Concordia University

Laboratory Report

COEN - 316

Lab – 4

CPU Datapath

Submission Date: December 1, 2019

Prepared by

Salman Rahman 27853815

Name Student ID

“I certify that this submission is my original work and meets the Faculty's Expectations of Originality”

Contents

[Figures 2](#_Toc26108677)

[ Objective 3](#_Toc26108678)

[ Introduction 3](#_Toc26108679)

[ Conceptual Diagram of datapath 5](#_Toc26108680)

[ Conclusion 8](#_Toc26108681)

[ Results 9](#_Toc26108682)

[o ModelSim Simulation of D-Cache: (please zoom in the figures if required) 9](#_Toc26108683)

[o ModelSim Simulation of I-Cache: (please zoom in the figures if required) 12](#_Toc26108684)

[o ModelSim Simulation of Sign\_extend: (please zoom in the figures if required) 13](#_Toc26108685)

[o Precision log file 16](#_Toc26108686)

[o RTL schematic of the Datapath 19](#_Toc26108687)

[o VHDL Code 20](#_Toc26108688)

[ Sign extend component 20](#_Toc26108689)

[ 32 bit – 2 input Mux 21](#_Toc26108690)

[ 5 bit – 2 input Mux 22](#_Toc26108691)

[ d\_cache 23](#_Toc26108692)

[ i\_cache 24](#_Toc26108693)

[ pc\_register 26](#_Toc26108694)

[ datapath 27](#_Toc26108695)

# Figures

[Figure 1 Instructions format of MIPS 4](#_Toc26108672)

[Figure 2 Conceptual diagram of the designed datapath 5](#_Toc26108673)

[Figure 3 Simulation of d-cache unit is shown with different sets of inputs 9](#_Toc26108674)

[Figure 4 Simulation of i-cache unit is shown with different sets of inputs 12](#_Toc26108675)

[Figure 5 RTL schematic of the VHDL code of Datapath 19](#_Toc26108676)

* Objective

The objective of the lab was to design the datapath using all the components created in the previous labs which are the ALU, REGFILE and the NEXT-ADDRESS unit. To complete the data path two more components were required are I-CACHE and D-CACHE

* Introduction

The datapath of the processor handles the execution of the instructions. The datapath designed during the lab consists of the following components:

1. PC
2. I-CACHE
3. reg\_des – Mux
4. REGFILE
5. alu\_src – Mux
6. ALU – Mux
7. D-Cache
8. reg\_in\_src Mux
9. Sign EXTEND block
10. NEXT-ADDRESS

The designed datapath is capable of executing 20 different instructions of three different types as follows:

1. J-Format instruction (format shown in figure below)
   1. Jump – j there
   2. Jump register – jr rs
2. I-Format instruction (format shown in figure below)
   1. Load upper immediate – lui rt, immediate
   2. Set less than immediate – slti rt,rs,immediate
   3. Add immediate – addi rt,rs,immediate
   4. AND immediate – andi rt,rs,immediate
   5. OR immediate – ori rt, rs,immediate
   6. XOR immediate – xori rt,rs,immediate
   7. Load word – lw rt, immediate(rs)
   8. Store word – sw rt, immediate(rs)
   9. Branch less than 0 – bltz rs, there
   10. Branch equal – beq rs, rt, there
   11. Branch not equal 0 – bne rs, rt, there
3. R-Format instruction (format shown in figure below)
   1. Addition– add rd, rs,rt
   2. Subtraction – sub rd, rs, rt
   3. Set less than – slt rd, rs, rt
   4. AND (logical) – and rd, rs, rt
   5. OR (logical) – or rd, rs, rt
   6. XOR (logical) – xor rd, rs, rt
   7. NOR (logical) – nor rd, rs, rt

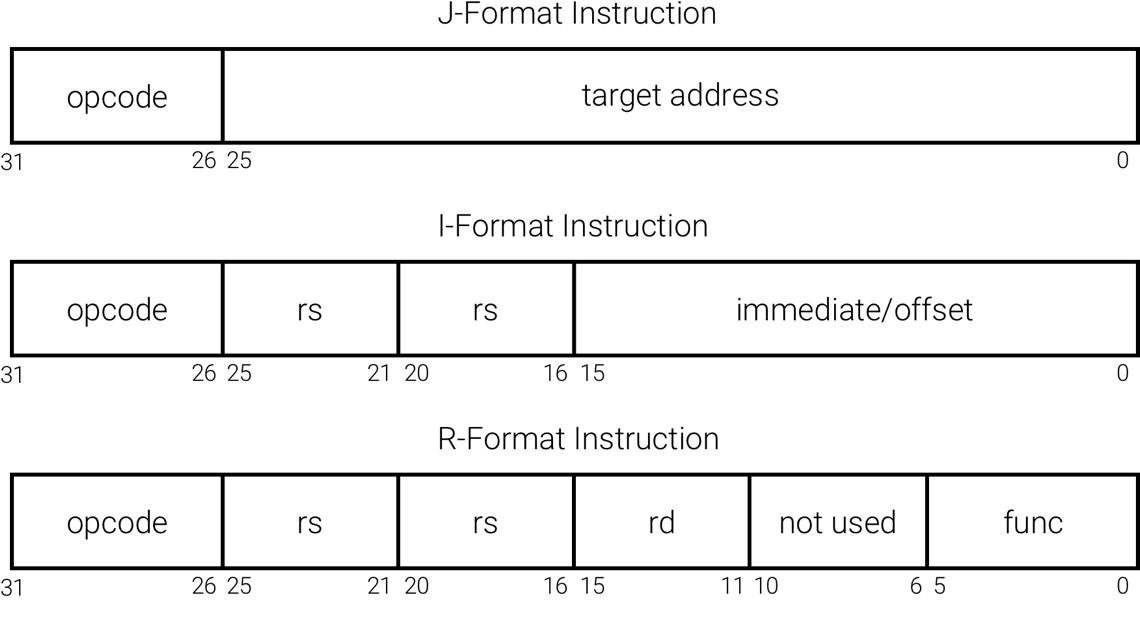


Figure Instructions format of MIPS

* Conceptual Diagram of datapath

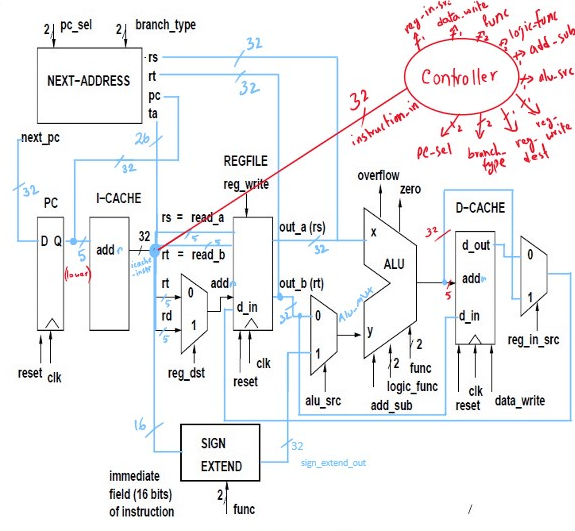


Figure Conceptual diagram of the designed datapath

The conceptual diagram of the data path was created using the diagram provided in the lab manual. The control signals and their bus width are shown in the figure below. The datapath consists of the following components:

1. **NEXT-ADDRESS**

The next address component of the datapath provides the CPU with the next address to be executed with the input values of rs, rt, pc, ta, pc\_sel and branch\_type. The value of the next address is stored in the program counter. The next address implements three different addressing modes: register addressing, PC-relative addressing, and pseudo-direct.

1. **PC**

The program counter can be pictured as a simple D-flip flop, that stores the value received from the next-address components. And, it provides the next address to the rest of the datapath at every clock cycle.

1. **I-CACHE**

Instruction cache holds all the instructions of the program, at specific memory locations which can be pointed by the program counter (PC). For the designed datapath, there are a maximum of 32 locations where the instructions can be stored and retrieved from the I-Cache. The I-Cache provides the datapath with 32-bit instructions based on the memory location accessed by the PC.

1. **REGFILE**

The register file consists of 32 registers of 32 bits. These registers can be accessed by the program being ran and can store data in any of the 32 registers. Similar to the PC, the register file can be pictures as 32 different D-flip flops, which works on rising edge of the clock. Data is written to the register addressed by the reg\_dst MUX, while the reg\_write signal is equal to “1”. The address provided from read\_a and read\_b is used to access the corresponding register in the REGFILE.

1. **Reg\_des MUX**

Selects the address of the register where the data received from either the d-cache or the alu should be written to. The address is selected between the rt or rd found in the instruction.

1. **SIGN EXTEND**

The sign extender extends the 16 bit immediate field for I-format instructions to 32 bit instructions. The type of sign extension depends on the 2 bit func provided to the sign extend block. There are four different types of sign extension that can occur as shown in figure below:

|  |  |  |
| --- | --- | --- |
| func | instruction type | sign extension |
| 00 | load upper immediate |  |
| 01 | set less than immediate |  |
| 10 | arithmetic |  |
| 11 | logical |  |

1. **Alu\_src MUX**

Selects the value between the out\_b of register file or the 32-bit instruction output of the sign extend block.

1. **ALU**

Adds or subtracts the x and y values receives from the register file and the alu\_src mux based on the control signal provided. The functionality of the Alu was described in lab 1 and skipped in this lab report.

1. **D-CACHE**

The D-cache works similar the RAM unit of a CPU. It provides data that was previously stored in it from previous operations. To access the memory locations in the D-Cache, lower order 5-bit of the output from the Alu is provided. The D-Cache is similar to the REGFILE as it contains 32 registers of 32-bit. The registers in the D-cache can be written every rising clock edge while the data\_write signal is set to “1”.

1. **Reg\_in\_src MUX**

This mux selects the value between the ALU output and the D-Cache output to be written to the REGFILE.

The control signals comes from the controller that decides what the signal values should be from the instruction being processes, the controller was not implemented in lab 4.

* Conclusion

In conclusion, the Datapath was successfully designed and synthesized by designing the internal components of the Datapath separately. The simulation of the Datapath was not completed since the controller needs to be implemented and the control signals needs to be sent to the Datapath according to the instruction being executed. A total of ten control signals are required to use the Datapath to successfully execute twenty different instruction of different formats.

* Results
  + ModelSim Simulation of D-Cache: (please zoom in the figures if required)

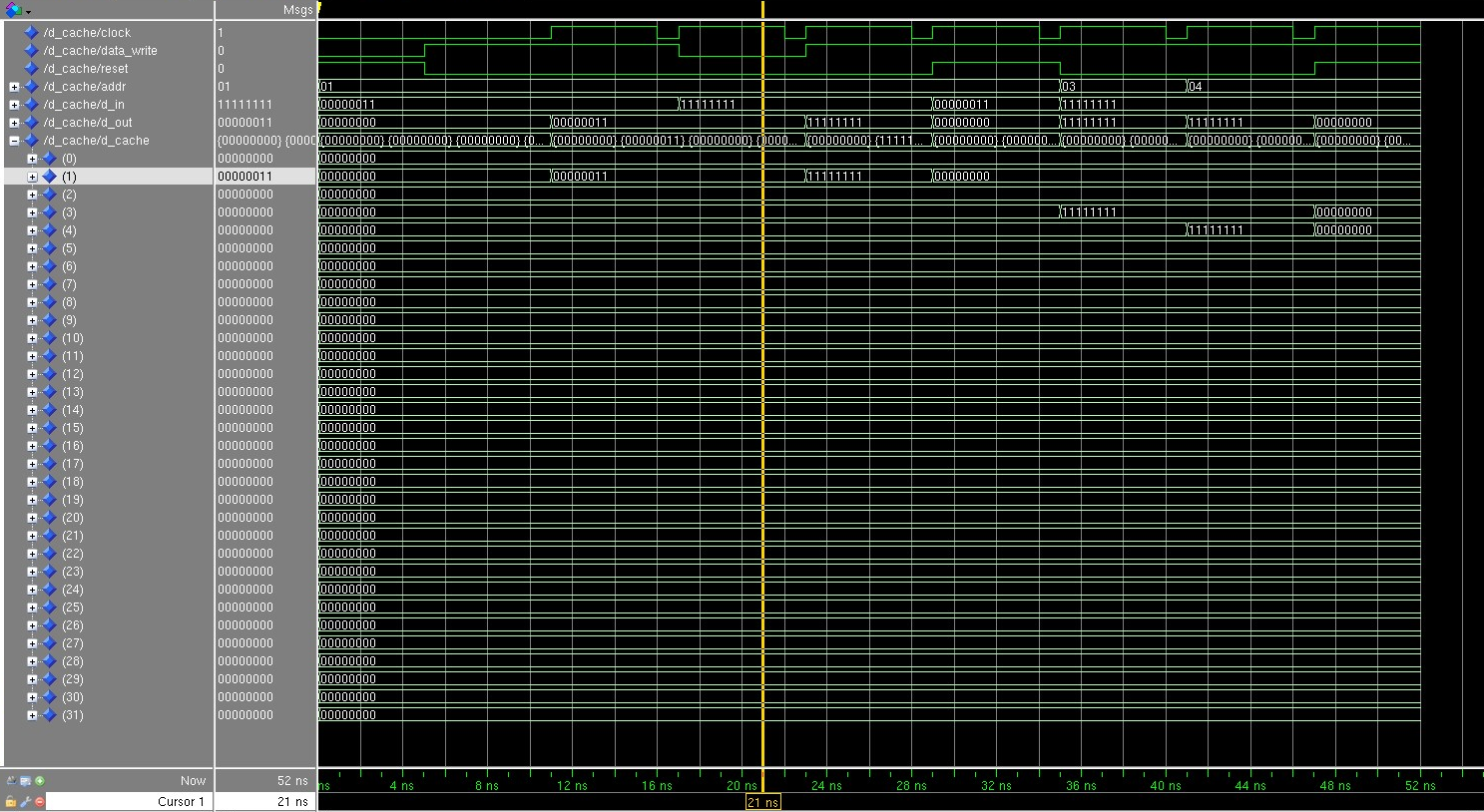


Figure Simulation of d-cache unit is shown with different sets of inputs

The testing of the d-cache was done by saving data to specific addresses in the d-cache, using the data\_write signal set to “1” for the d-cache and at rising clock edge. When the data\_write was set to “0” the register didn’t save the value to the d-cache as seen in the figure above.

**The do file is shown below that was used to test the D-Cache unit:**

restart -f -- to reset values

#restart -f -- to reset values

add wave \*

# test initial reset

force clock 0

force reset 1

force data\_write 0

force addr 00001

force d\_in 16#00000011

run 5

#test clock

force clock 0

force reset 0

force data\_write 1

force addr 00001

force d\_in 16#00000011

run 5

force clock 0

run 1

force clock 1

force reset 0

force data\_write 1

force addr 00001

force d\_in 16#00000011

run 5

# test data write

force clock 0

run 1

force clock 1

force reset 0

force data\_write 0

force addr 00001

force d\_in 16#11111111

run 5

force clock 0

run 1

force clock 1

force reset 0

force data\_write 1

force addr 00001

force d\_in 16#11111111

run 5

#test sudden reset again

force clock 0

run 1

force clock 1

force reset 1

force data\_write 1

force addr 00001

force d\_in 16#00000011

run 5

# test address

force clock 0

run 1

force clock 1

force reset 0

force data\_write 1

force addr 00011

force d\_in 16#11111111

run 5

force clock 0

run 1

force clock 1

force reset 0

force data\_write 1

force addr 00100

force d\_in 16#11111111

run 5

# test all resets again

force clock 0

run 1

force clock 1

force reset 1

force data\_write 1

force addr 00100

force d\_in 16#11111111

run 5

* + ModelSim Simulation of I-Cache: (please zoom in the figures if required)

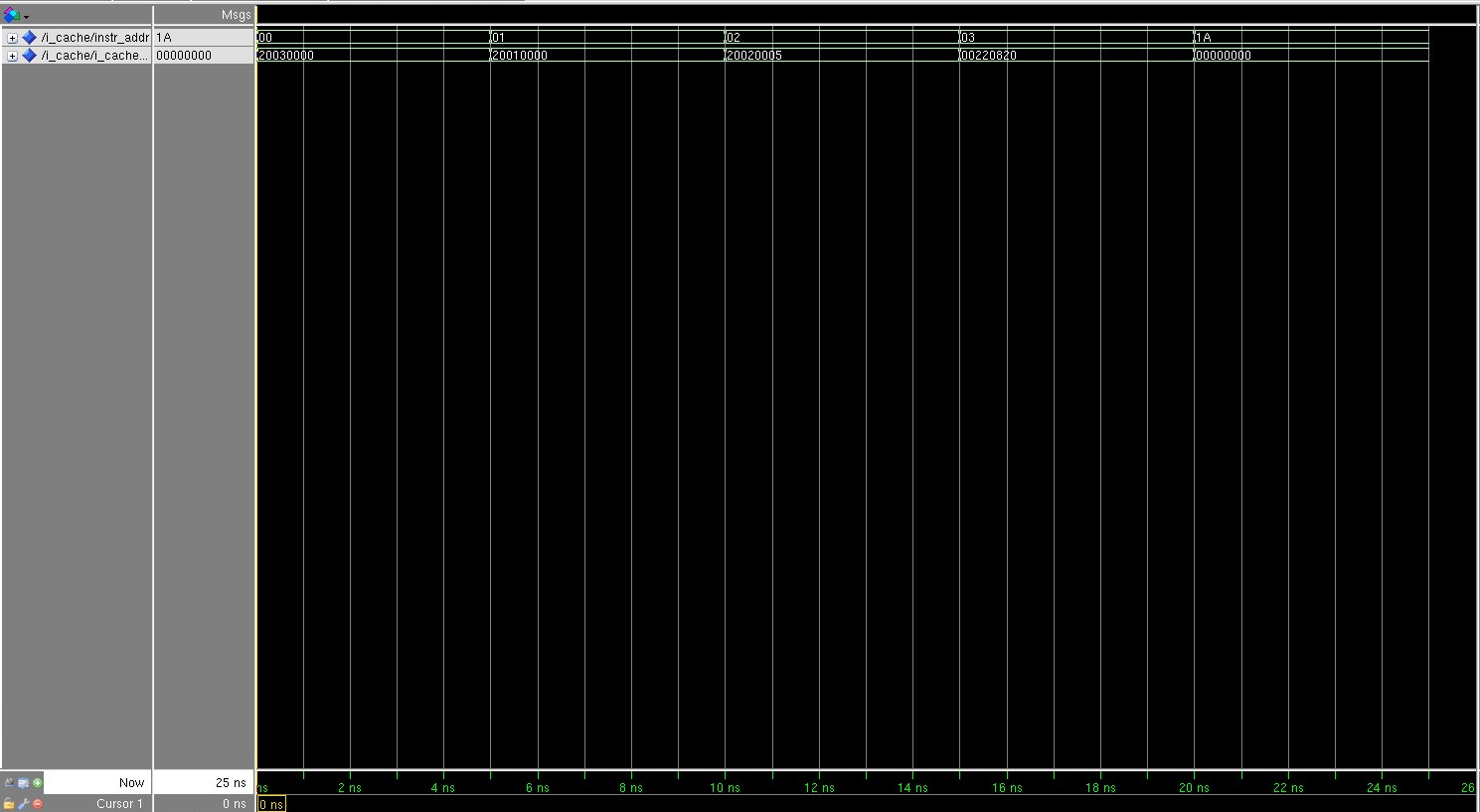


Figure Simulation of i-cache unit is shown with different sets of inputs

Some of the instructions were had coded in the i-cache and then the pc signal was forced from the do file to test some of the outputs from the i-cache, after testing four different addresses in the i-cache it was confirmed that the i-cache component created was working as expected.

**DO file for the i-cache testing**

restart -f -- to reset values

add wave \*

force instr\_addr 00000

run 5

force instr\_addr 00001

run 5

force instr\_addr 00010

run 5

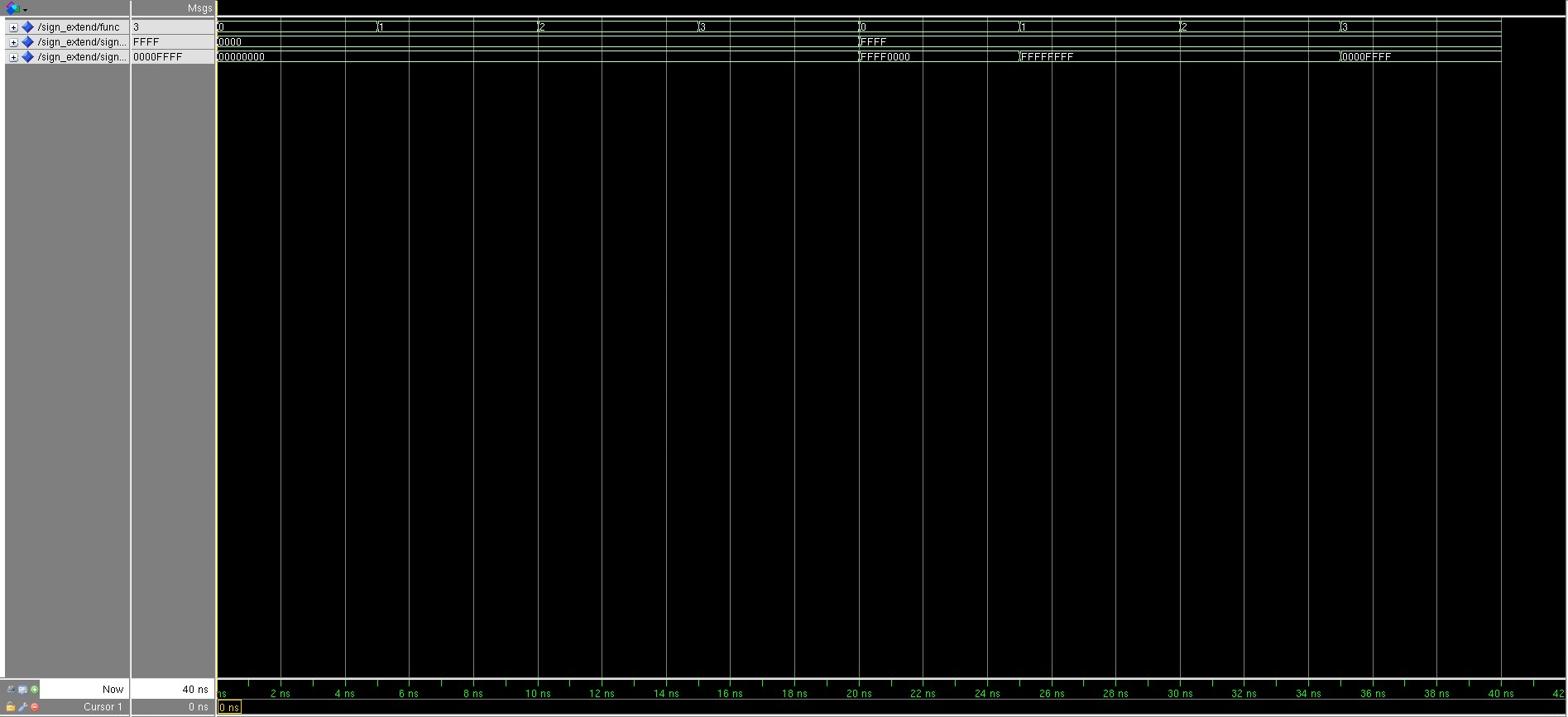
force instr\_addr 00011

run 5

force instr\_addr 11010

run 5

* + ModelSim Simulation of Sign\_extend: (please zoom in the figures if required)



The sign extend was simulated by providing 16-bit to the input of the sign extend component and the four different function code was provided as the table shown earlier in the report for the sign extend block. The output of the sign extend block shown in the figure above was checked against the table of expected outputs to verify the sign extend component is working properly.

**Do file for the sign extend block**

restart -f -- to reset values

add wave \*

force sign\_extend\_in 16#0000

force func 00

run 5

force sign\_extend\_in 16#0000

force func 01

run 5

force sign\_extend\_in 16#0000

force func 10

run 5

force sign\_extend\_in 16#0000

force func 11

run 5

force sign\_extend\_in 16#FFFF

force func 00

run 5

force sign\_extend\_in 16#FFFF

force func 01

run 5

force sign\_extend\_in 16#FFFF

force func 10

run 5

force sign\_extend\_in 16#FFFF

force func 11

run 5

* + Precision log file

# Info: [9566]: Logging session transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/precision.log

// Precision RTL Synthesis 64-bit 2016.1.0.15 (Production Release) Wed Jun 8 09:35:56 PDT 2016

//

// Copyright (c) Mentor Graphics Corporation, 1996-2016, All Rights Reserved.

// Portions copyright 1991-2008 Compuware Corporation

// UNPUBLISHED, LICENSED SOFTWARE.

// CONFIDENTIAL AND PROPRIETARY INFORMATION WHICH IS THE

// PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS

//

// Running on Linux sal\_rahm@flying-dragon.encs.concordia.ca #1 SMP Thu Nov 14 10:04:03 CST 2019 3.10.0-1062.4.3.el7.x86\_64 x86\_64

//

// Start time Sat Nov 30 21:27:36 2019

# -------------------------------------------------

# Info: [9566]: Logging session transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/precision.log

# COMMAND: new\_project -name lab4 -folder /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4 -createimpl\_name lab4\_impl\_1

# Info: [9574]: Input directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4

# Info: [9569]: Moving session transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/precision.log

# Info: [9555]: Created project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp in folder /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4.

# Info: [9531]: Created directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1.

# Info: [9554]: Created implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

# Info: [9575]: The Results Directory has been set to: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1/

# Info: [9566]: Logging project transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1/precision.log

# Info: [9566]: Logging suppressed messages transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1/precision.log.suppressed

# Info: [9550]: Activated implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

new\_project -name lab4 -folder /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4 -createimpl\_name lab4\_impl\_1

# COMMAND: add\_input\_file {../../32-bit-CPU/datapath.vhdl}

add\_input\_file {../../32-bit-CPU/datapath.vhdl}

# COMMAND: add\_input\_file {../../32-bit-CPU/32-bit-register.vhd ../../32-bit-CPU/alu.vhd ../../32-bit-CPU/d\_cache.vhdl ../../32-bit-CPU/datapath.vhdl ../../32-bit-CPU/i\_cache.vhdl ../../32-bit-CPU/mux.vhdl ../../32-bit-CPU/mux\_5\_bit.vhdl ../../32-bit-CPU/next\_address.vhd ../../32-bit-CPU/pc\_register.vhdl ../../32-bit-CPU/sign\_extend.vhdl}

# Warning: [15238]: Input file /nfs/home/s/sal\_rahm/316/32-bit-CPU/datapath.vhdl already exists in the list. Command options ignored.

add\_input\_file {../../32-bit-CPU/32-bit-register.vhd ../../32-bit-CPU/alu.vhd ../../32-bit-CPU/d\_cache.vhdl ../../32-bit-CPU/datapath.vhdl ../../32-bit-CPU/i\_cache.vhdl ../../32-bit-CPU/mux.vhdl ../../32-bit-CPU/mux\_5\_bit.vhdl ../../32-bit-CPU/next\_address.vhd ../../32-bit-CPU/pc\_register.vhdl ../../32-bit-CPU/sign\_extend.vhdl}

# COMMAND: setup\_design -manufacturer Xilinx -family "VIRTEX-II Pro" -part 2VP30ff896 -speed -7

# Info: [15298]: Setting up the design to use synthesis library "xcv2p.syn"

# Info: [575]: The global max fanout is currently set to 10000 for Xilinx - VIRTEX-II Pro.

# Info: [15324]: Setting Part to: "2VP30ff896".

# Info: [15325]: Setting Process to: "7".

# Info: [7512]: The place and route tool for current technology is ISE.

setup\_design -manufacturer Xilinx -family "VIRTEX-II Pro" -part 2VP30ff896 -speed -7

# COMMAND: setup\_design -frequency 100 -max\_fanout=10000

# Info: [575]: The global max fanout is currently set to 10000 for Xilinx - VIRTEX-II Pro.

setup\_design -frequency 100 -max\_fanout=10000

# COMMAND: compile

# Info: [3022]: Reading file: /CMC/tools/mentor/precision/Mgc\_home/pkgs/psr/techlibs/xcv2p.syn.

# Info: [634]: Loading library initialization file /CMC/tools/mentor/precision/Mgc\_home/pkgs/psr/userware/xilinx\_rename.tcl

# Info: XILINX

# Info: [40000]: vhdlorder, Release 2016a.7

# Info: [40000]: Files sorted successfully.

# Info: [40000]: hdl-analyze, Release RTLC-Precision 2016a.7

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/datapath.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/32-bit-register.vhd" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/alu.vhd" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/d\_cache.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/i\_cache.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/mux.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/mux\_5\_bit.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/next\_address.vhd" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/pc\_register.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/../../32-bit-CPU/sign\_extend.vhdl" ...

# Info: [659]: Top module of the design is set to: datapath.

# Info: [657]: Current working directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1.

# Info: [40000]: RTLC-Driver, Release RTLC-Precision 2016a.7

# Info: [40000]: Last compiled on Jun 2 2016 06:11:46

# Info: [44512]: Initializing...

# Info: [44504]: Partitioning design ....

# Info: [40000]: RTLCompiler, Release RTLC-Precision 2016a.7

# Info: [40000]: Last compiled on Jun 2 2016 06:47:43

# Info: [44512]: Initializing...

# Info: [44522]: Root Module work.datapath(datapath\_arch): Pre-processing...

# Info: [44506]: Module work.pc\_register(pc\_register\_arch): Pre-processing...

# Info: [44506]: Module work.i\_cache(i\_cache\_arch): Pre-processing...

# Info: [44506]: Module work.next\_address(next\_address\_arch): Pre-processing...

# Info: [44506]: Module work.two\_input\_mux\_5\_bit(two\_input\_5\_bit\_mux\_arch): Pre-processing...

# Info: [44506]: Module work.regfile(register\_file\_arch): Pre-processing...

# Info: [45251]: Built-in hardware memory core inferred for variable ': regfile.reg\_arr depth = 32, width = 32'.

# Info: [44506]: Module work.sign\_extend(sign\_extend\_arch): Pre-processing...

# Info: [44506]: Module work.two\_input\_mux(two\_input\_mux\_arch): Pre-processing...

# Info: [44506]: Module work.alu(alu\_architecture): Pre-processing...

# Info: [44506]: Module work.d\_cache(d\_cache\_arch): Pre-processing...

# Info: [45251]: Built-in hardware memory core inferred for variable ': d\_cache.d\_cache depth = 32, width = 32'.

# Info: [44508]: Module work.pc\_register(pc\_register\_arch): Compiling...

# Info: [44508]: Module work.i\_cache(i\_cache\_arch): Compiling...

# Info: [44508]: Module work.next\_address(next\_address\_arch): Compiling...

# Info: [44508]: Module work.two\_input\_mux\_5\_bit(two\_input\_5\_bit\_mux\_arch): Compiling...

# Info: [44508]: Module work.regfile(register\_file\_arch): Compiling...

# Info: [44508]: Module work.sign\_extend(sign\_extend\_arch): Compiling...

# Info: [44508]: Module work.two\_input\_mux(two\_input\_mux\_arch): Compiling...

# Info: [44508]: Module work.alu(alu\_architecture): Compiling...

# Info: [44508]: Module work.d\_cache(d\_cache\_arch): Compiling...

# Info: [44523]: Root Module work.datapath(datapath\_arch): Compiling...

# Info: [44846]: Rebalanced Expression Tree...

# Info: [44842]: Compilation successfully completed.

# Info: [44841]: Counter Inferencing === Detected : 1, Inferred (Modgen/Selcounter/AddSub) : 0 (0 / 0 / 0), AcrossDH (Merged/Not-Merged) : (0 / 0), Not-Inferred (Acrossdh/Attempted) : (0 / 0), Local Vars : 1 ===

# Info: [44856]: Total lines of RTL compiled: 779.

# Info: [44835]: Total CPU time for compilation: 0.0 secs.

# Info: [44513]: Overall running time for compilation: 7.0 secs.

# Info: [657]: Current working directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1.

# Info: [15330]: Doing rtl optimizations.

# Info: [660]: Finished compiling design.

compile

# COMMAND: synthesize

# Info: [657]: Current working directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1.

# Info: [4556]: 5 Instances are flattened in hierarchical block .work.datapath.datapath\_arch.

# Info: [20013]: Precision will use 9 processor(s).

# Info: # [15002]: Optimizing design view:.work.regfile.register\_file\_arch\_unfold\_2174

# Info: # [15002]: Optimizing design view:.work.d\_cache.d\_cache\_arch

# Info: # [15002]: Optimizing design view:.work.next\_address.next\_address\_arch\_unfold\_2623

# Info: # [15002]: Optimizing design view:.work.alu.alu\_architecture

# Info: # [15002]: Optimizing design view:.work.datapath.datapath\_arch

# Info: [12035]: -- Running timing characterization...

# Info: # [15002]: Optimizing design view:.work.d\_cache.d\_cache\_arch

# Info: # [15002]: Optimizing design view:.work.next\_address.next\_address\_arch\_unfold\_2623

# Info: # [15002]: Optimizing design view:.work.alu.alu\_architecture

# Info: # [15002]: Optimizing design view:.work.datapath.datapath\_arch

# Info: # [15002]: Optimizing design view:.work.regfile.register\_file\_arch\_unfold\_2174

# Info: [8048]: Added global buffer BUFGP for Port port:clk

# Info: [3027]: Writing file: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1/datapath.edf.

# Info: [3027]: Writing file: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4\_impl\_1/datapath.ucf.

# Info: [12045]: Starting timing reports generation...

# Info: [12046]: Timing reports generation done.

# Info: [12048]: POST-SYNTHESIS TIMING REPORTS ARE ESTIMATES AND SHOULD NOT BE RELIED ON TO MAKE QoR DECISIONS. For accurate timing information, please run place-and-route (P&R) and review P&R generated timing reports.

# Info: [660]: Finished synthesizing design.

# Info: [11019]: Total CPU time for synthesis: 3.7 s secs.

# Info: [11020]: Overall running time for synthesis: 6.6 s secs.

synthesize

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: save\_project

# Info: [9562]: Saved implementation lab4\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

save\_project

# COMMAND: exit -force

# Info: [9530]: Closed project: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab4/lab4.psp.

close\_project -discard

exit -force

* + RTL schematic of the Datapath

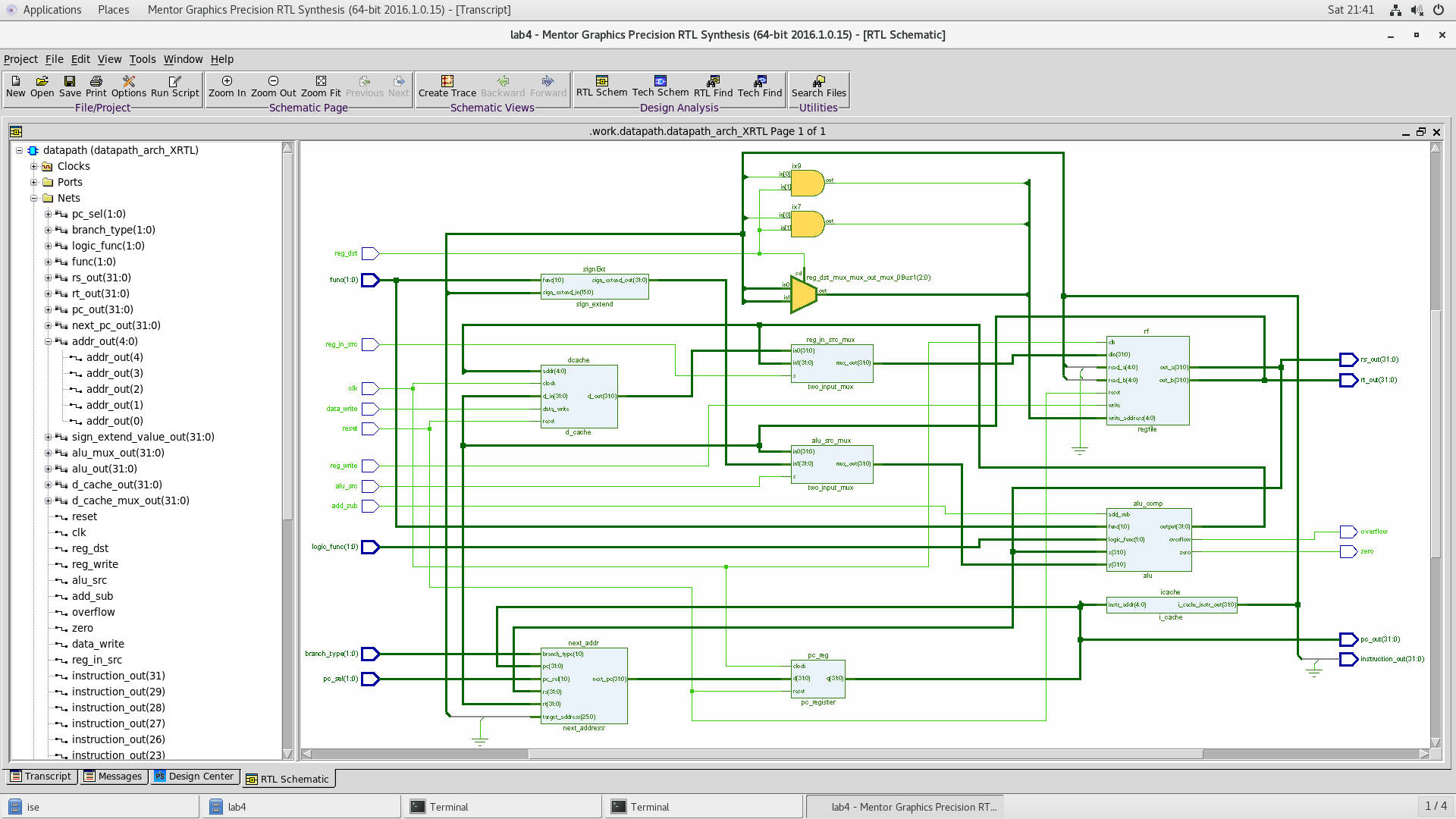


Figure RTL schematic of the VHDL code of Datapath

The RTL schematic shows the synthesis of the VHDL components created for the datapath. All the components were individually created for the datapath and port mapped together in the datapath as separate components which is reflected in the RTL schematic. All the separate components are shown as boxes which resembles the conceptual diagram provided earlier in the report. The datapath will be completely function with the controller that will be created in the next lab.

* + VHDL Code
    - Sign extend component
* -- salman rahman
* -- 27853815
* LIBRARY IEEE;
* USE IEEE.std\_logic\_1164.ALL;
* USE IEEE.std\_logic\_unsigned.ALL;
* ENTITY sign\_extend IS
* PORT (
* func : IN std\_logic\_vector(1 downto 0);
* sign\_extend\_in : IN std\_logic\_vector(15 DOWNTO 0);
* sign\_extend\_out : OUT std\_logic\_vector(31 DOWNTO 0)
* );
* END sign\_extend;
* ARCHITECTURE sign\_extend\_arch OF sign\_extend IS
* BEGIN
* process(func, sign\_extend\_in)
* variable sign\_extend\_out\_var: std\_logic\_vector(sign\_extend\_out'length-1 downto sign\_extend\_out'right);
* begin
* CASE func IS
* WHEN "00" =>
* sign\_extend\_out <= sign\_extend\_in & "0000000000000000";
* WHEN "01" =>
* -- if sign\_extend\_in(sign\_extend\_in'length-1) = '1' then
* if sign\_extend\_in(sign\_extend\_in'length-1) = '1' then
* sign\_extend\_out\_var := (others => '1');
* sign\_extend\_out\_var(sign\_extend\_in'length-1 downto sign\_extend\_in'right) := sign\_extend\_in;
* sign\_extend\_out <= sign\_extend\_out\_var;
* else
* sign\_extend\_out\_var := (others => '0');
* sign\_extend\_out\_var(sign\_extend\_in'length-1 downto sign\_extend\_in'right) := sign\_extend\_in;
* sign\_extend\_out <= sign\_extend\_out\_var;
* end if;
* WHEN "10" =>
* if sign\_extend\_in(sign\_extend\_in'length-1) = '1' then
* sign\_extend\_out\_var := (others => '1');
* sign\_extend\_out\_var(sign\_extend\_in'length-1 downto sign\_extend\_in'right) := sign\_extend\_in;
* sign\_extend\_out <= sign\_extend\_out\_var;
* else
* sign\_extend\_out\_var := (others => '0');
* sign\_extend\_out\_var(sign\_extend\_in'length-1 downto sign\_extend\_in'right) := sign\_extend\_in;
* sign\_extend\_out <= sign\_extend\_out\_var;
* end if;
* WHEN "11" =>
* -- if sign\_extend\_in(sign\_extend\_in'length-1) = '1' then
* sign\_extend\_out\_var := (others => '0');
* sign\_extend\_out\_var(sign\_extend\_in'length-1 downto sign\_extend\_in'right) := sign\_extend\_in;
* sign\_extend\_out <= sign\_extend\_out\_var;
* -- else
* --  sign\_extend\_out\_var := (others => '0');
* --  sign\_extend\_out\_var(sign\_extend\_in'length-1 downto sign\_extend\_in'right) := sign\_extend\_in;
* --  sign\_extend\_out <= sign\_extend\_out\_var;
* -- end if;
* WHEN OTHERS =>
* sign\_extend\_out <= (others=>'0');
* END CASE;
* end process;
* END sign\_extend\_arch;
  + - 32 bit – 2 input Mux

-- salman rahman

-- 27853815

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

USE IEEE.std\_logic\_unsigned.ALL;

ENTITY two\_input\_mux IS

    PORT (

        s : IN std\_logic;

        in0 : IN std\_logic\_vector(31 DOWNTO 0);

        in1 : IN std\_logic\_vector(31 DOWNTO 0);

        mux\_out : OUT std\_logic\_vector(31 DOWNTO 0)

    );

END two\_input\_mux;

ARCHITECTURE two\_input\_mux\_arch OF two\_input\_mux IS

BEGIN

process(s, in0, in1)

  begin

    CASE s IS

        WHEN '0' =>

            mux\_out <= in0;

        WHEN '1' =>

            mux\_out <= in1;

        WHEN OTHERS =>

            mux\_out <= (others=>'0');

    END CASE;

end process;

END two\_input\_mux\_arch;

* + - 5 bit – 2 input Mux

-- salman rahman

-- 27853815

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

USE IEEE.std\_logic\_unsigned.ALL;

ENTITY two\_input\_mux\_5\_bit IS

    PORT (

        s : IN std\_logic;

        in0 : IN std\_logic\_vector(4 DOWNTO 0);

        in1 : IN std\_logic\_vector(4 DOWNTO 0);

        mux\_out : OUT std\_logic\_vector(4 DOWNTO 0)

    );

END two\_input\_mux\_5\_bit;

ARCHITECTURE two\_input\_5\_bit\_mux\_arch OF two\_input\_mux\_5\_bit IS

BEGIN

process(s, in0, in1)

  begin

    CASE s IS

        WHEN '0' =>

            mux\_out <= in0;

        WHEN '1' =>

            mux\_out <= in1;

        WHEN OTHERS =>

            mux\_out <= (others=>'0');

    END CASE;

end process;

END two\_input\_5\_bit\_mux\_arch;

* + - d\_cache

-- salman rahman

-- 27853815

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

USE IEEE.std\_logic\_unsigned.ALL;

use ieee.numeric\_std.all;

ENTITY d\_cache IS

    PORT (

        clock : IN std\_logic;

        data\_write : IN std\_logic;

        reset : IN std\_logic;

        addr : IN std\_logic\_vector(4 DOWNTO 0);

        d\_in : IN std\_logic\_vector(31 DOWNTO 0);

        d\_out : OUT std\_logic\_vector(31 DOWNTO 0)

    );

END d\_cache;

ARCHITECTURE d\_cache\_arch OF d\_cache IS

    TYPE data\_cache IS ARRAY(0 TO 31) OF std\_logic\_vector(d\_in'length - 1 DOWNTO d\_in'right);

    SIGNAL d\_cache : data\_cache;

BEGIN

    PROCESS (clock, data\_write, reset, addr, d\_in)

    BEGIN

        IF reset = '1' THEN

            -- FOR i IN 0 TO 31 LOOP

            --     d\_cache(i) <= (OTHERS => '0');

            -- END LOOP;

            d\_cache <= (others => (others => '0'));

        ELSIF (clock = '1' AND clock'event AND data\_write = '1') THEN

            -- IF data\_write = '1' THEN

            -- d\_cache(to\_integer(unsigned(addr))) <= d\_in;

            d\_cache(CONV\_INTEGER(addr)) <= d\_in;

            -- END IF;

        END IF;

    END PROCESS;

    d\_out <= d\_cache(CONV\_INTEGER(addr));

END d\_cache\_arch;

* + - i\_cache

-- salman rahman

-- 27853815

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

USE IEEE.std\_logic\_unsigned.ALL;

ENTITY i\_cache IS

    PORT (

        instr\_addr : IN std\_logic\_vector(4 DOWNTO 0);

        i\_cache\_instr\_out : OUT std\_logic\_vector(31 DOWNTO 0)

    );

END i\_cache;

ARCHITECTURE i\_cache\_arch OF i\_cache IS

BEGIN

    -- eg. program

    -- 00000 addi r1, r0, 1 ; r1 = r0 + 1 = 0 + 1

    -- 00001 addi r2, r0, 2 ; r2 = r0 + 2 = 0 + 2

    -- 00010 there: add r2, r2, r1 ; r2 = r2 + r1 = r2 + 1

    -- 00011 j there ; goto label there

process(instr\_addr)

  begin

    CASE instr\_addr IS

            -- addi r1, r0, 1

        WHEN "00000" =>

            i\_cache\_instr\_out <= "00100000000000110000000000000000";

            --addi r2, r0, 2

        WHEN "00001" =>

            i\_cache\_instr\_out <= "00100000000000010000000000000000";

            -- add r2, r2, r1

        WHEN "00010" =>

            i\_cache\_instr\_out <= "00100000000000100000000000000101";

            -- jump 00010

        WHEN "00011" =>

            i\_cache\_instr\_out <= "00000000001000100000100000100000";

            -- do not care

            WHEN "00100" =>

            i\_cache\_instr\_out <= "00100000010000101111111111111111";

            -- do not care

            WHEN "00101" =>

            i\_cache\_instr\_out <= "00010000010000110000000000000001";

            WHEN "00110" =>

            i\_cache\_instr\_out <= "00001000000000000000000000000011";

            WHEN "00111" =>

            i\_cache\_instr\_out <= "10101100000000010000000000000000";

            WHEN "01000" =>

            i\_cache\_instr\_out <= "10001100000001000000000000000000";

            WHEN "01001" =>

            i\_cache\_instr\_out <= "00110000100001000000000000001010";

            WHEN "01010" =>

            i\_cache\_instr\_out <= "00110100100001000000000000000001";

            WHEN "01011" =>

            i\_cache\_instr\_out <= "00111000100001000000000000001011";

            WHEN "01100" =>

            i\_cache\_instr\_out <= "00111000100001000000000000000000";

            -- do not care

        WHEN OTHERS =>

            i\_cache\_instr\_out <= (OTHERS => '0');

    END CASE;

end process;

END i\_cache\_arch;

* + - pc\_register

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE work.ALL;

ENTITY pc\_register IS

    PORT (

        reset : IN std\_logic;

        clock : IN std\_logic;

        d : IN std\_logic\_vector(31 DOWNTO 0);

        q : OUT std\_logic\_vector(31 DOWNTO 0)

    );

END pc\_register;

ARCHITECTURE pc\_register\_arch OF pc\_register IS

BEGIN

    PROCESS (clock, reset, d)

      begin

        IF reset = '1' THEN

            q <= (OTHERS => '0');

        ELSIF (clock = '1' AND clock'event) THEN

            q <= d;

        END IF;

    END PROCESS;

END pc\_register\_arch;

* + - datapath

-- salman rahman

-- 27853815

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY datapath IS

    PORT (

        -- global signals

        reset : IN std\_logic;

        clk : IN std\_logic;

        -- next address ports

        pc\_sel : IN std\_logic\_vector(1 DOWNTO 0);

        branch\_type : IN std\_logic\_vector(1 DOWNTO 0);

        -- reg\_des MUX ports

        reg\_dst : IN std\_logic;

        -- sign extend ports

        -- func : IN std\_logic\_vector(1 DOWNTO 0);

        -- regfile ports block

        reg\_write : IN std\_logic;

        -- alu\_src MUX ports

        alu\_src : IN std\_logic;

        -- alu ports

        add\_sub : IN std\_logic;

        logic\_func : IN std\_logic\_vector(1 DOWNTO 0);

        func : IN std\_logic\_vector(1 DOWNTO 0);

        overflow : OUT std\_logic;

        zero : OUT std\_logic;

        -- d-cache ports

        data\_write : IN std\_logic;

        -- reg\_in\_src MUX port

        reg\_in\_src : IN std\_logic;

        -- control port

        instruction\_out : OUT std\_logic\_vector(31 DOWNTO 0);

        rs\_out : OUT std\_logic\_vector(31 DOWNTO 0);

        rt\_out : OUT std\_logic\_vector(31 DOWNTO 0);

        pc\_out : OUT std\_logic\_vector(31 DOWNTO 0)

    );

END datapath;

ARCHITECTURE datapath\_arch OF datapath IS

    COMPONENT pc\_register

        PORT (

            reset : IN std\_logic := '0';

            clock : IN std\_logic := '0';

            d : IN std\_logic\_vector(31 DOWNTO 0) := (OTHERS => '0');

            q : OUT std\_logic\_vector(31 DOWNTO 0)

        );

    END COMPONENT;

    COMPONENT i\_cache

        PORT (

            instr\_addr : IN std\_logic\_vector(4 DOWNTO 0) := (OTHERS => '0');

            i\_cache\_instr\_out : OUT std\_logic\_vector(31 DOWNTO 0)

        );

    END COMPONENT;

    COMPONENT next\_address

        PORT (

            rt, rs : IN std\_logic\_vector(31 DOWNTO 0) := (OTHERS => '0');

            pc : IN std\_logic\_vector(31 DOWNTO 0) := (OTHERS => '0');

            target\_address : IN std\_logic\_vector(25 DOWNTO 0) := (OTHERS => '0');

            branch\_type : IN std\_logic\_vector(1 DOWNTO 0) := (OTHERS => '0');

            pc\_sel : IN std\_logic\_vector(1 DOWNTO 0) := (OTHERS => '0');

            next\_pc : OUT std\_logic\_vector(31 DOWNTO 0)

        );

    END COMPONENT;

    COMPONENT two\_input\_mux IS

        PORT (

            s : IN std\_logic;

            in0 : IN std\_logic\_vector(31 DOWNTO 0);

            in1 : IN std\_logic\_vector(31 DOWNTO 0);

            mux\_out : OUT std\_logic\_vector(31 DOWNTO 0)

        );

    END COMPONENT;

    COMPONENT two\_input\_mux\_5\_bit IS

        PORT (

            s : IN std\_logic;

            in0 : IN std\_logic\_vector(4 DOWNTO 0);

            in1 : IN std\_logic\_vector(4 DOWNTO 0);

            mux\_out : OUT std\_logic\_vector(4 DOWNTO 0)

        );

    END COMPONENT;

    COMPONENT regfile

        PORT (

            din : IN std\_logic\_vector(31 DOWNTO 0) := (OTHERS => '0');

            reset : IN std\_logic := '0';

            clk : IN std\_logic := '0';

            write : IN std\_logic := '0';

            read\_a : IN std\_logic\_vector(4 DOWNTO 0) := (OTHERS => '0');

            read\_b : IN std\_logic\_vector(4 DOWNTO 0) := (OTHERS => '0');

            write\_address : IN std\_logic\_vector(4 DOWNTO 0) := (OTHERS => '0');

            out\_a : OUT std\_logic\_vector(31 DOWNTO 0);

            out\_b : OUT std\_logic\_vector(31 DOWNTO 0)

        );

    END COMPONENT;

    COMPONENT alu

        PORT (

            -- two input operands x and y both 32-bits

            x, y : IN std\_logic\_vector(31 DOWNTO 0) := (OTHERS => '0');

            -- 0 = add, 1 = sub

            add\_sub : IN std\_logic := '0';

            -- 00 = AND, 01 = OR, 10 = XOR, 11 = NOR

            logic\_func : IN std\_logic\_vector(1 DOWNTO 0) := (OTHERS => '0');

            -- 00 = lui, 01 = setlessthan0, 10 = arith, 11 = logic

            func : IN std\_logic\_vector(1 DOWNTO 0) := (OTHERS => '0');

            output : OUT std\_logic\_vector(31 DOWNTO 0);

            overflow : OUT std\_logic;

            zero : OUT std\_logic

        );

    END COMPONENT;

    COMPONENT d\_cache

        PORT (

            clock : IN std\_logic := '0';

            data\_write : IN std\_logic := '0';

            reset : IN std\_logic := '0';

            addr : IN std\_logic\_vector(4 DOWNTO 0) := (OTHERS => '0');

            d\_in : IN std\_logic\_vector(31 DOWNTO 0) := (OTHERS => '0');

            d\_out : OUT std\_logic\_vector(31 DOWNTO 0)

        );

    END COMPONENT;

    COMPONENT sign\_extend

        PORT (

            func : IN std\_logic\_vector(1 DOWNTO 0) := (OTHERS => '0');

            sign\_extend\_in : IN std\_logic\_vector(15 DOWNTO 0) := (OTHERS => '0');

            sign\_extend\_out : OUT std\_logic\_vector(31 DOWNTO 0)

        );

    END COMPONENT;

    -- internal component output signal declarations

    -- internal signals for pc register

    SIGNAL q\_out : std\_logic\_vector(31 DOWNTO 0);

    -- internal signals for next\_address

    SIGNAL next\_pc\_out : std\_logic\_vector(31 DOWNTO 0);

    -- internal signals for i\_cache

    SIGNAL instruction\_cache\_out : std\_logic\_vector(31 DOWNTO 0);

    -- internal signal reg\_dst mux

    SIGNAL addr\_out : std\_logic\_vector(4 DOWNTO 0);

    -- internal signal sign\_extend

    SIGNAL sign\_extend\_value\_out : std\_logic\_vector(31 DOWNTO 0);

    -- internal signals register file

    SIGNAL rs\_data\_out : std\_logic\_vector(31 DOWNTO 0);

    SIGNAL rt\_data\_out : std\_logic\_vector(31 DOWNTO 0);

    -- rs register bits - (25 downto 21)

    -- rt register bits - (20 downto 16)

    -- rd register bits - (15 downto 11)

    -- internal signal alu\_src mux

    SIGNAL alu\_mux\_out : std\_logic\_vector(31 DOWNTO 0);

    -- alu internal signals

    SIGNAL alu\_out : std\_logic\_vector(31 DOWNTO 0);

    -- d cache signals

    SIGNAL d\_cache\_out : std\_logic\_vector(31 DOWNTO 0);

    -- internal signal reg\_in\_src mux

    SIGNAL d\_cache\_mux\_out : std\_logic\_vector(31 DOWNTO 0);

BEGIN

    pc\_reg : pc\_register PORT MAP(

        reset => reset,

        clock => clk,

        d => next\_pc\_out,

        q => q\_out

    );

    icache : i\_cache PORT MAP(

        instr\_addr => q\_out(4 DOWNTO 0),

        i\_cache\_instr\_out => instruction\_cache\_out

    );

    next\_addr : next\_address PORT MAP(

        rt => rt\_data\_out,

        rs => rs\_data\_out,

        pc => q\_out,

        target\_address => instruction\_cache\_out(25 DOWNTO 0),

        branch\_type => branch\_type,

        pc\_sel => pc\_sel,

        next\_pc => next\_pc\_out

    );

    reg\_dst\_mux : two\_input\_mux\_5\_bit PORT MAP(

        s => reg\_dst,

        in0 => instruction\_cache\_out(20 DOWNTO 16),

        in1 => instruction\_cache\_out(15 DOWNTO 11),

        mux\_out => addr\_out

    );

    rf : regfile PORT MAP(

        din => d\_cache\_mux\_out,

        reset => reset,

        clk => clk,

        write => reg\_write,

        read\_a => instruction\_cache\_out(25 DOWNTO 21),

        read\_b => instruction\_cache\_out(20 DOWNTO 16),

        write\_address => addr\_out,

        out\_a => rs\_data\_out,

        out\_b => rt\_data\_out

    );

    signExt : sign\_extend PORT MAP(

        func => func,

        sign\_extend\_in => instruction\_cache\_out(15 DOWNTO 0),

        sign\_extend\_out => sign\_extend\_value\_out

    );

    alu\_src\_mux : two\_input\_mux PORT MAP(

        s => alu\_src,

        in0 => rt\_data\_out,

        in1 => sign\_extend\_value\_out,

        mux\_out => alu\_mux\_out

    );

    alu\_comp : alu PORT MAP(

        x => rs\_data\_out,

        y => alu\_mux\_out,

        add\_sub => add\_sub,

        logic\_func => logic\_func,

        func => func,

        output => alu\_out,

        overflow => overflow,

        zero => zero

    );

    dcache : d\_cache PORT MAP(

        clock => clk,

        data\_write => data\_write,

        reset => reset,

        addr => alu\_out(4 DOWNTO 0),

        d\_in => rt\_data\_out,

        d\_out => d\_cache\_out

    );

    reg\_in\_src\_mux : two\_input\_mux PORT MAP(

        s => reg\_in\_src,

        in0 => d\_cache\_out,

        in1 => alu\_out,

        mux\_out => d\_cache\_mux\_out

    );

    rs\_out <= rs\_data\_out;

    rt\_out <= rt\_data\_out;

    pc\_out <= q\_out;

    instruction\_out <= instruction\_cache\_out;

END datapath\_arch;