Lab 5

Ly Nguyen

February 18, 2022

Programmable Devices

Introduction

In this lab, we implemented a one-digit-counter to a seven-segment display using components, casting/converting of types, signals, and knowledge and work from previous labs.

Theory

A one-digit-counter counts from 0 to 9 and repeats starting at 0 again.

A clock-divider lowers the frequency of a clock to make a display, like the seven-segment display, readable between clock changes.

A seven-segment decoder takes an input and decodes it into the segments of the seven segment display.

Equipment and Method

Programs:

Quartus Prime Version 20.1.1 Build 720 11/11/2020 SJ Elite Edition

ModelSim SE-64 2021.2

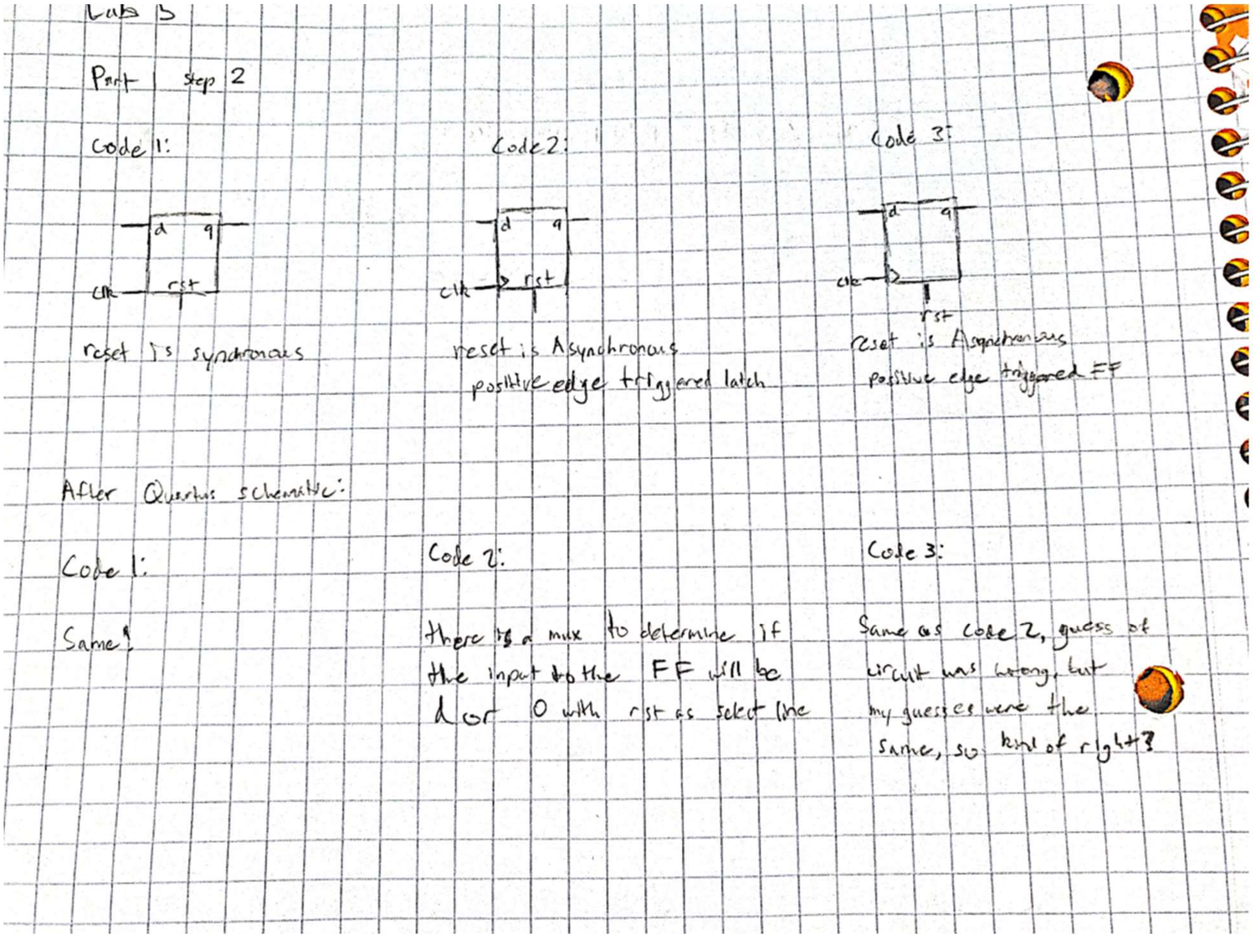
Procedure:

To start, the first portion of our lab was an observation and prediction section. Moving on to the second portion of the lab, we took the package code from the lab manual (Appendix A), and put it into Quartus as a VHDL file where the component names are changed to match the file names of previous labs (i.e. oneDigitCounter → counter because the one digit counter file from lab 1 was saved as counter.vhd). This the components from the mypack.vhd, we implemented the components from the package using intermediate signals to have the correct types for the inputs and outputs in the architecture. For this, I had four intermediate signals, three of which were for type conversion.

For the implementation of components, we had std_logic in for the clock divider, intermediate signal type std_logic for the output of the clock divider and the input of the one-digit-counter, and type converted the integer output of the counter from integer, to signed, to a 4-bit std_logic_vector that was the input for the seven segment display, which output a 7-bit std_logic_vector.

Results

Part 1:



Lab 5

Part 1 step 2

Code 1:

d
q
clk
rst

reset is synchronous

Code 2:

d
q
clk
rst

reset is Asynchronous
positive edge triggered latch

Code 3:

d
q
clk
rst

reset is Asynchronous
positive edge triggered FF

After Quartus schematic:

Code 1:

Same!

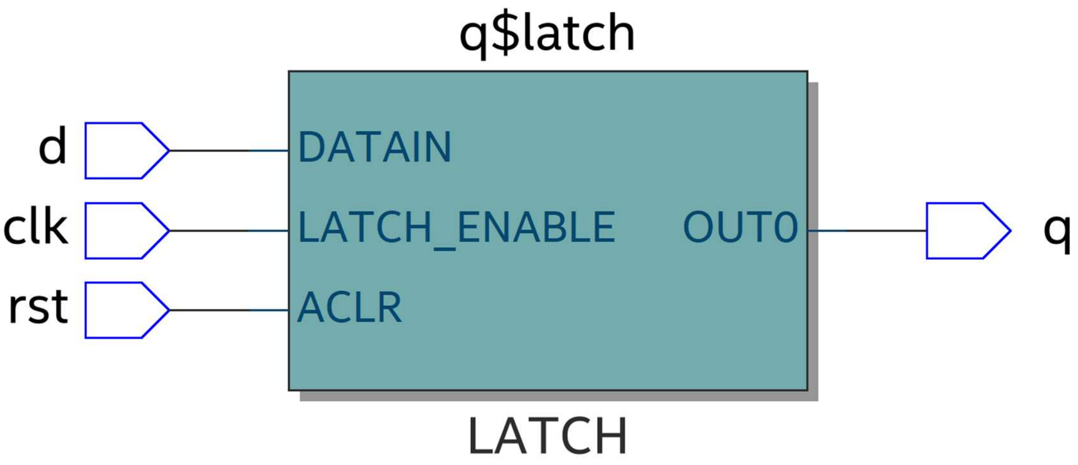
Code 2:

there is a mux to determine if
the input to the FF will be
d or 0 with rst as select line

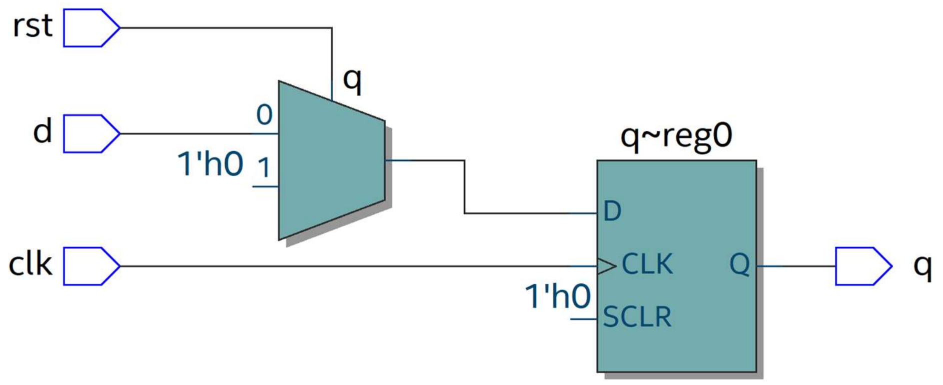
Code 3:

Same as code 2, guess of
circuit was wrong, but
my guesses were the
same, so kind of right?

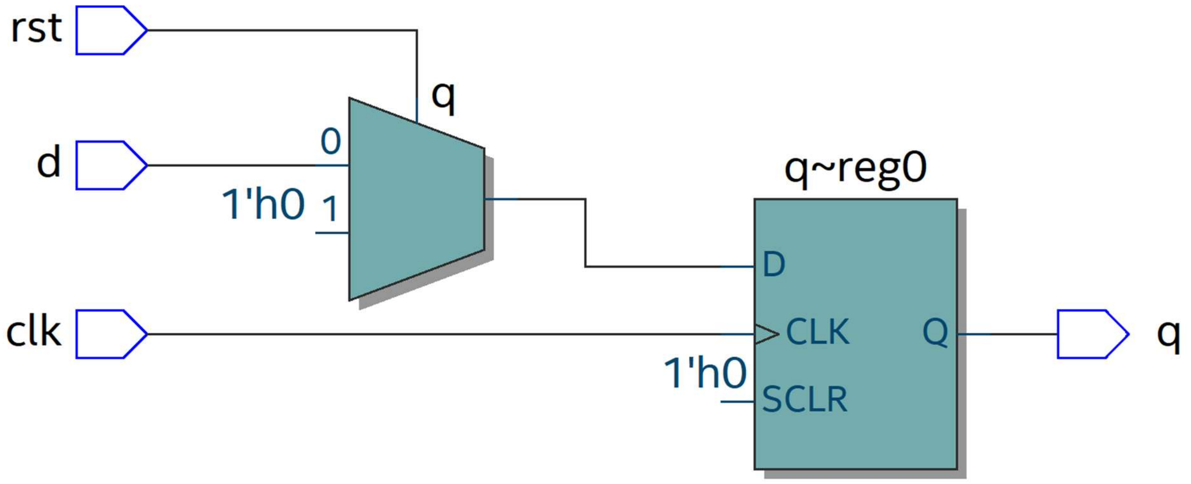
Code 1:



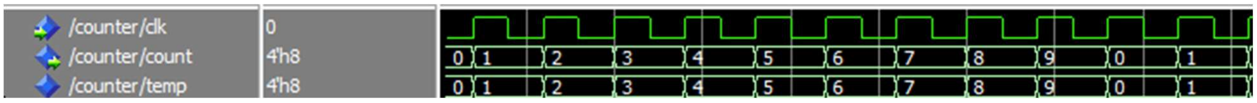
Code 2:



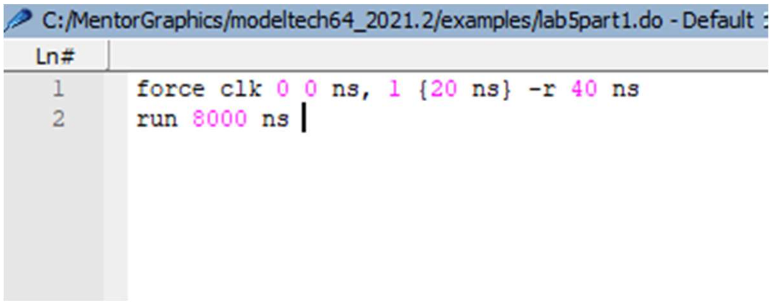
Code 3:



Part 1 Step 3:

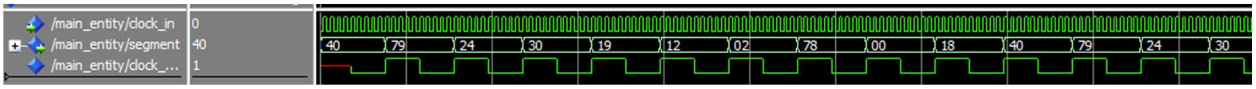


```
/counter/clk     0
/counter/count   4'h8    0 1 2 3 4 5 6 7 8 9 0 1
/counter/temp    4'h8    0 1 2 3 4 5 6 7 8 9 0 1
```


C:/MentorGraphics/modeltech64_2021.2/examples/lab5part1.do - Default
Ln#

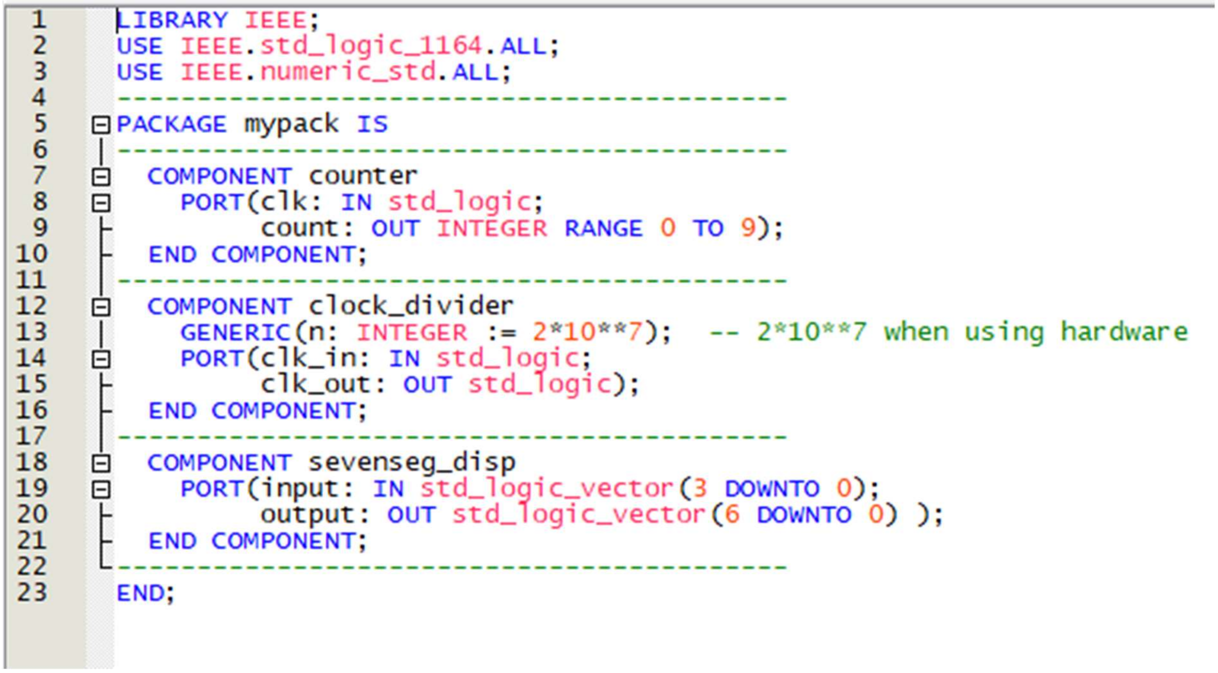
```
force clk 0 0 ns, 1 {20 ns} -r 40 ns
run 8000 ns
```

Part 2:



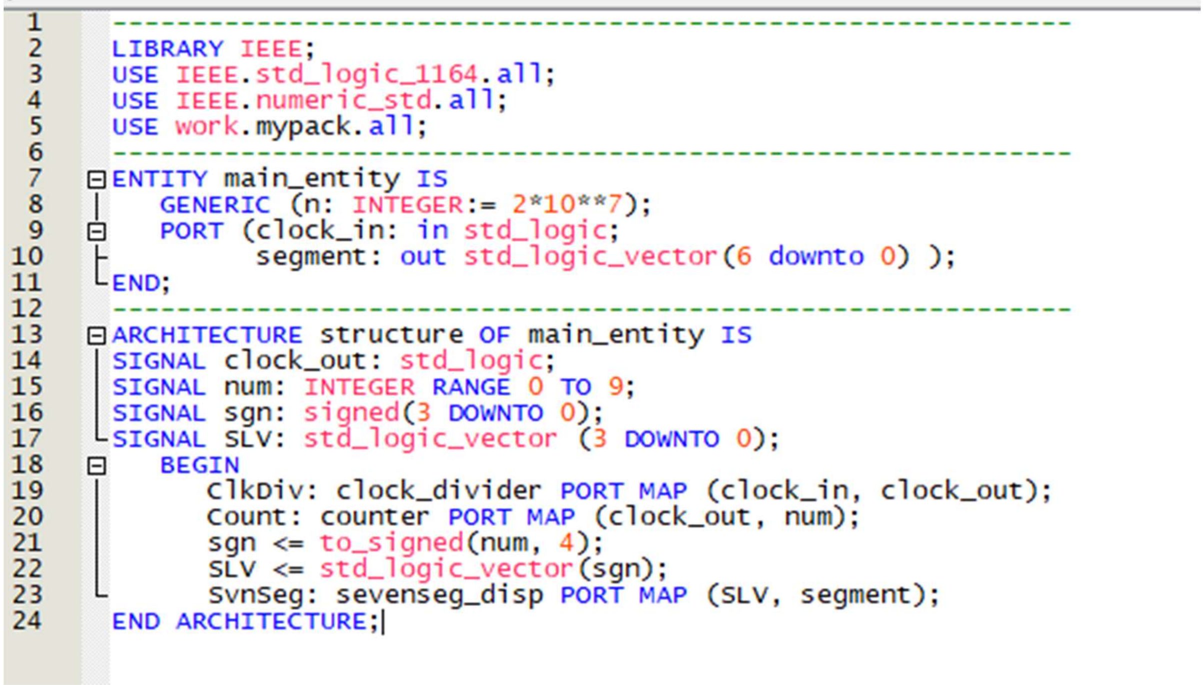
```
radix hexadecimal
force clock_in 0 0 ns, 1 {20 ns} -r 40 ns
run 8000 ns
```

mypack.vhd:



```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
-----------------------------------------
PACKAGE mypack IS
-----------------------------------------
  COMPONENT counter
    PORT(clk: IN std_logic;
         count: OUT INTEGER RANGE 0 TO 9);
  END COMPONENT;
-----------------------------------------
  COMPONENT clock_divider
    GENERIC(n: INTEGER := 2*10**7);   -- 2*10**7 when using hardware
    PORT(clk_in: IN std_logic;
         clk_out: OUT std_logic);
  END COMPONENT;
-----------------------------------------
  COMPONENT sevenseg_disp
    PORT(input: IN std_logic_vector(3 DOWNTO 0);
         output: OUT std_logic_vector(6 DOWNTO 0) );
  END COMPONENT;
-----------------------------------------
END;
```

Main_entity.vhd:

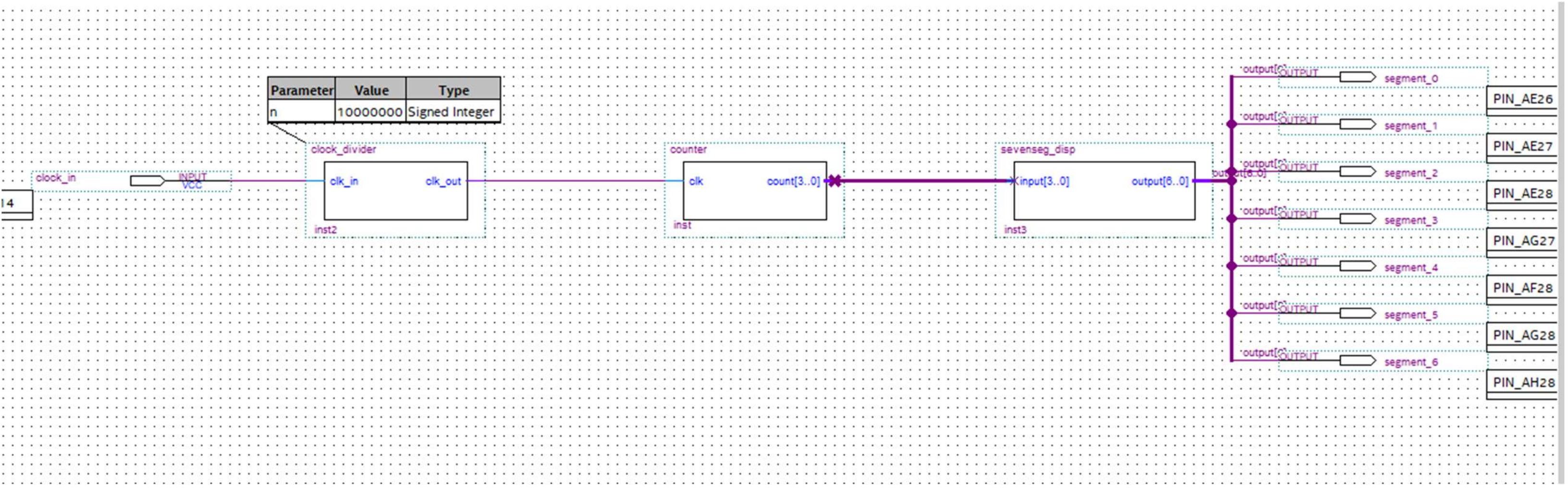


```vhdl
------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.numeric_std.all;
USE work.mypack.all;
------------------------------------------------------------
ENTITY main_entity IS
   GENERIC (n: INTEGER:= 2*10**7);
   PORT (clock_in: in std_logic;
         segment: out std_logic_vector(6 downto 0) );
END;
------------------------------------------------------------
ARCHITECTURE structure OF main_entity IS
SIGNAL clock_out: std_logic;
SIGNAL num: INTEGER RANGE 0 TO 9;
SIGNAL sgn: signed(3 DOWNTO 0);
SIGNAL SLV: std_logic_vector (3 DOWNTO 0);
   BEGIN
      ClkDiv: clock_divider PORT MAP (clock_in, clock_out);
      Count: counter PORT MAP (clock_out, num);
      sgn <= to_signed(num, 4);
      SLV <= std_logic_vector(sgn);
      SvnSeg: sevenseg_disp PORT MAP (SLV, segment);
END ARCHITECTURE;
```

Part 3:

Demonstrated

Part 4:



Parameter	Value	Type
n	10000000	Signed Integer

Discussion and Conclusion

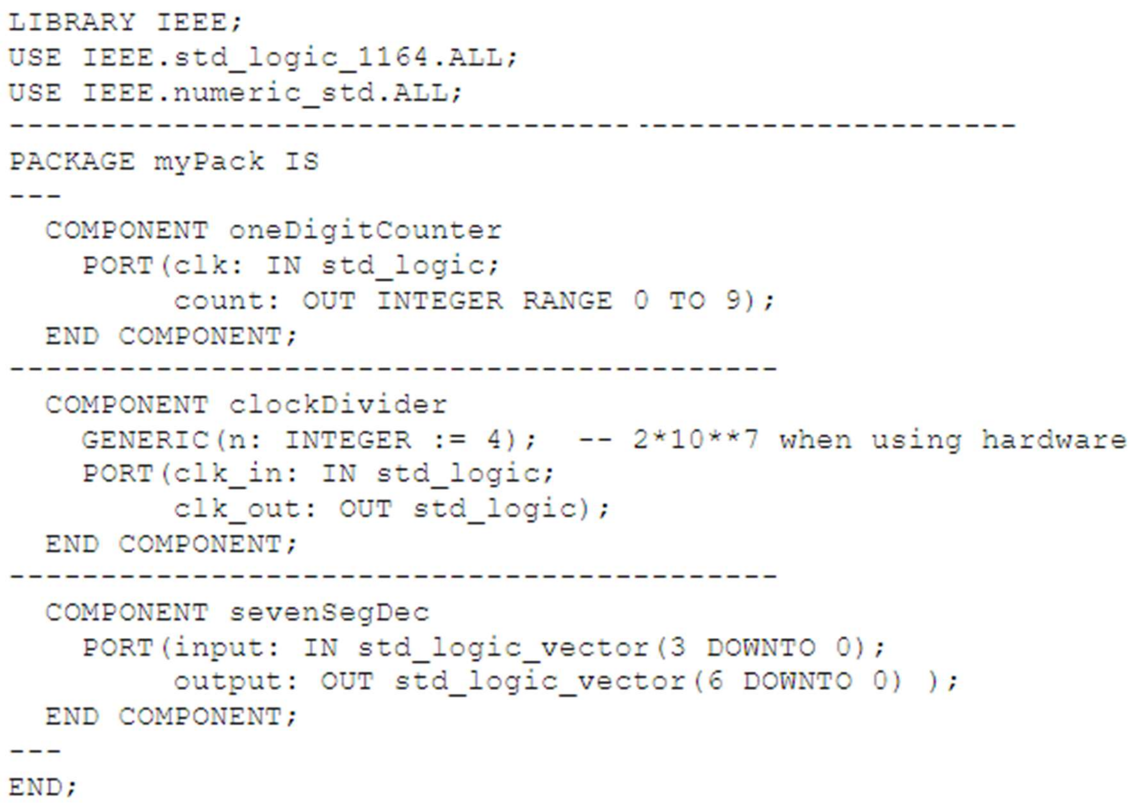
For part 1, after observing the code, we predicted that code 1 was synchronous, and codes 2 and 3 were asynchronous, which were correct. Our graphical elements were not the same as the complied results though. This is because of the fact that we restricted our predictions with the use of only one flip flop and did not consider the use of a multiplexer.

References

Lecture slides and lab manual

Appendices

Appendix A: MyPack code form the Lab Manual



```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
--------------------------------------------------------
PACKAGE myPack IS
---
  COMPONENT oneDigitCounter
    PORT(clk: IN std_logic;
         count: OUT INTEGER RANGE 0 TO 9);
  END COMPONENT;
-------------------------------------------
  COMPONENT clockDivider
    GENERIC(n: INTEGER := 4);  -- 2*10**7 when using hardware
    PORT(clk_in: IN std_logic;
         clk_out: OUT std_logic);
  END COMPONENT;
-------------------------------------------
  COMPONENT sevenSegDec
    PORT(input: IN std_logic_vector(3 DOWNTO 0);
         output: OUT std_logic_vector(6 DOWNTO 0) );
  END COMPONENT;
---
END;
```