

جامعة نيويورك أبوظبي



NYU ABU DHABI

ADVANCED DIGITAL LOGIC
ENGR – UH 2310

Lab 1

GROUP 1
SPRING, 2025

This report is entirely our own work, and we have kept a copy for our own records. We are aware of the University's policies on cheating, plagiarism, and the resulting consequences of their breach.

Submitted by:

Name	Net ID
Damiane Kapanadze	dk4770
Seoyoon Jung	sj4260
Sipan Hovsepian	sh7437

Contents

1	Task 1: 4-input NAND gate	2
1.1	VHDL Code	2
1.2	Simulation: VHDL Testbench Code	2
1.3	Constraints	4
1.4	Test Cases on FPGA Board	5
2	Task 2: Half adder	7
2.1	VHDL Code	7
2.2	Simulation: VHDL Testbench Code	7
2.3	Constraints	9
2.4	Test Cases on FPGA Board	10
3	Task 3: 4-bit binary adder	12
3.1	VHDL Code	12
3.2	Simulation	13
3.3	Constraints	16
3.4	Test Cases on FPGA Board	17
4	Task 4: Implementing given 4-input Minterms	19
4.1	VHDL code	19
4.2	Simulation: VHDL Testbench Code	20
4.3	Constraints	22
4.4	Test Cases on FPGA Board	22
5	Task 5: 8:1 MUX	25
5.1	VHDL code	25
5.2	Theoretical question	26
5.3	Simulation: VHDL Testbench Code	27
5.4	Constraints	30
5.5	Test Cases on FPGA Board	30

1 Task 1: 4-input NAND gate

1.1 VHDL Code

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity NANDGATE_4 is      -- NANDGATE_4 is identifier name for a 4
   input NAND GATE
5   Port ( A : in  STD_LOGIC;          --Input A of the
        NAND Gate
6         B : in  STD_LOGIC;
7         C : in  STD_LOGIC;
8         D : in  STD_LOGIC;
9         O : out  STD_LOGIC);        --Output O
10 end NANDGATE_4;
11
12 architecture Behavioral of NANDGATE_4 is
13
14 begin
15     -- The output O is assigned as the negation of the AND
16     operation of inputs A, B, C, and D
17 O <= not (A and B and C and D);
18
19 end Behavioral;
```

Listing 1: VHDL code for Lab 1_1

1.2 Simulation: VHDL Testbench Code

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;
3
4 -- Testbench entity for the 4-input NAND gate. No ports are
5 -- needed since this is a self-contained testbench.
6
7 ENTITY lab1_tb IS
8 END lab1_tb;
9
10 ARCHITECTURE behavior OF lab1_tb IS
11
12     -- Component declaration for the 4-input NAND gate under
13     test.
14
15 COMPONENT NANDGATE_4
16 PORT(
17     A : IN  std_logic;
18     B : IN  std_logic;
19     C : IN  std_logic;
```

```

18         D : IN   std_logic;
19         O : OUT  std_logic           --result of NAND
           operation
20     );
21     END COMPONENT;
22
23
24     -- Testbench signals for providing inputs to the NAND gate.
25
26     signal A_tb : std_logic := '0';
27     signal B_tb : std_logic := '0';
28     signal C_tb : std_logic := '0';
29     signal D_tb : std_logic := '0';
30
31     -- Testbench signal to capture the output from the NAND
           gate.
32     signal O_tb : std_logic;
33
34
35
36 BEGIN
37     -- Instantiation of the NANDGATE_4 component.
38
39     uut: NANDGATE_4 PORT MAP (
40         A => A_tb,
41         B => B_tb,
42         C => C_tb,
43         D => D_tb,
44         O => O_tb
45     );
46
47     -- Stimulus process for input A:
48     -- Waits for 200 ns and then inverts signal A_tb.
49
50
51     stim_proc: process
52     begin
53         wait for 200 ns;
54         A_tb <= not A_tb;
55     end process;
56
57     stim_proc2: process
58     begin
59         wait for 100 ns;
60         B_tb <= not B_tb;
61     end process;
62
63     stim_proc3: process
64     begin
65         wait for 50 ns;
66         C_tb <= not C_tb;

```

```

67     end process;
68
69     stim_proc4: process
70     begin
71         wait for 25 ns;
72         D_tb <= not D_tb;
73     end process;
74 END;

```

Listing 2: Simulation code for Lab 1_1



Figure 1: Simulation for test cases in task 1: Output=0

As expected, NAND gate outputs 0 if and only if all the inputs are 1.



Figure 2: Simulation for testcases in task 1: Output=1

In all other cases, the output of NAND Gate is 1.

1.3 Constraints

```

1 NET "A" LOC = "P11";
2 NET "B" LOC = "L3";
3 NET "c" LOC = "K3";
4 NET "D" LOC = "B4";
5 NET "O" LOC = "P4";

```

Listing 3: Constraints file for Lab 1_1

1.4 Test Cases on FPGA Board

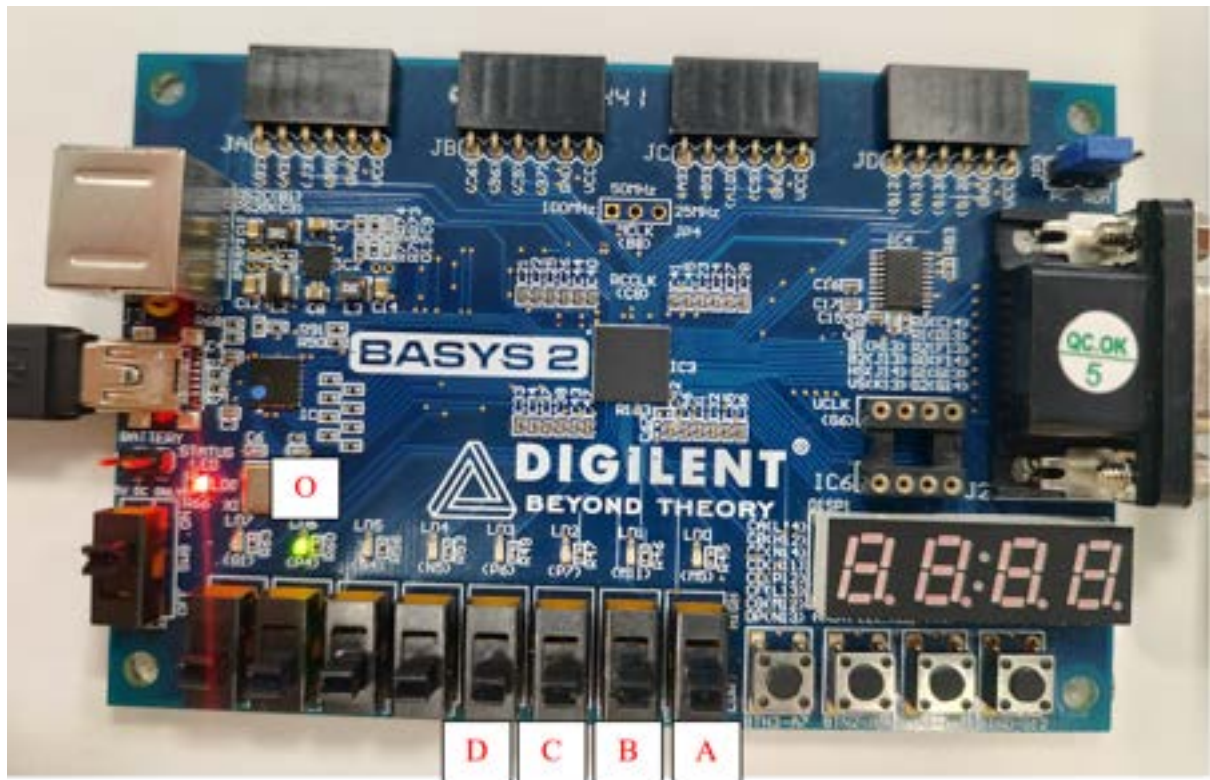


Figure 3: NAND(0000) - expected 1 ✓

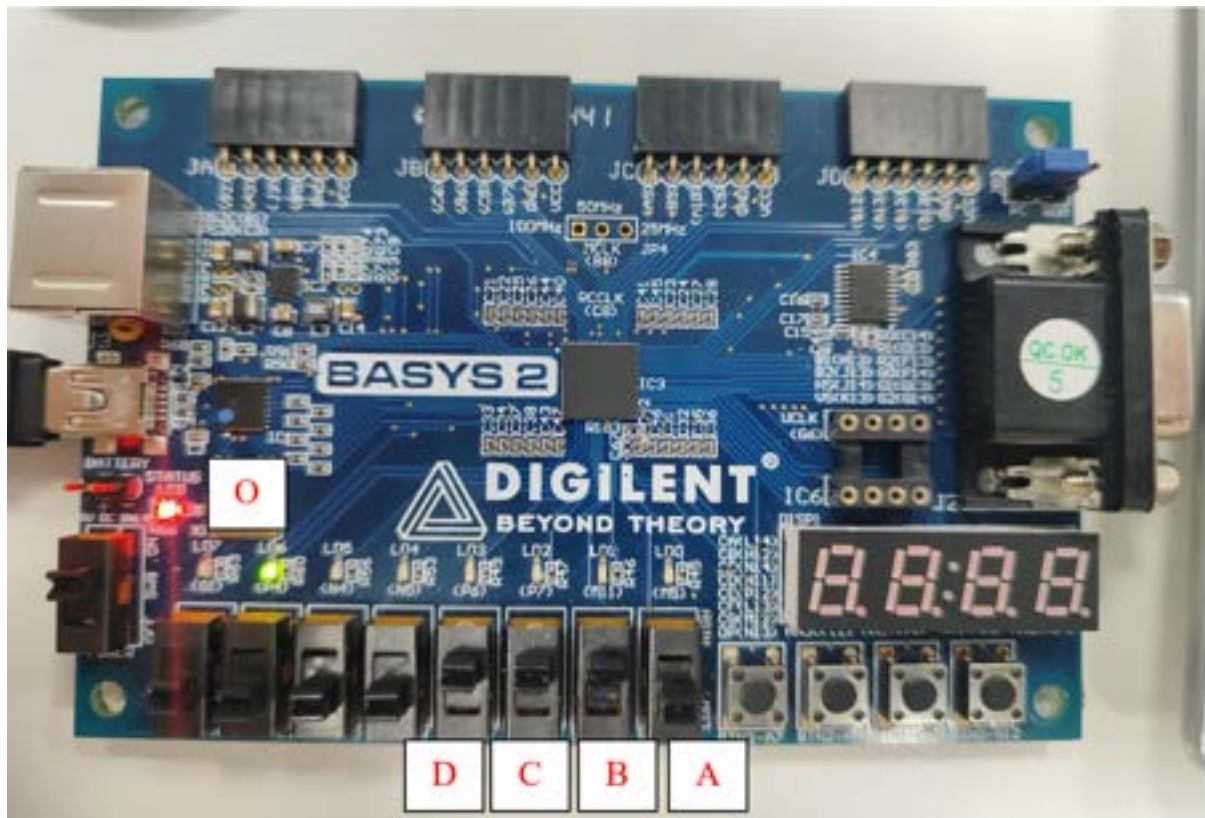


Figure 4: NAND(0111) - expected 0 ✓

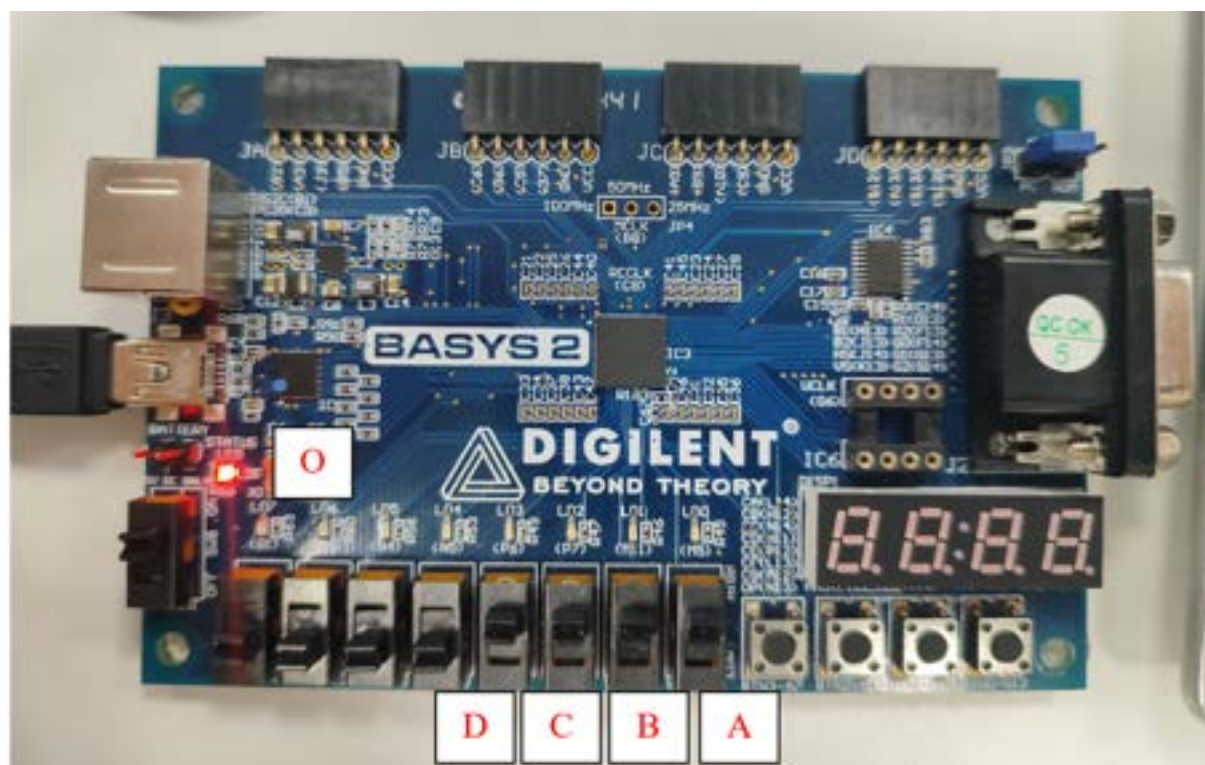


Figure 5: NAND(1111) - expected 0 ✓

2 Task 2: Half adder

2.1 VHDL Code

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity half_adder is          --half_adder is identifier name
5     Port ( A : in  STD_LOGIC;    --first input
6           B : in  STD_LOGIC;    --second input
7           S : out  STD_LOGIC;    --sum
8           C : out  STD_LOGIC);  --carry
9 end half_adder;
10
11
12 architecture Behavioral of half_adder is
13
14 begin
15     S <= A xor B;              --the XOR relation can be derived from
16                                truth table
17     C <= A and B;
18 end Behavioral;
```

Listing 4: VHDL code for Lab 1.2

2.2 Simulation: VHDL Testbench Code

```
1
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.ALL;
4
5     -- testbench entity for half adder, no ports are
6     -- needed since this is a self-contained
7     -- testbench
8
9 ENTITY lab1_task2_tb IS
10 END lab1_task2_tb;
11
12 ARCHITECTURE behavior OF lab1_task2_tb IS
13
14     -- Component Declaration for the half adder
15
16     COMPONENT half_adder
17     PORT(
18         A : IN  std_logic;
19         B : IN  std_logic;
20         S : OUT  std_logic;
21         C : OUT  std_logic
22     );
23 END COMPONENT;
```



```

22
23         -- testbench signals for providing inputs to the half
           adder
24
25     --Inputs
26     signal A_tb : std_logic := '0';
27     signal B_tb : std_logic := '0';
28
29         -- testbench signals of outputs
30     --Outputs
31     signal S_tb : std_logic;
32     signal C_tb : std_logic;
33
34
35 BEGIN
36
37         -- Instantiation of half adder
38     uut: half_adder PORT MAP (
39         A => A_tb,
40         B => B_tb,
41         S => S_tb,
42         C => C_tb
43     );
44
45
46     -- Stimulus process, times are assigned to ensure different
           values for inputs
47     stim_proc1: process
48     begin
49         wait for 100 ns;
50         A_tb <= not A_tb;
51     end process;
52
53         stim_proc2: process
54     begin
55         wait for 50 ns;
56         B_tb <= not B_tb;
57     end process;
58
59 END;
```

Listing 5: Simulation file for Lab 1_2

The simulation results for all 4 cases match the expected results.

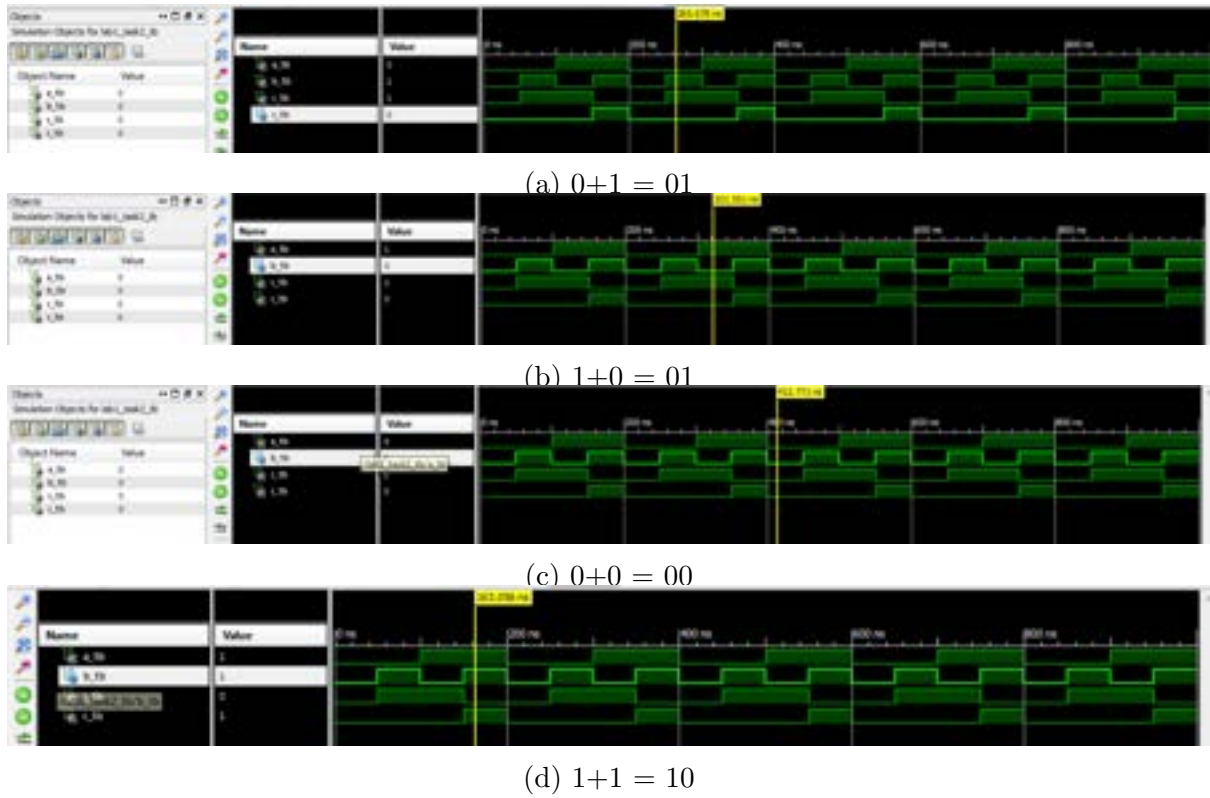


Figure 6: Simulation for test cases in task 2

2.3 Constraints

```

1 NET "A" LOC = "L3";
2 NET "B" LOC = "P11";
3 NET "S" LOC = "M11";
4 NET "C" LOC = "M5";

```

Listing 6: Constraints file for Lab 1.2

2.4 Test Cases on FPGA Board

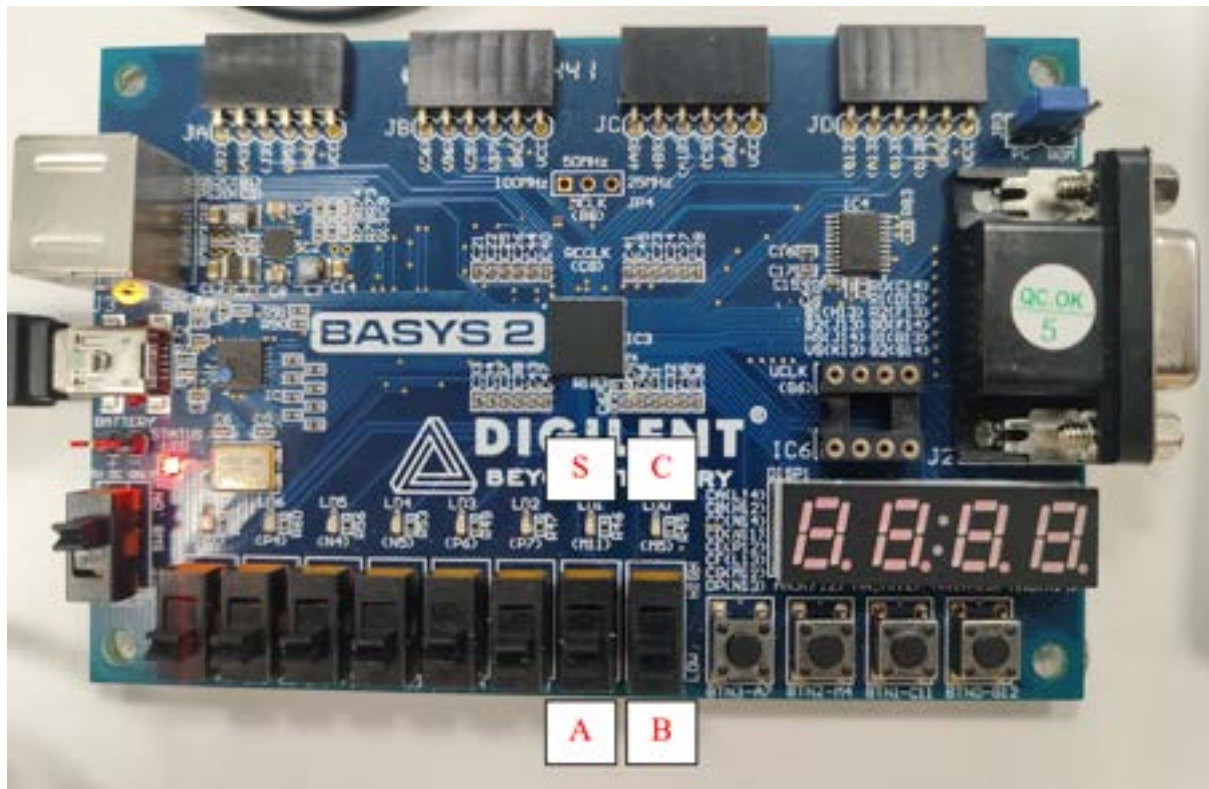


Figure 7: $0 + 0$ - expected 00 ✓

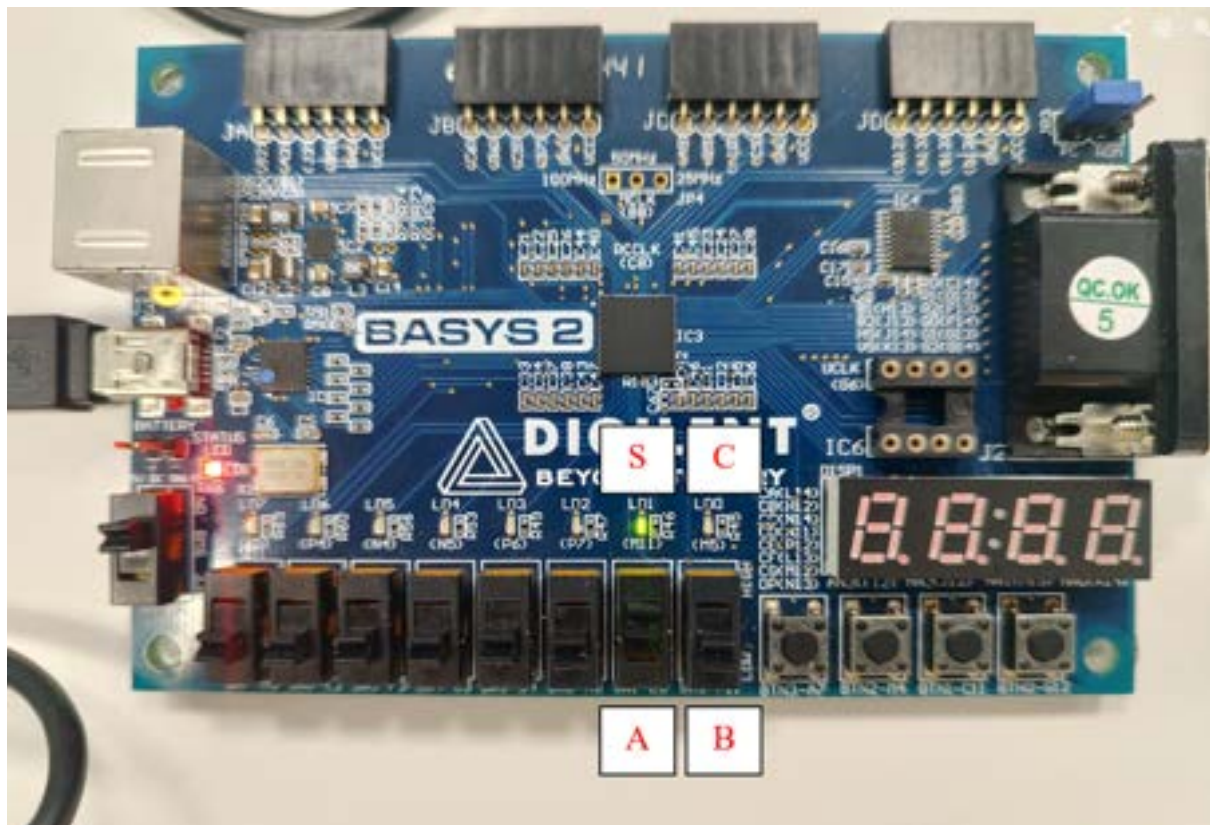


Figure 8: $1 + 0$ - expected 01 ✓

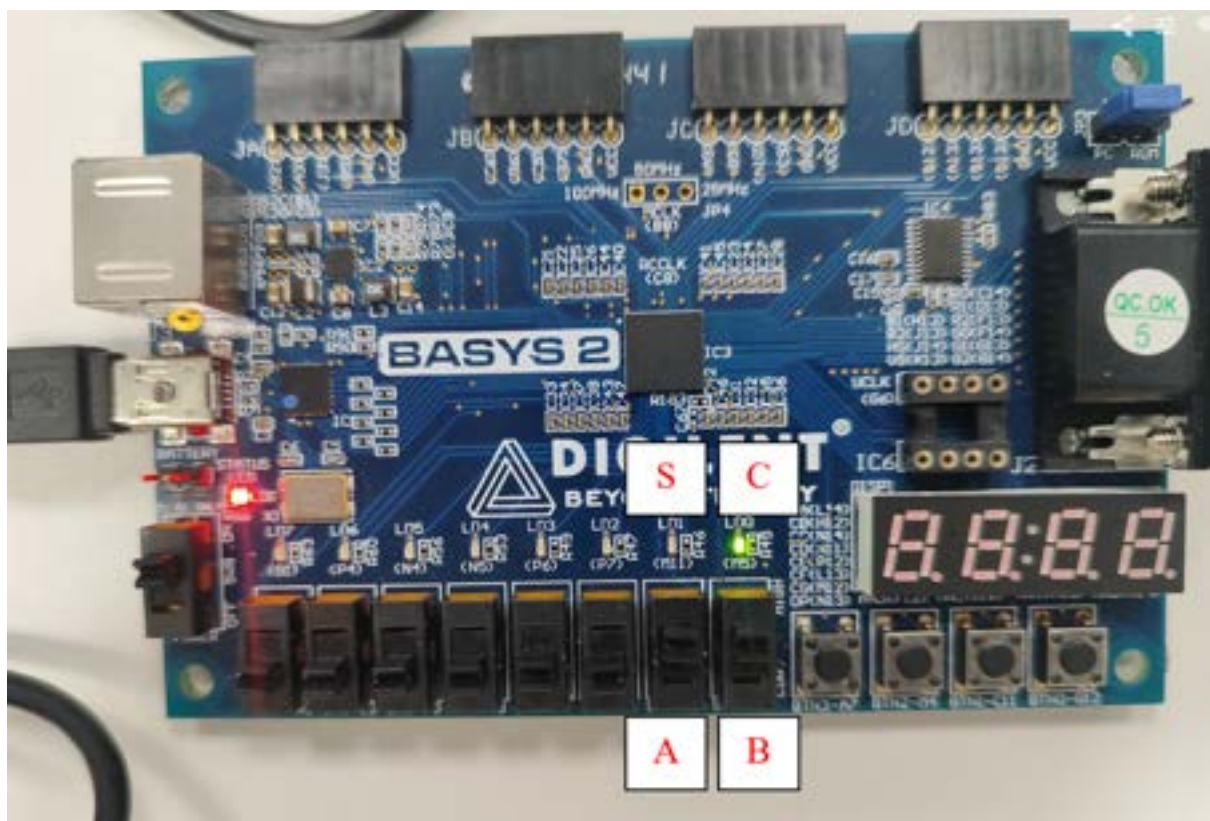


Figure 9: $1 + 1$ - expected 10 ✓

3 Task 3: 4-bit binary adder

3.1 VHDL Code

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4
5 entity adder_4bit is                                --adder_4bit is
6     identifier name
7     Port ( A0 : in  STD_LOGIC;
8            A1 : in  STD_LOGIC;
9            A2 : in  STD_LOGIC;
10           A3 : in  STD_LOGIC;
11           B0 : in  STD_LOGIC;
12           B1 : in  STD_LOGIC;
13           B2 : in  STD_LOGIC;
14           B3 : in  STD_LOGIC;
15           S0 : out  STD_LOGIC;
16           S1 : out  STD_LOGIC;
17           S2 : out  STD_LOGIC;
18           S3 : out  STD_LOGIC;
19           C0 : out  STD_LOGIC); -- carry
20 end adder_4bit;
21
22 architecture Behavioral of adder_4bit is
23 begin
24     S0 <= A0 xor B0;
25     S1 <= (A1 xor B1) xor (A0 and B0);
26     S2 <= (A2 xor B2) xor ((A1 and B1) or ((A0 and B0) and (A1
27     xor B1)));
28     S3 <= (A3 xor B3) xor ((A2 and B2) or (((A1 and B1) or ((A0
29     and B0) and (A1 xor B1))) and (A2 xor B2)));
30     C0 <= (A3 and B3) or (((A2 and B2) or (((A1 and B1) or ((A0
31     and B0) and (A1 xor B1))) and (A2 xor B2))) and (A3 xor B3)
32     );
33
34 -- relations are derived from truth table
35
36 end Behavioral;
```

Listing 7: VHDL code for Lab 1.3

3.2 Simulation

```
1
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.ALL;
4
5     -- testbench entity for 4 bit adder, no ports are needed
6     -- since this is a self contained testbench
7
8 ENTITY lab1_task3_tb IS
9
10 ARCHITECTURE behavior OF lab1_task3_tb IS
11
12     -- Component Declaration for the 4 bit adder
13
14     COMPONENT adder_4bit
15     PORT(
16         A0 : IN    std_logic;
17         A1 : IN    std_logic;
18         A2 : IN    std_logic;
19         A3 : IN    std_logic;
20         B0 : IN    std_logic;
21         B1 : IN    std_logic;
22         B2 : IN    std_logic;
23         B3 : IN    std_logic;
24         S0 : OUT   std_logic;
25         S1 : OUT   std_logic;
26         S2 : OUT   std_logic;
27         S3 : OUT   std_logic;
28         C0 : OUT   std_logic
29     );
30 END COMPONENT;
31
32     --testbench signals for providing inputs to the 4 bit
33     --adder
34
35 --Inputs
36 signal A0_tb : std_logic := '0';
37 signal A1_tb : std_logic := '0';
38 signal A2_tb : std_logic := '0';
39 signal A3_tb : std_logic := '0';
40 signal B0_tb : std_logic := '0';
41 signal B1_tb : std_logic := '0';
42 signal B2_tb : std_logic := '0';
43 signal B3_tb : std_logic := '0';
44
45 --Outputs
46 signal S0_tb : std_logic;
47 signal S1_tb : std_logic;
48 signal S2_tb : std_logic;
```

```

48     signal S3_tb : std_logic;
49     signal C0_tb : std_logic;
50
51
52
53 BEGIN
54
55
56     uut: adder_4bit PORT MAP (
57         A0 => A0_tb,
58         A1 => A1_tb,
59         A2 => A2_tb,
60         A3 => A3_tb,
61         B0 => B0_tb,
62         B1 => B1_tb,
63         B2 => B2_tb,
64         B3 => B3_tb,
65         S0 => S0_tb,
66         S1 => S1_tb,
67         S2 => S2_tb,
68         S3 => S3_tb,
69         C0 => C0_tb
70     );
71
72
73
74     -- Stimulus process
75     stim_proc: process
76
77         -- 3 randomly chosen test cases were selected (otherwise
78         it could be more
79
80     begin
81         --test 1
82         A0_tb <= '1';
83         A1_tb <= '1';
84         A2_tb <= '1';
85         A3_tb <= '1';
86         B0_tb <= '0';
87         B1_tb <= '0';
88         B2_tb <= '1';
89         B3_tb <= '0';
90
91         wait for 100 ns;
92         --test 2
93         A0_tb <= '0';
94         A1_tb <= '0';
95         A2_tb <= '0';
96         A3_tb <= '1';
97         B0_tb <= '1';
98         B1_tb <= '1';

```



```

98         B2_tb <= '1';
99         B3_tb <= '1';
100
101     wait for 150 ns;
102     --test 3
103         A0_tb <= '1';
104         A1_tb <= '1';
105         A2_tb <= '1';
106         A3_tb <= '1';
107         B0_tb <= '1';
108         B1_tb <= '1';
109         B2_tb <= '1';
110         B3_tb <= '1';
111     wait for 150 ns;
112
113
114     end process;
115
116 END;

```

Listing 8: Simulation code for Lab 1.3



Figure 10: Simulation for test cases in task 3: $1111 + 0100 = 10011$



Figure 11: Simulation for test cases in task 3: $1000 + 1111 = 10111$

The simulation results for all 3 cases match the expected results.



Figure 12: Simulation for test cases in task 3: $1111 + 1111 = 11110$

3.3 Constraints

```

1 Net "A3" LOC = "N3";
2 Net "A2" LOC = "E2";
3 Net "A1" LOC = "F3";
4 Net "A0" LOC = "G3";
5 Net "B3" LOC = "B4";
6 Net "B2" LOC = "K3";
7 Net "B1" LOC = "L3";
8 Net "B0" LOC = "P11";
9 Net "S3" LOC = "P6";
10 Net "S2" LOC = "P7";
11 Net "S1" LOC = "M11";
12 Net "S0" LOC = "M5";
13 Net "C0" LOC = "N5";

```

Listing 9: Constraints file for Lab 1_3

3.4 Test Cases on FPGA Board

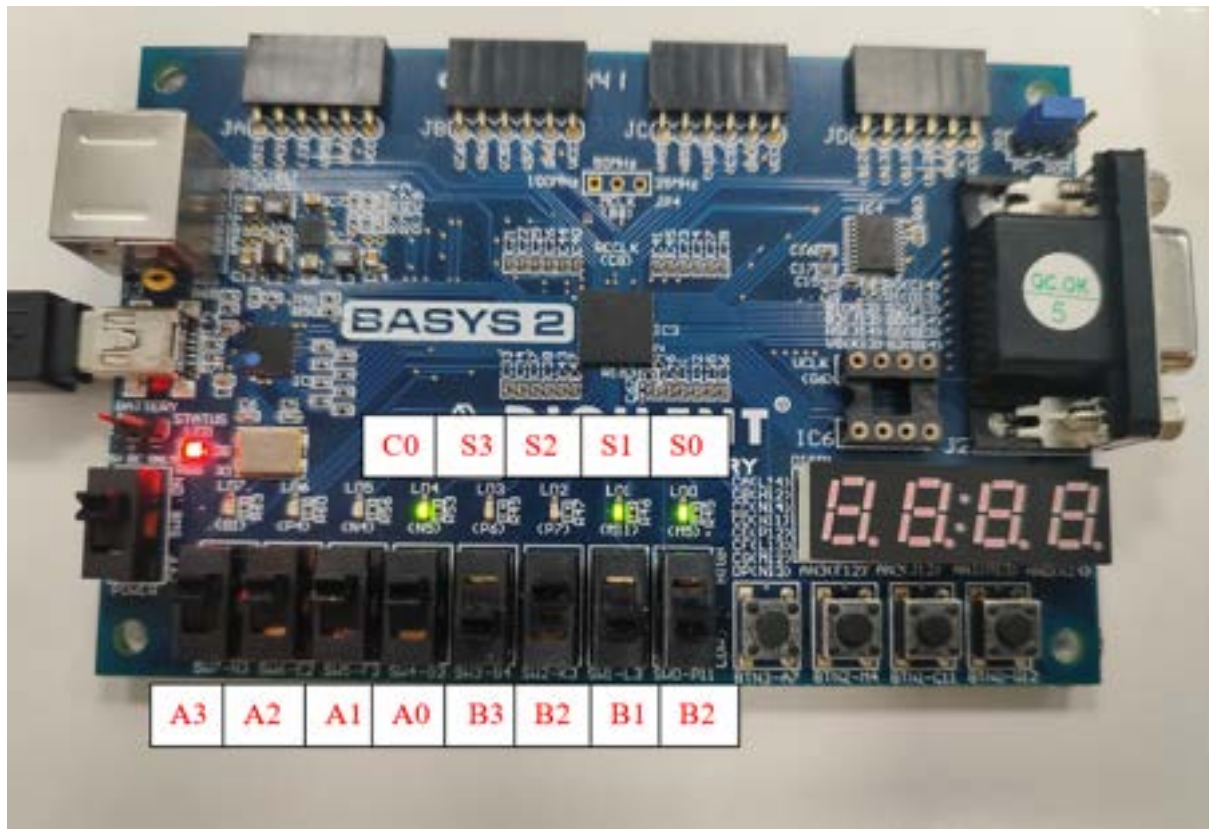


Figure 13: $1111 + 0100$: expected 10011 ✓

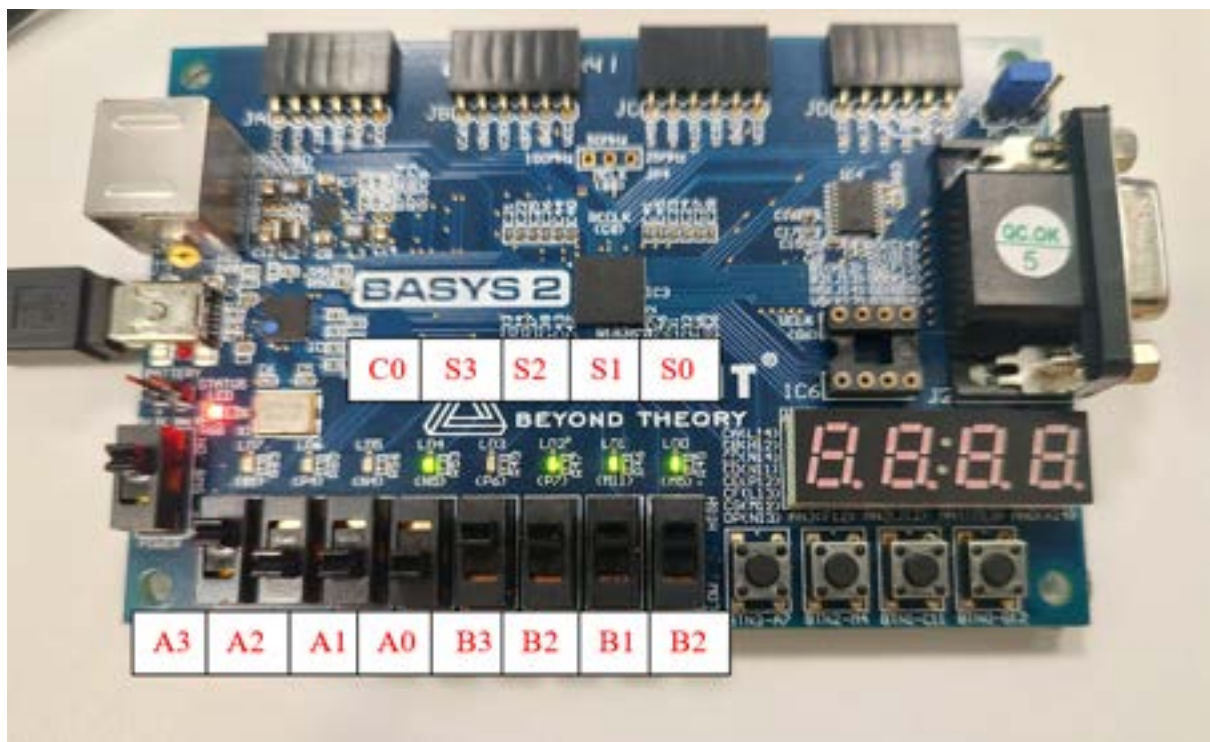


Figure 14: $1000 + 1111$: expected 10111 ✓

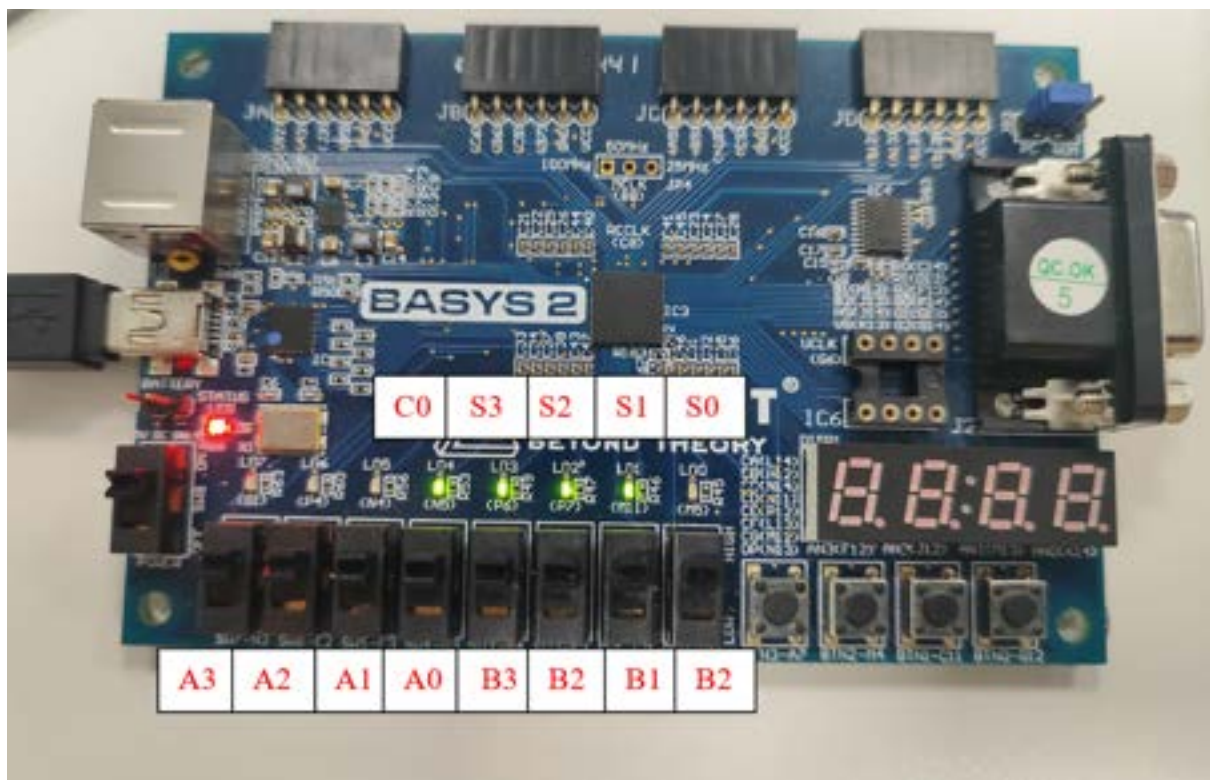


Figure 15: $1111 + 1111$: expected 11110 ✓

4 Task 4: Implementing given 4-input Minterms

4.1 VHDL code

F1 is implemented by concurrent signal assignment; F2 is implemented by sequential Case-When statements.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity minterm is
5     Port ( A : in  STD_LOGIC;
6           B : in  STD_LOGIC;
7           C : in  STD_LOGIC;
8           D : in  STD_LOGIC;
9           F1 : out STD_LOGIC;
10          F2 : out STD_LOGIC);
11 end minterm;
12
13 architecture Behavioral of minterm is
14
15 begin
16
17     -- 1. Concurrent signal assignment (logic equation)
18     F1 <= (not A and not B and not C and D) -- Minterm 1 (0001)
19         or (not A and not B and C and D)    -- Minterm 3 (0011)
20         or (A and not B and not C and D)    -- Minterm 9 (1001)
21         or (A and not B and C and D);       -- Minterm 11 (1011)
22
23     -- 2. Sequential (Case-When inside a process)
24     process(A, B, C, D)
25         variable input_vector : std_logic_vector(3 downto 0);
26     begin
27         input_vector := (A & B & C & D); -- Convert inputs to a
28         4-bit vector
29         F2 <= '0'; -- F2 default value
30         case input_vector is
31             when "0001" => F2 <= '1'; -- Minterm 1
32             when "0011" => F2 <= '1'; -- Minterm 3
33             when "1001" => F2 <= '1'; -- Minterm 9
34             when "1011" => F2 <= '1'; -- Minterm 11
35             when others => F2 <= '0'; -- Default case to prevent
36                                     latch
37         end case;
38     end process;
39 end Behavioral;
```

Listing 10: VHDL code for Lab 1.4

4.2 Simulation: VHDL Testbench Code

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY minterm_tb IS
5  END minterm_tb;
6
7  ARCHITECTURE behavior OF minterm_tb IS
8
9      -- Component Declaration
10     COMPONENT minterm
11     PORT(
12         A : IN  std_logic;
13         B : IN  std_logic;
14         C : IN  std_logic;
15         D : IN  std_logic;
16         F1 : OUT std_logic;
17         F2 : OUT std_logic
18     );
19     END COMPONENT;
20
21     -- Inputs
22     SIGNAL A_tb : std_logic := '0';
23     SIGNAL B_tb : std_logic := '0';
24     SIGNAL C_tb : std_logic := '0';
25     SIGNAL D_tb : std_logic := '0';
26
27     -- Outputs
28     SIGNAL F1_tb : std_logic;
29     SIGNAL F2_tb : std_logic;
30
31 BEGIN
32     -- Instantiation
33     uut: minterm PORT MAP (
34         A => A_tb,
35         B => B_tb,
36         C => C_tb,
37         D => D_tb,
38         F1 => F1_tb,
39         F2 => F2_tb
40     );
41
42     -- Stimulus process
43     stim_proc1: process
44     begin
45         -- Toggle A every 200 ns
46         wait for 200 ns;
47         A_tb <= not A_tb;
48     end process;
49
```

```

50 stim_proc2: process
51 begin
52     -- Toggle B every 100 ns
53     wait for 100 ns;
54     B_tb <= not B_tb;
55 end process;
56
57 stim_proc3: process
58 begin
59     -- Toggle C every 50 ns
60     wait for 50 ns;
61     C_tb <= not C_tb;
62 end process;
63
64 stim_proc4: process
65 begin
66     -- Toggle D every 25 ns
67     wait for 25 ns;
68     D_tb <= not D_tb;
69 end process;
70
71 end behavior;

```

Listing 11: VHDL Testbench code Lab 1.4



Figure 16: Simulation for task 4

The simulation demonstrates that F1 and F2 are identically activated if and only if the inputs are 0001, 0011, 1001, and 1011.

4.3 Constraints

```
1 Net "A" LOC = "N3";  
2 Net "B" LOC = "E2";  
3 Net "C" LOC = "F3";  
4 Net "D" LOC = "G3";  
5 Net "F1" LOC = "M11";  
6 Net "F2" LOC = "M5";
```

Listing 12: Constrains for Lab 1_4

4.4 Test Cases on FPGA Board

Below are testcases for each Minterm: 1(0001), 3(0011), 9(1001), 11(1011), and an extra case that is not a Minterm: 5(0101). The LEDs F1 and F2 are the output signal, where F1 is defined by the Concurrent signal assignment (logic equation) and F2 is defined by the Sequential assignment (Case-When). Note that activation of F1 and F2 are identical, meaning both methods of implementation produce the same result.

Figure 5 is a testcase for a non-minterm (i.e. 5).

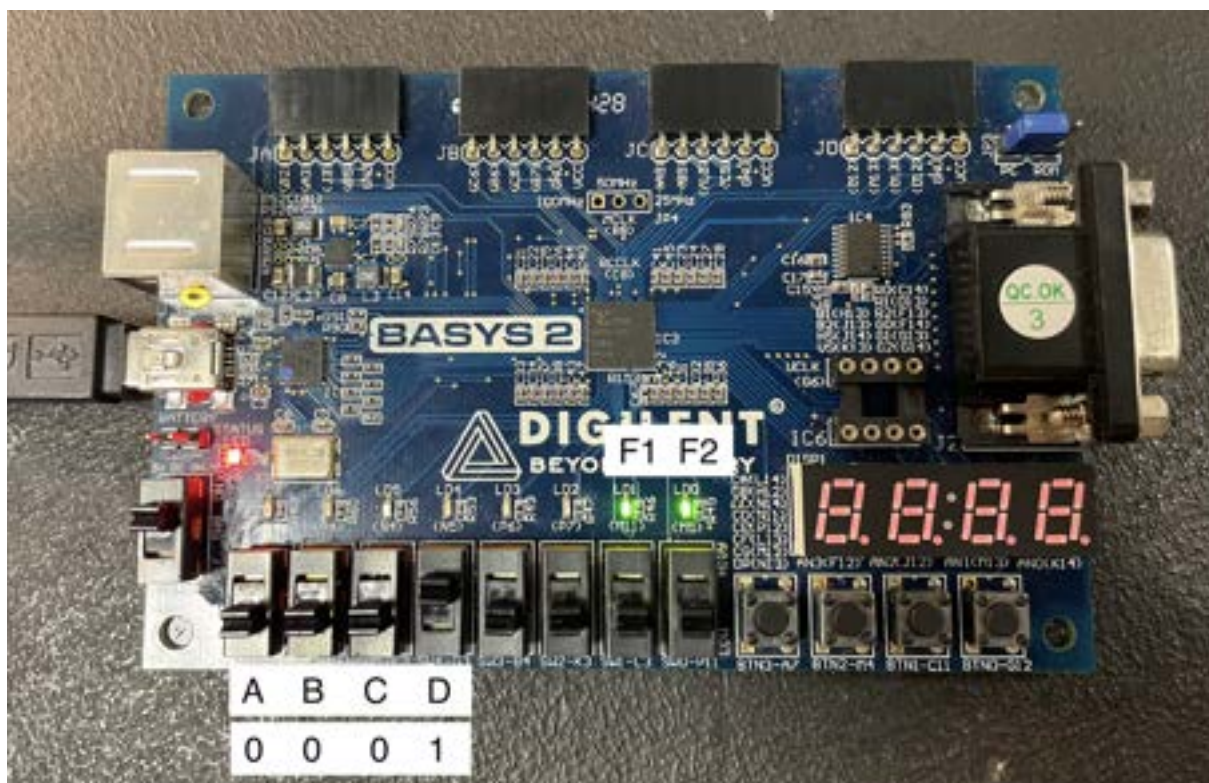


Figure 17: Minterm 1(0001) - expected 1✓

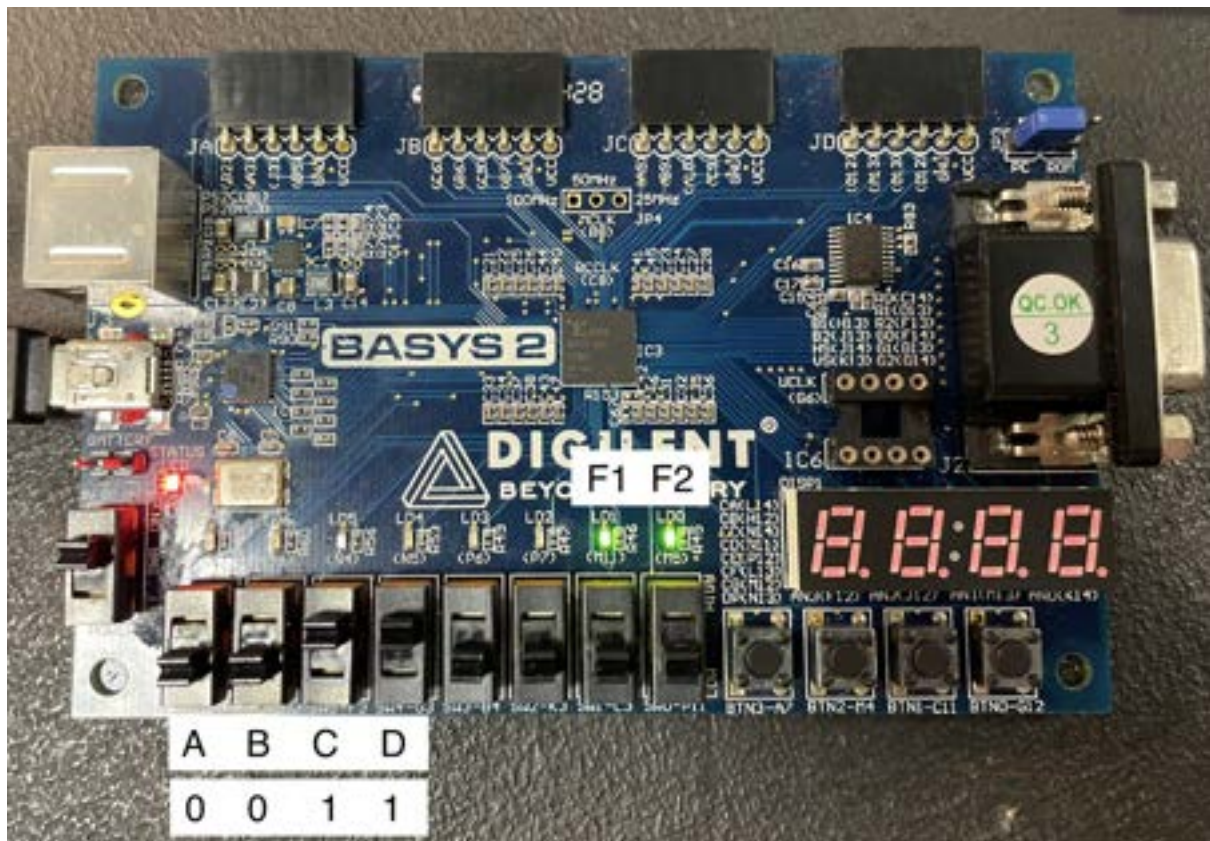


Figure 18: Minterm 3(0011) - expected 1✓

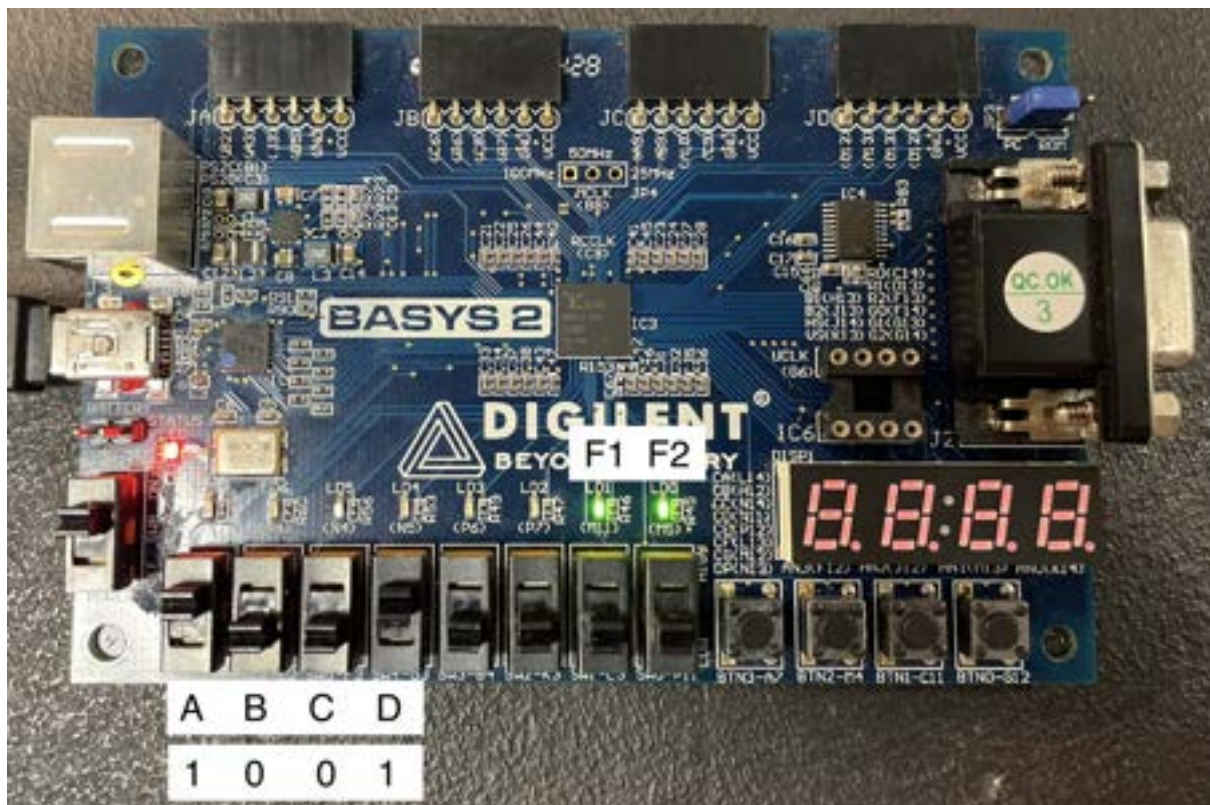


Figure 19: Minterm 9(1001) - expected 1✓

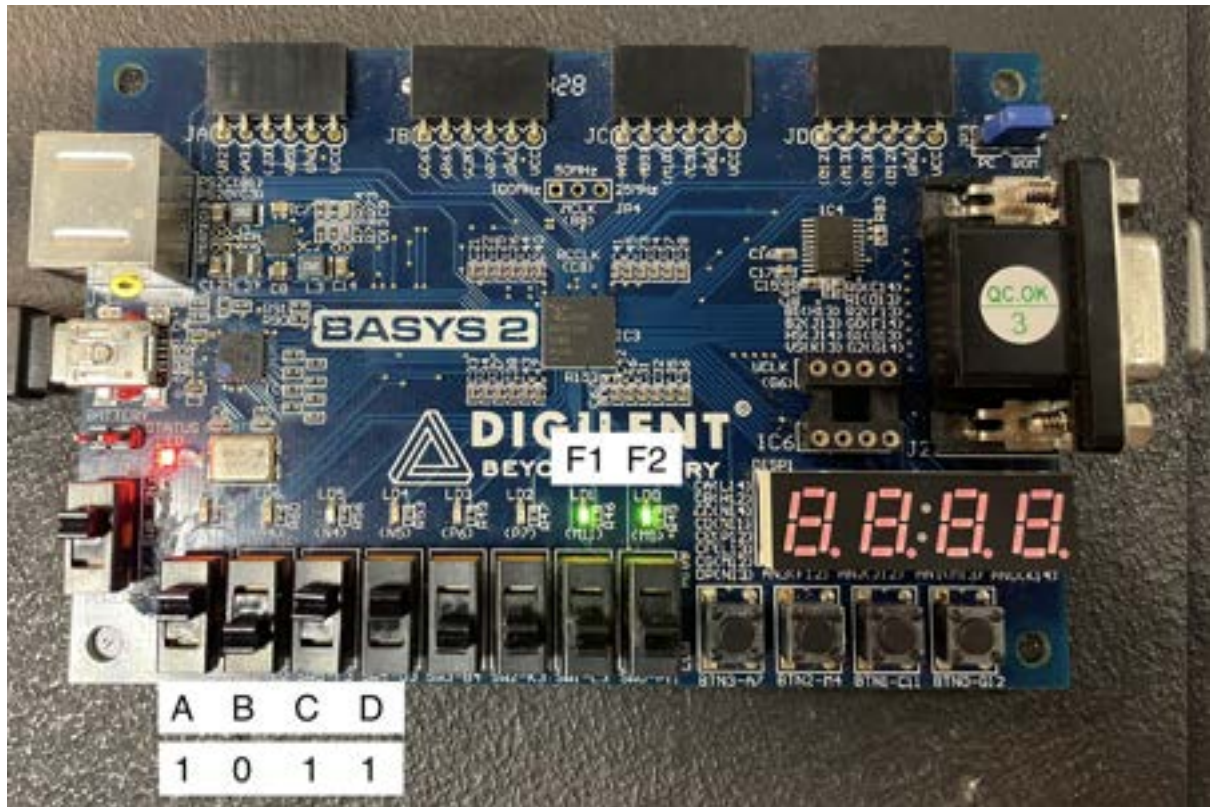


Figure 20: Minterm 11(1011) - expected 1✓

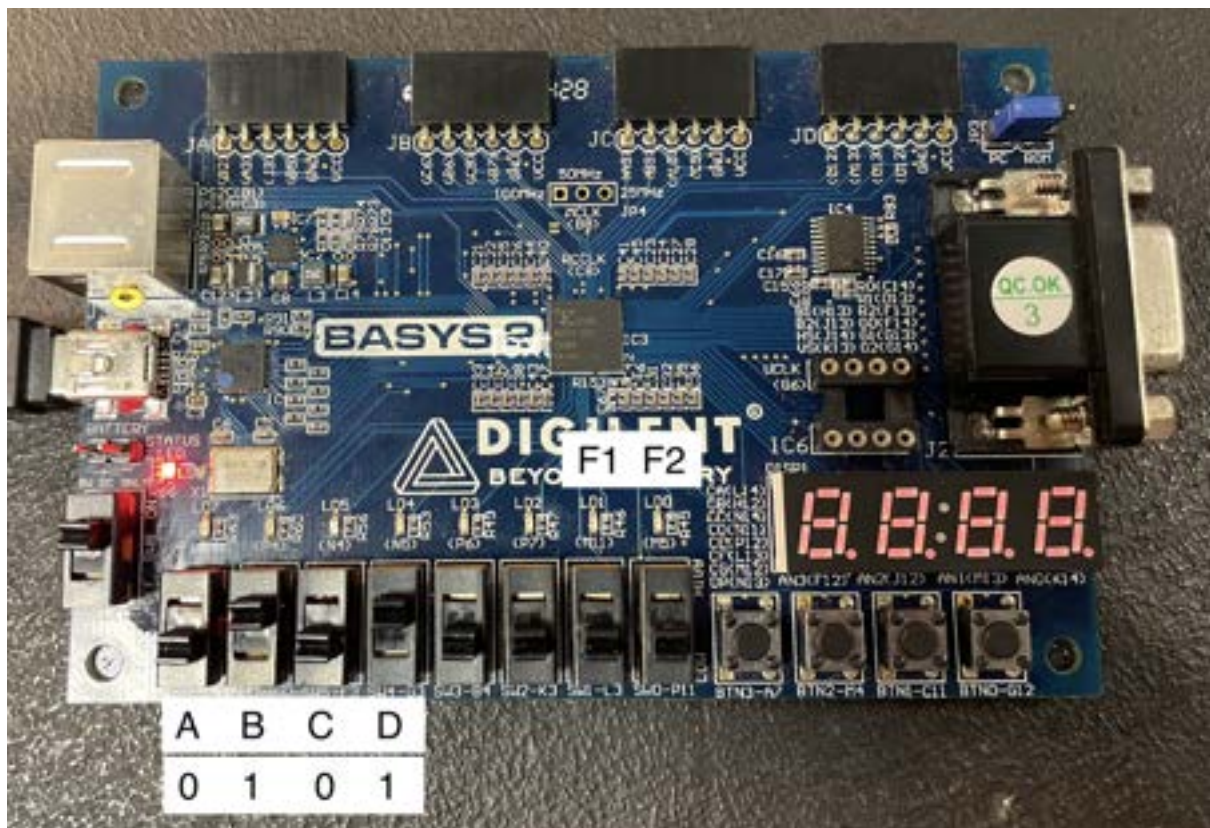


Figure 21: Minterm 5(0101) - expected 0✓

5 Task 5: 8:1 MUX

5.1 VHDL code

Case statements

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity Lab1_5_MUX is
5     Port ( D0 : in  STD_LOGIC;
6           D1 : in  STD_LOGIC;
7           D2 : in  STD_LOGIC;
8           D3 : in  STD_LOGIC;
9           D4 : in  STD_LOGIC;
10          D5 : in  STD_LOGIC;
11          D6 : in  STD_LOGIC;
12          D7 : in  STD_LOGIC;
13          S0 : in  STD_LOGIC;
14          S1 : in  STD_LOGIC;
15          S2 : in  STD_LOGIC;
16          O  : out  STD_LOGIC);
17 end Lab1_5_MUX;
18
19 architecture Behavioral of Lab1_5_MUX is
20 begin
21     -- Multiplexer logic using a process
22     process (D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2)
23     begin
24
25         case STD_LOGIC_VECTOR'(S2 & S1 & S0) is
26             when "000" => O <= D0;
27             when "001" => O <= D1;
28             when "010" => O <= D2;
29             when "011" => O <= D3;
30             when "100" => O <= D4;
31             when "101" => O <= D5;
32             when "110" => O <= D6;
33             when "111" => O <= D7;
34             when others => O <= '0'; -- Default case (should not
35                                     normally happen)
36         end case;
37     end process;
38 end Behavioral;
```

Listing 13: VHDL using case statements for Lab 1_5 MUX

With-Select

```
1 -- Architecture Definition using With-Select
2 architecture Behavioral of Lab1_5_MUX is
3 begin
4     -- Use the combined value of S2, S1, and S0 as the
5     selection signal
6     with STD_LOGIC_VECTOR'(S2 & S1 & S0) select
7         0 <= D0 when "000",
8         D1 when "001",
9         D2 when "010",
10        D3 when "011",
11        D4 when "100",
12        D5 when "101",
13        D6 when "110",
14        D7 when "111",
15        '0' when others;
16 end Behavioral;
```

Listing 14: Testbench for Lab 1.5 MUX

When-Else

```
1 -- Architecture Definition using With-Select
2 architecture Behavioral of Lab1_5_MUX is
3 begin
4     -- Use S2, S1, and S0 to select the appropriate input
5     0 <= D0 when (S2 = '0' and S1 = '0' and S0 = '0') else
6         D1 when (S2 = '0' and S1 = '0' and S0 = '1') else
7         D2 when (S2 = '0' and S1 = '1' and S0 = '0') else
8         D3 when (S2 = '0' and S1 = '1' and S0 = '1') else
9         D4 when (S2 = '1' and S1 = '0' and S0 = '0') else
10        D5 when (S2 = '1' and S1 = '0' and S0 = '1') else
11        D6 when (S2 = '1' and S1 = '1' and S0 = '0') else
12        D7 when (S2 = '1' and S1 = '1' and S0 = '1') else
13        '0';
14 end Behavioral;
```

Listing 15: Testbench for Lab 1.5 MUX

5.2 Theoretical question

VHDL programming difference between Case statements and the With-Select/When-Else statements:

Case statements are sequential, meaning they are evaluated only when there is an event on the signal list. They are placed inside a *process* block. Meanwhile, With-select and When-else are both concurrent statements that are always active within the architecture, and they do not require process blocks. With-select is compact and suitable for direct mapping of select signals to outputs, like multiplexers. Unlike with-select, When-else evaluates conditions sequentially from top to bottom, which can introduce priority encoding for evaluating conditions in order.

5.3 Simulation: VHDL Testbench Code

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;
3
4 -- Uncomment the following library declaration if using
5 -- arithmetic functions with Signed or Unsigned values
6 --USE ieee.numeric_std.ALL;
7
8 ENTITY lab1_5_tb IS
9 END lab1_5_tb;
10
11 ARCHITECTURE behavior OF lab1_5_tb IS
12
13     -- Component Declaration for the Unit Under Test (UUT)
14
15     COMPONENT Lab1_5_MUX
16     PORT(
17         D0 : IN    std_logic;
18         D1 : IN    std_logic;
19         D2 : IN    std_logic;
20         D3 : IN    std_logic;
21         D4 : IN    std_logic;
22         D5 : IN    std_logic;
23         D6 : IN    std_logic;
24         D7 : IN    std_logic;
25         S0 : IN    std_logic;
26         S1 : IN    std_logic;
27         S2 : IN    std_logic;
28         O  : OUT   std_logic
29     );
30     END COMPONENT;
31
32
33     --Inputs
34     signal D0_tb : std_logic := '0';
35     signal D1_tb : std_logic := '0';
36     signal D2_tb : std_logic := '0';
37     signal D3_tb : std_logic := '0';
38     signal D4_tb : std_logic := '0';
39     signal D5_tb : std_logic := '0';
40     signal D6_tb : std_logic := '0';
41     signal D7_tb : std_logic := '0';
42     signal S0_tb : std_logic := '0';
43     signal S1_tb : std_logic := '0';
44     signal S2_tb : std_logic := '0';
45
46     --Outputs
47     signal O_tb : std_logic;
```

```

50 BEGIN
51
52     -- Instantiate the Unit Under Test (UUT)
53     uut: Lab1_5_MUX PORT MAP (
54         D0 => D0_tb,
55         D1 => D1_tb,
56         D2 => D2_tb,
57         D3 => D3_tb,
58         D4 => D4_tb,
59         D5 => D5_tb,
60         D6 => D6_tb,
61         D7 => D7_tb,
62         S0 => S0_tb,
63         S1 => S1_tb,
64         S2 => S2_tb,
65         0 => 0_tb
66     );
67
68
69
70     -- Stimulus process
71     stim_proc: process
72     begin
73         -- hold reset state for 100 ns.
74         wait for 100 ns;
75
76         -- Stimulus 1: Select all 0 (S2=0, S1=0, S0=0)
77         S0_tb <= '0';
78         S1_tb <= '0';
79         S2_tb <= '0';
80         D0_tb <= '0';
81         D1_tb <= '0';
82         D2_tb <= '0';
83         D3_tb <= '0';
84         D4_tb <= '0';
85         D5_tb <= '0';
86         D6_tb <= '0';
87         D7_tb <= '0';
88         wait for 100 ns;
89
90         -- Stimulus 2: Select D0 (S2=0, S1=0, S0=0)
91         S0_tb <= '0';
92         S1_tb <= '0';
93         S2_tb <= '0';
94         D0_tb <= '1'; -- Set D1 to 1, others to 0
95         D1_tb <= '0';
96         D2_tb <= '0';
97         D3_tb <= '0';
98         D4_tb <= '0';
99         D5_tb <= '0';
100        D6_tb <= '0';

```



```

101     D7_tb <= '0';
102     wait for 100 ns;
103
104     -- Stimulus 3: Select D2 (S2=0, S1=1, S0=0)
105     S0_tb <= '0';
106     S1_tb <= '1';
107     S2_tb <= '0';
108     D0_tb <= '0';
109     D1_tb <= '0';
110     D2_tb <= '1';    -- Set D2 to 1, others to 0
111     D3_tb <= '0';
112     D4_tb <= '0';
113     D5_tb <= '0';
114     D6_tb <= '0';
115     D7_tb <= '0';
116     wait for 100 ns;
117
118     -- Stimulus 4: Select D3 (S2=0, S1=1, S0=1)
119     S0_tb <= '1';
120     S1_tb <= '1';
121     S2_tb <= '0';
122     D0_tb <= '0';
123     D1_tb <= '0';
124     D2_tb <= '0';
125     D3_tb <= '1';    -- Set D3 to 1, others to 0
126     D4_tb <= '0';
127     D5_tb <= '0';
128     D6_tb <= '0';
129     D7_tb <= '0';
130     wait for 100 ns;
131
132
133
134     -- End of test
135     wait;
136 end process;
137
138 END;

```

Listing 16: Testbench for Lab 1-5 MUX

Here we can see that the multiplexer is working correctly, giving the correct Data signal based on S0, S1, S2. It works the same on the FPGA board as well.

Table 1: Simulation Inputs and Outputs for 8-to-1 Multiplexer

S2	S1	S0	D0	D1	D2	D3	D4	D5	D6	D7	O
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	0	1	0	0	0	0	0	1
0	1	1	0	0	0	1	0	0	0	0	1



Figure 22: Simulation for 3 test cases for task 5

This table does a good job of visualizing that the multiplexer is working. When select is 0, output is D0, and so on.

5.4 Constraints

```

1 NET "D0" LOC = "P11";
2 NET "D1" LOC = "L3";
3 NET "D2" LOC = "K3";
4 NET "D3" LOC = "B4";
5 NET "D4" LOC = "G3";
6 NET "D5" LOC = "F3";
7 NET "D6" LOC = "E2";
8 NET "D7" LOC = "N3";
9
10 NET "S0" LOC = "G12";
11 NET "S1" LOC = "C11";
12 NET "S2" LOC = "M4";
13
14 NET "O" LOC = "M5";

```

Listing 17: Pin Assignments for Lab 1 MUX

5.5 Test Cases on FPGA Board

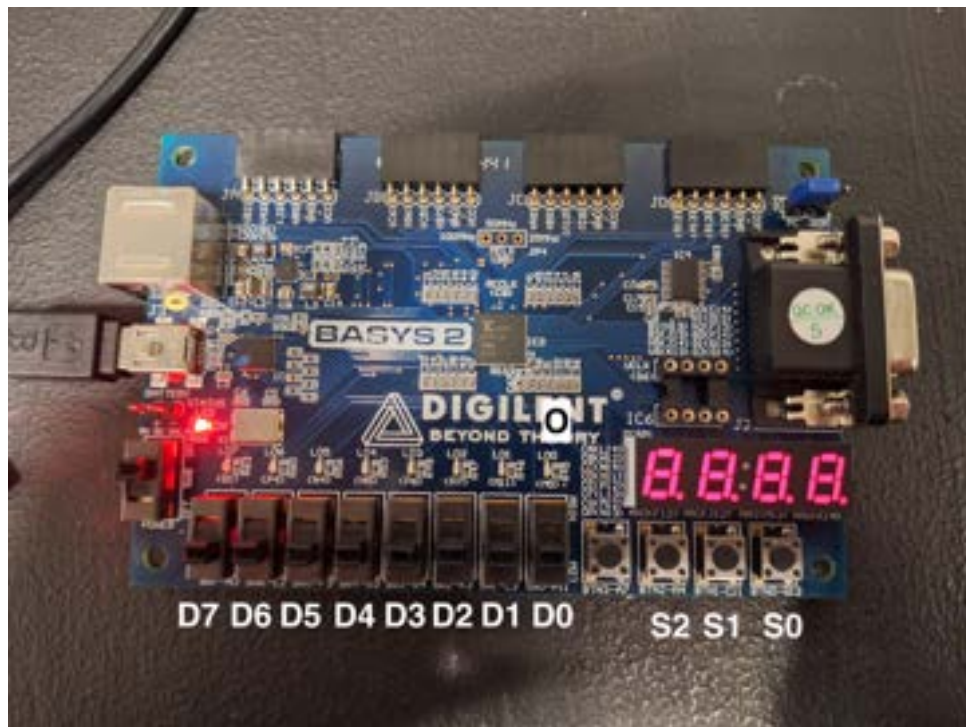


Figure 23: Everything off - expected 0 ✓

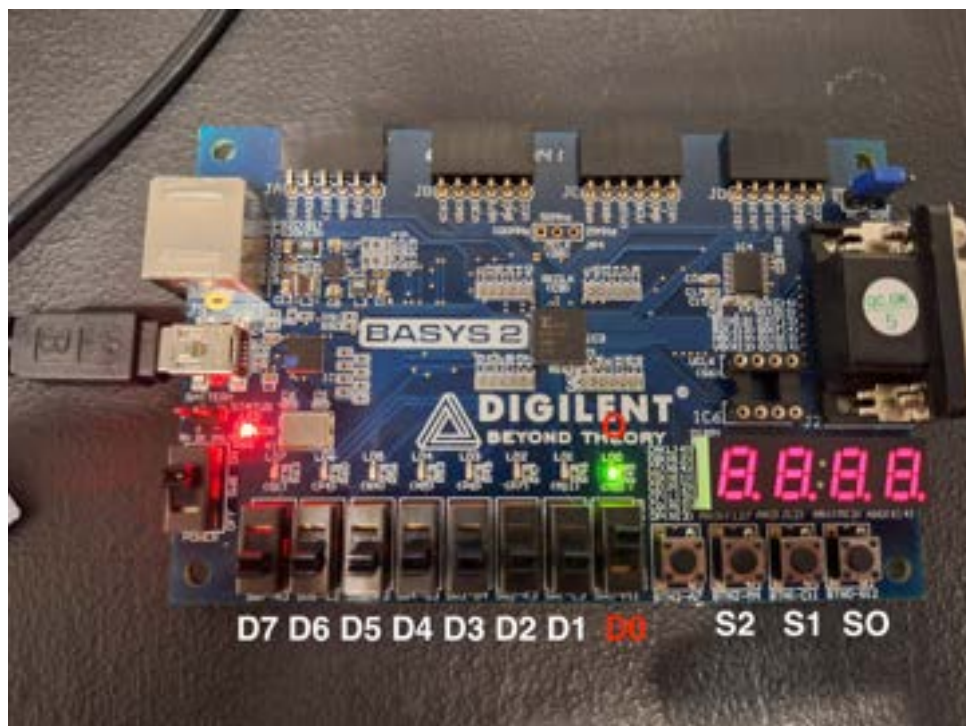


Figure 24: D0 is on - expected 1 ✓

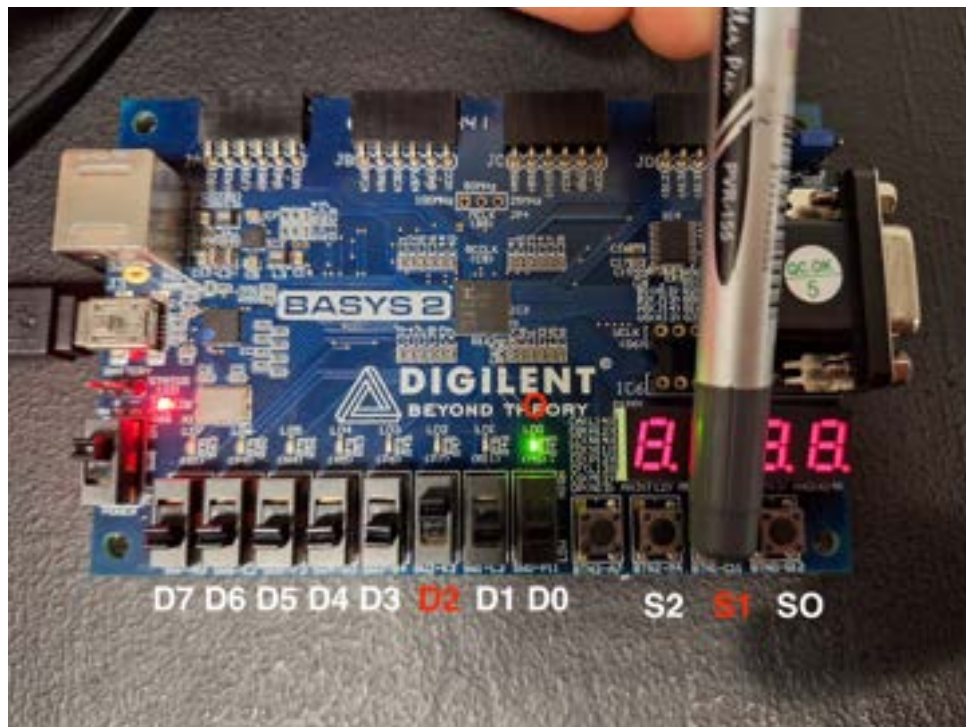


Figure 25: D2 is on, S1 is pressed - expected 1 ✓

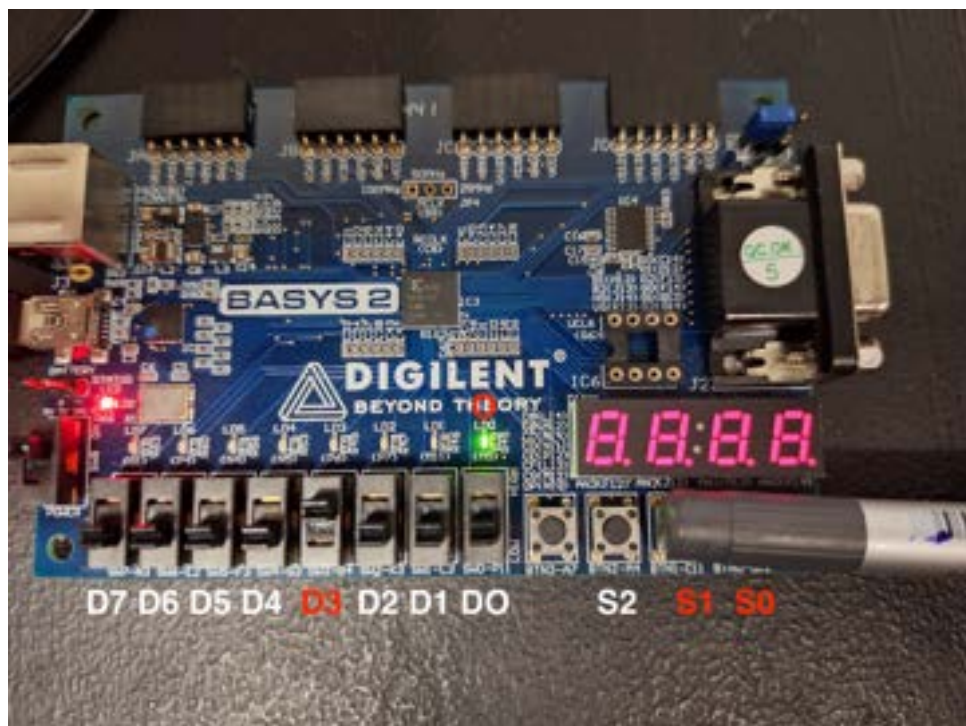


Figure 26: D3 is on, S1 and S0 is pressed - expected 1 ✓