In22-S2-CS1050- Computer Organization and Digital Design

Lab 9- 10

NANO-PROCESSOR

Design Competition
Lab Report

Abeysundara K.N.B. - 230010L
Seneviratne N.S - 230601B
H.M. Pasindu Lakmal - 220261L
Pankaja Waidyasekara – 230678 N

TABLE OF CONTENTS

Contents

1. Basic Nano-processor

1.1 Introduction

The objective of this lab is to create a 4-bit nano-processor that can execute instructions stored in it's program ROM. The high level design was given and the basic nano-processor was designed accordingly to fulfil it's expected functionality.

The required functionalities on the nano-processor are:

1. **MOVI** Ra,d - Move immediate 4-bit value d to register R.
2. **ADD** Ra,Rb - Add values in register Ra & Rb and store the result in Ra.
3. **NEG** R - 2's complement of the binary value in register R.
4. **JZR** R,d - Jump to line d(of instructions) if the value in register R is 0.

To build the following nano-processor, we have designed the following sub-components;

1. 4-bit Add/Subtract Unit
2. 3-bit Adder
3. 3-bit Program Counter
4. 2-way 3-bit Multiplexer
5. 2-way 4-bit Multiplexer
6. 8-way 4-bit Multiplexer
7. Register Bank
8. Program ROM
9. Instruction Decoder
10. Slow Clock

High level architecture design of the basic nano-processor

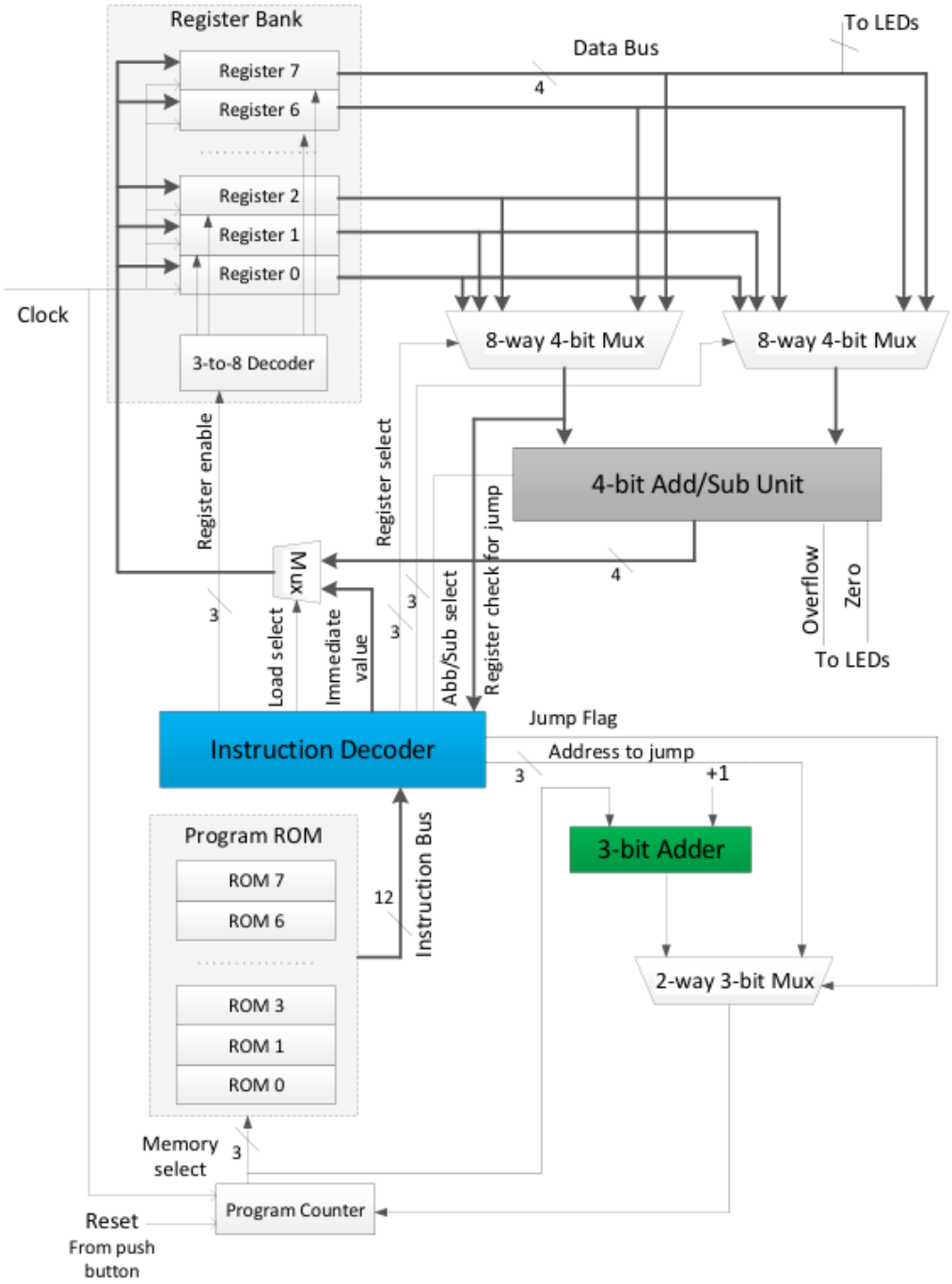


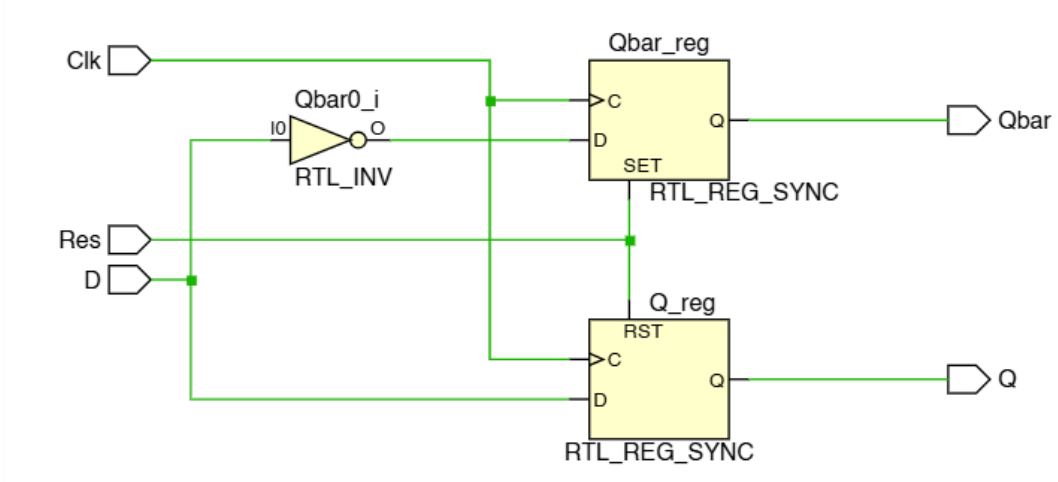
Figure 1 – High-level diagram of the nanoprocessor.

1.2 D-Flip Flop

1.2.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity D_FF is
 5.     Port (
 6.         D     : in  STD_LOGIC;
 7.         Res   : in  STD_LOGIC;
 8.         Clk   : in  STD_LOGIC;
 9.         Q     : out STD_LOGIC
10.     );
11. end D_FF;
12.
13. architecture Behavioral of D_FF is
14. begin
15.     process (Clk, Res)
16. begin
17.     if Res = '1' then
18.         Q <= '0';
19.     elsif rising_edge(Clk) then
20.         Q <= D;
21.     end if;
22. end process;
23.
24. end Behavioral;
```

1.2.2 Design Schematic

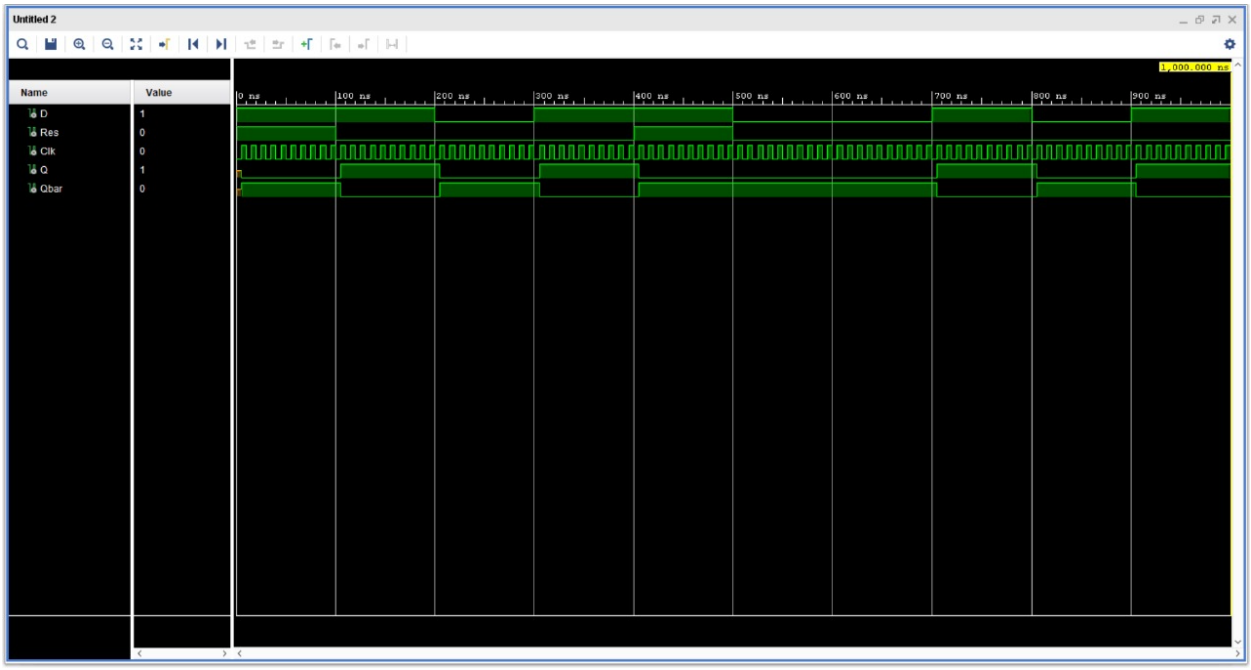


1.2.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. library IEEE;
 4. use IEEE.STD_LOGIC_1164.ALL;
 5.
 6. entity TB_DFF is
 7. --
 8. end TB_DFF;
 9.
10. architecture Behavioral of TB_DFF is
11.     -- Component Declaration
12.     component D_FF
13.         Port (
14.             D     : in  STD_LOGIC;
15.             Res   : in  STD_LOGIC;
16.             Clk   : in  STD_LOGIC;
17.             Q     : out STD_LOGIC
18.
19.         );
20.     end component;
21.
22.     -- Test Signals
23.     signal D     : STD_LOGIC := '0';
24.     signal Res   : STD_LOGIC := '0';
25.     signal Clk   : STD_LOGIC := '0';
26.     signal Q     : STD_LOGIC;
27.
28.
29. begin
30.     -- Instantiate the Unit Under Test (UUT)
31.     uut: D_FF Port Map (
32.         D     => D,
33.         Res   => Res,
34.         Clk   => Clk,
35.         Q     => Q
36.
37.     );
38.
39.     -- Clock Generation: 10 ns period
40.     clk_process : process
41.     begin
42.         while true loop
43.             Clk <= '0'; wait for 5 ns;
44.             Clk <= '1'; wait for 5 ns;
45.         end loop;
46.     end process;
47.
48.     -- Stimulus Process
49.     stim_proc: process
50.     begin
```

```
51.         -- Test 1: Apply Reset with D=1
52.         Res <= '1'; D <= '1'; wait for 100 ns;
53.
54.         -- Test 2: Deassert Reset, D=1
55.         Res <= '0'; D <= '1'; wait for 100 ns;
56.
57.         -- Test 3: D = 0
58.         D <= '0'; wait for 100 ns;
59.
60.         -- Test 4: D = 1
61.         D <= '1'; wait for 100 ns;
62.
63.         -- Test 5: Reassert Reset
64.         Res <= '1'; wait for 100 ns;
65.
66.         -- Test 6: Deassert Reset, D = 0
67.         Res <= '0'; D <= '0'; wait for 100 ns;
68.
69.         -- Test 7: Hold D = 0
70.         wait for 100 ns;
71.
72.         -- Test 8: D = 1
73.         D <= '1'; wait for 100 ns;
74.
75.         -- Test 9: D = 0
76.         D <= '0'; wait for 100 ns;
77.
78.         -- Test 10: D = 1
79.         D <= '1'; wait for 100 ns;
80.
81.         -- End of simulation
82.         wait;
83.     end process;
84.
85. end Behavioral;
86.
```

1.2.4 Timing Diagram



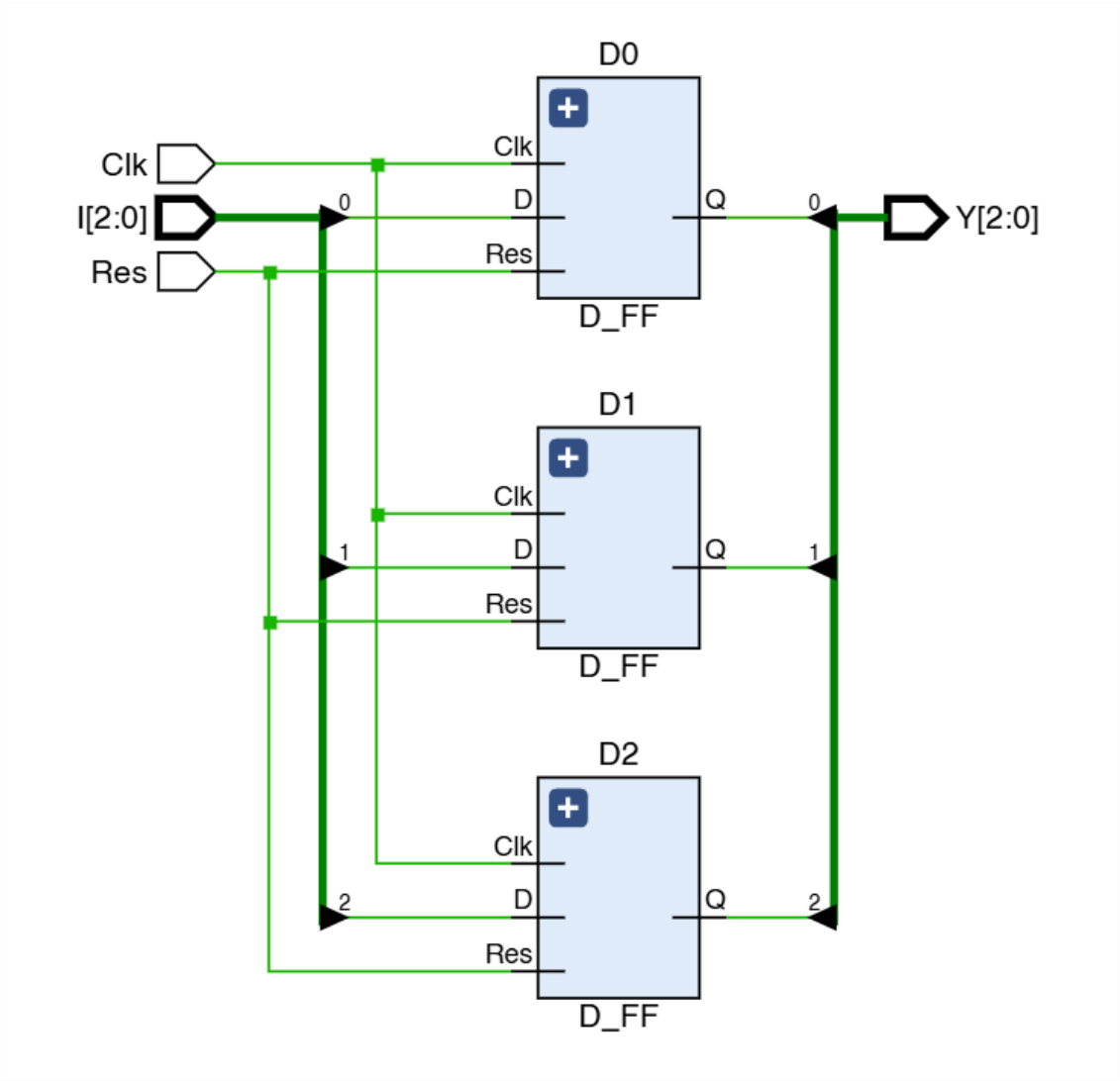
1.3 Program Counter

1.3.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Program_Counter is
 5.     Port (
 6.         Clk : in  STD_LOGIC;
 7.         Res : in  STD_LOGIC;
 8.         I   : in  STD_LOGIC_VECTOR (2 downto 0);
 9.         Y   : out STD_LOGIC_VECTOR (2 downto 0)
10.     );
11. end Program_Counter;
12.
13. architecture Behavioral of Program_Counter is
14.
15.     component D_FF
16.         Port (
17.             D     : in  STD_LOGIC;
18.             Res   : in  STD_LOGIC;
19.             Clk   : in  STD_LOGIC;
```

```
20.             Q     : out STD_LOGIC
21.         );
22.     end component;
23.
24. begin
25.
26.     D0 : D_FF
27.         port map (
28.             D     => I(0),
29.             Res   => Res,
30.             Clk   => Clk,
31.             Q     => Y(0)
32.         );
33.
34.     D1 : D_FF
35.         port map (
36.             D     => I(1),
37.             Res   => Res,
38.             Clk   => Clk,
39.             Q     => Y(1)
40.         );
41.
42.     D2 : D_FF
43.         port map (
44.             D     => I(2),
45.             Res   => Res,
46.             Clk   => Clk,
47.             Q     => Y(2)
48.         );
49.
50. end Behavioral;
51.
```

1.3.2 Elaborated Design Schematic



1.3.3 Simulation Source Code

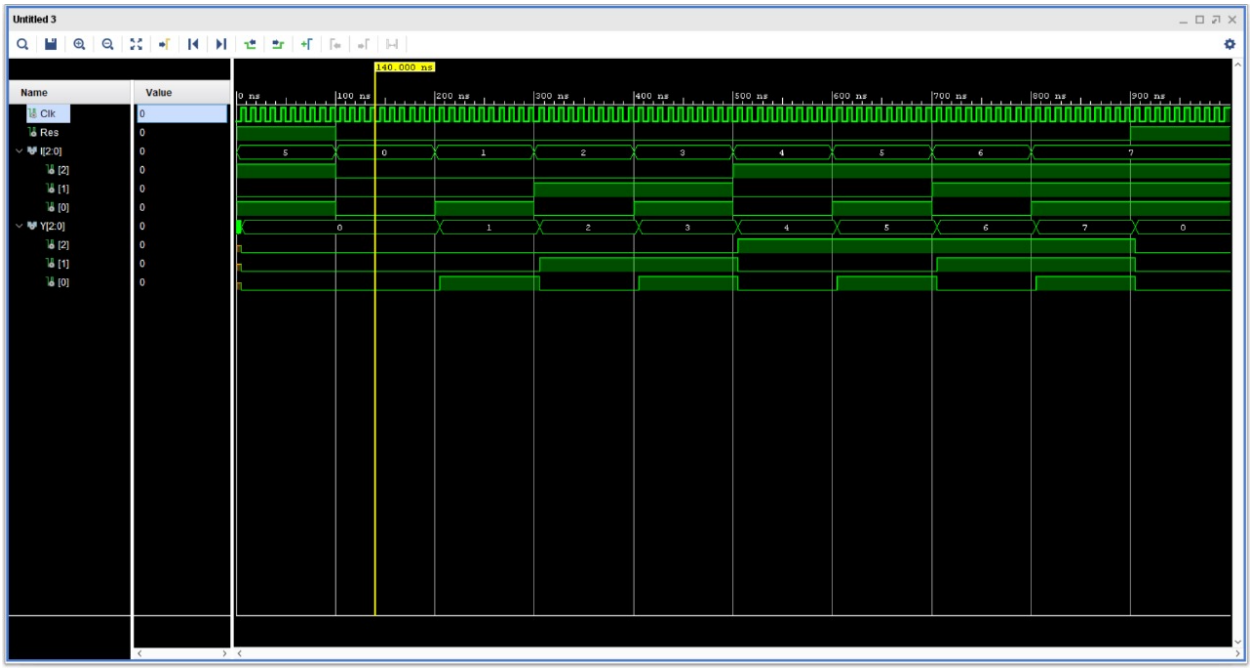
```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity TB_Program_Counter is
5. --
6. end TB_Program_Counter;
7.
```

```
 8. architecture Behavioral of TB_Program_Counter is
 9.
10.     -- Component Declaration
11.
12.     component Program_Counter
13.         Port (
14.             Clk : in  STD_LOGIC;
15.
16.             Res : in  STD_LOGIC;
17.
18.             I   : in  STD_LOGIC_VECTOR (2 downto 0);
19.
20.             Y   : out STD_LOGIC_VECTOR (2 downto 0)
21.       );
22.     end component;
23.     -- Test Signals
24.
25.     signal Clk : STD_LOGIC := '0';
26.
27.     signal Res : STD_LOGIC := '0';
28.
29.     signal I   : STD_LOGIC_VECTOR (2 downto 0) := "000";
30.
31.     signal Y   : STD_LOGIC_VECTOR (2 downto 0);
32.
33. begin
34.
35.     -- Instantiate the Unit Under Test (UUT)
36.
37.     uut: Program_Counter Port Map (
38.
39.         Clk => Clk,
40.
41.         Res => Res,
42.
43.         I   => I,
44.
45.         Y   => Y
46.
47.     );
48.
49.     -- Clock Generation: 10 ns period
50.
51.     clk_process : process
52.
53.     begin
54.
55.         while true loop
56.
57.             Clk <= '0'; wait for 5 ns;
58.
59.             Clk <= '1'; wait for 5 ns;
60.
61.         end loop;
```

```
 62.
 63.     end process;
 64.
 65.     -- Stimulus Process
 66.
 67.     stim_proc: process
 68.
 69.     begin
 70.
 71.         -- Test 1: Reset high, input doesn't matter
 72.
 73.         Res <= '1'; I <= "101"; wait for 100 ns;
 74.
 75.
 76.
 77.         -- Test 2: Load 000
 78.
 79.         Res <= '0'; I <= "000"; wait for 100 ns;
 80.
 81.
 82.
 83.         -- Test 3: Load 001
 84.
 85.         I <= "001"; wait for 100 ns;
 86.
 87.
 88.
 89.         -- Test 4: Load 010
 90.
 91.         I <= "010"; wait for 100 ns;
 92.
 93.
 94.
 95.         -- Test 5: Load 011
 96.
 97.         I <= "011"; wait for 100 ns;
 98.
 99.
100.
101.         -- Test 6: Load 100
102.
103.         I <= "100"; wait for 100 ns;
104.
105.
106.
107.         -- Test 7: Load 101
108.
109.         I <= "101"; wait for 100 ns;
110.
111.
112.
113.         -- Test 8: Load 110
114.
115.         I <= "110"; wait for 100 ns;
```

```
116.
117.
118.
119.         -- Test 9: Load 111
120.
121.         I <= "111"; wait for 100 ns;
122.
123.
124.
125.         -- Test 10: Reset again
126.
127.         Res <= '1'; wait for 100 ns;
128.
129.
130.
131.         -- End of simulation
132.
133.         wait;
134.
135.     end process;
136.
137.
138.
139. end Behavioral;
140.
```

1.3.4 Timing Diagram

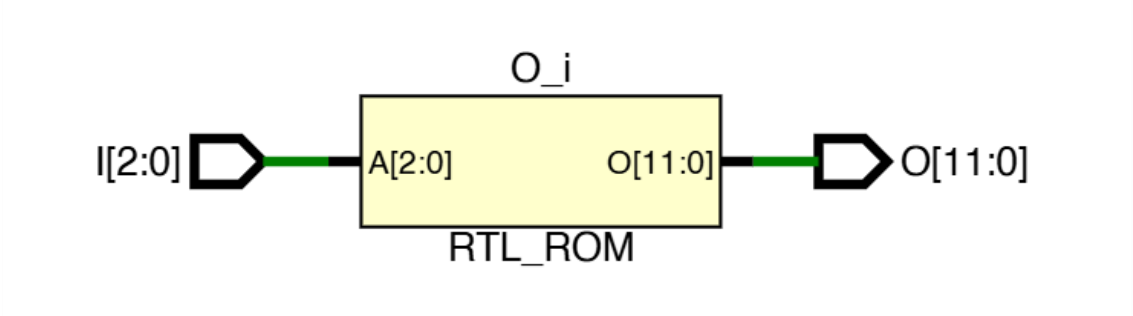


1.4 Program ROM

1.4.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use ieee.numeric_std.all;
 4.
 5.
 6. entity Program_Rom is
 7.     Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
 8.            O : out STD_LOGIC_VECTOR (11 downto 0));
 9. end Program_Rom;
10.
11. architecture Behavioral of Program_Rom is
12.
13.     type rom_type is array (0 to 7) of std_logic_vector (11 downto 0);
14.         signal program_ROM : rom_type := (
15.             "100010000010", - MOVI R1, 2
16.             "101110000011", --MOVI R7,3
17.             "100100000001", --MOVI R2, 1
18.             "010100000000", --NEG R2
19.             "001110010000", --ADD R7, R1
20.             "000010100000", --ADD R1, R2
21.             "110010000110", --JZR R1, 7
22.             "110000000100" --JZR R0, 5
23.             );
24.
25. begin
26.
27.     O <= program_ROM(to_integer(unsigned(I)));
28.
29. end Behavioral;
30.
```

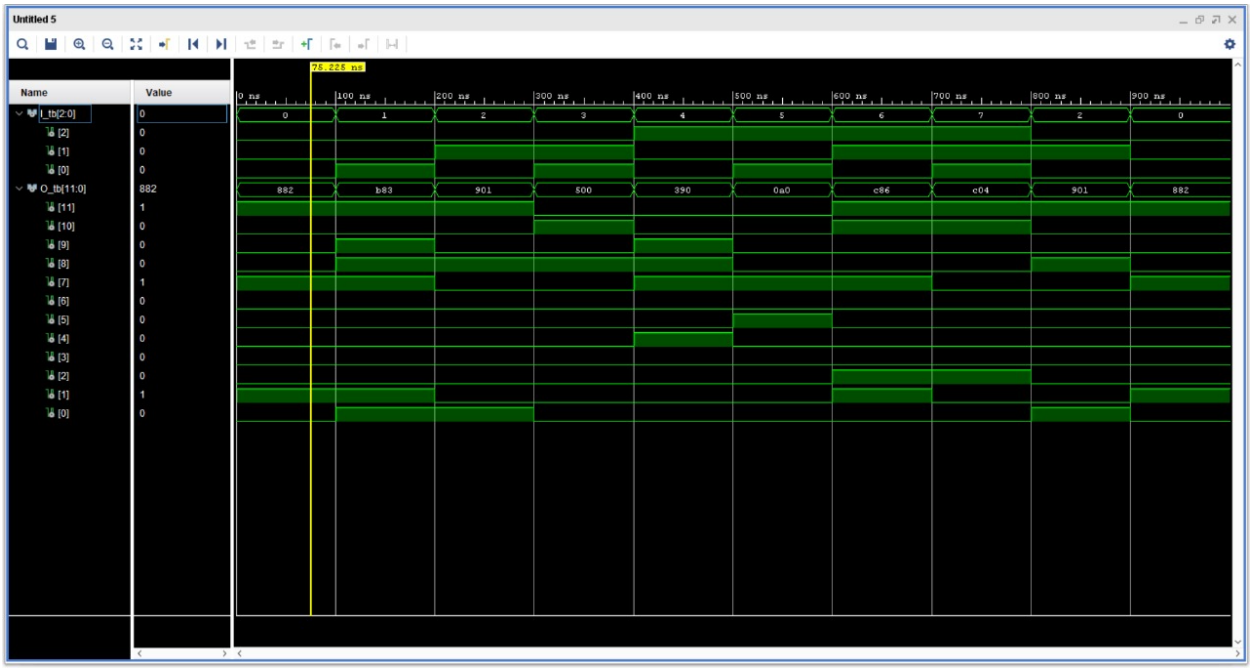
1.4.2 Elaborated Design Schematic



1.4.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use IEEE.NUMERIC_STD.ALL;
 4.
 5. entity TB_Program_Rom is
 6. end TB_Program_Rom;
 7.
 8. architecture tb of TB_Program_Rom is
 9.
10.     -- Declare the component under test
11.     component Program_Rom
12.         Port (
13.             I : in  STD_LOGIC_VECTOR (2 downto 0);
14.             O : out STD_LOGIC_VECTOR (11 downto 0)
15.         );
16.     end component;
17.
18.     -- Signals to connect to UUT
19.     signal I_tb : STD_LOGIC_VECTOR (2 downto 0) := (others => '0');
20.     signal O_tb : STD_LOGIC_VECTOR (11 downto 0);
21.
22. begin
23.
24.     -- Instantiate the UUT
25.     uut: Program_Rom
26.         port map (
27.             I => I_tb,
28.             O => O_tb
29.         );
30.
31.     -- Stimulus process
32.     stim_proc: process
33.     begin
34.         -- Test 10 different inputs with 100 ns delay
35.         I_tb <= "000"; wait for 100 ns;  -- Expect "100010000010"
36.         I_tb <= "001"; wait for 100 ns;  -- Expect "101110000011"
37.         I_tb <= "010"; wait for 100 ns;  -- Expect "100100000001"
38.         I_tb <= "011"; wait for 100 ns;  -- Expect "010100000000"
39.         I_tb <= "100"; wait for 100 ns;  -- Expect "001110010000"
40.         I_tb <= "101"; wait for 100 ns;  -- Expect "000010100000"
41.         I_tb <= "110"; wait for 100 ns;  -- Expect "110010000110"
42.         I_tb <= "111"; wait for 100 ns;  -- Expect "110000000100"
43.         I_tb <= "010"; wait for 100 ns;  -- Re-check
44.         I_tb <= "000"; wait for 100 ns;  -- Re-check
45.
46.         wait;  -- End simulation
47.     end process;
48.
49. end tb;
50.
```

1.4.4 Timing Diagram

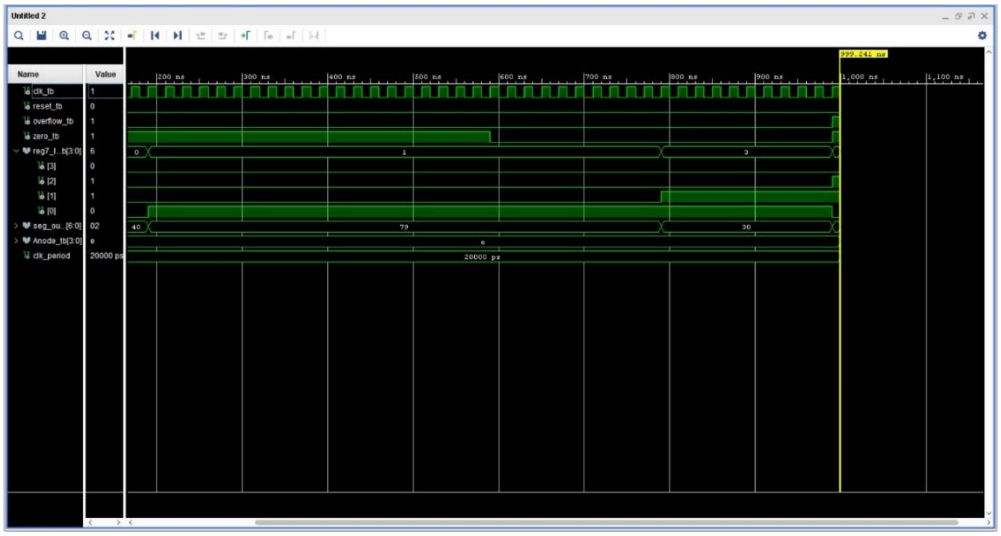


1.4.5 Simulation Source Code for the given function (add numbers from 1 to 3)

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use ieee.numeric_std.all;
 4.
 5.
 6. entity Program_Rom is
 7.     Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
 8.            O : out STD_LOGIC_VECTOR (11 downto 0));
 9. end Program_Rom;
10.
11. architecture Behavioral of Program_Rom is
12.
13.     type rom_type is array (0 to 6) of std_logic_vector (11 downto 0);
14.         signal program_ROM : rom_type := (
15.             "101110000001", --Movi R7,1
16.             "101100000010", --MOVI R6,2
17.             "101010000011", --MOVI R5,3
18.             "001111100000", --ADD R7, R6
19.             "001111010000", --ADD R7, R5
20.             "110010000110", --JZR R1, 7
21.              "110000000100"  --JZR R0, 5
22.             );
23.
24. begin
```

```
25.
26.     O <= program_ROM(to_integer(unsigned(I)));
27.
28. end Behavioral;
29.
```

1.4.6 Timing Diagram

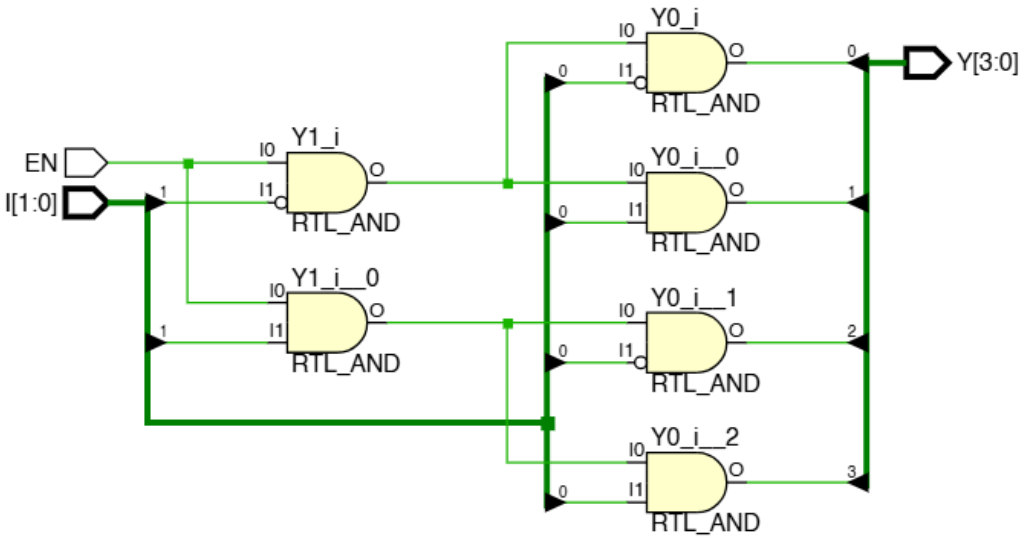


.5 2-4 Decoder

1.5.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Decoder_2_to_4 is
 5.     Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
 6.            EN : in STD_LOGIC;
 7.            Y : out STD_LOGIC_VECTOR (3 downto 0));
 8. end Decoder_2_to_4;
 9.
10. architecture Behavioral of Decoder_2_to_4 is
11.
12. begin
13.   Y(0) <= EN AND NOT (I(1)) AND NOT (I(0));
14.   Y(1) <= EN AND NOT (I(1)) AND I(0);
15.   Y(2) <= EN AND I(1) AND NOT (I(0));
16.   Y(3) <= EN AND I(1) AND I(0);
17.
18. end Behavioral;
19.
```

1.5.2 Elaborated Design Schematic

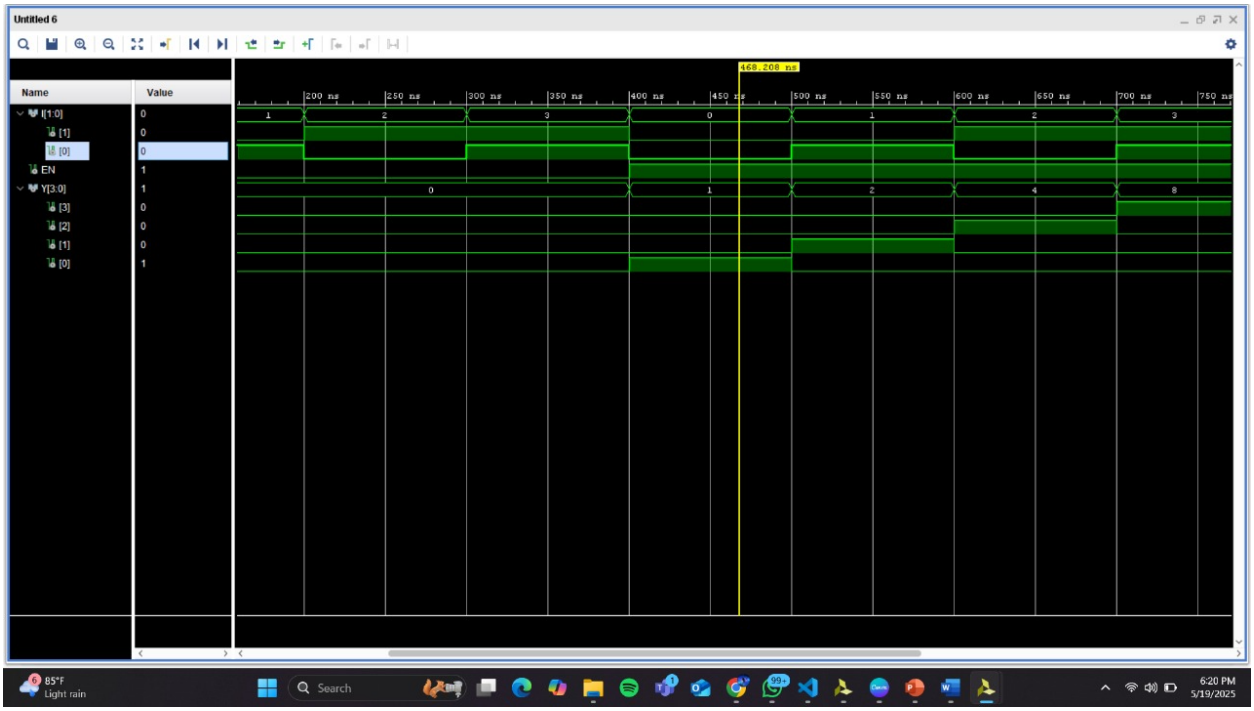


1.5.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_Decoder_2_to_4 is
 5. end TB_Decoder_2_to_4;
 6.
 7. architecture Behavioral of TB_Decoder_2_to_4 is
 8.
 9.     -- Component Declaration
10.     component Decoder_2_to_4
11.         Port (
12.             I  : in  STD_LOGIC_VECTOR (1 downto 0);
13.             EN : in  STD_LOGIC;
14.             Y  : out STD_LOGIC_VECTOR (3 downto 0)
15.         );
16.     end component;
17.
18.     -- Test Signals
19.     signal I  : STD_LOGIC_VECTOR (1 downto 0) := "00";
20.     signal EN : STD_LOGIC := '0';
21.     signal Y  : STD_LOGIC_VECTOR (3 downto 0);
22.
23. begin
24.
25.     -- Instantiate the UUT
26.     uut: Decoder_2_to_4 Port Map (
```

```
27.         I  => I,
28.         EN => EN,
29.         Y  => Y
30.     );
31.
32.     -- Stimulus Process
33.     stim_proc: process
34.     begin
35.         -- Test 1: Disabled, I = "00"
36.         EN <= '0'; I <= "00"; wait for 100 ns;
37.
38.         -- Test 2: Disabled, I = "01"
39.         I <= "01"; wait for 100 ns;
40.
41.         -- Test 3: Disabled, I = "10"
42.         I <= "10"; wait for 100 ns;
43.
44.         -- Test 4: Disabled, I = "11"
45.         I <= "11"; wait for 100 ns;
46.
47.         -- Test 5: Enabled, I = "00"
48.         EN <= '1'; I <= "00"; wait for 100 ns;
49.
50.         -- Test 6: Enabled, I = "01"
51.         I <= "01"; wait for 100 ns;
52.
53.         -- Test 7: Enabled, I = "10"
54.         I <= "10"; wait for 100 ns;
55.
56.         -- Test 8: Enabled, I = "11"
57.         I <= "11"; wait for 100 ns;
58.
59.         -- Test 9: Toggle Enable off then back on
60.         EN <= '0'; I <= "10"; wait for 100 ns;
61.         EN <= '1'; I <= "10"; wait for 100 ns;
62.
63.         -- End of simulation
64.         wait;
65.     end process;
66.
67. end Behavioral;
68.
```

1.5.4 Timing Diagram



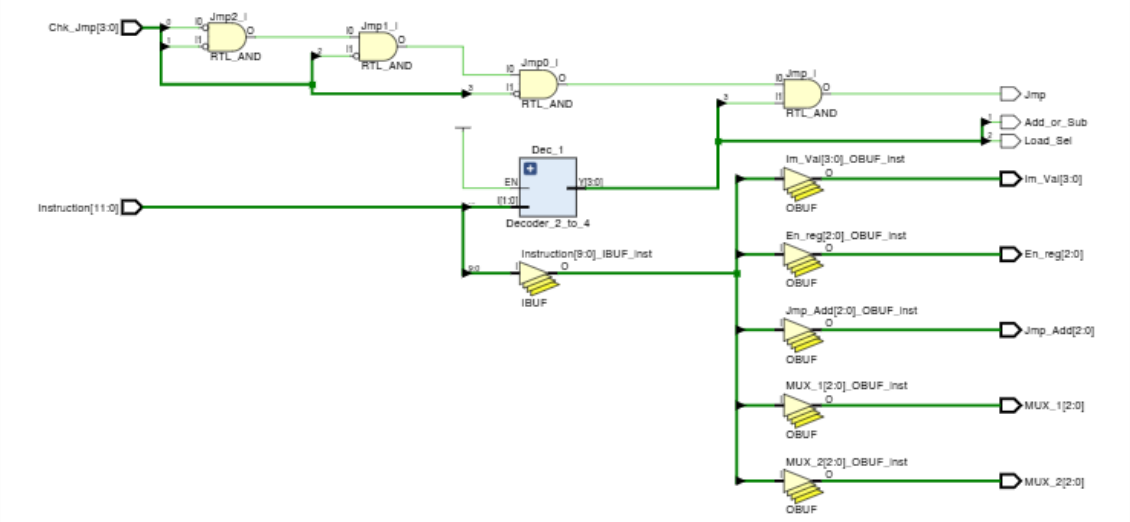
1.6 Instruction Decoder

1.6.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4.
 5. entity Instruction_Decoder is
 6.     Port (Instruction : in STD_LOGIC_VECTOR (11 downto 0);
 7.           Chk_Jmp : in STD_LOGIC_VECTOR (3 downto 0);
 8.           En_reg : out STD_LOGIC_VECTOR (2 downto 0);
 9.           MUX_1 : out STD_LOGIC_VECTOR (2 downto 0);
10.           MUX_2 : out STD_LOGIC_VECTOR (2 downto 0);
11.           Jmp : out STD_LOGIC;
12.           Add_or_Sub : out STD_LOGIC;
13.           Im_Val : out STD_LOGIC_VECTOR (3 downto 0);
14.           Load_Sel : out STD_LOGIC;
15.           Jmp_Add :  out STD_LOGIC_VECTOR (2 downto 0)
16.           );
17. end Instruction_Decoder;
18.
19. architecture Behavioral of Instruction_decoder is
```

```
20. component Decoder_2_to_4
21.     Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
22.        EN : in STD_LOGIC;
23.        Y : out STD_LOGIC_VECTOR (3 downto 0));
24. end component;
25. signal dec_line : STD_LOGIC_VECTOR (3 downto 0);
26.
27. begin
28.
29. Dec_1 : Decoder_2_to_4
30.     PORT MAP (I => Instruction (11 downto 10),
31.               EN => '1',
32.               Y => dec_line);
33.
34. MUX_1 <= Instruction (9 downto 7);
35. MUX_2 <= Instruction (6 downto 4);
36. Add_or_Sub <= dec_line(1); --add=>0 and sub=>1
37. Load_Sel <= dec_line(2);
38. Jmp <= not Chk_Jmp(0) and not Chk_Jmp(1) and not Chk_Jmp(2) and not Chk_Jmp(3)
and dec_line(3);
39. En_Reg <= Instruction (9 downto 7);
40. Jmp_add <= Instruction (2 downto 0);
41. Im_Val <= Instruction(3 downto 0);
42.
43. end Behavioral;
44.
```

1.6.2 Elaborated Design Schematic



When the instruction code consists of 12 bits, the combination of the 1st 2 Most significant bits (MSB) represents the OP code, which tells the system which type of instruction should be executed.

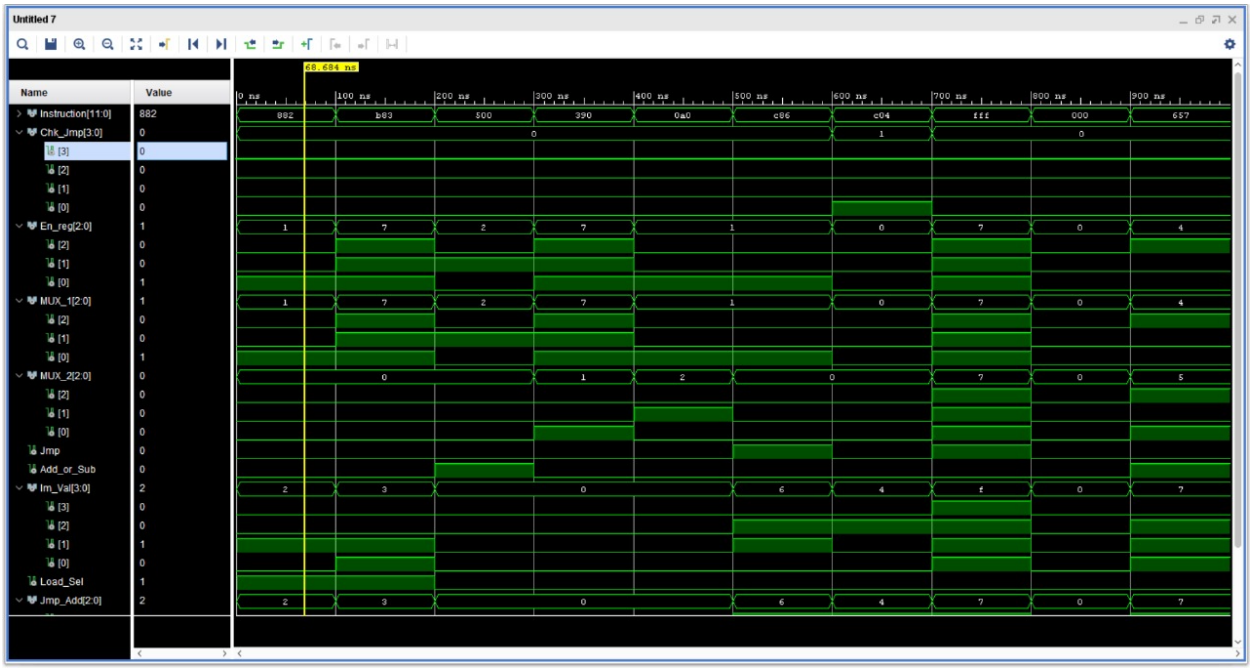
Op Code	Instruction	Example
10	MOVI R,d	1 0 R R R 0 0 0 d d d d (Store the values represented in d d d d in the register shown in R R R)
00	ADD Ra, Rb	0 0 Ra Ra Ra Rb Rb Rb 0 0 0 0 (Add the values stored in (Ra Ra Ra) and (Rb Rb Rb) and store it back to register Ra)
01	NEG R	0 1 R R R 0 0 0 0 0 0 0 (Subtract the value stored in register (R R R) from 0 to make the value negative, and store it back in (R R R)
11	JZR R,d	1 1 R R R 0 0 0 0 d d d (If (R R R) = 0, program counter should call ROM (d d d), else program counter should increase it's value by 1)'

1.6.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_Instruction_Decoder is
 5. end TB_Instruction_Decoder;
 6.
 7. architecture Behavioral of TB_Instruction_Decoder is
 8.
 9.     -- Component Declaration
10.     component Instruction_Decoder
11.         Port (
12.             Instruction : in  STD_LOGIC_VECTOR (11 downto 0);
13.             Chk_Jmp     : in  STD_LOGIC_VECTOR (3 downto 0);
14.             En_reg      : out STD_LOGIC_VECTOR (2 downto 0);
15.             MUX_1       : out STD_LOGIC_VECTOR (2 downto 0);
16.             MUX_2       : out STD_LOGIC_VECTOR (2 downto 0);
17.             Jmp         : out STD_LOGIC;
18.             Add_or_Sub  : out STD_LOGIC;
19.             Im_Val      : out STD_LOGIC_VECTOR (3 downto 0);
20.             Load_Sel    : out STD_LOGIC;
21.             Jmp_Add     : out STD_LOGIC_VECTOR (2 downto 0)
22.         );
23.     end component;
24.
25.     -- Test Signals
26.     signal Instruction : STD_LOGIC_VECTOR (11 downto 0) := (others => '0');
27.     signal Chk_Jmp     : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
28.     signal En_reg      : STD_LOGIC_VECTOR (2 downto 0);
29.     signal MUX_1       : STD_LOGIC_VECTOR (2 downto 0);
30.     signal MUX_2       : STD_LOGIC_VECTOR (2 downto 0);
31.     signal Jmp         : STD_LOGIC;
32.     signal Add_or_Sub  : STD_LOGIC;
33.     signal Im_Val      : STD_LOGIC_VECTOR (3 downto 0);
34.     signal Load_Sel    : STD_LOGIC;
35.     signal Jmp_Add     : STD_LOGIC_VECTOR (2 downto 0);
36.
37. begin
38.
39.     -- Instantiate the UUT
40.     uut: Instruction_Decoder Port Map (
41.         Instruction => Instruction,
42.         Chk_Jmp     => Chk_Jmp,
43.         En_reg      => En_reg,
44.         MUX_1       => MUX_1,
45.         MUX_2       => MUX_2,
46.         Jmp         => Jmp,
47.         Add_or_Sub  => Add_or_Sub,
48.         Im_Val      => Im_Val,
49.         Load_Sel    => Load_Sel,
50.         Jmp_Add     => Jmp_Add
```

```
 51.     );
 52.
 53.     -- Stimulus Process
 54.     stim_proc: process
 55.     begin
 56.         -- Test 1: MOVI instruction to R1
 57.         Instruction <= "100010000010"; -- MOVI R1, 2
 58.         Chk_Jmp <= "0000"; wait for 100 ns;
 59.
 60.         -- Test 2: MOVI instruction to R7
 61.         Instruction <= "101110000011"; -- MOVI R7, 3
 62.         wait for 100 ns;
 63.
 64.         -- Test 3: NEG R2
 65.         Instruction <= "010100000000"; -- NEG R2
 66.         wait for 100 ns;
 67.
 68.         -- Test 4: ADD R7, R1
 69.         Instruction <= "001110010000"; -- ADD R7, R1
 70.         wait for 100 ns;
 71.
 72.         -- Test 5: SUB R1, R2
 73.         Instruction <= "000010100000"; -- ADD (but will interpret via
dec_line(1) logic)
 74.         wait for 100 ns;
 75.
 76.         -- Test 6: JZR R1, 7 – should set Jmp high if Chk_Jmp = "0000"
 77.         Instruction <= "110010000110";
 78.         Chk_Jmp <= "0000"; wait for 100 ns;
 79.
 80.         -- Test 7: JZR R0, 5 – no jump if Chk_Jmp ? "0000"
 81.         Instruction <= "110000000100";
 82.         Chk_Jmp <= "0001"; wait for 100 ns;
 83.
 84.         -- Test 8: Test with all bits high
 85.         Instruction <= (others => '1');
 86.         Chk_Jmp <= "0000"; wait for 100 ns;
 87.
 88.         -- Test 9: All zeros
 89.         Instruction <= (others => '0');
 90.         Chk_Jmp <= "0000"; wait for 100 ns;
 91.
 92.         -- Test 10: Random combination
 93.         Instruction <= "011001010111";
 94.         Chk_Jmp <= "0000"; wait for 100 ns;
 95.
 96.         -- End simulation
 97.         wait;
 98.     end process;
 99.
100. end Behavioral;
101.
```

1.6.4 Timing Diagram

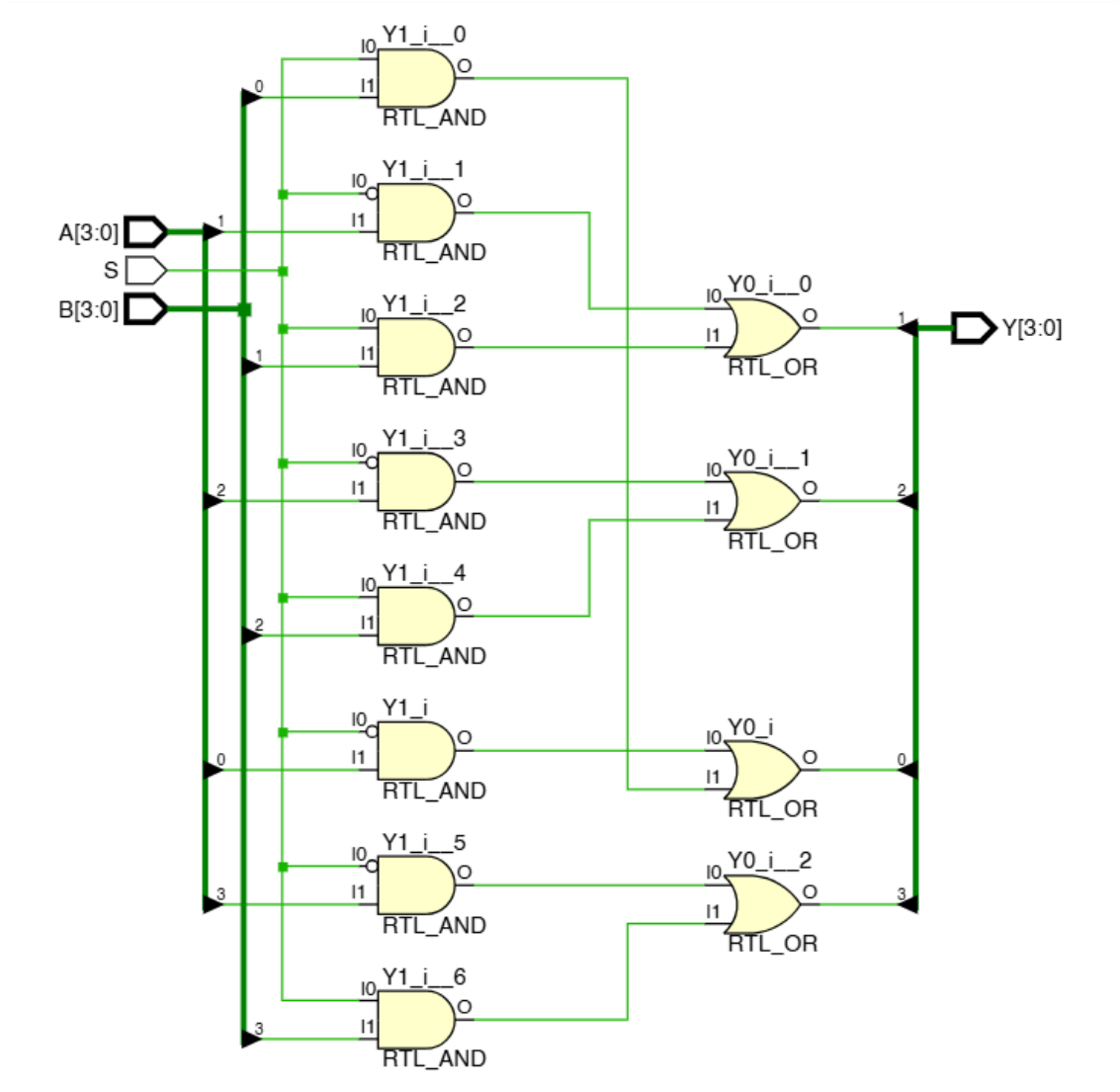


1.7 2 way 4 bit Multiplexer

1.7.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Multiplexer_2_way_4_bit is
 5.     Port ( S : in STD_LOGIC;
 6.            A : in STD_LOGIC_VECTOR (3 downto 0);
 7.            B : in STD_LOGIC_VECTOR (3 downto 0);
 8.            Y : out STD_LOGIC_VECTOR (3 downto 0));
 9. end Multiplexer_2_way_4_bit;
10.
11. architecture Behavioral of Multiplexer_2_way_4_bit is
12.
13. begin
14.
15. Y(0) <= (NOT S AND A(0)) OR (S AND B(0));
16. Y(1) <= (NOT S AND A(1)) OR (S AND B(1));
17. Y(2) <= (NOT S AND A(2)) OR (S AND B(2));
18. Y(3) <= (NOT S AND A(3)) OR (S AND B(3));
19.
20. end Behavioral;
```

1.7.2 Elaborated Design Schematic



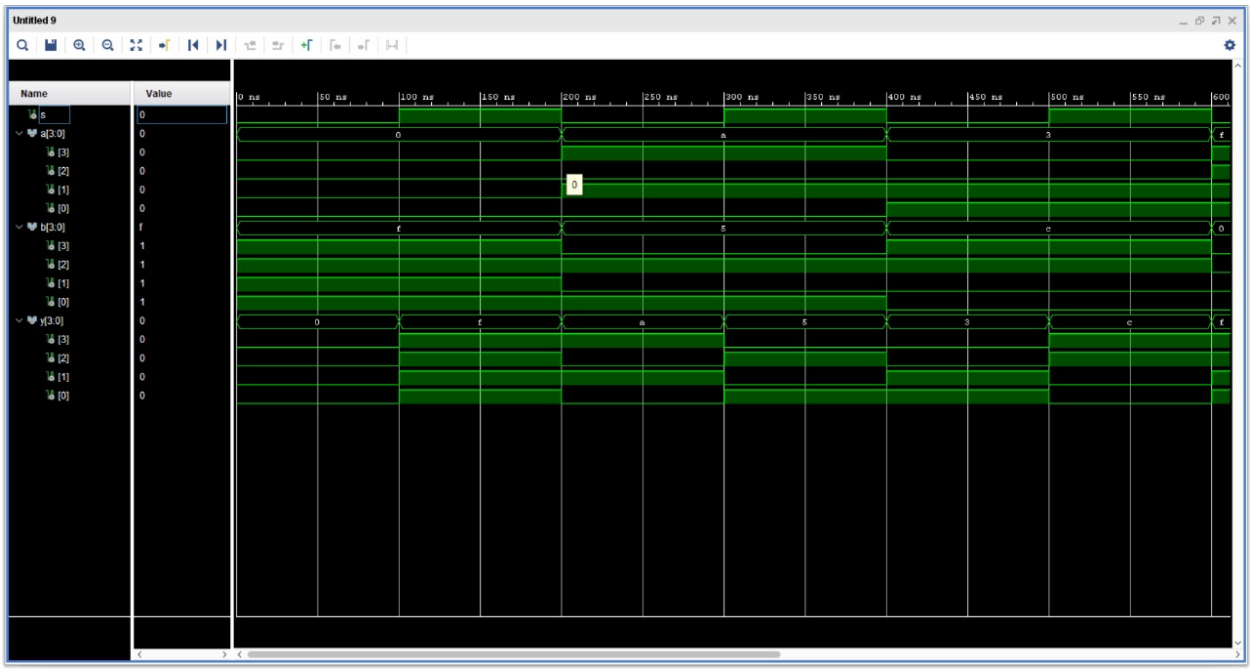
1.7.3 Simulation Source Code

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity TB_Multiplexer_2_way_4_bit is
5. --  Port ( );
6. end TB_Multiplexer_2_way_4_bit;
7.
```

```
 8. architecture Behavioral of TB_Multiplexer_2_way_4_bit is
 9.
10. component Multiplexer_2_way_4_bit
11.     Port ( S : in STD_LOGIC;
12.            A : in STD_LOGIC_VECTOR (3 downto 0);
13.            B : in STD_LOGIC_VECTOR (3 downto 0);
14.            Y : out STD_LOGIC_VECTOR (3 downto 0));
15. end component;
16.
17. signal s: STD_LOGIC;
18. signal a,b,y: STD_LOGIC_VECTOR (3 downto 0);
19.
20. begin
21.
22. UUT: Multiplexer_2_way_4_bit PORT MAP(
23.     S => s,
24.     A => a,
25.     B => b,
26.     Y => y);
27.
28. process begin
29.     s <= '0';
30.     a <= "0000";
31.     b <= "1111";
32.     wait for 100 ns;
33.
34.     s <= '1';
35.     a <= "0000";
36.     b <= "1111";
37.     wait for 100 ns;
38.
39.     s <= '0';
40.     a <= "1010";
41.     b <= "0101";
42.     wait for 100 ns;
43.
44.     s <= '1';
45.     a <= "1010";
46.     b <= "0101";
47.     wait for 100 ns;
48.
49.     s <= '0';
50.     a <= "0011";
51.     b <= "1100";
52.     wait for 100 ns;
53.
54.     s <= '1';
55.     a <= "0011";
56.     b <= "1100";
57.     wait for 100 ns;
58.
59.     s <= '0';
60.     a <= "1111";
61.     b <= "0000";
```

```
62.      wait for 100 ns;
63.
64.      s <= '1';
65.      a <= "1111";
66.      b <= "0000";
67.      wait for 100 ns;
68.
69.      s <= '0';
70.      a <= "0101";
71.      b <= "1010";
72.      wait for 100 ns;
73.
74.      s <= '1';
75.      a <= "0101";
76.      b <= "1010";
77.      wait for 100 ns;
78.
79.      end process;
80.
81. end Behavioral;
82.
```

1.7.4 Timing Diagram

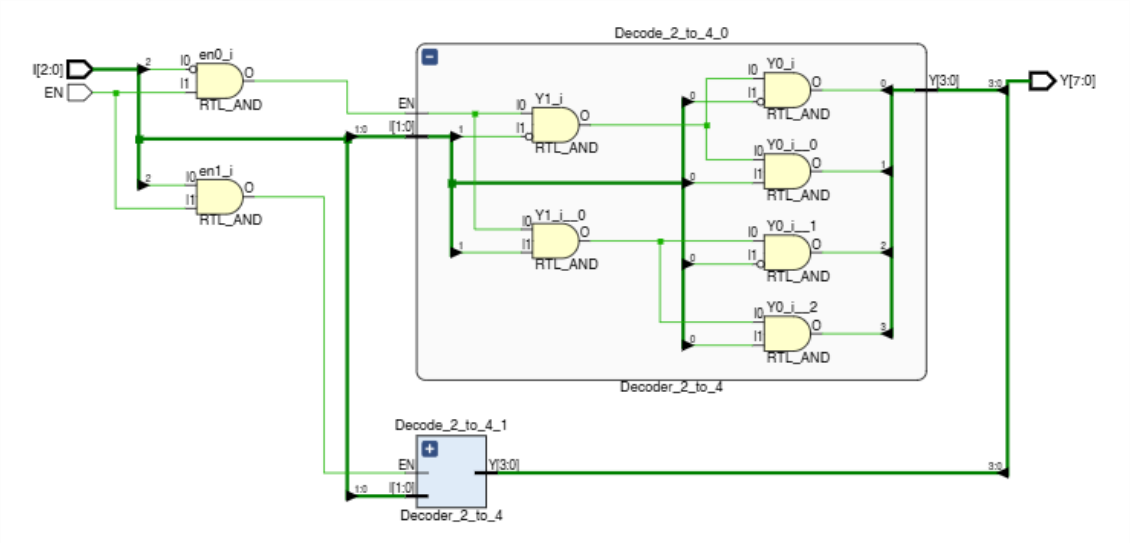


1.8 3-8 Decoder

1.8.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4.
 5.
 6. entity Decoder_3_to_8 is
 7.     Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
 8.            EN : in STD_LOGIC;
 9.            Y : out STD_LOGIC_VECTOR (7 downto 0));
10. end Decoder_3_to_8;
11.
12. architecture Behavioral of Decoder_3_to_8 is
13.    component Decoder_2_to_4
14.    port(
15.    I: in STD_LOGIC_VECTOR;
16.    EN : in STD_LOGIC;
17.    Y : out STD_LOGIC_VECTOR);
18. end component;
19.
20. signal I0,I1 : STD_LOGIC_VECTOR(1 downto 0);
21. signal Y0,Y1 : STD_LOGIC_VECTOR(3 downto 0);
22. signal en0,en1,I2: STD_LOGIC;
23.
24. begin
25.       Decode_2_to_4_0:Decoder_2_to_4
26.       port map(
27.       I => I0,
28.       EN => en0,
29.       Y => Y0);
30.
31.      Decode_2_to_4_1: Decoder_2_to_4
32.      port map(
33.      I => I1,
34.      EN => en1,
35.      Y => Y1);
36.
37.      en0 <=NOT(I(2)) AND EN;
38.      en1 <= I(2) AND EN;
39.      I0 <= I(1 downto 0);
40.      I1 <= I(1 downto 0);
41.      I2 <= I(2);
42.      Y(3 downto 0) <= Y0;
43.      Y(7 downto 4) <= Y1;
44.
45. end Behavioral;
46.
```

1.8.2 Elaborated Design Schematic

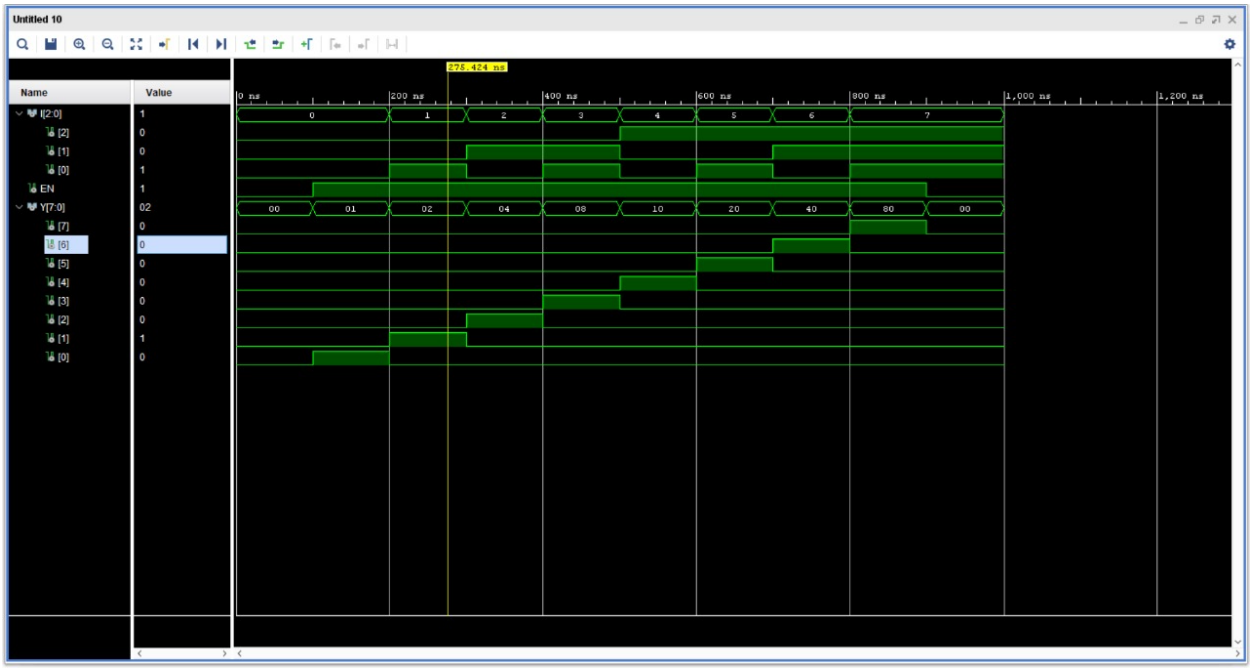


1.8.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_Decoder_3_to_8 is
 5. --
 6. end TB_Decoder_3_to_8;
 7.
 8. architecture Behavioral of TB_Decoder_3_to_8 is
 9.
10.     -- Component Declaration
11.     component Decoder_3_to_8
12.         Port (
13.             I  : in  STD_LOGIC_VECTOR (2 downto 0);
14.             EN : in  STD_LOGIC;
15.             Y  : out STD_LOGIC_VECTOR (7 downto 0)
16.         );
17.     end component;
18.
19.     -- Test Signals
20.     signal I  : STD_LOGIC_VECTOR (2 downto 0) := (others => '0');
21.     signal EN : STD_LOGIC := '0';
22.     signal Y  : STD_LOGIC_VECTOR (7 downto 0);
23.
24. begin
25.
26.     -- Instantiate the Unit Under Test (UUT)
```

```
27.     uut: Decoder_3_to_8 Port Map (
28.         I  => I,
29.         EN => EN,
30.         Y  => Y
31.     );
32. 
33.     -- Stimulus Process
34.     stim_proc: process
35.     begin
36.         -- Test 1: All inputs 0, EN=0 => Y should be 00000000
37.         I <= "000"; EN <= '0'; wait for 100 ns;
38. 
39.         -- Test 2: All inputs 0, EN=1 => Y(0) = '1'
40.         I <= "000"; EN <= '1'; wait for 100 ns;
41. 
42.         -- Test 3: I = 001, EN = 1 => Y(1) = '1'
43.         I <= "001"; wait for 100 ns;
44. 
45.         -- Test 4: I = 010, EN = 1 => Y(2) = '1'
46.         I <= "010"; wait for 100 ns;
47. 
48.         -- Test 5: I = 011, EN = 1 => Y(3) = '1'
49.         I <= "011"; wait for 100 ns;
50. 
51.         -- Test 6: I = 100, EN = 1 => Y(4) = '1'
52.         I <= "100"; wait for 100 ns;
53. 
54.         -- Test 7: I = 101, EN = 1 => Y(5) = '1'
55.         I <= "101"; wait for 100 ns;
56. 
57.         -- Test 8: I = 110, EN = 1 => Y(6) = '1'
58.         I <= "110"; wait for 100 ns;
59. 
60.         -- Test 9: I = 111, EN = 1 => Y(7) = '1'
61.         I <= "111"; wait for 100 ns;
62. 
63.         -- Test 10: EN = 0 => All outputs low again
64.         EN <= '0'; wait for 100 ns;
65. 
66.         -- End simulation
67.         wait;
68.     end process;
69. 
70. end Behavioral;
71. 
```

1.8.4 Timing Diagram



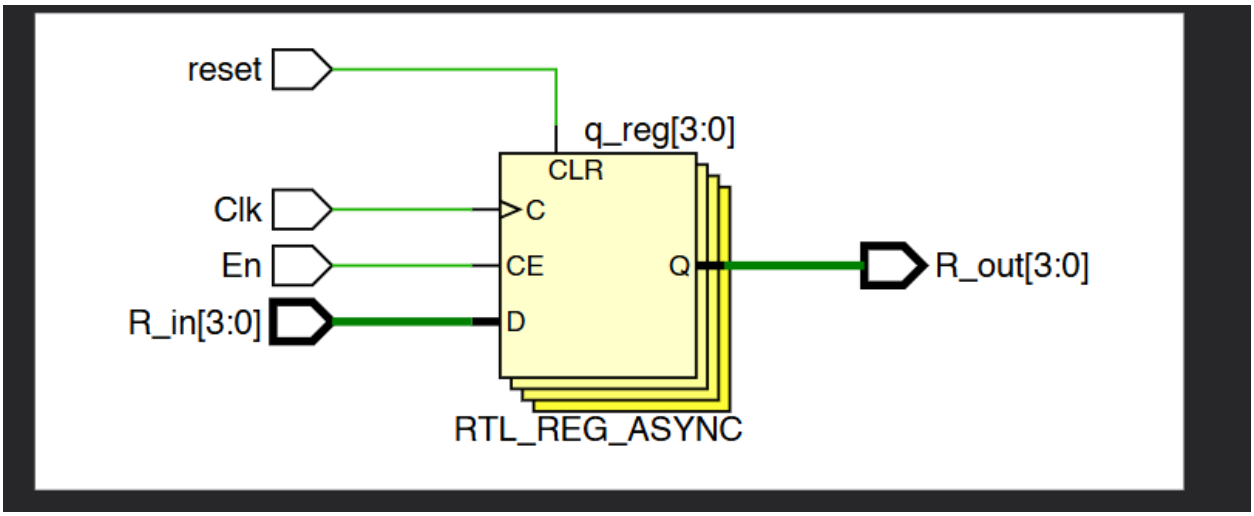
1.9 4 bit Register

1.9.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Register_4_bit is
 5.     Port (
 6.         R_in  : in  STD_LOGIC_VECTOR (3 downto 0);
 7.         En    : in  STD_LOGIC;
 8.         Clk   : in  STD_LOGIC;
 9.         reset : in  STD_LOGIC;
10.         R_out : out STD_LOGIC_VECTOR (3 downto 0)
11.     );
12. end Register_4_bit;
13.
14. architecture Behavioral of Register_4_bit is
15.     signal q : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
16. begin
17.
18.   process (Clk, reset)
19.   begin
20.         if reset = '1' then
21.             q <= (others => '0');  -- Reset output to 0000
22.         elsif rising_edge(Clk) then
```

```
23.             if En = '1' then
24.                 q <= R_in;  -- Load input value when enabled
25.             end if;
26.         end if;
27.     end process;
28.
29.     -- Output assignment
30.     R_out <= q;
31.
32. end Behavioral;
33.
```

1.9.2 Elaborated Design Schematic



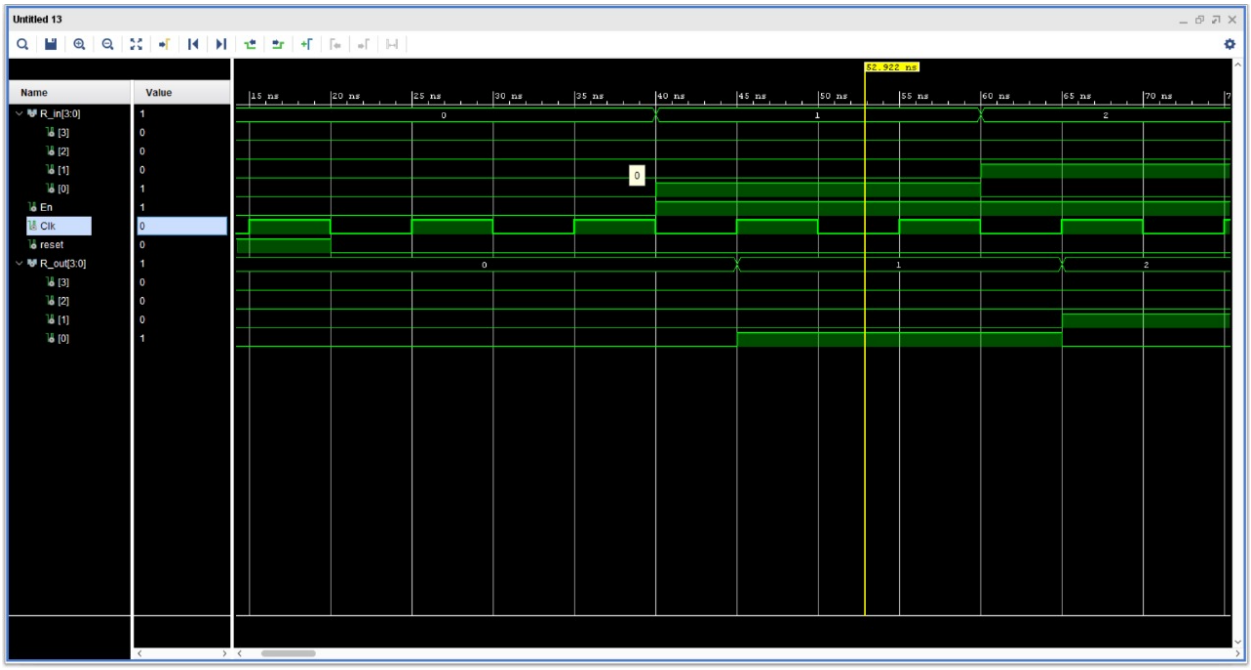
1.9.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_Register_4_bit is
 5. end TB_Register_4_bit;
 6.
 7. architecture Behavioral of TB_Register_4_bit is
 8.
 9.     -- Component Declaration
10.     component Register_4_bit
11.         Port (
12.             R_in  : in  STD_LOGIC_VECTOR (3 downto 0);
13.             En    : in  STD_LOGIC;
```

```
14.             Clk   : in  STD_LOGIC;
15.             reset : in  STD_LOGIC;
16.             R_out : out STD_LOGIC_VECTOR (3 downto 0)
17.         );
18.     end component;
19.
20.     -- Test Signals
21.     signal R_in  : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
22.     signal En    : STD_LOGIC := '0';
23.     signal Clk   : STD_LOGIC := '0';
24.     signal reset : STD_LOGIC := '0';
25.     signal R_out : STD_LOGIC_VECTOR (3 downto 0);
26.
27. begin
28.
29.     -- Instantiate the Unit Under Test (UUT)
30.     uut: Register_4_bit Port Map (
31.         R_in  => R_in,
32.         En    => En,
33.         Clk   => Clk,
34.         reset => reset,
35.         R_out => R_out
36.     );
37.
38.     -- Clock Generation: 10 ns period
39.     clk_process : process
40.     begin
41.         while true loop
42.             Clk <= '0'; wait for 5 ns;
43.             Clk <= '1'; wait for 5 ns;
44.         end loop;
45.     end process;
46.
47.     -- Stimulus Process
48.     stim_proc: process
49.     begin
50.         -- Test 1: Reset active
51.         reset <= '1'; wait for 20 ns;
52.         reset <= '0'; wait for 20 ns;
53.
54.         -- Test 2: En=1, input = "0001"
55.         R_in <= "0001"; En <= '1'; wait for 20 ns;
56.
57.         -- Test 3: En=1, input = "0010"
58.         R_in <= "0010"; wait for 20 ns;
59.
60.         -- Test 4: En=0, input = "0011"
61.         R_in <= "0011"; En <= '0'; wait for 20 ns;
62.
63.         -- Test 5: Reset active again
64.         reset <= '1'; wait for 20 ns;
65.         reset <= '0'; wait for 20 ns;
66.
67.         -- Test 6: En=1, input = "0100"
```

```
68.          R_in <= "0100"; En <= '1'; wait for 20 ns;
69.
70.          -- End simulation
71.          wait;
72.      end process;
73.
74. end Behavioral;
75.
```

1.9.4 Timing Diagram



1.10 Register Bank

1.10.1 VHDL Source Code

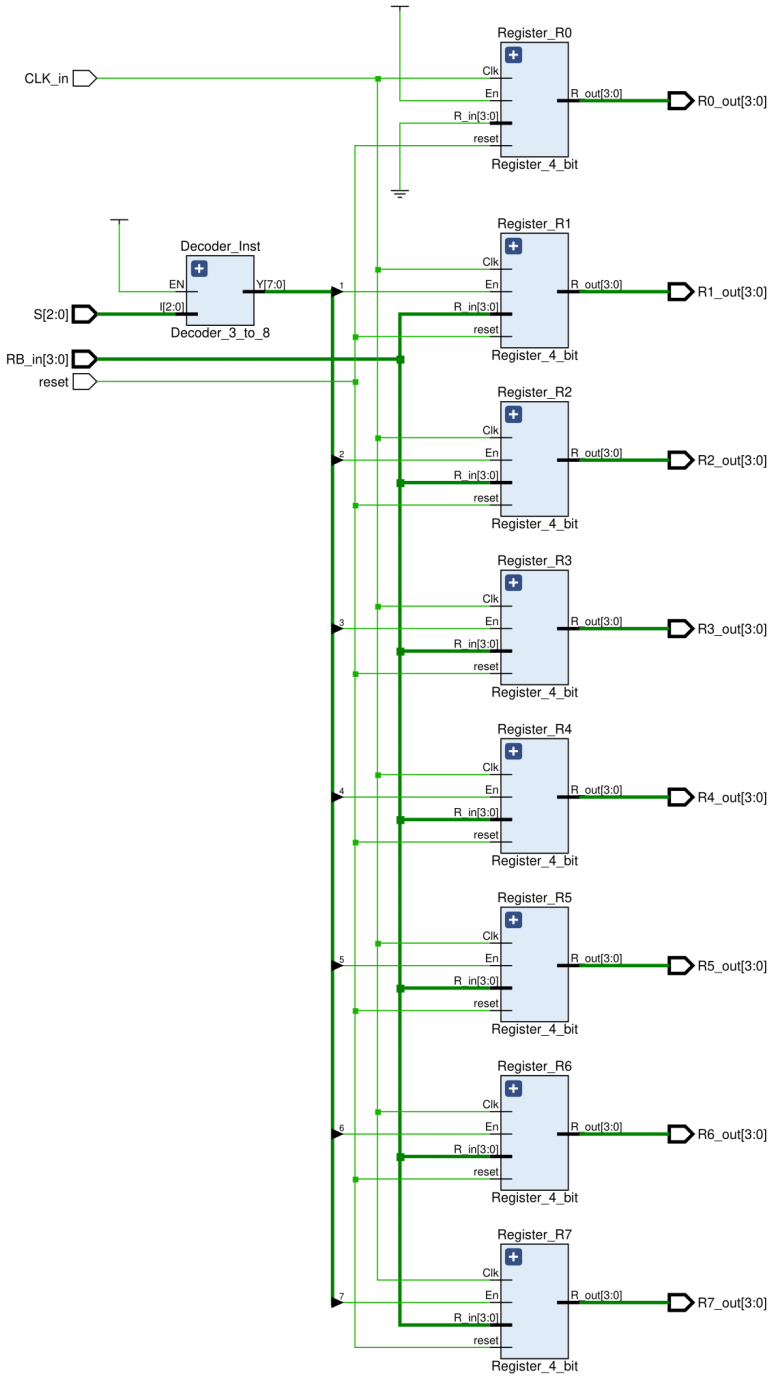
```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Register_Bank is
 5.     Port (
 6.         S       : in STD_LOGIC_VECTOR (2 downto 0);  -- 3-bit select input
 7.         RB_in   : in STD_LOGIC_VECTOR (3 downto 0);  -- 4-bit register input
 8.         CLK_in  : in STD_LOGIC;
 9.         reset   : in STD_LOGIC;                   -- Clock signal
```

```
10.         R0_out  : out STD_LOGIC_VECTOR (3 downto 0);
11.         R1_out  : out STD_LOGIC_VECTOR (3 downto 0);
12.         R2_out  : out STD_LOGIC_VECTOR (3 downto 0);
13.         R3_out  : out STD_LOGIC_VECTOR (3 downto 0);
14.         R4_out  : out STD_LOGIC_VECTOR (3 downto 0);
15.         R5_out  : out STD_LOGIC_VECTOR (3 downto 0);
16.         R6_out  : out STD_LOGIC_VECTOR (3 downto 0);
17.         R7_out  : out STD_LOGIC_VECTOR (3 downto 0)
18.     );
19. end Register_Bank;
20.
21. architecture Behavioral of Register_Bank is
22.
23.     -- 3-to-8 decoder component declaration
24.     component Decoder_3_to_8
25.         Port (
26.             I  : in  STD_LOGIC_VECTOR (2 downto 0);
27.             EN : in  STD_LOGIC;
28.             Y  : out STD_LOGIC_VECTOR (7 downto 0)
29.         );
30.     end component;
31.
32.     -- 4-bit register component declaration
33.     component Register_4_bit
34.         Port (
35.             R_in  : in  STD_LOGIC_VECTOR (3 downto 0);
36.             EN    : in  STD_LOGIC;
37.             CLK   : in  STD_LOGIC;
38.             reset : in  STD_LOGIC;
39.             R_out : out STD_LOGIC_VECTOR (3 downto 0)
40.         );
41.     end component;
42.
43.     -- Internal signal to carry decoder output (8 enable lines)
44.     signal tempEN : STD_LOGIC_VECTOR(7 downto 0);
45.
46. begin
47.
48.     -- Decoder instantiation: decode select signal S into 8 enable lines
49.     Decoder_Inst : Decoder_3_to_8
50.         Port Map (
51.             I  => S,
52.             EN => '1',          -- Decoder always enabled
53.             Y  => tempEN
54.         );
55.
56.     -- R0: always outputs "0000", not controlled by decoder
57.     Register_R0 : Register_4_bit
58.         Port Map (
59.             R_in  => "0000",
60.             EN    => '1',
61.             reset => reset,
62.             CLK   => CLK_in,
63.             R_out => R0_out
```

```
64.         );
65.
66.     -- R1 to R7: controlled by decoder output tempEN(1) to tempEN(7)
67.     Register_R1 : Register_4_bit
68.         Port Map (
69.             R_in  => RB_in,
70.             EN    => tempEN(1),
71.             reset => reset,
72.             CLK   => CLK_in,
73.             R_out => R1_out
74.         );
75.
76.     Register_R2 : Register_4_bit
77.         Port Map (
78.             R_in  => RB_in,
79.             EN    => tempEN(2),
80.             reset => reset,
81.             CLK   => CLK_in,
82.             R_out => R2_out
83.         );
84.
85.     Register_R3 : Register_4_bit
86.         Port Map (
87.             R_in  => RB_in,
88.             EN    => tempEN(3),
89.             reset => reset,
90.             CLK   => CLK_in,
91.             R_out => R3_out
92.         );
93.
94.     Register_R4 : Register_4_bit
95.         Port Map (
96.             R_in  => RB_in,
97.             EN    => tempEN(4),
98.             reset => reset,
99.             CLK   => CLK_in,
100.            R_out => R4_out
101.        );
102.
103.    Register_R5 : Register_4_bit
104.        Port Map (
105.            R_in  => RB_in,
106.            EN    => tempEN(5),
107.            reset => reset,
108.            CLK   => CLK_in,
109.            R_out => R5_out
110.        );
111.
112.    Register_R6 : Register_4_bit
113.        Port Map (
114.            R_in  => RB_in,
115.            EN    => tempEN(6),
116.            reset => reset,
117.            CLK   => CLK_in,
```

```
118.             R_out => R6_out
119.         );
120.
121.     Register_R7 : Register_4_bit
122.         Port Map (
123.             R_in  => RB_in,
124.             EN    => tempEN(7),
125.             reset => reset,
126.             CLK   => CLK_in,
127.             R_out => R7_out
128.         );
129.
130. end Behavioral;
131.
```

1.10.2 Elaborated Design Schematic

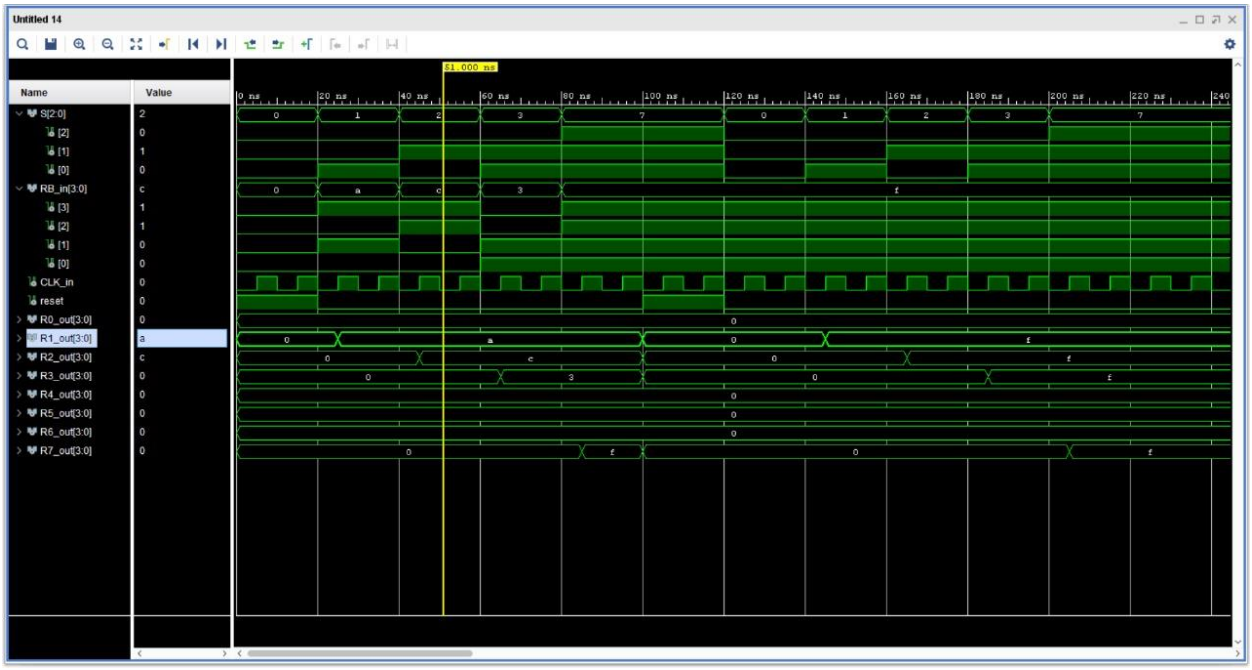


1.10.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_Register_Bank is
 5. end TB_Register_Bank;
 6.
 7. architecture Behavioral of TB_Register_Bank is
 8.
 9.     -- Component Declaration
10.     component Register_Bank
11.         Port (
12.             S       : in STD_LOGIC_VECTOR (2 downto 0);  -- 3-bit select input
13.             RB_in   : in STD_LOGIC_VECTOR (3 downto 0);  -- 4-bit register input
14.             CLK_in  : in STD_LOGIC;
15.             reset   : in STD_LOGIC;
16.             R0_out  : out STD_LOGIC_VECTOR (3 downto 0);
17.             R1_out  : out STD_LOGIC_VECTOR (3 downto 0);
18.             R2_out  : out STD_LOGIC_VECTOR (3 downto 0);
19.             R3_out  : out STD_LOGIC_VECTOR (3 downto 0);
20.             R4_out  : out STD_LOGIC_VECTOR (3 downto 0);
21.             R5_out  : out STD_LOGIC_VECTOR (3 downto 0);
22.             R6_out  : out STD_LOGIC_VECTOR (3 downto 0);
23.             R7_out  : out STD_LOGIC_VECTOR (3 downto 0)
24.         );
25.     end component;
26.
27.     -- Signals for simulation
28.     signal S       : STD_LOGIC_VECTOR (2 downto 0) := "000";
29.     signal RB_in   : STD_LOGIC_VECTOR (3 downto 0) := "0000";
30.     signal CLK_in  : STD_LOGIC := '0';
31.     signal reset   : STD_LOGIC := '0';
32.     signal R0_out  : STD_LOGIC_VECTOR (3 downto 0);
33.     signal R1_out  : STD_LOGIC_VECTOR (3 downto 0);
34.     signal R2_out  : STD_LOGIC_VECTOR (3 downto 0);
35.     signal R3_out  : STD_LOGIC_VECTOR (3 downto 0);
36.     signal R4_out  : STD_LOGIC_VECTOR (3 downto 0);
37.     signal R5_out  : STD_LOGIC_VECTOR (3 downto 0);
38.     signal R6_out  : STD_LOGIC_VECTOR (3 downto 0);
39.     signal R7_out  : STD_LOGIC_VECTOR (3 downto 0);
40.
41. begin
42.
43.     -- Instantiate the Unit Under Test (UUT)
44.     UUT: Register_Bank
45.         Port Map (
46.             S       => S,
47.             RB_in   => RB_in,
48.             CLK_in  => CLK_in,
49.             reset   => reset,
50.             R0_out  => R0_out,
```

```
51.             R1_out  => R1_out,
52.             R2_out  => R2_out,
53.             R3_out  => R3_out,
54.             R4_out  => R4_out,
55.             R5_out  => R5_out,
56.             R6_out  => R6_out,
57.             R7_out  => R7_out
58.         );
59.
60.     -- Clock Generation: 10 ns period
61.     clk_process : process
62.     begin
63.         while true loop
64.             CLK_in <= '0'; wait for 5 ns;
65.             CLK_in <= '1'; wait for 5 ns;
66.         end loop;
67.     end process;
68.
69.     -- Stimulus Process
70.     stim_proc: process
71.     begin
72.         -- Reset the system
73.         reset <= '1'; wait for 20 ns;
74.         reset <= '0';
75.
76.         -- Write to Register R1
77.         S <= "001"; RB_in <= "1010"; wait for 20 ns;
78.
79.         -- Write to Register R2
80.         S <= "010"; RB_in <= "1100"; wait for 20 ns;
81.
82.         -- Write to Register R3
83.         S <= "011"; RB_in <= "0011"; wait for 20 ns;
84.
85.         -- Write to Register R7
86.         S <= "111"; RB_in <= "1111"; wait for 20 ns;
87.
88.         -- Activate reset
89.         reset <= '1'; wait for 20 ns;
90.         reset <= '0';
91.
92.         -- Check the contents of all registers
93.         S <= "000"; wait for 20 ns;
94.         S <= "001"; wait for 20 ns;
95.         S <= "010"; wait for 20 ns;
96.         S <= "011"; wait for 20 ns;
97.         S <= "111"; wait for 20 ns;
98.
99.         -- End simulation
100.        wait;
101.    end process;
102.
103. end Behavioral;
```

1.10.4 Timing Diagram



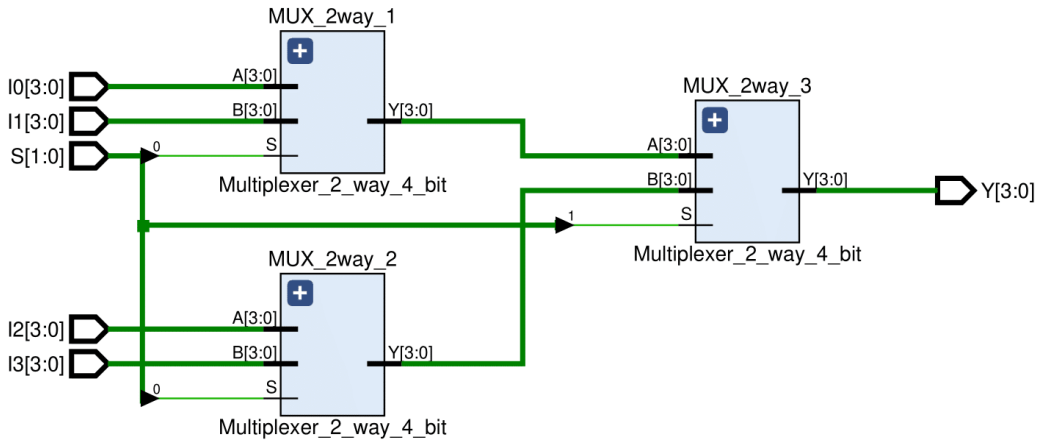
1.11 4 way 4 bit Multiplexer

1.11.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Multiplexer_4_way_4_bit  is
 5.     Port ( S : in STD_LOGIC_VECTOR (1 downto 0);
 6.            I0 : in STD_LOGIC_VECTOR (3 downto 0);
 7.            I1 : in STD_LOGIC_VECTOR (3 downto 0);
 8.            I2 : in STD_LOGIC_VECTOR (3 downto 0);
 9.            I3 : in STD_LOGIC_VECTOR (3 downto 0);
10.            Y : out STD_LOGIC_VECTOR (3 downto 0));
11. end Multiplexer_4_way_4_bit ;
12.
13. architecture Behavioral of Multiplexer_4_way_4_bit  is
14. component Multiplexer_2_way_4_bit
15.      Port( S : in STD_LOGIC;
16.            A : in STD_LOGIC_VECTOR (3 downto 0);
17.            B : in STD_LOGIC_VECTOR (3 downto 0);
18.            Y : out STD_LOGIC_VECTOR (3 downto 0));
19. end component;
20. signal y0, y1: STD_LOGIC_VECTOR (3 downto 0);
```

```
21.
22. begin
23.
24. MUX_2way_1 : Multiplexer_2_way_4_bit
25.     port map( A => I0,
26.           B => I1,
27.           S => S(0),
28.           Y => y0);
29.
30. MUX_2way_2 : Multiplexer_2_way_4_bit
31.     port map( A => I2,
32.           B => I3,
33.           S => S(0),
34.           Y => y1);
35.
36. MUX_2way_3 : Multiplexer_2_way_4_bit
37.     port map(A => y0,
38.           B => y1,
39.           S => S(1),
40.           Y => Y);
41.
42. end Behavioral;
43.
```

1.11.2 Elaborated Design Schematic

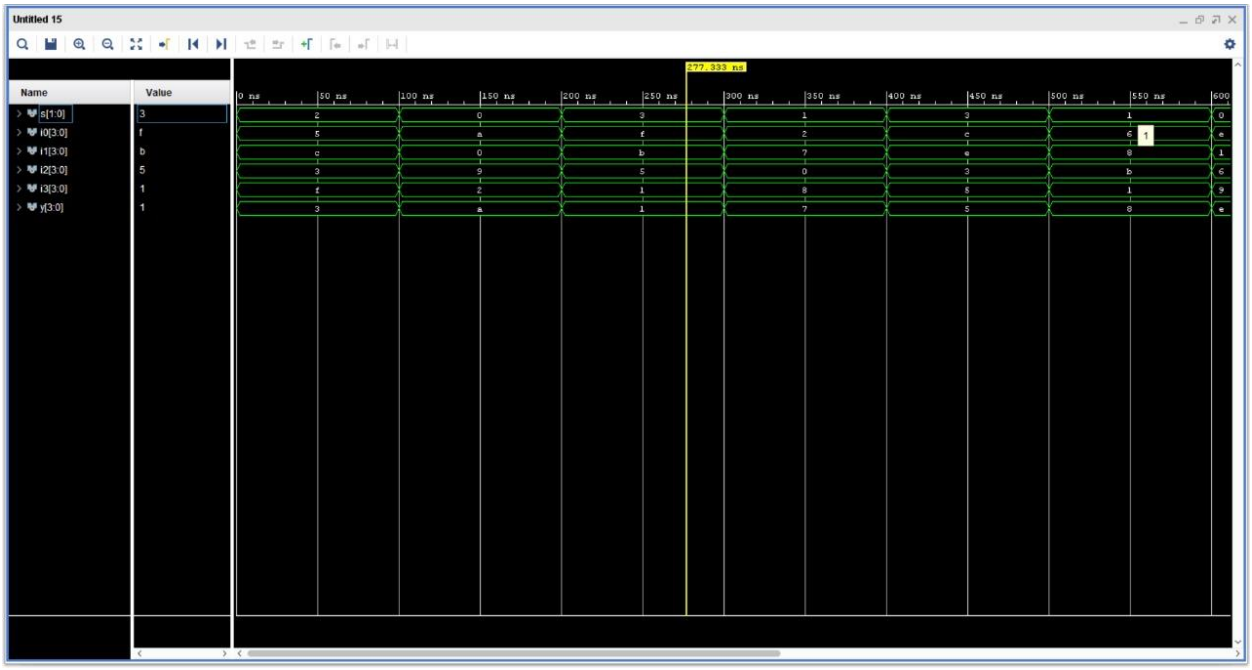


1.11.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4.
 5.
 6. entity TB_Multiplexer_4_way_4_bit is
 7. --  Port ( );
 8. end TB_Multiplexer_4_way_4_bit;
 9.
10. architecture Behavioral of TB_Multiplexer_4_way_4_bit is
11. component Multiplexer_4_way_4_bit
12.     port(S : in STD_LOGIC_VECTOR (1 downto 0);
13.            I0 : in STD_LOGIC_VECTOR (3 downto 0);
14.            I1 : in STD_LOGIC_VECTOR (3 downto 0);
15.            I2 : in STD_LOGIC_VECTOR (3 downto 0);
16.            I3 : in STD_LOGIC_VECTOR (3 downto 0);
17.            Y : out STD_LOGIC_VECTOR (3 downto 0));
18. end component;
19. signal s: STD_LOGIC_VECTOR (1 downto 0);
20. signal i0, i1, i2, i3, y: STD_LOGIC_VECTOR (3 downto 0);
21.
22. begin
23.
24. UUT: Multiplexer_4_way_4_bit PORT MAP(
25.     S => s,
26.     I0 => i0,
27.     I1 => i1,
28.     I2 => i2,
29.     I3 => i3,
30.     Y => y);
31.
32. process begin
33.
34.     -- Test 1
35.     I0 <= "0101"; I1 <= "1100"; I2 <= "0011"; I3 <= "1111"; S <= "10"; wait for
100 ns;
36.
37.     -- Test 2
38.     I0 <= "1010"; I1 <= "0000"; I2 <= "1001"; I3 <= "0010"; S <= "00"; wait for
100 ns;
39.
40.     -- Test 3
41.     I0 <= "1111"; I1 <= "1011"; I2 <= "0101"; I3 <= "0001"; S <= "11"; wait for
100 ns;
42.
43.     -- Test 4
44.     I0 <= "0010"; I1 <= "0111"; I2 <= "0000"; I3 <= "1000"; S <= "01"; wait for
100 ns;
45.
46.     -- Test 5
```

```
47.     I0 <= "1100"; I1 <= "1110"; I2 <= "0011"; I3 <= "0101"; S <= "11"; wait for
100 ns;
48.
49.     -- Test 6
50.     I0 <= "0110"; I1 <= "1000"; I2 <= "1011"; I3 <= "0001"; S <= "01"; wait for
100 ns;
51.
52.     -- Test 7
53.     I0 <= "1110"; I1 <= "0001"; I2 <= "0110"; I3 <= "1001"; S <= "00"; wait for
100 ns;
54.
55.     -- Test 8
56.     I0 <= "0000"; I1 <= "1101"; I2 <= "0100"; I3 <= "1010"; S <= "10"; wait for
100 ns;
57.
58.     -- Test 9
59.     I0 <= "0011"; I1 <= "1111"; I2 <= "0001"; I3 <= "0111"; S <= "11"; wait for
100 ns;
60.
61.     -- Test 10
62.     I0 <= "1011"; I1 <= "0110"; I2 <= "1001"; I3 <= "1100"; S <= "01"; wait for
100 ns;
63.
64.
65.
66.     -- Finish simulation
67.     wait;
68. end process;
69.
70.
71. end Behavioral;
72.
```

1.11.4 Timing Diagram



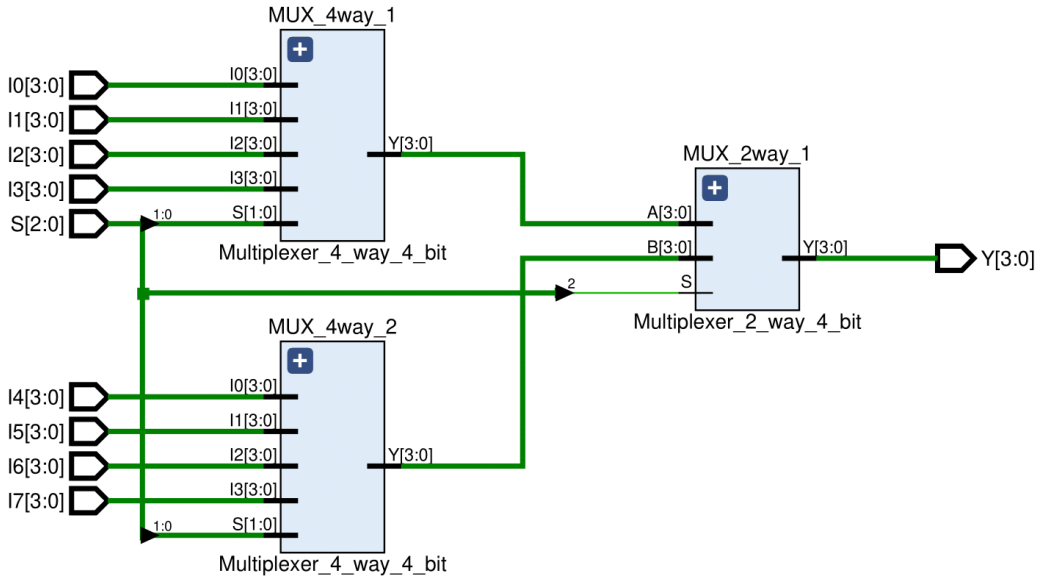
1.12 8 way 4 bit Multiplexer

1.12.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4.
 5.
 6. entity Multiplexer_8_way_4_bit is
 7.     Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
 8.            I1 : in STD_LOGIC_VECTOR (3 downto 0);
 9.            I2 : in STD_LOGIC_VECTOR (3 downto 0);
10.            I3 : in STD_LOGIC_VECTOR (3 downto 0);
11.            I4 : in STD_LOGIC_VECTOR (3 downto 0);
12.            I5 : in STD_LOGIC_VECTOR (3 downto 0);
13.            I6 : in STD_LOGIC_VECTOR (3 downto 0);
14.            I7 : in STD_LOGIC_VECTOR (3 downto 0);
15.            I0 : in STD_LOGIC_VECTOR (3 downto 0);
16.            Y : out STD_LOGIC_VECTOR (3 downto 0));
17. end Multiplexer_8_way_4_bit;
18.
19. architecture Behavioral of Multiplexer_8_way_4_bit is
20. component Multiplexer_4_way_4_bit
```

```
21.     Port ( S : in STD_LOGIC_VECTOR (1 downto 0);
22.        I0 : in STD_LOGIC_VECTOR (3 downto 0);
23.        I1 : in STD_LOGIC_VECTOR (3 downto 0);
24.        I2 : in STD_LOGIC_VECTOR (3 downto 0);
25.        I3 : in STD_LOGIC_VECTOR (3 downto 0);
26.        Y : out STD_LOGIC_VECTOR (3 downto 0));
27. end component;
28.
29. component Multiplexer_2_way_4_bit
30.      Port( S : in STD_LOGIC;
31.      A : in STD_LOGIC_VECTOR (3 downto 0);
32.      B : in STD_LOGIC_VECTOR (3 downto 0);
33.      Y : out STD_LOGIC_VECTOR (3 downto 0));
34. end component;
35.
36. signal y0, y1 : STD_LOGIC_VECTOR (3 downto 0);
37.
38. begin
39.
40. MUX_4way_1 : Multiplexer_4_way_4_bit
41.     port map (I0 => I0,
42.               I1 => I1,
43.               I2 => I2,
44.               I3 => I3,
45.               S(0) => S(0),
46.               S(1) => S(1),
47.               Y => y0);
48.
49. MUX_4way_2 : Multiplexer_4_way_4_bit
50.     port map (I0 => I4,
51.               I1 => I5,
52.               I2 => I6,
53.               I3 => I7,
54.               S(0) => S(0),
55.               S(1) => S(1),
56.               Y => y1);
57.
58. MUX_2way_1 : Multiplexer_2_way_4_bit
59.     port map(A => y0,
60.              B => y1,
61.              S => S(2),
62.              Y => Y);
63.
64. end Behavioral;
65.
```

1.12.2 Elaborated Design Schematic



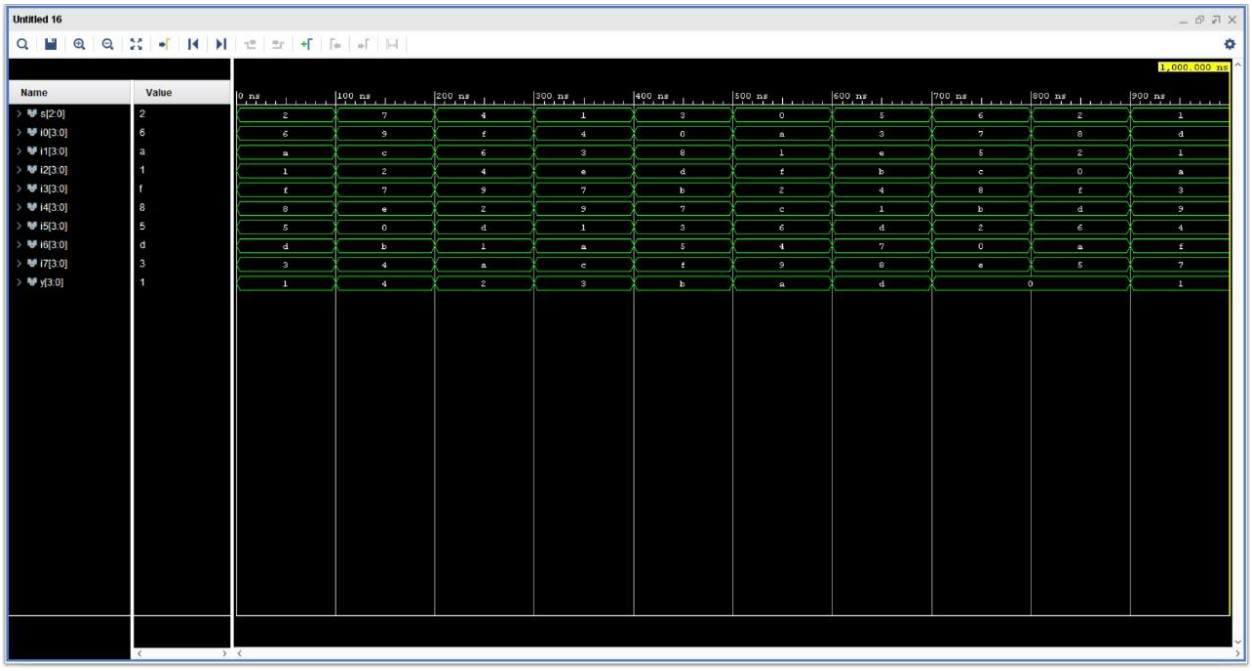
1.12.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4.
 5.
 6. entity TB_Multiplexer_8_way_4_bit is
 7. --  Port ( );
 8. end TB_Multiplexer_8_way_4_bit;
 9.
10. architecture Behavioral of TB_Multiplexer_8_way_4_bit is
11. component Multiplexer_8_way_4_bit
12.
13.     Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
14.            I1 : in STD_LOGIC_VECTOR (3 downto 0);
15.            I2 : in STD_LOGIC_VECTOR (3 downto 0);
16.            I3 : in STD_LOGIC_VECTOR (3 downto 0);
17.            I4 : in STD_LOGIC_VECTOR (3 downto 0);
18.            I5 : in STD_LOGIC_VECTOR (3 downto 0);
19.            I6 : in STD_LOGIC_VECTOR (3 downto 0);
20.            I7 : in STD_LOGIC_VECTOR (3 downto 0);
```

```
21.            I0 : in STD_LOGIC_VECTOR (3 downto 0);
22.            Y : out STD_LOGIC_VECTOR (3 downto 0));
23. end component;
24.
25. signal s: STD_LOGIC_VECTOR (2 downto 0);
26. signal i0, i1, i2, i3, i4, i5, i6, i7, y: STD_LOGIC_VECTOR (3 downto 0);
27.
28. begin
29.
30. UUT: Multiplexer_8_way_4_bit PORT MAP(
31.     S => s,
32.     I0 => i0,
33.     I1 => i1,
34.     I2 => i2,
35.     I3 => i3,
36.     I4 => i4,
37.     I5 => i5,
38.     I6 => i6,
39.     I7 => i7,
40.     Y => y);
41.
42. process begin
43.
44.     -- Test 1
45.     I0 <= "0110"; I1 <= "1010"; I2 <= "0001"; I3 <= "1111";
46.     I4 <= "1000"; I5 <= "0101"; I6 <= "1101"; I7 <= "0011";
47.     S <= "010"; wait for 100 ns;
48.
49.     -- Test 2
50.     I0 <= "1001"; I1 <= "1100"; I2 <= "0010"; I3 <= "0111";
51.     I4 <= "1110"; I5 <= "0000"; I6 <= "1011"; I7 <= "0100";
52.     S <= "111"; wait for 100 ns;
53.
54.     -- Test 3
55.     I0 <= "1111"; I1 <= "0110"; I2 <= "0100"; I3 <= "1001";
56.     I4 <= "0010"; I5 <= "1101"; I6 <= "0001"; I7 <= "1010";
57.     S <= "100"; wait for 100 ns;
58.
59.     -- Test 4
60.     I0 <= "0100"; I1 <= "0011"; I2 <= "1110"; I3 <= "0111";
61.     I4 <= "1001"; I5 <= "0001"; I6 <= "1010"; I7 <= "1100";
62.     S <= "001"; wait for 100 ns;
63.
64.     -- Test 5
65.     I0 <= "0000"; I1 <= "1000"; I2 <= "1101"; I3 <= "1011";
66.     I4 <= "0111"; I5 <= "0011"; I6 <= "0101"; I7 <= "1111";
67.     S <= "011"; wait for 100 ns;
68.
69.     -- Test 6
70.     I0 <= "1010"; I1 <= "0001"; I2 <= "1111"; I3 <= "0010";
71.     I4 <= "1100"; I5 <= "0110"; I6 <= "0100"; I7 <= "1001";
72.     S <= "000"; wait for 100 ns;
73.
74.     -- Test 7
```

```
75.      I0 <= "0011"; I1 <= "1110"; I2 <= "1011"; I3 <= "0100";
76.      I4 <= "0001"; I5 <= "1101"; I6 <= "0111"; I7 <= "1000";
77.      S <= "101"; wait for 100 ns;
78.
79.      -- Test 8
80.      I0 <= "0111"; I1 <= "0101"; I2 <= "1100"; I3 <= "1000";
81.      I4 <= "1011"; I5 <= "0010"; I6 <= "0000"; I7 <= "1110";
82.      S <= "110"; wait for 100 ns;
83.
84.      -- Test 9
85.      I0 <= "1000"; I1 <= "0010"; I2 <= "0000"; I3 <= "1111";
86.      I4 <= "1101"; I5 <= "0110"; I6 <= "1010"; I7 <= "0101";
87.      S <= "010"; wait for 100 ns;
88.
89.      -- Test 10
90.      I0 <= "1101"; I1 <= "0001"; I2 <= "1010"; I3 <= "0011";
91.      I4 <= "1001"; I5 <= "0100"; I6 <= "1111"; I7 <= "0111";
92.      S <= "001"; wait for 100 ns;
93.
94.  end process;
95.
96.
97.  end Behavioral;
98.
```

1.12.4 Timing Diagram

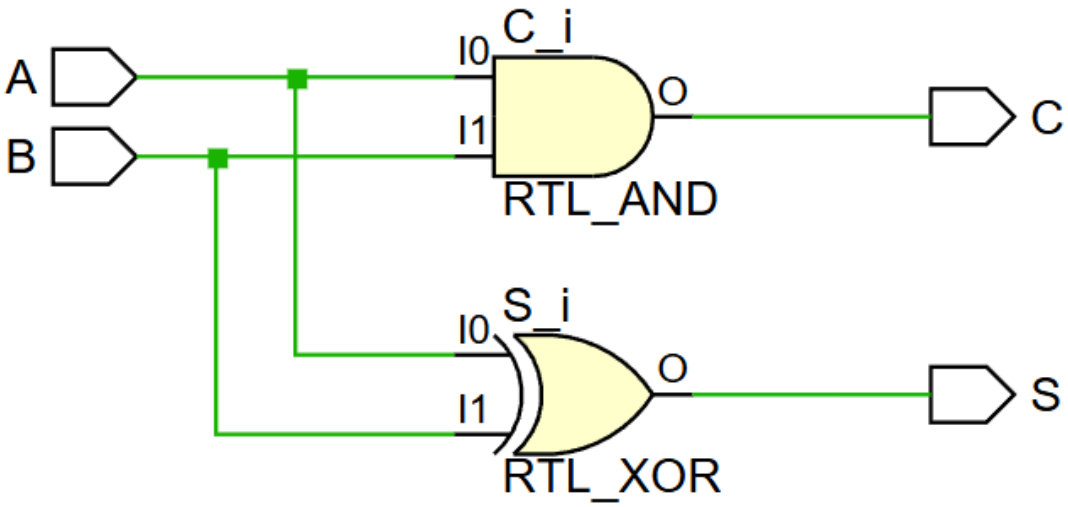


1.13 Half Adder

1.13.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Half_Adder is
 5.     Port ( A : in STD_LOGIC;
 6.            B : in STD_LOGIC;
 7.            S : out STD_LOGIC;
 8.            C : out STD_LOGIC);
 9. end Half_Adder;
10.
11. architecture Behavioral of Half_Adder is
12.
13. begin
14.
15. S <= A XOR B;
16. C <= A AND B;
17.
18. end Behavioral;
19.
```

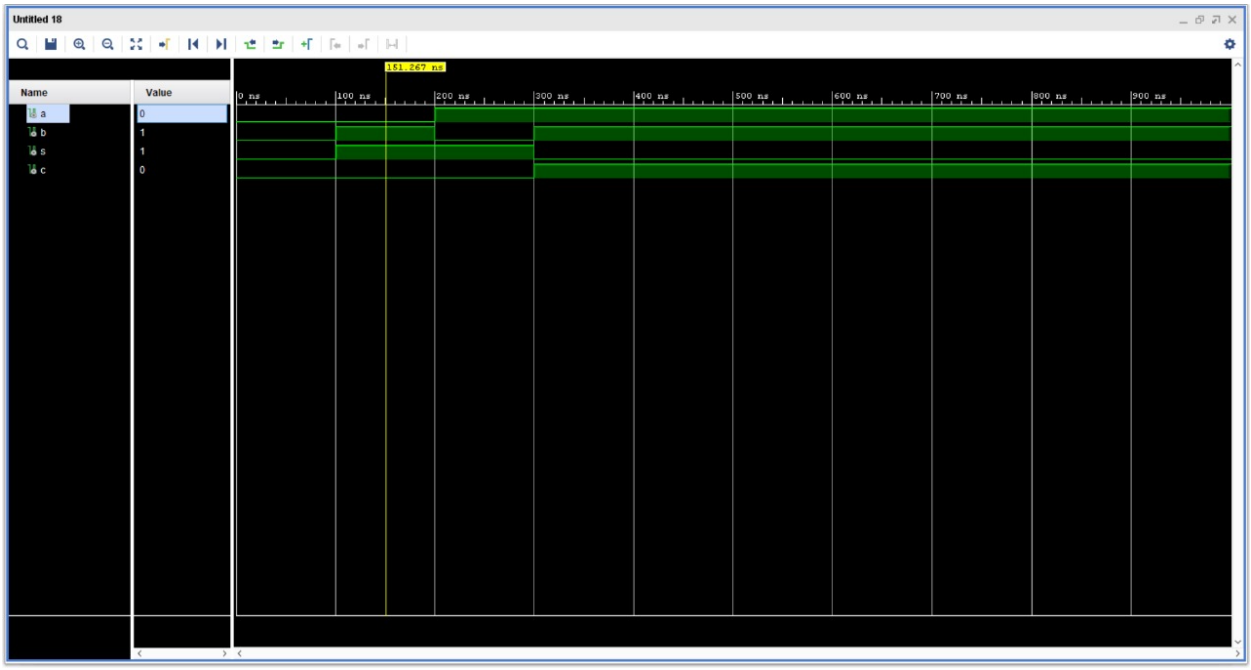
1.13.2 Elaborated Design Schematic



1.13.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4.
 5. entity TB_Half_Adder is
 6. --  Port ( );
 7. end TB_Half_Adder ;
 8.
 9. architecture Behavioral of TB_Half_Adder  is
10. component Half_Adder
11.     port (
12.     A,B : in std_logic;
13.     S,C : out std_logic);
14. end component;
15.
16. signal a,b: std_logic;
17. signal s,c: std_logic;
18.
19. begin
20.
21. UUT: Half_Adder PORT MAP(
22.     A => a,
23.     B => b,
24.     S => s,
25.     C => c);
26.
27. process begin
28.     a <= '0';
29.     b <= '0';
30.     WAIT FOR 100 ns;
31.
32.     b <= '1';
33.     WAIT FOR 100 ns;
34.
35.     b <= '0';
36.     a <= '1';
37.     WAIT FOR 100 ns;
38.
39.     b <= '1';
40.
41.     WAIT;
42. end process;
43.
44.
45.
46. end Behavioral;
```

1.13.4 Timing Diagram



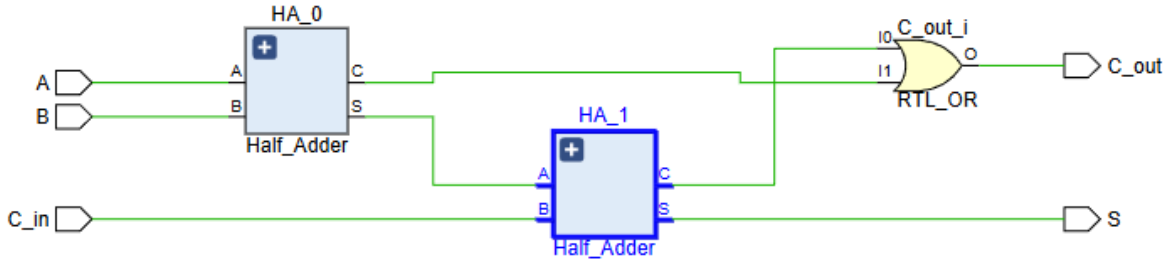
1.14 Full Adder

1.14.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Full_Adder is
 5.     Port ( A : in STD_LOGIC;
 6.            B : in STD_LOGIC;
 7.            C_in : in STD_LOGIC;
 8.            S : out STD_LOGIC;
 9.            C_out : out STD_LOGIC);
10. end Full_Adder;
11.
12. architecture Behavioral of Full_Adder is
13. component Half_Adder
14.     Port (
15.     A : in STD_LOGIC;
16.     B : in STD_LOGIC;
17.     S : out STD_LOGIC;
18.     C : out STD_LOGIC);
```

```
19. end component;
20.
21. signal HA0_S, HA1_S, HA0_C, HA1_C :STD_logic;
22.
23. begin
24.
25. HA_0 : Half_Adder
26.     port map(
27.     A => A,
28.     B => B,
29.     S => HA0_S,
30.     C => HA0_C);
31.
32. HA_1 : Half_Adder
33.     port map(
34.     A => HA0_S,
35.     B => C_in,
36.     S => HA1_S,
37.     C => HA1_C);
38.
39. S <= HA1_S;
40. C_out <= HA1_C OR HA0_C;
41.
42. end Behavioral;
43.
```

1.14.2 Elaborated Design Schematic

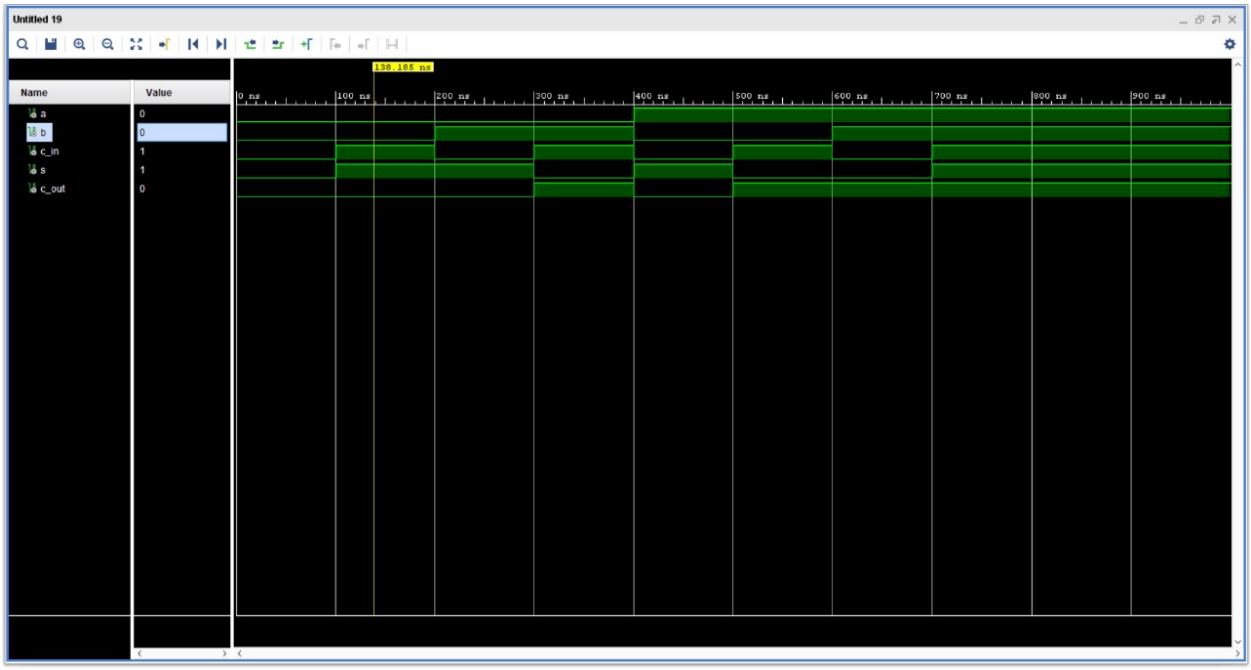


1.14.3 Simulation Source Code

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity TB_Full_Adder is
5. --  Port ( );
6. end TB_Full_Adder;
7.
```

```
 8. architecture Behavioral of TB_Full_Adder is
 9. component Full_Adder
10.     port (
11.     A,B,C_in : in STD_logic;
12.     S, C_out : out STD_logic);
13. end component;
14.
15. signal a, b, c_in, s, c_out : std_logic;
16.
17. begin
18.
19. UUT: Full_Adder PORT MAP (
20.     A => a,
21.     B => b,
22.     C_in => c_in,
23.     S => s,
24.     C_out => c_out);
25.
26. process begin
27.     a <= '0';
28.     b <= '0';
29.     c_in <= '0';
30.     WAIT FOR 100 ns;
31.
32.
33.     c_in <= '1';
34.     WAIT FOR 100 ns;
35.
36.     b <= '1';
37.     c_in <= '0';
38.     WAIT FOR 100 ns;
39.
40.     c_in <= '1';
41.     WAIT FOR 100 ns;
42.
43.     a <= '1';
44.     b <= '0';
45.     c_in <= '0';
46.     WAIT FOR 100 ns;
47.
48.     c_in <= '1';
49.     WAIT FOR 100 ns;
50.
51.     b <= '1';
52.     c_in <= '0';
53.     WAIT FOR 100 ns;
54.
55.     c_in <= '1';
56.     WAIT;
57. end process;
58.
59. end Behavioral;
60.
```

1.14.4 Timing Diagram



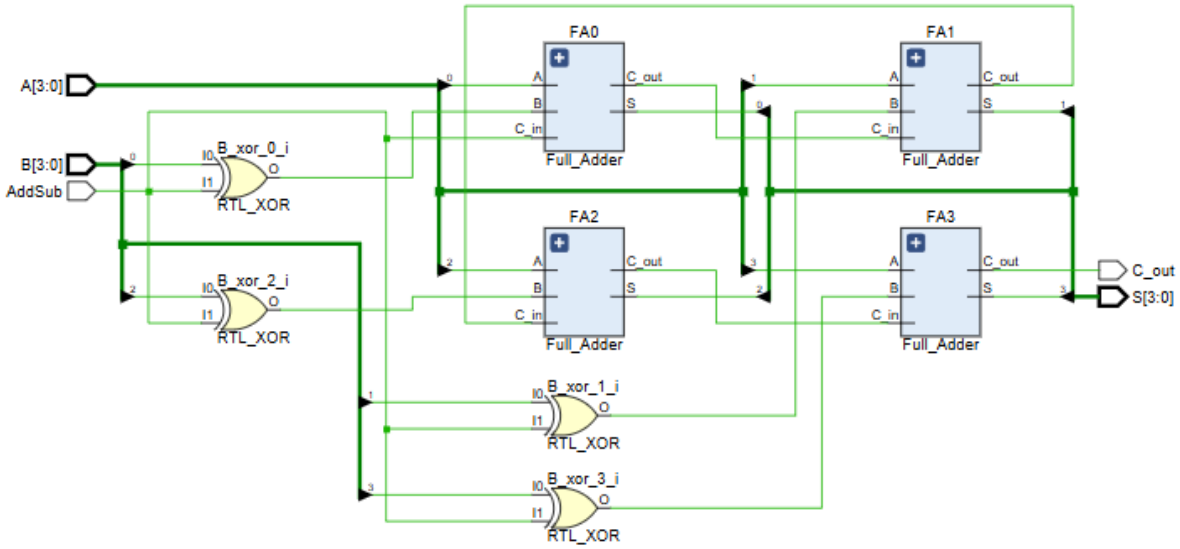
1.15 Ripple Carry Adder and Subtractor

1.15.1 VHDL Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity RCAS_4 is
 5.     Port (
 6.         A      : in  STD_LOGIC_VECTOR(3 downto 0);
 7.         B      : in  STD_LOGIC_VECTOR(3 downto 0);
 8.         AddSub : in  STD_LOGIC;  -- 0 for Add, 1 for Sub
 9.         S      : out STD_LOGIC_VECTOR(3 downto 0);
10.         C_out  : out STD_LOGIC
11.     );
12. end RCAS_4;
13.
14. architecture Behavioral of RCAS_4 is
15.     component Full_Adder
16.         Port (
17.             A     : in  STD_LOGIC;
18.             B     : in  STD_LOGIC;
19.             C_in  : in  STD_LOGIC;
20.             S     : out STD_LOGIC;
```

```
21.             C_out : out STD_LOGIC
22.         );
23. end component;
24.
25. signal B_xor     : STD_LOGIC_VECTOR(3 downto 0);
26. signal C         : STD_LOGIC_VECTOR(2 downto 0);
27.
28. begin
29.     -- XOR B with AddSub to perform subtraction when AddSub = 1
30.     B_xor(0) <= B(0) xor AddSub;
31.     B_xor(1) <= B(1) xor AddSub;
32.     B_xor(2) <= B(2) xor AddSub;
33.     B_xor(3) <= B(3) xor AddSub;
34.
35.     -- Ripple Carry Adder Chain
36.     FA0: Full_Adder port map (A => A(0), B => B_xor(0), C_in => AddSub,    S =>
S(0), C_out => C(0));
37.     FA1: Full_Adder port map (A => A(1), B => B_xor(1), C_in => C(0),      S =>
S(1), C_out => C(1));
38.     FA2: Full_Adder port map (A => A(2), B => B_xor(2), C_in => C(1),      S =>
S(2), C_out => C(2));
39.     FA3: Full_Adder port map (A => A(3), B => B_xor(3), C_in => C(2),      S =>
S(3), C_out => C_out);
40.
41. end Behavioral;
42.
```

1.15.2 Elaborated Design Schematic

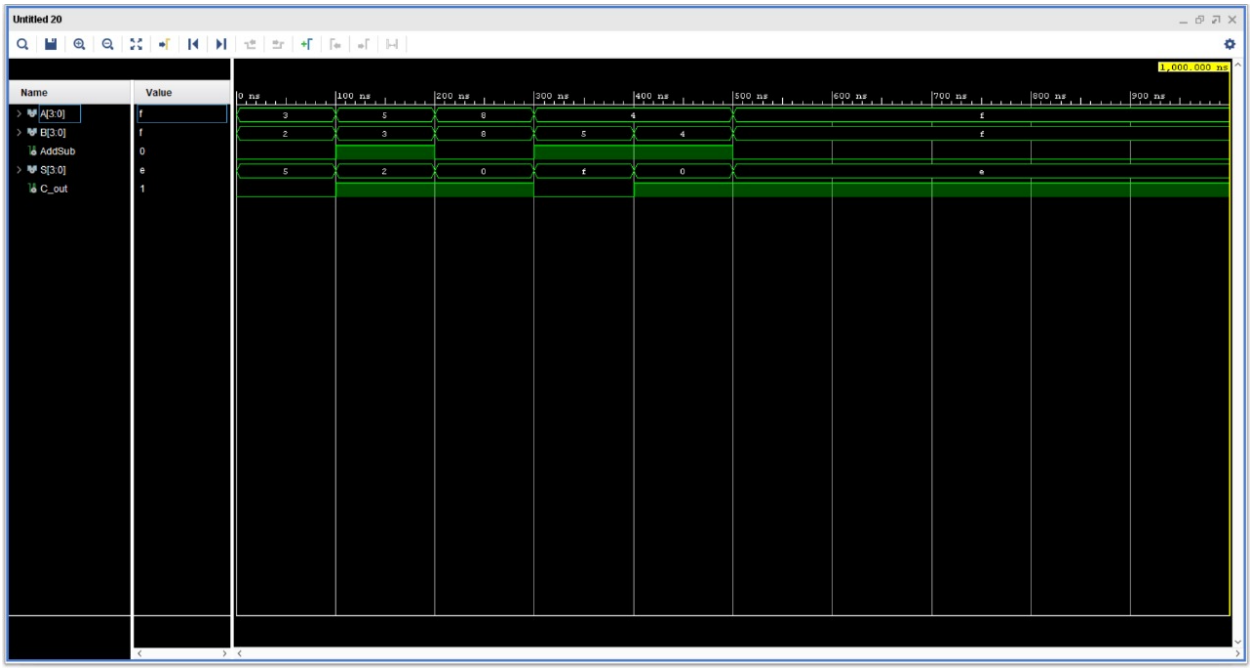


1.15.3 Simulation Source Code

```
 1.   library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_RCAS_4 is
 5. end TB_RCAS_4;
 6.
 7. architecture Behavioral of TB_RCAS_4 is
 8.
 9.     -- Component Declaration
10.     component RCAS_4
11.         Port (
12.             A      : in  STD_LOGIC_VECTOR(3 downto 0);
13.             B      : in  STD_LOGIC_VECTOR(3 downto 0);
14.             AddSub : in  STD_LOGIC;
15.             S      : out STD_LOGIC_VECTOR(3 downto 0);
16.             C_out  : out STD_LOGIC
17.         );
18.     end component;
19.
20.     -- Test Signals
21.     signal A, B : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');
22.     signal AddSub : STD_LOGIC := '0';
23.     signal S : STD_LOGIC_VECTOR(3 downto 0);
24.     signal C_out : STD_LOGIC;
25.
26. begin
27.
28.     -- Instantiate the Unit Under Test (UUT)
29.     uut: RCAS_4 Port Map (
30.         A => A,
31.         B => B,
32.         AddSub => AddSub,
33.         S => S,
34.         C_out => C_out
35.     );
36.
37.     -- Stimulus Process
38.     stim_proc: process
39.     begin
40.         -- Test 1: Addition 3 + 2
41.         A <= "0011"; B <= "0010"; AddSub <= '0'; wait for 100 ns;
42.
43.         -- Test 2: Subtraction 5 - 3
44.         A <= "0101"; B <= "0011"; AddSub <= '1'; wait for 100 ns;
45.
46.         -- Test 3: Addition with carry 8 + 8
47.         A <= "1000"; B <= "1000"; AddSub <= '0'; wait for 100 ns;
48.
49.         -- Test 4: Subtraction with borrow 4 - 5
50.         A <= "0100"; B <= "0101"; AddSub <= '1'; wait for 100 ns;
```

```
51.
52.         -- Test 5: Zero result 4 - 4
53.         A <= "0100"; B <= "0100"; AddSub <= '1'; wait for 100 ns;
54.
55.         -- Test 6: Maximum addition 15 + 15
56.         A <= "1111"; B <= "1111"; AddSub <= '0'; wait for 100 ns;
57.
58.         -- End simulation
59.         wait;
60.     end process;
61.
62. end Behavioral;
63.
```

1.15.4 Timing Diagram



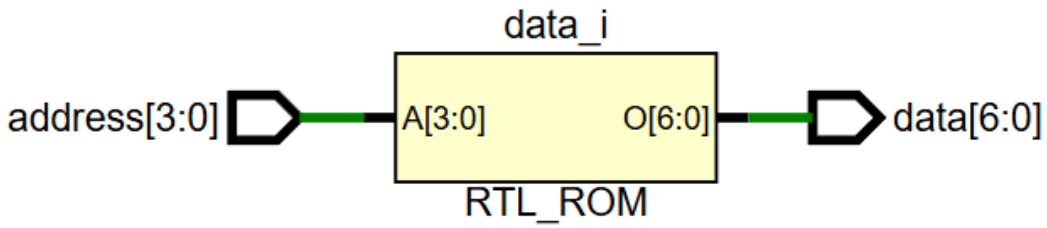
1.16 Seven Segment Display

1.16.1 VHDL Source Code

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.NUMERIC_STD.ALL;
4.
5. entity LUT_16_7 is
```

```
 6.     Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
 7.            data : out STD_LOGIC_VECTOR (6 downto 0));
 8. end LUT_16_7;
 9.
10. architecture Behavioral of LUT_16_7 is
11.
12. type rom_type is array (0 to 15) of std_logic_vector(6 downto 0);
13. signal sevenSegment_ROM : rom_type := (
14.     "1000000", -- 0
15.     "1111001", -- 1
16.     "0100100", -- 2
17.     "0110000", -- 3
18.     "0011001", -- 4
19.     "0010010", -- 5
20.     "0000010", -- 6
21.     "1111000", -- 7
22.     "0000000", -- 8
23.     "0010000", -- 9
24.     "0001000", -- A
25.     "0000011", -- b
26.     "1000110", -- C
27.     "0100001", -- d
28.     "0000110", -- E
29.     "0001110"  -- F
30. );
31.
32.
33.  begin
34.
35. data <= sevenSegment_ROM(to_integer(unsigned(address)));
36.  end Behavioral;
37.
```

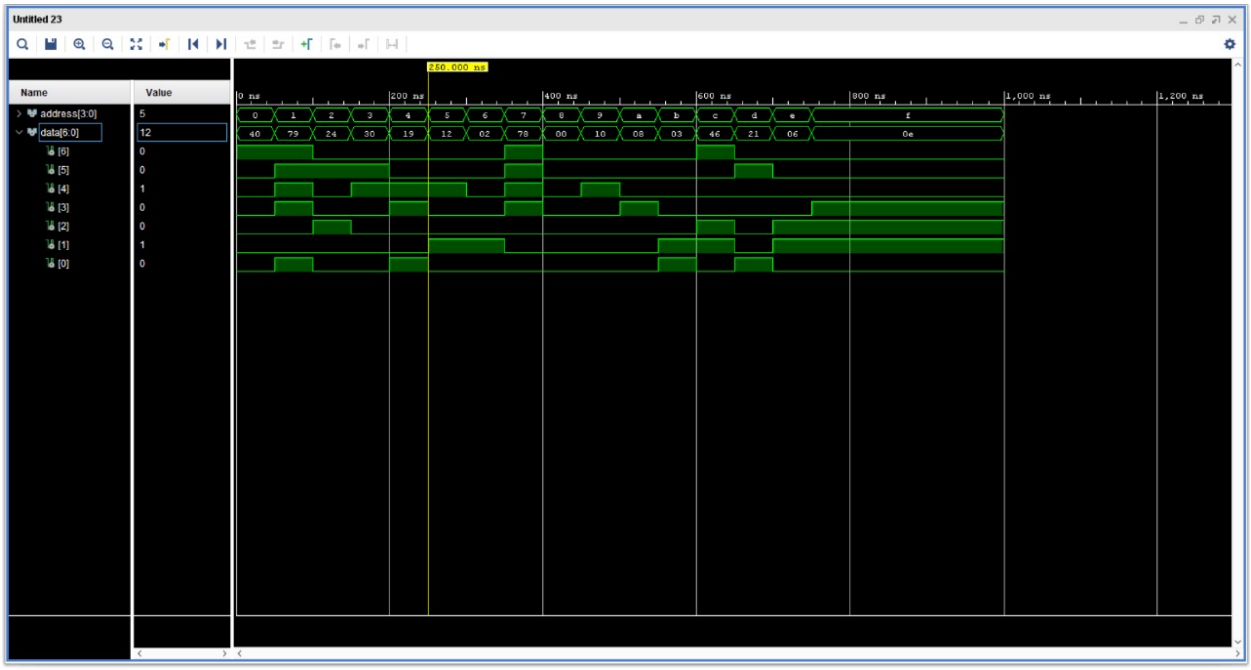
1.16.2 Elaborated Design Schematic



1.16.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use IEEE.NUMERIC_STD.ALL;
 4.
 5. entity TB_LUT_16_7 is
 6. end TB_LUT_16_7;
 7.
 8. architecture Behavioral of TB_LUT_16_7 is
 9.
10.     -- Component Declaration
11.     component LUT_16_7
12.         Port (
13.             address : in STD_LOGIC_VECTOR (3 downto 0);
14.             data : out STD_LOGIC_VECTOR (6 downto 0)
15.         );
16.     end component;
17.
18.     -- Test Signals
19.     signal address : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
20.     signal data : STD_LOGIC_VECTOR (6 downto 0);
21.
22. begin
23.
24.     -- Instantiate the Unit Under Test (UUT)
25.     uut: LUT_16_7 Port Map (
26.         address => address,
27.         data => data
28.     );
29.
30.     -- Stimulus Process
31.     stim_proc: process
32.     begin
33.         -- Test each address value from 0 to 15
34.         for i in 0 to 15 loop
35.             address <= std_logic_vector(to_unsigned(i, 4));
36.             wait for 50 ns; -- wait for signal update
37.         end loop;
38.
39.         wait;
40.     end process;
41.
42. end Behavioral;
43.
```

1.16.4 Timing Diagram



1.17 Basic Constraint File

```
 1. ## Clock (clk_in)
 2. set_property PACKAGE_PIN W5            [get_ports clk_in]
 3.     set_property IOSTANDARD LVCMOS33   [get_ports clk_in]
 4. create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
clk_in]
 5.
 6. ## Reset Button (reset)
 7. set_property PACKAGE_PIN U18           [get_ports reset]
 8.     set_property IOSTANDARD LVCMOS33   [get_ports reset]
 9.
10. ## reg7_LED (4 bits from R7 register)
11. set_property PACKAGE_PIN U16           [get_ports {reg7_LED[0]}]
12.     set_property IOSTANDARD LVCMOS33   [get_ports {reg7_LED[0]}]
13. set_property PACKAGE_PIN E19           [get_ports {reg7_LED[1]}]
14.     set_property IOSTANDARD LVCMOS33   [get_ports {reg7_LED[1]}]
15. set_property PACKAGE_PIN U19           [get_ports {reg7_LED[2]}]
16.     set_property IOSTANDARD LVCMOS33   [get_ports {reg7_LED[2]}]
17. set_property PACKAGE_PIN V19           [get_ports {reg7_LED[3]}]
18.     set_property IOSTANDARD LVCMOS33   [get_ports {reg7_LED[3]}]
19.
20. ## Status Flags
21. set_property PACKAGE_PIN L1            [get_ports overflow_flag]
```

```
22.     set_property IOSTANDARD LVCMOS33   [get_ports overflow_flag]
23. set_property PACKAGE_PIN P1            [get_ports zero_flag]
24.     set_property IOSTANDARD LVCMOS33   [get_ports zero_flag]
25.
26. ## Seven Segment Display Segments (seven_segment[6:0])
27. set_property PACKAGE_PIN W7            [get_ports {seven_segment[0]}]
28.     set_property IOSTANDARD LVCMOS33   [get_ports {seven_segment[0]}]
29. set_property PACKAGE_PIN W6            [get_ports {seven_segment[1]}]
30.     set_property IOSTANDARD LVCMOS33   [get_ports {seven_segment[1]}]
31. set_property PACKAGE_PIN U8            [get_ports {seven_segment[2]}]
32.     set_property IOSTANDARD LVCMOS33   [get_ports {seven_segment[2]}]
33. set_property PACKAGE_PIN V8            [get_ports {seven_segment[3]}]
34.     set_property IOSTANDARD LVCMOS33   [get_ports {seven_segment[3]}]
35. set_property PACKAGE_PIN U5            [get_ports {seven_segment[4]}]
36.     set_property IOSTANDARD LVCMOS33   [get_ports {seven_segment[4]}]
37. set_property PACKAGE_PIN V5            [get_ports {seven_segment[5]}]
38.     set_property IOSTANDARD LVCMOS33   [get_ports {seven_segment[5]}]
39. set_property PACKAGE_PIN U7            [get_ports {seven_segment[6]}]
40.     set_property IOSTANDARD LVCMOS33   [get_ports {seven_segment[6]}]
41.
42. set_property PACKAGE_PIN U2 [get_ports {Anode[0]}]
43.     set_property IOSTANDARD LVCMOS33 [get_ports {Anode[0]}]
44. set_property PACKAGE_PIN U4 [get_ports {Anode[1]}]
45.     set_property IOSTANDARD LVCMOS33 [get_ports {Anode[1]}]
46. set_property PACKAGE_PIN V4 [get_ports {Anode[2]}]
47.     set_property IOSTANDARD LVCMOS33 [get_ports {Anode[2]}]
48. set_property PACKAGE_PIN W4 [get_ports {Anode[3]}]
49.     set_property IOSTANDARD LVCMOS33 [get_ports {Anode[3]}]
50.
```

2. Nano-processor (Reduced LUTs)

2.1 Introduction

The enhanced version of the nano processor introduces significant improvements in both timing performance and resource optimization. In significance, the system clock has been optimized from 11 clock cycles to just 2, increasing the processor's execution speed and efficiency. In addition, the lookup table (LUT) usage for the program counter has been reduced from 25 to 21, reflecting a more efficient hardware design with minimized logic resource consumption. Overall, the These enhancements contribute to a more compact and faster processor architecture, making it better suited for resource-constrained FPGA implementations and embedded system applications.

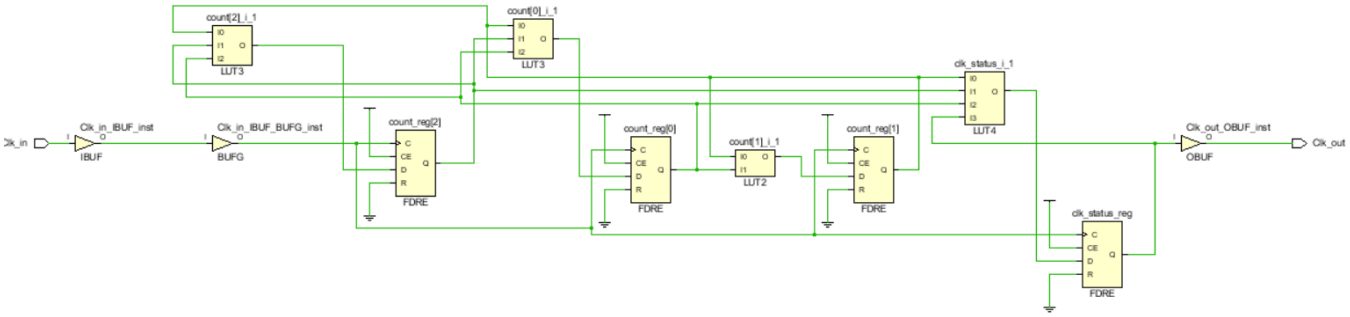
2.2 Slow Clock

2.2.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use IEEE.NUMERIC_STD.ALL;
 4.
 5. entity Clock is
 6.     Port (
 7.         Clk_in  : in  STD_LOGIC;
 8.         Clk_out : out STD_LOGIC
 9.     );
10. end Clock;
11.
12. architecture Behavioral of Clock is
13.     constant N         : unsigned(2 downto 0) := to_unsigned(5, 3);  -- Can be
changed
14.     signal count       : unsigned(2 downto 0) := (others => '0');
15.     signal clk_status  : std_logic := '0';
16.     signal next_count  : unsigned(2 downto 0);
17.     signal toggle      : std_logic;
18. begin
19.
20.
21.     toggle <= '1' when count = N - 1 else '0';
22.
```

```
23.     next_count <= (others => '0') when toggle = '1' else count + 1;
24.
25.     process (Clk_in)
26.     begin
27.         if rising_edge(Clk_in) then
28.             count      <= next_count;
29.             clk_status <= clk_status xor toggle;
30.         end if;
31.     end process;
32.
33.     Clk_out <= clk_status;
34.
35. end Behavioral;
36.
```

2.2.2 Elaborated Design Schematics



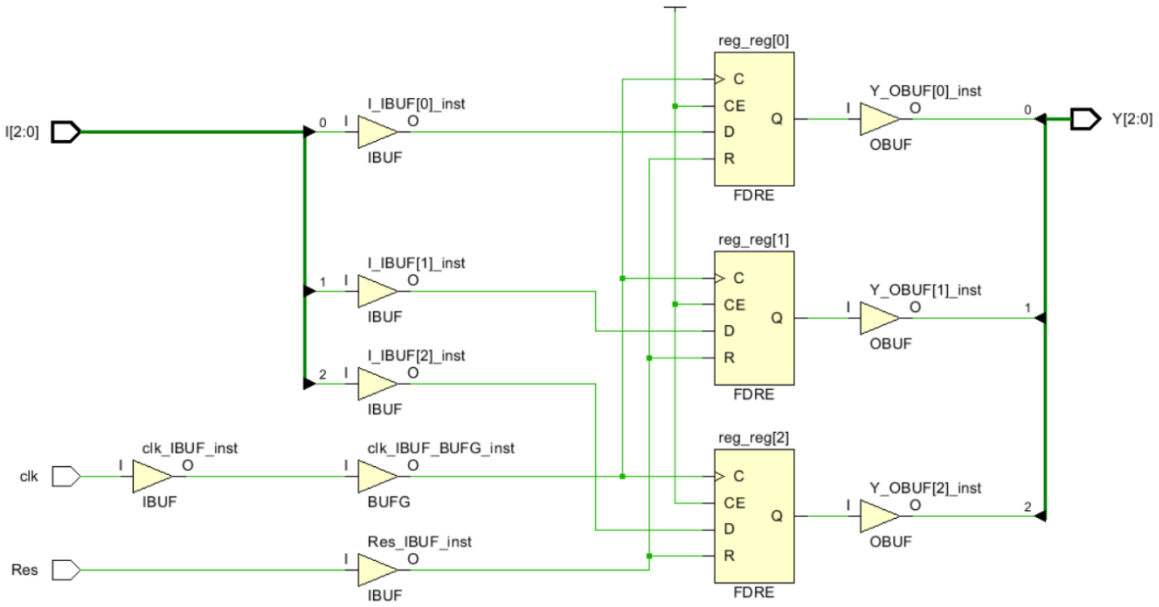
2.3 Program Counter

2.3.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Program_Counter is
 5.     Port (
 6.         clk : in  STD_LOGIC;
 7.         Res : in  STD_LOGIC;
 8.         I   : in  STD_LOGIC_VECTOR (2 downto 0);
 9.         Y   : out STD_LOGIC_VECTOR (2 downto 0)
10.     );
11. end Program_Counter;
12.
13. architecture Behavioral of Program_Counter is
```

```
    signal reg : STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
begin

    process(clk)
    begin
        if rising_edge(clk) then
            if Res = '1' then
                reg <= (others => '0');
            else
                reg <= I;
            end if;
        end if;
    end process;

    Y <= reg;

end Behavioral;

```

2.3.2 Elaborated Design Schematics



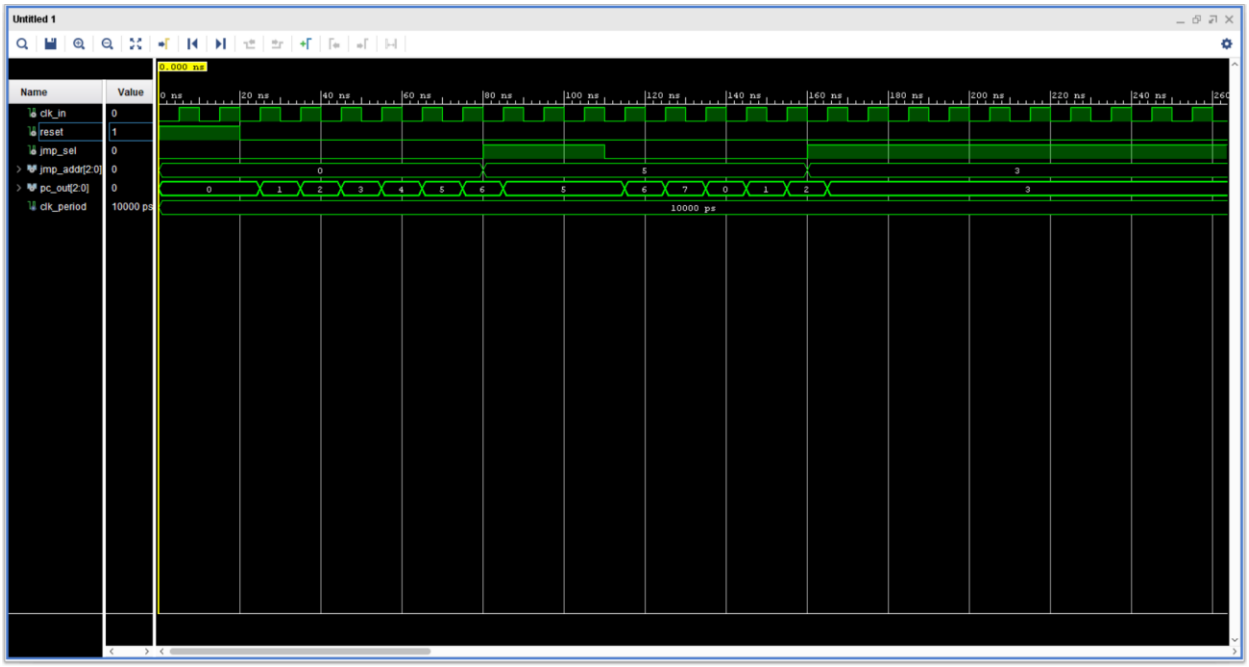
2.3.3 Simulation Source Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Test_Program_Counter_TB is
-- Testbench has no ports
end Test_Program_Counter_TB;
```

```
 7.
 8. architecture Behavioral of Test_Program_Counter_TB is
 9.
10.     -- Component under test
11.     component Test_Program_Counter
12.         Port (
13.             clk_in  : in  STD_LOGIC;
14.             reset   : in  STD_LOGIC;
15.             jmp_sel : in  STD_LOGIC;
16.             jmp_addr: in  STD_LOGIC_VECTOR(2 downto 0);
17.             pc_out  : out STD_LOGIC_VECTOR(2 downto 0)
18.         );
19.     end component;
20.
21.     -- Signals to connect to the DUT
22.     signal clk_in    : STD_LOGIC := '0';
23.     signal reset     : STD_LOGIC := '1';
24.     signal jmp_sel   : STD_LOGIC := '0';
25.     signal jmp_addr  : STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
26.     signal pc_out    : STD_LOGIC_VECTOR(2 downto 0);
27.
28.     constant clk_period : time := 10 ns;
29.
30. begin
31.
32.     -- Instantiate the Unit Under Test (UUT)
33.     uut: Test_Program_Counter
34.         Port map (
35.             clk_in  => clk_in,
36.             reset   => reset,
37.             jmp_sel => jmp_sel,
38.             jmp_addr=> jmp_addr,
39.             pc_out  => pc_out
40.         );
41.
42.     -- Clock generation process
43.     clk_process: process
44.     begin
45.         while True loop
46.             clk_in <= '0';
47.             wait for clk_period/2;
48.             clk_in <= '1';
49.             wait for clk_period/2;
50.         end loop;
51.     end process;
52.
53.     -- Stimulus process
54.     stim_proc: process
55.     begin
56.         -- Initial reset active for 2 clock cycles
57.         reset <= '1';
58.         wait for clk_period * 2;
59.         reset <= '0';
60.         wait for clk_period;
```

```
61.
62.         -- Let PC increment normally (jmp_sel = 0)
63.         jmp_sel <= '0';
64.         wait for clk_period * 5;
65.
66.         -- Apply jump address 5, enable jump (jmp_sel = 1)
67.         jmp_addr <= "101";  -- decimal 5
68.         jmp_sel <= '1';
69.         wait for clk_period * 3;
70.
71.         -- Disable jump, continue normal increment
72.         jmp_sel <= '0';
73.         wait for clk_period * 5;
74.
75.         -- Apply jump address 3, enable jump
76.         jmp_addr <= "011";  -- decimal 3
77.         jmp_sel <= '1';
78.         wait for clk_period * 2;
79.
80.         -- Finish simulation
81.         wait;
82.     end process;
83.
84. end Behavioral;
85.
```

2.3.4 Timing Diagram



3.Improved Nano Processor

3.1 Introduction

This project is about an improved version of our earlier nano processor design. The original processor could only do basic addition and subtraction using a ripple carry adder. Now, we've made several upgrades to make the processor more powerful and flexible. One of the main improvements is that we changed the instruction format and redesigned the instruction decoder to better handle different types of operations.

The biggest change is the addition of a full Arithmetic and Logic Unit (ALU). Instead of just an adder and subtractor, the new ALU includes a ripple carry adder/subtractor, a multiplier, a comparator, and bitwise logic operations like AND, OR, and XOR.

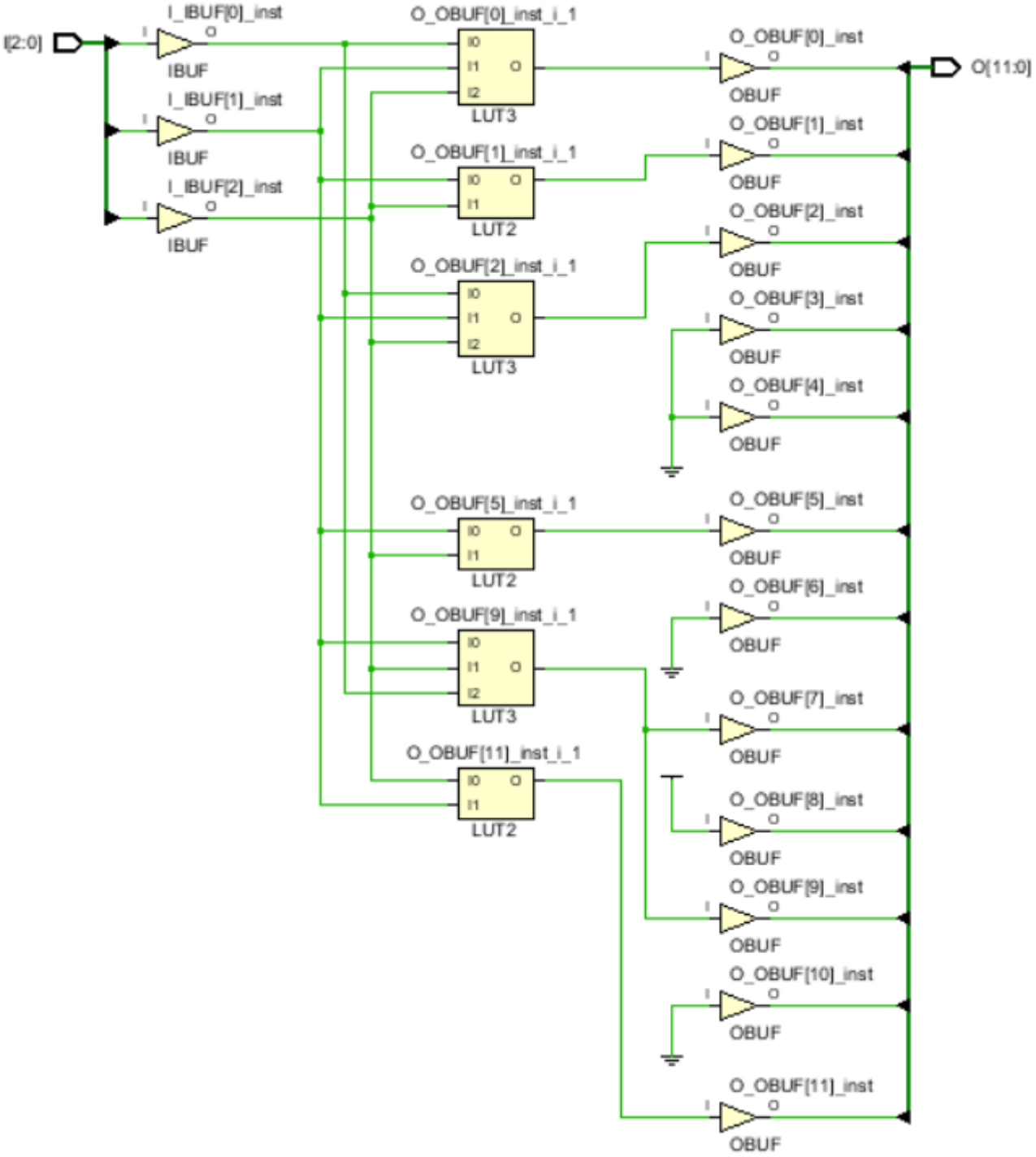
With these improvements, the processor can now perform more complex operations, making it more efficient and useful for a wider range of tasks.

3.1 Program Rom

3.1.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use ieee.numeric_std.all;
 4.
 5. entity Program_Rom is
 6.     Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
 7.            O : out STD_LOGIC_VECTOR (11 downto 0));
 8. end Program_Rom;
 9.
10. architecture Behavioral of Program_Rom is
11.     type rom_type is array (0 to 9) of std_logic_vector (11 downto 0);
12.     signal program_ROM : rom_type := (
13.            0  => "101110000011",  -- MOVI R7, 3
14.            1  => "100100000010",  -- MOVI R2, 2
15.            2  => "001110100101",  -- MUL R7, R2
16.            3  => "001110100000",  -- ADD R7, R2
17.            4  => "001110100001",  -- SUB R7, R2
18.            5  => "001110100000",  -- ADD R7, R2
19.            6  => "001110100010",  -- AND R7, R2
20.            7  => "001110100011",  -- OR  R7, R2
21.            8  => "001110100100",  -- XOR  R7, R2
22.            9  => "001110100111"   -- CMP R7, R2
23.     );
24.
25. begin
26.     O <= program_ROM(to_integer(unsigned(I)));
27. end Behavioral;
```

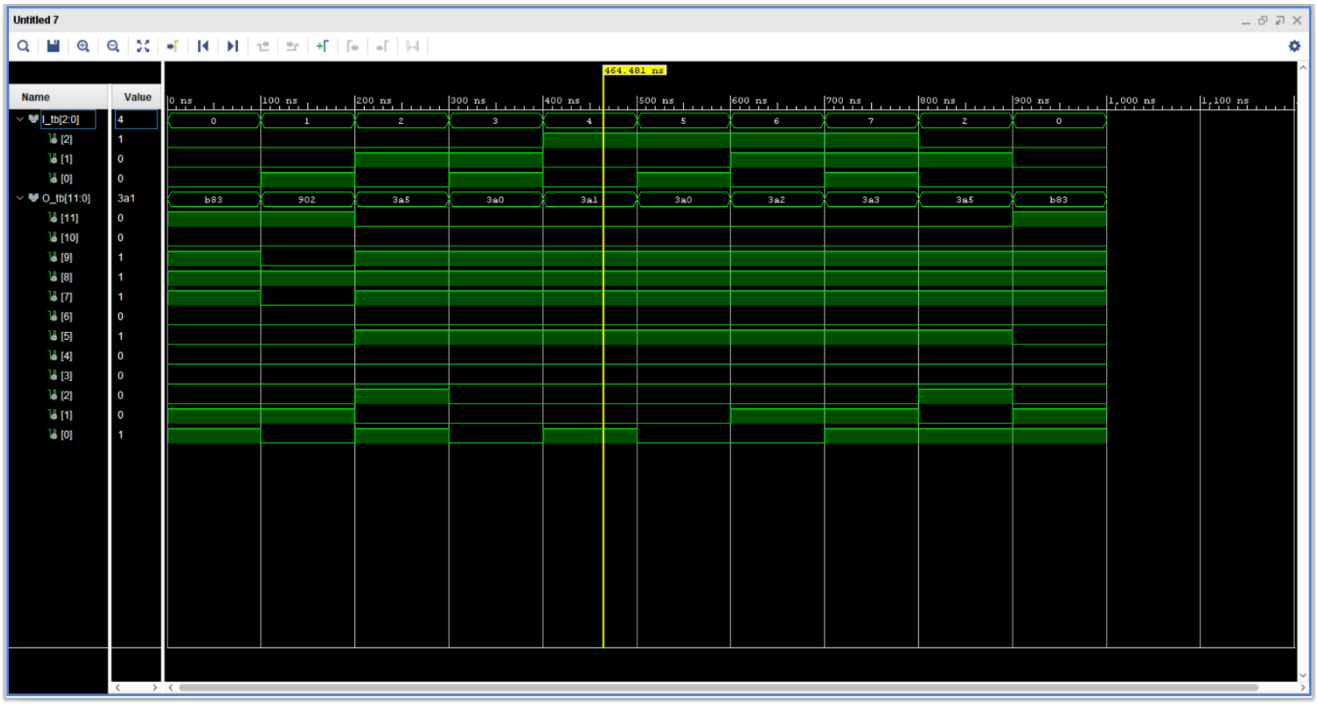
3.1.2 Elaborated Design Schematics



3.1.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use IEEE.NUMERIC_STD.ALL;
 4.
 5. entity TB_Program_Rom is
 6. end TB_Program_Rom;
 7.
 8. architecture tb of TB_Program_Rom is
 9.
10.     -- Declare the component under test
11.     component Program_Rom
12.         Port (
13.             I : in  STD_LOGIC_VECTOR (2 downto 0);
14.             O : out STD_LOGIC_VECTOR (11 downto 0)
15.         );
16.     end component;
17.
18.     -- Signals to connect to UUT
19.     signal I_tb : STD_LOGIC_VECTOR (2 downto 0) := (others => '0');
20.     signal O_tb : STD_LOGIC_VECTOR (11 downto 0);
21.
22. begin
23.
24.     -- Instantiate the UUT
25.     uut: Program_Rom
26.         port map (
27.             I => I_tb,
28.             O => O_tb
29.         );
30.
31.     -- Stimulus process
32.     stim_proc: process
33.     begin
34.         -- Test 10 different inputs with 100 ns delay
35.         I_tb <= "000"; wait for 100 ns;  -- Expect "100010000010"
36.         I_tb <= "001"; wait for 100 ns;  -- Expect "101110000011"
37.         I_tb <= "010"; wait for 100 ns;  -- Expect "100100000001"
38.         I_tb <= "011"; wait for 100 ns;  -- Expect "010100000000"
39.         I_tb <= "100"; wait for 100 ns;  -- Expect "001110010000"
40.         I_tb <= "101"; wait for 100 ns;  -- Expect "000010100000"
41.         I_tb <= "110"; wait for 100 ns;  -- Expect "110010000110"
42.         I_tb <= "111"; wait for 100 ns;  -- Expect "110000000100"
43.         I_tb <= "010"; wait for 100 ns;  -- Re-check
44.         I_tb <= "000"; wait for 100 ns;  -- Re-check
45.
46.         wait;  -- End simulation
47.     end process;
48.
49. end tb;
```

3.1.4 Timing Diagram



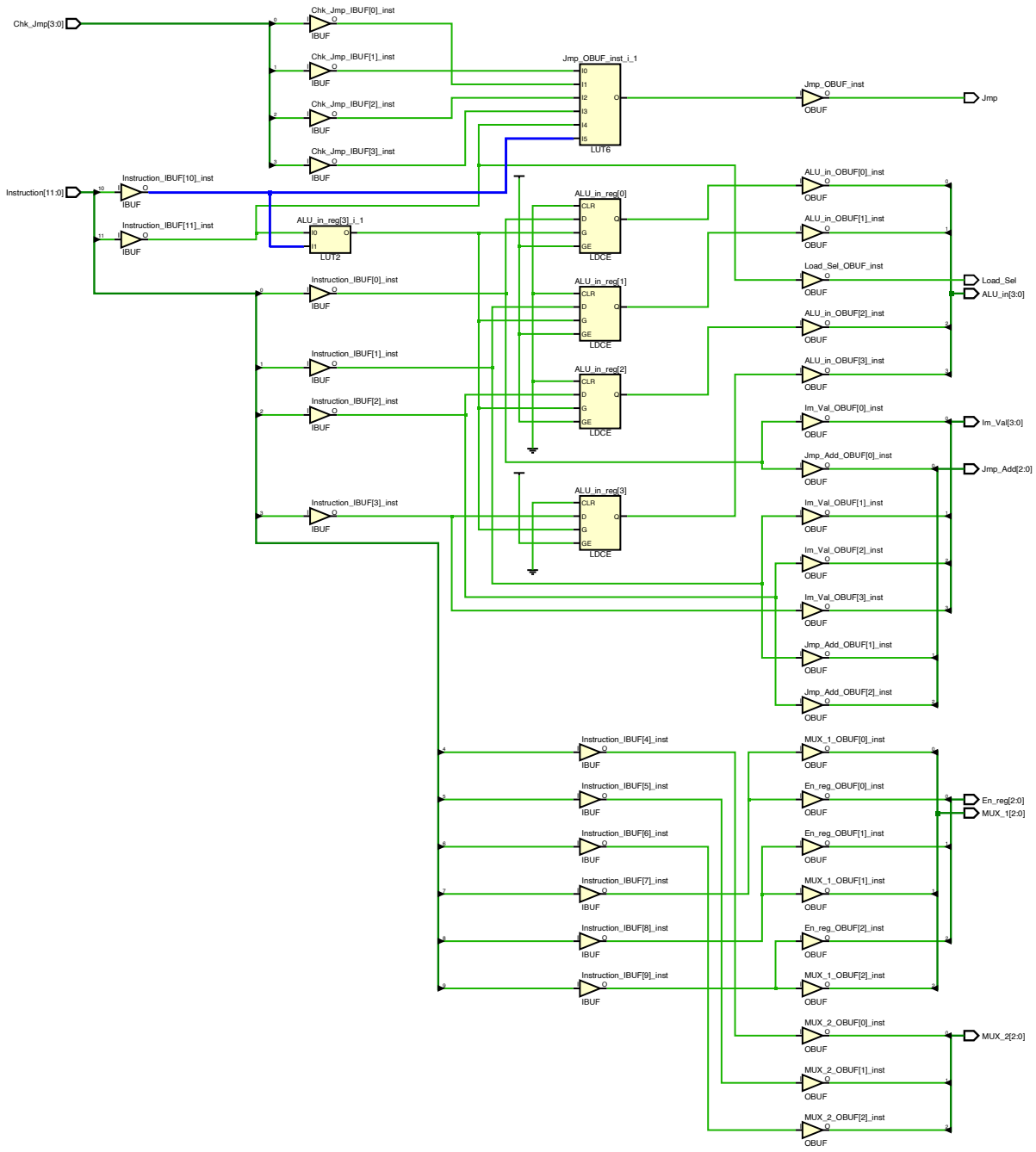
3.2 Instruction Decoder

3.2.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4.
 5. entity Instruction_Decoder is
 6.     Port (Instruction : in STD_LOGIC_VECTOR (11 downto 0);
 7.           Chk_Jmp : in STD_LOGIC_VECTOR (3 downto 0);
 8.           En_reg : out STD_LOGIC_VECTOR (2 downto 0);
 9.           MUX_1 : out STD_LOGIC_VECTOR (2 downto 0);
10.           MUX_2 : out STD_LOGIC_VECTOR (2 downto 0);
11.           Jmp : out STD_LOGIC;
12.           ALU_in : out STD_LOGIC_vector(3 downto 0);
13.           Im_Val : out STD_LOGIC_VECTOR (3 downto 0);
14.           Load_Sel : out STD_LOGIC;
15.           Jmp_Add :  out STD_LOGIC_VECTOR (2 downto 0)
16.           );
17.
18. end Instruction_Decoder;
```

```
19.
20. architecture Behavioral of Instruction_decoder is
21.
22.
23. begin
24.
25. MUX_1 <= Instruction (9 downto 7);
26. MUX_2 <= Instruction (6 downto 4);
27. ALU_in <= Instruction (3 downto 0) when Instruction(11 downto 10) = "00";
28. Load_Sel <= Instruction(11);
29. Jmp <= '1' when Chk_Jmp = "0000" and Instruction(11 downto 10) = "11" else '0';
30. En_Reg <= Instruction (9 downto 7);
31. Jmp_add <= Instruction (2 downto 0);
32. Im_Val <= Instruction(3 downto 0);
33.
34. end Behavioral;
35.
```

3.2.2 Elaborated Design Schematics



For the improved nano-processor, for increased number of operations, we have slightly changed the instruction bit format.

OP code – First 2 bits of instruction.

Sub OP code – Last 4 bits of the instruction, only if OP code = “00”.

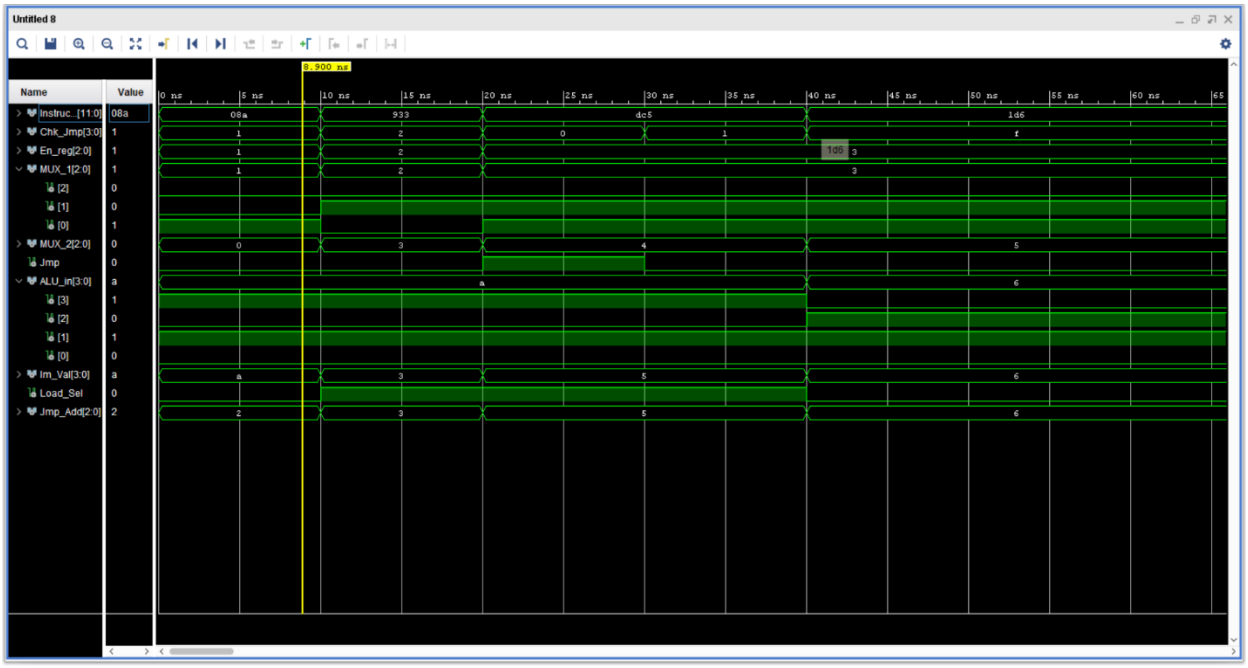
OP code	Sub OP code	Instruction	Description
10	-	10 RRR 000 dddd	Move the value dddd to register RRR
00	0000 - add	00 RaRaRa RbRbRb 0000	Add values in Ra, Rb and store in Ra
00	0001 - subtract	00 RaRaRa RbRbRb 0001	Subtract values in Ra, Rb and store in Ra
00	1000 - negate	00 RaRaRa 000 1000	Negate the value in Ra
00	0010 - AND	00 RaRaRa RbRbRb 0010	Bit operation AND to the values in Ra, Rb
00	0011 - OR	00 RaRaRa RbRbRb 0011	Bit operation OR to the values in Ra, Rb
00	0100 - XOR	00 RaRaRa RbRbRb 0100	Bit operation XOR to the values in Ra, Rb
00	0101 - multiply	00 RaRaRa RbRbRb 0101	Multiply Values in Ra, Rb and store in Ra
00	0111 - compare	00 RaRaRa RbRbRb 0111	Compare the value in Ra to Rb and provide an out signal saying whether it is less than, greater than or equal.

3.2.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_Instruction_Decoder is
 5. end TB_Instruction_Decoder;
 6.
 7. architecture Behavioral of TB_Instruction_Decoder is
 8.
 9.     -- Component Declaration
10.     component Instruction_Decoder
11.         Port (
12.             Instruction : in STD_LOGIC_VECTOR (11 downto 0);
13.             Chk_Jmp : in STD_LOGIC_VECTOR (3 downto 0);
14.             En_reg : out STD_LOGIC_VECTOR (2 downto 0);
15.             MUX_1 : out STD_LOGIC_VECTOR (2 downto 0);
16.             MUX_2 : out STD_LOGIC_VECTOR (2 downto 0);
17.             Jmp : out STD_LOGIC;
18.             ALU_in : out STD_LOGIC_vector(3 downto 0);
19.             Im_Val : out STD_LOGIC_VECTOR (3 downto 0);
20.             Load_Sel : out STD_LOGIC;
21.             Jmp_Add : out STD_LOGIC_VECTOR (2 downto 0)
22.         );
23.     end component;
24.
25.     -- Signals for inputs
26.     signal Instruction : STD_LOGIC_VECTOR (11 downto 0);
27.     signal Chk_Jmp : STD_LOGIC_VECTOR (3 downto 0);
28.
29.     -- Signals for outputs
30.     signal En_reg : STD_LOGIC_VECTOR (2 downto 0);
31.     signal MUX_1 : STD_LOGIC_VECTOR (2 downto 0);
32.     signal MUX_2 : STD_LOGIC_VECTOR (2 downto 0);
33.     signal Jmp : STD_LOGIC;
34.     signal ALU_in : STD_LOGIC_VECTOR (3 downto 0);
35.     signal Im_Val : STD_LOGIC_VECTOR (3 downto 0);
36.     signal Load_Sel : STD_LOGIC;
37.     signal Jmp_Add : STD_LOGIC_VECTOR (2 downto 0);
38.
39. begin
40.
41.     -- Instantiate the Unit Under Test
42.     uut: Instruction_Decoder
43.         port map (
44.             Instruction => Instruction,
45.             Chk_Jmp => Chk_Jmp,
46.             En_reg => En_reg,
47.             MUX_1 => MUX_1,
48.             MUX_2 => MUX_2,
49.             Jmp => Jmp,
```

```
50.             ALU_in => ALU_in,
51.             Im_Val => Im_Val,
52.             Load_Sel => Load_Sel,
53.             Jmp_Add => Jmp_Add
54.         );
55.
56.     -- Stimulus process
57.     stim_proc: process
58.     begin
59.         -- Test 1: ALU operation (Instruction bits 11-10 = "00")
60.         Instruction <= "000010001010"; -- example: ALU operation
61.         Chk_Jmp <= "0001";
62.         wait for 10 ns;
63.
64.         -- Test 2: Load operation (Instruction(11) = '1')
65.         Instruction <= "100100110011"; -- example: load
66.         Chk_Jmp <= "0010";
67.         wait for 10 ns;
68.
69.         -- Test 3: Jump operation with Chk_Jmp = "0000"
70.         Instruction <= "110111000101"; -- Instruction(11-10)="11"
71.         Chk_Jmp <= "0000";
72.         wait for 10 ns;
73.
74.         -- Test 4: Jump instruction, but Chk_Jmp not "0000" (no jump expected)
75.         Instruction <= "110111000101";
76.         Chk_Jmp <= "0001";
77.         wait for 10 ns;
78.
79.         -- Test 5: Another ALU instruction
80.         Instruction <= "000111010110";
81.         Chk_Jmp <= "1111";
82.         wait for 10 ns;
83.
84.         wait;
85.     end process;
86.
87. end Behavioral;
88.
```

3.2.4 Timing Diagram



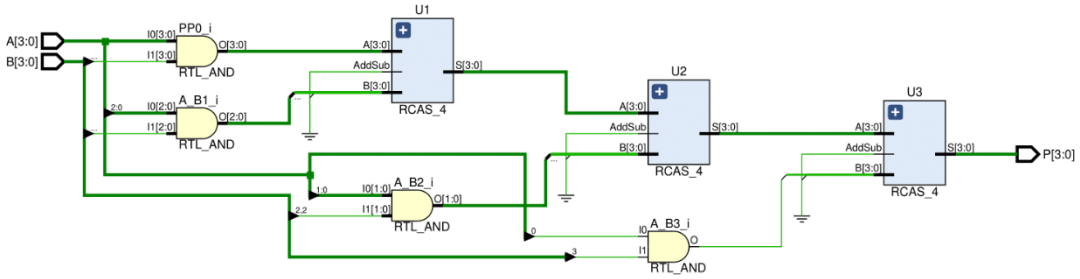
3.3 Multiplier

3.3.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Multiplier is
 5.     Port (
 6.         A : in  STD_LOGIC_VECTOR(3 downto 0);
 7.         B : in  STD_LOGIC_VECTOR(3 downto 0);
 8.         P : out STD_LOGIC_VECTOR(3 downto 0)  -- changed from 8 to 4 bits
 9.     );
10. end Multiplier;
11.
12. architecture Behavioral of Multiplier is
13.
14.     signal PP0, PP1, PP2, PP3 : STD_LOGIC_VECTOR(3 downto 0);
15.     signal T1, T2 : STD_LOGIC_VECTOR(3 downto 0);
16.
17.     component RCAS_4
18.         Port (
19.             A      : in  STD_LOGIC_VECTOR(3 downto 0);
20.             B      : in  STD_LOGIC_VECTOR(3 downto 0);
```

```
21.             AddSub : in  STD_LOGIC;
22.             S      : out STD_LOGIC_VECTOR(3 downto 0);
23.             C_out  : out STD_LOGIC
24.         );
25.     end component;
26.
27.     -- helper signals for shifting
28.     signal A_B1 : STD_LOGIC_VECTOR(3 downto 0);
29.     signal A_B2 : STD_LOGIC_VECTOR(3 downto 0);
30.     signal A_B3 : STD_LOGIC_VECTOR(3 downto 0);
31.
32. begin
33.
34.     -- Base partial products
35.     PP0 <= A and (B(0) & B(0) & B(0) & B(0));  -- No shift
36.
37.     A_B1 <= A and (B(1) & B(1) & B(1) & B(1));
38.     PP1  <= A_B1(2 downto 0) & '0';  -- Shifted left by 1
39.
40.     A_B2 <= A and (B(2) & B(2) & B(2) & B(2));
41.     PP2  <= A_B2(1 downto 0) & "00"; -- Shifted left by 2
42.
43.     A_B3 <= A and (B(3) & B(3) & B(3) & B(3));
44.     PP3  <= A_B3(0) & "000";         -- Shifted left by 3
45.
46.     -- Ripple carry addition (3 additions)
47.     U1: RCAS_4 port map (A => PP0, B => PP1, AddSub => '0', S => T1, C_out =>
open);
48.     U2: RCAS_4 port map (A => T1,  B => PP2, AddSub => '0', S => T2, C_out =>
open);
49.     U3: RCAS_4 port map (A => T2,  B => PP3, AddSub => '0', S => P,  C_out =>
open);
50.
51. end Behavioral;
```

3.3.2 Elaborated Design Schematics

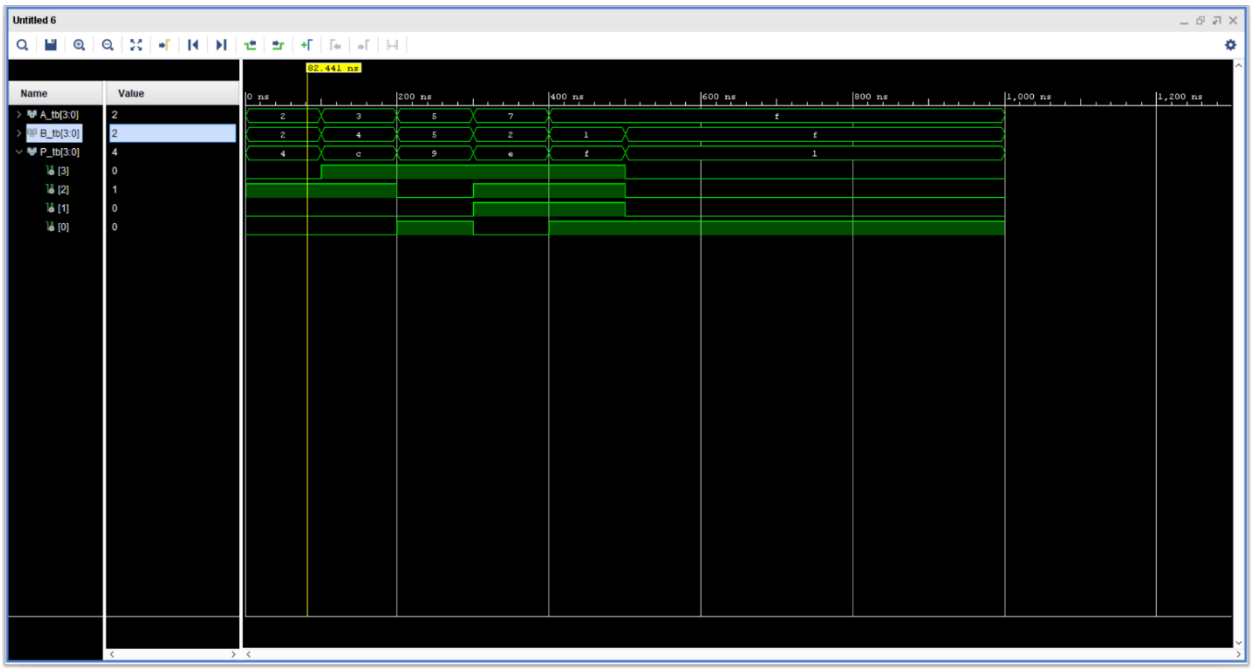


3.3.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity TB_Multiplier is
 5. end TB_Multiplier;
 6.
 7. architecture behavior of TB_Multiplier is
 8.
 9.     -- Component Declaration for the Unit Under Test (UUT)
10.     component Multiplier
11.         Port (
12.             A : in  STD_LOGIC_VECTOR(3 downto 0);
13.             B : in  STD_LOGIC_VECTOR(3 downto 0);
14.             P : out STD_LOGIC_VECTOR(3 downto 0)
15.         );
16.     end component;
17.
18.     -- Testbench signals
19.     signal A_tb : STD_LOGIC_VECTOR(3 downto 0);
20.     signal B_tb : STD_LOGIC_VECTOR(3 downto 0);
21.     signal P_tb : STD_LOGIC_VECTOR(3 downto 0);
22.
23. begin
24.
25.     -- Instantiate the Unit Under Test (UUT)
26.     uut: Multiplier
27.         port map (
28.             A => A_tb,
29.             B => B_tb,
30.             P => P_tb
31.         );
32.
33.     -- Stimulus process
34.     stim_proc: process
35.     begin
36.         -- Test 1: 2 x 2 = 4
37.         A_tb <= "0010";
38.         B_tb <= "0010";
39.         wait for 100 ns;
40.
41.         -- Test 2: 3 x 4 = 12
42.         A_tb <= "0011";
43.         B_tb <= "0100";
44.         wait for 100 ns;
45.
46.         -- Test 3: 5 x 5 = 25 -> result is 1001 (only 4 LSB bits of 11001)
47.         A_tb <= "0101";
48.         B_tb <= "0101";
49.         wait for 100 ns;
```

```
50.
51.         -- Test 4: 7 x 2 = 14
52.         A_tb <= "0111";
53.         B_tb <= "0010";
54.         wait for 100 ns;
55.
56.         -- Test 5: 15 x 1 = 15
57.         A_tb <= "1111";
58.         B_tb <= "0001";
59.         wait for 100 ns;
60.
61.         -- Test 6: 15 x 15 = 225 → lower 4 bits of 0xE1 = 0001
62.         A_tb <= "1111";
63.         B_tb <= "1111";
64.         wait for 100 ns;
65.
66.         wait;
67.     end process;
68.
69. end behavior;
70.
```

3.3.4 Timing Diagram

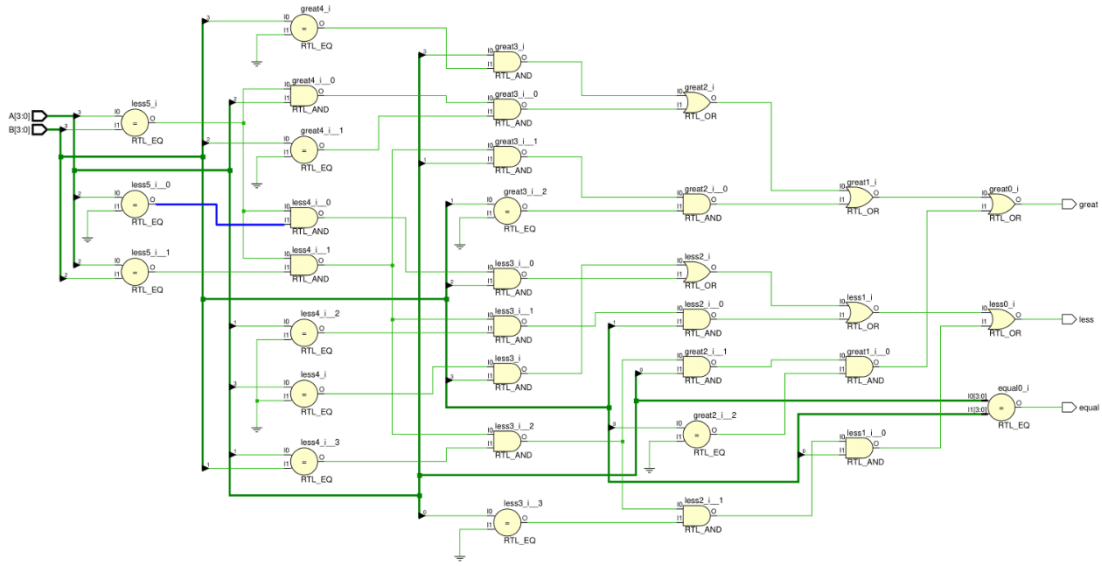


3.4 Comparator

3.4.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3.
 4. entity Comparator is
 5.     Port (
 6.         A      : in  STD_LOGIC_VECTOR(3 downto 0);
 7.         B      : in  STD_LOGIC_VECTOR(3 downto 0);
 8.         equal  : out STD_LOGIC;
 9.         less   : out STD_LOGIC;
10.         great  : out STD_LOGIC
11.     );
12. end Comparator;
13.
14. architecture Behavioral of Comparator is
15.
16. signal vA_less  : STD_LOGIC;
17. signal vA_equal : STD_LOGIC;
18. signal vA_grt   : STD_LOGIC;
19.
20. begin
21.
22.     -- Equal condition (bitwise equality)
23.     vA_equal <= '1' when (A = B) else '0';
24.
25.     -- Less than condition (unsigned magnitude comparison, MSB to LSB)
26.     vA_less <= '1' when
27.                 (A(3) = '0' and B(3) = '1') or
28.                 (A(3) = B(3) and A(2) = '0' and B(2) = '1') or
29.                 (A(3) = B(3) and A(2) = B(2) and A(1) = '0' and B(1) = '1') or
30.                 (A(3) = B(3) and A(2) = B(2) and A(1) = B(1) and A(0) = '0' and
B(0) = '1')
31.             else '0';
32.
33.     -- Greater than condition (unsigned magnitude comparison, MSB to LSB)
34.     vA_grt <= '1' when
35.                 (A(3) = '1' and B(3) = '0') or
36.                 (A(3) = B(3) and A(2) = '1' and B(2) = '0') or
37.                 (A(3) = B(3) and A(2) = B(2) and A(1) = '1' and B(1) = '0') or
38.                 (A(3) = B(3) and A(2) = B(2) and A(1) = B(1) and A(0) = '1' and
B(0) = '0')
39.             else '0';
40.
41.     -- Output results
42.     great <= vA_grt;
43.     less  <= vA_less;
44.     equal <= vA_equal;
45.
46. end Behavioral;
```

3.4.2 Elaborated Design Schematics

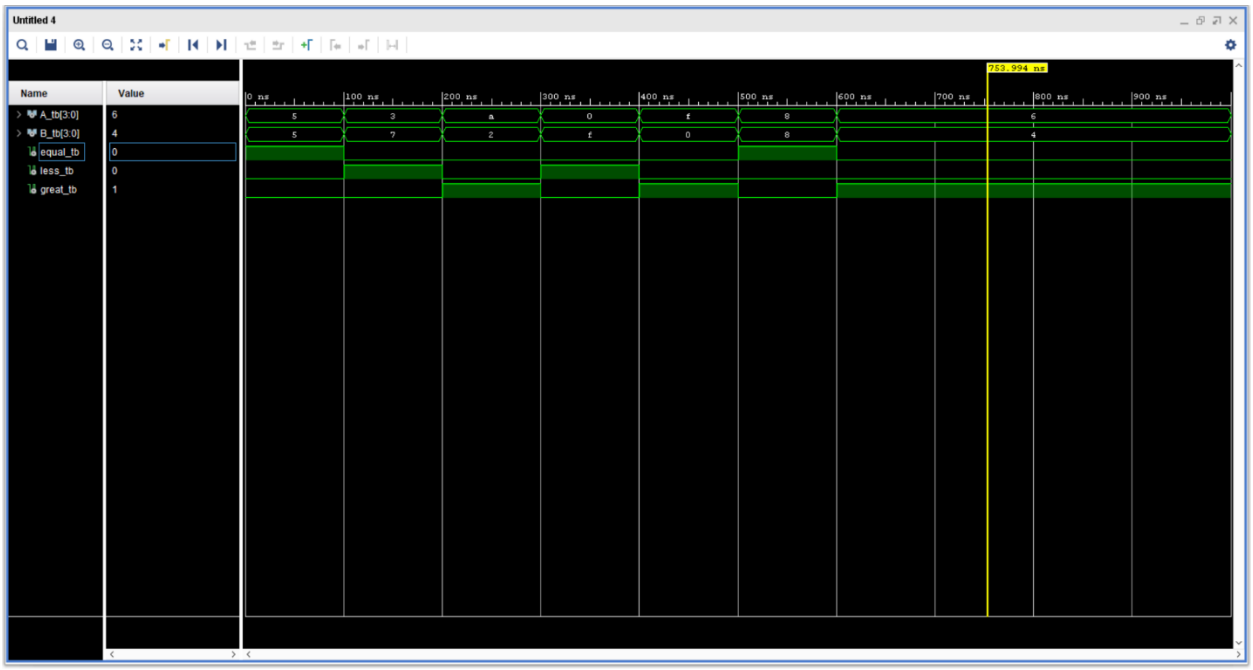


3.4.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
 4.
 5. entity TB_Comparator is
 6. end TB_Comparator;
 7.
 8. architecture behavior of TB_Comparator is
 9.
10.     -- Component Declaration for the Unit Under Test (UUT)
11.     component Comparator
12.         Port (
13.             A      : in  STD_LOGIC_VECTOR(3 downto 0);
14.             B      : in  STD_LOGIC_VECTOR(3 downto 0);
15.             equal  : out STD_LOGIC;
16.             less   : out STD_LOGIC;
17.             great  : out STD_LOGIC
18.         );
19.     end component;
20.
21.     -- Testbench signals
```

```
22.     signal A_tb     : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');
23.     signal B_tb     : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');
24.     signal equal_tb : STD_LOGIC;
25.     signal less_tb  : STD_LOGIC;
26.     signal great_tb : STD_LOGIC;
27.
28. begin
29.
30.     -- Instantiate the Unit Under Test (UUT)
31.     uut: Comparator Port Map (
32.         A      => A_tb,
33.         B      => B_tb,
34.         equal  => equal_tb,
35.         less   => less_tb,
36.         great  => great_tb
37.     );
38.
39.     -- Stimulus process
40.     stim_proc: process
41.     begin
42.         -- Test Case 1: A = 5, B = 5 → equal
43.         A_tb <= "0101"; B_tb <= "0101";
44.         wait for 100 ns;
45.
46.         -- Test Case 2: A = 3, B = 7 → less
47.         A_tb <= "0011"; B_tb <= "0111";
48.         wait for 100 ns;
49.
50.         -- Test Case 3: A = 10, B = 2 → greater
51.         A_tb <= "1010"; B_tb <= "0010";
52.         wait for 100 ns;
53.
54.         -- Test Case 4: A = 0, B = 15 → less
55.         A_tb <= "0000"; B_tb <= "1111";
56.         wait for 100 ns;
57.
58.         -- Test Case 5: A = 15, B = 0 → greater
59.         A_tb <= "1111"; B_tb <= "0000";
60.         wait for 100 ns;
61.
62.         -- Test Case 6: A = 8, B = 8 → equal
63.         A_tb <= "1000"; B_tb <= "1000";
64.         wait for 100 ns;
65.
66.         -- Test Case 7: A = 6, B = 4 → greater
67.         A_tb <= "0110"; B_tb <= "0100";
68.         wait for 100 ns;
69.
70.         -- End simulation
71.         wait;
72.     end process;
73.
74. end behavior;
```

3.4.4 Timing Diagram



3.5 Arithmetic and Logic Unit

3.5.1 VHDL Design Source Code

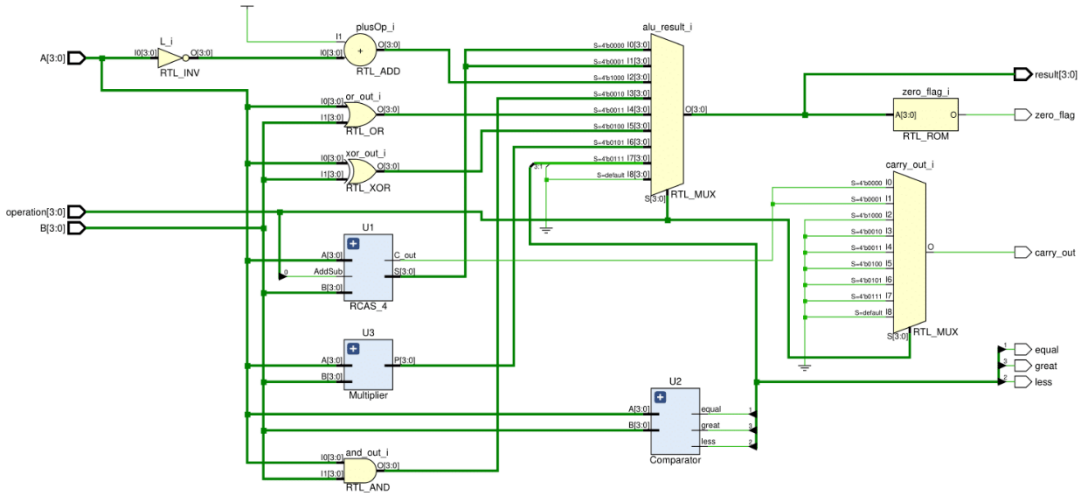
```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use IEEE.NUMERIC_STD.ALL;
 4.
 5. entity ALU is
 6.     Port (
 7.         A          : in  STD_LOGIC_VECTOR(3 downto 0);
 8.         B          : in  STD_LOGIC_VECTOR(3 downto 0);
 9.         operation  : in  STD_LOGIC_VECTOR(3 downto 0);
10.         result     : out STD_LOGIC_VECTOR(3 downto 0);
11.         carry_out  : out STD_LOGIC;
12.         equal      : out STD_LOGIC;
13.         less       : out STD_LOGIC;
14.         great      : out STD_LOGIC;
15.         zero_flag  : out STD_LOGIC
16.     );
17. end ALU;
18.
19. architecture Behavioral of ALU is
20.
```

```
21.     -- Component declarations
22.     component RCAS_4
23.         Port (
24.             A      : in  STD_LOGIC_VECTOR(3 downto 0);
25.             B      : in  STD_LOGIC_VECTOR(3 downto 0);
26.             AddSub : in  STD_LOGIC;
27.             S      : out STD_LOGIC_VECTOR(3 downto 0);
28.             C_out  : out STD_LOGIC
29.         );
30.     end component;
31.
32.     component Comparator
33.         Port (
34.             A      : in  STD_LOGIC_VECTOR(3 downto 0);
35.             B      : in  STD_LOGIC_VECTOR(3 downto 0);
36.             equal  : out STD_LOGIC;
37.             less   : out STD_LOGIC;
38.             great  : out STD_LOGIC
39.         );
40.     end component;
41.
42.     component Multiplier
43.         Port (
44.             A      : in  STD_LOGIC_VECTOR(3 downto 0);
45.             B      : in  STD_LOGIC_VECTOR(3 downto 0);
46.             P      : out STD_LOGIC_VECTOR(3 downto 0)
47.         );
48.     end component;
49.
50.     -- Internal signals
51.     signal sum_out, and_out, or_out, xor_out, neg_out, mul_out :
STD_LOGIC_VECTOR(3 downto 0);
52.     signal cmp_eq, cmp_lt, cmp_gt : STD_LOGIC;
53.     signal carry_tmp : STD_LOGIC;
54.     signal alu_result : STD_LOGIC_VECTOR(3 downto 0);
55.
56. begin
57.
58.     -- Instantiate adder/subtractor
59.     U1: RCAS_4 port map (
60.         A      => A,
61.         B      => B,
62.         AddSub => operation(0),  -- 0 for ADD, 1 for SUB
63.         S      => sum_out,
64.         C_out  => carry_tmp
65.     );
66.
67.     -- Bitwise logic
68.     and_out <= A and B;
69.     or_out  <= A or B;
70.     xor_out <= A xor B;
71.
72.     -- Negation (2's complement of A)
73.     neg_out <= std_logic_vector(unsigned(not A) + 1);
```

```
 74.
 75.     -- Comparator
 76.     U2: Comparator port map (
 77.         A     => A,
 78.         B     => B,
 79.         equal => cmp_eq,
 80.         less  => cmp_lt,
 81.         great => cmp_gt
 82.     );
 83.
 84.     -- Multiplier
 85.     U3: Multiplier port map (
 86.         A => A,
 87.         B => B,
 88.         P => mul_out
 89.     );
 90.
 91.     -- ALU Operation logic
 92.     process (operation, sum_out, and_out, or_out, xor_out, neg_out, cmp_eq,
cmp_lt, cmp_gt, mul_out, carry_tmp)
 93.     begin
 94.         case operation is
 95.             when "0000" =>  -- ADD
 96.                 alu_result <= sum_out;
 97.                 carry_out <= carry_tmp;
 98.             when "0001" =>  -- SUB
 99.                 alu_result <= sum_out;
100.                 carry_out <= carry_tmp;
101.             when "1000" =>  -- NEG
102.                 alu_result <= neg_out;
103.                 carry_out <= '0';
104.             when "0010" =>  -- AND
105.                 alu_result <= and_out;
106.                 carry_out <= '0';
107.             when "0011" =>  -- OR
108.                 alu_result <= or_out;
109.                 carry_out <= '0';
110.             when "0100" =>  -- XOR
111.                 alu_result <= xor_out;
112.                 carry_out <= '0';
113.             when "0101" =>  -- MUL
114.                 alu_result <= mul_out;
115.                 carry_out <= '0';
116.             when "0111" =>  -- CMP: encoded
117.                 alu_result <= cmp_gt & cmp_lt & cmp_eq & '0';  -- Optional
visual encoding
118.                 carry_out <= '0';
119.             when others =>
120.                 alu_result <= (others => '0');
121.                 carry_out <= '0';
122.         end case;
123.     end process;
124.
125.     -- Assign final outputs
```

```
126.     result <= alu_result;
127.     equal  <= cmp_eq;
128.     less   <= cmp_lt;
129.     great  <= cmp_gt;
130.     zero_flag <= '1' when alu_result = "0000" else '0';
131.
132. end Behavioral;
133.
```

3.5.2 Elaborated Design Schematics



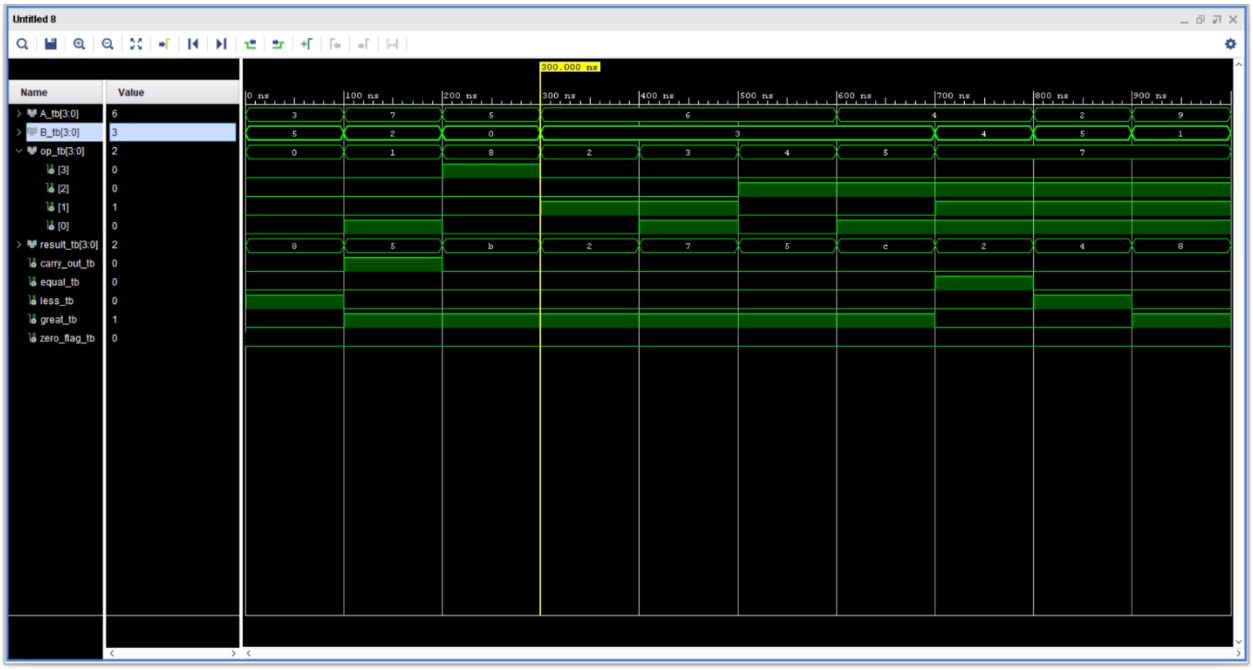
3.5.3 Simulation Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use IEEE.NUMERIC_STD.ALL;
 4.
 5. entity TB_ALU is
 6. end TB_ALU;
 7.
 8. architecture Behavioral of TB_ALU is
 9.
10.     -- DUT component
11.     component ALU
12.         Port (
13.             A         : in  STD_LOGIC_VECTOR(3 downto 0);
```

```
14.             B         : in  STD_LOGIC_VECTOR(3 downto 0);
15.             operation : in  STD_LOGIC_VECTOR(3 downto 0);
16.             result    : out STD_LOGIC_VECTOR(3 downto 0);
17.             carry_out : out STD_LOGIC;
18.             equal     : out STD_LOGIC;
19.             less      : out STD_LOGIC;
20.             great     : out STD_LOGIC;
21.             zero_flag : out STD_LOGIC
22.         );
23.     end component;
24.
25.     -- Signals to connect to DUT
26.     signal A_tb, B_tb     : STD_LOGIC_VECTOR(3 downto 0);
27.     signal op_tb          : STD_LOGIC_VECTOR(3 downto 0);
28.     signal result_tb      : STD_LOGIC_VECTOR(3 downto 0);
29.     signal carry_out_tb   : STD_LOGIC;
30.     signal equal_tb       : STD_LOGIC;
31.     signal less_tb        : STD_LOGIC;
32.     signal great_tb       : STD_LOGIC;
33.     signal zero_flag_tb   : STD_LOGIC;
34.
35. begin
36.
37.     -- Instantiate DUT
38.     UUT: ALU
39.         port map (
40.             A         => A_tb,
41.             B         => B_tb,
42.             operation => op_tb,
43.             result    => result_tb,
44.             carry_out => carry_out_tb,
45.             equal     => equal_tb,
46.             less      => less_tb,
47.             great     => great_tb,
48.             zero_flag => zero_flag_tb
49.         );
50.
51.     -- Stimulus process
52.     stim_proc: process
53.     begin
54.         -- ADD: 3 + 5 = 8
55.         A_tb <= "0011"; B_tb <= "0101"; op_tb <= "0000";
56.         wait for 100 ns;
57.
58.         -- SUB: 7 - 2 = 5
59.         A_tb <= "0111"; B_tb <= "0010"; op_tb <= "0001";
60.         wait for 100 ns;
61.
62.         -- NEG: -A (Two's complement of 5)
63.         A_tb <= "0101"; B_tb <= "0000"; op_tb <= "1000";
64.         wait for 100 ns;
65.
66.         -- AND: 6 and 3 = 2
67.         A_tb <= "0110"; B_tb <= "0011"; op_tb <= "0010";
```

```
68.         wait for 100 ns;
69.
70.         -- OR: 6 or 3 = 7
71.         A_tb <= "0110"; B_tb <= "0011"; op_tb <= "0011";
72.         wait for 100 ns;
73.
74.         -- XOR: 6 xor 3 = 5
75.         A_tb <= "0110"; B_tb <= "0011"; op_tb <= "0100";
76.         wait for 100 ns;
77.
78.         -- MUL: 4 × 3 = 12
79.         A_tb <= "0100"; B_tb <= "0011"; op_tb <= "0101";
80.         wait for 100 ns;
81.
82.         -- CMP: A = 4, B = 4 => equal
83.         A_tb <= "0100"; B_tb <= "0100"; op_tb <= "0111";
84.         wait for 100 ns;
85.
86.         -- CMP: A = 2, B = 5 => less
87.         A_tb <= "0010"; B_tb <= "0101"; op_tb <= "0111";
88.         wait for 100 ns;
89.
90.         -- CMP: A = 9, B = 1 => greater
91.         A_tb <= "1001"; B_tb <= "0001"; op_tb <= "0111";
92.         wait for 100 ns;
93.
94.         wait;
95.     end process;
96.
97. end Behavioral;
98.
```

3.5.4 Timing Diagram



3.6 Improved Nano Processor

3.6.1 VHDL Design Source Code

```
 1. library IEEE;
 2. use IEEE.STD_LOGIC_1164.ALL;
 3. use ieee.numeric_std.all;
 4.
 5. entity Processor is
 6.     port (clk_in : in STD_LOGIC;
 7.           reset : in STD_LOGIC;
 8.           overflow_flag : out STD_LOGIC;
 9.           zero_flag : out STD_LOGIC;
10.           reg7_LED : out STD_LOGIC_VECTOR (3 downto 0);
11.           seven_segment : out STD_LOGIC_VECTOR(6 downto 0);
12.
13.           equal_to     : out STD_LOGIC;
14.           less_than  : out STD_LOGIC;
15.           greater_than     : out STD_LOGIC;
16.
17.           Anode: out STD_LOGIC_VECTOR(3 downto 0)
18.           );
19.
```

```
20.
21. end Processor;
22.
23. architecture Behavioral of Processor is
24. component Program_Rom
25.     Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
26.            O : out STD_LOGIC_VECTOR (11 downto 0));
27. end component;
28. component Instruction_Decoder
29.     Port (Instruction : in STD_LOGIC_VECTOR (11 downto 0);
30.         Chk_Jmp : in STD_LOGIC_VECTOR (3 downto 0);
31.         En_reg : out STD_LOGIC_VECTOR (2 downto 0);
32.         MUX_1 : out STD_LOGIC_VECTOR (2 downto 0);
33.         MUX_2 : out STD_LOGIC_VECTOR (2 downto 0);
34.         Jmp : out STD_LOGIC;
35.         ALU_in : out STD_LOGIC_vector(3 downto 0);
36.         Im_Val : out STD_LOGIC_VECTOR (3 downto 0);
37.         Load_Sel : out STD_LOGIC;
38.         Jmp_Add :  out STD_LOGIC_VECTOR (2 downto 0)
39.         );
40. end component;
41.
42. component Multiplexer_2_way_4_bit
43.     Port ( S : in STD_LOGIC;
44.            A : in STD_LOGIC_VECTOR (3 downto 0);
45.            B : in STD_LOGIC_VECTOR (3 downto 0);
46.            Y : out STD_LOGIC_VECTOR (3 downto 0));
47. end component;
48.
49. component Register_Bank
50.     Port (
51.         S       : in STD_LOGIC_VECTOR (2 downto 0);  -- 3-bit select input
52.         RB_in   : in STD_LOGIC_VECTOR (3 downto 0);  -- 4-bit register input
53.         CLK_in  : in STD_LOGIC;
54.         reset   : in STD_LOGIC;                      -- Clock signal
55.         R0_out  : out STD_LOGIC_VECTOR (3 downto 0);
56.         R1_out  : out STD_LOGIC_VECTOR (3 downto 0);
57.         R2_out  : out STD_LOGIC_VECTOR (3 downto 0);
58.         R3_out  : out STD_LOGIC_VECTOR (3 downto 0);
59.         R4_out  : out STD_LOGIC_VECTOR (3 downto 0);
60.         R5_out  : out STD_LOGIC_VECTOR (3 downto 0);
61.         R6_out  : out STD_LOGIC_VECTOR (3 downto 0);
62.         R7_out  : out STD_LOGIC_VECTOR (3 downto 0)
63.     );
64. end component;
65.
66. component Clock
67.     Port ( Clk_in : in STD_LOGIC;
68.        Clk_out : out STD_LOGIC);
69. end component;
70.
71. component Multiplexer_8_way_4_bit
72.     Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
73.        I1 : in STD_LOGIC_VECTOR (3 downto 0);
```

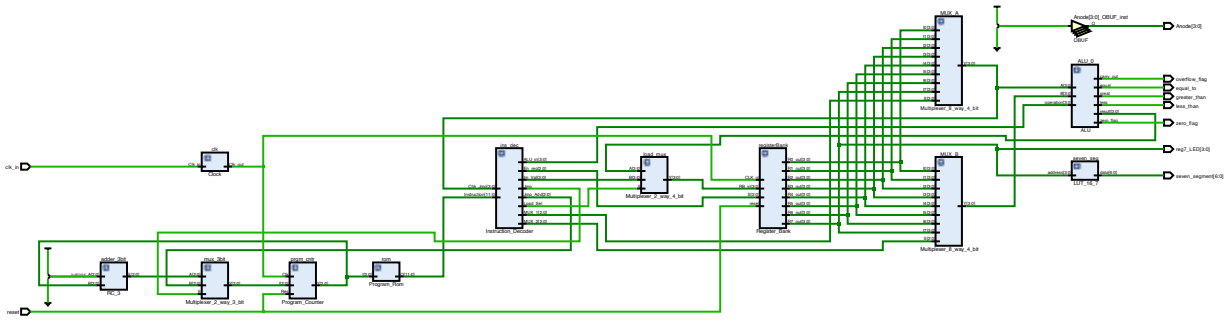
```
        I2 : in STD_LOGIC_VECTOR (3 downto 0);
        I3 : in STD_LOGIC_VECTOR (3 downto 0);
        I4 : in STD_LOGIC_VECTOR (3 downto 0);
        I5 : in STD_LOGIC_VECTOR (3 downto 0);
        I6 : in STD_LOGIC_VECTOR (3 downto 0);
        I7 : in STD_LOGIC_VECTOR (3 downto 0);
        I0 : in STD_LOGIC_VECTOR (3 downto 0);
        Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component ALU
    Port (
        A         : in  STD_LOGIC_VECTOR(3 downto 0);
        B         : in  STD_LOGIC_VECTOR(3 downto 0);
        operation : in  STD_LOGIC_VECTOR(3 downto 0);
        result    : out STD_LOGIC_VECTOR(3 downto 0);
        carry_out : out STD_LOGIC;
        equal     : out STD_LOGIC;
        less      : out STD_LOGIC;
        great     : out STD_LOGIC;
        zero_flag : out STD_LOGIC
    );
end component;

component Multiplexer_2_way_3_bit
    Port ( S : in STD_LOGIC;
       A : in STD_LOGIC_VECTOR (2 downto 0);
       B : in STD_LOGIC_VECTOR (2 downto 0);
       Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component Program_counter
    port (
    clk : in  std_logic;                -- system clock
    Res : in  std_logic;                -- active-high push-button reset
    I   : in  std_logic_vector(2 downto 0);      -- absolute 3-bit jump address
    Y   : out std_logic_vector(2 downto 0)       -- address into program ROM
);    -- address into program ROM
end component;

component RC_3
    Port (
        A     : in  STD_LOGIC_VECTOR(2 downto 0);
        B     : in  STD_LOGIC_VECTOR(2 downto 0);
        S     : out STD_LOGIC_VECTOR(2 downto 0));
end component;

component LUT_16_7
    Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
       data : out STD_LOGIC_VECTOR (6 downto 0));
end component;

signal instruction_out: STD_LOGIC_VECTOR (11 downto 0);
```

```
127. signal mux1_sig, mux2_sig, reg_en, address, address_plus1 : STD_LOGIC_VECTOR (2
downto 0);
128. signal jp, load, add_sub_carry, add_sub_zero: STD_LOGIC;
129. signal  immediate: STD_LOGIC_VECTOR (3 downto 0);
130. signal seven_seg_out: STD_LOGIC_VECTOR (6 downto 0);
131. signal reg_b, muxA_out, muxB_out, alu_out, alu_input: STD_LOGIC_VECTOR  (3
downto 0);
132. signal r0, r1, r2 ,r3, r4, r5 ,r6 ,r7 : STD_LOGIC_VECTOR(3 downto 0);
133. signal clk_o, equal, lesser, greater: STD_LOGIC;
134. signal mux_3bit_o, add_3bit: STD_LOGIC_VECTOR (2 downto 0);
135. signal pc_out: STD_LOGIC_VECTOR(2 downto 0);
136. begin
137.
138. clk: Clock
139.     port map (Clk_in => clk_in,
140.               Clk_out => clk_o);
141.
142. rom: Program_Rom
143.     PORT MAP(I => std_logic_vector(pc_out),
144.              O => instruction_out);
145.
146. ins_dec : Instruction_Decoder
147.     port map( Instruction => instruction_out,
148.               Chk_Jmp => muxA_out,
149.               En_reg => reg_en,
150.               MUX_1 => mux1_sig,
151.               MUX_2 => mux2_sig,
152.               Jmp => jp,
153.               ALU_in => alu_input,
154.               Im_Val => immediate,
155.               Load_Sel => load,
156.               Jmp_Add => address);
157.
158. load_mux: Multiplexer_2_way_4_bit
159.     port map (S => load,
160.               A => alu_out,
161.               B => immediate,
162.               Y => reg_b);
163.
164. registerBank : Register_Bank
165.     port map (RB_in => reg_b,
166.               S => reg_en,
167.               reset => reset,
168.               CLK_in => clk_o,
169.               R0_out => r0,
170.               R1_out => r1,
171.               R2_out => r2,
172.               R3_out => r3,
173.               R4_out => r4,
174.               R5_out => r5,
175.               R6_out => r6,
176.               R7_out => r7);
177.
178. MUX_A: Multiplexer_8_way_4_bit
```

```
179.     port map( S => mux1_sig,
180.               I0 => r0,
181.               I1 => r1,
182.               I2 => r2,
183.               I3 => r3,
184.               I4 => r4,
185.               I5 => r5,
186.               I6 => r6,
187.               I7 => r7,
188.               Y => muxA_out);
189.
190. MUX_B: Multiplexer_8_way_4_bit
191.     port map( S => mux2_sig,
192.               I0 => r0,
193.               I1 => r1,
194.               I2 => r2,
195.               I3 => r3,
196.               I4 => r4,
197.               I5 => r5,
198.               I6 => r6,
199.               I7 => r7,
200.               Y => muxB_out);
201.
202. ALU_0 : ALU
203.     port map(
204.               A => muxA_out,
205.               B => muxB_out,
206.               operation => ALU_input,
207.               result  => alu_out,
208.               carry_out => add_sub_carry,
209.               equal => equal,
210.               less => lesser,
211.               great => greater,
212.               zero_flag => add_sub_zero);
213.
214. adder_3bit: RC_3
215.     port map (A => "001",
216.               B => std_logic_vector(pc_out),
217.               S => add_3bit);
218.
219. mux_3bit: Multiplexer_2_way_3_bit
220.     port map (S => jp,
221.               A =>add_3bit,--- address
222.               B => address,---add_3bit
223.               Y => mux_3bit_o);
224.
225.
226.
227. prgm_cntr : Program_Counter
228.     port map (
229.         clk => clk_o,
230.         Res => reset,
231.         I => mux_3bit_o,
232.         Y => pc_out
```

```
233.     );
234.
235.
236.
237. seven_seg : LUT_16_7
238.     Port map ( address => r7,
239.                data => seven_seg_out);
240.
241. overflow_flag <= add_sub_carry;
242. zero_flag <= add_sub_zero;
243. reg7_LED <= r7;
244. Anode<= "1110";
245. seven_segment <= seven_seg_out;
246.
247. equal_to <= equal;
248. less_than <= lesser;
249. greater_than <= greater;
250.
251. end Behavioral;
252.
```

3.6.2 Elaborated Design Schematics



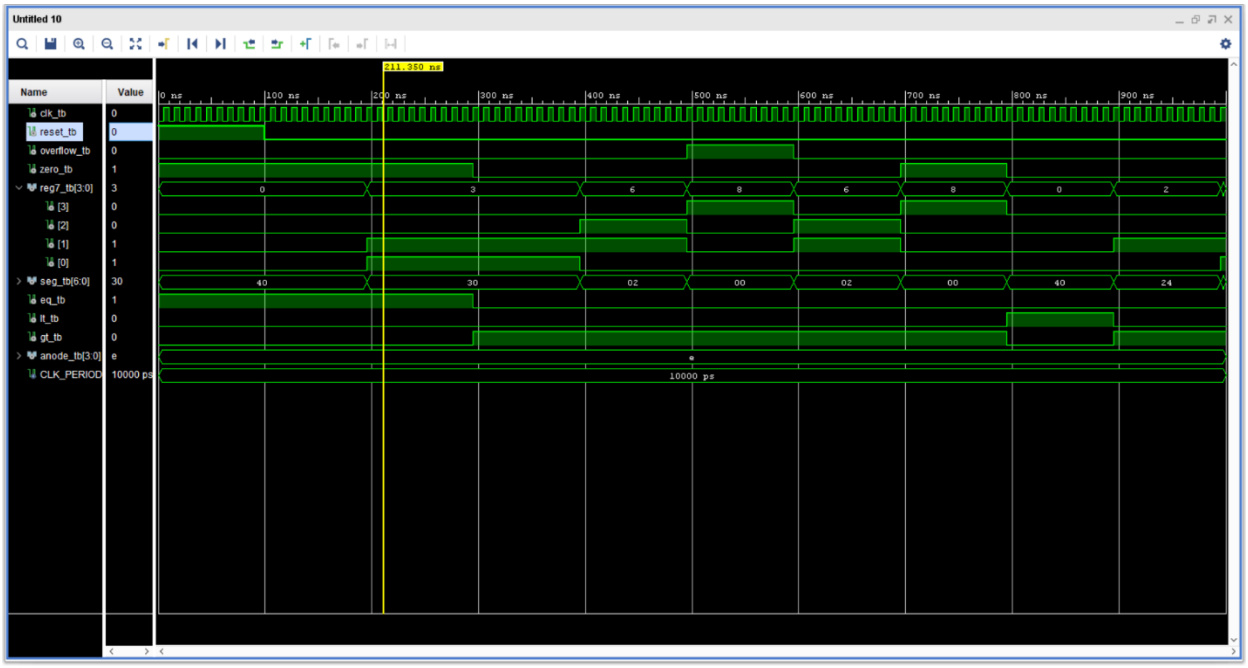
3.6.3 Simulation Source Code

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.NUMERIC_STD.ALL;
4.
5. entity Processor_Testbench is
6. end Processor_Testbench;
7.
8. library IEEE;
```

```
 9. use IEEE.STD_LOGIC_1164.ALL;
10. use IEEE.NUMERIC_STD.ALL;
11.
12. entity TB_Processor is
13. -- No ports for a testbench
14. end TB_Processor;
15.
16. architecture Behavioral of TB_Processor is
17.
18.     -- Component Declaration
19.     component Processor
20.         port (
21.             clk_in : in STD_LOGIC;
22.             reset : in STD_LOGIC;
23.             overflow_flag : out STD_LOGIC;
24.             zero_flag : out STD_LOGIC;
25.             reg7_LED : out STD_LOGIC_VECTOR (3 downto 0);
26.             seven_segment : out STD_LOGIC_VECTOR(6 downto 0);
27.
28.             equal_to     : out STD_LOGIC;
29.             less_than    : out STD_LOGIC;
30.             greater_than : out STD_LOGIC;
31.
32.             Anode        : out STD_LOGIC_VECTOR(3 downto 0)
33.         );
34.     end component;
35.
36.     -- Testbench Signals
37.     signal clk_tb        : STD_LOGIC := '0';
38.     signal reset_tb      : STD_LOGIC := '0';
39.     signal overflow_tb   : STD_LOGIC;
40.     signal zero_tb       : STD_LOGIC;
41.     signal reg7_tb       : STD_LOGIC_VECTOR(3 downto 0);
42.     signal seg_tb        : STD_LOGIC_VECTOR(6 downto 0);
43.     signal eq_tb         : STD_LOGIC;
44.     signal lt_tb         : STD_LOGIC;
45.     signal gt_tb         : STD_LOGIC;
46.     signal anode_tb      : STD_LOGIC_VECTOR(3 downto 0);
47.
48.
49.     -- Clock period constant
50.     constant CLK_PERIOD : time := 10 ns;
51.
52. begin
53.
54.     -- Instantiate the Unit Under Test (UUT)
55.     uut: Processor
56.         port map (
57.             clk_in => clk_tb,
58.             reset => reset_tb,
59.             overflow_flag => overflow_tb,
60.             zero_flag => zero_tb,
61.             reg7_LED => reg7_tb,
62.             seven_segment => seg_tb,
```

```
63.             equal_to => eq_tb,
64.             less_than => lt_tb,
65.             greater_than => gt_tb,
66.             Anode => anode_tb
67.         );
68.
69.     -- Clock generation process
70.     clk_process :process
71.     begin
72.         while true loop
73.             clk_tb <= '0';
74.             wait for CLK_PERIOD / 2;
75.             clk_tb <= '1';
76.             wait for CLK_PERIOD / 2;
77.         end loop;
78.     end process;
79.
80.     -- Stimulus process
81.     stim_proc: process
82.     begin
83.         -- Apply reset
84.         reset_tb <= '1';
85.         wait for 100 ns;
86.         reset_tb <= '0';
87.
88.         -- Wait to let the program execute
89.         wait for 500 ns;
90.
91.         -- You can add assertions or signal monitoring here
92.
93.         wait;  -- Wait forever
94.     end process;
95.
96. end Behavioral;
97.
```

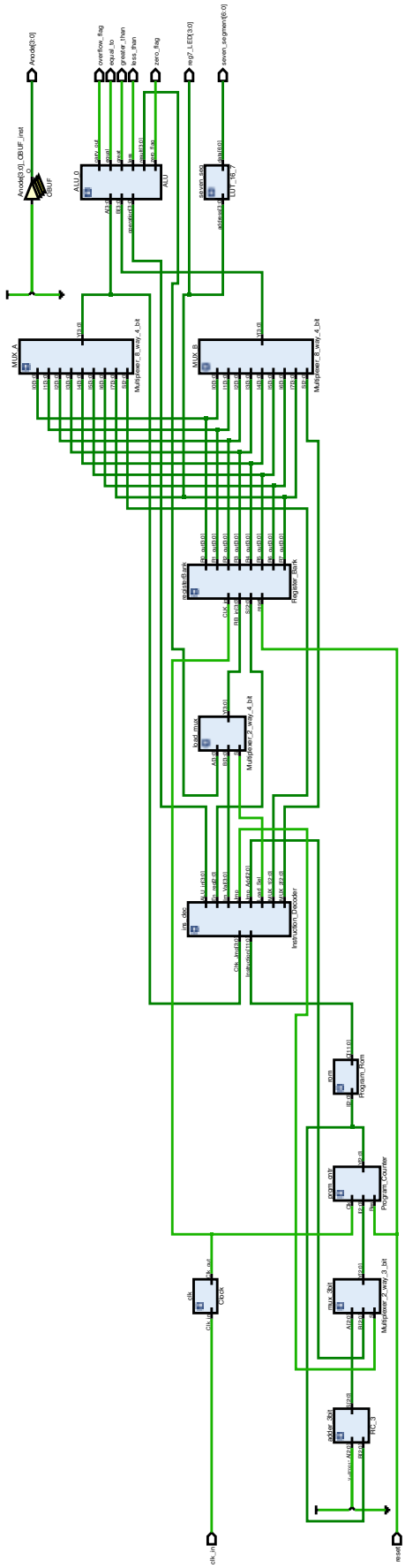
3.6.4 Timing Diagram



3.7 Constraint File

```
 1. ## Clock (clk_in)
 2. set_property PACKAGE_PIN W5 [get_ports clk_in]
 3. set_property IOSTANDARD LVCMOS33 [get_ports clk_in]
 4. create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
clk_in]
 5.
 6. ## Reset Button (reset)
 7. set_property PACKAGE_PIN U18 [get_ports reset]
 8. set_property IOSTANDARD LVCMOS33 [get_ports reset]
 9.
10. ## Register 7 output to LEDs
11. set_property PACKAGE_PIN U16 [get_ports {reg7_LED[0]}]
12. set_property IOSTANDARD LVCMOS33 [get_ports {reg7_LED[0]}]
13. set_property PACKAGE_PIN E19 [get_ports {reg7_LED[1]}]
14. set_property IOSTANDARD LVCMOS33 [get_ports {reg7_LED[1]}]
15. set_property PACKAGE_PIN U19 [get_ports {reg7_LED[2]}]
16. set_property IOSTANDARD LVCMOS33 [get_ports {reg7_LED[2]}]
17. set_property PACKAGE_PIN V19 [get_ports {reg7_LED[3]}]
18. set_property IOSTANDARD LVCMOS33 [get_ports {reg7_LED[3]}]
19.
20. ## Status Flags
21. set_property PACKAGE_PIN L1 [get_ports overflow_flag]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports overflow_flag]
set_property PACKAGE_PIN P1 [get_ports zero_flag]
set_property IOSTANDARD LVCMOS33 [get_ports zero_flag]

## Comparator Flags
set_property PACKAGE_PIN M1 [get_ports equal_to]
set_property IOSTANDARD LVCMOS33 [get_ports equal_to]
set_property PACKAGE_PIN N1 [get_ports less_than]
set_property IOSTANDARD LVCMOS33 [get_ports less_than]
set_property PACKAGE_PIN R2 [get_ports greater_than]
set_property IOSTANDARD LVCMOS33 [get_ports greater_than]

## Seven Segment Display (Segments a-g)
set_property PACKAGE_PIN W7 [get_ports {seven_segment[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seven_segment[0]}]
set_property PACKAGE_PIN W6 [get_ports {seven_segment[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seven_segment[1]}]
set_property PACKAGE_PIN U8 [get_ports {seven_segment[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seven_segment[2]}]
set_property PACKAGE_PIN V8 [get_ports {seven_segment[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seven_segment[3]}]
set_property PACKAGE_PIN U5 [get_ports {seven_segment[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seven_segment[4]}]
set_property PACKAGE_PIN V5 [get_ports {seven_segment[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seven_segment[5]}]
set_property PACKAGE_PIN U7 [get_ports {seven_segment[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seven_segment[6]}]

## Seven Segment Anode Control (Common Cathode Selection)
set_property PACKAGE_PIN U2 [get_ports {Anode[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {Anode[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Anode[1]}]
set_property PACKAGE_PIN V4 [get_ports {Anode[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Anode[2]}]
set_property PACKAGE_PIN W4 [get_ports {Anode[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Anode[3]}]

## Debug Output (optional 4-bit signal)
set_property PACKAGE_PIN N2 [get_ports {debug_out[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {debug_out[0]}]
set_property PACKAGE_PIN P2 [get_ports {debug_out[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {debug_out[1]}]
set_property PACKAGE_PIN T1 [get_ports {debug_out[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {debug_out[2]}]
set_property PACKAGE_PIN T2 [get_ports {debug_out[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {debug_out[3]}]

```


Anode[6:0]
overflow_flag
equal_to
greater_than
less_than
zero_flag
seg7_LED[3:0]
seven_segment[6:0]
Anode[6:0]_OBUF_inst
OBUF
ALU_0
ALU
LUT_16_7
MUX_A
Multiplexer_8_way_4_bit
MUX_B
Multiplexer_8_way_4_bit
Register_Bank
Multiplexer_2_way_4_bit
Instruction_Decoder
Program_Rom
Program_Counter
clk
Clock
Multiplexer_2_way_3_bit
REG_3
clk_in
reset

4. Assembly Code

```
1. MOVI R7, 1    -- store 1 in R7 register
2. MOVI R6, 2    -- store 2 in R6 register
3. MOVI R5, 3    -- store 3 in R5 register
4.
5. ADD R7,R6  -- add value in register R7 and register R6 and store it in Register R7
6. ADD R7,R5  -- add value in register R7 and register R5 and store it in Register R7
7.
```

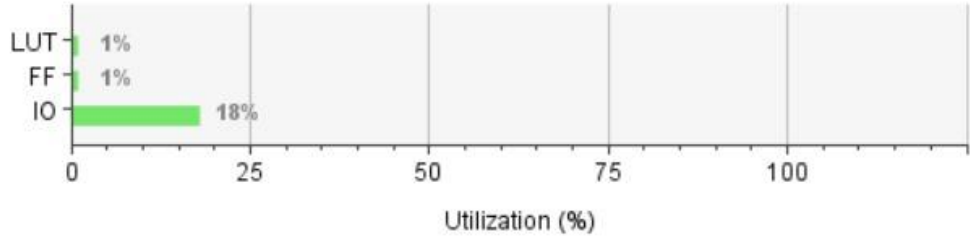
This assembly code can be used to add the integers from 1 to 3 and store it in the R7 Register. Hence, converting these instructions to binary (machine language) is given as follows.

```
1.       "101110000001", --MOVI R7,1
2.       "101100000010", --MOVI R6,2
3.       "101010000011", --MOVI R5,3
4.       "001111100000", --ADD R7, R6
5.       "001111010000", --ADD R7, R5
```

5. Utilization Reports

5.1 Basic Nano processor

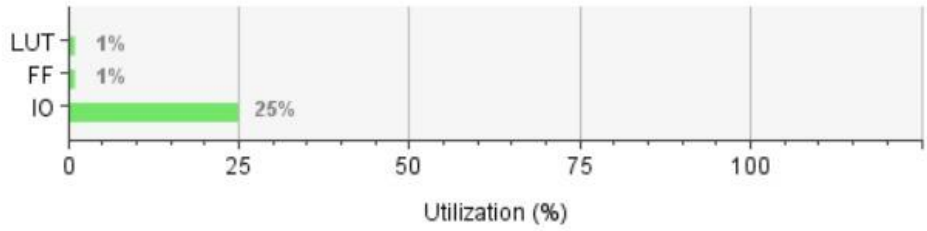
Resource	Utilization	Available	Utilization %
LUT	37	20800	0.18
FF	56	41600	0.13
IO	19	106	17.92



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
N Processor	37	56	22	37	5	19	1
clk (Clock)	11	34	14	11	2	0	0
prgm_cntr (Program_...	22	3	6	22	2	0	0
registerBank (Register...	4	19	6	4	0	0	0
Register_R1 (Regis...	0	4	1	0	0	0	0
Register_R5 (Regis...	0	4	1	0	0	0	0
Register_R6 (Regis...	0	4	1	0	0	0	0
Register_R7 (Regis...	4	7	3	4	0	0	0

5.2 Improved Nano processor

Resource	Utilization	Available	Utilization %
LUT	60	20800	0.29
FF	50	41600	0.12
IO	26	106	24.53



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
N Processor	60	50	28	60	8	26	1
clk (Clock)	11	34	14	11	2	0	0
ins_dec (Instruction_Decoder)	6	3	5	6	0	0	0
prgm_cntr (Program_Counter)	33	3	15	33	2	0	0
registerBank (Register_Bank)	10	10	7	10	1	0	0
Register_R2 (Register_4_bit)	2	4	3	2	0	0	0
Register_R7 (Register_4_bit_0)	8	6	5	8	1	0	0

6. Conclusion

This project gave us a well understanding of the internal structure of a nano processor, the integration and collaboration of various components within a processor, the process of instruction decoding, and instruction storage.

We had to debug some outputs due to the fact that initially, the required output was not displaying.

Due to the limited 4-bit width of our microprocessor, we were only able to solve smaller problems, such as calculating the total of integers between 1 and 3.

Since the microprocessor operates solely in machine language, we had to hardcode assembly instructions as binary values.

7. Contributions

1. Abeysundara K.N.B – 230010L
    a. 2 way 4 bit Multiplexer.
    b. 4 way 4 bit Multiplexer.
    c. 8 way 4 bit Multiplexer.
    d. 2 way 3 bit Multiplexer.
    e. 2-4 Decoder
    f. 3-8 Decoder
    g. Instruction Decoder
    h. Improved Nanoprocessor Integration

2. Senevirathne S.N – 230601B
    a. Ripple Carry Adder and Subtractor
    b. Comparator
    c. Multiplier
    d. ALU
    e. Bit Operators
    f. Constraint Files
    g. Basic Nanoprocessor integration

3. H.M Pasindu Lakmal – 230261L
    a. Program ROM
    b. Register Bank

4. Pankaja Waidyasekara – 230678N
    a. Nanoprocessor LUT – clock
    b. Nanoprocessor LUT – Program Counter