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

- 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:

    a. PC
    b. I-CACHE
    c. reg_des – Mux
    d. REGFILE
    e. alu_src – Mux
    f. ALU – Mux
    g. D-Cache
    h. reg_in_src Mux
    i. Sign EXTEND block
    j. NEXT-ADDRESS

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

    a. J-Format instruction (format shown in figure below)
        I. Jump – j there
        II. Jump register – jr rs

    b. I-Format instruction (format shown in figure below)
        I. Load upper immediate – lui rt, immediate
        II. Set less than immediate – slti rt,rs,immediate
        III. Add immediate – addi rt,rs,immediate
        IV. AND immediate – andi rt,rs,immediate
        V. OR immediate – ori rt, rs,immediate
        VI. XOR immediate – xori rt,rs,immediate
        VII. Load word – lw rt, immediate(rs)
        VIII. Store word – sw rt, immediate(rs)
        IX. Branch less than 0 – bltz rs, there
        X. Branch equal – beq rs, rt, there
        XI. Branch not equal 0 – bne rs, rt, there

    c. R-Format instruction (format shown in figure below)
        I. Addition– add rd, rs,rt
        II. Subtraction – sub rd, rs, rt
        III. Set less than – slt rd, rs, rt
        IV. AND (logical) – and rd, rs, rt

V. OR (logical) – or rd, rs, rt
VI. XOR (logical) – xor rd, rs, rt
VII. NOR (logical) – nor rd, rs, rt

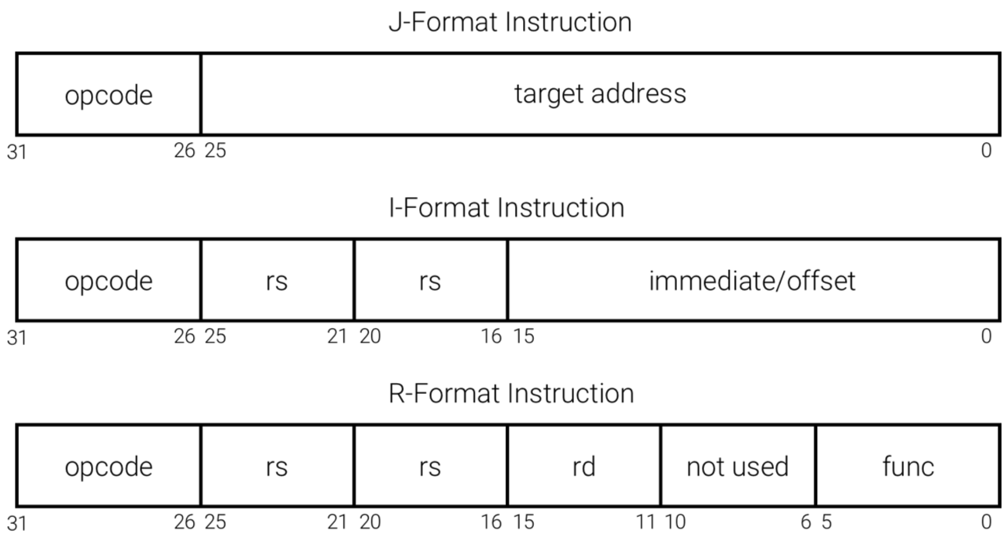


Figure 1 Instructions format of MIPS

- Conceptual Diagram of datapath

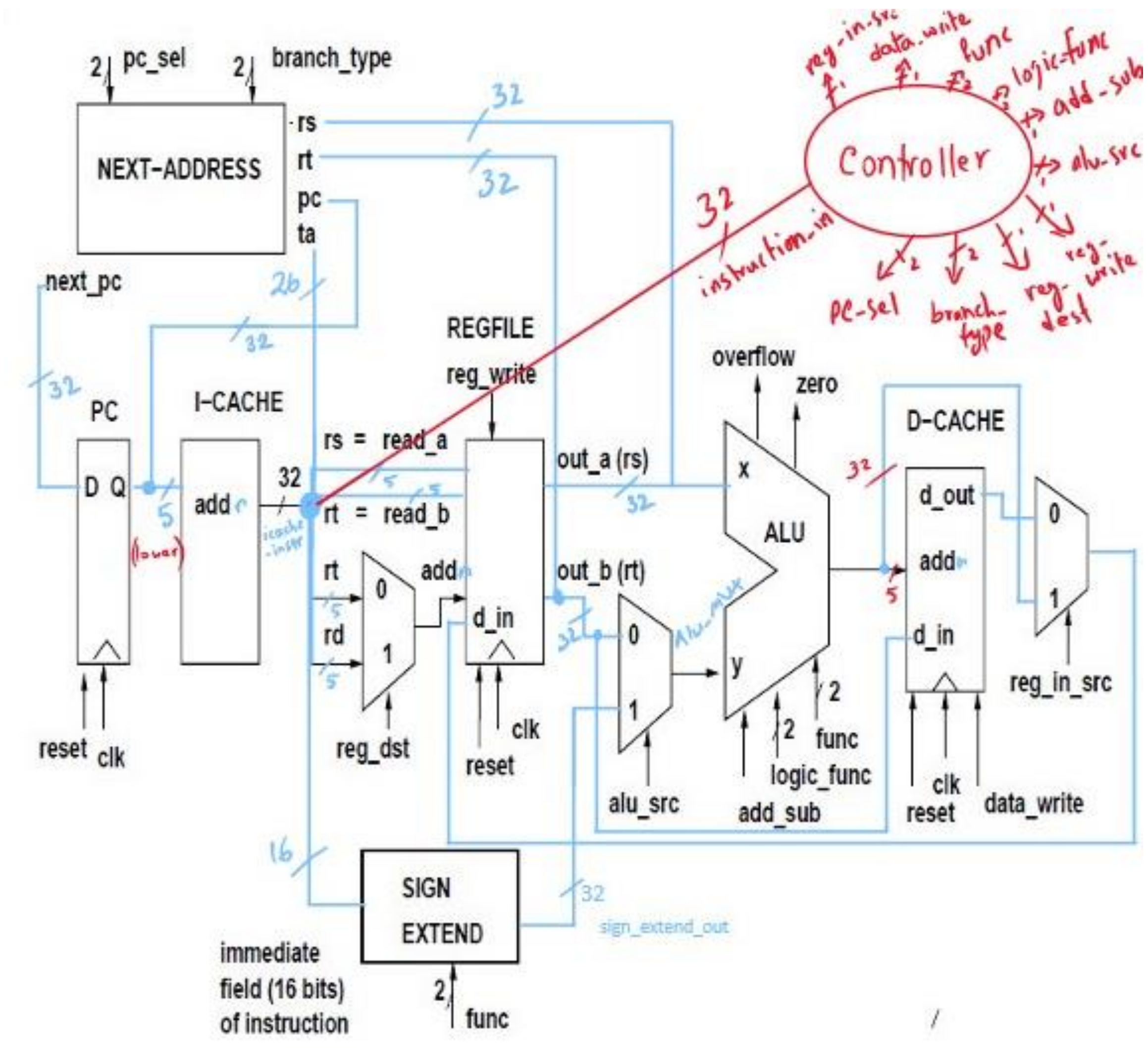


Figure 2 Conceptual diagram of the designed datapath

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:

i. **<u>NEXT-ADDRESS</u>**

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.

ii. <u>PC</u>

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.

iii. <u>I-CACHE</u>

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.

iv. <u>REGFILE</u>

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.

v. <u>Reg_des MUX</u>

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.

vi. <u>SIGN EXTEND</u>

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	$i_{15}i_{15} \dots i_{15}i_{14}i_{13} \dots i_1 i_0$
01	set less than immediate	$00 \dots 00 i_{15}i_{14}i_{13} \dots i_1 i_0$
10	arithmetic	$i_{15}i_{15} \dots i_{15}i_{14}i_{13} \dots i_1 i_0$
11	logical	$00 \dots 00 i_{15}i_{14}i_{13} \dots i_1 i_0$

vii. **Alu_src MUX**

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

viii. **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.

ix. **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”.

x. **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)

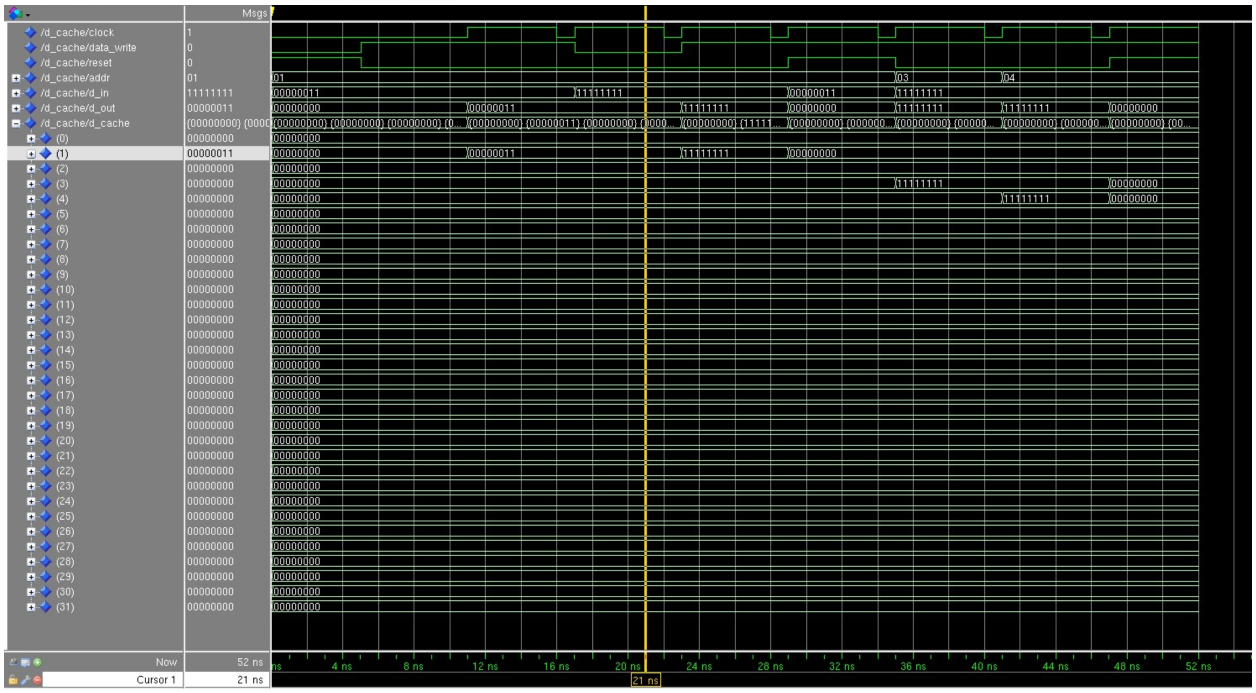


Figure 3 Simulation of d-cache unit is shown with different sets of inputs

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)

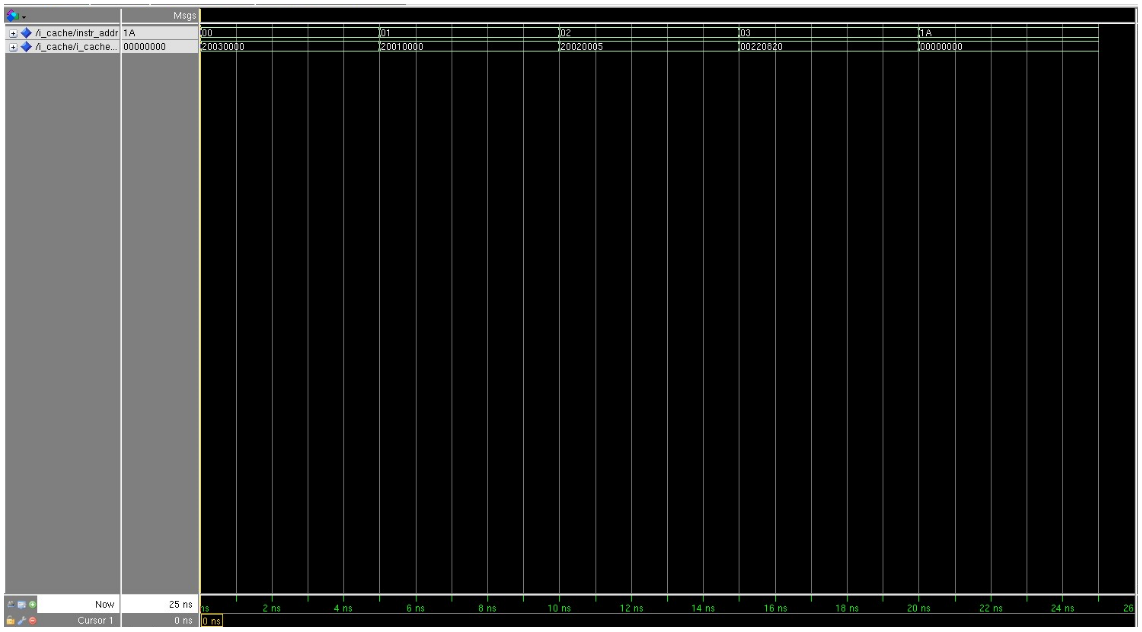


Figure 4 Simulation of i-cache unit is shown with different sets of inputs

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

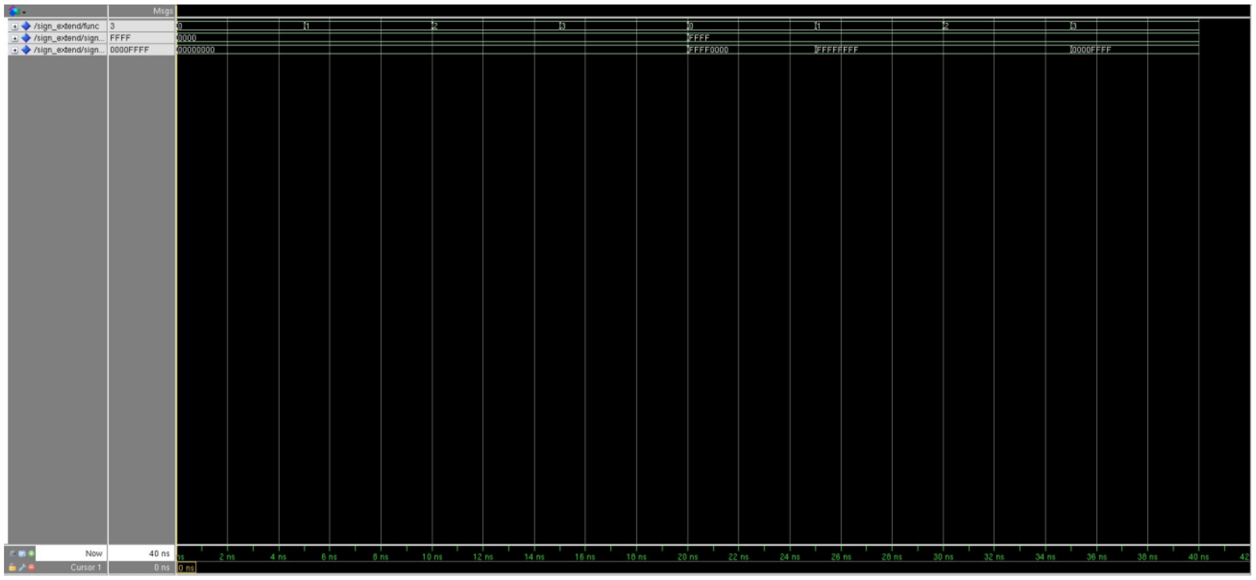
force instr_addr 00001
run 5

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)


Msgs
/sign_extend/func 3
/sign_extend/sign... FFFF
/sign_extend/sign... 0000FFFF
0 1 2 3 0 1 2 3
0000 FFFF
00000000 FFFF0000 FFFFFFFF 0000FFFF
Now 40 ns
Cursor 1 0 ns
2 ns 4 ns 6 ns 8 ns 10 ns 12 ns 14 ns 16 ns 18 ns 20 ns 22 ns 24 ns 26 ns 28 ns 30 ns 32 ns 34 ns 36 ns 38 ns 40 ns

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
```

o 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

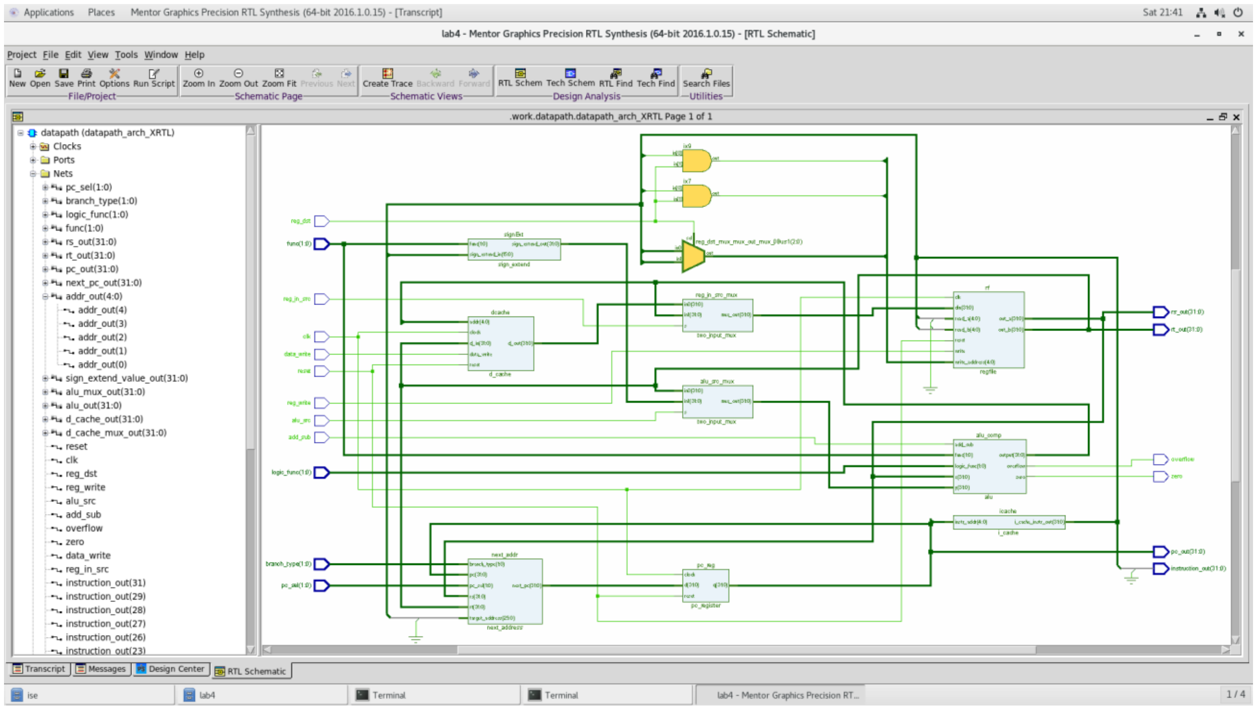


Figure 5 RTL schematic of the VHDL code of Datapath

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);
```

```vhdl
        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 <= "00100000000000011000000000000000";
            --addi r2, r0, 2
        WHEN "00001" =>
            i_cache_instr_out <= "00100000000000010000000000000000";
            -- add r2, r2, r1
        WHEN "00010" =>
            i_cache_instr_out <= "00100000000000010000000000000101";
            -- 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 <= "10001100000000100000000000000000";

            WHEN "01001" =>
            i_cache_instr_out <= "00110000100000100000000000001010";


            WHEN "01010" =>
            i_cache_instr_out <= "00110100100000100000000000000001";

            WHEN "01011" =>
            i_cache_instr_out <= "00111000100000100000000000001011";


            WHEN "01100" =>
            i_cache_instr_out <= "00111000100000100000000000000000";

            -- do not care
        WHEN OTHERS =>
            i_cache_instr_out <= (OTHERS => '0');
    END CASE;

end process;
END i_cache_arch;
```

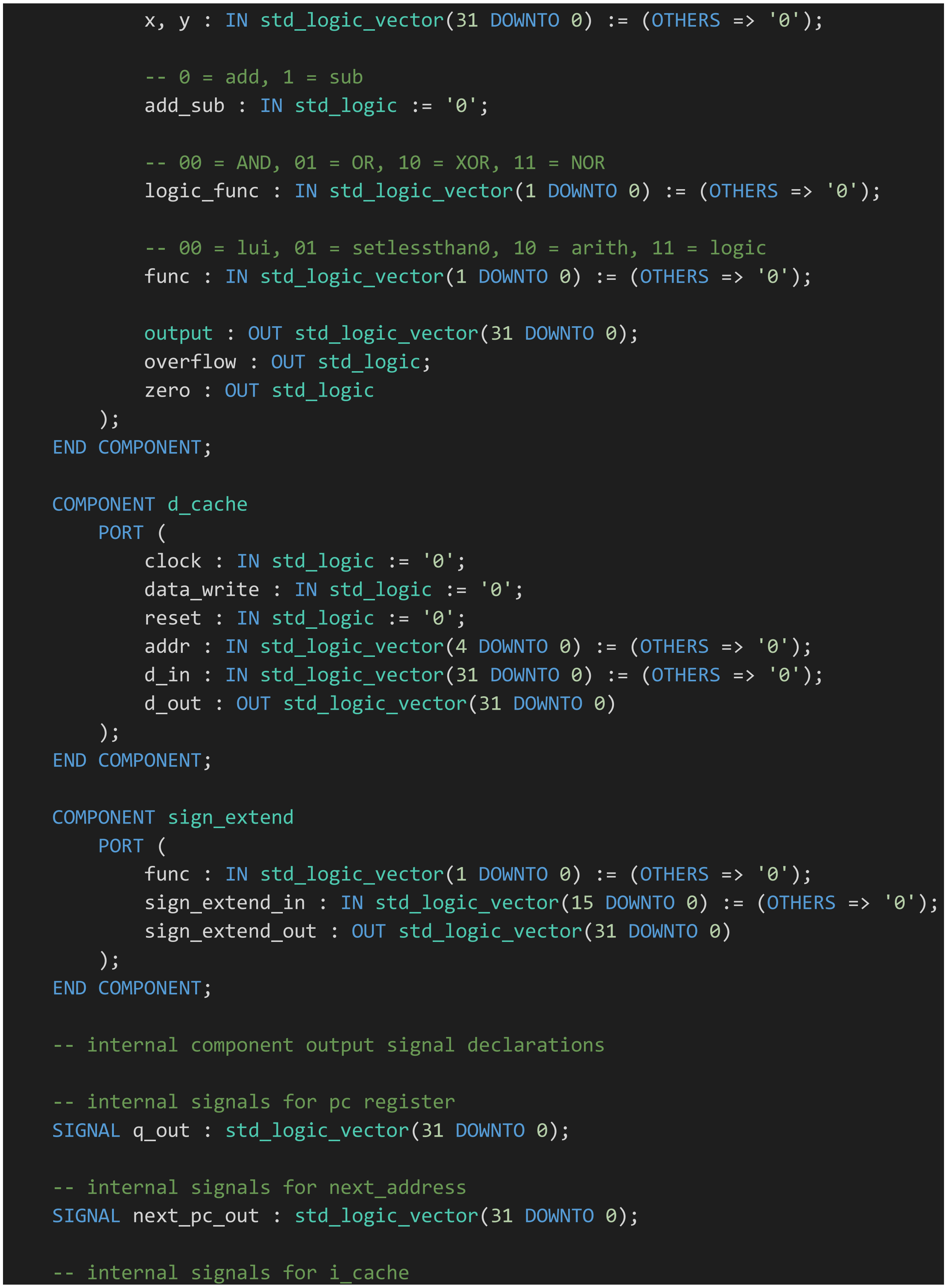
- pc_register

```
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
```



```vhdl
            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,
```

```vhdl
        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;
```