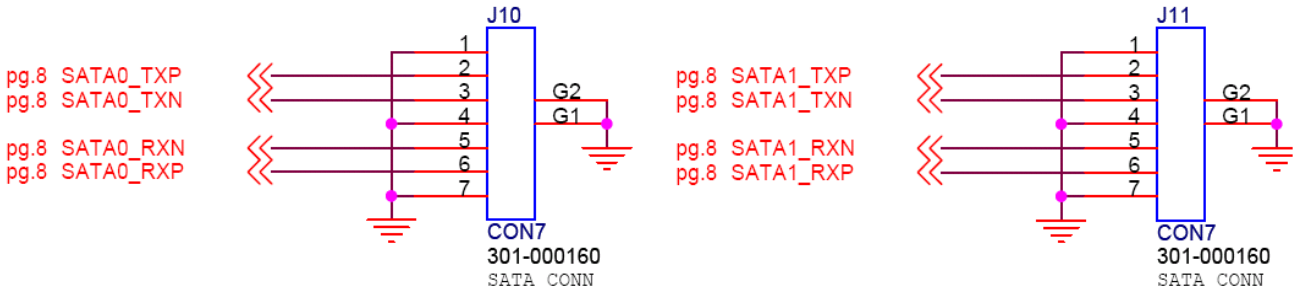


Figure 31: SATA Connector Schematic

14. Power Supply

The debug board design requires a 5V, a 3.3V, a 1.8V, a 3.8VSB and a -12V supply. A 12V and 5VSB DC supply is drawn from the motherboard and it has to be regulated.

14.1 Step-Down Converter for 5V and 3.3V

Two MICREL power management and conversion ICs designed my Zilker Laboratory are used for the 5V and 3.3V regulation because Egenera would like to experiment with the MICREL power management and conversion ICs. If this regulator works well then this product can be integrated into future designs. The Zilker laboratory provides their customers with template circuit designs for the required voltage conversions. The voltage conversion template circuit for the 5V and 3.3V are provided in Appendix H: 5V Regulator and Appendix I: 3.3V Regulator respectively. Even though these regulators will output the necessary voltage for the debug board, further implementation of the circuit is required to integrate them to the debug board [16].

Before making these implementations and a general idea of the regulator is needed. The schematics in Appendix H: 5V Regulator and Appendix I: 3.3V Regulator may look complicated, but the concept is very simple. The Zilker laboratory utilizes a buck converter design and the connections for the main components are illustrated in Figure 32.

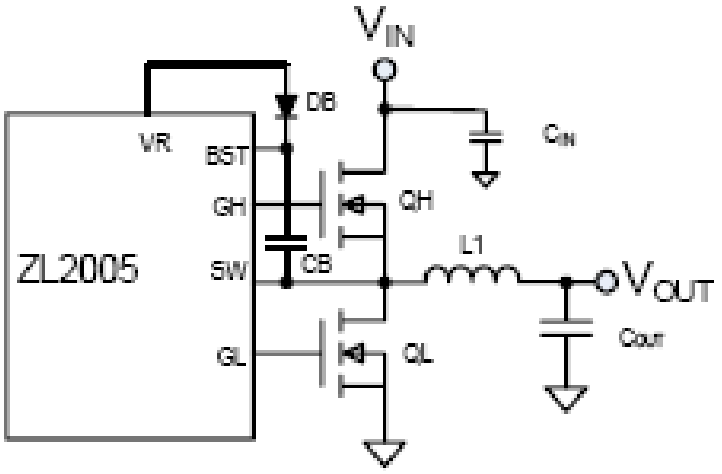


Figure 32: Step-Down Converter

The design requires two MOSFETS, QH and QL. The two MOSFET will inversely turn ON and OFF depending on the duty cycle, V_{OUT}/V_{IN}. When QH is on, the current flowing through the inductor will climb up and the drop across the inductor L1 will be $V_{IN} - V_{OUT}$. When QH switches off, the current will drop; however the current will still flows through the QL MOSFET [16]. This relationship is illustrated in Figure 33

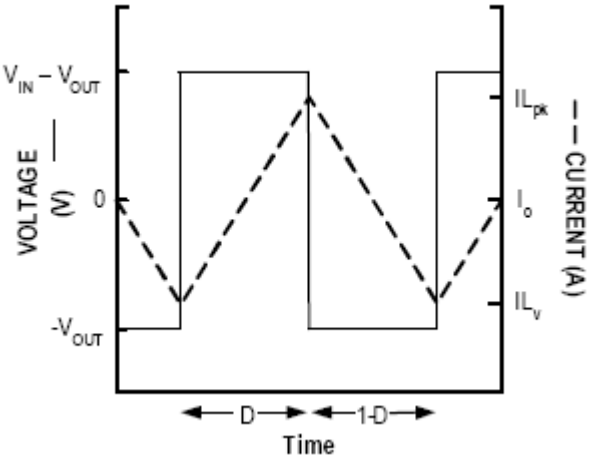


Figure 33: Inductor Wave Form

The switching frequency causes a low impedance state in the Cout capacitor and it removes the AC components and provides the load with a DC voltage. The value of Cout and L1 depends on how efficient the design should be. Smaller values of capacitance and inductance increases the switching frequency, thus power efficiency suffers a little bit.

The Table 14 shows the connection to V0 and V1 which will produce the desired Vout. For a 5V regulator, both V0 and V1 must be connected to V25 pin, which is the internal 2.5 reference voltage pin. For a 3.3V supply the V0 pin is not connected and the V1 pin is pulled high with the V25 pin. The EN pin is also connected to V25 pin through a 10 kΩ pull up resistor to enable the chip [16].

Table 14: V_{OUT} Configuration Table

		V0		
		Low	Open	High
V1	Low	0.6 V	0.8 V	1.0 V
	Open	1.2 V	1.5 V	1.8 V
	High	2.5 V	3.3 V	5.0 V

Since there are two Micrel Chips being used, the SCL and SDA pins are interconnected through an IS2 bus for autonomous sequencing, so the serial data and the clock will be synchronized.

14.2 Integrating MICREL schematics to the Debug Board Design

Some of the pins are connected to the MCU's I/O pins. These pins are the enable pins, margin pins, power good pins and the I^2C signals for both of the regulators. Since the MCU is powered through standby power it can disable the 5V and 3.3 V supply if the power good signals reports an error. The remaining pins are connected to both the power rails through depopulated resistors. This means that the traces for the pins are available and the pins could be biased to either high or low if needed. The reason for such design is that the MICREL chip's design was not optimized because of time constraint. Thus, by making these traces, further design implementation could be carried out to the power regulator in the future even after the manufacturing process. The finalized 5V and 3.3V regulation circuits are provided in Appendix J: Finalized 5V Regulator and Appendix K: Finalized 3.3V Regulator respectively.

14. 3 12V to -12V Conversion

The PT8NR112S takes in positive 12V input and outputs -12V. The datasheet provides the standard application for the regulator. It is shown in Figure 34.

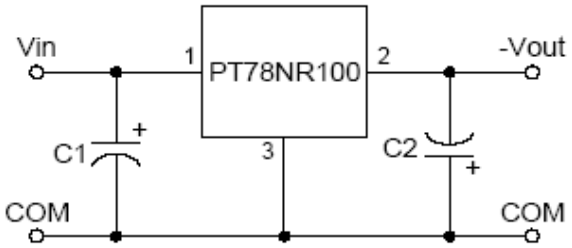


Figure 34: Standard Regulator Application

Using the example shown in Figure 34, a circuit for the -12V converter is designed. The design requires two electrolytic capacitors at both input and output for proper operation. The values of the electrolytic capacitors are 330 uF and 0.1 uF bypass capacitors are also used. The -12 V converter schematic used in the design is shown in Figure 35.

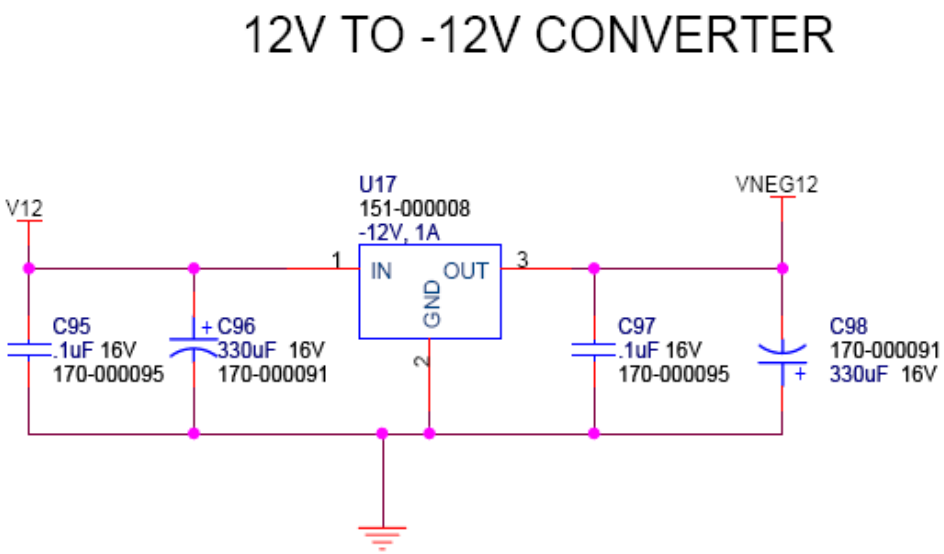


Figure 35: 12 to -12V Converter Schematic

15. Reuse Designs

The mouse, keyboard, and the Video designs were reused from the older debug board schematics because the signals and the parts that are used in the latest board are the same. Since the video, keyboard, and mouse signals derive from the host and doesn't need to be connected to the MCU on the debug card, they can be directly connected to their dedicated ports.

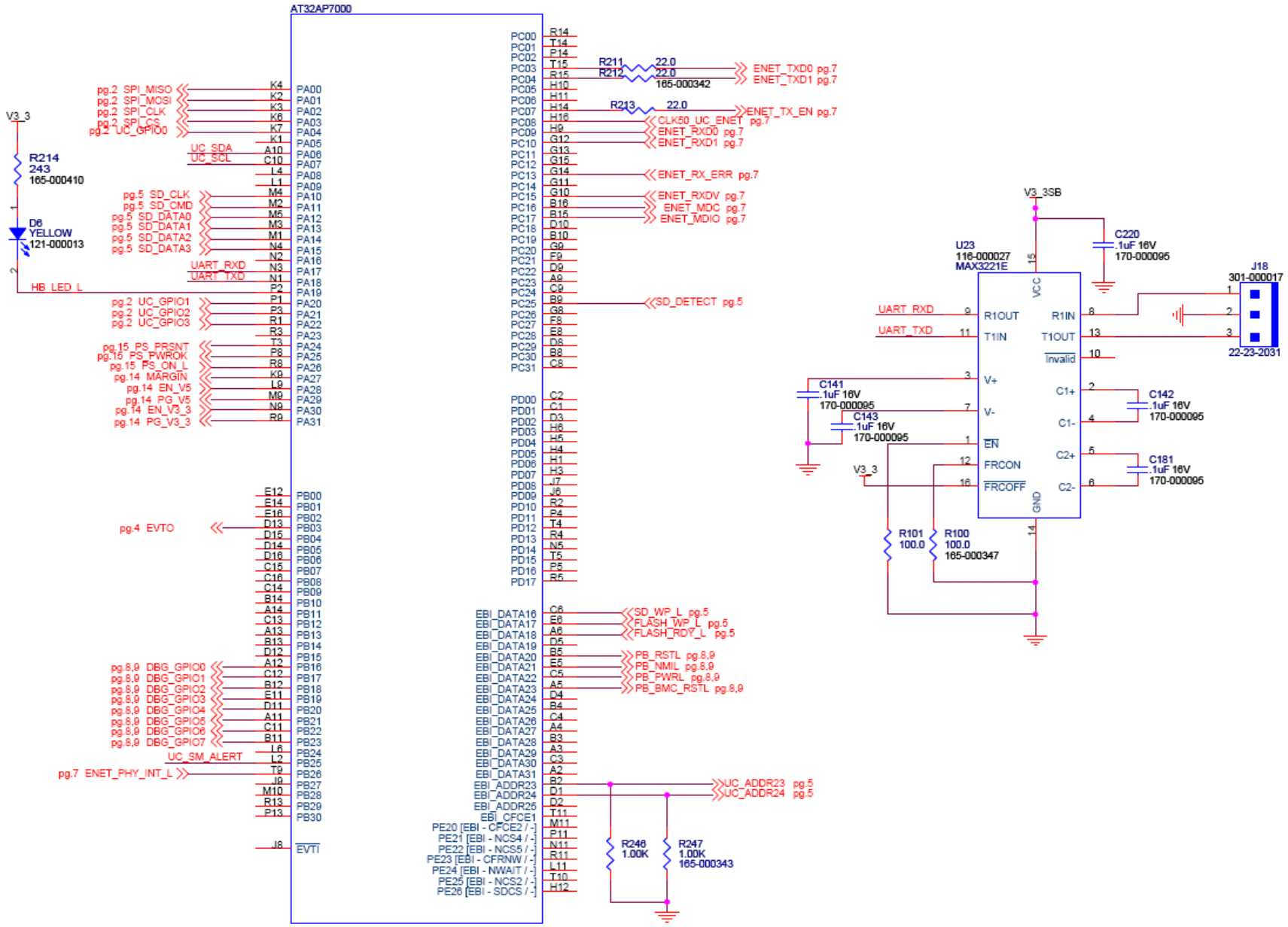
16. Conclusion

The debug board project covered a large number of component designs. The Atmel MCU was the core of the design and the CPLD also played an integral part in the design. As for the design requirements the following specification were met.

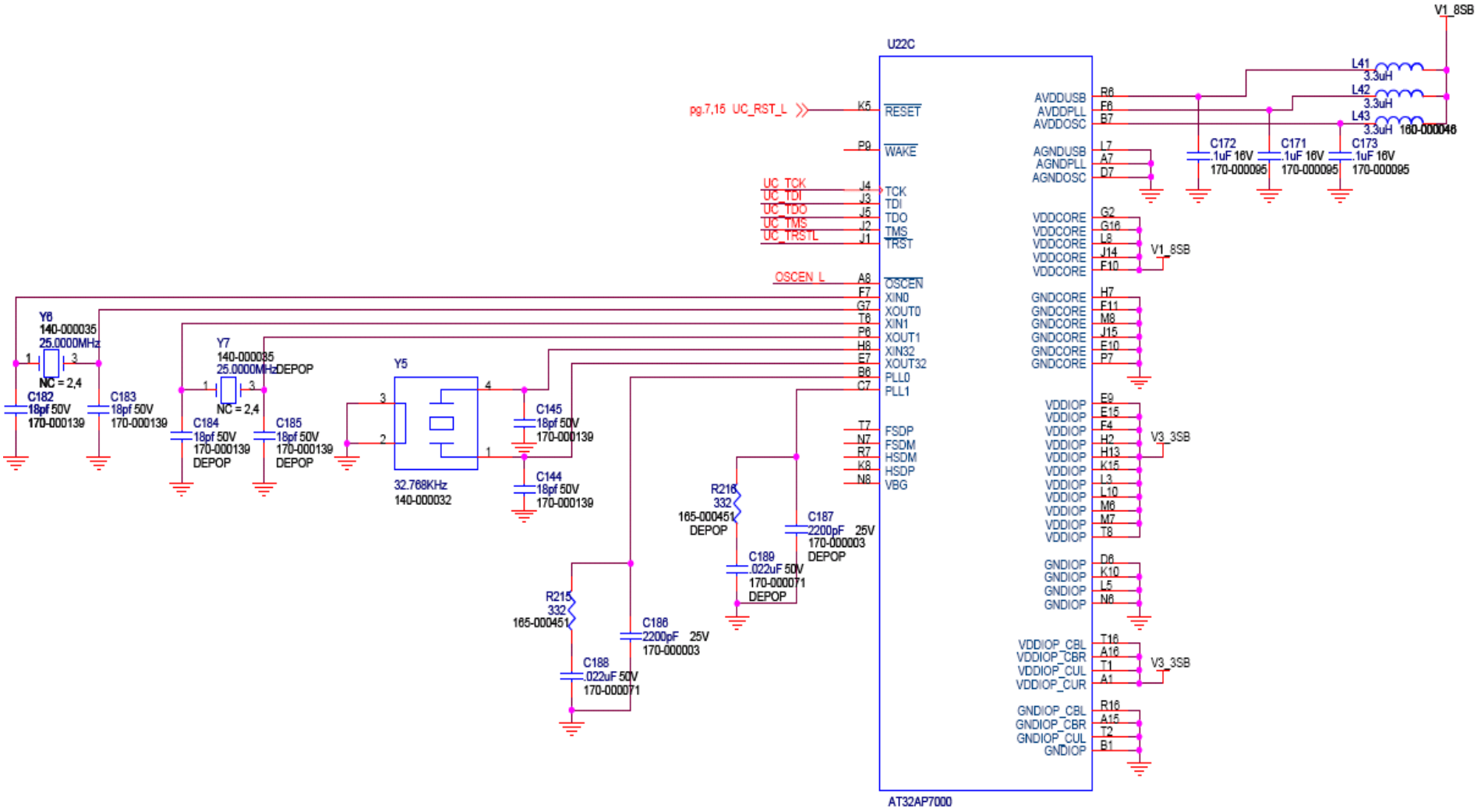
- Designed for video connectivity for Pilot II based video solutions, where the video signals will be driven from the Pilot II and is connected to the VGA connector
- Design supports SATA, PS/2 Mouse and Keyboard ports
- Supports four USB Type A ports
- Design supports a Low Pin Count (LPC) bus that intakes Port 80-83 data and display it on a seven segment LED
- Requires redirection and buffering to an external host through Ethernet for remote access capability.
- Low cost and high performance Rom Emulator that supports LPC and Firmware Hub (FWH)
- Supports eight settings of general purpose signals to the motherboard through DIP switches
- Four Push Buttons for Power Control, Reset, NMI, and BMC Reset

The additional requirement of controlling the GPIO through the MCU was also implemented. The majority of the CPLD code for the Altera CPLD was also coded and simulated. The simulation indicates that the code will sucessfully run on the Altera part.

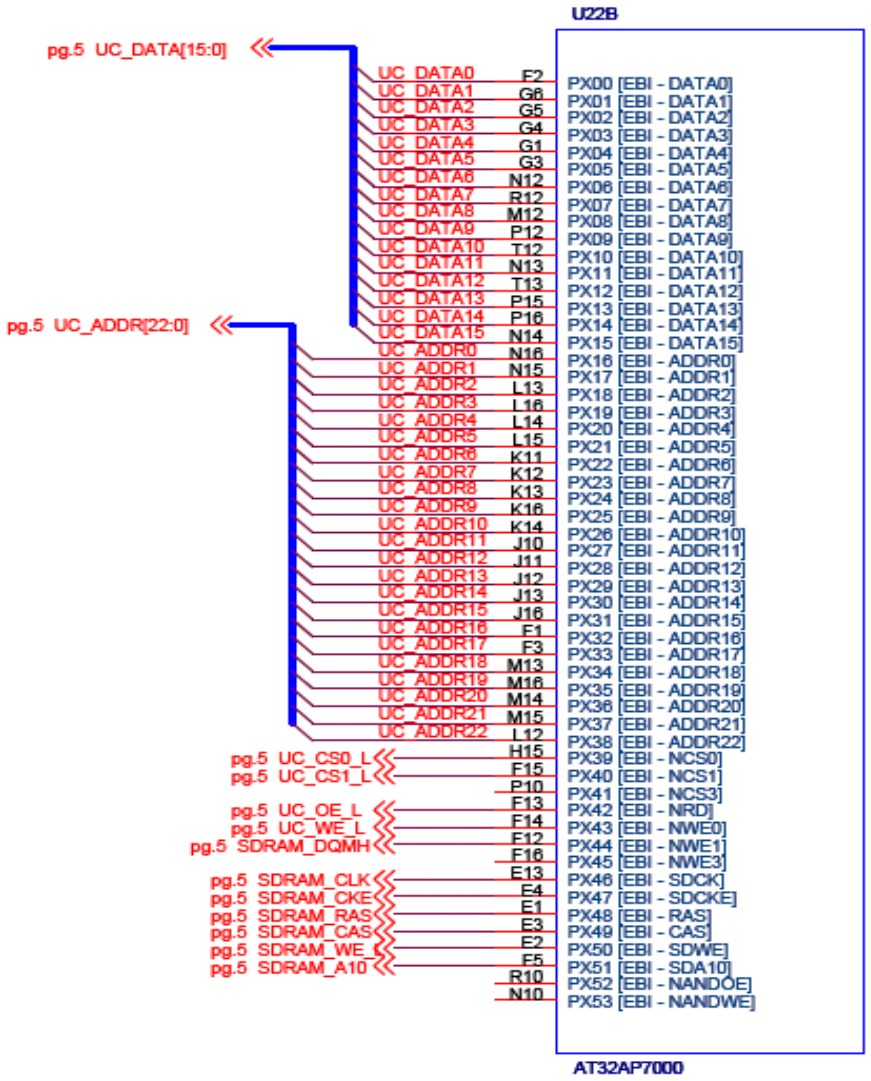
Appendix A: MCU Schematic Part A



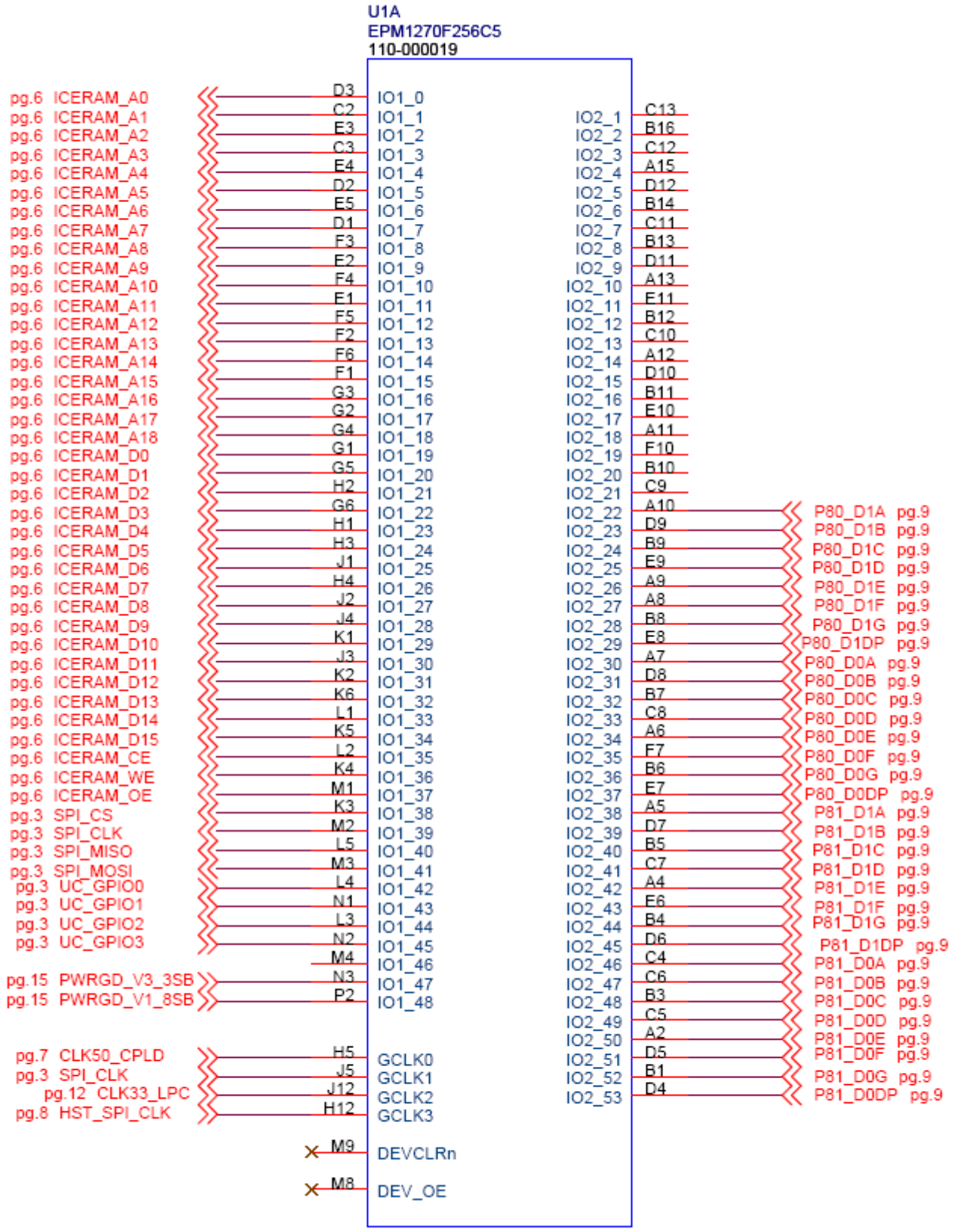
Appendix A: MCU Schematic Part B



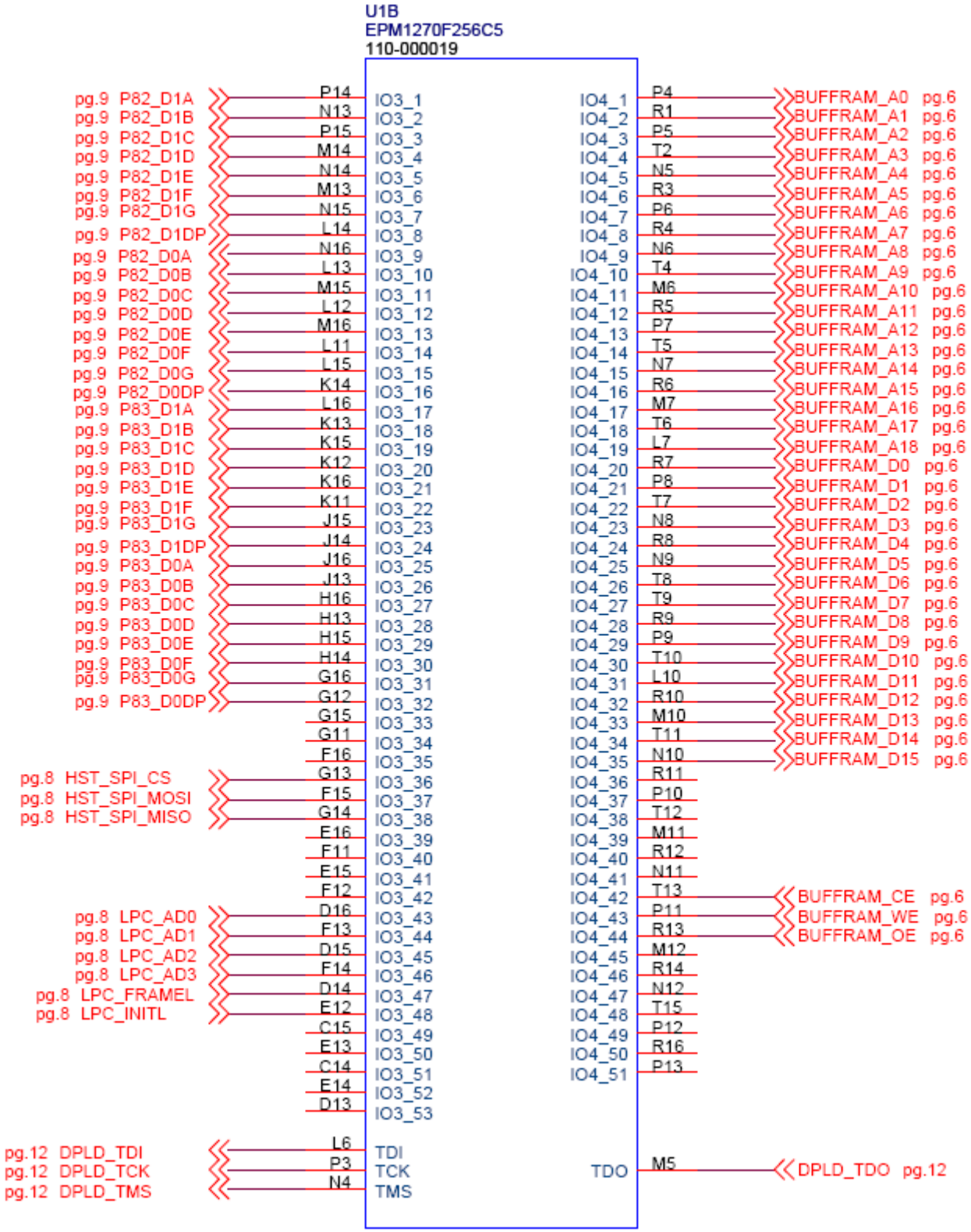
Appendix A: MCU Schematic Part C



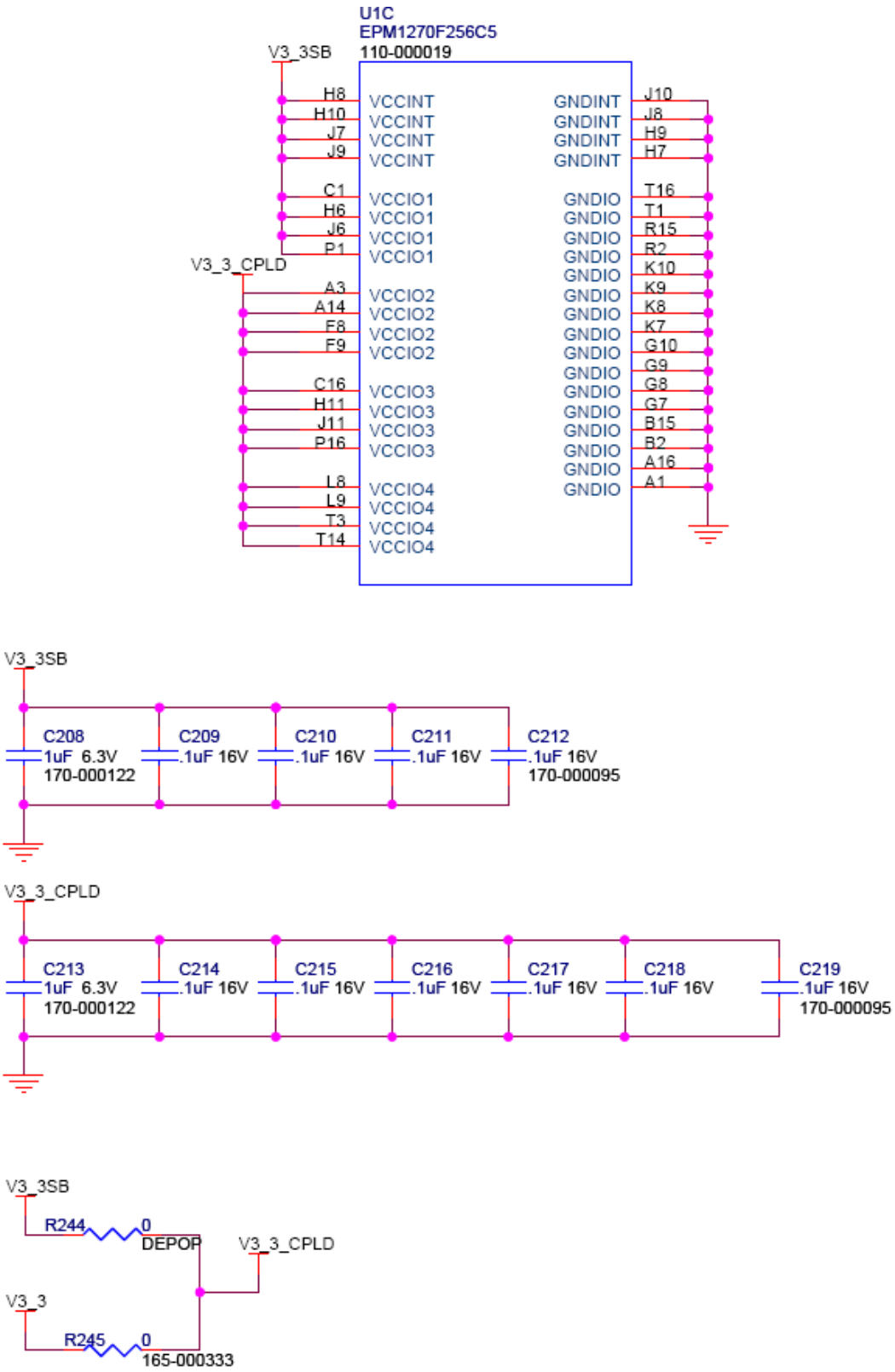
Appendix B: CPLD Schematic Part A



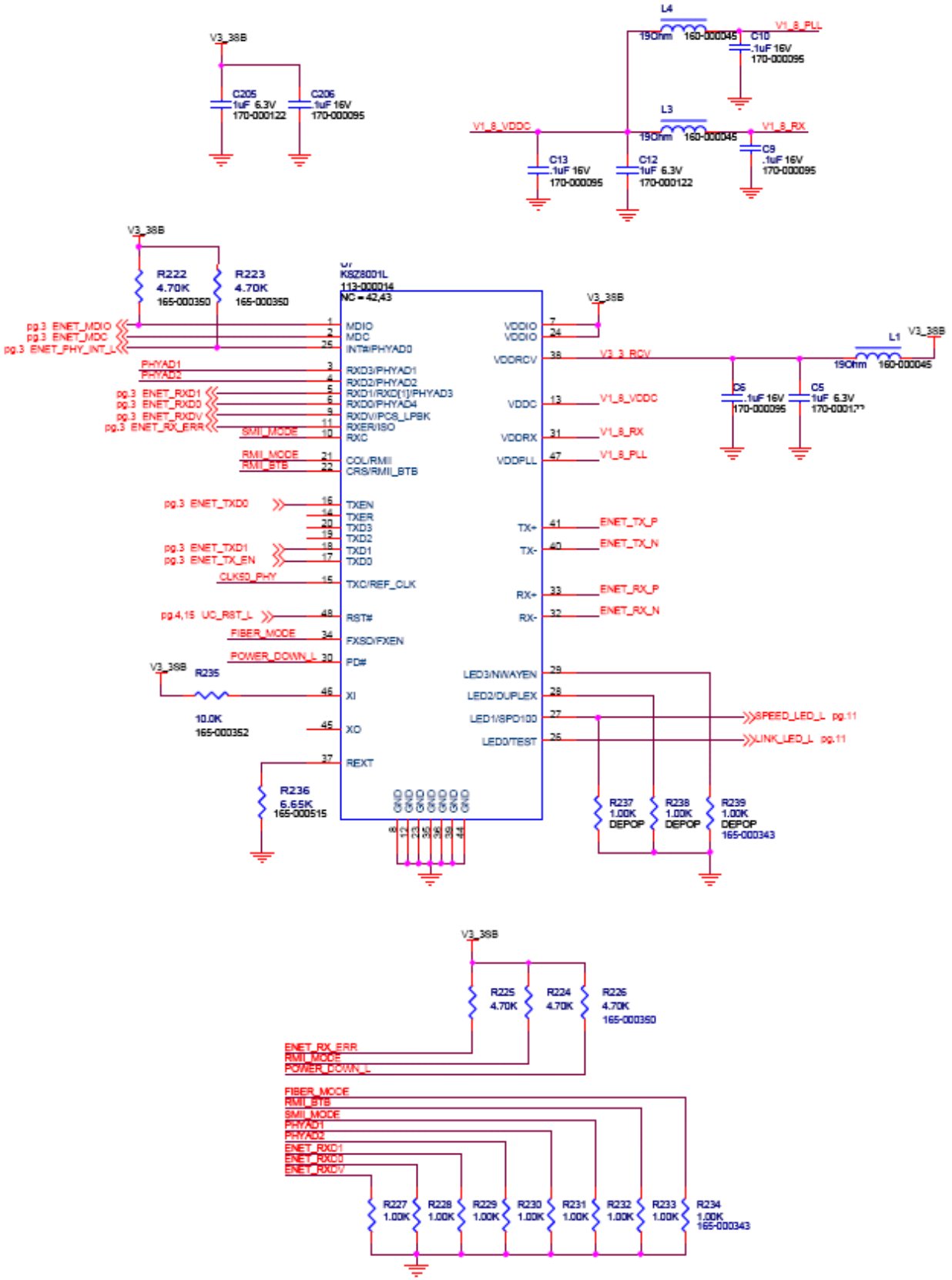
Appendix B: CPLD Schematic Part B



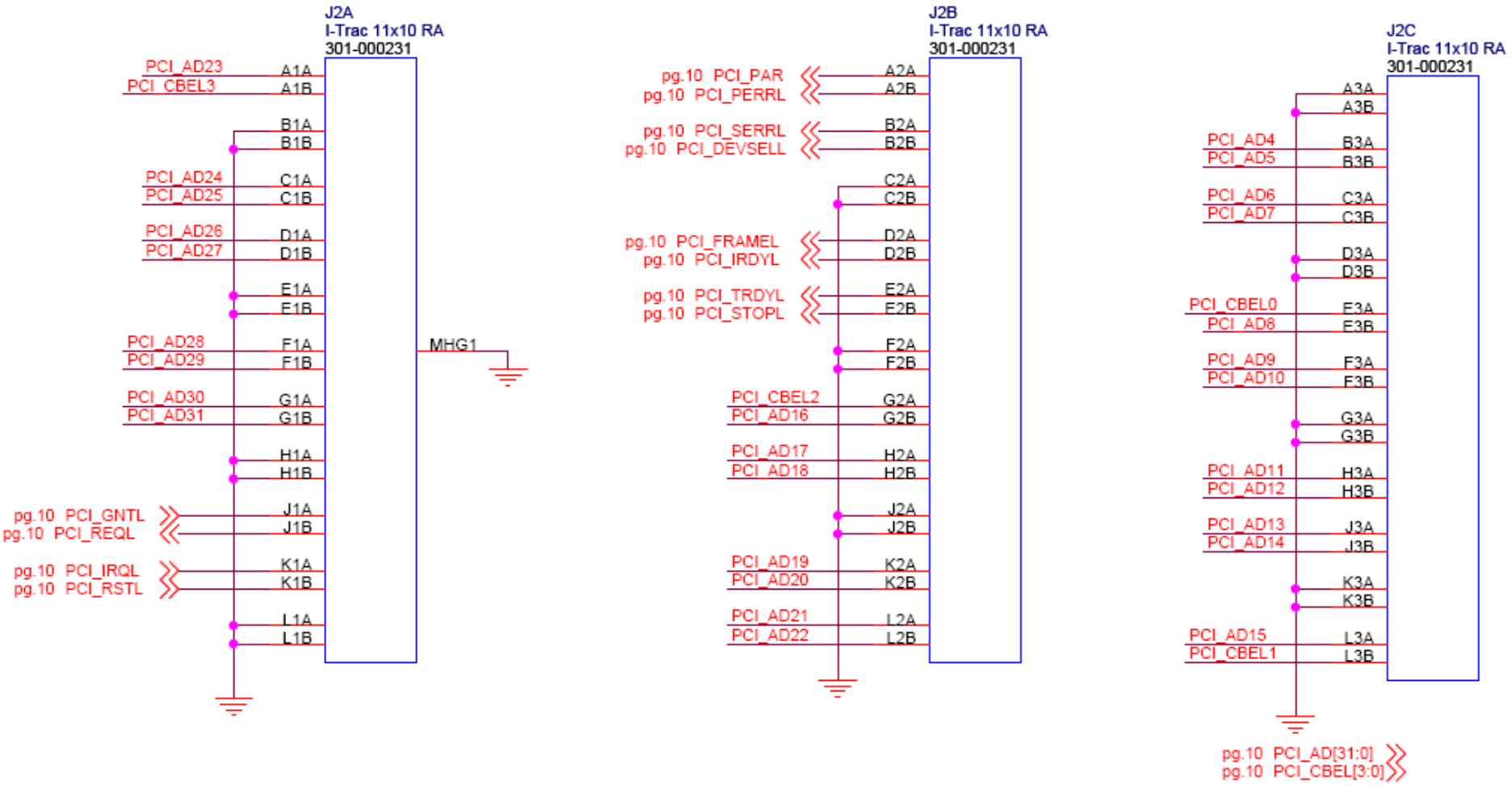
Appendix B: CPLD Schematic Part C



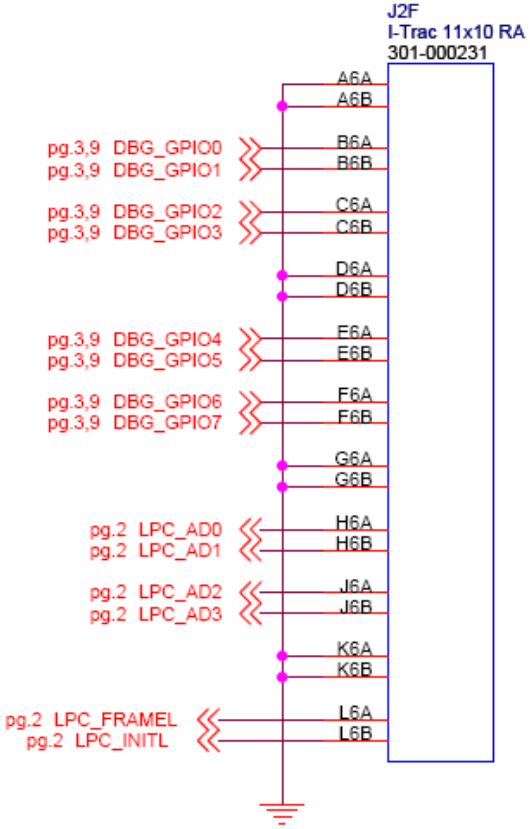
Appendix C: Ethernet PHY Schematic

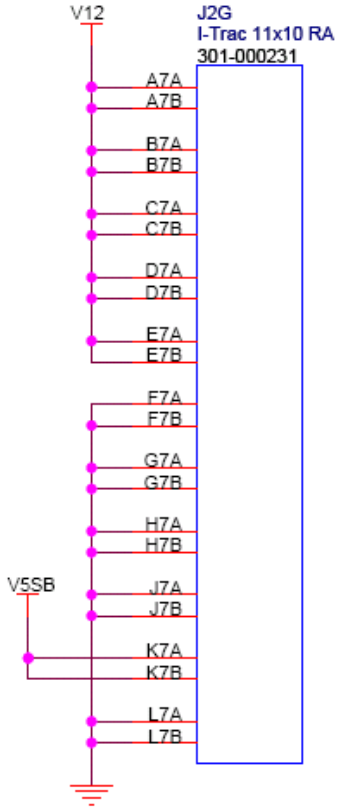


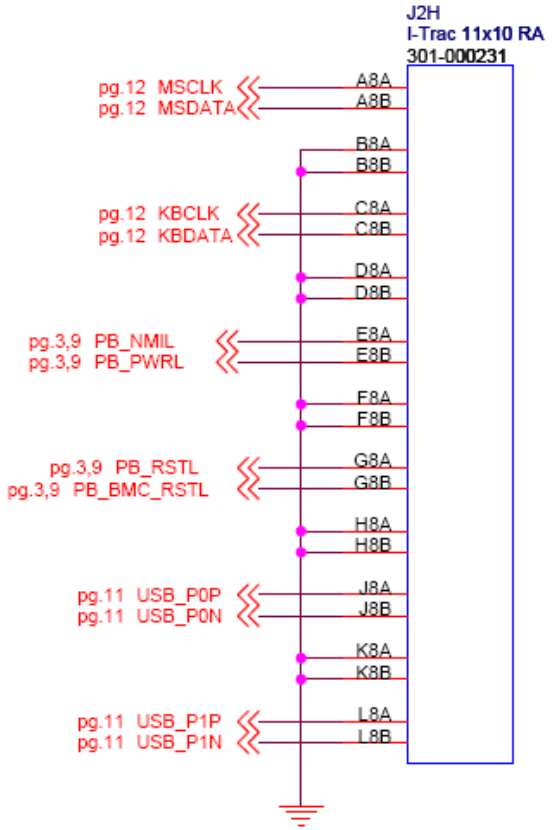
Appendix D: Debug Connector Schematic Part A



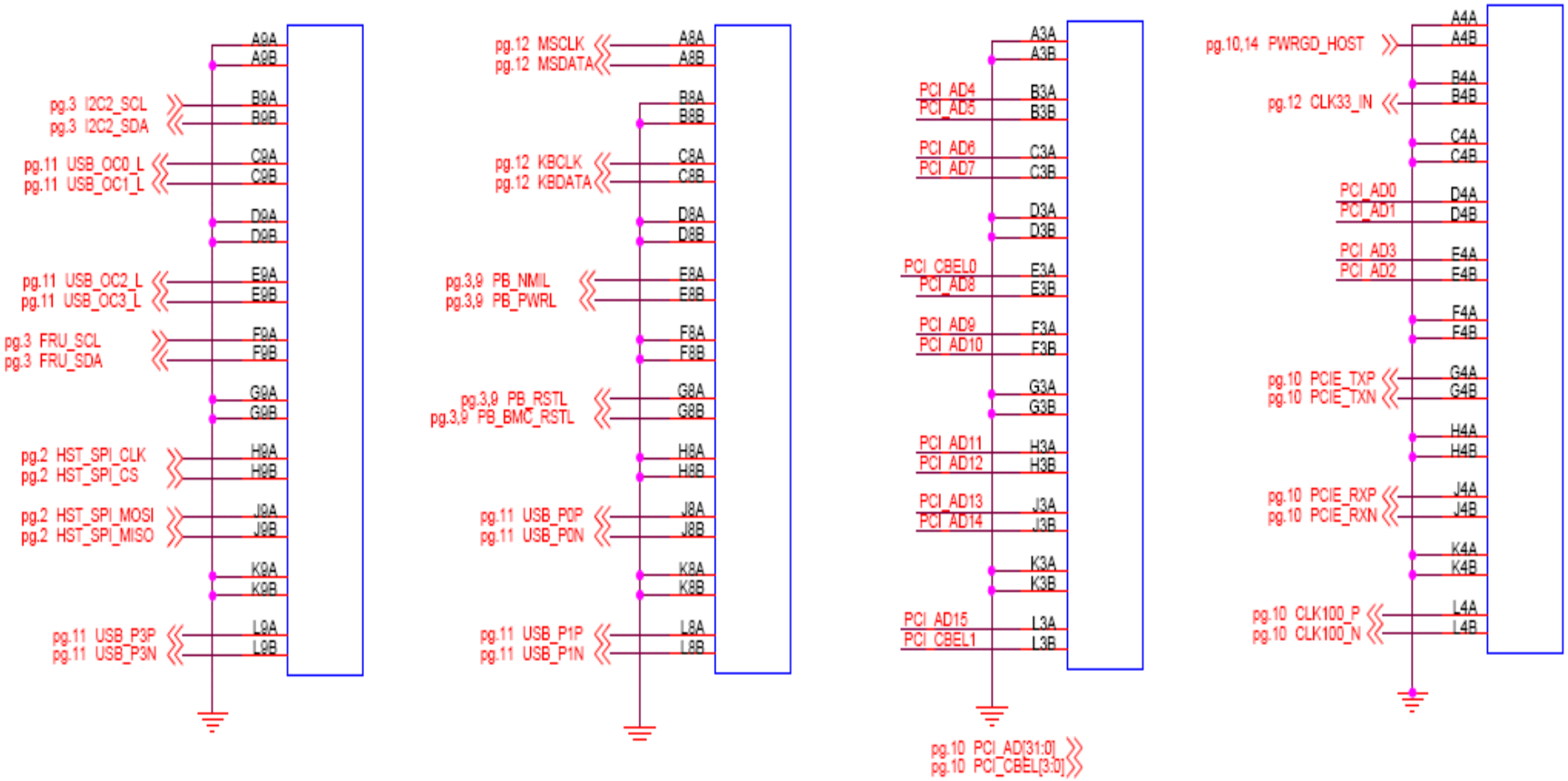
Appendix D: Debug Connector Schematic Part B



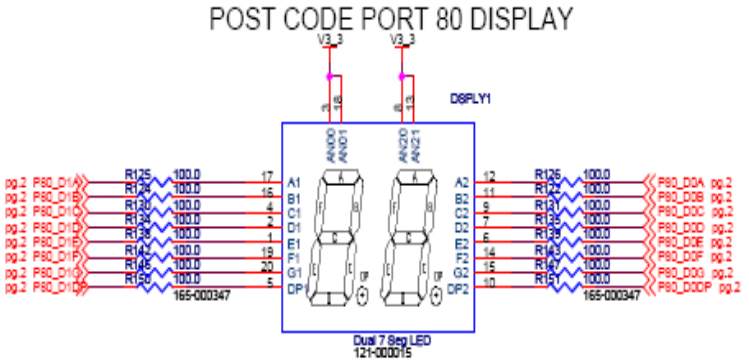


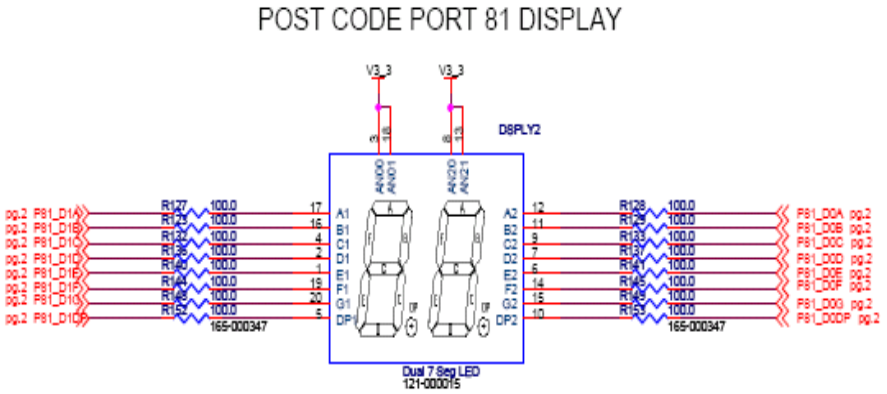


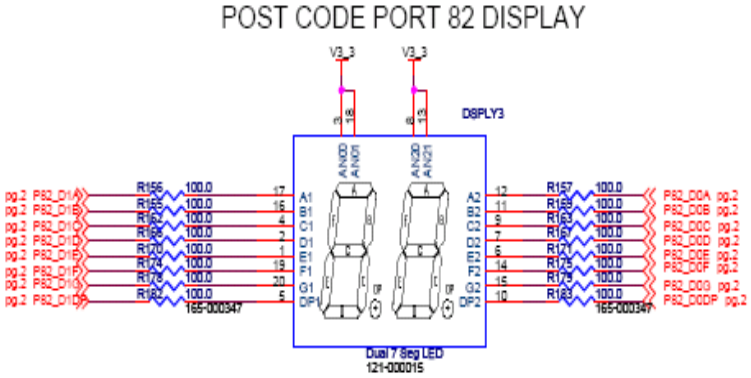
Appendix D: Debug Connector Schematic Part C

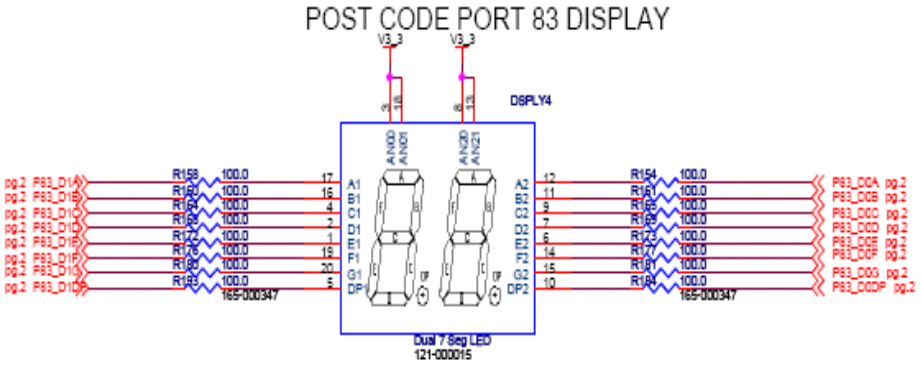


Appendix E: Port 80-83 Display Schematic





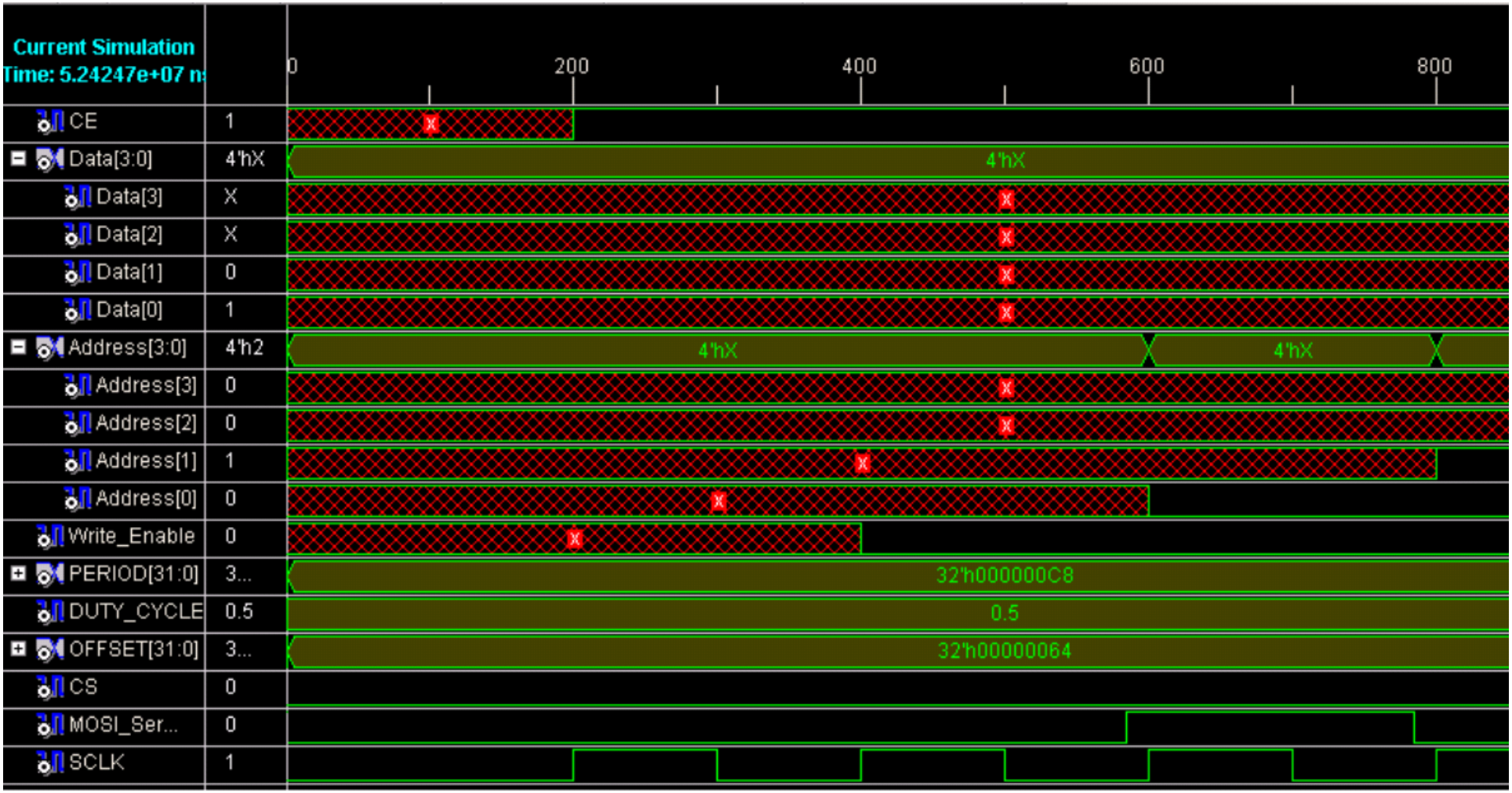




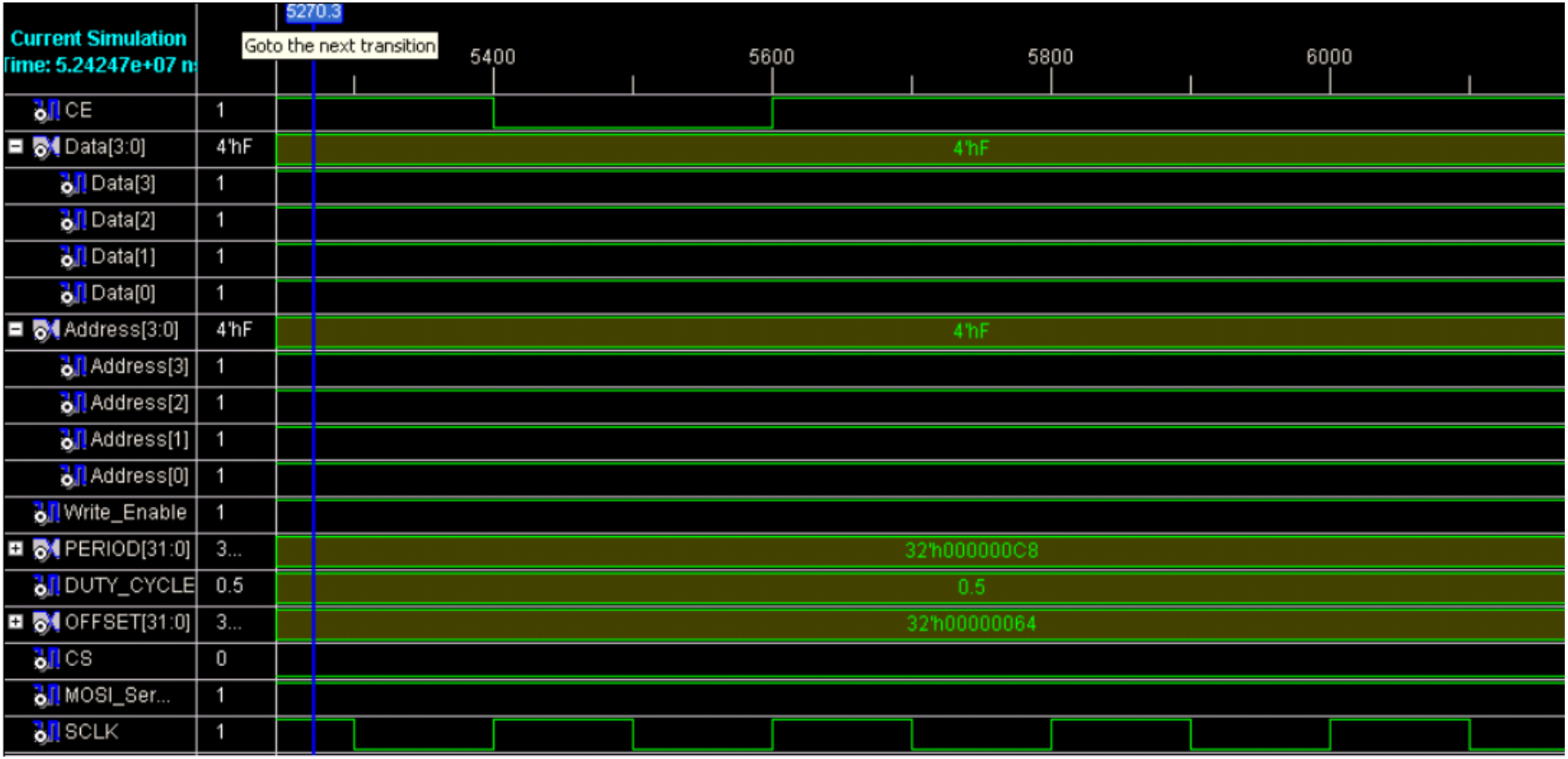
Appendix F: CPLD SPI Interface Code

```
entity nauset_CPLD_SPI is
Port (
        MOSI_Serial_Data: in STD_LOGIC;
        SCLK: in STD_LOGIC;
        CS: in STD_LOGIC;
        Address: out STD_LOGIC_VECTOR (3 downto 0);
        Data: out STD_LOGIC_VECTOR (3 downto 0);
        CE: out STD_LOGIC;
        Write_Enable: out STD_LOGIC );

end nauset_CPLD_SPI;

architecture Behavioral of nauset_CPLD_SPI is
        signal mem_35: STD_LOGIC_VECTOR (8 downto 0);
        signal chip_enable: STD_LOGIC;

        begin

        Process(SCLK)
                variable j: integer range 0 to 9 :=0;
                begin
                if rising_edge(SCLK)then

                        mem_35(j) <= MOSI_Serial_Data;
                        j:=j+1;
                        CE<='1';
                        Write_Enable <= mem_35(0);
                        Address(3 downto 0) <= mem_35(4 downto 1);
                        Data(3 downto 0) <= mem_35(8 downto 5);
                        if j=9 then

                                CE <='0';
                                j :=0;

                        end if;

                end if;
        end Process;
end Behavioral;
```

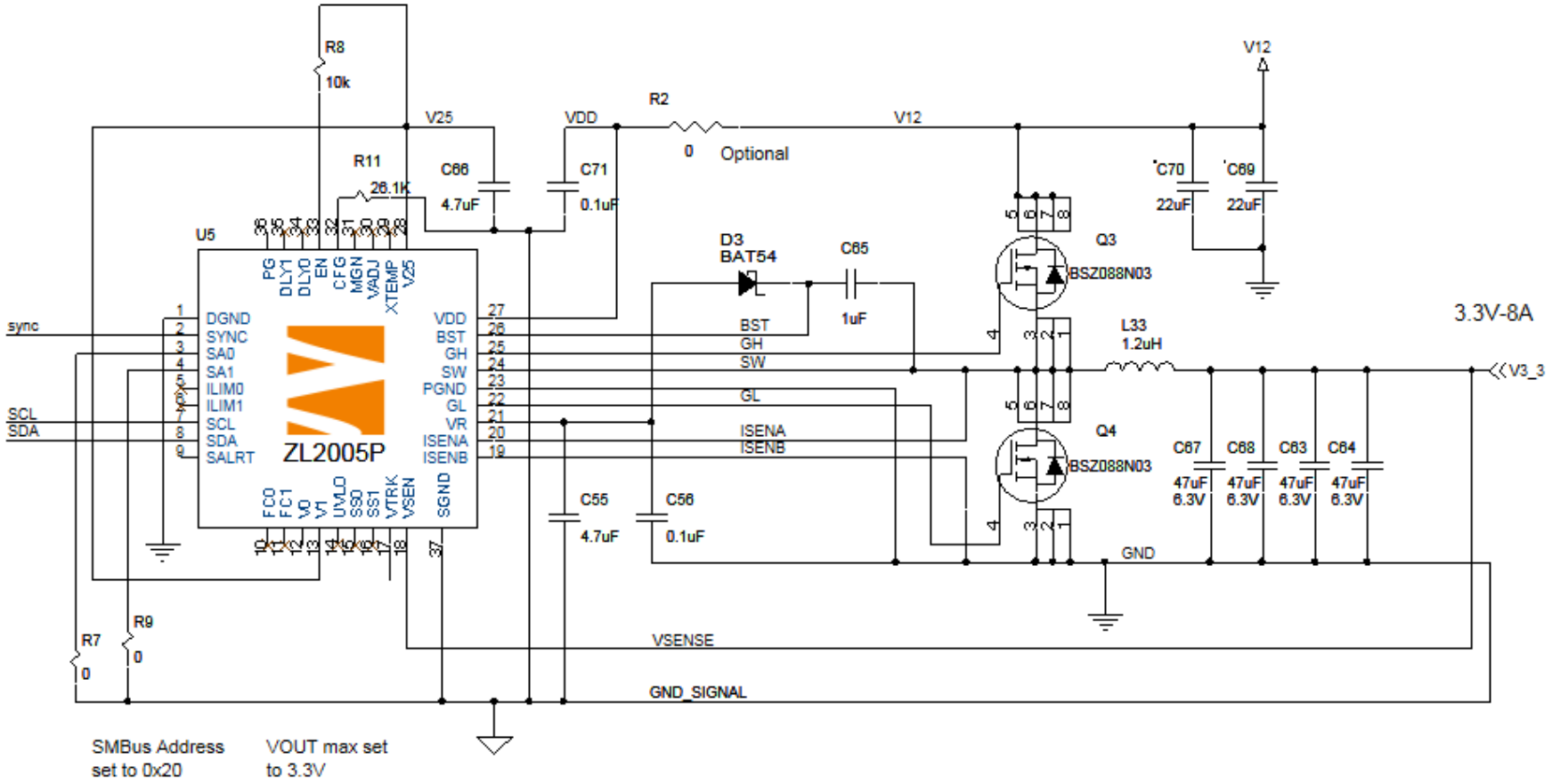
Appendix G: CPLD Simulation Part A



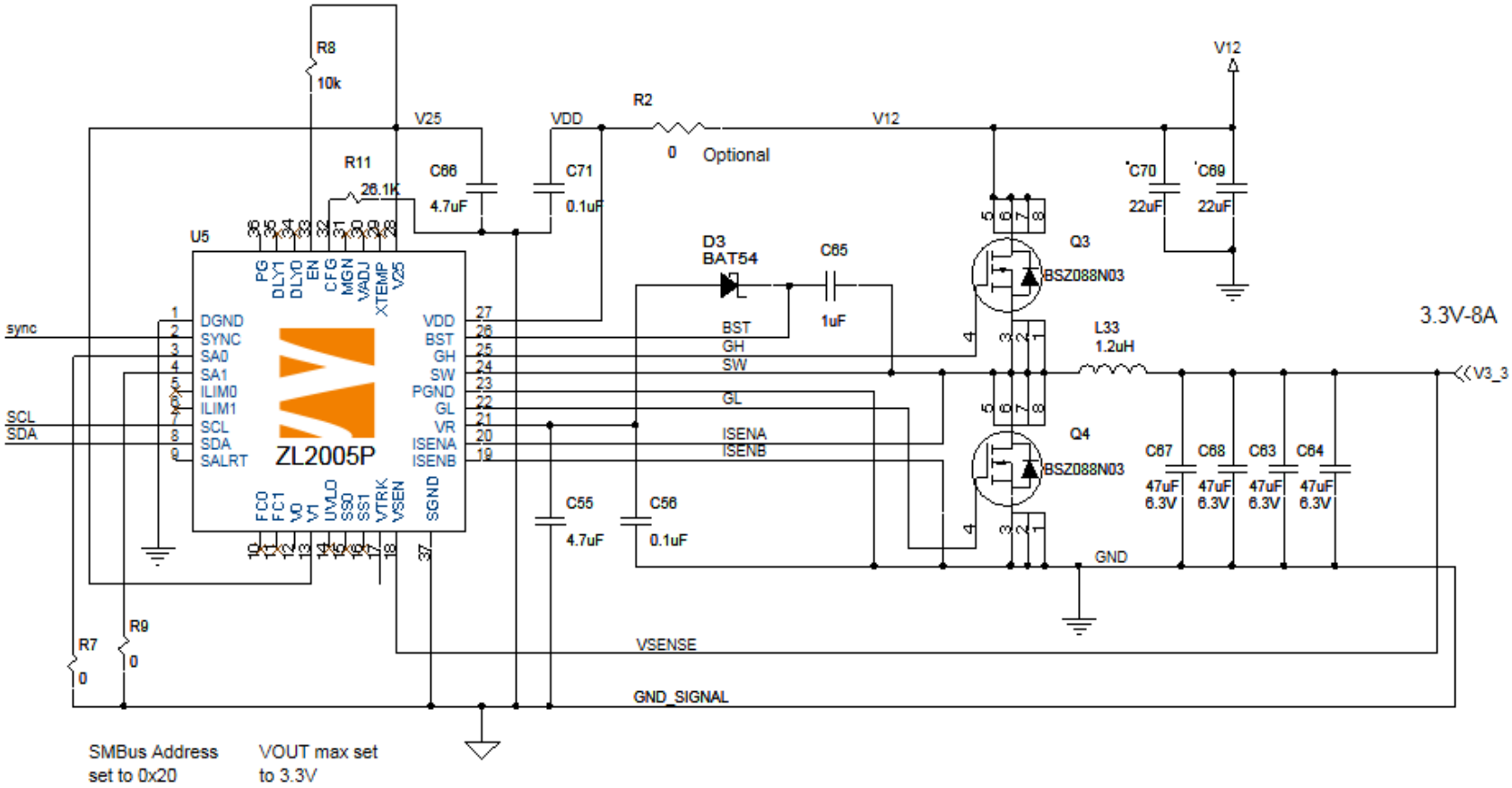
Appendix G: CPLD Simulation Part B



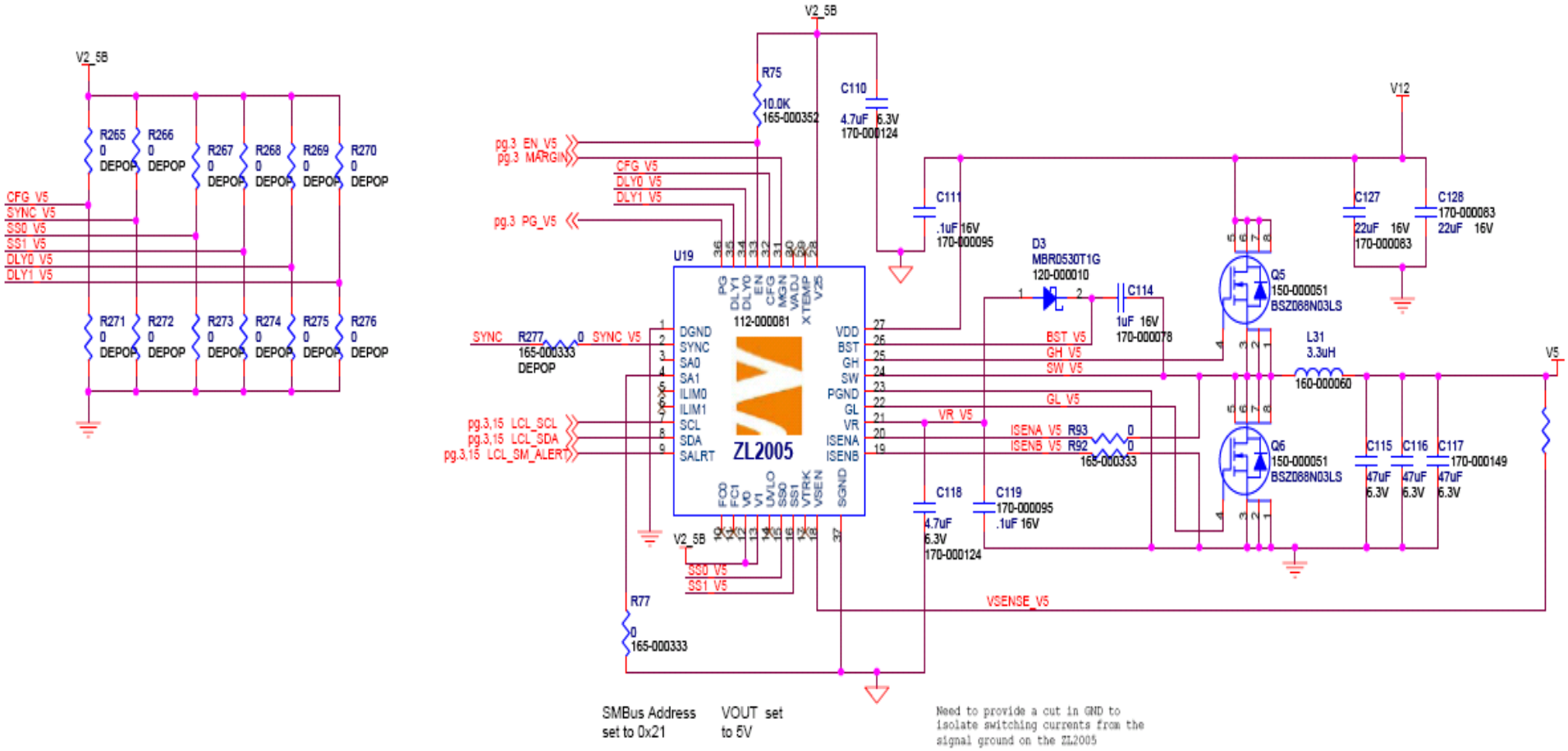
Appendix H: 5V Regulator Template



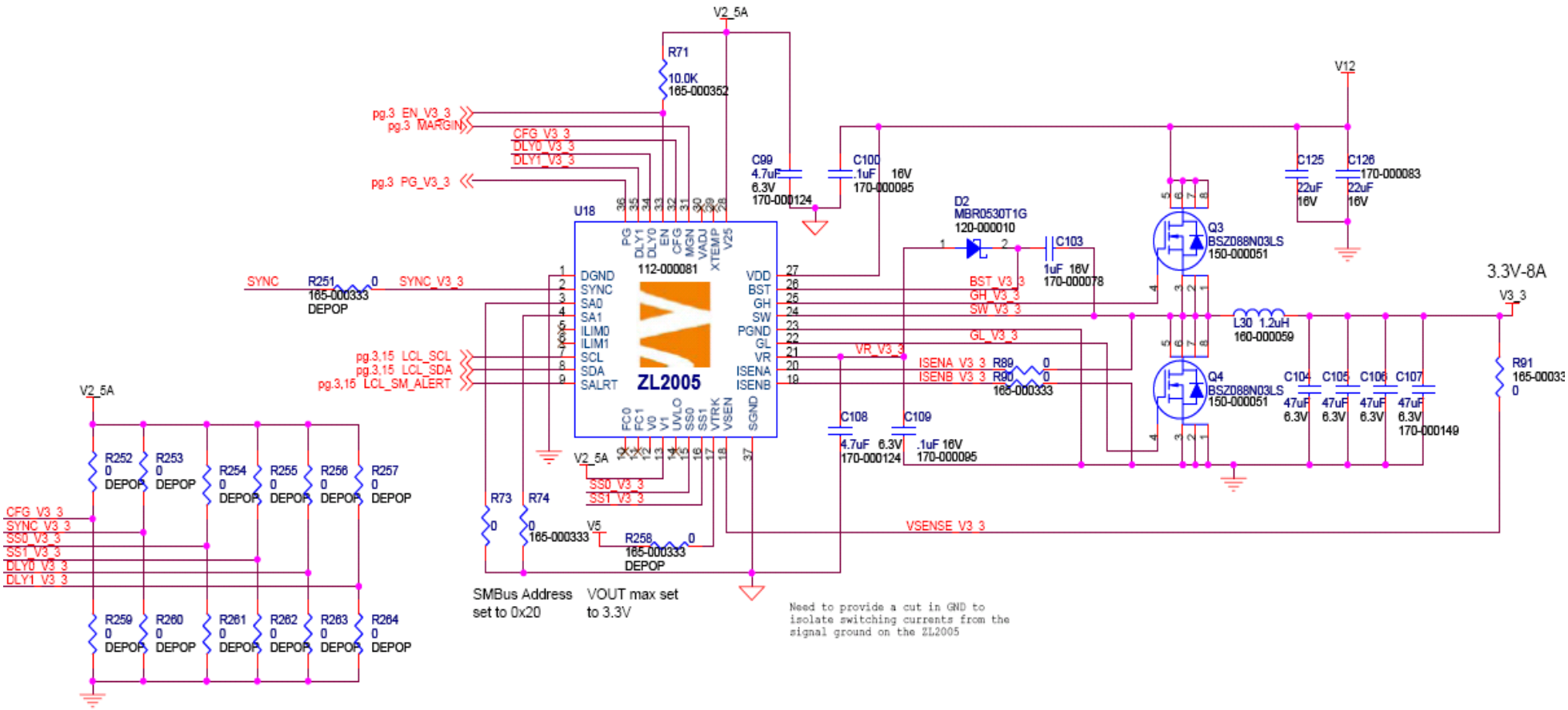
Appendix I: 3.3V Regulator Template



Appendix J: Finalized 5V Regulator



Appendix K: Finalized 3.3V Regulator



Reference

[1] "512K Words x 16 Bits x 2 Banks (16-MBIT) SYNCHRONOUS DYNAMIC RAM." . November 2006. ISSI. 24 Apr 2008 <www.issi.com>.

[2] "8-Mbit (512K x 16) Static RAM." . September 26 2007. Cypress Perform. 24 Apr 2008

[3] "AVR®32 32-bit MicroController AT32AP7000 Priliminary Sumary." ATMEL Inc. 24 Apr 2008 <www.atmel.com/literature>.

[4] Boutin, Matthew. Egenera Sandwich Board Schematics. January 2007

[5] "Crystal Oscillator Circuit Design." (1997) 02/11/2008 <http://mxcom.com>.

[6]Davis , Leroy. "PCI-Express 1x Connector Pin Out ." . Feburary 13 2008. 24 Apr 2008 <http://www.interfacebus.com/Design_PCI_Express_1x_PinOut.html>.

[7] Davis , Leroy. "Peripheral Component Interface Pinout." . March 26 2006. 24 Apr 2008 <http://www.interfacebus.com/Design_PCI_Pins.html>.

[8] Khirman, Stas. "JTAG FAQ." 02/21/2004 02/11/2008 < http://hri.sourceforge.net/tools/jtag_faq.html>.

[9] "KSZ8001L/S." 1.8V, 3.3V 10/100BASE-T/TX/FX Physical Layer Transceiver 03 2006 02/11/2008 <http:/www.micrel.com>

[10] "MAX 3000A Programmable Logic Device Family." . June 2006. Altera Corp. 24 Apr 2008 <www.altera.com/literature>.

[11] "MAX3221/MAX3223/MAX3243*." (2002) 02/11/2008 http://www.maxim-ic.com/

[12] "MirrorBit® Flash Family." . Feburary 12 2008. Spansion. 24 Apr 2008

[13] "RS232 serial cables pinout." (1997) 02/11/2008 <http://www.lammertbies.nl/comm/cable/RS-232.html#pins>.

[14] "SPI to I2C using MAXII CPLD." . December 2007. Altera Corp. 24 Apr 2008 <www.altera.com/literature>.

[15] "Two-Wire Serial EEPROM." 2005 02/11/2008 http://www.atmel.com/literature

[16] "ZL2005 Data Sheet." Digital-DC™ Integrated Power Management and Conversion IC. 2006. Zilker Labs, Inc. 24 Apr 2008