ECE 368 Digital Design

Spring 2014

Project: Lab1 – UMD RISC 24

Dates Performed: Wednesday, March 5th, 2014

Submission Date: Wednesday, April 2nd , 2014

Team #2 :

Massarrah Tannous __________________________

Daniel Noyes _________________________________

Table of Contents

Problem Statement

This lab is considered part 1 of the project. The main purpose of this lab is to build a pipe-lined ALU. This is done via two entities, control entity and FPU entity. The control entity is considered the center of the machine that controls the flow of the instruction in the pipeline. Whereas the Functional Processing Unit (FPU) entity builds upon the ALU that was implemented in the previous laboratory. The main purpose of the FPU is to process the instruction where several registers must be used, such that general purpose registers.

In order to complete this lab, we had to build the components of both entities based on figure 1 that was given. In addition, simulate those components through test benches. Moreover, integrate the two entities with the keyboard and the VGA. Through this integration, the assembly instructions are inputted through the keyboard and the results of the FPU is outputted on the VGA.

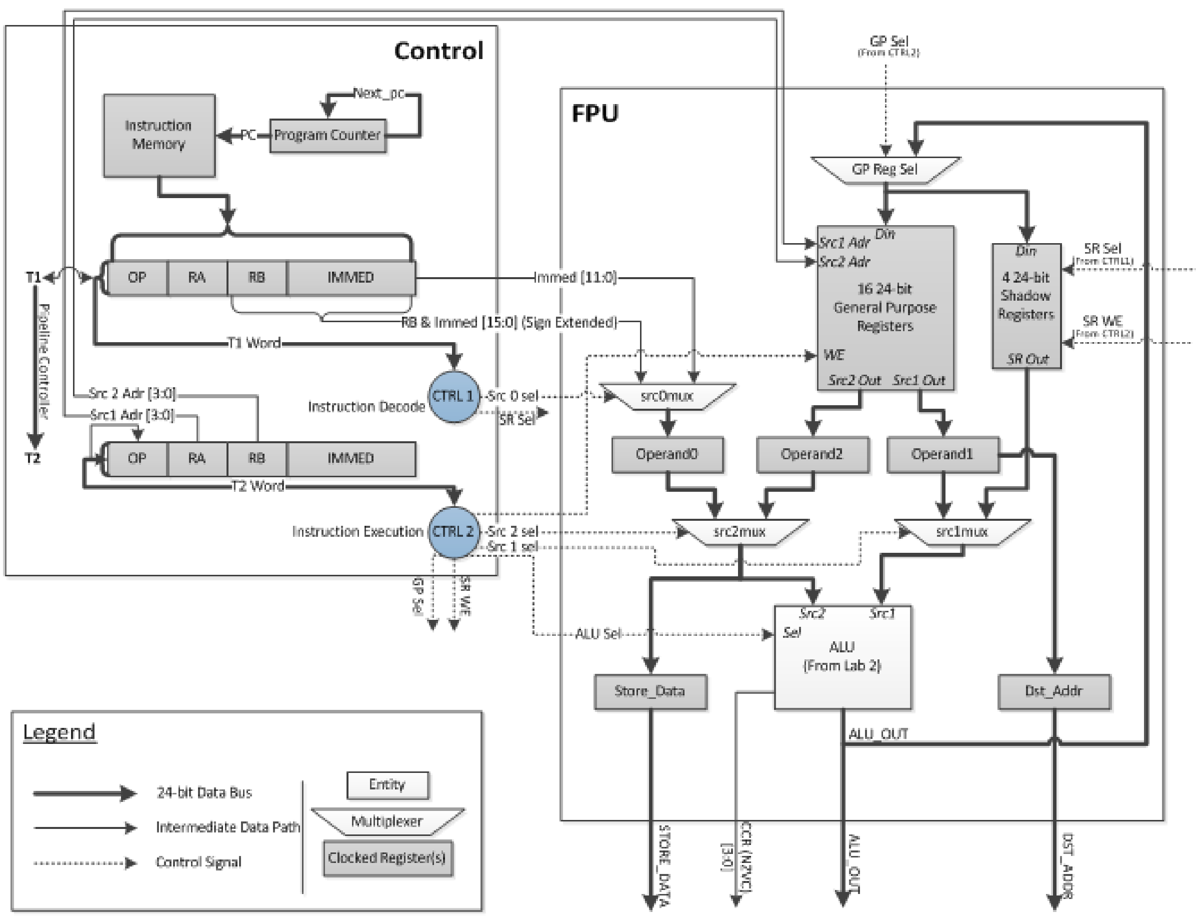

Figure 1: Pipelined FPU Data and Control Paths

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. Part of figure 2 is surrounded with a polygon which represent the parts implemented for the purposes of this lab. The attached papers, figures 18-20, are the original hand drawn design for the overall system entities (figure 18), control unit (figure 19), FPU (figure 20).

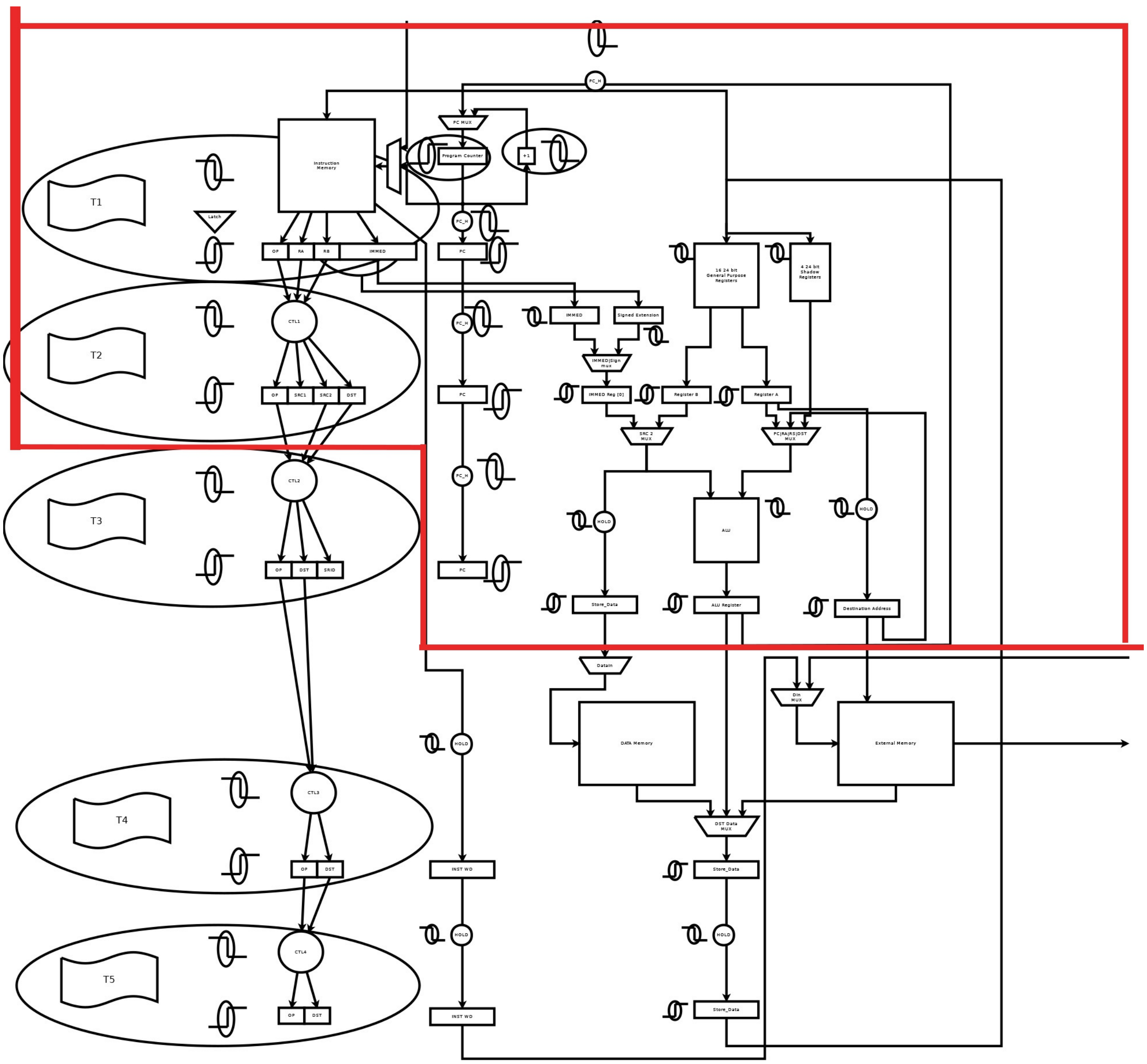

Figure 2: Original Design

VHDL component specification and schematic designs

The specifications of each components used for the Control unit and the FPU controller are shown in Appendix A (Figures A1.1 through A.17). 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

As shown in the hand drawn designs, figure 18, the overall system consists of several entities: PS2 controller, Debug Unit, VGA controller, Control Unit, FPU Unit.

The specifications and the schematics for the PS2 Controller and the VGA are similar to Lab 3 (figure 2 and 3 of lab 3). Whereas the debug unit is not finalized yet so the final schematic and specifications are not available. The following shows the specification and schematic for the Control Unit, FPU, UMD_RISC24. The UMD RISC 24 contains the control entity and the FPU.

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.
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 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

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

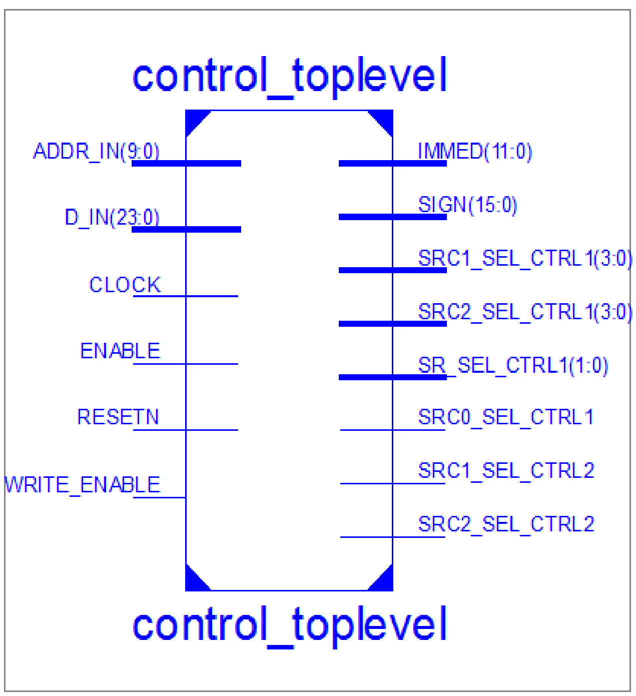

Figure 3: Top level of the control unit

Schematic

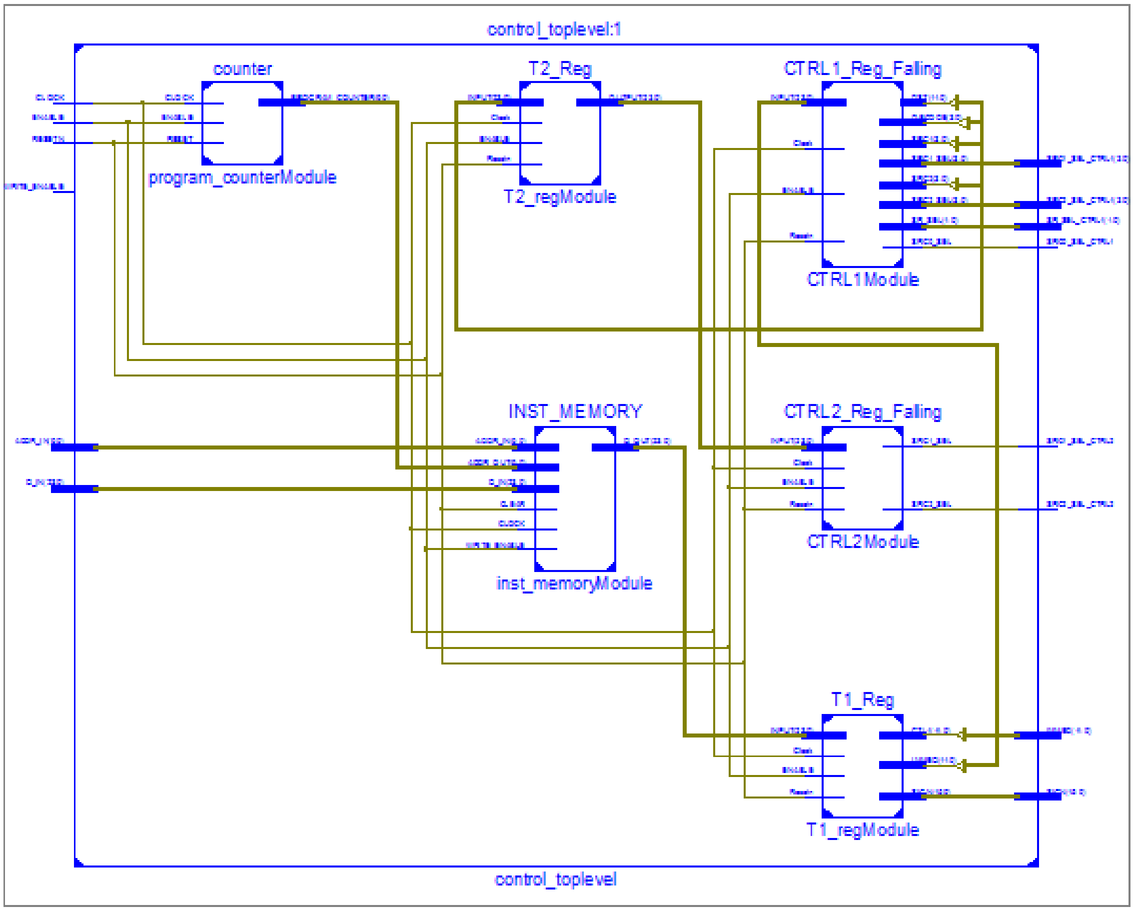

Figure 4: RTL diagram of the control unit v1.

FPU Entity

Specification

The main purpose of the FPU entity is to maintain registers and arithmetic operations based on the ALU. The top level of the FPU is shown in figure 5 and its corresponding RTL diagram is shown in figure 6. Tables 3 and 4 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.
SIGN	Signed data from the instruction
IMMED	Immediate data from the instruction
REG_DATA_IN	Register Data input for general and shadow registers
SRC1	General purpose Reg A Address
SRC2	General purpose Reg B Address
GP_DIN_SEL	Reg Address to write to
GP_WE	Write enable for general purpose register
SR_SEL	Shadow Register Select
SR_DIN_SEL	Shadow Register to write to
SR_WE	Write enable for shadow register
SRC0_SEL	Select between immed and signed extension
SRC1_MUX	Select between PC, RA, SR, DST_ADDR (4 select)
SRC2_MUX	Select between immed or Reg B (2 select)
ALU_OPCODE	OPCODE for the ALU
PC	Program Counter

Table 3: Description of the control unit inputs.

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 for the control unit.

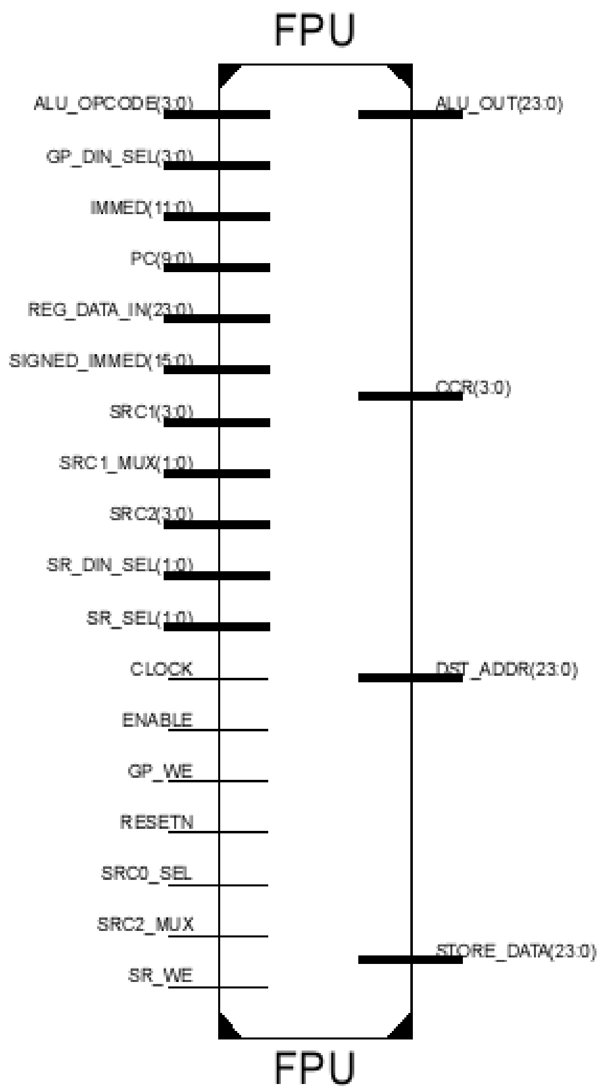

Figure 5: Top level of the FPU

Schematic

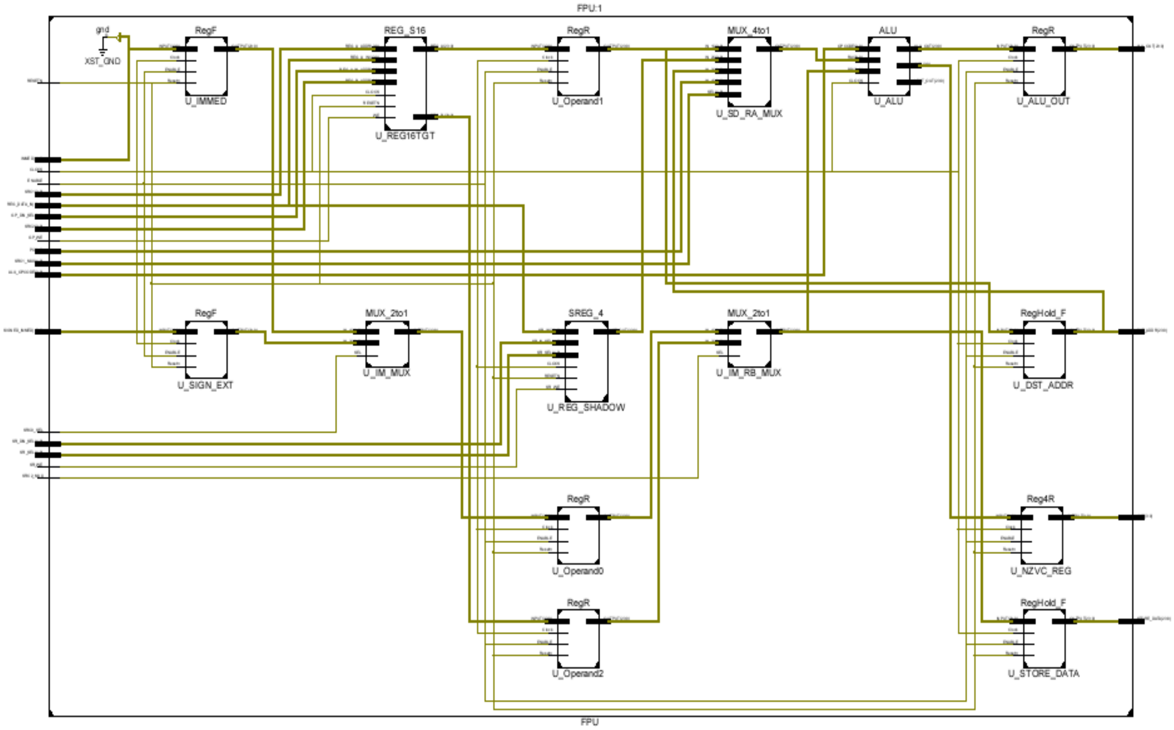

Figure 6: RTL Diagram of the FPU entity.

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 7. Tables 5 and 6 show the description of the inputs and the outputs corresponding to figure 7.

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 5: 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 6: Description of the outputs of the UMD RISC24.

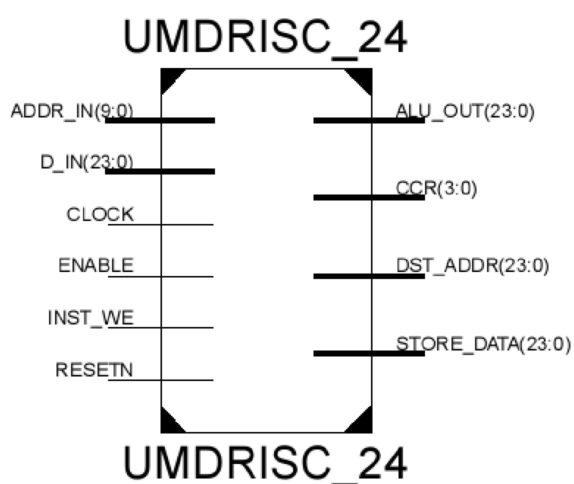

Figure 7: Top level for the UMD RISC24

VHDL Code and Test Bench Code

The VHDL code for both the VGA has been emailed as a zip file to Dr. Fortier and to David Prairie.

Designs comparisons

The original designs schematics (figures 18-20) are on based upon Dr. Fortier's UMD RISC24 schematics in the lab handout. While analyzing the schematics and testing we noticed several flaws in the given schematics. The main errors were in the shadow registers and the shadow registers. More errors were notice in the write enable signals in which they were supposed be in the write back stage of the control component. As shown is the result design schematic of the control unit (Figure 4), the CTRL2 component doesn't contain gp_sel signal part of the output. The purpose of the gp_sel signal is to enable write back to the general purpose registers were it supposed to be part of CTRL6.

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 entity is not finalized yet, at this point we haven't tested the hardware.

Simulation:

We created test benches for each component in the control entity and FPU entity, then we created test benches for each entity.

The following figures (8 through 17) are the timing diagrams of the various test benches. Figure 8 shows a general purpose register that is capable of load and store data. Figure 9 shows the simulation for a program counter. As shown in Figure 10, a 24 bit register is created and it latches data on the falling edge in the clock. This simulation also shows a 5 stage pipeline of figure 2. Whereas figure 11 shows a hold and latch register that latches data on rising edge and hold the data on falling edge. This represent the general functionality of CRTL1, CTRL2, CTRL3, CTRL4, and CTRL5 of figure 2. The FPU contains several type of multiplexers to select between different type of data based on the opcode. Figures 12, 13, and 14 shows a 2 to 1 Multiplexer, 3 to 1 multiplexer, and 4 to 1 Multiplexer. Whereas figure 15 shows a simulation for a keyboard. Figure 16 is a simulation of the Control Entity, whereas figure 17 is the simulation for the FPU.

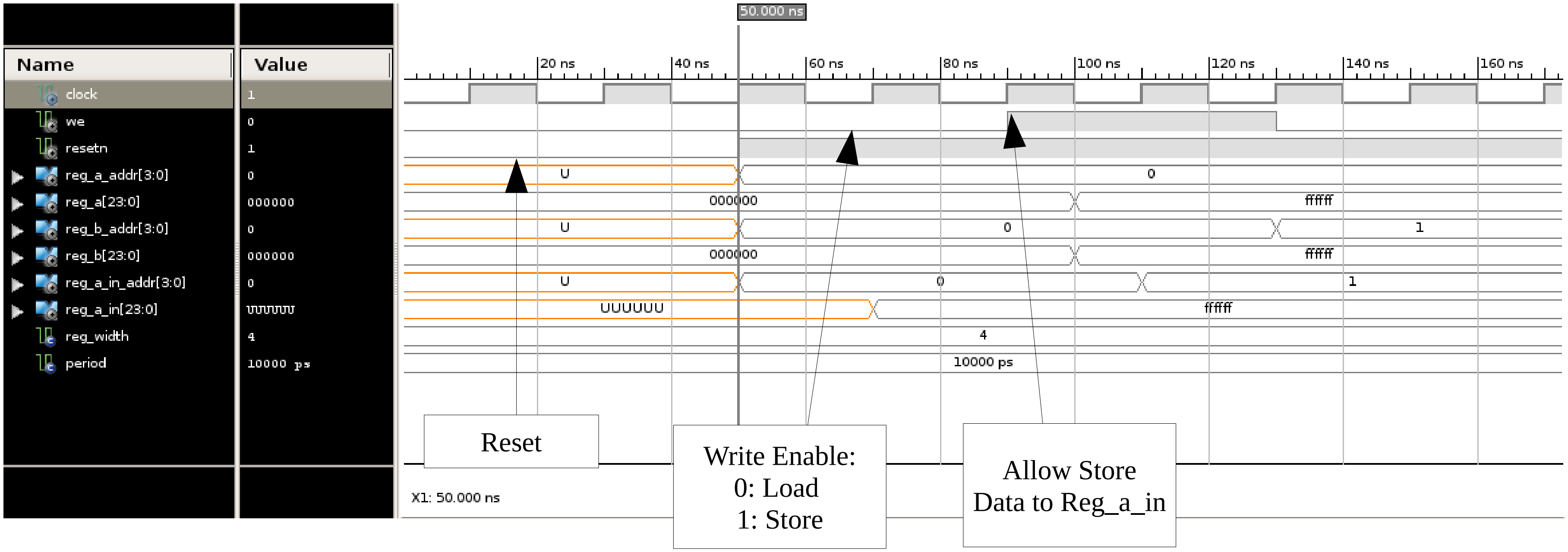

Figure 8: Timing Diagram of the general purpose register.

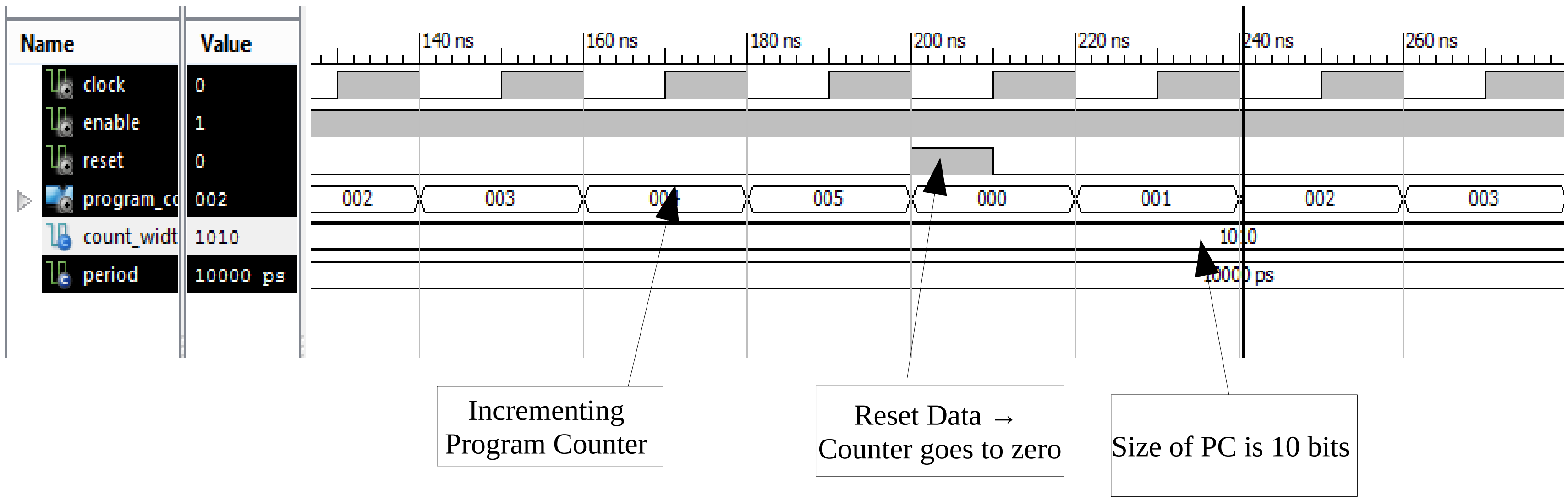

Figure 9: Timing Diagram for the Program Counter

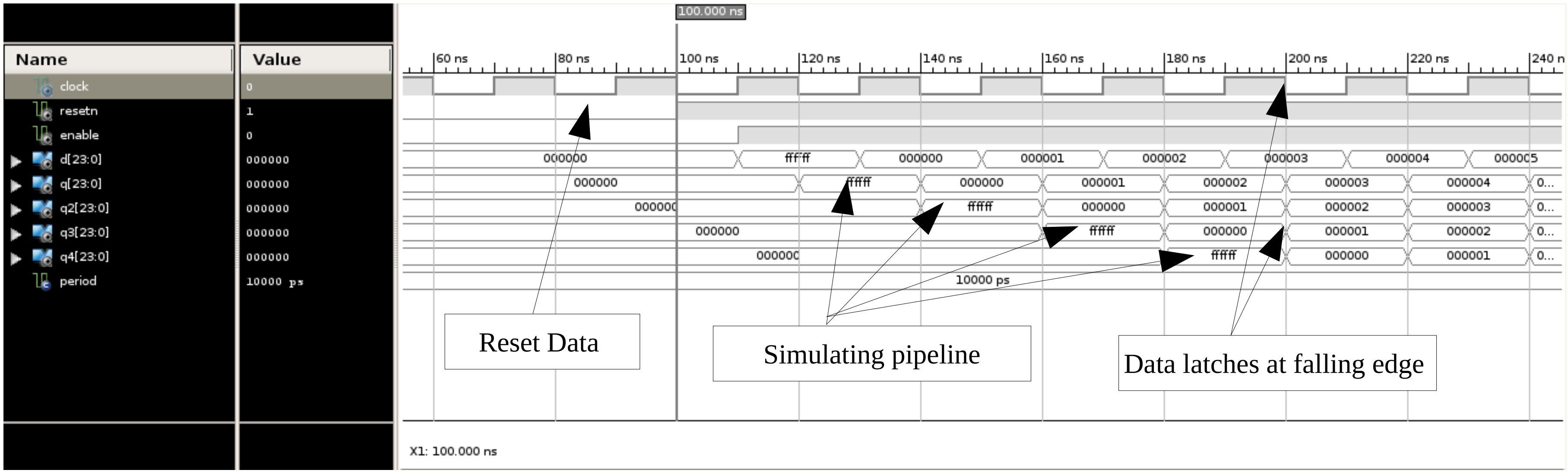

Figure 10: Timing diagram for a register that latches data at falling edge.

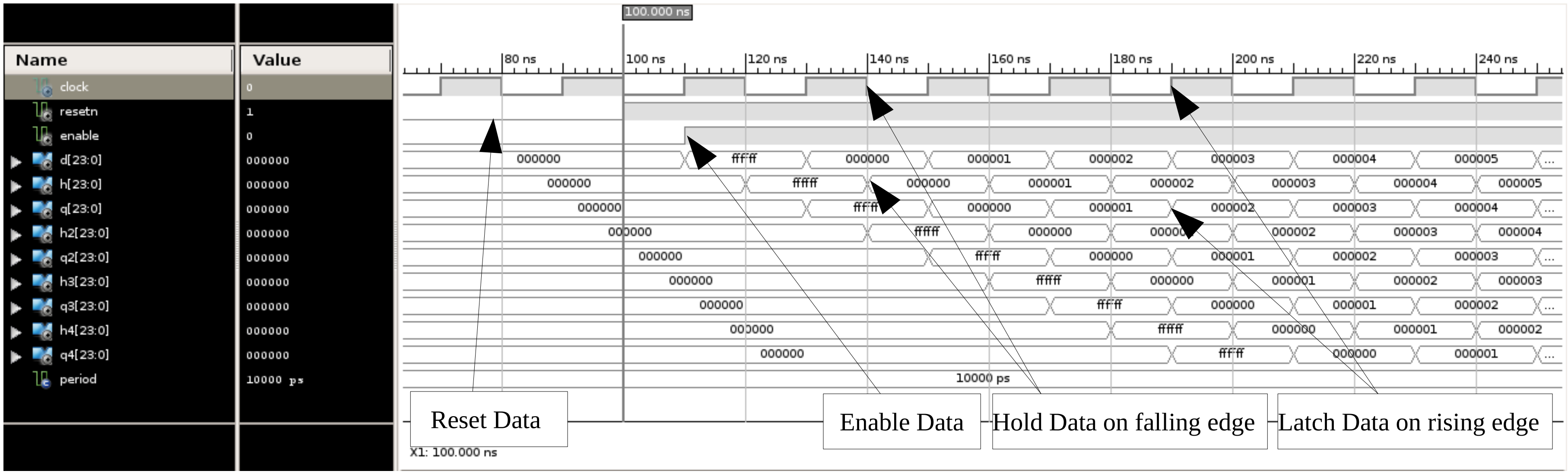

Figure 11: Timing diagram for a hold and latch register.

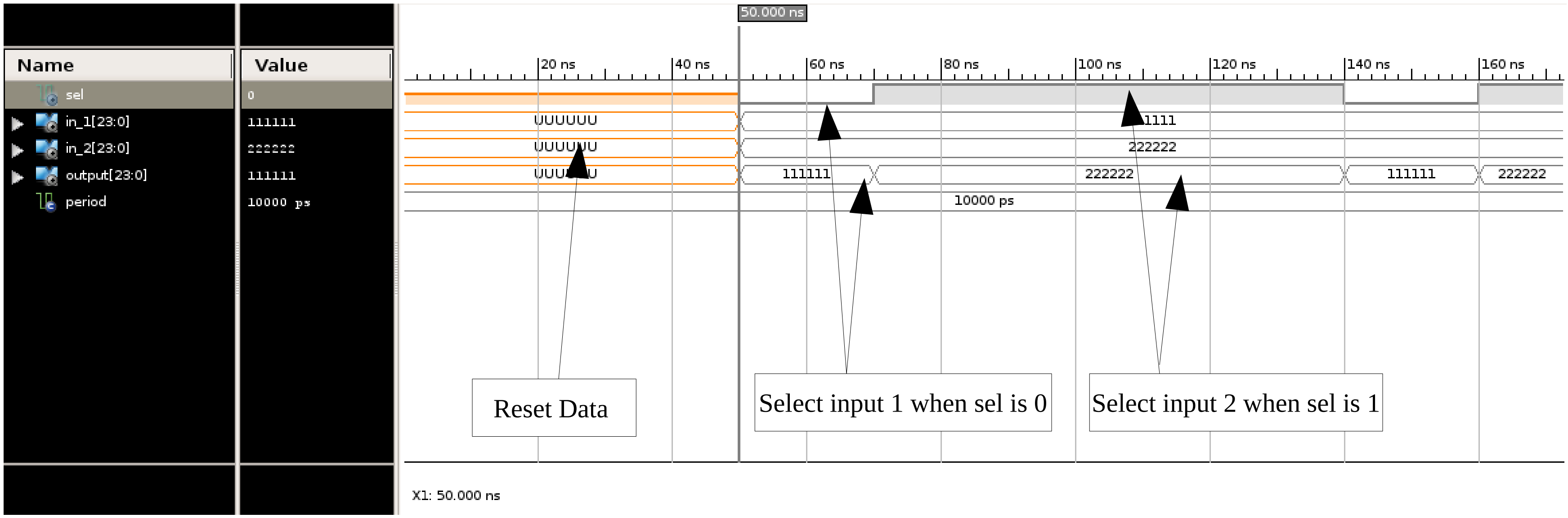

Figure 12: Timing Diagram for 2 to 1 multiplexer.

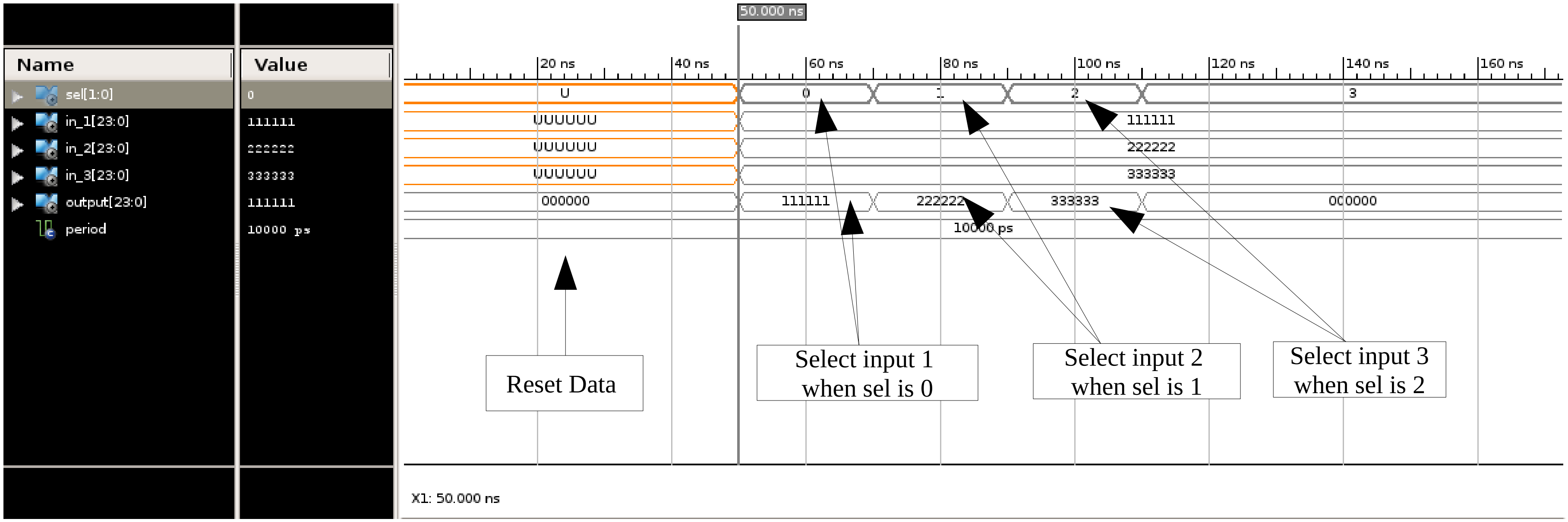

Figure 13: Timing Diagram for 3 to 1 Multiplexer.

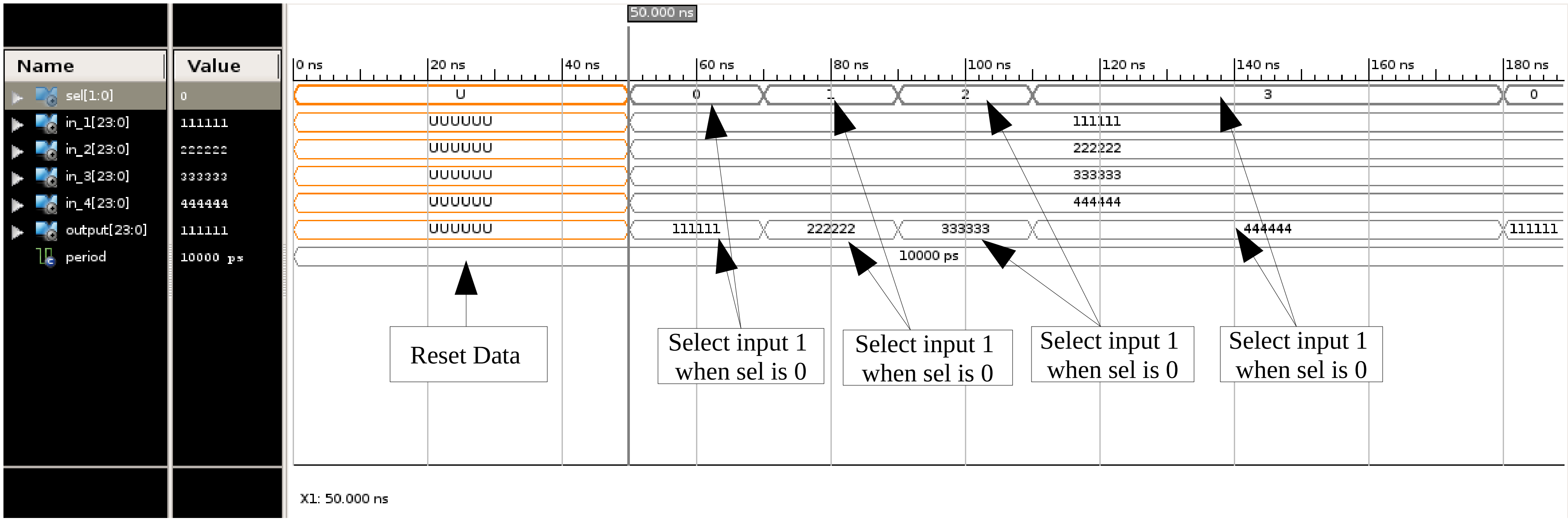

Figure 14: Timing Diagram for 4 to 1 Multiplexer.

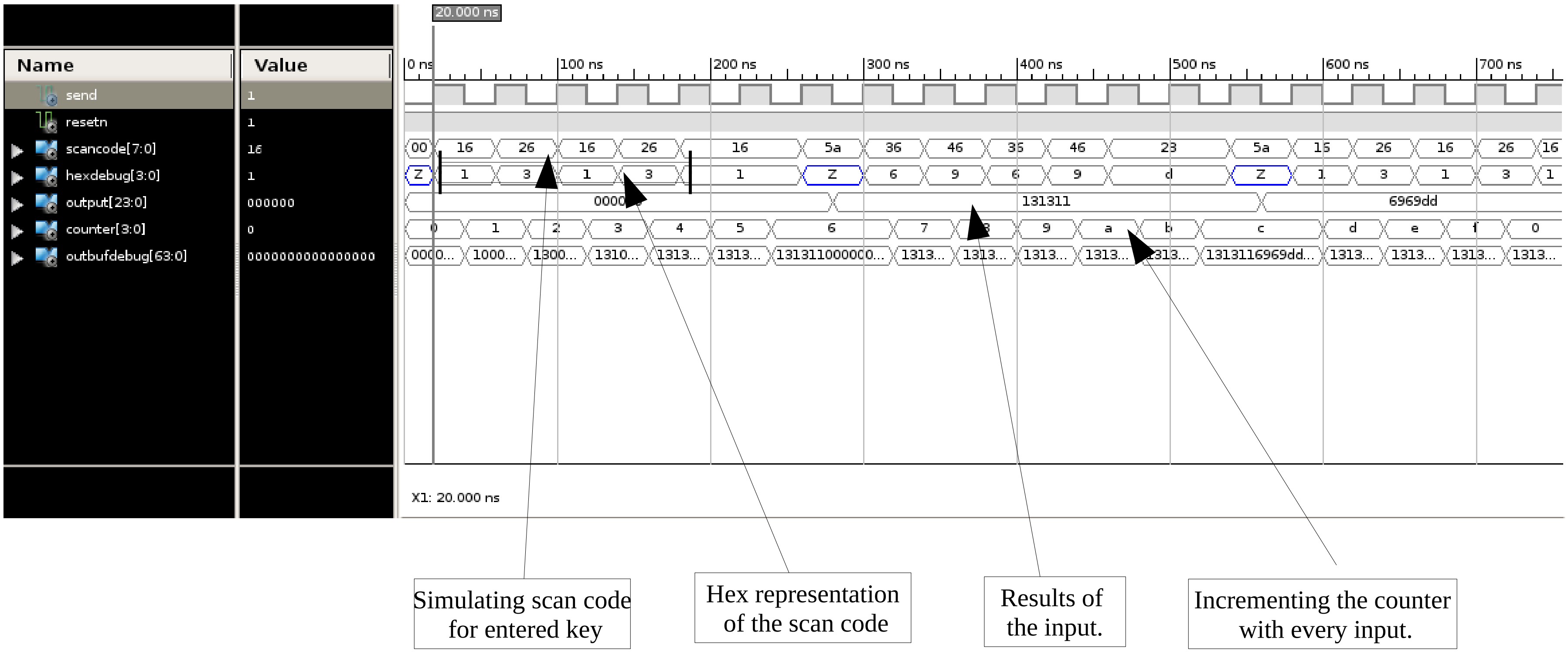

Figure 15: Timing Diagram for keyboard input.

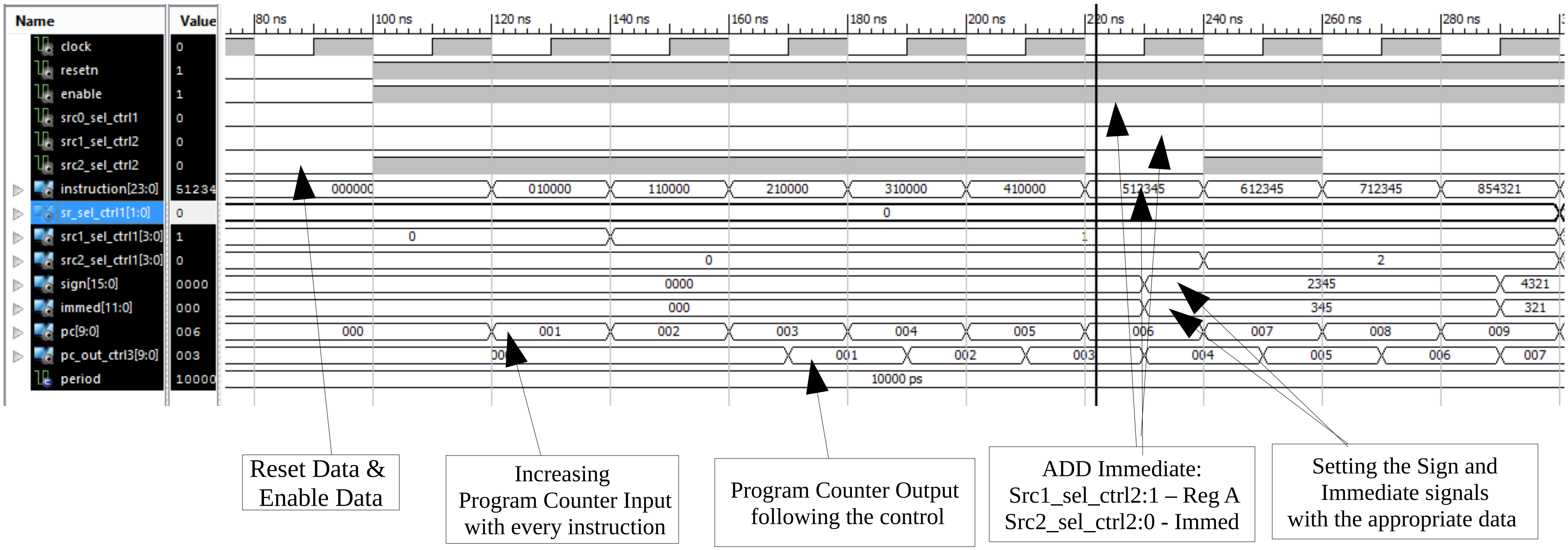

Figure 16: Timing Diagram for the Control Unit

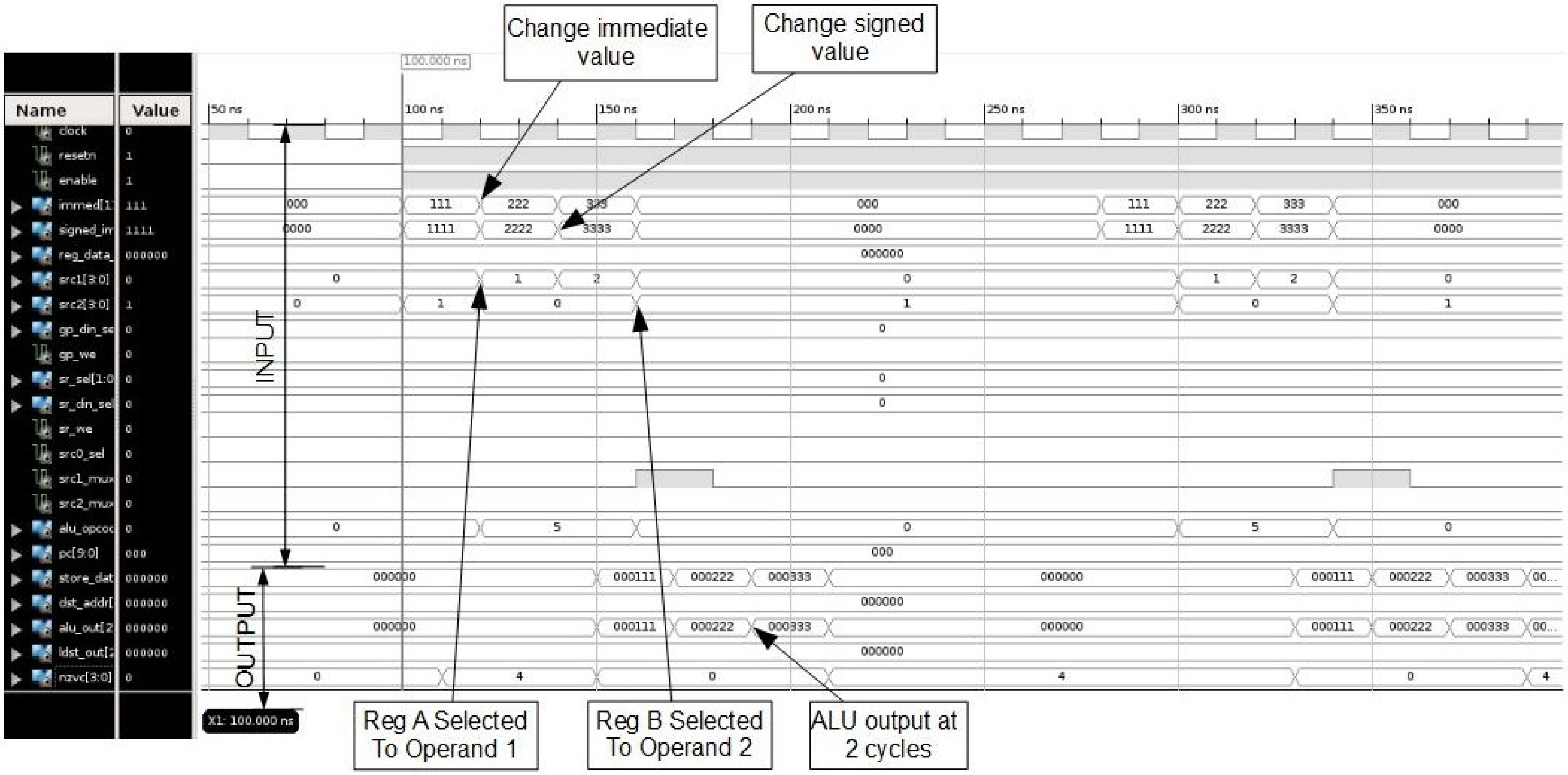

Figure 17: Timing Diagram showing the functionality of the FPU entity.

UCF File

Since we are not at the stage at which the VGA is finalized. The UCF for the VGA controller and PS2 controller are not finalized yet.

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 for simple instruction such as ADD Immediate. This lab is helpful for the basis of the entire project where we have the basics to advance and implement more complicated instructions such as Branch Instructions.

We implemented the control and the FPU entities and their components. 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.

Reflection

This Lab overall was a big splash, now we are learning to develop and design code from scratch. This is the first time that we implemented a device with no common base. In previous labs, we only had to adjust a few lines of code. Unlike this lab, we did not have the luxury of pre-existed code to build on. After developing a basic overlay of the RISC architecture from Professor Fortier's notes, More knowledge and experience went through. Overall, we are enjoying the learning process and implementing what we learned in this project. This lab is still incomplete and still has a few top level components to be tested. One of the problems that we faced in this lab was learning how Xilinx synthesis the code.

One main part was building instruction memory and trying to have a reset function built in and that resulted in 40% of the chip being used. Upon observation and talking with Dr. Fortier, it was stumbling on the reason why the synthesis exploded. The first reason why it exploded was that the Xilinx package thought to make the code into Muxes and filled most of the real-estate of the FPGA. The way to prevent it from doing this was to take out the clear utility that was implemented in the design. An alternative approach is still being explored to see what is the best option for the project.

With the help of spring break, creating and developing code was easier with the less work load. With the project getting bigger, there will be needed time management to accomplish the code orderly. With more learning about coding in VHDL, more ideas have been populated and designed. There will be no rest from coding as the next part of the project is to implement more functionality.

ECE 368
Digital Design
Spring 2014

Project: Lab1 – UMD RISC 24

Appendix A: Component Specification Design Layout Section

Control Unit- Top Level

Inputs:

CLOCK, RESETN, ENABLE, WRITE_ENABLE : <std_logic>

ADDR_IN <std_logic_vector : <std_logic_vector: 9 down to 0>

D_IN : <std_logic_vector: 23 down to 0>

Outputs:

SR_SEL_CTRL1 <std_logic_vector: 1 down to 0>

SRC0_SEL_CTRL1, SRC1_SEL_CTRL2, SRC2_SEL_CTRL2 : <std_logic>

SRC1_SEL_CTRL1, SRC2_SEL_CTRL1 <std_logic_vector: 3 down to 0>

SIGN <std_logic_vector: 15 down to 0>

IMMED <std_logic_vector: 11 down to 0>

Functionality: The control entity is considered the center of the machine that controls the flow of the instruction in the pipeline. Passes signals from the debug unit into its components

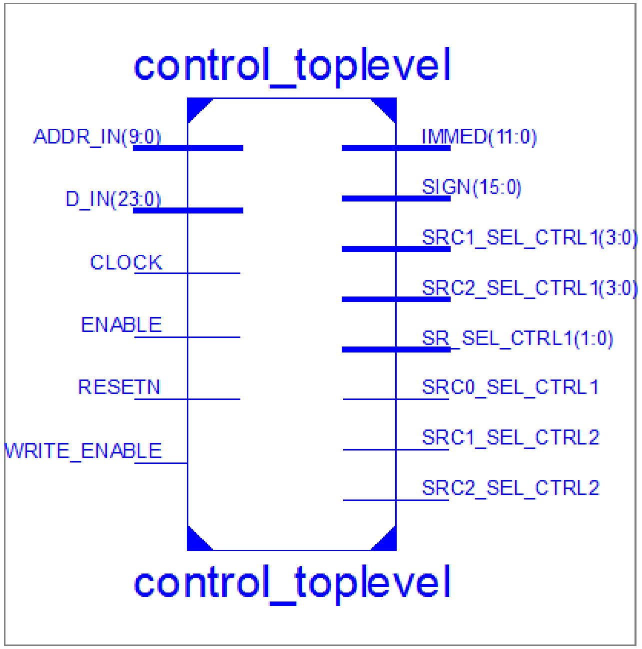

Figure A1.1 The top level of the control entity.

Control Unit- Components

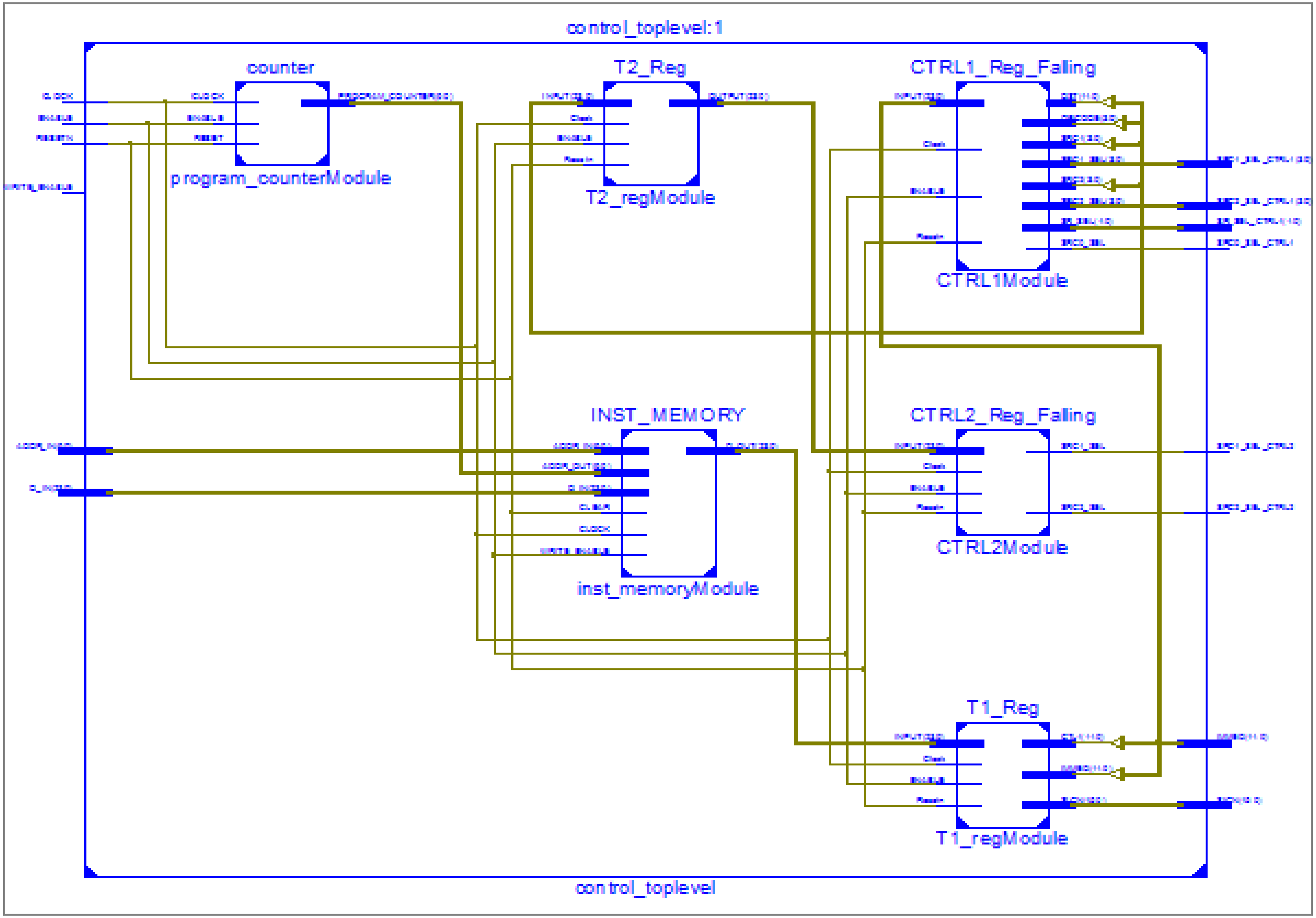

Figure A1.2 Components of the Control Entity on their connections.

Control Unit- Counter

Inputs:

CLOCK, RESETN, ENABLE, Reset : <std_logic>

Outputs:

Program_Counter <std_logic_vector: 9 down to 0>

Functionality: This components is the program counter where it automatically increments with every clock cycle.

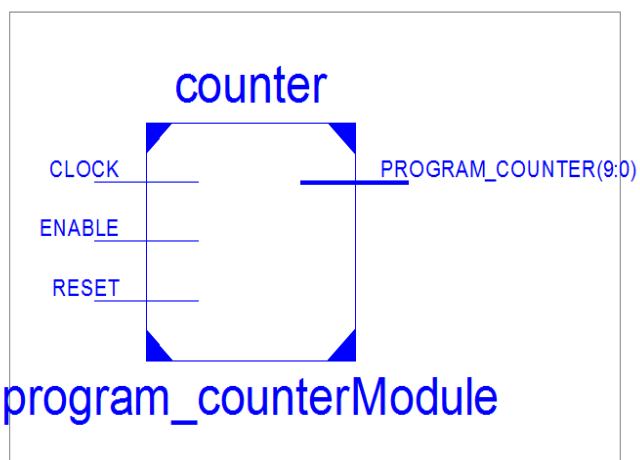

Figure A1.3 The program counter component in the control entity .

Control Unit- Inst_Memory

Inputs:

CLOCK, CLEAR, WRITE_ENABLE : <std_logic>
ADDR_IN,ADDR_OUT <std_logic_vector : <std_logic_vector: 9 down to 0>
D_IN : <std_logic_vector: 23 down to 0>

Outputs:

D_OUT : <std_logic_vector: 23 down to 0>

Functionality: This component is the instruction memory that acts as read-only register containing 1024, 24-bit it is indexed by the program counter ADDR_OUT. The signal ADDR_IN was used for debugging purposes.

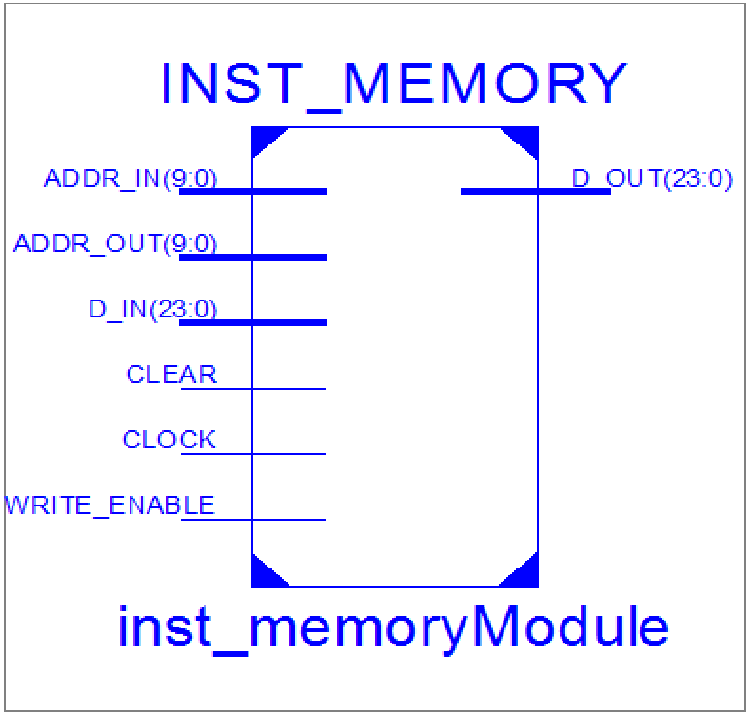

Figure A 1.4 The instruction memory component in the control entity .

Control Unit- T1_Reg

Inputs:

CLOCK, RESETN, ENABLE, WRITE_ENABLE : <std_logic>
Input : <std_logic_vector: 23 down to 0>

Outputs:

CTRL1 <std_logic_vector: 11 down to 0>
SIGN <std_logic_vector: 11 down to 0>
IMMED <std_logic_vector: 15 down to 0>

Functionality: This component latches instructions from the instruction memory on rising edge and passes the information to Control Register 1. This acts as the first pipeline for the RISC machine.

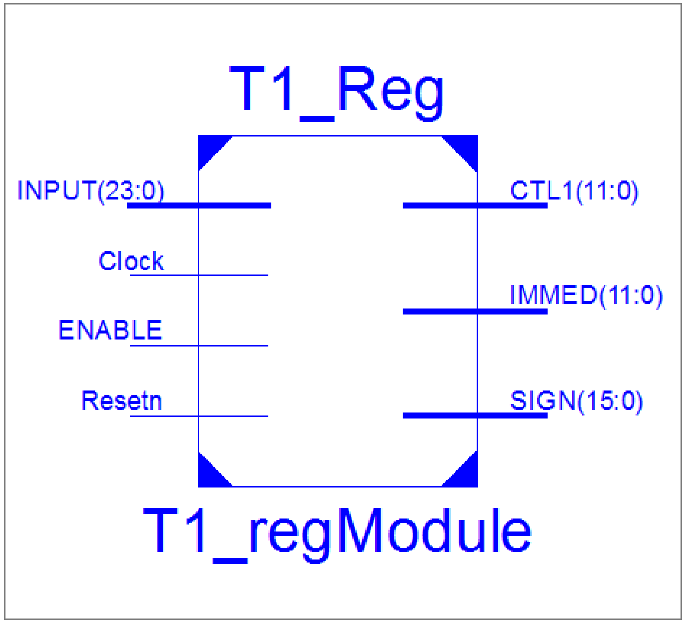

Figure A1.5 The 1st pipeline component in the control entity .

Control Unit- CTRL1_Reg_Falling

Inputs:

CLOCK, RESETN, WRITE_ENABLE : <std_logic>
Input : <std_logic_vector: 23 down to 0>

Outputs:

SR_SEL <std_logic_vector: 1 down to 0>
SRC0_SEL, SRC1_SEL, SRC2_SEL : <std_logic>
OPCODE, SRC1, SRC2 <std_logic_vector: 3 down to 0>
SIGN <std_logic_vector: 15 down to 0>
DST <std_logic_vector: 11 down to 0>

Functionality: This component latches instructions from the T1 register on falling edge and decodes and passes the data on T2 register.

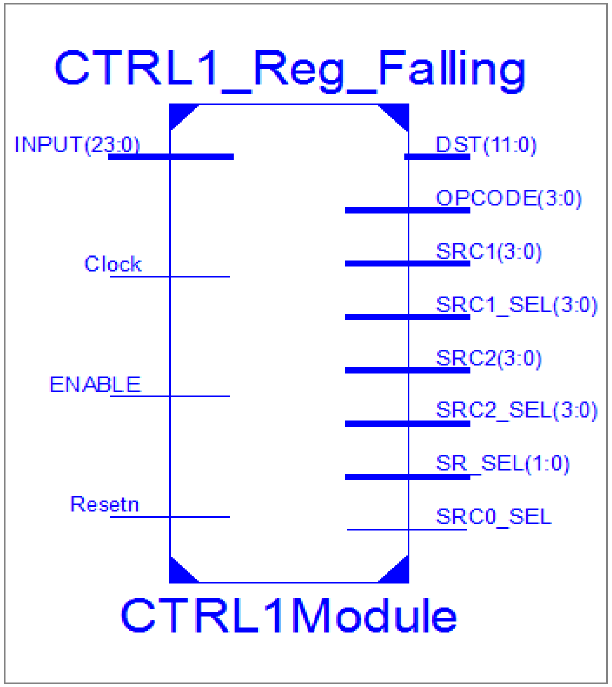

Figure A1.6 The CTRL1 component in the control entity .

Control Unit- T2_Reg

Inputs:

CLOCK, RESETN, ENABLE, WRITE_ENABLE : <std_logic>
Input : <std_logic_vector: 23 down to 0>

Outputs:

Output <std_logic_vector: 23 down to 0>

Functionality: This component latches instructions from the CTLR1 register on rising edge and passes the information to Control Register 2. This acts as the pipeline for the RISC machine.

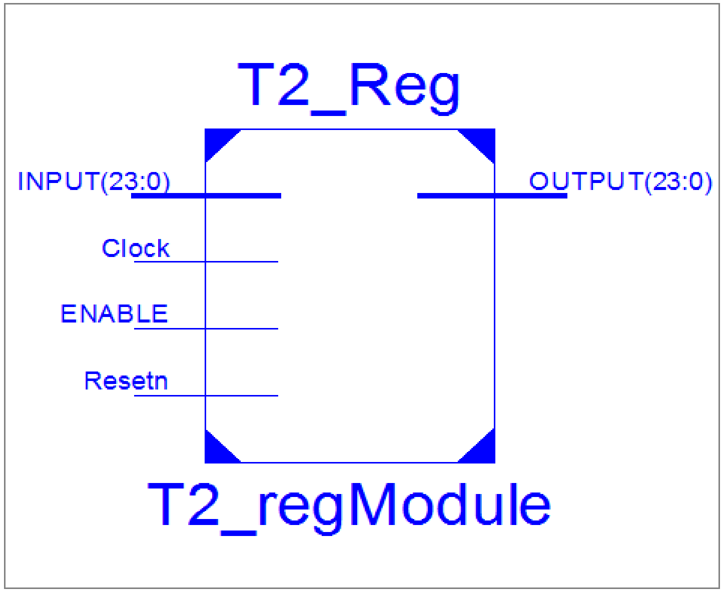

Figure A1.7 The 2[st] pipeline component in the control entity .

Control Unit- CTRL2_Reg_Falling

Inputs:

CLOCK, RESETN, ENABLE : <std_logic>
Input : <std_logic_vector: 23 down to 0>

Outputs:

SRC1_SEL, SRC2_SEL : <std_logic>

Functionality: This component latches instructions from the T2 register on falling edge and passes the information to T3 and selects the signals based on the opcode.

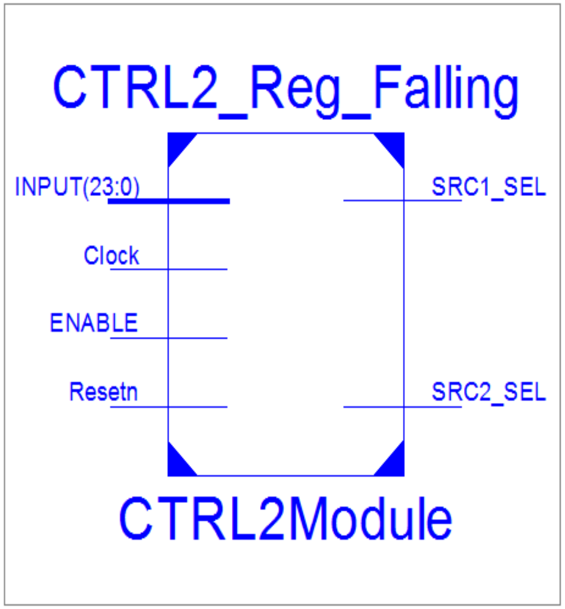

Figure A1.8 The CTRL1 component in the control entity .

FPU – Toplevel

Inputs:

CLOCK : Drives the FPU
RESETN : reset the FPU if reset = 0
ENABLE : Enable the FPU to flow through each register
-- **Data from T1**
IMMED : Immediate value 12 bits
SIGNED_IMMED : Signed value 16 bits
--**input Register**
REG_DATA_IN : Register Data input for general and shadow reg
--**CONTROL INPUT SIGNALS**
--**General Purpose Register interface**
SRC1 : General purpose Reg A Address
SRC2 : General purpose Reg B Address
GP_DIN_SEL : Reg Address to write to
GP_WE : write enable for general purpose register
--**Shadow Register**
SR_SEL : Shadow Register Select
SR_DIN_SEL : Shadow Register to write to
SR_WE : write enable for shadow register
--**MUX's**
SRC0_SEL : Select between immed and signed extension
SRC1_MUX : Select between PC, RA, SR, DST_ADDR (4 select)
SRC2_MUX : Select between immed or Reg B (2 select)
--**ALU**
ALU_OPCODE : OPCODE for the ALU

FPU – Toplevel

Outputs:

STORE_DATA : Destination Data
DST_ADDR : Destination Address
ALU_OUT : Output of the ALU
LDST_OUT : Output the load / store on the ALU
CCR : Condition Code Register (ALU)

Functionality: The FPU is designed to manipulate the data based on the control inputs. The Data can be stored in one of sixteen registers or 4 shadow registers(word). The data is then processed through the ALU which will output the data based on the inputs. The inputs can be Register A, and either immediate/signed data or register B. The output of the data will be the result from the opcode given from the control.

FPU Toplevel : RTL

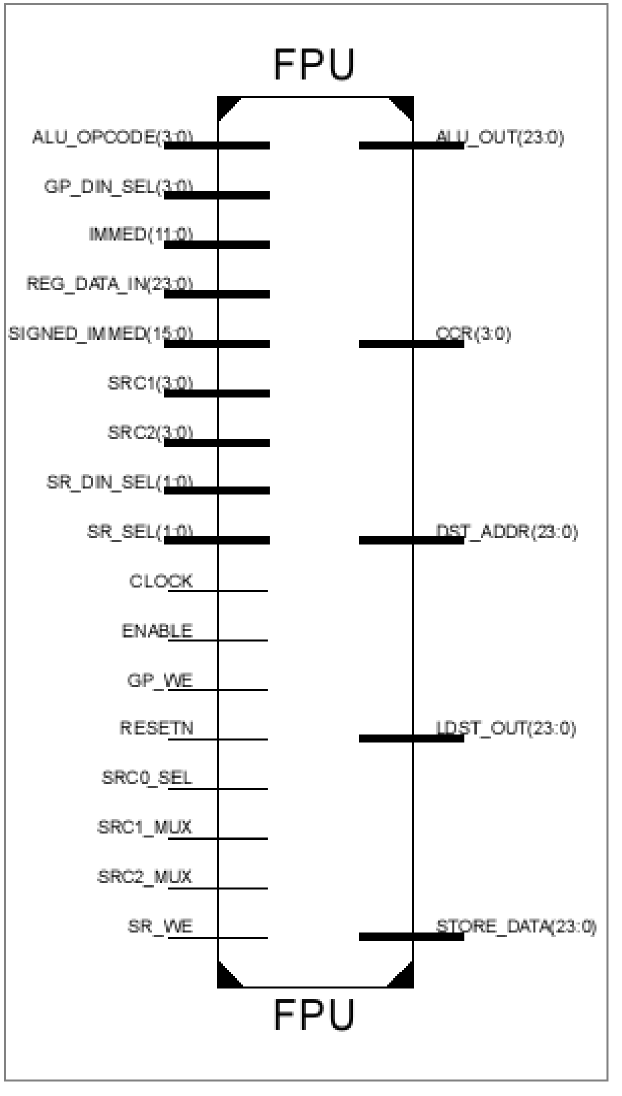

Figure A1.9 The toplevel of the FPU control entity .

FPU Toplevel : RTL inside blocks

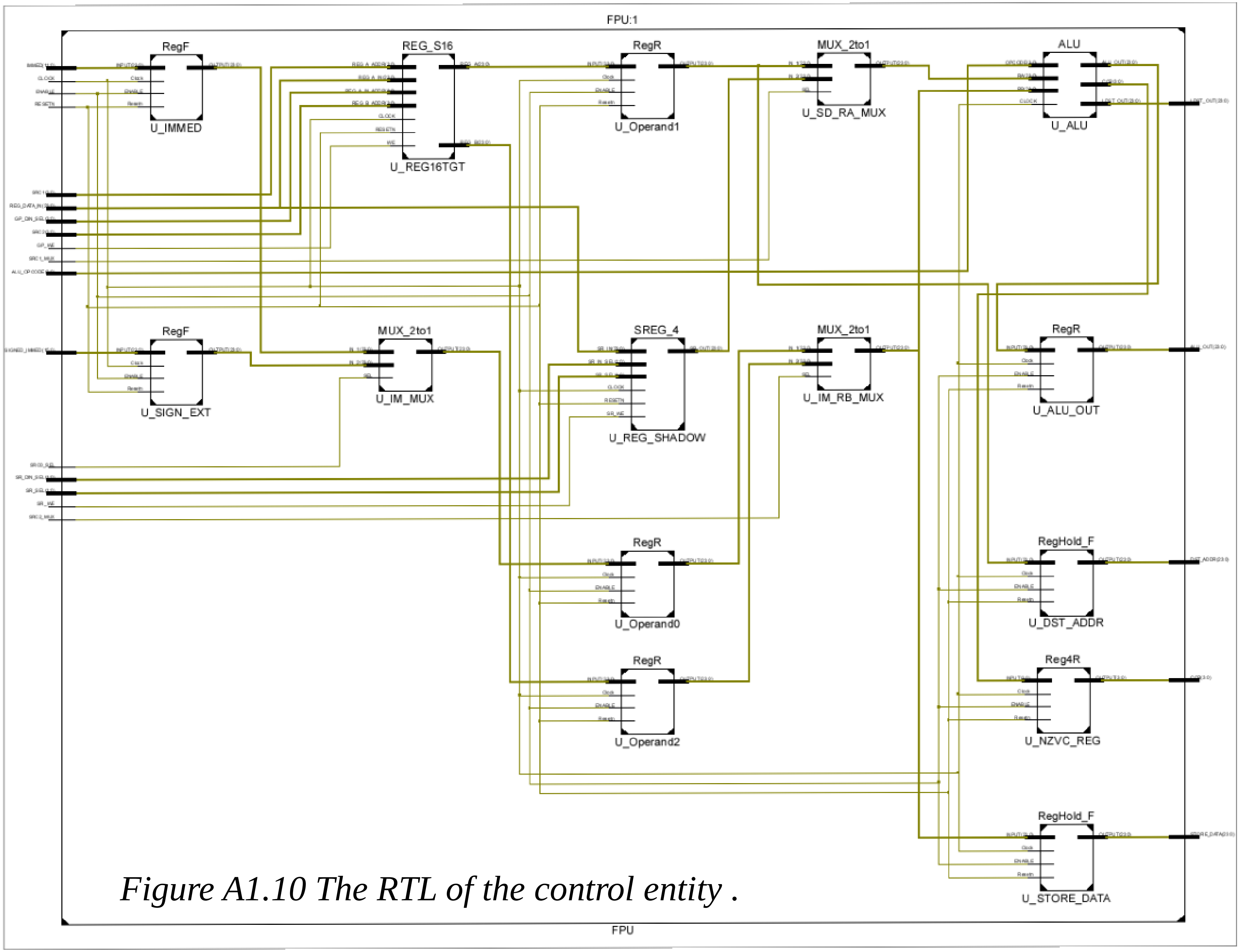

Figure A1.10 The RTL of the control entity .

RegF – Register Latch Falling

Inputs:

Clock : clock drive the register latch
Enable : enable the latching and saving state
Resetn : Reset the register when reset is a 0
INPUT : input data

Outputs:

OUTPUT : output the current data latched

Functionality: 24 bit register that latches data on the falling edge

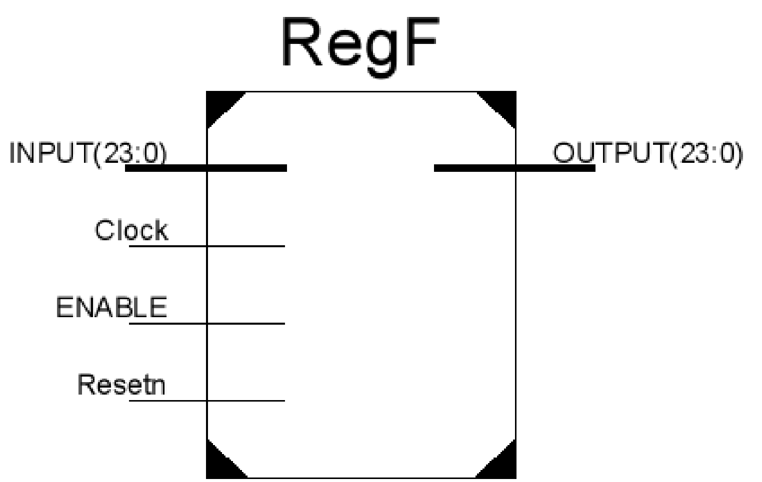

Figure A1.11 Register Falling component in the FPU.

RegR – Register Latch Rising

Inputs:

Clock : clock drive the register latch
Enable : enable the latching and saving state
Resetn : Reset the register when reset is a 0
INPUT : input data

Outputs:

OUTPUT : output the current data latched

Functionality: 24 bit register that latches data on the rising edge

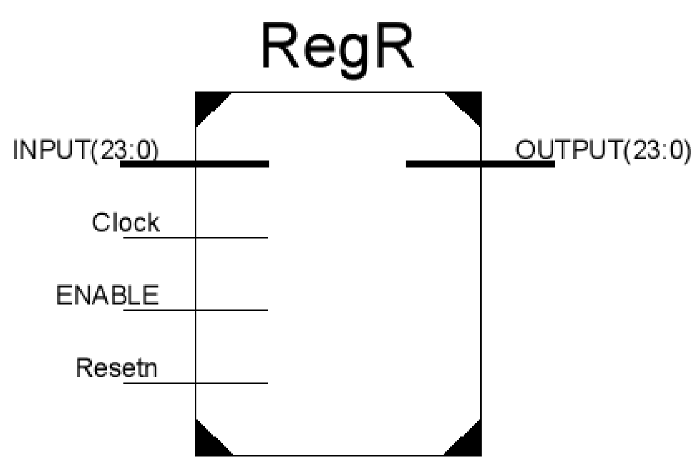

Figure A1.12 Register Rising component in the FPU.

RegHold_F – Reg with holder, latch on falling

Inputs:

Clock : clock drive the register latch
Enable : enable the latching and saving state
Resetn : Reset the register when reset is a 0
INPUT : input data

Outputs:

OUTPUT : output the current data latched from the hold register

Functionality: 24 bit register that has a hold register built in that will latch the data to the hold register on the falling edge while outputting the data on the rising edge.

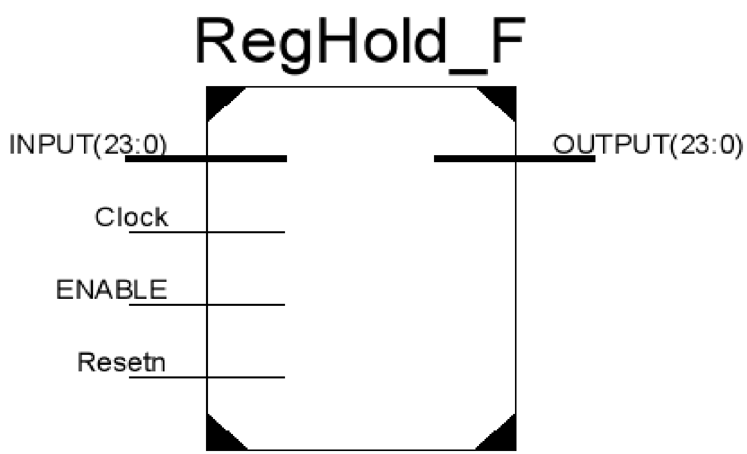

Figure A1.13 Register Hold component in the FPU.

REG_S16 – General purpose Registe

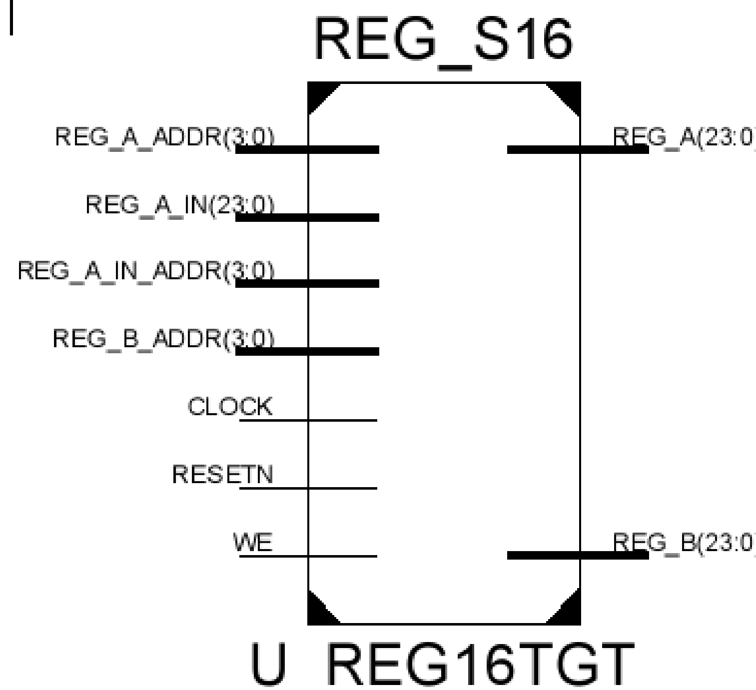

Figure A1.14 General Purpose Register component in the FPU.

Inputs:

Clock : clock drive the register latch
WE : enable write to register
Resetn : Reset the register when reset is a 0
REG_A_ADDR : Register A address (0-F)
REG_A_IN : Input Data to Register A
REG_B_ADDR : Register A address (0-F)
REG_B_IN : Input Data to Register B

Outputs:

REG_A : output data on Register A based on the REG_A_ADDR position
REG_B : output data on Register B based on the REG_B_ADDR position

Functionality: 16 24 bit registers that have the capability of loading and store data

SREG_4 – Shadow Register

Inputs:

Clock : clock drive the register latch
SR_WE : enable write to register
Resetn : Reset the register when reset is a 0
SR_IN : Input Data to the Shadow register
SR_IN_SEL : Select the input word for the register
SR_SEL : Select the output word from the register

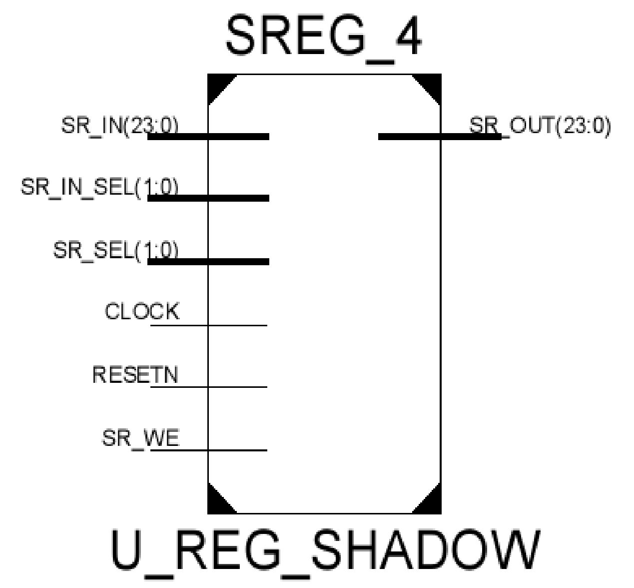

Figure A1.15 Shadow Register component in the FPU.

Outputs:

SR_OUT : output data of the shadow register based on the SR_SEL input

Functionality: 4 24 bit registers designed for special instructions with words data

MUX_2to1 – 2 to 1 Mux

Inputs:

SEL : Select bit ; 0 <= IN_1, 1 <= IN_2
IN_1 : Input data 1
IN_2 : Input data 2

Outputs:

OUTPUT : output data based on the select bit

Functionality: The 2to1 mux will output data from two inputs based on a select bit

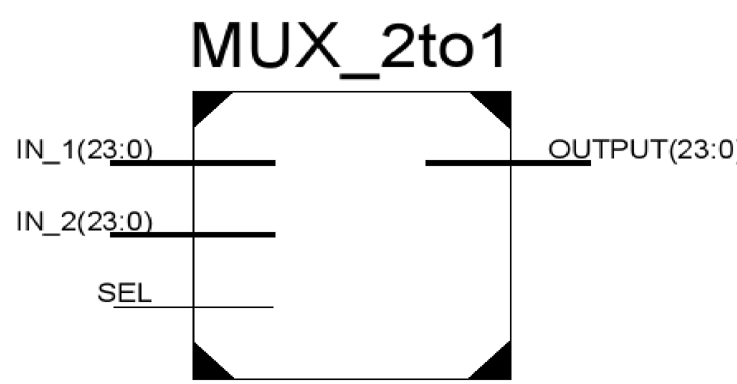

Figure A1.16 Multiplexer component in the FPU.

ALU v2 – ALU unit

Inputs:

Clock : clock drive the register latch
OPCODE : operation code, function that the ALU must do
RA : register A
RB : register B

Outputs:

ALU_OUT : output data of the ALU result based on the opcode
CCR : condition code register
LDST_OUT : load/ store output

Functionality: ALU, the arithmetic logic unit will perform specific functions based on the operation code and will output the result plus condition code based on the input data(RA,RB).

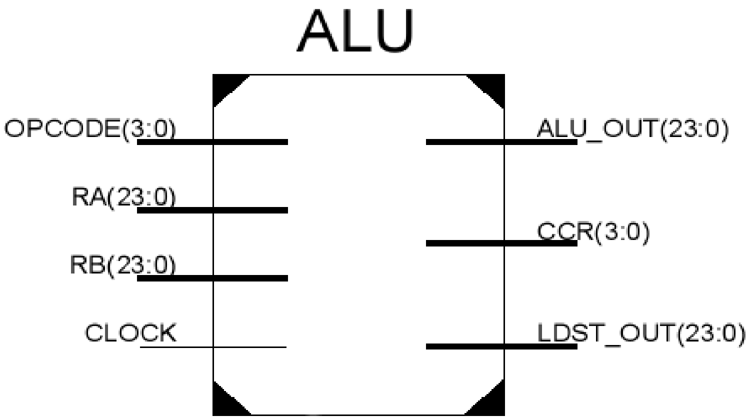

Figure A1.17 Top level of the ALU component in the FPU.