Ryerson University

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

Course Title:	**Computer Organization and Architecture**
Course Number:	**COE 608**
Semester/Year (e.g.F2016)	**W2024**

Instructor:	**Dr. Khalid A. Hafeez**

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

Submission Date	**Jan 31 2024**
Due Date:	Feb 1 2024

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Yang	**Nini**	**501137659**	**11**	Nini Yang

*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

Lab Objective

The objective of this lab was to simulate the Register and 32 Counter required in 32 bit CPUs.

Experiment Details

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity register1 is
port(
    d  : in std_logic;
    ld : in std_logic;
    clr : in std_logic;
    clk : in std_logic;
    Q : out std_logic
    );
    end register1;

    architecture Behavior of register1 is
    begin
       process (ld, clr, clk)
       begin
          if clr = '1' then
             Q <= '0';
          elsif ((clk'event and clk='1') and (ld = '1')) then
             Q <= d;
          end if;
       end process;
    end Behavior;
```

Figure 1: Code for Register1.vhd, 1 bit Register

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2to1 is
port ( s : in std_logic;
       w0,w1 : in std_logic_vector(31 downto 0);
       f : out std_logic_vector(31 downto 0));
end mux2to1;

architecture Behavior of mux2to1 is
begin
   with s select
         f <= w0 when '0',
         w1 when others;

end Behavior;
```

Figure 2: Code for mux2to1.vhd, 2 to 1 multiplexer

```
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 add is
7   port (A : in std_logic_vector(31 downto 0);
8       B : out std_logic_vector(31 downto 0)
9       );
10
11  end add;
12
13  architecture Behavior of add is
14  begin
15  B <= A + 4;
16  end Behavior;
17
```

Figure 3: Code for Add.vhd, Adder block

```
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  end register32;
15
16  architecture Behavior of register32 is
17  begin
18     process (ld,clr,clk)
19     begin
20        if clr = '1' then
21           Q <= (others => '0');
22        elsif ((clk'event and clk = '1' ) and (ld = '1' )) then
23           Q <= d;
24        end if;
25     end process;
26  end Behavior;
27
```

Figure 4: Code for register32.vhd, 32 bit register

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity pc is
port (
      clr : in std_logic;
      clk : in std_logic;
      ld : in std_logic;
      inc  : in std_logic;
      d  : in std_logic_vector(31 downto 0);
      q  : out std_logic_vector(31 downto 0)
      );
      end pc;

architecture Behavior of    pc is
   component add
      port (
         A : in std_logic_vector(31 downto 0);
         B : out std_logic_vector(31 downto 0)
      );
   end component;
   component mux2to1
      port (
         s     : in std_logic;
         w0,w1 : in std_logic_vector(31 downto 0);
         f     : out std_logic_vector(31 downto 0)
      );
   end component;
   component register32
      port (
         d     :  in std_logic_vector(31 downto 0);
         ld    :  in std_logic;
         clr   :  in std_logic;
         clk   :  in std_logic;
         Q     :  out std_logic_vector(31 downto 0)
      );
   end component;
   signal add_out : std_logic_vector(31 downto 0);
   signal mux_out : std_logic_vector(31 downto 0);
   signal q_out : std_logic_vector(31 downto 0);
   begin
      add0: add port map(q_out, add_out);
      mux0: mux2to1 port map (inc, d, add_out, mux_out);
      reg0: register32 port map (mux_out, ld, clr, clk, q_out);
      q <= q_out;
end Behavior;
```

Figure 5: Code for Pc.vhd, Program Counter

The program counter (PC) code combines the 32 bit register, 1 bit register, 2:1 multiplexer, and adder to form the program counter. They were incorporated as components since it reduces the number of lines of code compared to directly pasting code from each component. In addition, the code is cleaner and easier to read and debug.

Results

The resulting waveforms for each component is displayed below.

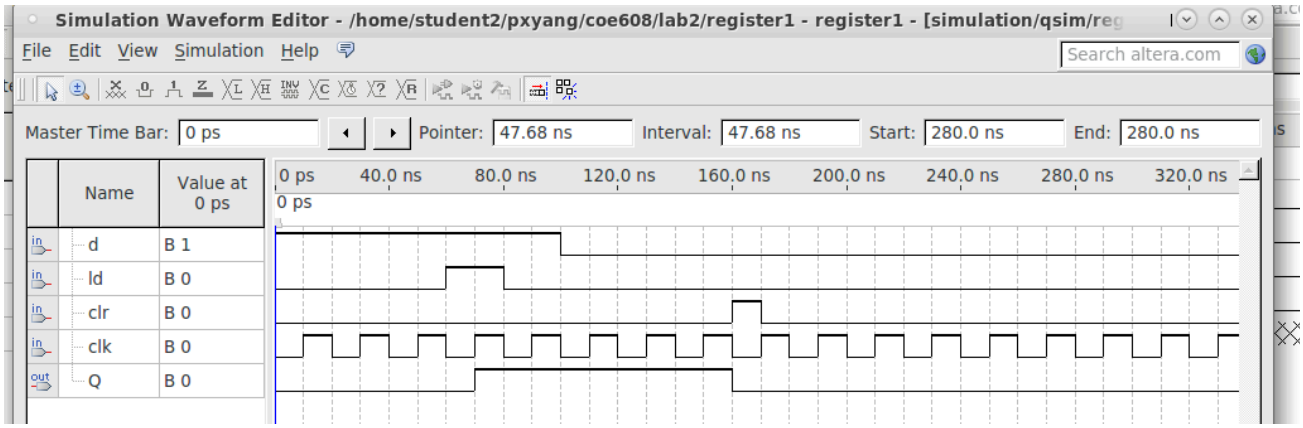

Figure 6: Waveform for Register1.vhd, 1 bit register

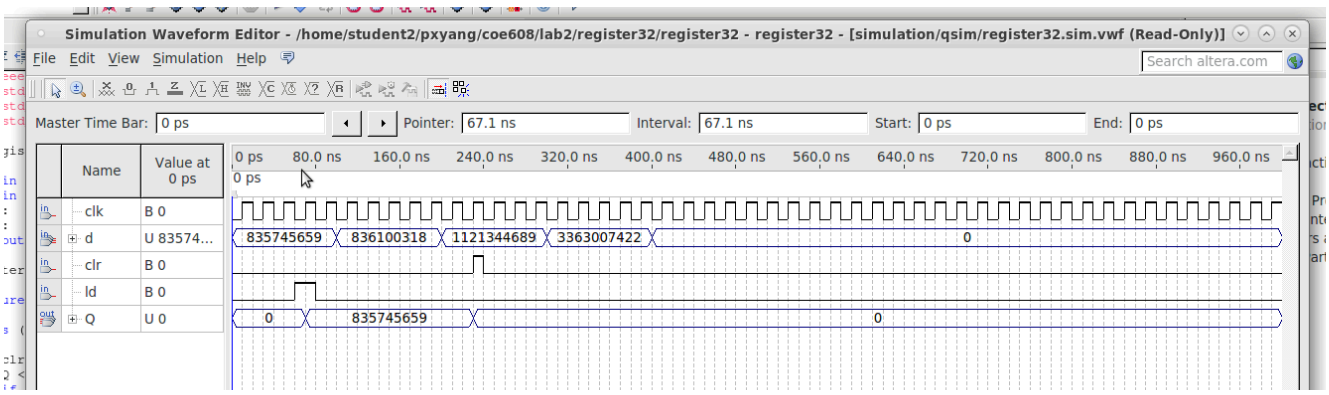

Figure 7: Waveform for Register32.vhd, 32 bit register

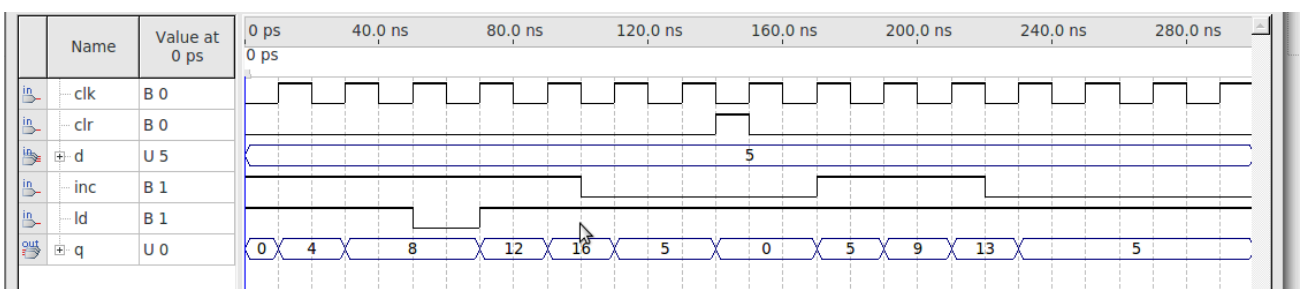

Figure 8: Waveform for Pc.vhd, Program Counter

Input d was used to determine PC values, ld input loaded PC with the input at d when clk was HIGH. Inc input incremented PC by 4 during execution, and was used to select either input d or pc + 4.

Discussion:

When compared with the expected waveforms taken from Lab2_Tutorial lab manual, the resulting waveforms have the same outputs given the same inputs. Therefore, a functioning and successful program counter was created. The final PC waveform displays a functioning 32 bit program counter, similar to those found in 32 bit CPUs.

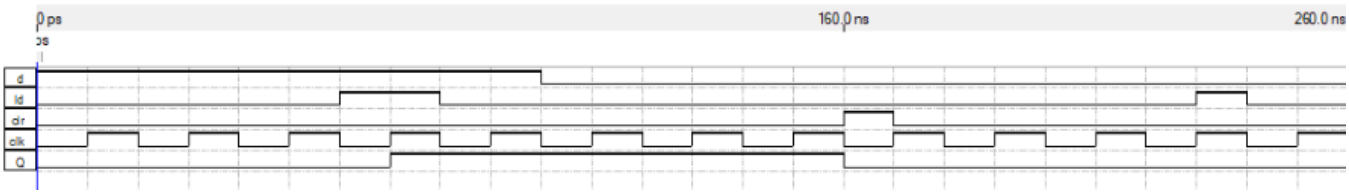

Figure 9: Expected waveform for the 1 bit Register

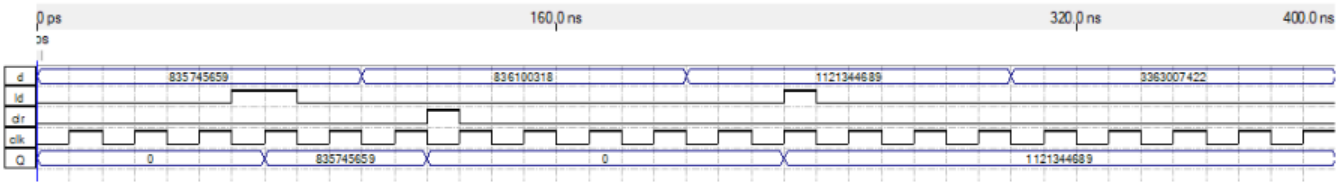

Figure 10: Expected waveform for the 32 bit Register

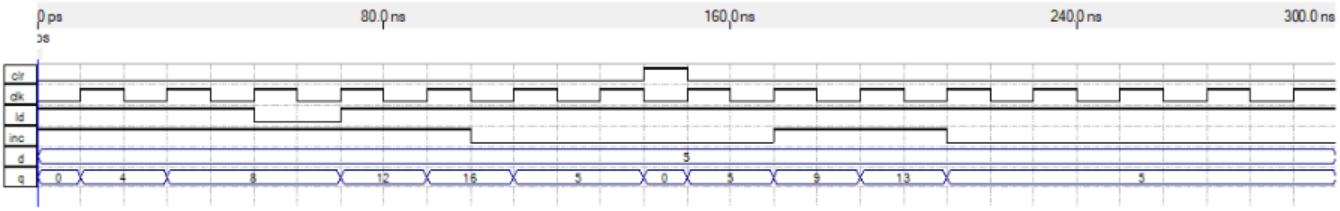

Figure 11: Expected waveform for the program counter

References:

- ➔ Ryerson University, “Lab2_Manual_W2021”, D2L
- ➔ Ryerson University, “Lab2_Tutorial”, D2L