Concordia University

Laboratory Report

COEN - 316

Lab – 5

**Datapath/Control Unit Integration**

**and system testing**

Submission Date: December 05, 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](#_Toc26120563)

[ Objective 3](#_Toc26120564)

[ Introduction 3](#_Toc26120565)

[ Conceptual Diagram of Datapath and the controller interfaced 4](#_Toc26120566)

[ Conceptual Diagram of CPU 5](#_Toc26120567)

[ Results 7](#_Toc26120568)

[o ModelSim Simulation CPU: (please zoom in the figures if required) 7](#_Toc26120569)

[o RTL schematic of the CPU generated 9](#_Toc26120570)

[o RTL schematic of the Datapath generated in lab 4 10](#_Toc26120571)

[o Precision log file 11](#_Toc26120572)

[o Impact log file 14](#_Toc26120573)

[o Listing of the directory contents of the generated system ace file 16](#_Toc26120574)

[ Conclusion 16](#_Toc26120575)

[o VHDL Code 17](#_Toc26120576)

[ CPU component 17](#_Toc26120577)

[ Controller component 21](#_Toc26120578)

[ datapath 28](#_Toc26120579)

# Figures

[Figure 1 Conceptual diagram of the designed Datapath and the controller interfaced together to create the MIPS CPU 4](#_Toc26120580)

[Figure 2 Port mapping of the Datapath to the controller (width of the signals are not shown in the figure) 5](#_Toc26120581)

[Figure 3 Control signal values 6](#_Toc26120582)

[Figure 4 Simulation of CPU unit that has Datapath and the Controller interfaced together 7](#_Toc26120583)

[Figure 6 RTL schematic of the VHDL code of CPU with Datapath and Controller interfaced together 9](#_Toc26120584)

[Figure 7 RTL schematic of the VHDL code of Datapath 10](#_Toc26120585)

* Objective

The objective of the lab was to design the controller for the MIPS CPU and integrate it with the Datapath designed in previous lab. The Datapath designed in lab 4 has ports to receive the control signals from the controller based on the instructions executed.

* Introduction

The Datapath designed in the previous lab needs 10 different control signals from the controller to be designed in lab 5.The control signals are as follows:

1. 2-bit pc\_sel
2. 2-bit branch\_type
3. 1-bit reg\_dest
4. 1-bit reg\_write
5. 1-bit alu\_src
6. 1-bit add\_sub
7. 2-bit logic\_func
8. 2-bit func
9. 1-bit data\_write
10. 1-bit reg\_in\_src

The ten control signals mentioned above must be generated from the controller and the value of the signals are decided by the controller based on the instruction executed. The controller specifically checks the 2-bit func and opcode of the instruction that is to be executed and generates the ten signals accordingly.

The conceptual diagram of the complete CPU with the Datapath and the controller is shown in the figures below. The details of the internal components of the Datapath was discussed in lab 4 report, but a summary of the list of components created for the Datapath are as follows:

1. Next\_address
2. Pc
3. I-cache
4. D-cache
5. Regfile
6. Reg\_des Mux
7. Sign\_extend
8. Alu\_src Mux
9. Alu
10. D-cache
11. Reg\_in\_src Mux

All the components mentioned above were created in separate files and declared in the Datapath component as separate components using the structural VHDL method and then the individual components were port mapped together to create a single Datapath component. Structural method was used to create the Datapath component to allow easier debugging of the internal components of the Datapath and easier readability of the VHDL code, additionally the RTL generated for the Datapath can be related to the conceptual diagram of the Datapath easily.

* Conceptual Diagram of Datapath and the controller interfaced

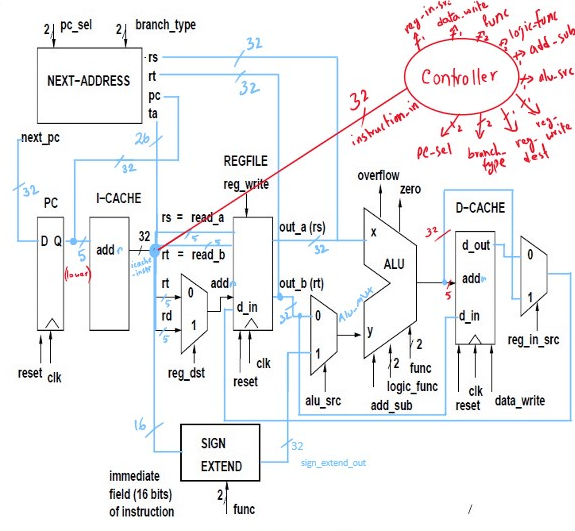


Figure Conceptual diagram of the designed Datapath and the controller interfaced together to create the MIPS CPU

The conceptual diagram of the Datapath and the controller are shown in the figure above and the interfacing of the two components can be seen in the same figure. The two components together create the MIPS CPU. The ten signals coming out of the controller are the signals shown as inputs to different components of the Datapath.

* Conceptual Diagram of CPU

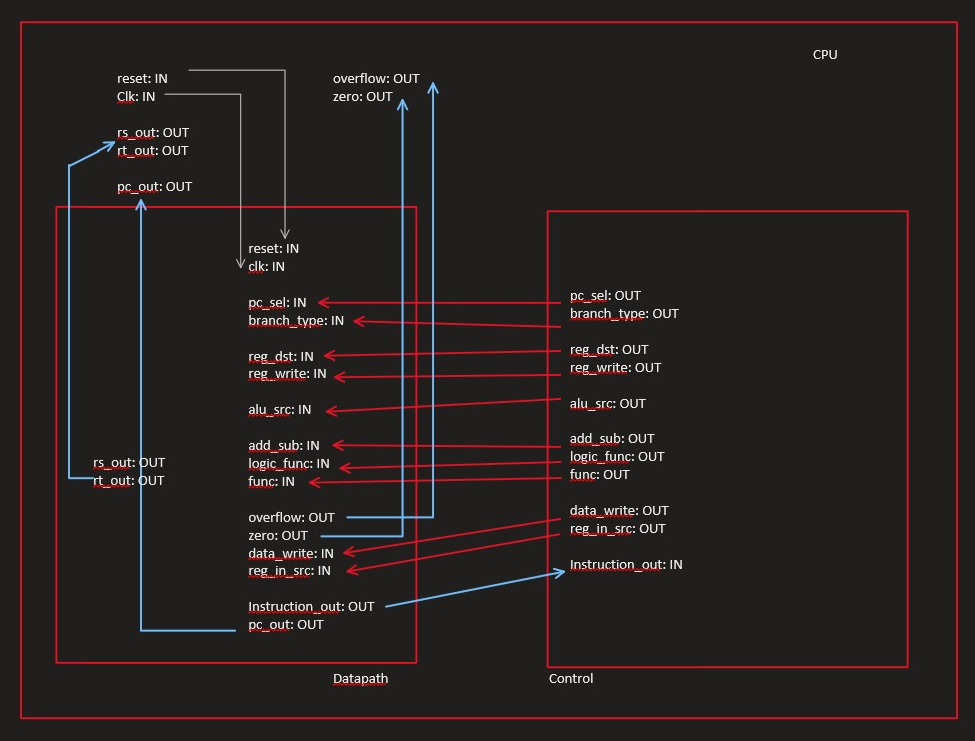


Figure Port mapping of the Datapath to the controller (width of the signals are not shown in the figure)

From the figure shown above we can see how the controller is port mapped with the Datapath which creates the CPU.

The controller designed during the lab sets the ten control signal based on the 6-bit opcode and 6-bit func code in the found in the instructions. A summary of the 20 control signals based on the op-code and func are shown in the figure below:



Figure Control signal values

Some of the control signals are don’t care based on the instruction being executed, those are marked with “X” and in the VHDL code they are either set to “0” or “1”.

* Results
  + ModelSim Simulation CPU: (please zoom in the figures if required)

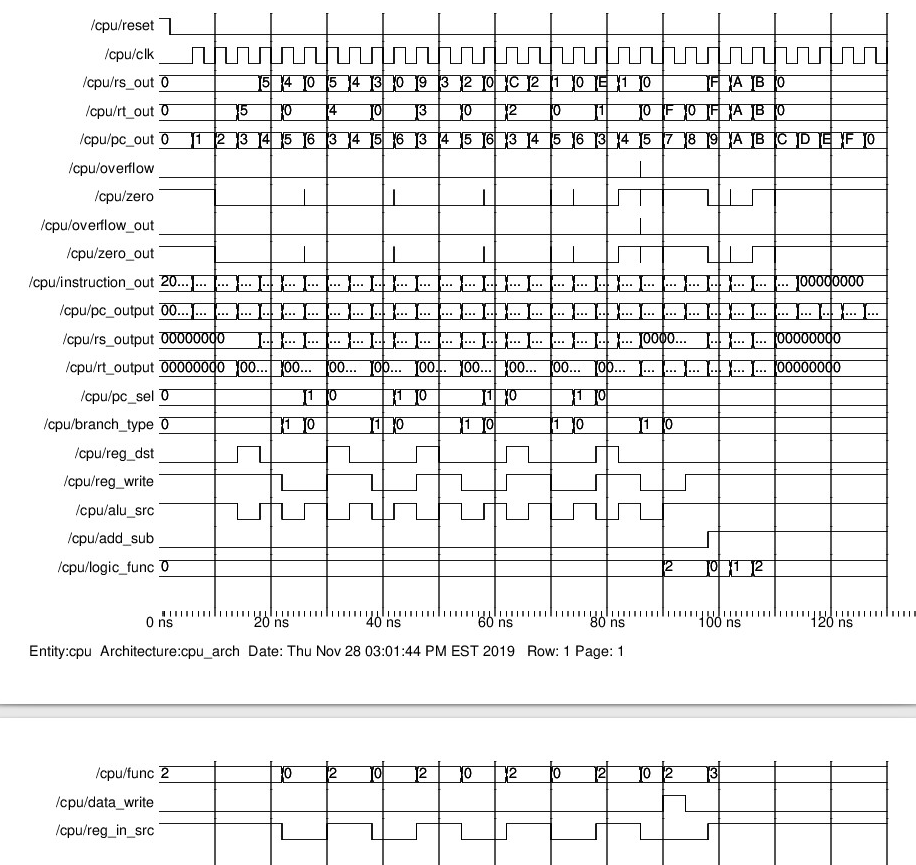


Figure Simulation of CPU unit that has Datapath and the Controller interfaced together

The CPU was testing using the following instructions in sequential order and the output shown in the figure above are as expected.

addi r3, r0, 0;

addi r1, r0, 0;

addi r2, r0, 5;

LOOP: addi r1, r1, r2;

addi r2, r2, -1;

beq r2,r3 (+1);

jump LOOP;

THERE sw r1, 0(r0);

lw r4, 0(r0);

andi r4, r4, 0x000A;

ori r4, r4, 0x0001;

xori r4, r4 , 0xB;

xori r4, r4, 0x0

The set of instructions ran above covers each class of instructions:

1. arithmetic with immediate
2. arithmetic without immediate
3. logic with immediate
4. logic without immediate
5. conditional branches
6. unconditional branch i.e. jump instruction
7. load from memory
8. store to memory

**The do file to test the CPU consists of focing the clk, reset signals only**

*Note: To get the rising edge of the clock the clock is always forced to 0 for 2ns and then forced to 1 to execute the next instruction.*

restart -f -- to reset values

add wave \*

force reset 1

force clk 0

run 2

force reset 0

run 2

force clk 0

run 2

force clk 1

run 2

… the above structure is executed numerous times to execute all the instructions in the I-Cache

* + RTL schematic of the CPU generated

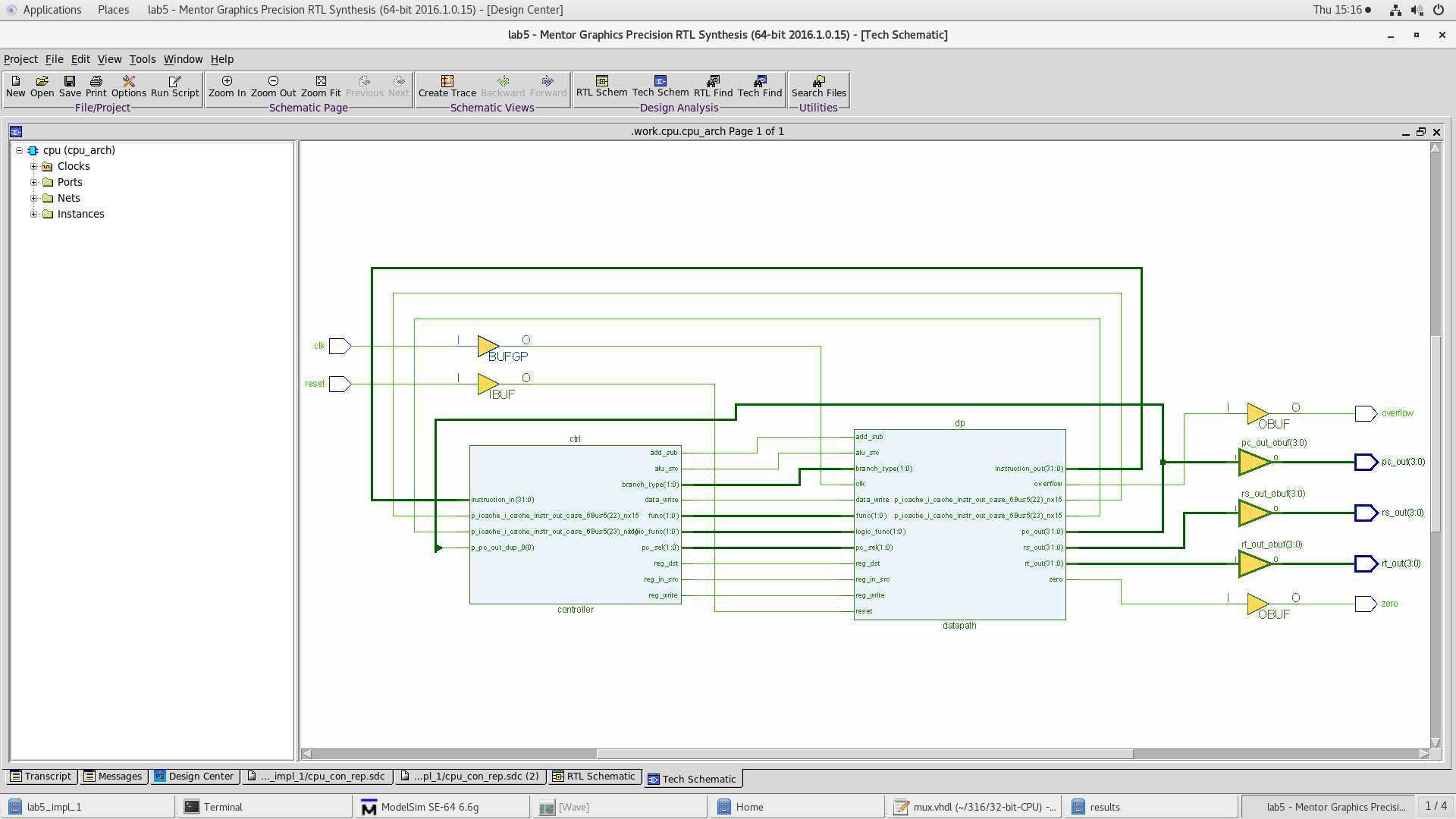


Figure RTL schematic of the VHDL code of CPU with Datapath and Controller interfaced together

The RTL schematic shows the synthesis of the MIPS CPU which consists of the Datapath and the Controller port mapped together. It resembles the conceptual diagram very closely shown in the report. Both the controller and the Datapath were created as a separate components and port mapped in another component called the “CPU”, this follows the structural VHDL method, which allows easier debugging and visualization of the components in the system.

* + RTL schematic of the Datapath generated in lab 4

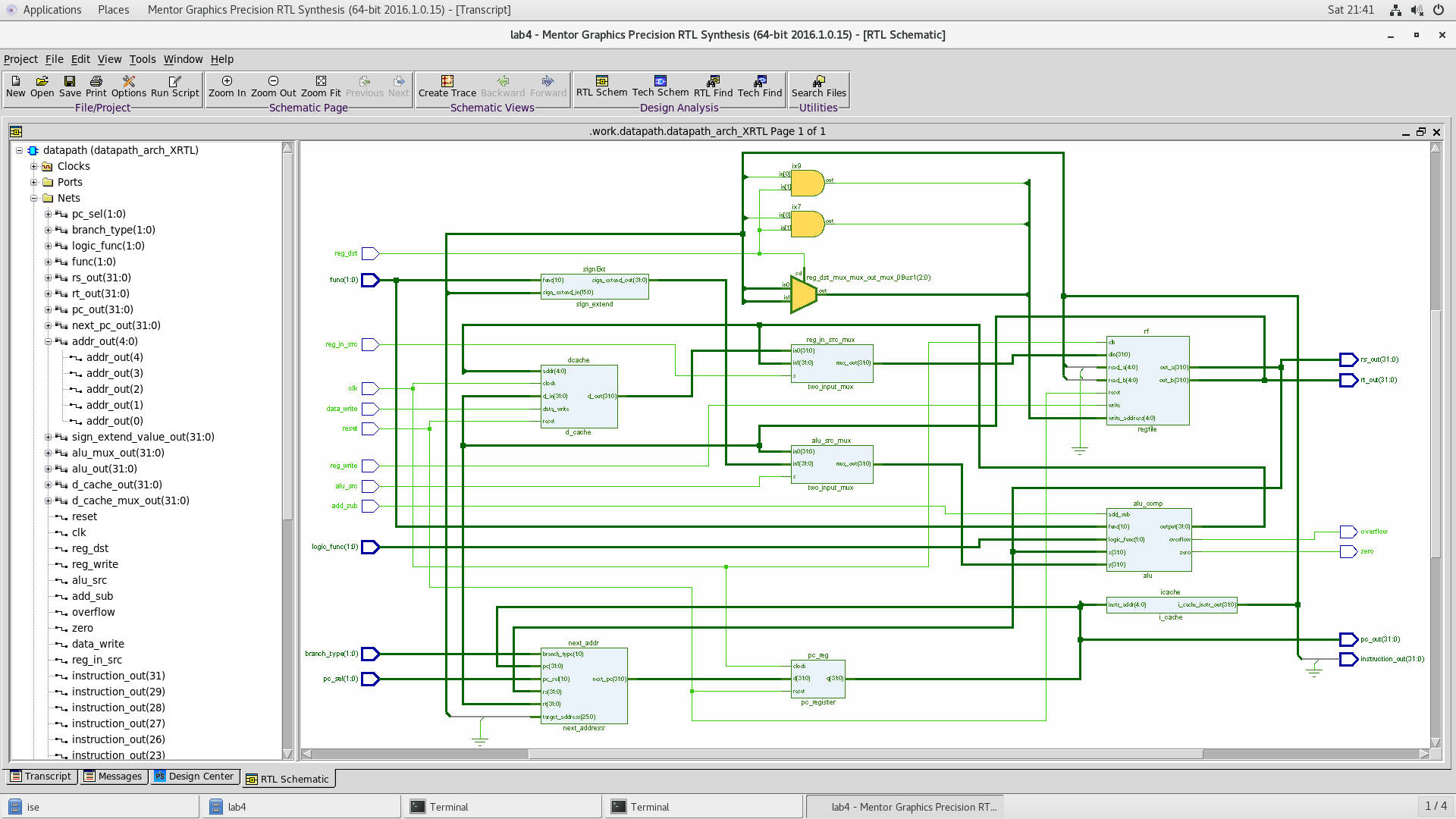


Figure RTL schematic of the VHDL code of Datapath

The RTL diagram of the Datapath is shown in the figure above, it is the same diagram submitted in the lab 4 report and included in this report for better visualization and completeness of the report.

* + Precision log file

# Info: [9566]: Logging session transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/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 22:12:57 2019

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

# Info: [9566]: Logging session transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/precision.log

# COMMAND: new\_project -name lab5 -folder /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5 -createimpl\_name lab5\_impl\_1

# Info: [9574]: Input directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5

# Info: [9569]: Moving session transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/precision.log

# Info: [9555]: Created project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5.psp in folder /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5.

# Info: [9531]: Created directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5\_impl\_1.

# Info: [9554]: Created implementation lab5\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5.psp.

# Info: [9575]: The Results Directory has been set to: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5\_impl\_1/

# Info: [9566]: Logging project transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5\_impl\_1/precision.log

# Info: [9566]: Logging suppressed messages transcript to file /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5\_impl\_1/precision.log.suppressed

# Info: [9550]: Activated implementation lab5\_impl\_1 in project /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5.psp.

new\_project -name lab5 -folder /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5 -createimpl\_name lab5\_impl\_1

# COMMAND: add\_input\_file {../../32-bit-CPU/32-bit-register.vhd ../../32-bit-CPU/alu.vhd ../../32-bit-CPU/controller.vhdl ../../32-bit-CPU/cpu.vhdl ../../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}

add\_input\_file {../../32-bit-CPU/32-bit-register.vhd ../../32-bit-CPU/alu.vhd ../../32-bit-CPU/controller.vhdl ../../32-bit-CPU/cpu.vhdl ../../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/lab5/../../32-bit-CPU/32-bit-register.vhd" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/alu.vhd" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/cpu.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/d\_cache.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/datapath.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/i\_cache.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/mux.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/mux\_5\_bit.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/next\_address.vhd" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/pc\_register.vhdl" ...

# Info: [42502]: Analyzing input file "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/sign\_extend.vhdl" ...

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

# Info: [657]: Current working directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5\_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.cpu(cpu\_arch): Pre-processing...

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

# Info: [44506]: 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'.

# Warning: [45729]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 42: Input port instruction\_in[25:6] has never been used.

# Warning: [45729]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/cpu.vhdl", line 76: signal pc\_output[31:4] has never been used.

# Warning: [45729]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/cpu.vhdl", line 77: signal rs\_output[31:4] has never been used.

# Warning: [45729]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/cpu.vhdl", line 78: signal rt\_output[31:4] has never been used.

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

# 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: [44508]: Module work.datapath(datapath\_arch): Compiling...

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

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) pc\_sel[1:0]: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) branch\_type[1:0]: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) reg\_dst: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) reg\_write: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) alu\_src: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) add\_sub: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) logic\_func[1:0]: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) func[1:0]: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) data\_write: Latch inferred.

# Info: [45205]: "/nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/../../32-bit-CPU/controller.vhdl", line 58: Module work.controller(controller\_arch), Net(s) reg\_in\_src: Latch inferred.

# 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: 1189.

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

# Info: [44513]: Overall running time for compilation: 1.0 secs.

# Info: [657]: Current working directory: /nfs/home/s/sal\_rahm/316/fpga\_adv/lab5/lab5\_impl\_1.

# Info: [15330]: Doing rtl optimizations.

# Info: [660]: Finished compiling design.

compile

* + Impact log file

iMPACT Version: 10.1

iMPACT log file Started on Sat Nov 30 22:44:46 2019

// \*\*\* BATCH CMD : setMode -acecf

// \*\*\* BATCH CMD : setAttribute -configdevice -attr size -value "134217728"

// \*\*\* BATCH CMD : setMode -acecf

// \*\*\* BATCH CMD : addConfigDevice -size 134217728 -name "Untitled" -path"/nfs/home/s/sal\_rahm/316/impact/lab5/"

// \*\*\* BATCH CMD : addCollection -name "Untitled"

// \*\*\* BATCH CMD : addDesign -version 0 -name "rev0"

// \*\*\* BATCH CMD : setMode -acecf

// \*\*\* BATCH CMD : addDeviceChain -index 0

// \*\*\* BATCH CMD : setAttribute -configdevice -attr compressed -value "FALSE"

// \*\*\* BATCH CMD : setAttribute -configdevice -attr compressed -value "FALSE"

// \*\*\* BATCH CMD : setMode -acecf

// \*\*\* BATCH CMD : setCurrentCollection -collection "Untitled"

// \*\*\* BATCH CMD : setCurrentDesign -version 0

// \*\*\* BATCH CMD : deleteDesign -version 0

// \*\*\* BATCH CMD : setCurrentDesign -version -1

// \*\*\* BATCH CMD : deleteCollection -name "Untitled"

// \*\*\* BATCH CMD : setAttribute -configdevice -attr size -value "128000000"

// \*\*\* BATCH CMD : setMode -acecf

// \*\*\* BATCH CMD : addConfigDevice -size 128000000 -name "XCACECF" -path"/nfs/home/s/sal\_rahm/316/impact/lab5/Untitled"

// \*\*\* BATCH CMD : addCollection -name "Untitled"

// \*\*\* BATCH CMD : setMode -acecf

// \*\*\* BATCH CMD : setAttribute -configdevice -attr size -value "134217728"

// \*\*\* BATCH CMD : setAttribute -configdevice -attr reserveSize -value "0"

// \*\*\* BATCH CMD : setAttribute -configdevice -attr name -value "XCCACE128-I"

// \*\*\* BATCH CMD : addDesign -version 0 -name "rev0"

// \*\*\* BATCH CMD : setMode -acecf

// \*\*\* BATCH CMD : addDeviceChain -index 0

------------------ GUI: Wizard Data Report -------------------------

File Mode : ACECF

Collection Name : Untitled

Collection Locatnion : /nfs/home/s/sal\_rahm/316/impact/lab5/

Collection Size : 0 Mbits

Reserved Space : 0 Mbits

Number of Design : 1

Config Address and Design Name list :

Version: 0 DesignName: rev0

------------------------ END of Report ------------------------------

V2Pro Part: xc2vp30 w/2 ppc

// \*\*\* BATCH CMD : setMode -acecf

1. Initializing V2Pro File...

// \*\*\* BATCH CMD : addDevice -p 1 -file"/nfs/home/s/sal\_rahm/316/ise/lab5/lab5/cpu.bit"

'1': Loading file '/nfs/home/s/sal\_rahm/316/ise/lab5/lab5/cpu.bit' ...

done.

INFO:iMPACT:1777 -

Reading /nfs/sw\_cmc/x86\_64.EL7/tools/xilinx\_10.1/ISE/virtex2p/data/xc2vp30.bsd...

INFO:iMPACT:501 - '1': Added Device xc2vp30 successfully.

----------------------------------------------------------------------

----------------------------------------------------------------------

Add one device.// \*\*\* BATCH CMD : setAttribute -configdevice -attr path -value"/nfs/home/s/sal\_rahm/316/impact/lab5"

// \*\*\* BATCH CMD : generate -active Untitled

INFO:iMPACT:794 - Creating SVF File /nfs/home/s/sal\_rahm/316/impact/lab5/Untitled/rev0/rev0.svf...

'1': Programming device...

PROGRESS\_START - Starting Operation.

Match\_cycle = NoWait.

Match cycle: NoWait

done.

INFO:iMPACT:579 - '1': Completed downloading bit file to device.

Match\_cycle = NoWait.

Match cycle: NoWait

INFO:iMPACT - '1': Checking done pin....done.

'1': Programmed successfully.

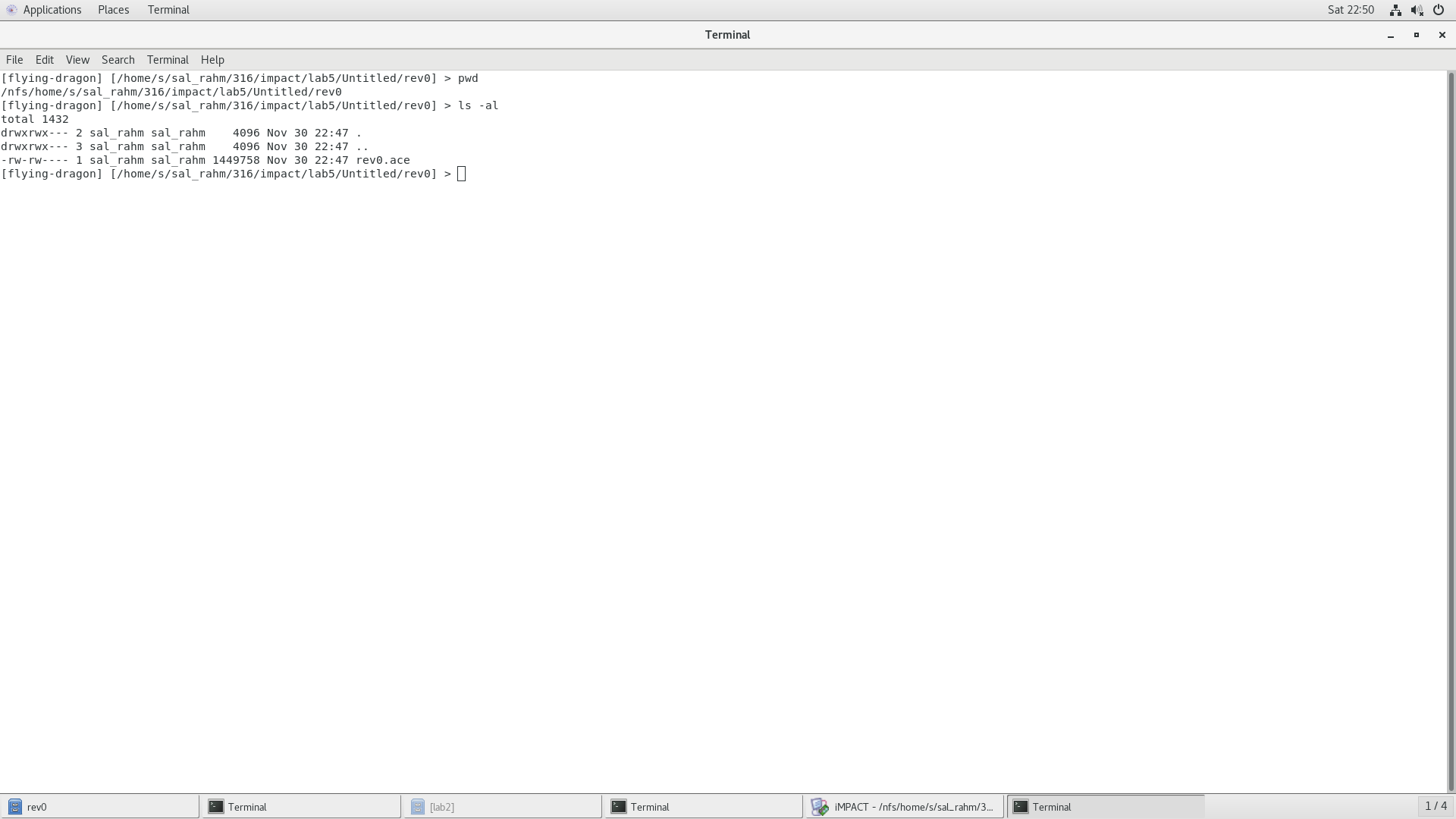
PROGRESS\_END - End Operation.

Elapsed time = 1 sec.

Creating Ace File /nfs/home/s/sal\_rahm/316/impact/lab5/Untitled/rev0/rev0.ace...

Copy active xilinx.sys /nfs/home/s/sal\_rahm/316/impact/lab5/Untitled/xilinx.systo root.

* + Listing of the directory contents of the generated system ace file



* Conclusion

In conclusion, the MIPS CPU was successfully implemented during the lab by interfacing the Datapath and Controller components together. A total of ten control signals were used from the controller to the Datapath to operate the MIPS CPU properly. The MIPS CPU were able to execute 20 different instructions as shown in figure 3.

* + VHDL Code
    - CPU component
* -- salman rahman
* -- 27853815
* LIBRARY IEEE;
* USE IEEE.std\_logic\_1164.ALL;
* -- USE IEEE.std\_logic\_unsigned.ALL;
* USE IEEE.std\_logic\_signed.ALL;
* ENTITY cpu IS
* PORT (
* reset : IN std\_logic;
* clk : IN std\_logic;
* rs\_out, rt\_out : OUT std\_logic\_vector(3 DOWNTO 0);
* -- output ports from the register file
* pc\_out : OUT std\_logic\_vector(3 DOWNTO 0);
* overflow, zero : OUT std\_logic
* );
* END cpu;
* ARCHITECTURE cpu\_arch OF cpu IS
* COMPONENT datapath
* PORT (
* reset : IN std\_logic;
* clk : IN std\_logic;
* pc\_sel : IN std\_logic\_vector(1 DOWNTO 0) := "00";
* branch\_type : IN std\_logic\_vector(1 DOWNTO 0) := "00";
* reg\_dst : IN std\_logic := '0';
* reg\_write : IN std\_logic := '0';
* alu\_src : IN std\_logic := '0';
* add\_sub : IN std\_logic := '0';
* logic\_func : IN std\_logic\_vector(1 DOWNTO 0) := "00";
* func : IN std\_logic\_vector(1 DOWNTO 0) := "00";
* overflow : OUT std\_logic;
* zero : OUT std\_logic;
* data\_write : IN std\_logic := '0';
* reg\_in\_src : IN std\_logic := '0';
* instruction\_out : OUT std\_logic\_vector(31 DOWNTO 0) := "00000000000000000000000000000000";
* rs\_out : OUT std\_logic\_vector(31 DOWNTO 0):= "00000000000000000000000000000000";
* rt\_out : OUT std\_logic\_vector(31 DOWNTO 0):= "00000000000000000000000000000000";
* pc\_out : OUT std\_logic\_vector(31 DOWNTO 0):= "00000000000000000000000000000000"
* );
* END COMPONENT;
* COMPONENT controller
* PORT (
* pc\_sel : OUT std\_logic\_vector(1 DOWNTO 0) :="00";
* branch\_type : OUT std\_logic\_vector(1 DOWNTO 0) :="00";
* reg\_dst : OUT std\_logic :='0';
* reg\_write : OUT std\_logic :='0';
* alu\_src : OUT std\_logic :='0';
* add\_sub : OUT std\_logic :='0' ;
* logic\_func : OUT std\_logic\_vector(1 DOWNTO 0):="00";
* func : OUT std\_logic\_vector(1 DOWNTO 0) :="00";
* data\_write : OUT std\_logic :='0';
* reg\_in\_src : OUT std\_logic :='0';
* instruction\_in : IN std\_logic\_vector(31 DOWNTO 0) := (OTHERS => '0')
* );
* END COMPONENT;
* -- datapath outputs
* SIGNAL overflow\_out : std\_logic;
* SIGNAL zero\_out : std\_logic;
* SIGNAL instruction\_out : std\_logic\_vector(31 DOWNTO 0);
* SIGNAL pc\_output : std\_logic\_vector(31 DOWNTO 0);
* SIGNAL rs\_output : std\_logic\_vector(31 DOWNTO 0);
* SIGNAL rt\_output : std\_logic\_vector(31 DOWNTO 0);
* -- control outputs
* SIGNAL pc\_sel : std\_logic\_vector(1 DOWNTO 0);
* SIGNAL branch\_type : std\_logic\_vector(1 DOWNTO 0);
* SIGNAL reg\_dst : std\_logic;
* SIGNAL reg\_write : std\_logic;
* SIGNAL alu\_src : std\_logic;
* SIGNAL add\_sub : std\_logic;
* SIGNAL logic\_func : std\_logic\_vector(1 DOWNTO 0);
* SIGNAL func : std\_logic\_vector(1 DOWNTO 0);
* SIGNAL data\_write : std\_logic;
* SIGNAL reg\_in\_src : std\_logic;
* BEGIN
* ctrl : controller PORT MAP(
* pc\_sel => pc\_sel,
* branch\_type => branch\_type,
* reg\_dst => reg\_dst,
* reg\_write => reg\_write,
* alu\_src => alu\_src,
* add\_sub => add\_sub,
* logic\_func => logic\_func,
* func => func,
* data\_write => data\_write,
* reg\_in\_src => reg\_in\_src,
* instruction\_in => instruction\_out
* );
* dp : datapath PORT MAP(
* reset => reset,
* clk => clk,
* pc\_sel => pc\_sel,
* branch\_type => branch\_type,
* reg\_dst => reg\_dst,
* reg\_write => reg\_write,
* alu\_src => alu\_src,
* add\_sub => add\_sub,
* logic\_func => logic\_func,
* func => func,
* overflow => overflow\_out,
* zero => zero\_out,
* data\_write => data\_write,
* reg\_in\_src => reg\_in\_src,
* instruction\_out => instruction\_out,
* rs\_out => rs\_output,
* rt\_out => rt\_output,
* pc\_out => pc\_output
* );
* rs\_out <=  rs\_output(3 downto 0);
* rt\_out <=  rt\_output(3 downto 0);
* pc\_out <=  pc\_output(3 downto 0);
* overflow <=  overflow\_out;
* zero <=  zero\_out;
* --    rs\_out <= not rs\_output(3 downto 0);
* --   rt\_out <= not rt\_output(3 downto 0);
* -- pc\_out <= not pc\_output(3 downto 0);
* --overflow <= not overflow\_out;
* --zero <= not zero\_out;
* END cpu\_arch;
  + - Controller component
* -- salman rahman
* -- 27853815
* LIBRARY ieee;
* USE ieee.std\_logic\_1164.ALL;
* -- USE ieee.numeric\_std.ALL;
* ENTITY controller IS
* PORT (
* -- next address ports
* pc\_sel : OUT std\_logic\_vector(1 DOWNTO 0);
* branch\_type : OUT std\_logic\_vector(1 DOWNTO 0);
* -- reg\_des MUX ports
* reg\_dst : OUT std\_logic;
* -- sign extend ports
* -- func : out std\_logic\_vector(1 DOWNTO 0);
* -- regfile ports block
* reg\_write : OUT std\_logic;
* -- alu\_src MUX ports
* alu\_src : OUT std\_logic;
* -- alu ports
* add\_sub : OUT std\_logic;
* logic\_func : OUT std\_logic\_vector(1 DOWNTO 0);
* func : OUT std\_logic\_vector(1 DOWNTO 0);
* -- overflow : OUT std\_logic;
* -- zero : OUT std\_logic
* -- d-cache ports
* data\_write : OUT std\_logic;
* -- reg\_in\_src MUX port
* reg\_in\_src : OUT std\_logic;
* -- control port
* instruction\_in : IN std\_logic\_vector(31 DOWNTO 0)
* );
* END controller;
* ARCHITECTURE controller\_arch OF controller IS
* SIGNAL op\_code : std\_logic\_vector(5 DOWNTO 0);
* SIGNAL func\_code : std\_logic\_vector(5 DOWNTO 0);
* BEGIN
* PROCESS (instruction\_in)
* BEGIN
* op\_code <= instruction\_in(31 DOWNTO 26);
* func\_code <= instruction\_in(5 DOWNTO 0);
* END PROCESS;
* PROCESS (op\_code, func\_code)
* BEGIN
* CASE op\_code IS
* WHEN "000000" =>
* CASE func\_code IS
* WHEN "100000" => -- add
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '0';
* logic\_func <= "00";
* func <= "10";
* data\_write <= '0';
* WHEN "100010" => -- sub
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '1';
* logic\_func <= "00";
* func <= "10";
* data\_write <= '0';
* WHEN "101010" => -- slt
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '1';
* logic\_func <= "00";
* func <= "01";
* data\_write <= '0';
* WHEN "100100" => -- and
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '1';
* logic\_func <= "00";
* func <= "11";
* data\_write <= '0';
* WHEN "100101" => -- or
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '1';
* logic\_func <= "01";
* func <= "11";
* data\_write <= '0';
* WHEN "100110" => -- xor
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '1';
* logic\_func <= "10";
* func <= "11";
* data\_write <= '0';
* WHEN "100111" => -- nor
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '1';
* logic\_func <= "11";
* func <= "11";
* data\_write <= '0';
* WHEN "001000" => -- jr
* pc\_sel <= "10";
* branch\_type <= "00";
* reg\_dst <= '1';
* reg\_write <= '0';
* reg\_in\_src <= '1';
* alu\_src <= '0';
* add\_sub <= '0';
* logic\_func <= "00";
* func <= "00";
* data\_write <= '0';
* WHEN OTHERS =>
* END CASE; -- end of func case
* WHEN "001111" => --lui
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '1';
* add\_sub <= '0';
* logic\_func <= "00";
* func <= "00";
* data\_write <= '0';
* WHEN "001000" => --addi
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '1';
* add\_sub <= '0';
* logic\_func <= "00";
* func <= "10";
* data\_write <= '0';
* WHEN "001010" => --slti
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '1';
* add\_sub <= '1';
* logic\_func <= "00";
* func <= "10";
* data\_write <= '0';
* WHEN "001100" => --andi
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '1';
* add\_sub <= '1';
* logic\_func <= "00";
* func <= "11";
* data\_write <= '0';
* WHEN "001101" => --ori
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '1';
* add\_sub <= '1';
* logic\_func <= "01";
* func <= "11";
* data\_write <= '0';
* WHEN "001110" => --xori
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '1';
* reg\_in\_src <= '1';
* alu\_src <= '1';
* add\_sub <= '1';
* logic\_func <= "10";
* func <= "11";
* data\_write <= '0';
* WHEN "100011" => --lw
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '1';
* reg\_in\_src <= '0';
* alu\_src <= '1';
* add\_sub <= '0';
* logic\_func <= "10";
* func <= "10";
* data\_write <= '0';
* WHEN "101011" => --sw
* pc\_sel <= "00";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '0';
* reg\_in\_src <= '0';
* alu\_src <= '1';
* add\_sub <= '0';
* logic\_func <= "10";
* func <= "10";
* data\_write <= '1';
* WHEN "000010" => --j
* pc\_sel <= "01";
* branch\_type <= "00";
* reg\_dst <= '0';
* reg\_write <= '0';
* reg\_in\_src <= '0';
* alu\_src <= '1';
* add\_sub <= '0';
* logic\_func <= "00";
* func <= "00";
* data\_write <= '0';
* WHEN "000001" => --bltz
* pc\_sel <= "00";
* branch\_type <= "11";
* reg\_dst <= '0';
* reg\_write <= '0';
* reg\_in\_src <= '0';
* alu\_src <= '1';
* add\_sub <= '1';
* logic\_func <= "00";
* func <= "00";
* data\_write <= '0';
* WHEN "000100" => --beq
* pc\_sel <= "00";
* branch\_type <= "01";
* reg\_dst <= '0';
* reg\_write <= '0';
* reg\_in\_src <= '0';
* alu\_src <= '0';
* add\_sub <= '0';
* logic\_func <= "00";
* func <= "00";
* data\_write <= '0';
* WHEN "000101" => --bne
* pc\_sel <= "00";
* branch\_type <= "10";
* reg\_dst <= '0';
* reg\_write <= '0';
* reg\_in\_src <= '0';
* alu\_src <= '1';
* add\_sub <= '0';
* logic\_func <= "00";
* func <= "00";
* data\_write <= '0';
* WHEN OTHERS =>
* END CASE;
* END PROCESS;
* END controller\_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;