ECE 368 Digital Design

Spring 2014

Project: Lab2 – UMD RISC 24

Dates Performed: Wednesday, April 2th, 2014

Submission Date: Wednesday, April 22nd , 2014

Team #2 :

Massarrah Tannous _____

Daniel Noyes _____

Table of Contents

Problem Statement

This lab is considered part 2 of the project. The main purpose of this is to extend and build upon part 1 of this project. For part 1 of this project, we had to create the control entity and the FPU entity. The control entity is considered the center of the machine that controls the flow of the instruction in the pipeline. Whereas the main purpose of the FPU is to process the instruction where several registers must be used, such that general purpose registers. Having the control unit separate from the FPU unit creates a Harvard Architectures. This type of architecture has its tradeoffs where it is speed efficient due to parallelism but not area efficient.

For the current part of the lab, we had to complete the 5 stage pipeline in the control unit and that is: Fetch, Decode, Execute, Memory and Write back. The 5 stage pipeline, allows fetching instructions from the instruction memory with every clock cycle. While an instruction is being fetched, other instructions are being processed through the pipeline. We also had to integrate into our machine the Instruction Memory, Data Memory, and External memory. The instruction memory is a centralized memory where all the instructions are stored and based on the Program Counter the instructions are fetched. The data memory and the external memory performs similarly which allow load and store into memory. In order to complete this lab, we had to create the hardware debugging interface which allows the assembly instructions are inputted through the keyboard and the results of the FPU is outputted on the VGA. The basic design for the UMD RISC24 is given in the lab handout as the following figure (*figure 1*).

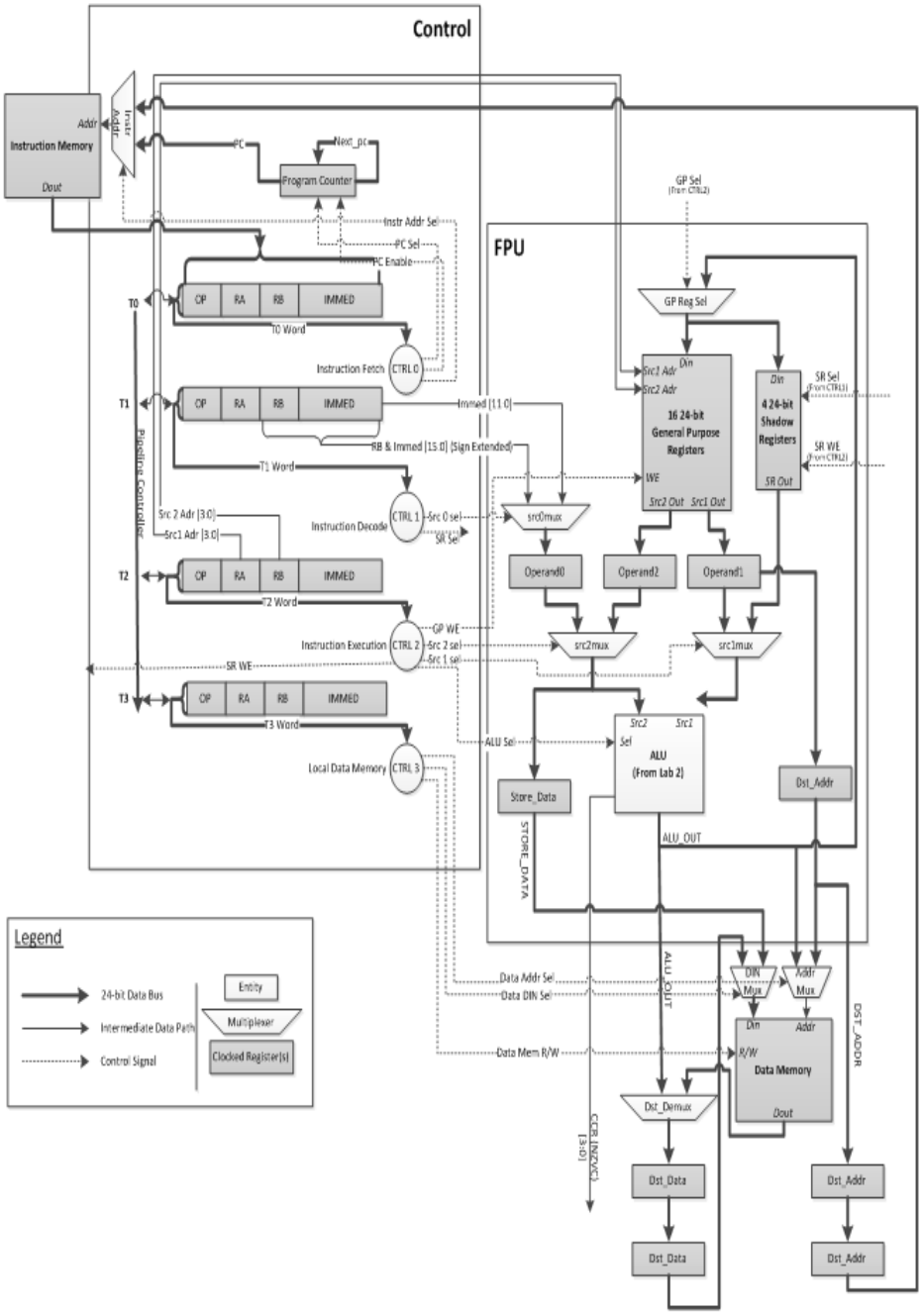

Figure 1: Pipelined FPU Data and Control Paths

Simple machine RTL block level design

(Original Designs)

The original design is very similar to the given block diagram in figure 1. Figure 2 shows the original block level design for the project. It also shows the timing diagrams at which edge (falling or rising edge) the data is latched. The attached papers, figures 11-13, are the original hand drawn design for the control unit (figure 11), FPU that includes the data and external memory (figure 20 from the previous report), data/external memory (figure 12), program counter (figure 13), and the interface of the debugger with the RISC UMD.

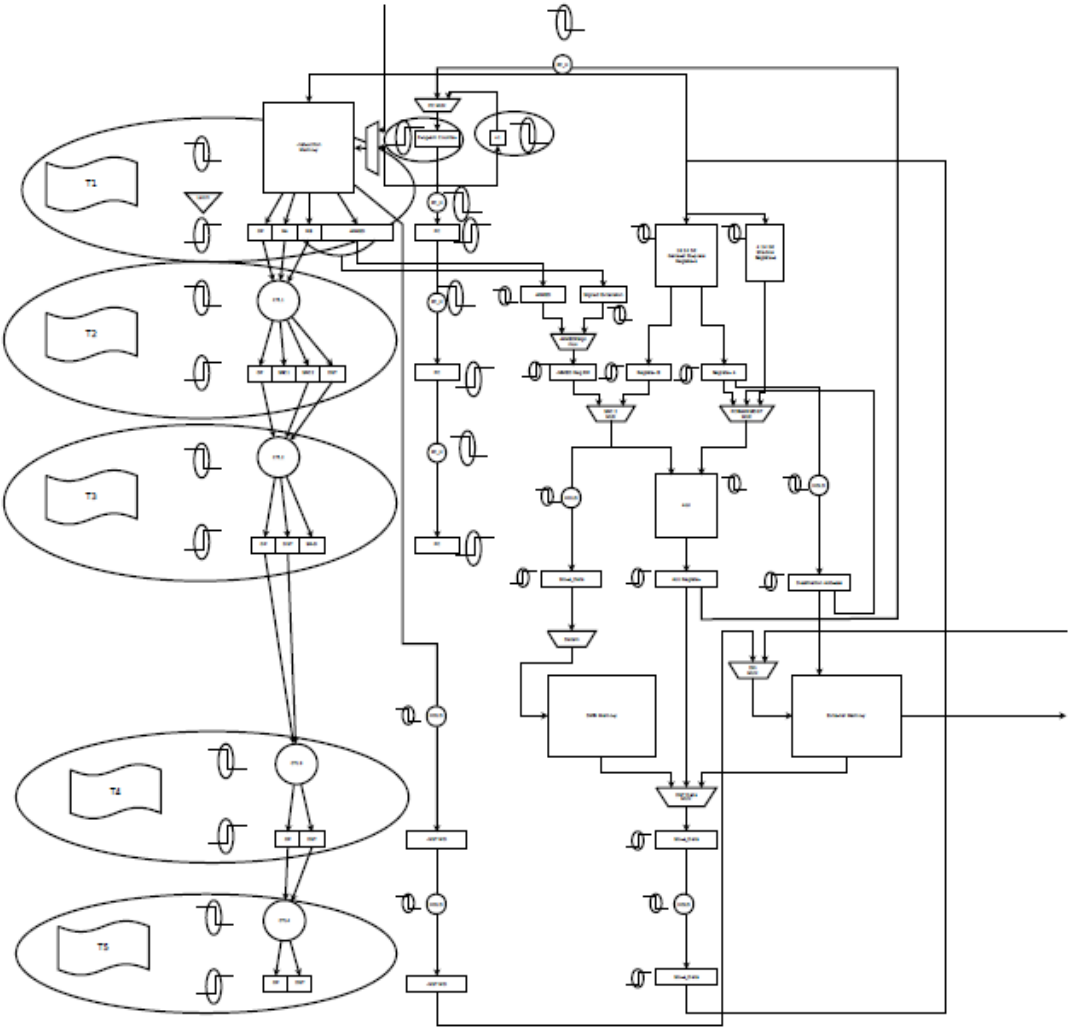

Figure 2: Original Design

VHDL component specification and schematic designs

For the specifications of each new component for the RISC Machine in: Control Unit, Counter, Data/External Memory are shown in the Appendix A (Figures A1.1 through A.17). For the FPU unit please refer for the appendix A of the previous report (Project- Lab1). For the components specifications for both the VGA and the PS2 controller, please refer to Appendix A of lab3.

VHDL system specification and resulting schematic design

Based on the overall designs, figure 18 of Project- lab1, the overall system consists of several entities: PS2 controller, Debug Unit, VGA controller, and UMD-RISC24. The UMD-RISC24 Consists of several entities: Control Unit, FPU Unit, PC, Instruction Memory, Data/External Memory.

The specifications and the schematics for the PS2 Controller and the VGA are similar to Lab 3 (figure 2 and 3 of lab 3). The specification and the schematic of the FPU unit are the same as Parject-Lab1 report Since the control unit has been updated with the 5 stage pipeline, the following shows the specifications and schematic (figure 3 & 4). We also show the specifications and the schematic of the UMD RISC24 and its integration with the debug unit .

Control Unit Entity

Specification

The control entity is considered the center of the machine that controls the flow of the instruction in the pipeline. It passes signals from the debug unit into its components. The top level of the control unit is shown in figure 3 and its corresponding RTL diagram is shown in figure 4. Tables 1 and 2 show the description of the inputs and the outputs corresponding to figure 3.

Inputs	Description
CLOCK	Drives the component.
RESETN	Resets the chip constants.
ENABLE	Enable/Disables the components.
INSTRUCTION	Enables/Disables writing to the instruction memory.
PC	Address of where to write into the instruction memory

Table 1: Description of the control unit inputs.

Outputs	Description
SR_SEL_CTRL1	Selects between the shadow register.
SRC0_SEL_CTRL1	Selects between Immediate (0) and Extended (1)
SRC1_SEL_CTRL1	Passes the RA data.
SRC2_SEL_CTRL1	Passes the RB data.
SRC1_SEL_CTRL2	Selects between register B (1) and Immediate (0)
SRC2_SEL_CTRL2	Selects between register A (0) and shadow register (1)
SIGN	Passes the signed data from the instruction
IMMED	Passes the immediate data from the instruction
ALU_SEL	Passes the opcode to the from the instruction at the CTRL 2.
PC_STACK	Passes the popped value from the stack
PC_OUT_CTRL3	Passes the program counter at the CTRL3
DATA_DIN_SEL	Selects between DST Reg(0) and STORE_DATA (1)
DATA_ADDR_SEL	Selects between DST Reg (0) and ALU (1)
DATA_RW	Selects based based on the opcode: READ(0) and Write(1)
DST_MUX_SEL	Selects between ALU(00), Data Memory (01), External Memory (10)
GP_DIN_SEL	Reg Address to write to
GP_WE	Write enable for general purpose register
SR_DIN_SEL	Shadow Register to write to
SR_WE	Write enable for shadow register

Table 2: Description of the outputs for the control unit.

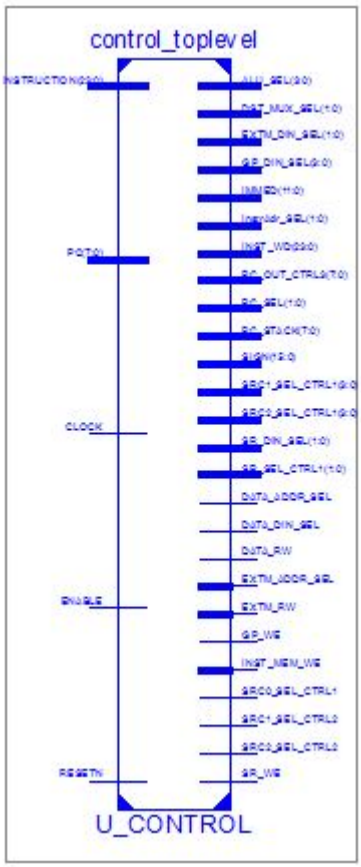

Figure 3: Top level of the control unit

Schematic

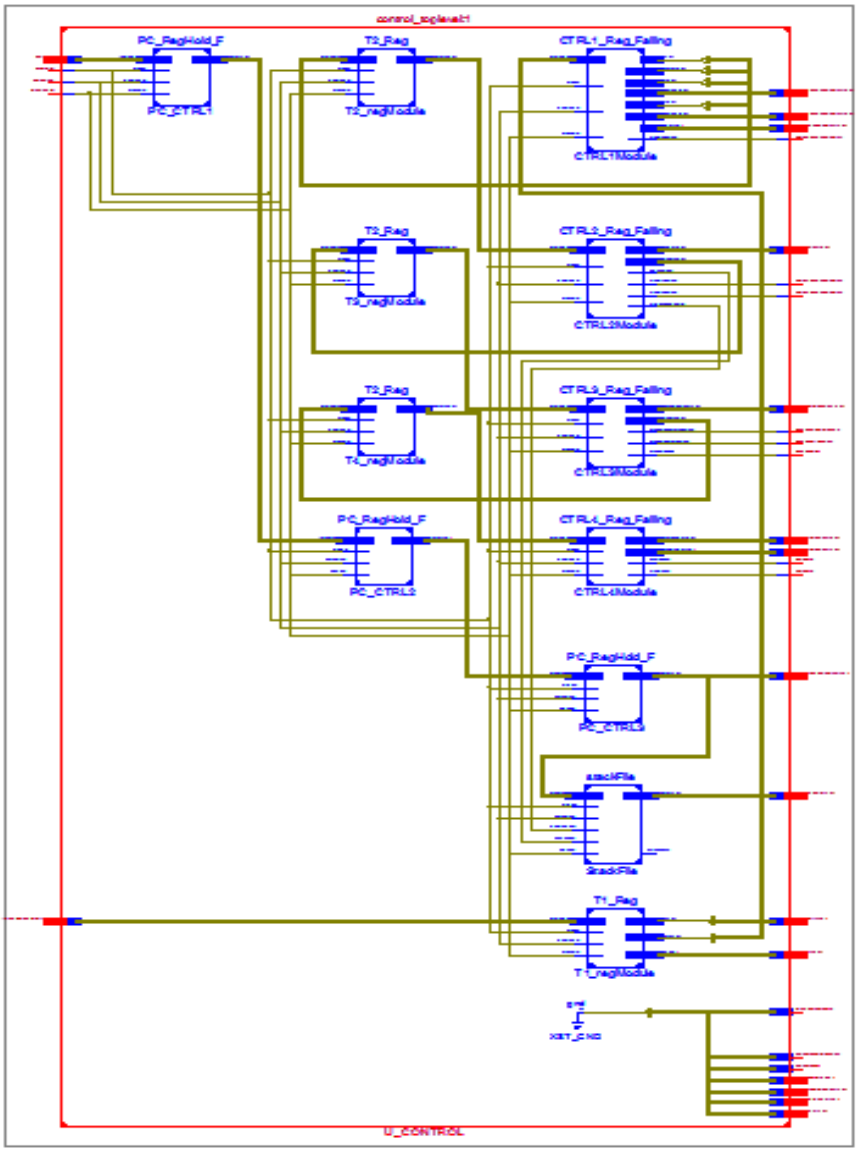

Figure 4: RTL diagram of the control unit v2.

UMD RISC 24

Specification

This entity consists of the control entity and the FPU. It interfaces with the debug unit to get the instructions process them and then pass back the results to the debug unit. The top level of the UMD RISC24 is shown in figure 5. Its corresponding RTL diagrams are shown in figure 6 and 7, where figure 6 shows the top level of the UMD RISC components and figure 7 shows all the components. Tables 5 and 6 show the description of the inputs and the outputs corresponding to figure 5.

Inputs	Description
CLOCK	Drives the component.
RESETN	Resets the chip constants.
ENABLE	Enable/Disables the components.
WRITE_ENABLE	Enables/Disables writing to the instruction memory.
ADDR_IN	Address of where to write into the instruction memory
D_IN	Overwrites the instruction memory with new instructions

Table 3: Description of the inputs of the UMD RISC24.

Outputs	Description
STORE_DATA	Destination Data
DST_ADDR	Destination Address
ALU_OUT	Results of the arithmetic operation
CCR	Condition Code Register (ALU)

Table 4: Description of the outputs of the UMD RISC24.

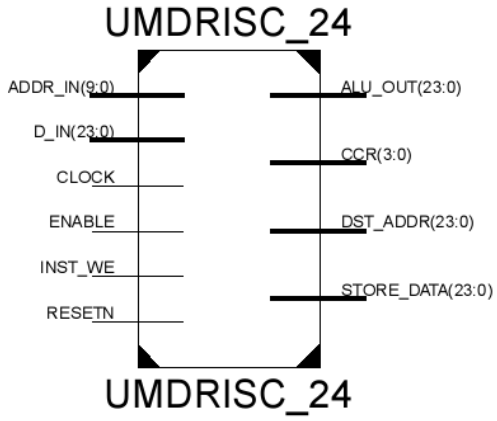

Figure 5: Top level for the UMD RISC24

Schematic

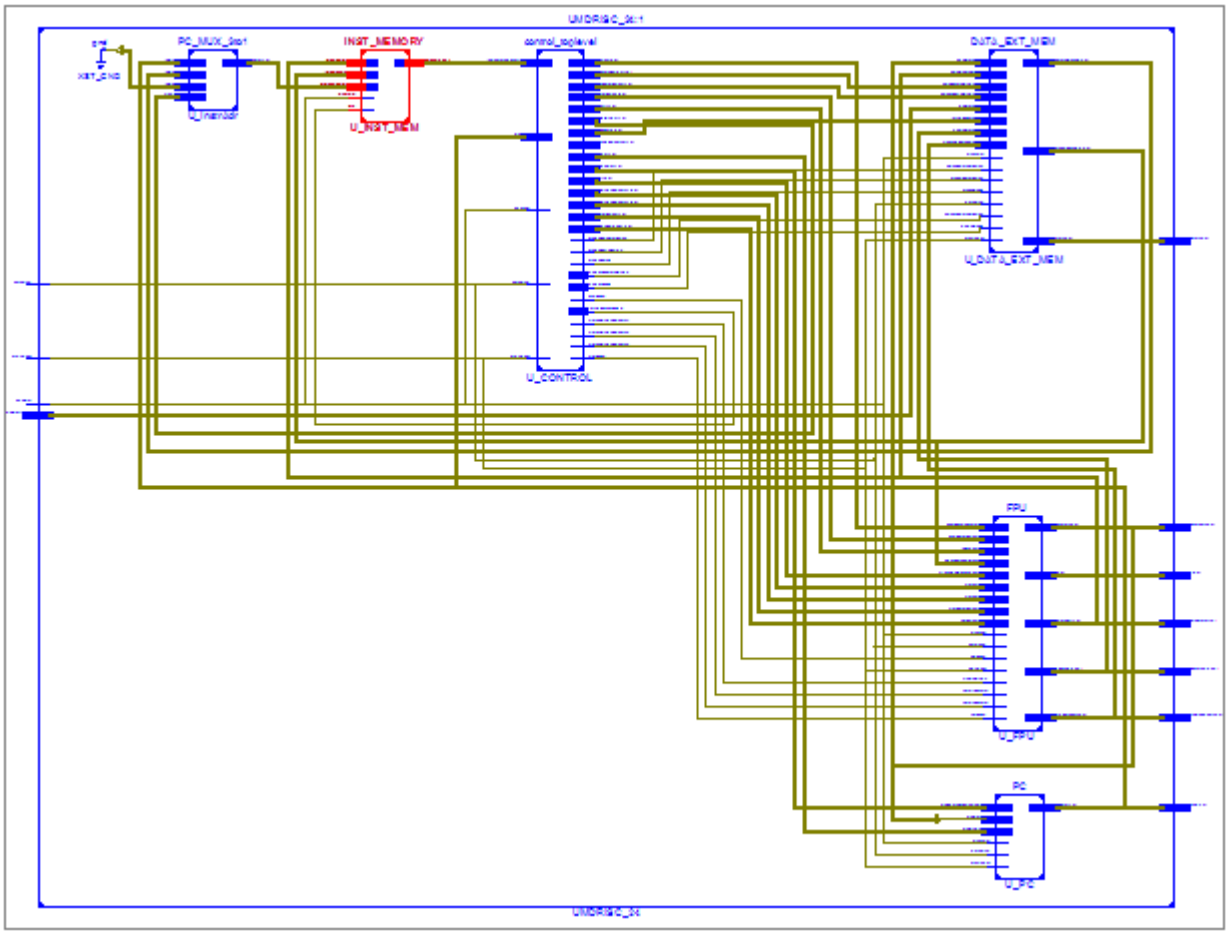

Figure 6: RTL Diagram of the the UMD RISC24 showing the toplevel of it is components.

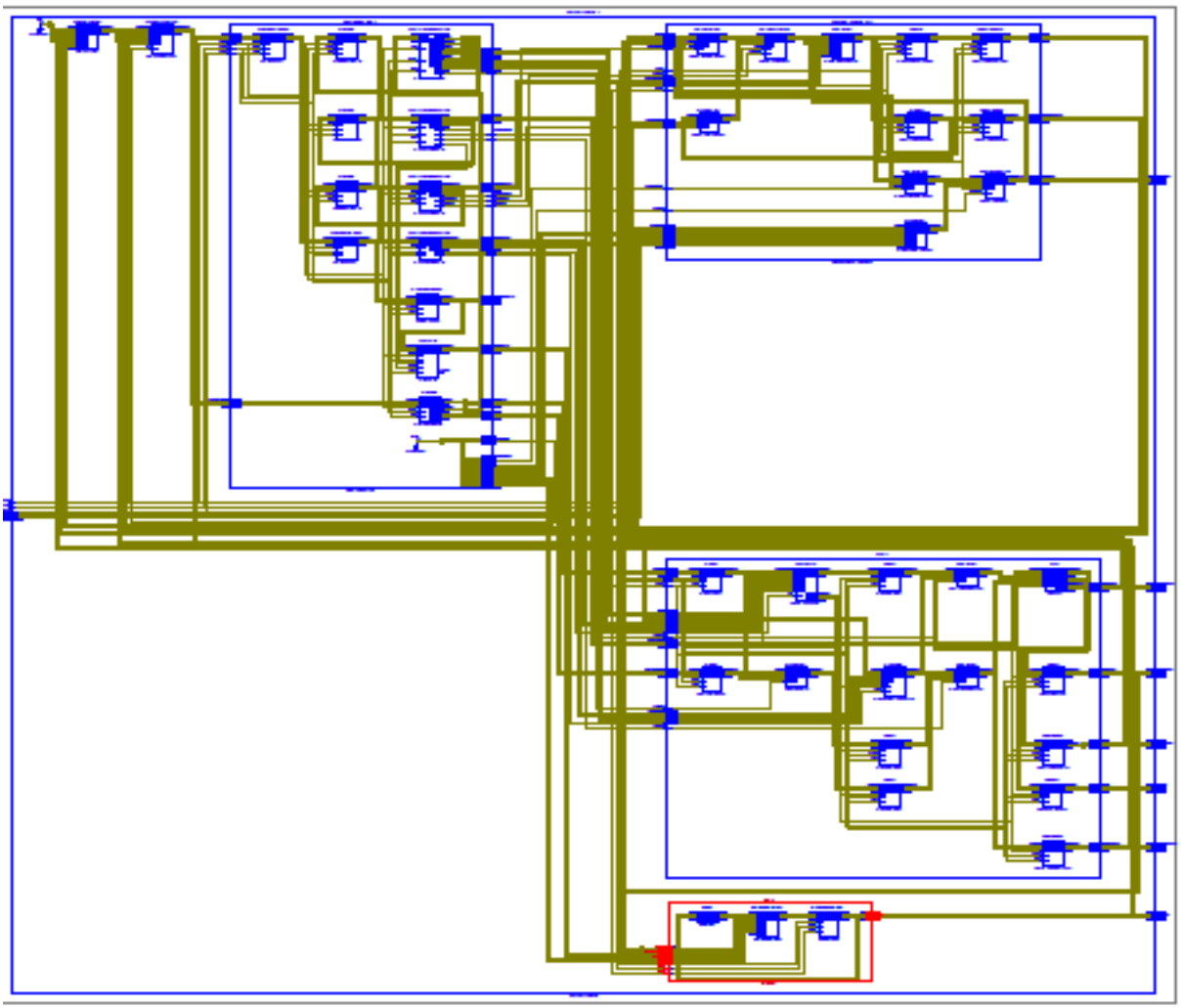

Figure 7: RTL Diagram of the the UMD RISC24 showing all the the components.

Debug Unit

Specification

This entity consists of the debugger structural, VGA structural and the UMD-RISC24 structural. It interfaces with on the top most level to the exterior pins on the FPGA. The goal of this entity is to successfully debug the RISC machine and allows to demonstrate that it works on a hardware level. The top level of the Debug unit is shown in Figure 8. The corresponding RTL diagram is shown in Figure 9. Tables 5 and 6 show the description of the inputs and the outputs corresponding to Figure 8.

Inputs	Description
CLK	Drives the component.
CLR	Resets the chip constants.
PS_CLK	Keyboard clock input
PS_DATA	Keyboard data input
Readsignal	Signal to indicate to read the signal
RISC_WE	Write enable to the risc

Table 5: Description of the inputs of the Video and Debugger top level.

Outputs	Description
PC_OUT	Program counter output
B1	Blue VGA input
G1	Green VGA input
R1	Red VGA input
HS	Horizontal sync
VS	Vertical sync

Table 6: Description of the outputs of the Video and Debugger top level.

Schematic

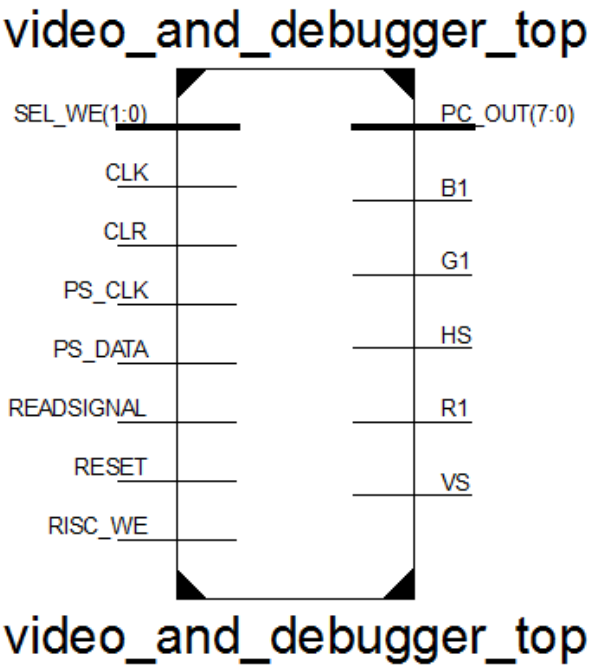

Figure 8: Top level for the video and debugger top level

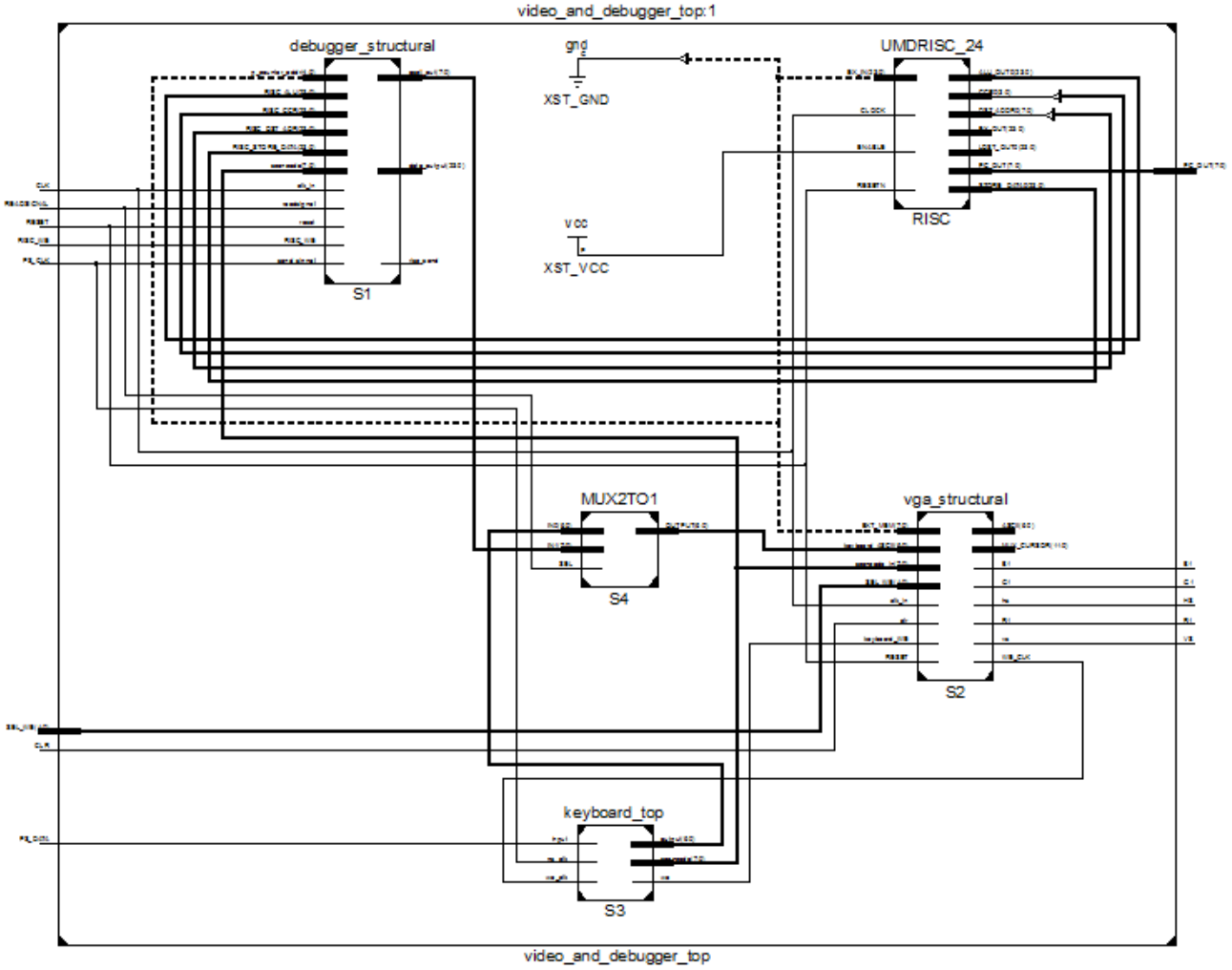

Figure 9: RTL Diagram of the the Video and Debugger showing the top-level of it is components.

VHDL Code and Test Bench Code

The VHDL code for the UMD Risc24 integrated with the debug unit has been emailed as a zip file to Dr. Fortier and to David Prairie.

Designs comparisons

In the original design schematics, most of the design was based upon Dr. Fortier's UMD RISC24 schematics in Lab#1 and Lab#2 handouts. While going over the schematics, there were minor errors picked up that indicated a bug or a flaw that can happen. One of the designs that we created was the Data Memory and External memory management block. This segment of the code was designed to fullfil the memory write back procedure in the overall RISC architecture. Most of the design still remained in place from the original handout. Slight modifications of the select and mux for both memory managements were adjusted to meet the specifications. For the Program counter, It was originally designed to be in the control unit but was later changed to the RISC structure level. This change was made due to the dependencies that the program counter was needed in the Lab.

For the top-level of the RISC, it always kept on changing from the previous lab and so on. There have been many changes and improvements to the code since the previous Lab. Since this lab is all progression and constantly improving the code, we can see as a whole that the structure tended to change over time. As more features were incorporated, a few examples were the Data memory and the External memory incorporated. The Control unit had upgrades with the load store commands and stack routine. Once the upgrades to the overall project was made, it was then to proceed to the debugging state with the debugger(Created and developed by Benjamin Doiron).

Test Plan

The overall test plan for this system is through both simulation and through hardware. For the simulation, we tested the behavior of each components and the entities through a test bench. For the hardware test plan, the plan is to input assembly instructions from the keyboard and compare the expected result with output on the VGA. The instructions input are based on different type instructions. For the purpose of this lab, the type of instructions are of formats R-format, I-format, and D-format. Since debug unit is in the debugging stage where it is not finalized yet. At this point we have slightly tested the hardware, but the results are not favorable.

Simulation:

For part 1 of the project, we created test benches for each component in the Control entity and FPU entity, then we created test benches for each entity. Those various test benches that where shown in figures 8 through 17 of Project-Lab1 report.

Since every type of component was tested earlier, we focused on testing the integration of these components. Figure 10, shows a simulation for the UMD RISC 24. This simulation demonstrate how the design will react under certain instructions. This is also elaborated in Table 7 in the Hardware Test Plan.

Hardware Test Plan:

In the Hardware Test Plan, we simulated tested the RISC device with 7 instructions to validate the proper output. Currently with the testing, there were a few bugs appeared and still in the process of ironing the bugs out. As demonstrated in both Table 7 and in Figure 10. Mainly, these bugs are due to that the data hazards that not being considered in the RISC24.

	Inputs				Output			Test Results
	Clock cycle	alu_opcode	store_data	dst_addr	alu_out	ldst_out	ccr	
	1	5	0	0	0	0	4	Passed
	2	5	1	0	2	0	0	Passed
	3	0	0	0	1	0	0	Passed
	4	2	0	0	0	0	4	Failed
	5	3	0	2	0	0	0	Failed
	6	0	0	2	2	0	0	Failed

Table 7: Simulation result checks

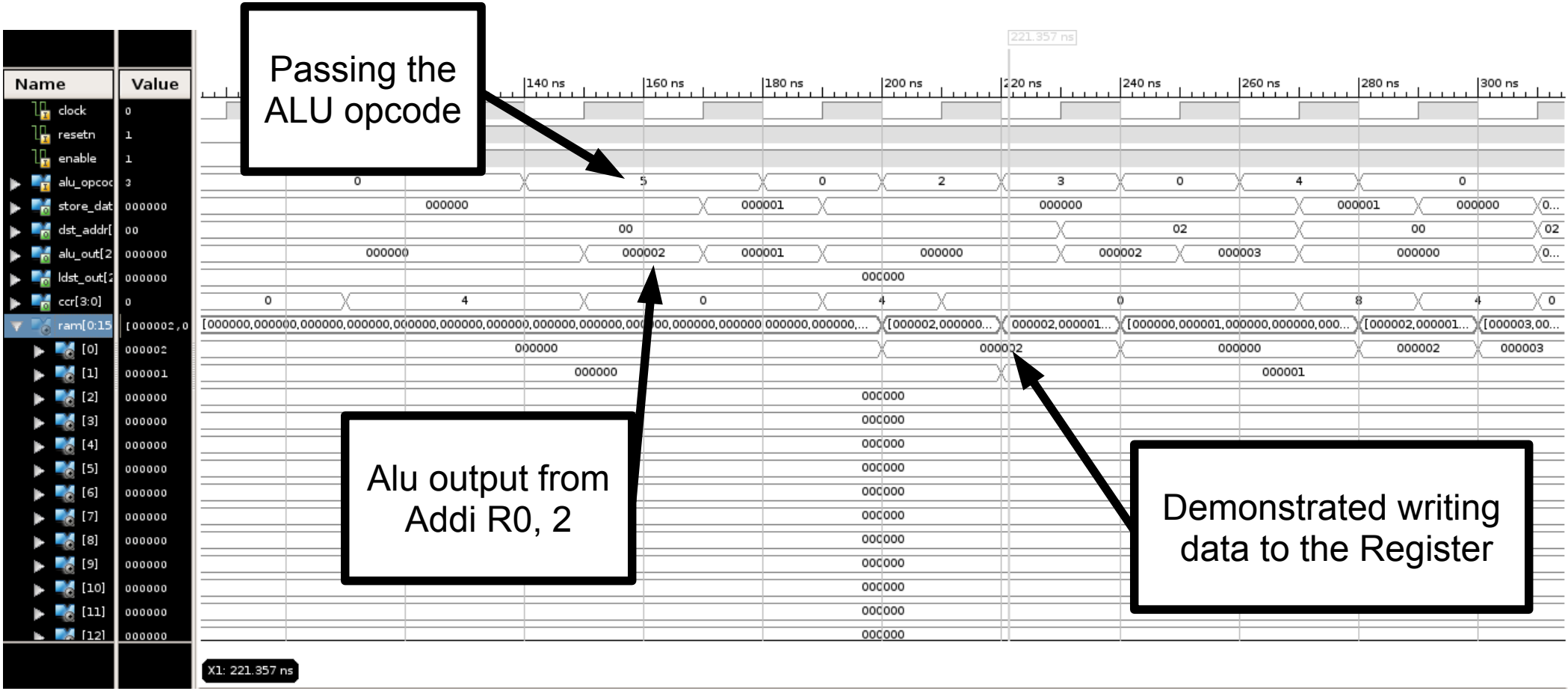

Figure 10: UMD-RISC24 simulation

UCF File

The UCF was fitted with the debug unit and emailed to Dr. Fortier and David Prairie.

Conclusion

Through this lab we learned a lot more about component designs and building functional entities. We also understood the operation of RISC machine through pipeline registers and control unit. We also learned about the implementation of the debug unit. This lab was helpful for the progress of the entire project where we have the basics to advance and implement more complicated instructions such as Branch Instructions.

We finalized the control unit, data / external memory, instruction memory, and program counter. We tested these entities through test bench simulations. These simulations were helpful in debugging and the resulting integration was easy to eliminate issues in the design. We noticed a bug in the results and that is due to the data hazards that will be implemented as progress.

Reflection:

This Lab taught us a lot on being a designer, in the previous lab we created code from scratch and we progressed in the design and implementation of it. We can see further improvements on the overall design and coding structural. For this lab, we split the work among each other for the RISC. Massarrah proceeded to develop the control unit and the stack routine. While Daniel created the data memory and external memory control plus the program counter. We also had an honorary group member Benjamin Doiron who joined with us and developed and created a debug unit that will maintain the UMD-RISC24 on hardware testing. This project would not have been done without the effort put in by all of us.

The work we created and collaborated on helped us as a whole to in further developing the project. We discussed new ideas and gained a better understanding of the design. Right now as a whole, the RISC is complete with minor bugs. These bugs are due to data hazards that are not considered in the RISC24. The data hazards could be considered through the concepts data forwarding. In data forwarding the previous cycle data that was calculated was evaluated based on the register changed from the previous cycle. The main debug unit that the RISC sits upon is complete but still needs to debug the VGA display. Once that we completed the primary objectives, we can further analyze the structure and hardware development of the overall design. This project has been both a challenge and a reward overall, we devoted much resources into the design and came out with a prototype RISC. Even though the prototype of the RISC is still buggy, it is a step in the right direction on completing the Project.

ECE 368
Digital Design
Spring 2014

Project: Lab2 – UMD RISC 24

Appendix A: Component Specification Design Layout Section