SCHOOL OF INFORMATION, COMPUTER AND COMMUNICATION TECHNOLOGY
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY

LAB REPORT

EES 370 DIGITAL CIRCUIT LABORATORY

Lab 09 Hardware Description Language I

By

Mr. Makhapoom Euaumpon	**ID: 6222780833**
Mr. Paphana Yiwsiw	**ID: 6222780379**

Group No. 12 Section 2

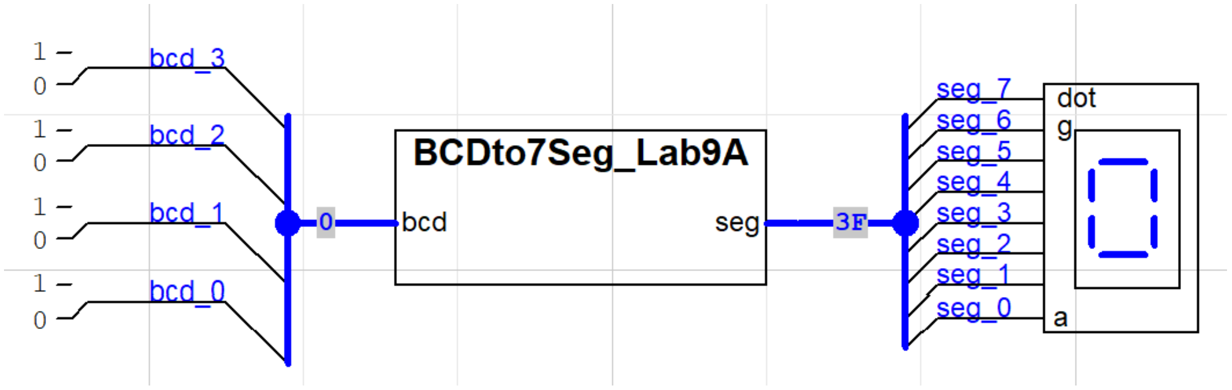
Date: 05 Apr 2021, Time: 13:00-16:00

Objectives

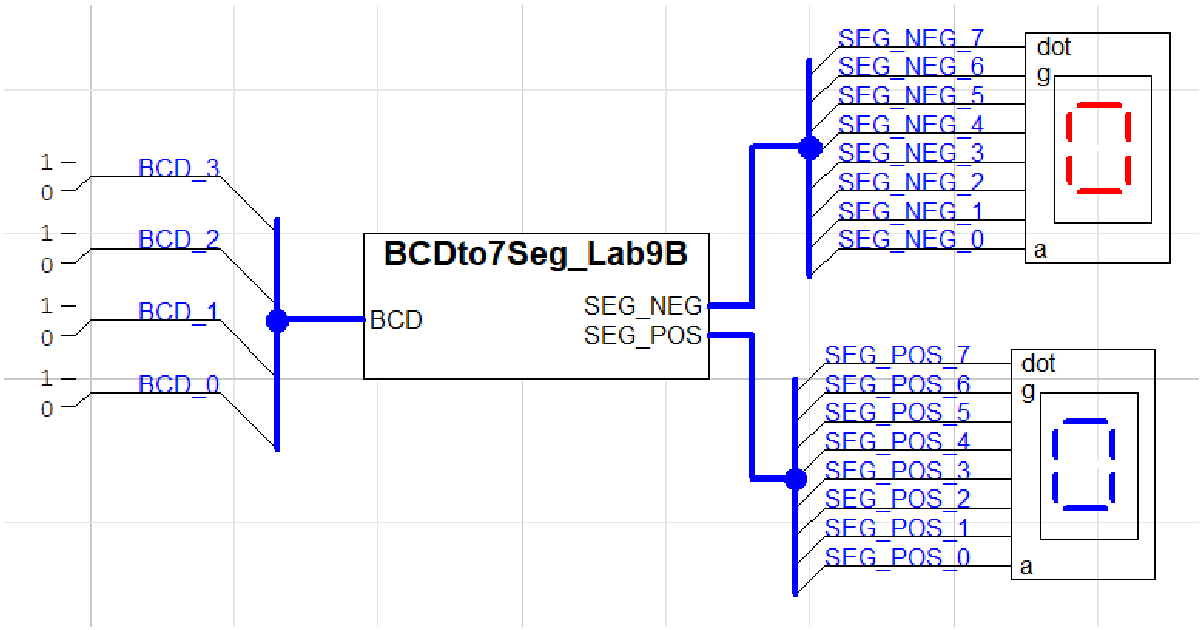
1. To introduce a programmable logic device (PLD), also called the field programmable gate array (FPGA)
2. To experience a hardware description language (HDL) programming to operate the FPGA.

Lab Result

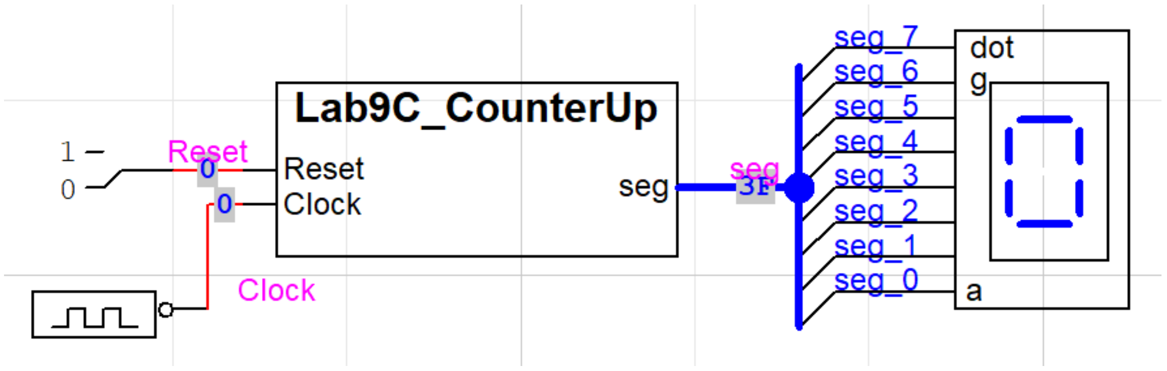
Part A



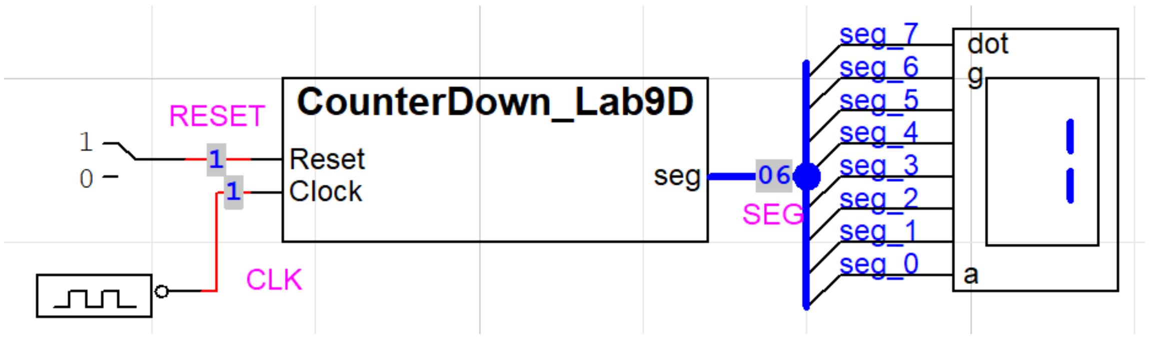
Part B



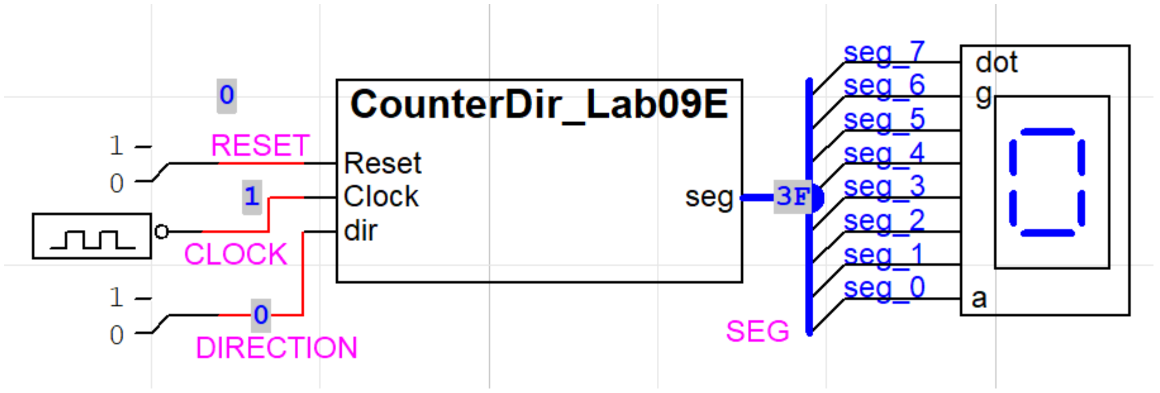
Part C



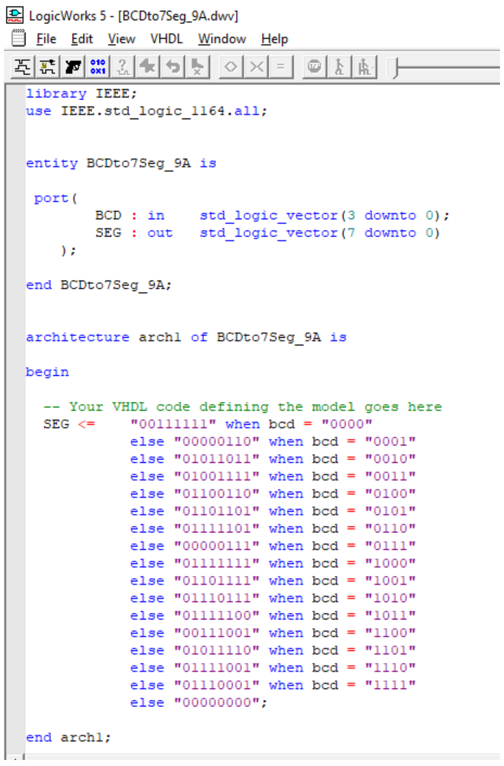
Part D



Part E

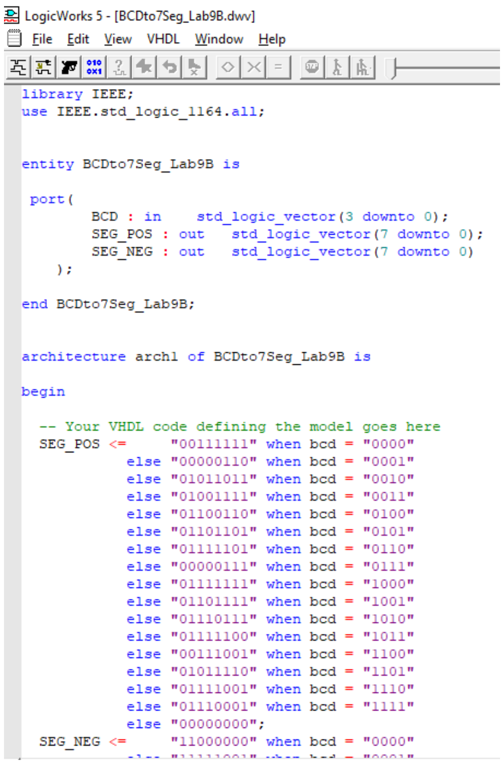


VHDL Code Part A

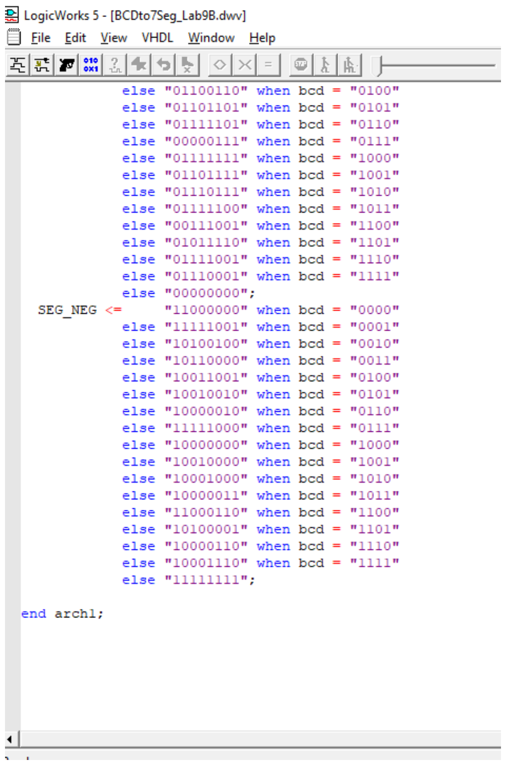


```
library IEEE;
use IEEE.std_logic_1164.all;


entity BCDto7Seg_9A is

 port(
        BCD : in    std_logic_vector(3 downto 0);
        SEG : out   std_logic_vector(7 downto 0)
    );

end BCDto7Seg_9A;


architecture arch1 of BCDto7Seg_9A is

begin

  -- Your VHDL code defining the model goes here
  SEG <=    "00111111" when bcd = "0000"
            else "00000110" when bcd = "0001"
            else "01011011" when bcd = "0010"
            else "01001111" when bcd = "0011"
            else "01100110" when bcd = "0100"
            else "01101101" when bcd = "0101"
            else "01111101" when bcd = "0110"
            else "00000111" when bcd = "0111"
            else "01111111" when bcd = "1000"
            else "01101111" when bcd = "1001"
            else "01110111" when bcd = "1010"
            else "01111100" when bcd = "1011"
            else "00111001" when bcd = "1100"
            else "01011110" when bcd = "1101"
            else "01111001" when bcd = "1110"
            else "01110001" when bcd = "1111"
            else "00000000";

end arch1;
```

VHDL Code Part B

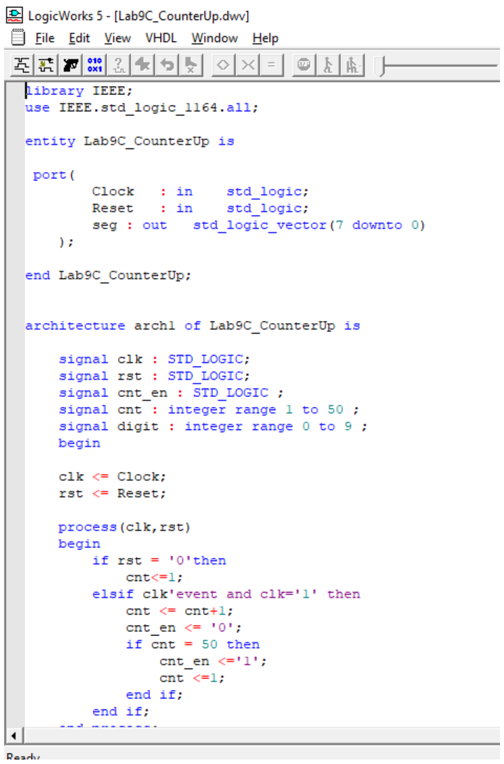

LogicWorks 5 - [BCDto7Seg_Lab9B.dwv]
File Edit View VHDL Window Help

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;


entity BCDto7Seg_Lab9B is

 port(
        BCD : in    std_logic_vector(3 downto 0);
        SEG_POS : out   std_logic_vector(7 downto 0);
        SEG_NEG : out   std_logic_vector(7 downto 0)
    );

end BCDto7Seg_Lab9B;


architecture arch1 of BCDto7Seg_Lab9B is

begin

  -- Your VHDL code defining the model goes here
  SEG_POS <=      "00111111" when bcd = "0000"
              else "00000110" when bcd = "0001"
              else "01011011" when bcd = "0010"
              else "01001111" when bcd = "0011"
              else "01100110" when bcd = "0100"
              else "01101101" when bcd = "0101"
              else "01111101" when bcd = "0110"
              else "00000111" when bcd = "0111"
              else "01111111" when bcd = "1000"
              else "01101111" when bcd = "1001"
              else "01110111" when bcd = "1010"
              else "01111100" when bcd = "1011"
              else "00111001" when bcd = "1100"
              else "01011110" when bcd = "1101"
              else "01111001" when bcd = "1110"
              else "01110001" when bcd = "1111"
              else "00000000";
  SEG_NEG <=      "11000000" when bcd = "0000"
```

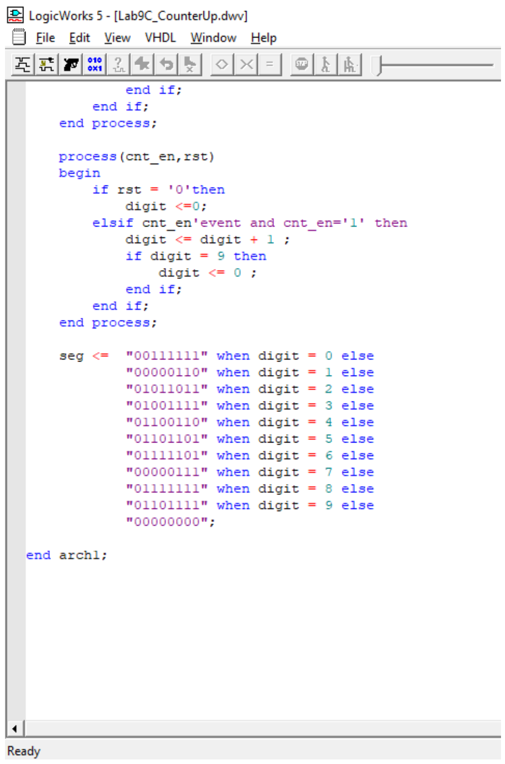

LogicWorks 5 - [BCDto7Seg_Lab9B.dwv]
File Edit View VHDL Window Help

```vhdl
              else "01100110" when bcd = "0100"
              else "01101101" when bcd = "0101"
              else "01111101" when bcd = "0110"
              else "00000111" when bcd = "0111"
              else "01111111" when bcd = "1000"
              else "01101111" when bcd = "1001"
              else "01110111" when bcd = "1010"
              else "01111100" when bcd = "1011"
              else "00111001" when bcd = "1100"
              else "01011110" when bcd = "1101"
              else "01111001" when bcd = "1110"
              else "01110001" when bcd = "1111"
              else "00000000";
  SEG_NEG <=      "11000000" when bcd = "0000"
              else "11111001" when bcd = "0001"
              else "10100100" when bcd = "0010"
              else "10110000" when bcd = "0011"
              else "10011001" when bcd = "0100"
              else "10010010" when bcd = "0101"
              else "10000010" when bcd = "0110"
              else "11111000" when bcd = "0111"
              else "10000000" when bcd = "1000"
              else "10010000" when bcd = "1001"
              else "10001000" when bcd = "1010"
              else "10000011" when bcd = "1011"
              else "11000110" when bcd = "1100"
              else "10100001" when bcd = "1101"
              else "10000110" when bcd = "1110"
              else "10001110" when bcd = "1111"
              else "11111111";

end arch1;
```

VHDL Code Part C

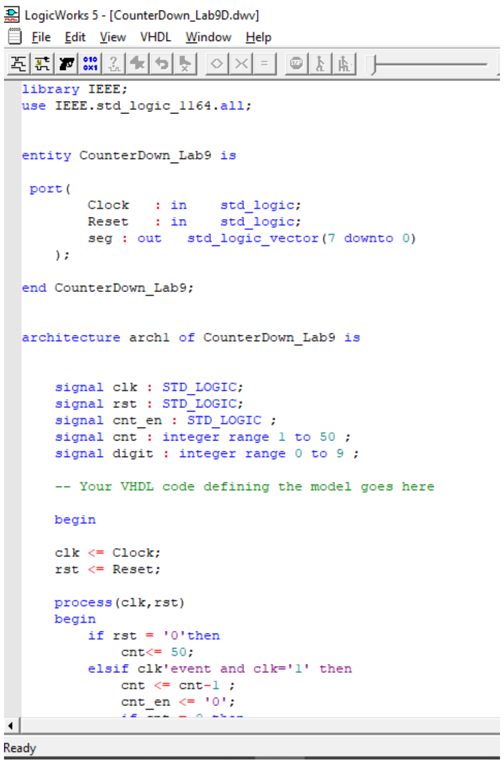

LogicWorks 5 - [Lab9C_CounterUp.dwv]
File Edit View VHDL Window Help

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity Lab9C_CounterUp is

 port(
        Clock   : in    std_logic;
        Reset   : in    std_logic;
        seg : out   std_logic_vector(7 downto 0)
    );

end Lab9C_CounterUp;


architecture arch1 of Lab9C_CounterUp is

    signal clk : STD_LOGIC;
    signal rst : STD_LOGIC;
    signal cnt_en : STD_LOGIC ;
    signal cnt : integer range 1 to 50 ;
    signal digit : integer range 0 to 9 ;
    begin

    clk <= Clock;
    rst <= Reset;

    process(clk,rst)
    begin
        if rst = '0'then
            cnt<=1;
        elsif clk'event and clk='1' then
            cnt <= cnt+1;
            cnt_en <= '0';
            if cnt = 50 then
                cnt_en <='1';
                cnt <=1;
            end if;
        end if;
```

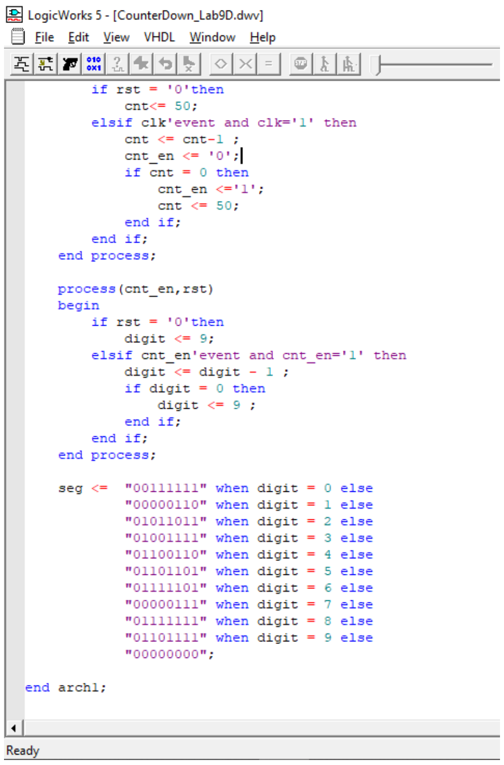

LogicWorks 5 - [Lab9C_CounterUp.dwv]
File Edit View VHDL Window Help
Ready

```vhdl
            end if;
        end if;
    end process;

    process(cnt_en,rst)
    begin
        if rst = '0'then
            digit <=0;
        elsif cnt_en'event and cnt_en='1' then
            digit <= digit + 1 ;
            if digit = 9 then
                digit <= 0 ;
            end if;
        end if;
    end process;

    seg <=  "00111111" when digit = 0 else
            "00000110" when digit = 1 else
            "01011011" when digit = 2 else
            "01001111" when digit = 3 else
            "01100110" when digit = 4 else
            "01101101" when digit = 5 else
            "01111101" when digit = 6 else
            "00000111" when digit = 7 else
            "01111111" when digit = 8 else
            "01101111" when digit = 9 else
            "00000000";

end arch1;
```

VHDL Code Part D

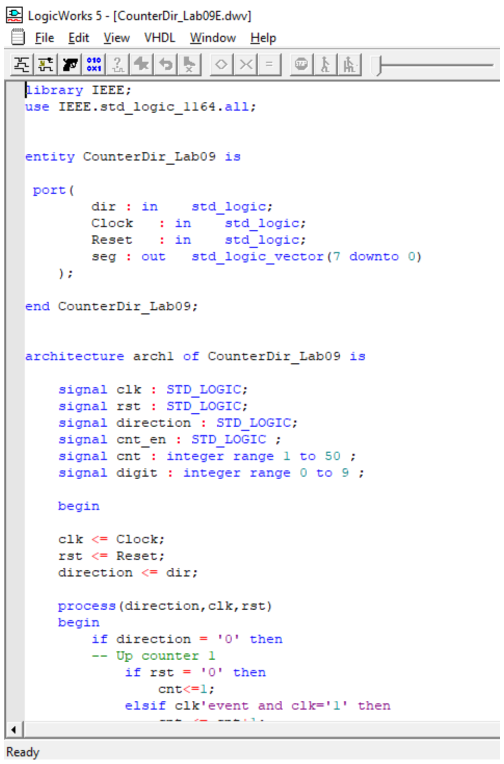

LogicWorks 5 - [CounterDown_Lab9D.dwv]
File Edit View VHDL Window Help
Ready

```
library IEEE;
use IEEE.std_logic_1164.all;


entity CounterDown_Lab9 is

 port(
        Clock   : in    std_logic;
        Reset   : in    std_logic;
        seg : out   std_logic_vector(7 downto 0)
    );

end CounterDown_Lab9;


architecture arch1 of CounterDown_Lab9 is


    signal clk : STD_LOGIC;
    signal rst : STD_LOGIC;
    signal cnt_en : STD_LOGIC ;
    signal cnt : integer range 1 to 50 ;
    signal digit : integer range 0 to 9 ;

    -- Your VHDL code defining the model goes here

    begin

    clk <= Clock;
    rst <= Reset;

    process(clk,rst)
    begin
        if rst = '0'then
            cnt<= 50;
        elsif clk'event and clk='1' then
            cnt <= cnt-1 ;
            cnt_en <= '0';
```

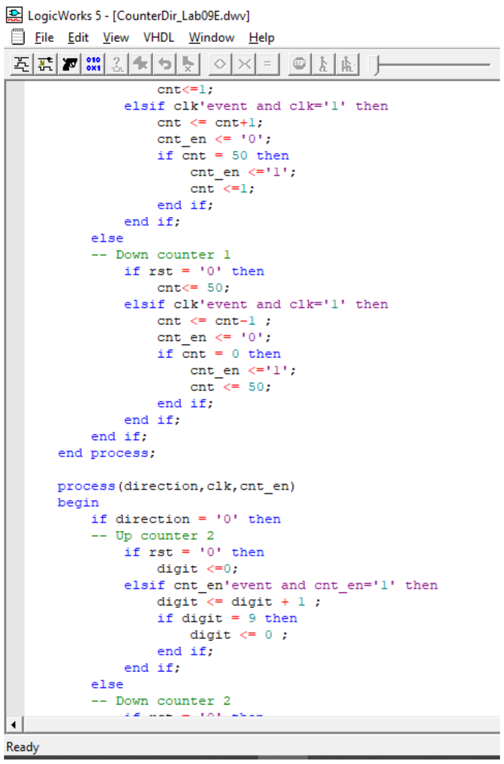

LogicWorks 5 - [CounterDown_Lab9D.dwv]
File Edit View VHDL Window Help
Ready

```
        if rst = '0'then
            cnt<= 50;
        elsif clk'event and clk='1' then
            cnt <= cnt-1 ;
            cnt_en <= '0';
            if cnt = 0 then
                cnt_en <='1';
                cnt <= 50;
            end if;
        end if;
    end process;

    process(cnt_en,rst)
    begin
        if rst = '0'then
            digit <= 9;
        elsif cnt_en'event and cnt_en='1' then
            digit <= digit - 1 ;
            if digit = 0 then
                digit <= 9 ;
            end if;
        end if;
    end process;

    seg <=  "00111111" when digit = 0 else
            "00000110" when digit = 1 else
            "01011011" when digit = 2 else
            "01001111" when digit = 3 else
            "01100110" when digit = 4 else
            "01101101" when digit = 5 else
            "01111101" when digit = 6 else
            "00000111" when digit = 7 else
            "01111111" when digit = 8 else
            "01101111" when digit = 9 else
            "00000000";

end arch1;
```

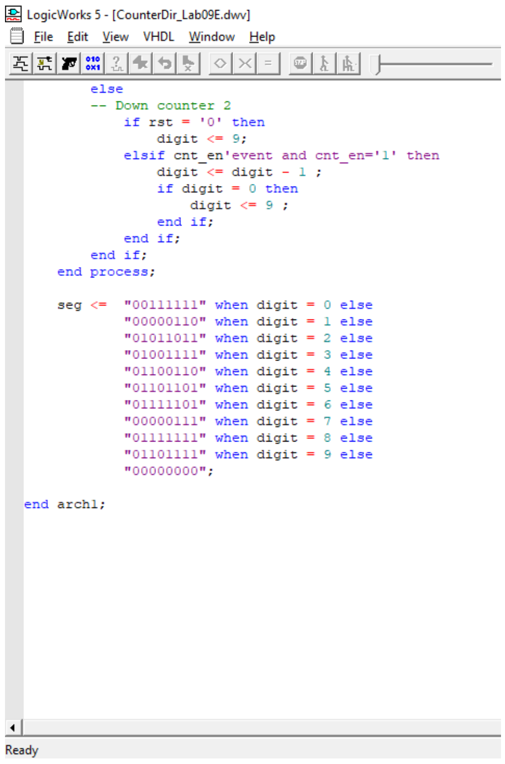
VHDL Code Part E


LogicWorks 5 - [CounterDir_Lab09E.dwv]
File Edit View VHDL Window Help
Ready

```
library IEEE;
use IEEE.std_logic_1164.all;


entity CounterDir_Lab09 is

 port(
        dir : in    std_logic;
        Clock   : in    std_logic;
        Reset   : in    std_logic;
        seg : out   std_logic_vector(7 downto 0)
    );

end CounterDir_Lab09;


architecture arch1 of CounterDir_Lab09 is

    signal clk : STD_LOGIC;
    signal rst : STD_LOGIC;
    signal direction : STD_LOGIC;
    signal cnt_en : STD_LOGIC ;
    signal cnt : integer range 1 to 50 ;
    signal digit : integer range 0 to 9 ;

    begin

    clk <= Clock;
    rst <= Reset;
    direction <= dir;

    process(direction,clk,rst)
    begin
        if direction = '0' then
        -- Up counter 1
            if rst = '0' then
                cnt<=1;
            elsif clk'event and clk='1' then
```


LogicWorks 5 - [CounterDir_Lab09E.dwv]
File Edit View VHDL Window Help
Ready

```
                cnt<=1;
            elsif clk'event and clk='1' then
                cnt <= cnt+1;
                cnt_en <= '0';
                if cnt = 50 then
                    cnt_en <='1';
                    cnt <=1;
                end if;
            end if;
        else
        -- Down counter 1
            if rst = '0' then
                cnt<= 50;
            elsif clk'event and clk='1' then
                cnt <= cnt-1 ;
                cnt_en <= '0';
                if cnt = 0 then
                    cnt_en <='1';
                    cnt <= 50;
                end if;
            end if;
        end if;
    end process;

    process(direction,clk,cnt_en)
    begin
        if direction = '0' then
        -- Up counter 2
            if rst = '0' then
                digit <=0;
            elsif cnt_en'event and cnt_en='1' then
                digit <= digit + 1 ;
                if digit = 9 then
                    digit <= 0 ;
                end if;
            end if;
        else
        -- Down counter 2
```


LogicWorks 5 - [CounterDir_Lab09E.dwv]
File Edit View VHDL Window Help
Ready

```vhdl
        else
        -- Down counter 2
            if rst = '0' then
                digit <= 9;
            elsif cnt_en'event and cnt_en='1' then
                digit <= digit - 1 ;
                if digit = 0 then
                    digit <= 9 ;
                end if;
            end if;
        end if;
    end process;

    seg <=  "00111111" when digit = 0 else
            "00000110" when digit = 1 else
            "01011011" when digit = 2 else
            "01001111" when digit = 3 else
            "01100110" when digit = 4 else
            "01101101" when digit = 5 else
            "01111101" when digit = 6 else
            "00000111" when digit = 7 else
            "01111111" when digit = 8 else
            "01101111" when digit = 9 else
            "00000000";

end arch1;
```

Discussion

In Part A, we learned more about how to take multiple bits input to generate multiple bits output in the PLD. To deliver output or take input in multiple bits, you have to use input as vectors and name that signal only once. The wire in the circuit of that signal will called as bus. As normally, in this part, we will create the model of BCD-to-7-Segment display in the same step as in previous lab. But, in this time, input and output signal will be in the vector form. The input is BCD vector, set the left bit number to 3 and right bit number to 0. This will make BCD input vector as 4 bits input. Same goes to the output which is SEG vector, set the left bit number to 7 and right bit number to 0. This will make SEG output vector as 8 bits output. Then, type in entire VHDL code as shown in the lab result part A and compile it to check that it doesn't contain any error. The VHDL code looks like this:

```
SEG <= “00111111” when BCD = “0000” else ... else “00000000”;
```

This section of code means that SEG output will be “00111111” when BCD input is “0000” otherwise it will be “00000000”. It can be interpreted as if it is number 0 the output to the 7-segment display is 00111111 which light up the corresponding segments in the display. The reason why the SEG output has 8 bits because the first bit represent the dot pin in the 7 segments display which we don't want to use it and we put the output of the first bit of SEG to 0. If the BCD is not 0000, it will continue to do in the else block and check condition in the when block. After all comparison and checking, if it doesn't match any, it will produce 00000000 as the output of SEG instead. It means that the 7-segment display won't show anything. This last else block will act like the default value if it doesn't match any of the when condition block above. To use the circuit, we have to break down the bus into single separated line. We can do it by right-click on the bus pin and select breakout. Then, set the spacing of between each individual pin and placed down on the circuit. After all line connected to switch and connect the 7-segment display, the final circuit result of part A is shown here (also shown in the lab result part).

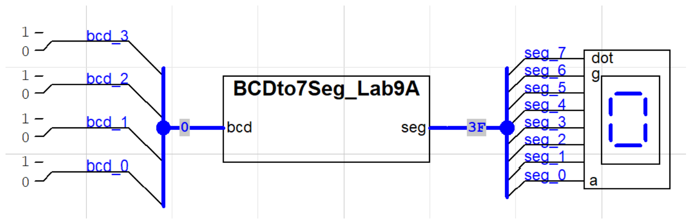


Figure 1: BCD-to-7-Segment Display Circuit of lab part A

In lab part B, it is quite similar to the part A but with 2 output vectors, SEG_POS for using with the active-high 7-Segment display and SEG_NEG for using with the active-low 7-Segment display (or in the LogicWorks, 7-Disp Inv). In this part, we can setup the model similar to the part A but add more output vector which is SEG_NEG and set left bit number to 7 and right bit number to 0 for it to have same 8-bit output with the SEG_POS. The VHDL code is similar to the part A which we can reused with the SEG_POS part. For the SEG_NEG part, we just change the output number of SEG_POS from 0 to 1 and vice versa to have it functioning with the active-low 7-Segment display. Then, connect the bus signal and the both type of 7-Segment displays. Thus, the final circuit of part B is shown below (also shown in lab result part).

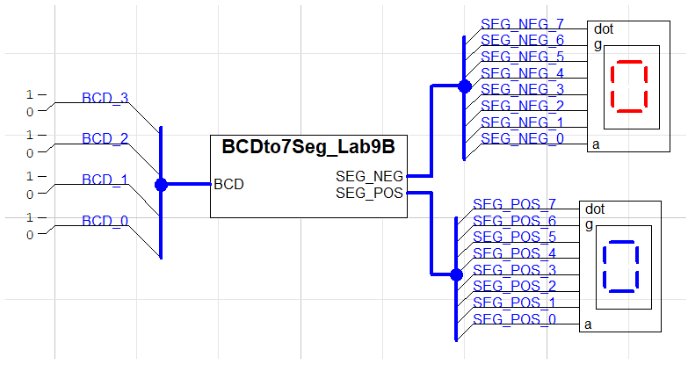


Figure 2: BCD to 7-Segment display (active-high and active-low) circuit of part B

In part C, we created the up-counter circuit by using the VHDL. This circuit will only count up starting from 0 to 9 and loop indefinitely every 50 rising edges of the clock signal and able to reset the counter to go back to 0. The reset signal is also an active-low signal. To setup the counter-up model, we have 2 one-bit inputs: CLK to represent clock and RST to represent reset signal. We have 1 output vector which is SEG for connect with the active-high 7-Segment display. SEG has 8-bit as same as the previous parts. Then, we type in the given code in the lab manual to the model and connect all necessary component to the circuit and run it to check the result. Noted that, when simulating the circuit, you should set the clock speed on the slide bar above to the max to get the quicker result. Now, we take a look at the code, we can see that there are consist of many local variables. The local variable cannot be observed in the circuit and not connect to any port/pin of the model, it only served as the parameter in evaluating the output of the model. In this program, it contains two types of local variable: STD_LOGIC is basic one-bit binary value and integer is whole number which you have to specify the range of possible value. For example, signal digit: integer range 0 to 9. It means digit is an integer value from 0 to 9. Down in the VHDL code, there is a process block which have the structure as shown down below:

```
...
process(clk, rst)
begin
     if rst='0' then ...
     elsif clk'event and clk='1' then ...
     end if;
end process;
...
```

The process block is used to run the command in sequential basis, not concurrently basis as normal VHDL code do. The code in sequential basis that can be implemented such as the nested if command. The process block in this case, received the clk and rst value into the process for it to be "processing". The first if block in the process means that if the rst signal is 0 which means the reset button is pressed, this process block will do as the then statement block said. Otherwise, it will look at the elsif block, basically equivalent to the else if block in many programming language. It will execute the then statement block if the clock signal (clk) is currently in rising edge. After all coding, here's final circuit.

In addition to part C, the given code can be separated into 2 processes block. The first process block is about converting clock signal from 50 cycle raw clock signal into 1 cycle cnt_en signal to be used as parameter in second process block. The second process block is about counting digit up from 0 to 9. Therefore, the final circuit of this part is shown down below (also shown in lab result part).

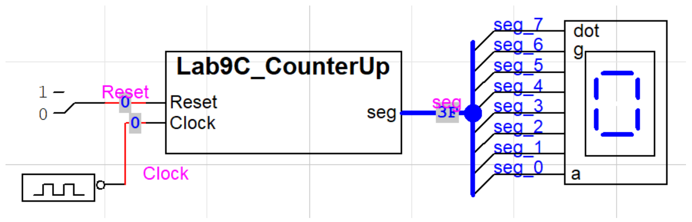


Figure 3: Counter-up circuit with VHDL of part C.

In part D, it is similar to the part C. But, instead of counting up from 0 to 9, it will be counting down from 9 to 0 and loop indefinitely until the reset button is pressed. The process in creating the circuit and model is exactly the same as part C. The only thing changed in this part is the VHDL code of counter down. In order to do counter down, you only change the second process. First point to change is the reset condition, if the reset button is pressed, the digit variable should change from the current value to 9. The second point is inside the elsif block. In this block it should change from increasing digit by 1 to decreasing digit by 1. This can be done by changing from digit + 1 into digit -1. The last point is the if digit hits 0, it should turn digit to 9. This can be done by changing inside the if digit = 9 block into if digit = 0 then digit <= 9 instead. Thus, the final VHDL code and implemented circuit of the counter down in this part should be as follows:

```
...
process(cnt_en,rst)
begin
     if rst = ‘0’ then digit <= 9 ;
     elsif cnt_en’event and cnt_en = ’1’ then
          digit <= digit - 1 ;
          if digit = 0 then
               digit <= 9 ;
          end if;
     end if;
end process;
...
```

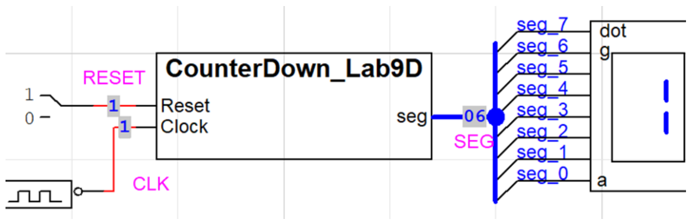


Figure 4: Counter-down circuit with VHDL of part D.

In part E, it is about combining the counter up from part C and the counter down from part D into single model by using VHDL. It will determine the direction of counting by the DIR signal. If DIR is 0, the circuit will count up. Otherwise, it will count down. The setup of model and circuit of this part is almost the same as previous part. Just add one more one-bit input (DIR) to the model setup and next step is writing a VHDL code. The VHDL code of this counter with direction is very simple. Just using two code from the second process block, one from each direction, and check with the if statement for which code to use. The parameter used in second process block is changed from 2 to 3 parameter in addition to direction signal. We use the if and else statement to check the direction signal (Also, declare one more local variable named direction). The first process block that is used to translating/converting the raw clock signal to cnt_en signal can stay exactly the same as in the previous 2 parts of the lab. Therefore, the final VHDL code of counter with specify direction and its implemented circuit should look like this:

```
...
signal direction: STD_LOGIC;
...
direction <= dir;
...
process(direction,cnt_en,rst)
begin
      if direction = ‘0’ then
      -- Up counter
            if rst = ‘0’ then digit <= 0 ;
            elsif cnt_en’event and cnt_en = ’1’ then
                  digit <= digit + 1 ;
                  if digit = 9 then
                        digit <= 0 ;
                  end if;
            end if;
      else
      -- Down counter
            if rst = ‘0’ then digit <= 9 ;
            elsif cnt_en’event and cnt_en = ’1’ then
                  digit <= digit - 1 ;
                  if digit = 0 then
                        digit <= 9 ;
                  end if;
            end if;
      end if;
end process;
...
```

*noted that ... means there is a code in between which is not related in this part.

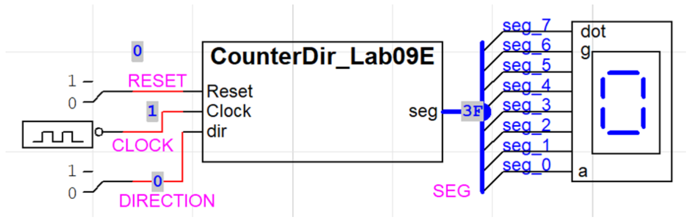


Figure 5: Counter with specify direction circuit with VHDL of part E.

Conclusion

In this lab, we learned how to create field programmable gate array (FPGA) from VDHL in LogicWorks 5 by using previous and this lab knowledge. And know how to implement the FPGA type circuit in VDHL. Also, we learn how to make counter up and counter down using FPGA and VDHL and lastly, we can learn and understand how to merge FPGA and Sequential Logic Circuit together. We can use this FPGA and VHDL knowledge to design and create more complicated sequential logic circuits in the future.