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)	W2021

Instructor:	Nagi Mekheil

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

Submission Date:	2021-02-06
Due Date:	2021-02-06

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Ahmed	Sameh	500907041	052	

*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

TA: Lei Gao

Table of Contents

Introduction

The lab report *Program Counter and Register Set Design* is presented herein. The lab demo took place online on February 5, 2021. The lab was completed using Quartus-II software.

Objective

The objective of this lab is to create and test a 32 bit Program Counter and simulate it. This objective will be done by completing the smaller objectives of making a 32 bit register, adder and a 2 to 1 multiplexer.

Design and Implementation

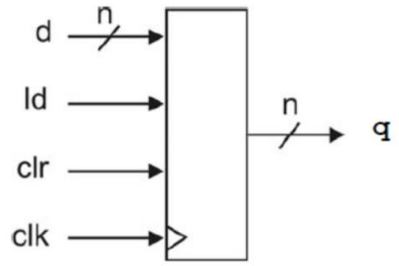
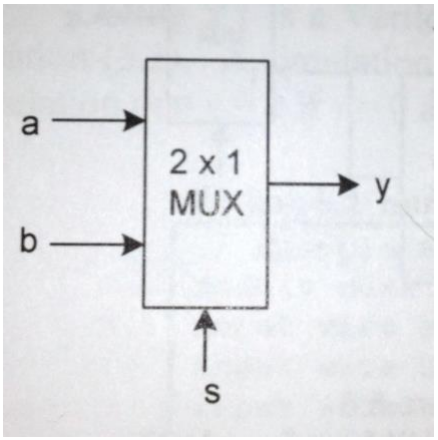


Figure 1: Block Diagram for n-bit Register

This image represents the block diagram for the registers to be created with any specified number of bits. In the lab both a 1-bit register will be created as well as a 32-bit register.



This image represents a 2 to 1 multiplexer which will be needed for the 32-bit PC.

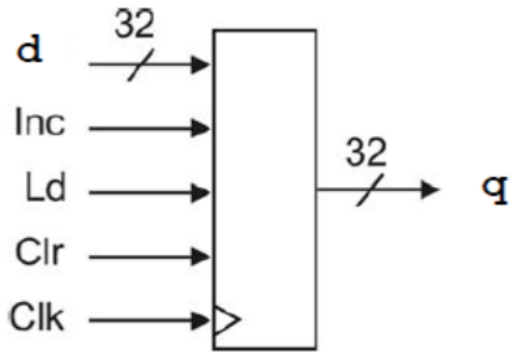


Figure 2: Program Counter

This image represents the diagram of a Program counter. The following image is the internal components of the Program counter.

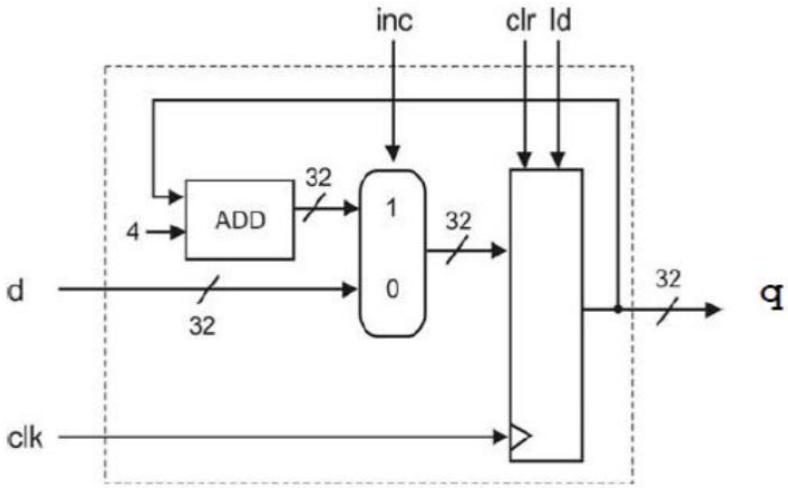


Figure 3: 32-bit Program Counter Internal Structure

Results and Conclusions

The following waveforms represent functional simulations.

The main inputs of the register include clock (clk), clear (clr), load/enable (ld) signals and an n-bit data (d). The n-bit output is denoted by (q).

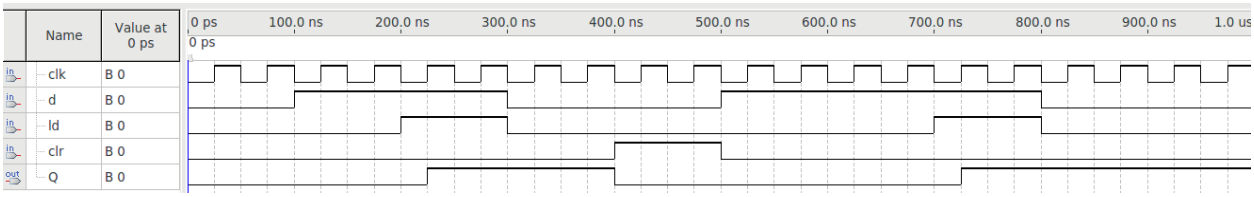


Figure 4: Waveform for 1-bit register

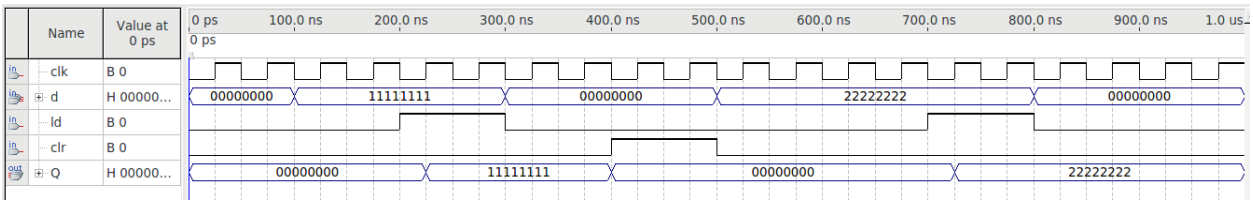


Figure 5: Waveform for 32-bit register

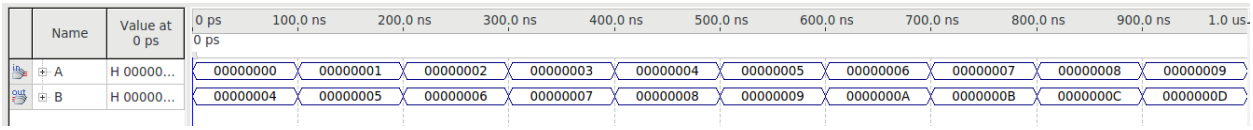


Figure 6: Waveform for add block

A and B represent the registers in the computer

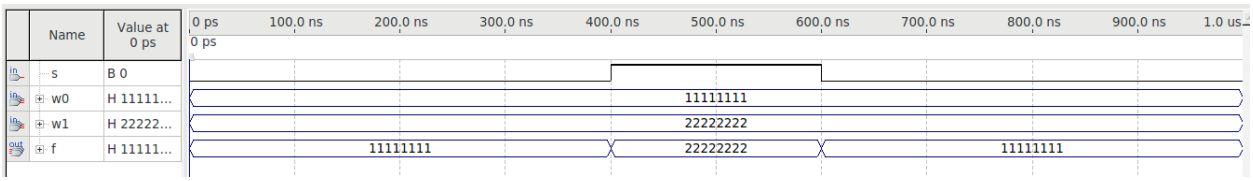


Figure 7: waveform for 2 to 1 multiplexer

S is a switch, f is the function and w0 is one input and w1 is another input

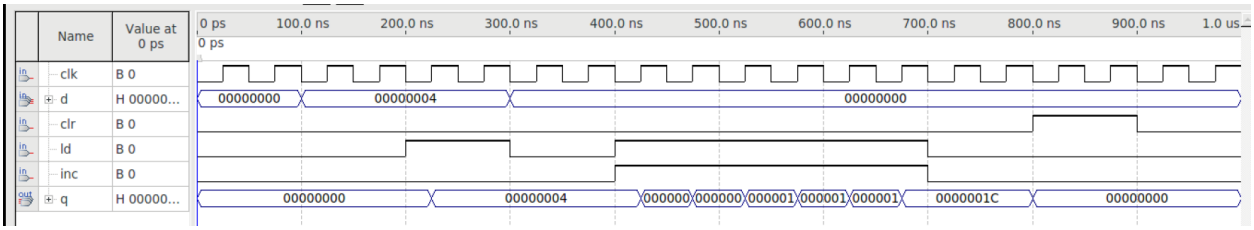


Figure 8: waveform for 32-bit Program Counter.

The inc represents the increment by 4 as written in the vhdl code for the add block.

Note : the unreadable output q from 425ns – 475ns =00000008, 475ns – 525ns – 0000000C, 525 ns – 575 ns = 00000010, 575ns – 625ns = 00000014, 625ns – 675ns = 00000018.

In conclusion the 32-bit program counter was made and simulated. This was done using the schematic and creating each component and connecting them. The individual components created that were necessary for this program counter was the add block, the 32-bit register and the 2 to 1 multiplexer. The program counter took these components and connected them. All of the components were made through VHDL.

Reference

1. Department of Electrical, Computer & Biomedical Engineering. (2021, Winter). *COE608 Lab 2– Program Counter and Register Set Design*. Toronto, Ontario: Ryerson University.

2. Department of Electrical, Computer & Biomedical Engineering. (2021, Winter). *COE608 Lab 2 Tutorial*. Toronto, Ontario: Ryerson University.

Appendix

The following image contains the VHDL code used to complete the objective simulation.

```
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 9: VHDL code for 1-bit Register

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

entity register32 is
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 register32;

architecture Behavior of register32 is
begin
process(ld, clr ,clk)
begin
if clr = '1' then
Q <= (others => '0');
elsif ((clk'event and clk = '1') and (ld ='1')) then
Q <= d;
end if;
end process;
end Behavior;
```

Figure 10: VHDL code for 32-bit Register

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

entity add is
port(
A   : in std_logic_vector(31 DOWNTO 0);
B : out std_logic_vector(31 DOWNTO 0)
);
end add;

architecture Behavior of add is
begin
B <= A + 4;
end Behavior;
```

Figure 11: VHDL code for add block

```
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 12: VHDL code for 2-to-1 multiplexer

```vhdl
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);
  --signal ld    : std_logic := '1';

  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 13: 32-bit Program counter VHDL code