Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	COE
Course Number:	608
Semester/Year (e.g.F2016)	W2024

Instructor:	Dr. Hafeez

Assignment/Lab Number:	2
Assignment/Lab Title:	Program Counter and Register Set Design

Submission Date:	30/1/2024
Due Date:	30/1/2024

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Mohamed	Lama	501042394	042	L A M A

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

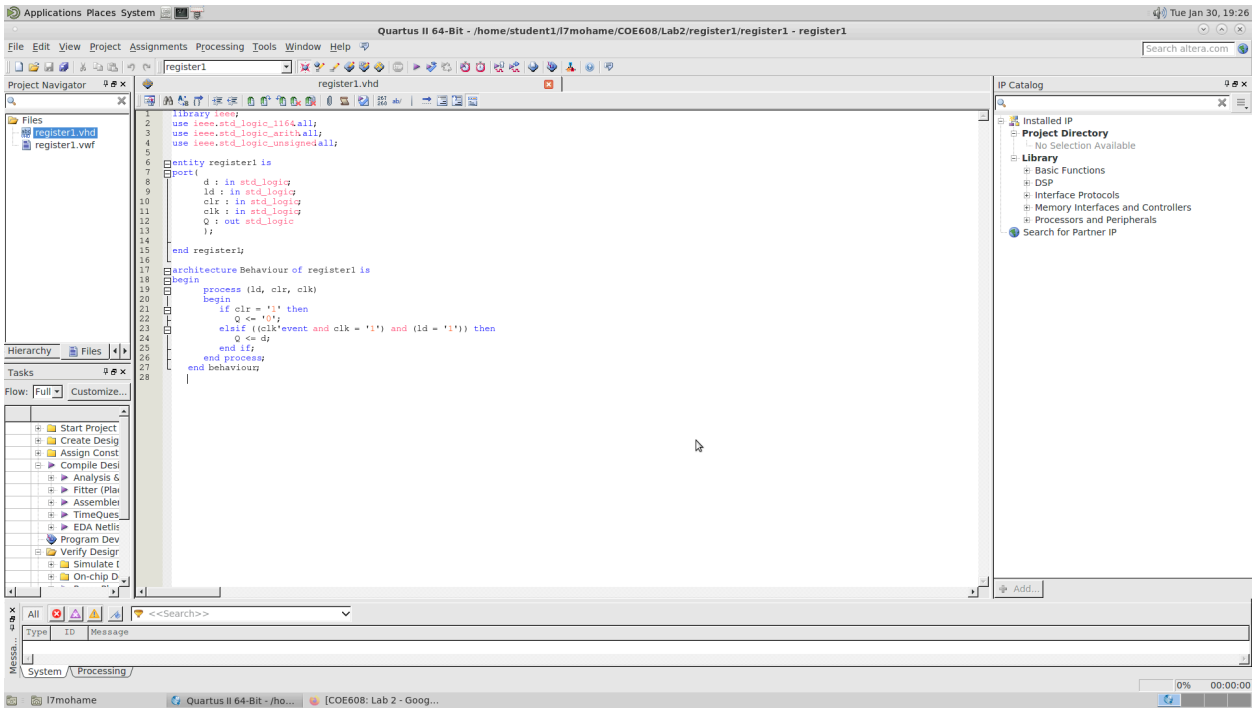
COE608: Lab 2

1. Lab Objective

We are designing a processor that requires several registers that are either 32-bit or 1-bit. Therefore we were required to implement a 1-bit and a 32-bit register using VHDL. Our objective was to then simulate the register set, as well as test it.
We then were required to implement a 32-bit Program Counter (PC) using VHDL, and also simulate and test it.

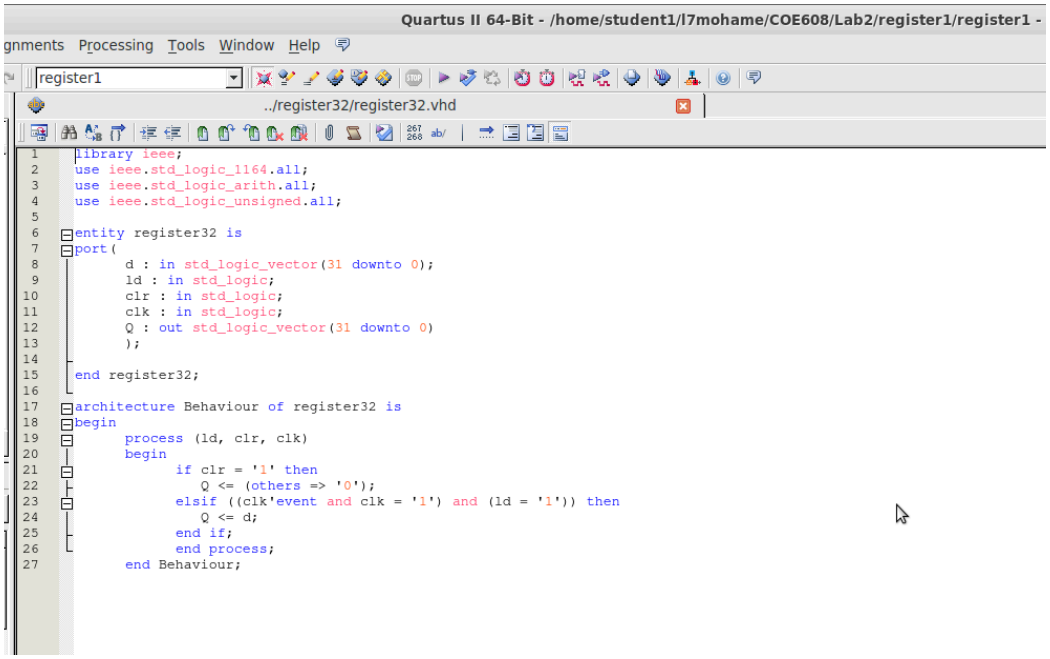
2. Experiment Details

1-bit Register:



- This is the VHDL code I used to implement the 1-bit register.

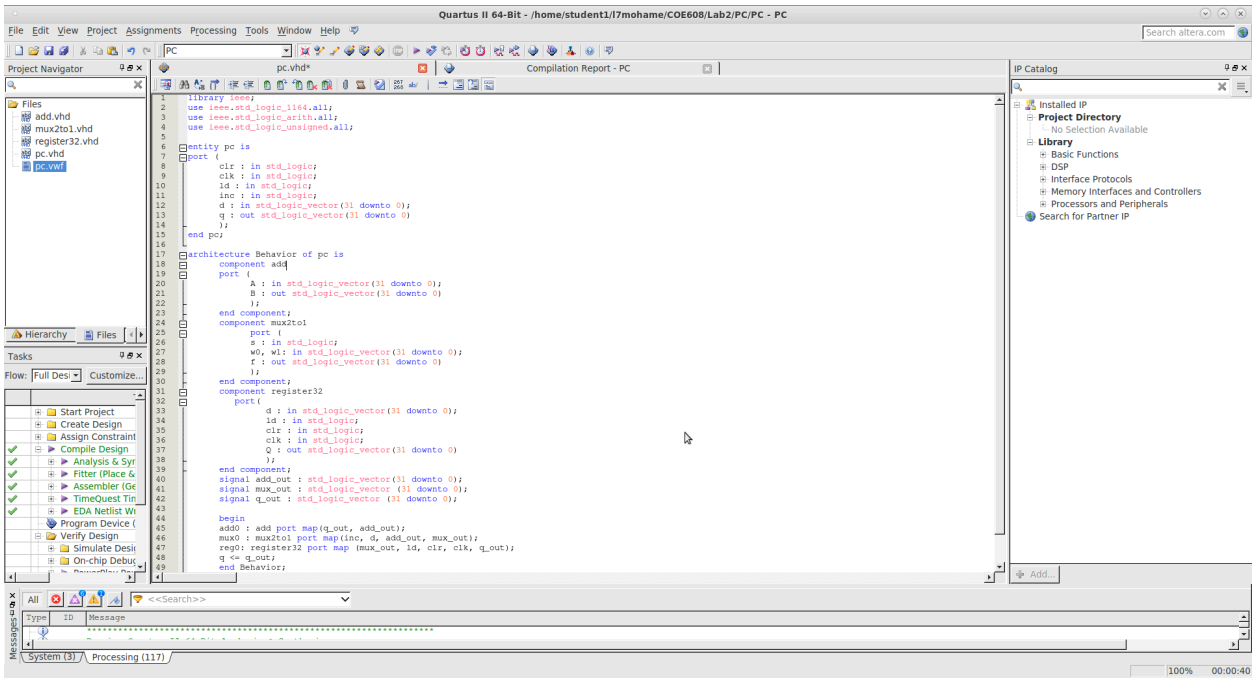
32-bit register:




```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity register32 is
7  port (
8        d : in std_logic_vector(31 downto 0);
9        ld : in std_logic;
10       clr : in std_logic;
11       clk : in std_logic;
12       Q : out std_logic_vector(31 downto 0)
13       );
14
15 end register32;
16
17 architecture Behaviour of register32 is
18 begin
19       process (ld, clr, clk)
20       begin
21             if clr = '1' then
22                Q <= (others => '0');
23             elsif ((clk'event and clk = '1') and (ld = '1')) then
24                Q <= d;
25             end if;
26             end process;
27       end Behaviour;
```

- This is the VHDL code that I used to implement the 32-bit register

Program Counter (PC):

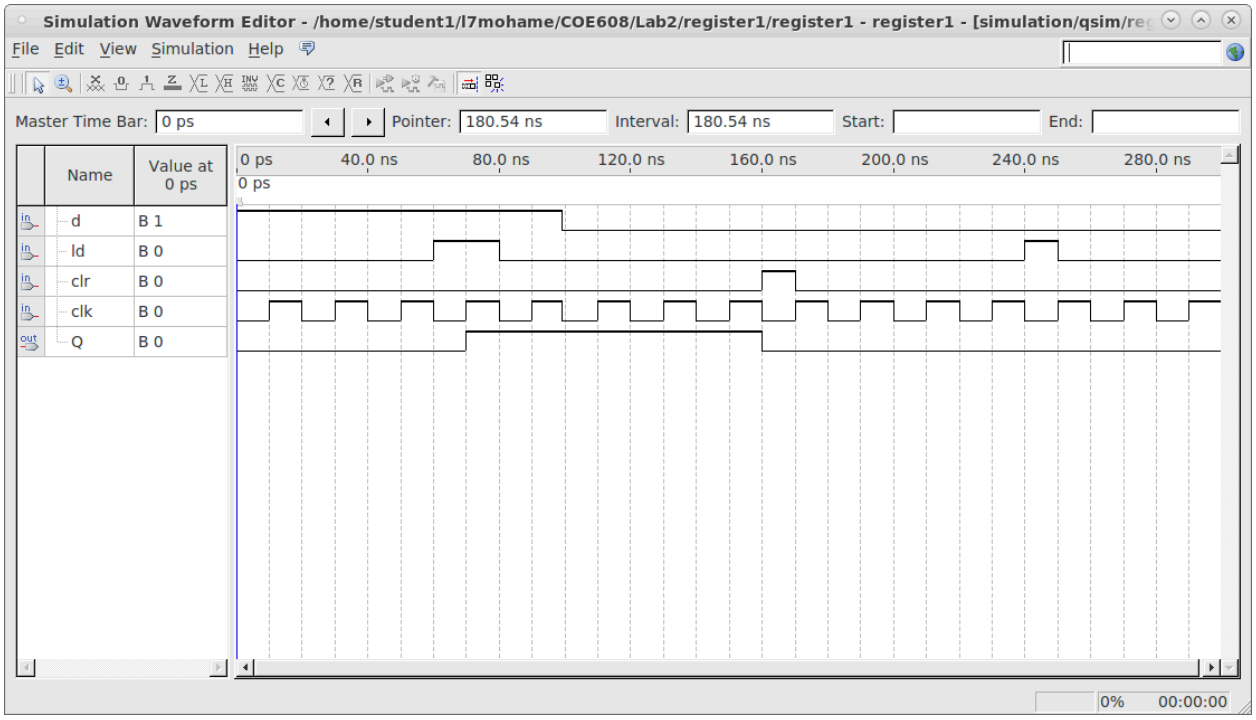



```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity pc is
7  port (
8        clr : in std_logic;
9        clk : in std_logic;
10       ld : in std_logic;
11       inc : in std_logic;
12       d : in std_logic_vector(31 downto 0);
13       q : out std_logic_vector(31 downto 0)
14       );
15 end pc;
16
17 architecture Behavior of pc is
18       component add
19       port (
20             A : in std_logic_vector(31 downto 0);
21             B : out std_logic_vector(31 downto 0)
22             );
23       end component;
24       component mux2to1
25             port (
26             s : in std_logic;
27             w0, w1: in std_logic_vector(31 downto 0);
28             f : out std_logic_vector(31 downto 0)
29             );
30       end component;
31       component register32
32          port(
33                d : in std_logic_vector(31 downto 0);
34                ld : in std_logic;
35                clr : in std_logic;
36                clk : in std_logic;
37                Q : out std_logic_vector(31 downto 0)
38                );
39       end component;
40       signal add_out : std_logic_vector(31 downto 0);
41       signal mux_out : std_logic_vector (31 downto 0);
42       signal q_out : std_logic_vector (31 downto 0);
43
44       begin
45       add0 : add port map(q_out, add_out);
46       mux0 : mux2to1 port map(inc, d, add_out, mux_out);
47       reg0: register32 port map (mux_out, ld, clr, clk, q_out);
48       q <= q_out;
49       end Behavior;
```

- This is the VHDL code that I used to implement the PC register.

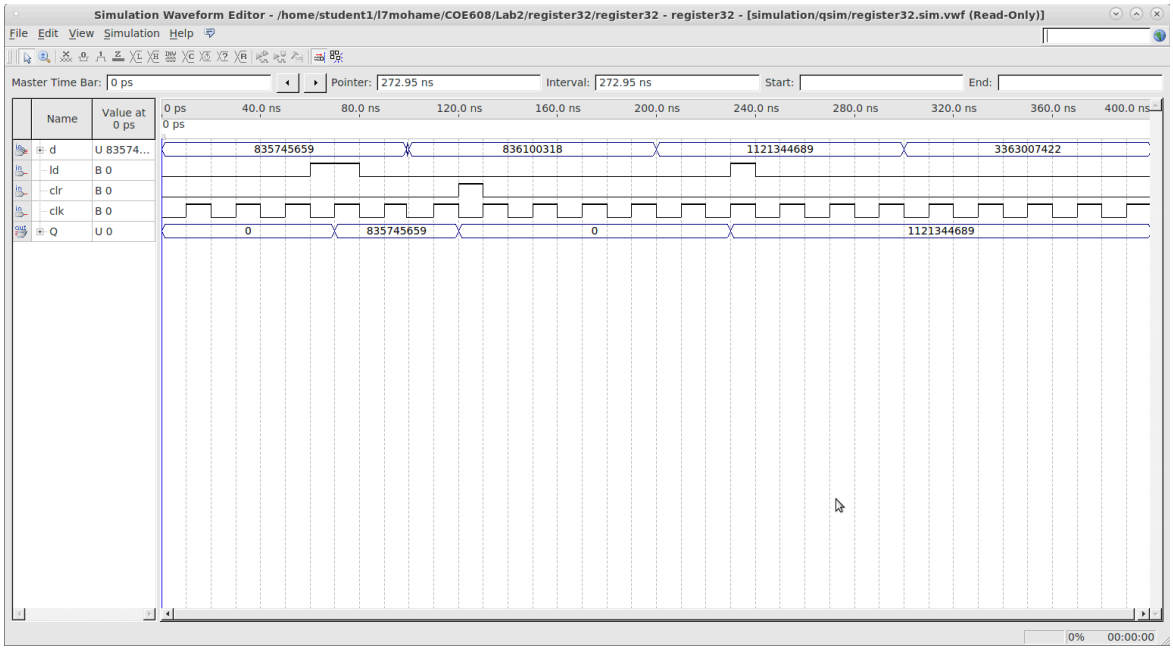
3. Results:

Register 1:



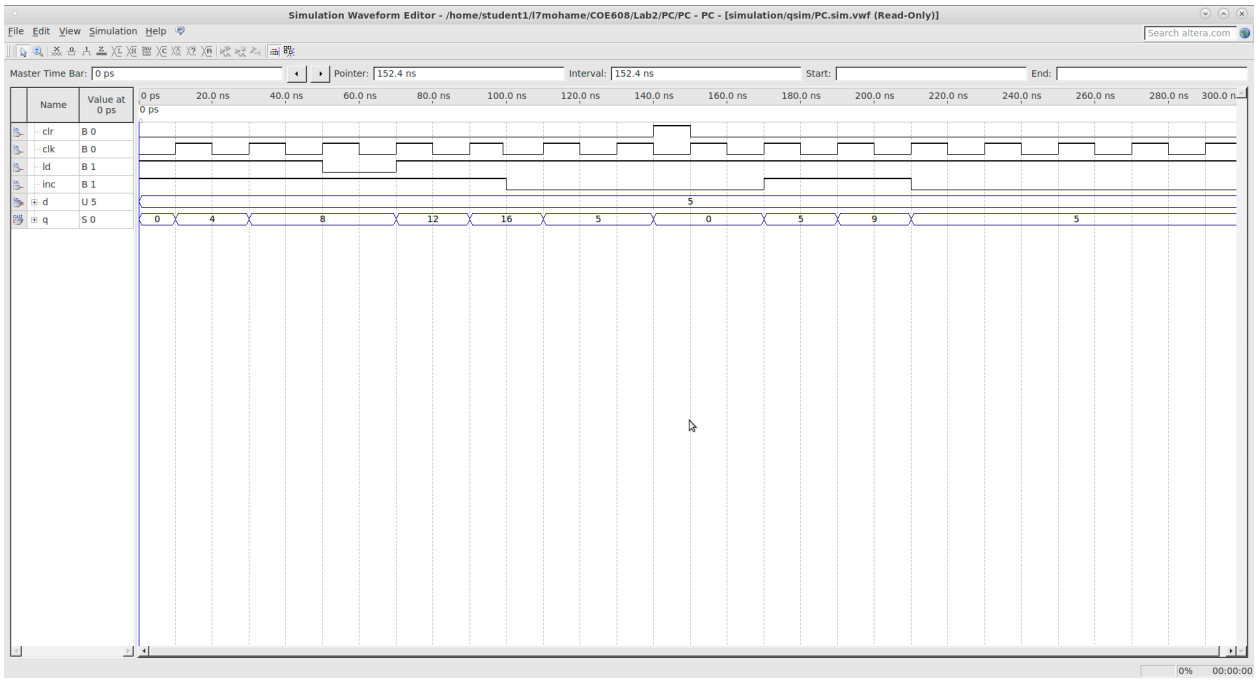
- This is the result after implementation and simulation of the 1-bit register.

Register 32:



- This is the result of the implementation and simulation of the 32-bit register.

PC:



- This is the result of the implementation and simulation of the PC register.

4. Discussion:

1-bit Register:

- As shown in the waveform, the result was as expected because the value that we loaded into the input data, ‘d’, was loaded into the register and assigned as the output after the load signal, ‘ld’, was equal to ‘1’ and ‘clr’ was 0.

32-bit Register:

- As can be seen from the waveform, the result was as expected because the value that we loaded into the input data, ‘d’, was assigned to the output value ‘Q’ when ‘ld’ was ‘1’ and ‘clr’ was ‘0’

PC register:

- As can be seen from the waveform, the result was as expected, however there is one missing increment after the output ‘9’. But other than that, the output is incremented by 4, and when ‘clr’ is 1, the output is 0.