LAB 7 TUTORIAL

SIMPLE PROCESSOR

OVERVIEW

In this lab we will implement a simple 8-bit microprocessor. We will use the processor to calculate the average of 4 numbers x1, x2, x3, and x4 as instructed in example 1 in the lab manual. We will also use the processor to implement the for loop in example 2 in the lab manual. The inputs are entered in the simulation phase through the Quartus II waveform editor.

Our implementation of the 8-bit microprocessor consists of:

1. 8-bit registers R1 and RC to store data.
2. An 8-bit ALU to perform operations on its inputs A and B.
3. An 8-bit ACC register in which intermediate operations are stored and fed back to the ALU.
4. A control unit to execute instructions pointed to by the Program Counter (PC).

PROCEDURE

Please follow the following instructions to implement the 8-bit microprocessor:

1. Create a new folder “Lab7” in your “BME328” folder.
2. Open the Quartus II software, and using the new project wizard, create a new project “Lab7” in your “Lab7” folder.

   MAKE SURE THAT YOU CHOOSE THE “EP2C35F672C6” DEVICE IN THE PROJECT WIZARD.

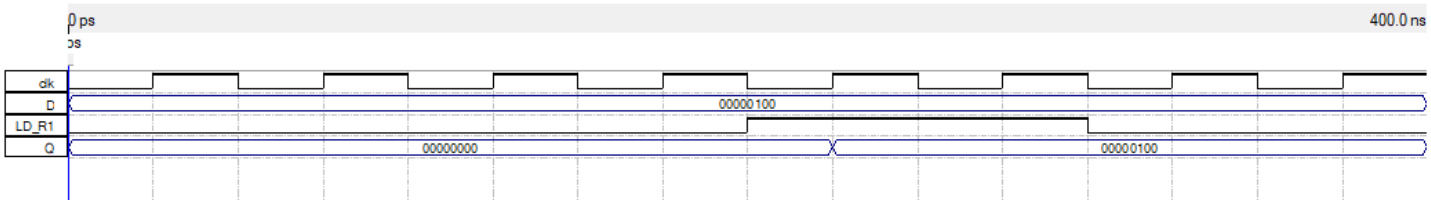
1. Register R1

Please follow the following instructions to implement Register R1:

1. Create a new VHDL file in your “Lab7” project (File > New > VHDL File).
2. Type the following in the Text Editor and save file as “R1.vhd”:

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity R1 is
5       port(
6           D       :       in      std_logic_vector(7 downto 0);
7           LD_R1   :       in      std_logic;
8           clk     :       in      std_logic;
9           Q       :       out     std_logic_vector(7 downto 0)
10          );
11  end R1;
12
13  architecture Behavior of R1 is
14  begin
15      process (LD_R1)
16      begin
17          if (clk'EVENT and clk = '1') then
18              if (LD_R1 = '1') then
19                  Q <= D;
20              end if;
21          end if;
22      end process;
23  end Behavior;
24
```

3. Set “R1.vhd” as the top-level entity. You can do this by right-clicking on “R1.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “R1.vhd” file. You can do this by right clicking on “R1.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “R1.vhd” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “R1.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.



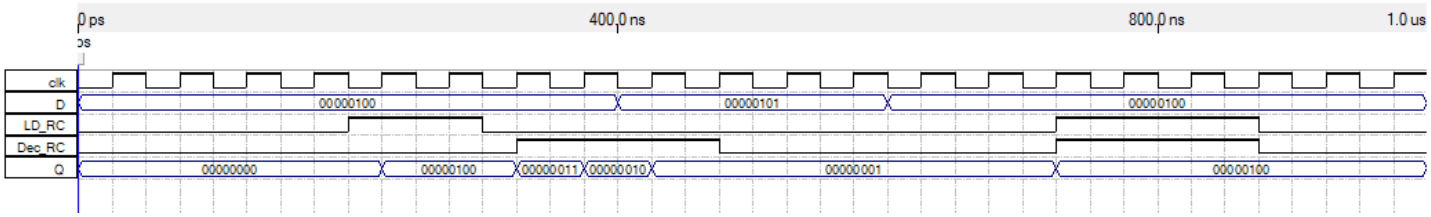
2. Register RC

Please follow the following instructions to implement Register RC:

1. Create a new VHDL file in your “Lab7” project (File > New > VHDL File).
2. Type the following in the Text Editor and save file as “RC.vhd”:

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity RC is
6       port (  D       :       in      unsigned (7 downto 0);
7               LD_RC   :       in      std_logic;
8               Dec_RC  :       in      std_logic;
9               clk     :       in      std_logic;
10              Q       :       out     unsigned (7 downto 0));
11  end RC;
12
13  architecture Behavior of RC is
14  signal Q_signal:        unsigned (7 downto 0);
15  begin
16      process (clk, LD_RC, Dec_RC)
17      begin
18          if (clk'EVENT and clk = '1') then
19              if (LD_RC = '1') then
20                  Q_signal <= D;
21              elsif (Dec_RC = '1') then
22                  Q_signal <= Q_signal - 1;
23              end if;
24          end if;
25      end process;
26      Q <= Q_signal;
27  end Behavior;
28
```

3. Set “RC.vhd” as the top-level entity. You can do this by right-clicking on “RC.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “RC.vhd” file. You can do this by right clicking on “RC.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “RC.vhd” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “RC.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.



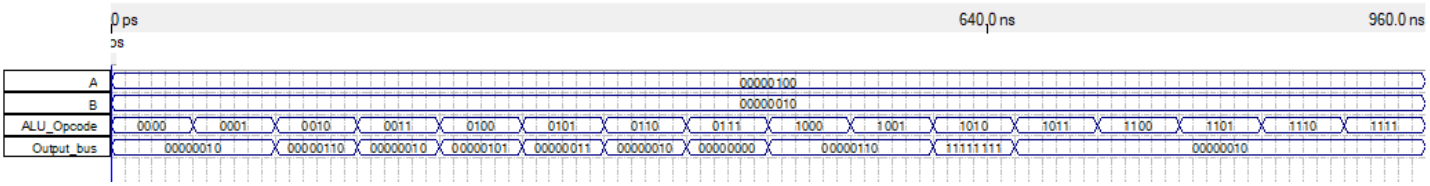
3. ALU

Please follow the following instructions to implement the ALU:

1. Create a new VHDL file in your “Lab7” project (File > New > VHDL File).
2. Type the following in the Text Editor and save file as “ALU.vhd”:

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4   use ieee.std_logic_unsigned.all;
5
6   entity ALU is
7       port (A, B       :  in      std_logic_vector(7 downto 0);
8             ALU_Opcode :  in      std_logic_vector(3 downto 0);
9             Output_bus :  out     std_logic_vector(7 downto 0));
10  end ALU;
11
12  architecture Behavior of ALU is
13      signal ALU_Result : std_logic_vector(7 downto 0);
14  begin
15      process (A, B, ALU_Opcode)
16      begin
17          case (ALU_Opcode) is
18              when "0000" => ALU_Result <= B;
19              when "0001" => ALU_Result <= B;
20              when "0010" => ALU_Result <= A + B;
21              when "0011" => ALU_Result <= A - B;
22              when "0100" => ALU_Result <= A + 1;
23              when "0101" => ALU_Result <= A - 1;
24              when "0110" => ALU_Result <= B;
25              when "0111" => ALU_Result <= A and B;
26              when "1000" => ALU_Result <= A or B;
27              when "1001" => ALU_Result <= A xor B;
28              when "1010" => ALU_Result <= A nand B;
29              when "1011" => ALU_Result <= B;
30              when "1100" => ALU_Result <= B;
31              when "1101" => ALU_Result <= B;
32              when "1110" => ALU_Result <= B;
33              when "1111" => ALU_Result <= B;
34          end case;
35      end process;
36      Output_bus <= ALU_Result;
37  end Behavior;
```

3. Set “ALU.vhd” as the top-level entity. You can do this by right-clicking on “ALU.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “ALU.vhd” file. You can do this by right clicking on “ALU.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “ALU.vhd” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “ALU.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.



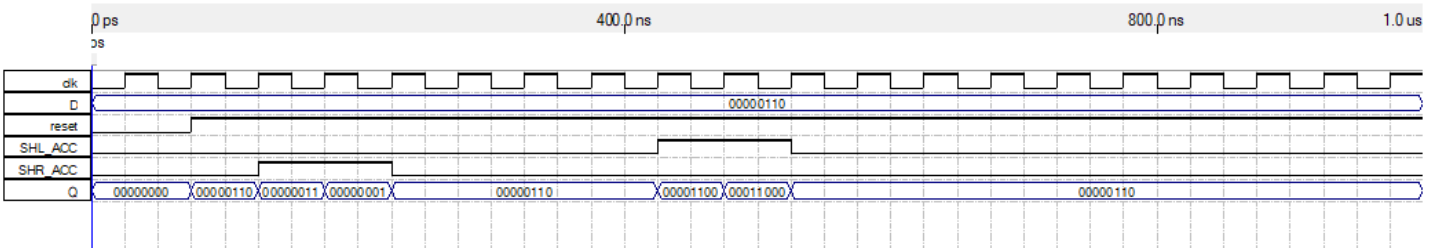
4. ACC

Please follow the following instructions to implement the ACC:

1. Create a new VHDL file in your “Lab7” project (File > New > VHDL File).
2. Type the following in the Text Editor and save file as “ACC.vhd”:

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity ACC is
6       port    (D                                      : in    unsigned(7 downto 0);
7                SHL_ACC, SHR_ACC, clk, reset           : in    std_logic;
8                Q                                      : out   unsigned(7 downto 0));
9   end ACC;
10
11  architecture Behavior of ACC is
12      signal store_signal : unsigned (7 downto 0) := (others => '0');
13  begin
14      process (SHL_ACC, clk, reset, SHR_ACC, D)
15      begin
16          if  (reset = '0') then
17              store_signal <= "00000000";
18          elsif ((reset = '1') and (rising_edge(clk))) then
19              store_signal <= D;
20              if (SHL_ACC = '1') then
21                  store_signal <= store_signal sll 1;
22              end if;
23              if (SHR_ACC = '1') then
24                  store_signal <= store_signal srl 1;
25              end if;
26          end if;
27      end process;
28      Q <= store_signal;
29  end Behavior;
```

3. Set “ACC.vhd” as the top-level entity. You can do this by right-clicking on “ACC.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “ACC.vhd” file. You can do this by right clicking on “ACC.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “ACC.vhd” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “ACC.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.



5. Control Unit (Example 1)

Please follow the following instructions to implement the Control Unit for Example 1 (average of 4 inputs):

1. Create a new VHDL file in your “Lab7” project (File > New > VHDL File).
2. Type the following in the Text Editor and save file as “PC_comb_ckt_example1.vhd”:

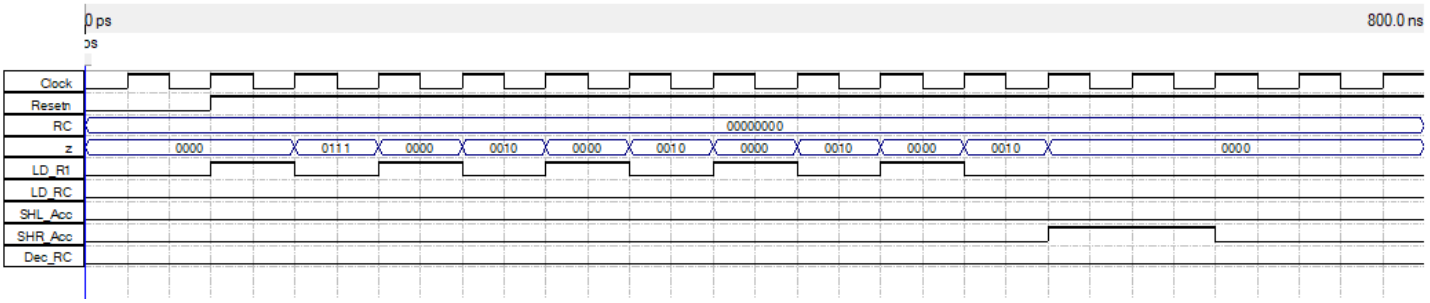
```
1   library ieee;
2   use ieee.std_logic_1164 .all;
3
4   entity PC_comb_ckt_example1  is
5       port(
6           Clock, Resetn    : in     std_logic;
7           RC               : in     std_logic_vector  ( 7 downto 0);
8           z                : out    std_logic_vector (3 downto 0);
9           Dec_RC, LD_R1, LD_RC, SHL_Acc, SHR_Acc : out std_logic
10          );
11  end PC_comb_ckt_example1 ;
12
13  architecture  Behavior of PC_comb_ckt_example1  is
14      type    state_type               is (s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11,
    s12, s13);
15      signal  y:                       state_type ;
16      signal  control_signal :         std_logic_vector (8 downto 0);
17      signal Op:                       std_logic_vector (7 downto 0) := X"00";
18      begin
19          process (Clock, Resetn)
20          begin
21              if  (Resetn = '0') then
22                  y <= s0;
23              elsif    (Clock'EVENT and Clock = '1') then
24                  case y is
25                      when s0  => y <= s1;
26                      when s1  => y <= s2;
27                      when s2  => y <= s3;
28                      when s3  => y <= s4;
29                      when s4  => y <= s5;
30                      when s5  => y <= s6;
31                      when s6  => y <= s7;
32                      when s7  => y <= s8;
33                      when s8  => y <= s9;
34                      when s9  => y <= s10;
35                      when s10 => y <= s11;
36                      when s11 => y <= s12;
37                      when s12 => y <= s13;
38                      when s13 =>
39                          if (Resetn = '0') then
40                              y <= s0;
41                          end if;
42                  end case;
43              end if;
44          end process ;
45
46
47          process (y)
48          begin
49              case y is
50                  when s0  =>  Op <=  X"0F";  -- rest
51                  when s1  =>  Op <=  X"00";  -- load R1 0
52                  when s2  =>  Op <=  X"07";  -- and
53                  when s3  =>  Op <=  X"00";  -- load X1
54                  when s4  =>  Op <=  X"02";  -- add
55                  when s5  =>  Op <=  X"00";  -- load X2
56                  when s6  =>  Op <=  X"02";  -- add
57                  when s7  =>  Op <=  X"00";  -- load X3
58                  when s8  =>  Op <=  X"02";  -- add
59                  when s9  =>  Op <=  X"00";  -- load X4
60                  when s10 =>  Op <=  X"02";  -- add
61                  when s11 =>  Op <= X"0C";   -- SRA
62                  when s12 =>  Op <=  X"0C";  -- SRA
63                  when s13 =>  Op <= X"0F";   -- rest
64              end case;
65
```

CONTINUED ON NEXT PAGE

```
66              case Op is
67                  when X"00" =>  control_signal  <= "000010000" ; --LDR1
68                  when X"01" =>  control_signal  <= "000100000" ; --LDRC
69                  when X"02" =>  control_signal  <= "000000010" ; --ADDA
70                  when X"03" =>  control_signal  <= "000000011" ; --SUBA
71                  when X"04" =>  control_signal  <= "000000100" ; --INCA
72                  when X"05" =>  control_signal  <= "000000101" ; --DECA
73                  when X"06" =>  control_signal  <= "001000000" ; --DEC_RC
74                  when X"07" =>  control_signal  <= "000000111" ; --ANDA
75                  when X"08" =>  control_signal  <= "000001000" ; --ORA
76                  when X"09" =>  control_signal  <= "000001001" ; --XORA
77                  when X"0A" =>  control_signal  <= "000001010" ; --NANDA
78                  when X"0B" =>  control_signal  <= "010000000" ; --SLA
79                  when X"0C" =>  control_signal  <= "100000000" ; --SRA
80                  when X"0D" =>  control_signal  <= "000000000" ; --BNEQ
81                  when others => control_signal  <= "000000000" ; --REST
82              end case;
83          end process ;
84          z <= control_signal (3 downto 0);
85          LD_R1 <= control_signal  (4);
86          LD_RC <= control_signal  (5);
87          Dec_RC <= control_signal  (6);
88          SHL_Acc <= control_signal  (7);
89          SHR_Acc <= control_signal  (8);
90  end Behavior ;
91
```

END OF VHDL FILE FOR CONTROL UNIT (EXAMPLE 1)

3. Set “PC_comb_ckt_example1.vhd” as the top-level entity. You can do this by right-clicking on “PC_comb_ckt_example1.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “PC_comb_ckt_example1.vhd” file. You can do this by right clicking on “PC_comb_ckt_example1.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “PC_comb_ckt_example1.vhd” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “PC_comb_ckt_example1.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.



6. Control Unit (Example 2)

Please follow the following instructions to implement the Control Unit for Example 2 (for loop):

1. Create a new VHDL file in your “Lab7” project (File > New > VHDL File).
2. Type the following in the Text Editor and save file as “PC_comb_ckt_example2.vhd”:

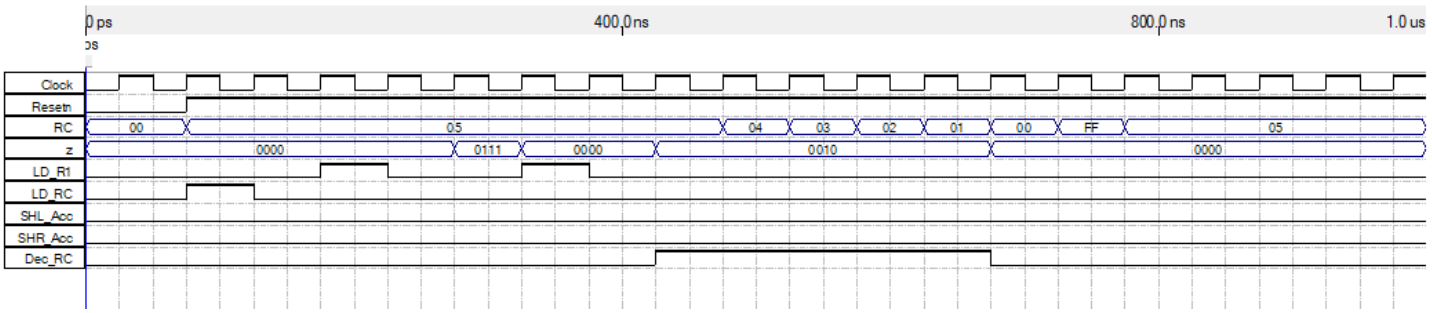
```
1   library ieee;
2   use ieee.std_logic_1164 .all;
3
4   entity PC_comb_ckt_example2  is
5       port(
6           Clock, Resetn    : in    std_logic;
7           RC               : in    std_logic_vector  ( 7 downto 0);
8           z                : out   std_logic_vector (3 downto 0);
9           Dec_RC, LD_R1, LD_RC, SHL_Acc, SHR_Acc : out std_logic
10          );
11  end PC_comb_ckt_example2 ;
12
13  architecture Behavior of PC_comb_ckt_example2  is
14      type    state_type              is (s0, s1, s2, s3, s4, s5, s6, s7, s8, s9);
15      signal  y:                      state_type;
16      signal  control_signal :        std_logic_vector (8 downto 0);
17      signal Op:                      std_logic_vector (7 downto 0) := X"00";
18      begin
19          process (Clock, Resetn)
20          begin
21              if  (Resetn = '0') then
22                  y <= s0;
23              elsif   (Clock'EVENT and Clock = '1') then
24                  case y is
25                      when s0  => y <= s1;
26                      when s1  => y <= s2;
27                      when s2  => y <= s3;
28                      when s3  => y <= s4;
29                      when s4  => y <= s5;
30                      when s5  => y <= s6;
31                      when s6  => y <= s7;
32                      when s7  => y <= s8;
33                      when s8  =>
34                          if (RC /= X"00") then
35                              y <= s8;
36                          else
37                              y <= s9;
38                          end if;
39                      when s9 =>
40                          if (Resetn = '0') then
41                              y <= s0;
42                          end if;
43                  end case;
44              end if;
45          end process;
46
47          process (y)
48          begin
49              case y is
50                  when s0  => Op <=  X"0F"; -- rest
51                  when s1  => Op <=  X"01"; -- load rc 5
52                  when s2  => Op <=  X"0F"; -- rest
53                  when s3  => Op <=  X"00"; -- load r1 0
54                  when s4  => Op <=  X"0F"; -- rest
55                  when s5  => Op <=  X"07"; -- and
56                  when s6  => Op <=  X"00"; -- load r1 x
57                  when s7  => Op <=  X"0F"; -- rest
58                  when s8  => Op <=  X"0D"; -- DEC_RC, result = result + x
59                  when s9  => Op <=  X"0F"; -- when loop is done, rest
60              end case;
61
62              case Op is
63                  when X"00" =>  control_signal  <= "000010000" ; --LDR1
64                  when X"01" =>  control_signal  <= "000100000" ; --LDRC
65                  when X"02" =>  control_signal  <= "000000010" ; --ADDA
66                  when X"03" =>  control_signal  <= "000000011" ; --SUBA
```

CONTINUED ON NEXT PAGE

```
67                  when X"04" =>  control_signal  <= "000000100" ; --INCA
68                  when X"05" =>  control_signal  <= "000000101" ; --DECA
69                  when X"06" =>  control_signal  <= "001000000" ; --DEC_RC
70                  when X"07" =>  control_signal  <= "000000111" ; --ANDA
71                  when X"08" =>  control_signal  <= "000001000" ; --ORA
72                  when X"09" =>  control_signal  <= "000001001" ; --XORA
73                  when X"0A" =>  control_signal  <= "000001010" ; --NANDA
74                  when X"0B" =>  control_signal  <= "010000000" ; --SLA
75                  when X"0C" =>  control_signal  <= "100000000" ; --SRA
76                  when X"0D" =>  control_signal  <= "001000010" ; --BNEQ
77                  when others => control_signal  <= "000000000" ; --REST
78              end case;
79
80
81          end process ;
82
83          z <= control_signal (3 downto 0);
84          LD_R1 <= control_signal  (4);
85          LD_RC <= control_signal  (5);
86          Dec_RC <= control_signal  (6);
87          SHL_Acc <= control_signal  (7);
88          SHR_Acc <= control_signal  (8);
89  end Behavior ;
90
```

END OF VHDL CODE FOR CONTROL UNIT (EXAMPLE 2)

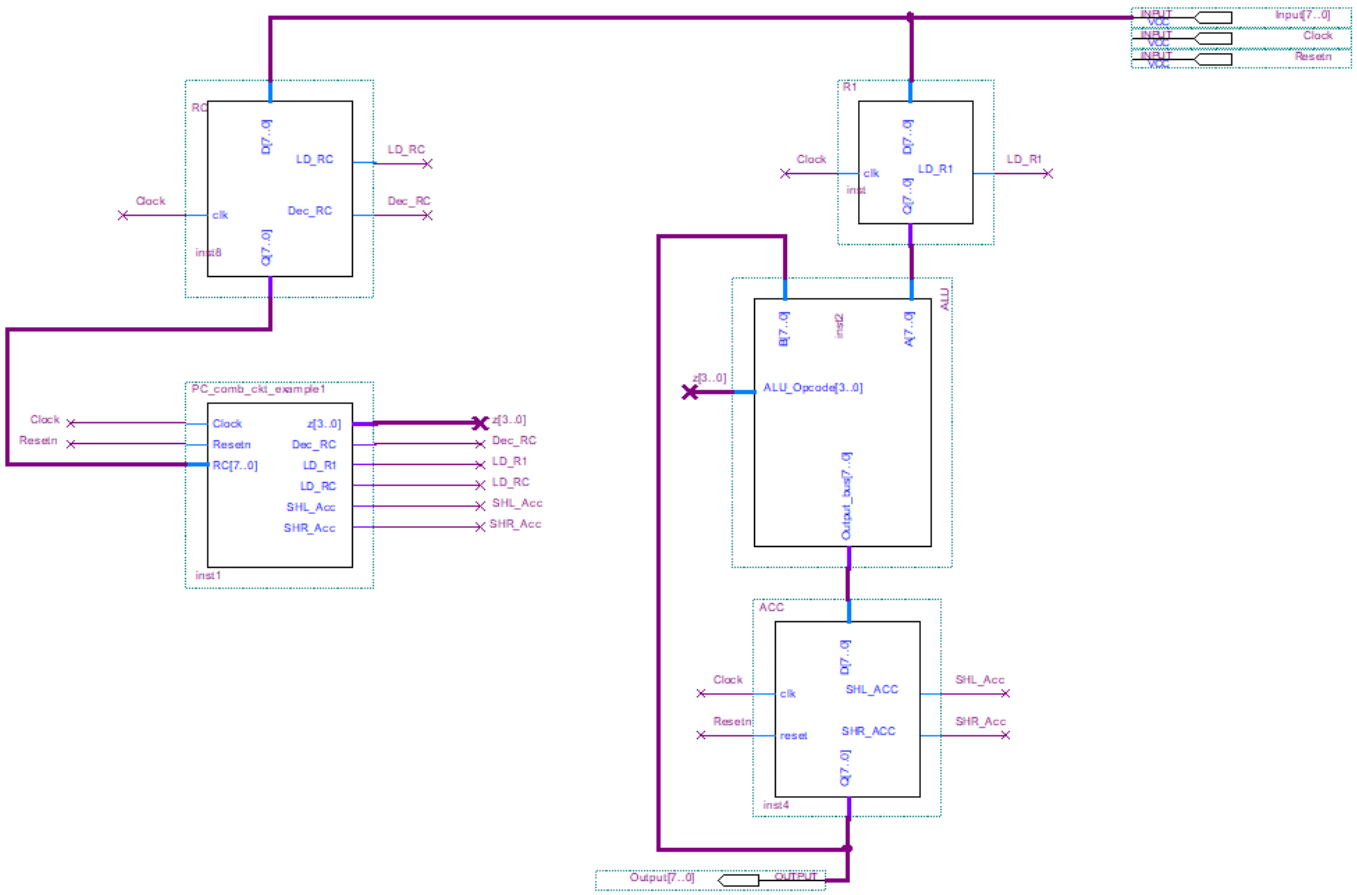
3. Set “PC_comb_ckt_example2.vhd” as the top-level entity. You can do this by right-clicking on “PC_comb_ckt_example2.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “PC_comb_ckt_example2.vhd” file. You can do this by right clicking on “PC_comb_ckt_example2.vhd” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “PC_comb_ckt_example2.vhd” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “PC_comb_ckt_example2.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.



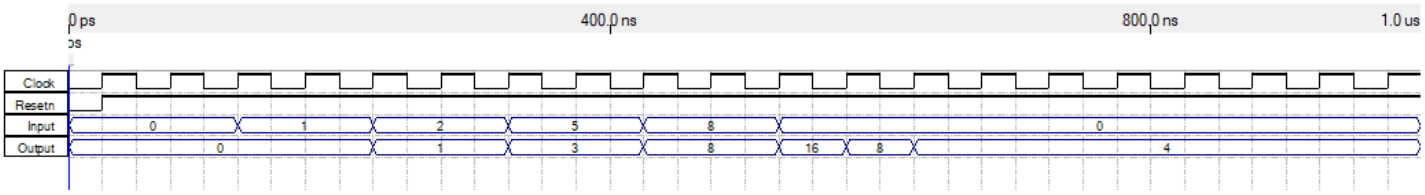
7. Final Design (Example 1)

Please follow the following instructions to implement the Final Design (Example 1):

1. Create a new Block Diagram/Schematic File in your “Lab7” project (File > New > Block Diagram/Schematic File).
2. Connect the components as shown in the Figure and save file as “Lab7_example1.bdf”:



3. Set “Lab7_example1.bdf” as the top-level entity. You can do this by right-clicking on “Lab7_example1.bdf” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “Lab7_example1.bdf” file. You can do this by right clicking on “Lab7_example1.bdf” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “Lab7_example1.bdf” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “Lab7_example1.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.

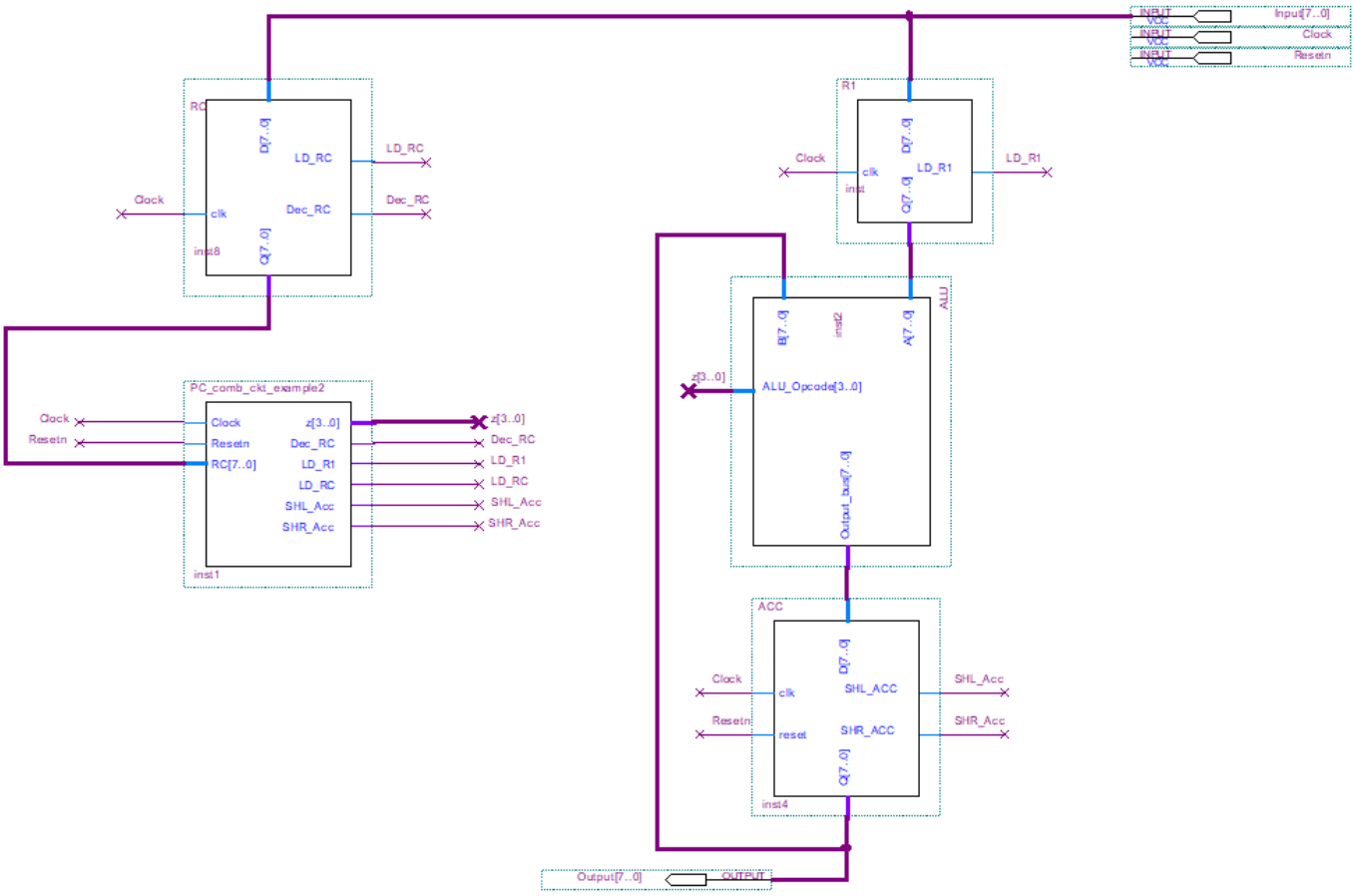


Simulation results for the average of 1, 2, 5, and 8. The result is 4.

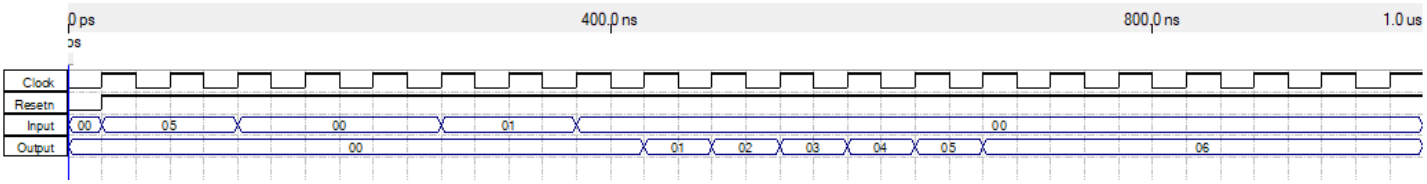
8. Final Design (Example 2)

Please follow the following instructions to implement the Final Design (Example 2):

1. Create a new Block Diagram/Schematic File in your “Lab7” project (File > New > Block Diagram/Schematic File).
2. Connect the components as shown in the Figure and save file as “Lab7_example2.bdf”:



3. Set “Lab7_example2.bdf” as the top-level entity. You can do this by right-clicking on “Lab7_example2.bdf” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Set at Top-Level Entity”.
4. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
5. Create a symbol for your “Lab7_example2.bdf” file. You can do this by right clicking on “Lab7_example2.bdf” in the “Files” section of the project navigator (located at the left of your Quartus II window) and selecting “Create Symbol Files for Current File”.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “Lab7_example2.bdf” and make sure that your simulation results match the results shown below.
8. Save the VWF file as “Lab7_example2.vwf” in your “Lab7” folder and take screenshots of your code and your simulation results.



Simulation results for I =5 and x = 1. Result equals 6 after the for loop completes its iterations.