Combinational Logic (Seven Segment Driver)

Christopher Hunt

Objectives

The objective of this lab is to design and implement a decoder on the FPGA that can convert a 4-bit binary number inputted using the switches into a single digit of hexadecimal on the seven-segment display. This will be achieved by creating a block diagram for the decoder design, determining the mapping for displaying each number 0-F on the seven-segment display, generating the functional truth table for the decoder, minimizing the logic for each segment of the decoder using Karnaugh Maps, simulating the design, and finally programming and testing the hardware implementation on the DE10-Lite. Through this lab, we will learn about the process of designing and implementing a decoder, using Karnaugh Maps for logic minimization, and gaining hands-on experience with FPGA programming and testing.

Equipment

- Quartus Prime Lite Edition V. 18.0
- DE10-Lite kit with MAX10 10M50DAF484C7G FPGA
- USB to USB-B cable

Design

Our task is to display the hex digit which corresponds to a 4-bit binary number. First we must identify the segments which are on for each state. In figure 1 we have highlighted the output for each state. These will then be mapped to the corresponding segments to be displayed (fig. 2).

Figure 1: Digits to Hex

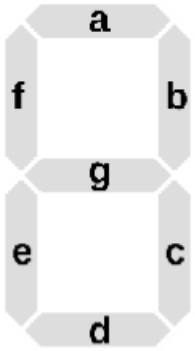

Figure 2: Digits to Hex

The next step in the design process is to map the inputs to the corresponding segement output. To accomplish that a truth table was constructed (Table 1).

Hex	Input	Sa	Sb	Sc	Sd	Se	Sf	Sg
0	0000	0	0	0	0	0	0	1
1	0001	1	0	0	1	1	1	1
2	0010	0	0	1	0	0	1	0
3	0011	0	0	0	0	1	1	0
4	0100	1	0	0	1	1	0	0
5	0101	0	1	0	0	1	0	0
6	0110	0	1	0	0	0	0	0
7	0111	0	0	0	1	1	1	1
8	1000	0	0	0	0	0	0	0
9	1001	0	0	0	0	1	0	0
A	1010	0	0	0	1	0	0	0
b	1011	1	1	0	0	0	0	0
C	1100	0	1	1	0	0	0	1
d	1101	1	0	0	0	0	1	0
E	1110	0	1	1	0	0	0	0
F	1111	0	1	1	1	0	0	0

Table 1: Input Switch to Output PIN Truth Table

Considering that the 7-segment display pins are active LOW, we expect a value of 0 for the segments we want to be turned on during a particular state, while a value of 1 indicates that the segments should be turned off. To accomplish this, we generate Karnaugh maps for each possible output S_a to S_g. These maps allow us to derive the boolean algebra expressions, which form the basis for our logic design (refer to Tables 4 through 8).

$D_3\ D_2$	**00**	**01**	**11**	**10**
$D_1\ D_0$	—	—	—	—
00	0	1	0	0
01	1	0	1	0
11	0	0	0	1
10	0	0	0	0

Table 2: S_a Karnaugh Map

$D_3\ D_2$	**00**	**01**	**11**	**10**
$D_1\ D_0$	—	—	—	—
00	0	0	1	0
01	0	1	0	0
11	0	0	1	1
10	0	1	1	0

Table 3: S_b Karnaugh Map

$$S_a = \bar{D}_3\bar{D}_2\bar{D}_1D_0 + \bar{D}_3D_2\bar{D}_1\bar{D}_0 + D_3D_2\bar{D}_1D_0 + D_3 + \bar{D}_2D_1D_0$$
$$S_b = D_2D_1\bar{D}_0 + D_3D_1D_0 + D_3D_2\bar{D}_0 + \bar{D}_3D_2\bar{D}_1D_0$$

$D_3\ D_2$	**00**	**01**	**11**	**10**
$D_1\ D_0$	—	—	—	—
00	0	0	1	0
01	0	0	0	0
11	0	0	1	0
10	1	0	1	0

Table 4: S_c Karnaugh Map

$D_3\ D_2$	**00**	**01**	**11**	**10**
$D_1\ D_0$	—	—	—	—
00	0	1	0	0
01	1	0	0	0
11	0	1	1	0
10	0	0	0	1

Table 5: S_d Karnaugh Map

$$S_c = D_3D_2\bar{D}_0 + D_3D_2D_1 + \bar{D}_3\bar{D}_2D_1\bar{D}_0$$
$$S_d = D_2D_1D_0 + \bar{D}_3\bar{D}_2\bar{D}_1D_0 + \bar{D}_3D_2\bar{D}_1\bar{D}_0 + D_3\bar{D}_2D_1\bar{D}_0$$

$D_3\ D_2$	**00**	**01**	**11**	**10**
$D_1\ D_0$	—	—	—	—
00	0	1	0	0
01	1	1	0	1
11	1	1	0	0
10	0	0	0	0

Table 6: S_e Karnaugh Map

$D_3\ D_2$	**00**	**01**	**11**	**10**
$D_1\ D_0$	—	—	—	—
00	0	0	0	0
01	1	0	1	0
11	1	1	0	0
10	1	0	0	0

Table 7: S_f Karnaugh Map

$$S_e = \bar{D}_2\bar{D}_1D_0 + \bar{D}_3D_2\bar{D}_1 + \bar{D}_3D_2D_0 + \bar{D}_3\bar{D}_2D_0$$
$$S_f = \bar{D}_3\bar{D}_2D_0 + \bar{D}_3\bar{D}_2D_1 + \bar{D}_3D_1D_0 + D_3D_2\bar{D}_1D_0$$

$D_3\ D_2$	**00**	**01**	**11**	**10**
$D_1\ D_0$	—	—	—	—
00	1	0	1	0
01	1	0	0	0
11	0	1	0	0
10	0	0	0	0

Table 8: S_g Karnaugh Map

$$S_g = \bar{D}_3\bar{D}_2\bar{D}_1 + D_3D_2\bar{D}_1\bar{D}_0 + \bar{D}_3D_2D_1D_0$$

Design Entry

Once the boolean logic equations are derived, we utilize Quartus Prime to construct the schematics for each individual logic design (refer to figures 3 through 9). These schematics are then exported as symbol files and

serve as logic blocks for the complete binary-to-hex digital display design (refer to figure 10). Subsequently, we proceed to FPGA pin placement, where the binary inputs are mapped to SW0 to SW3, and the display segments are mapped to their corresponding segments a through g (as illustrated in Table 9 and Table 10).

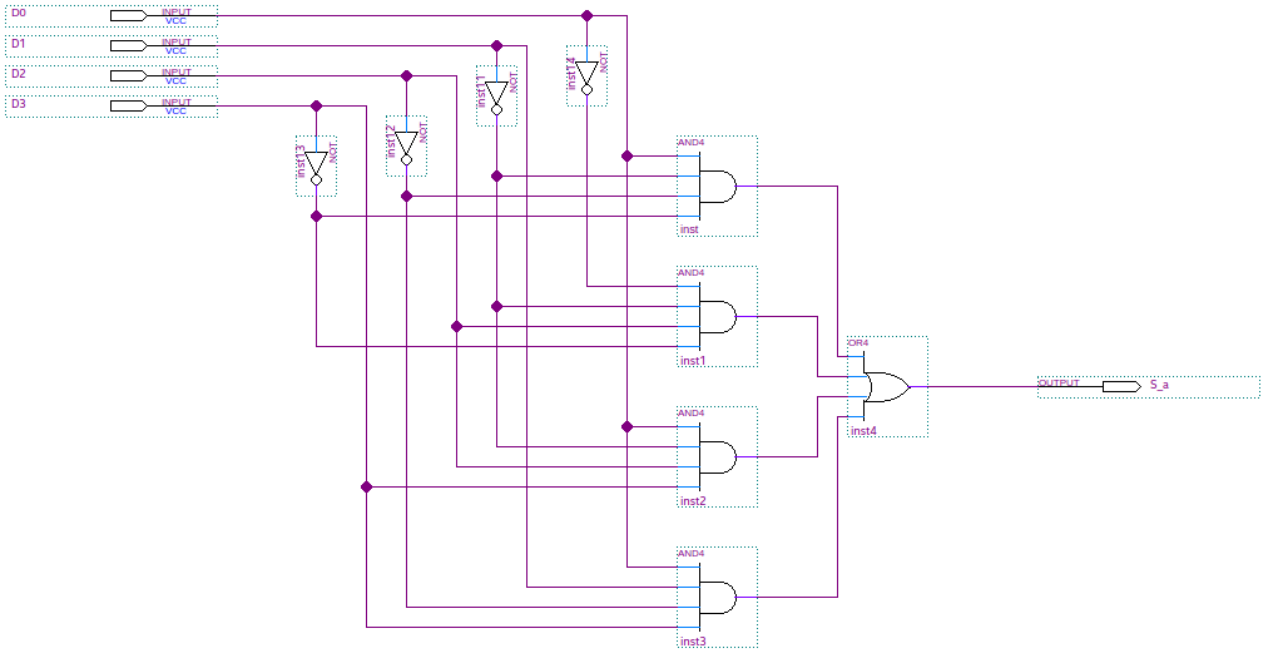

Figure 3: S_a Digital Logic Schematic

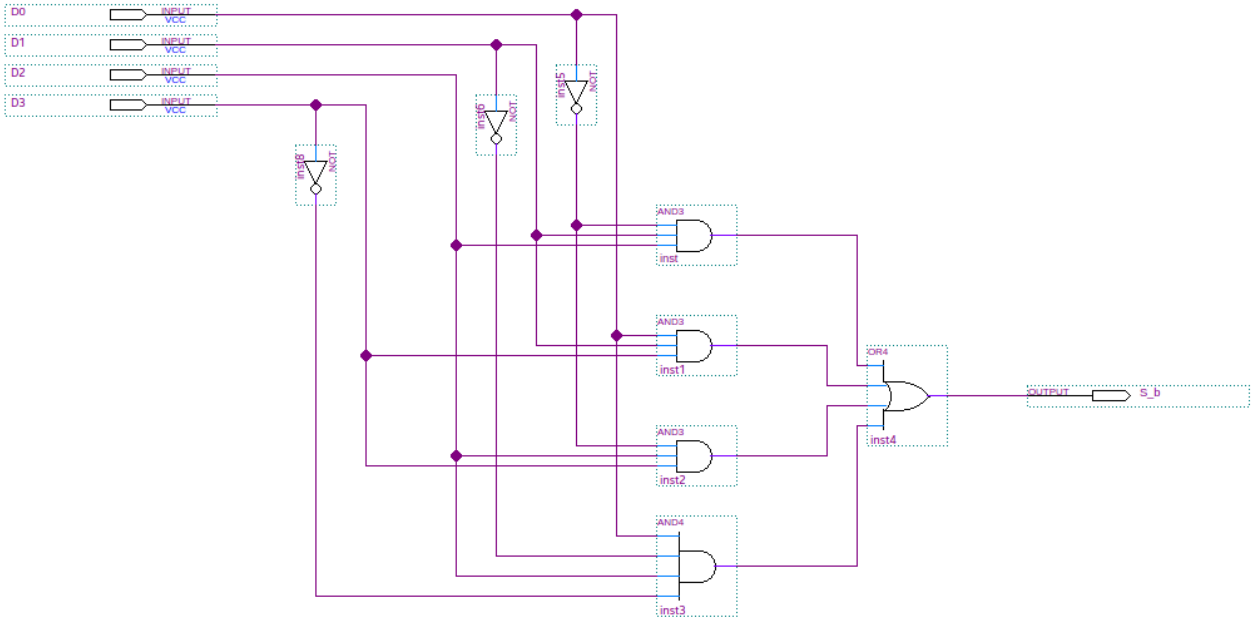

Figure 4: S_b Digital Logic Schematic

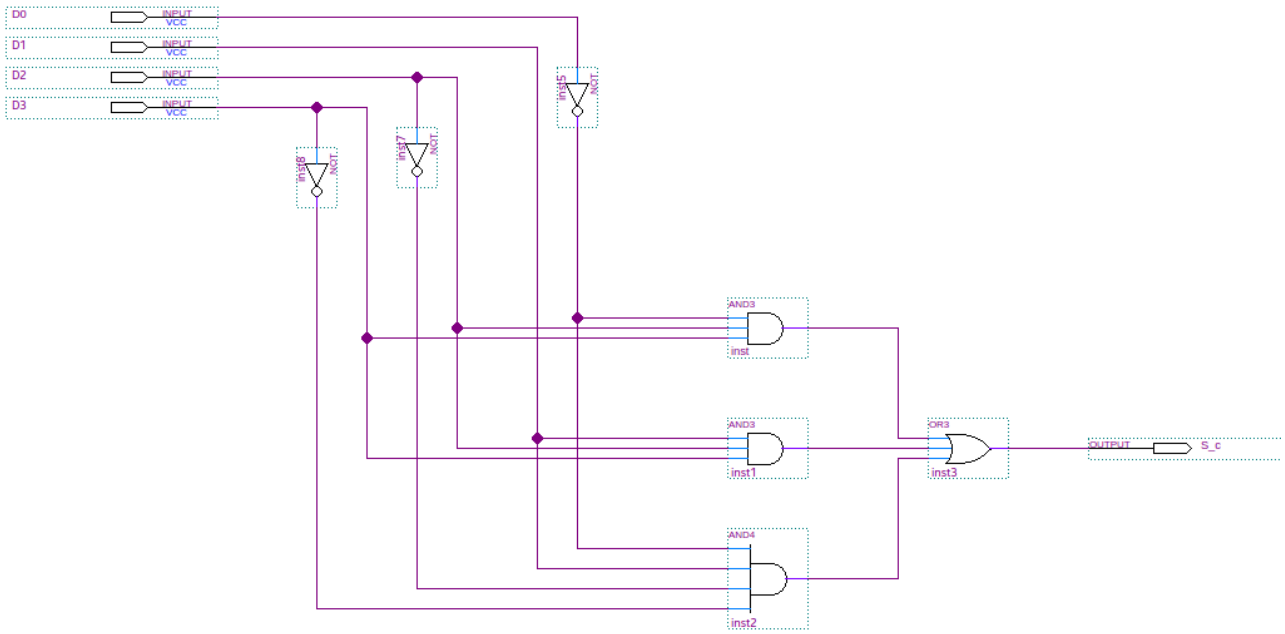

Figure 5: S_c Digital Logic Schematic

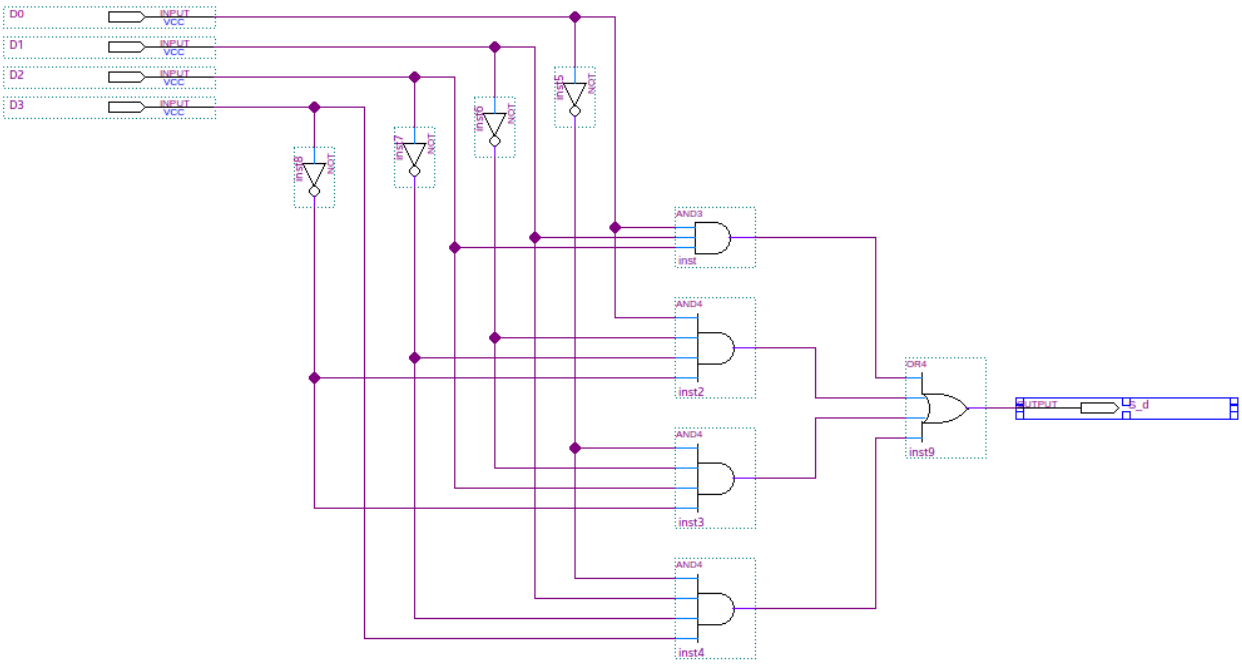

Figure 6: S_e Digital Logic Schematic

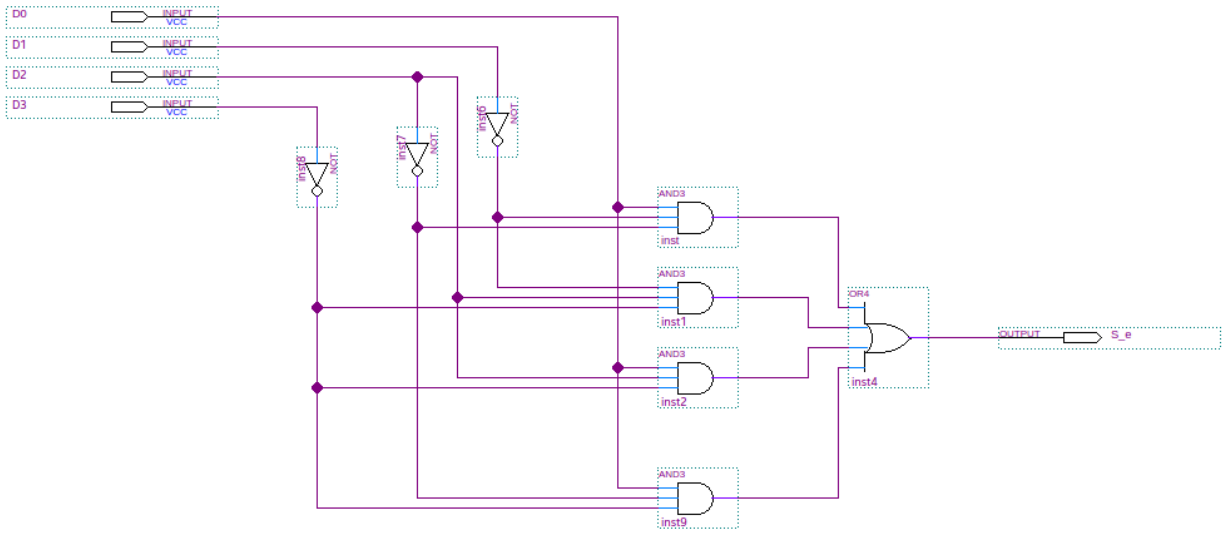

Figure 7: S_f Digital Logic Schematic

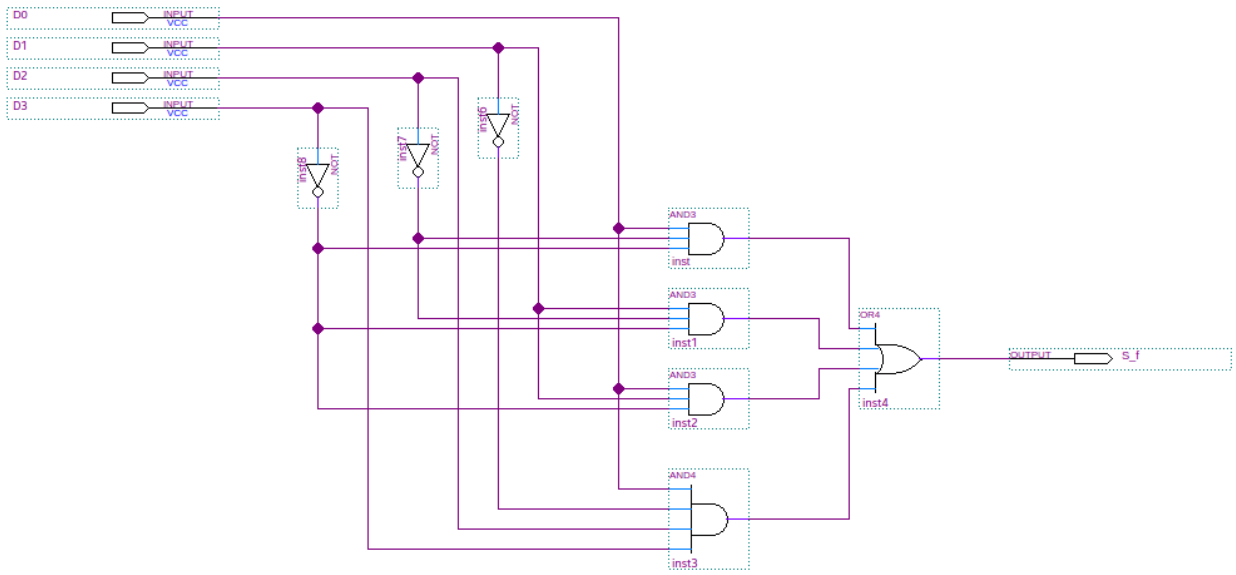

Figure 8: S_g Digital Logic Schematic

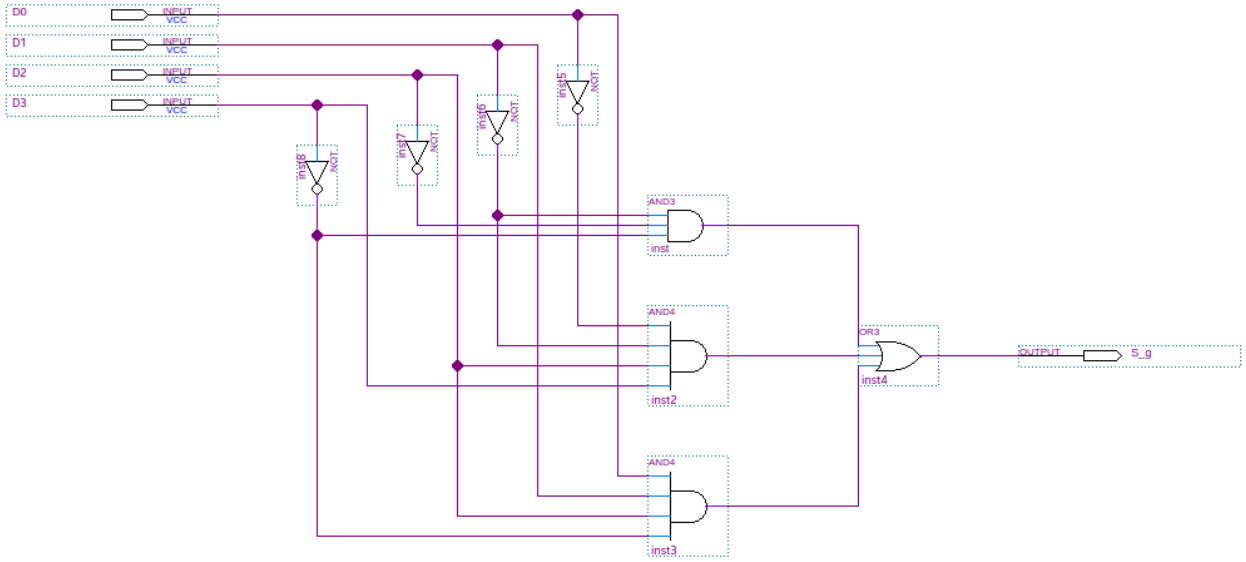

Figure 9: S_g Digital Logic Schematic

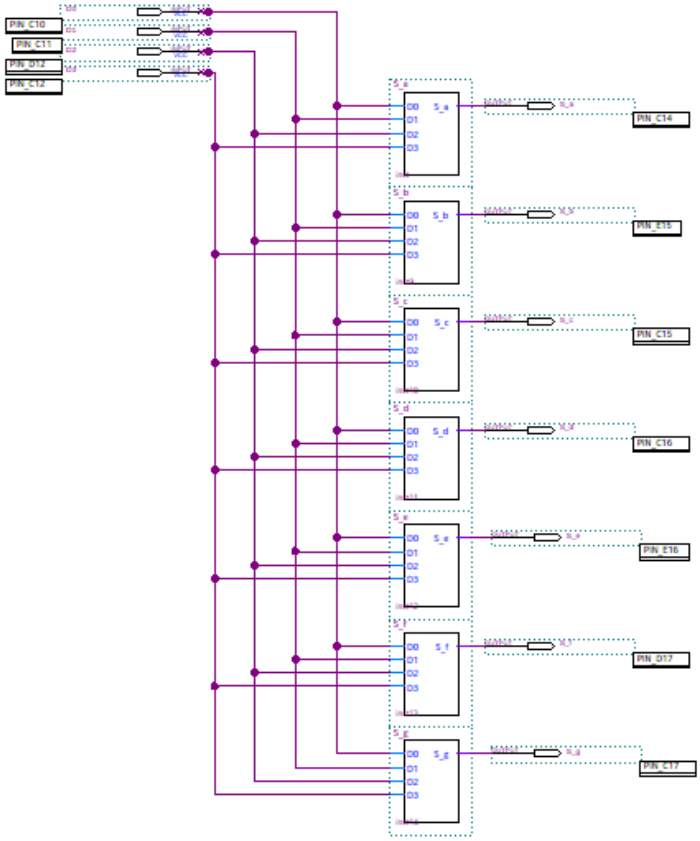

Figure 10: Full Binary To Hex Display Digital Logic Schematic

Input	FPGA PIN
D0	PIN_C10
D1	PIN_C11
D2	PIN_D12
D3	PIN_C12

Table 9: Input to FPGA PIN Mapping

Output	FPGA PIN
Sa	PIN_C14
Sb	PIN_E15
Sc	PIN_C15
Sd	PIN_C16
Se	PIN_E16
Sf	PIN_D17
Sg	PIN_C17

Table 10: Output to FPGA PIN Mapping

Design Simulation

These schematics were then exported as Verilog files (see Appendix). Before exporting the code to the DE10-Lite, the design was tested using ModelSim (fig. 11). The simulated output matched the expected trut table, Table 1.

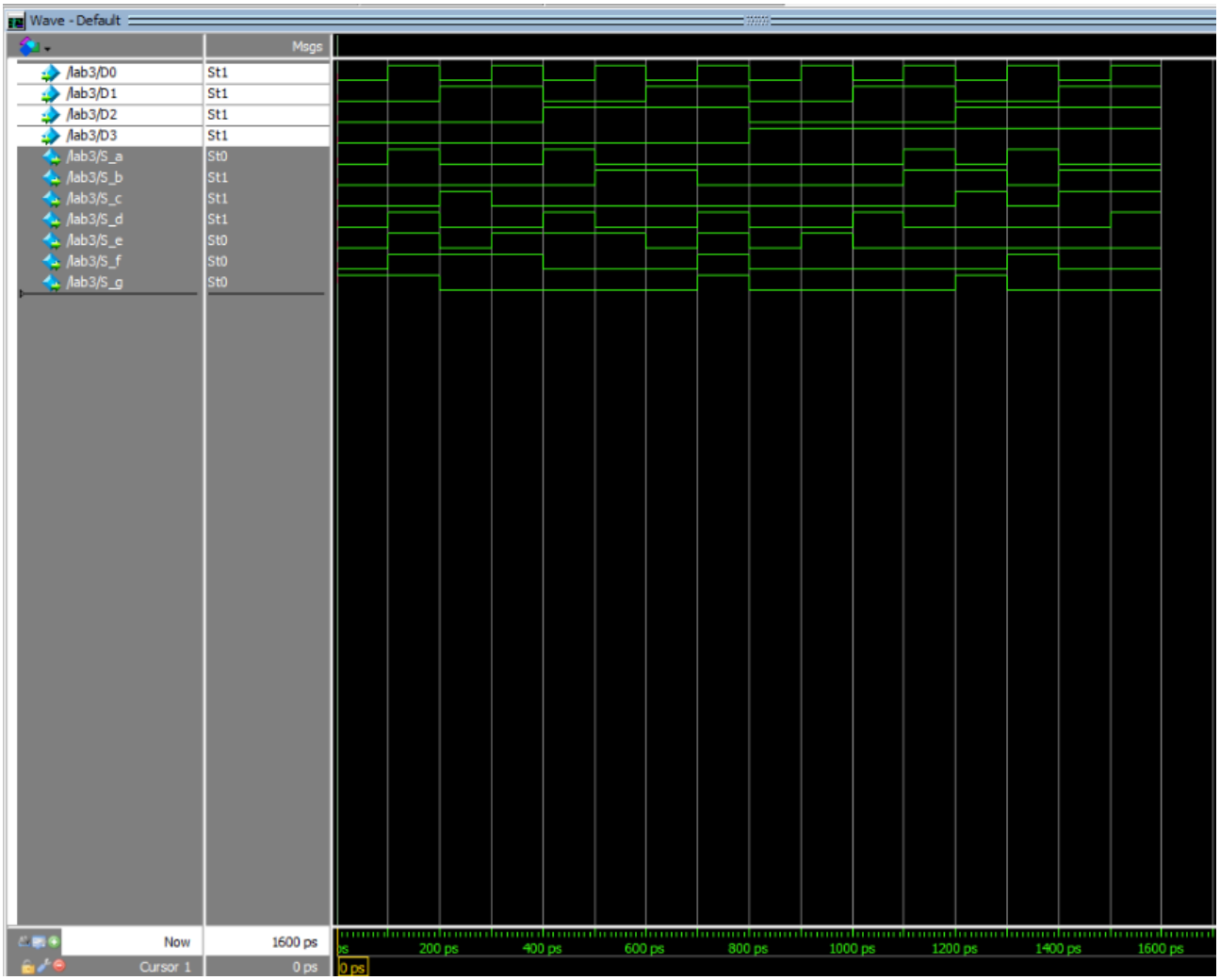

Figure 11: Full Binary To Hex Display Digital Logic Schematic

Design Implementation

Upon completing the design simulation without any errors the HDL files were implemented into the DE10-Lite.

Observations

During the lab execution, an error occurred due to an incorrect placement of a value in my original Karnaugh Map for S_e. Consequently, there was a flaw in my boolean logic, resulting in the incorrect display of digits 2 and 3. Upon identifying this issue, we isolated the affected segment and digits and traced the error back to the Karnaugh Map. Once the error was rectified, the output for each hex digit from 0 to F became correct.

Conclusion

In conclusion, the lab focused on the design and implementation of a decoder for converting a 4-bit binary number to a single digit of hexadecimal on a seven-segment display. Throughout the lab, various steps were undertaken to achieve this objective, including creating a block diagram, determining the display mapping, generating a functional truth table, minimizing the logic using Karnaugh Maps, simulating the design, and programming and testing the hardware on the FPGA.

By successfully completing the lab, we gained valuable knowledge and practical experience in designing digital logic circuits, using Karnaugh Maps for logic minimization, and programming FPGAs. We also developed an understanding of the relationship between binary numbers and their corresponding hexadecimal representation on a seven-segment display. This lab provided a hands-on opportunity to apply theoretical concepts and enhanced our skills in digital logic design and FPGA programming.

Overall, the lab helped deepen our understanding of combinational logic and its practical application in converting binary numbers to hexadecimal displays. It also highlighted the importance of careful design considerations, simulation, and hardware testing to ensure the desired functionality and accuracy of the implemented circuit.

Study Questions

1. When is a simulation necessary? Was it useful for this section?

Simulation was necessary for this lab. The error found in my Karnaugh Map for segment S_e would possibly not been caught if the simulation was not done. In the case for this lab the desired output was directly linked to the input with no unseen events occuring, this would have made the error easily caught if implemented into the hardware. In a scenario where there was more complex logic occur on the hardware but was not so easily witnessed, the error could have slipped undetected. Although simulation is an extra step that may seem extrenuous, it is still a crucial component to the design process of FPGA devices.

Appendix

```
// Copyright (C) 2020  Intel Corporation. All rights reserved.
// Your use of Intel Corporation's design tools, logic functions
// and other software and tools, and any partner logic
// functions, and any output files from any of the foregoing
// (including device programming or simulation files), and any
// associated documentation or information are expressly subject
// to the terms and conditions of the Intel Program License
// Subscription Agreement, the Intel Quartus Prime License Agreement,
// the Intel FPGA IP License Agreement, or other applicable license
// agreement, including, without limitation, that your use is for
// the sole purpose of programming logic devices manufactured by
// Intel and sold by Intel or its authorized distributors.  Please
// refer to the applicable agreement for further details, at
// https://fpgasoftware.intel.com/eula.

// PROGRAM            "Quartus Prime"
// VERSION            "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
// CREATED            "Fri Apr 28 17:47:04 2023"
// By Christopher Hunt
// lab3.v
module lab3(
  D0,
  D1,
  D2,
  D3,
  S_a,
  S_b,
  S_c,
  S_d,
  S_e,
  S_f,
  S_g
);


input wire    D0;
input wire    D1;
input wire    D2;
input wire    D3;
output wire   S_a;
output wire   S_b;
output wire   S_c;
output wire   S_d;
output wire   S_e;
output wire   S_f;
output wire   S_g;









S_a   b2v_inst(
  .D0(D0),
  .D1(D1),
  .D2(D2),
  .D3(D3),
  .S_a(S_a));
```

```
59
60
61    S_c   b2v_inst10(
62      .D0(D0),
63      .D1(D1),
64      .D2(D2),
65      .D3(D3),
66      .S_c(S_c));
67
68
69    S_d   b2v_inst11(
70      .D0(D0),
71      .D1(D1),
72      .D2(D2),
73      .D3(D3),
74      .S_d(S_d));
75
76
77    S_e   b2v_inst12(
78      .D0(D0),
79      .D1(D1),
80      .D2(D2),
81      .D3(D3),
82      .S_e(S_e));
83
84
85    S_f   b2v_inst13(
86      .D0(D0),
87      .D1(D1),
88      .D2(D2),
89      .D3(D3),
90      .S_f(S_f));
91
92
93    S_g   b2v_inst14(
94      .D0(D0),
95      .D1(D1),
96      .D2(D2),
97      .D3(D3),
98      .S_g(S_g));
99
100
101   S_b   b2v_inst9(
102     .D0(D0),
103     .D1(D1),
104     .D2(D2),
105     .D3(D3),
106     .S_b(S_b));
107
108
109   endmodule
```

```
1  // Copyright (C) 2020  Intel Corporation. All rights reserved.
2  // Your use of Intel Corporation's design tools, logic functions
3  // and other software and tools, and any partner logic
4  // functions, and any output files from any of the foregoing
5  // (including device programming or simulation files), and any
6  // associated documentation or information are expressly subject
7  // to the terms and conditions of the Intel Program License
8  // Subscription Agreement, the Intel Quartus Prime License Agreement,
```

```
9  // the Intel FPGA IP License Agreement, or other applicable license
10 // agreement, including, without limitation, that your use is for
11 // the sole purpose of programming logic devices manufactured by
12 // Intel and sold by Intel or its authorized distributors.  Please
13 // refer to the applicable agreement for further details, at
14 // https://fpgasoftware.intel.com/eula.
15
16 // PROGRAM              "Quartus Prime"
17 // VERSION              "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
18 // CREATED              "Fri Apr 28 17:48:03 2023"
19 // By Christopher Hunt
20 // S_a.v
21
22 module S_a(
23         D0,
24         D1,
25         D2,
26         D3,
27         S_a
28 );
29
30
31 input wire      D0;
32 input wire      D1;
33 input wire      D2;
34 input wire      D3;
35 output wire     S_a;
36
37 wire    SYNTHESIZED_WIRE_12;
38 wire    SYNTHESIZED_WIRE_13;
39 wire    SYNTHESIZED_WIRE_14;
40 wire    SYNTHESIZED_WIRE_3;
41 wire    SYNTHESIZED_WIRE_8;
42 wire    SYNTHESIZED_WIRE_9;
43 wire    SYNTHESIZED_WIRE_10;
44 wire    SYNTHESIZED_WIRE_11;
45
46
47
48
49 assign  SYNTHESIZED_WIRE_8 = D0 & SYNTHESIZED_WIRE_12 & SYNTHESIZED_WIRE_13 &
       SYNTHESIZED_WIRE_14;
50
51 assign  SYNTHESIZED_WIRE_11 = SYNTHESIZED_WIRE_3 & SYNTHESIZED_WIRE_12 & D2 &
       SYNTHESIZED_WIRE_14;
52
53 assign  SYNTHESIZED_WIRE_12 =  ~D1;
54
55 assign  SYNTHESIZED_WIRE_13 =  ~D2;
56
57 assign  SYNTHESIZED_WIRE_14 =  ~D3;
58
59 assign  SYNTHESIZED_WIRE_3 =  ~D0;
60
61 assign  SYNTHESIZED_WIRE_9 = D0 & SYNTHESIZED_WIRE_12 & D2 & D3;
62
63 assign  SYNTHESIZED_WIRE_10 = D0 & D1 & SYNTHESIZED_WIRE_13 & D3;
64
65 assign  S_a = SYNTHESIZED_WIRE_8 | SYNTHESIZED_WIRE_9 | SYNTHESIZED_WIRE_10 |
       SYNTHESIZED_WIRE_11;
```

```
66
67
68 endmodule
```

```
1  // Copyright (C) 2020  Intel Corporation. All rights reserved.
2  // Your use of Intel Corporation's design tools, logic functions
3  // and other software and tools, and any partner logic
4  // functions, and any output files from any of the foregoing
5  // (including device programming or simulation files), and any
6  // associated documentation or information are expressly subject
7  // to the terms and conditions of the Intel Program License
8  // Subscription Agreement, the Intel Quartus Prime License Agreement,
9  // the Intel FPGA IP License Agreement, or other applicable license
10 // agreement, including, without limitation, that your use is for
11 // the sole purpose of programming logic devices manufactured by
12 // Intel and sold by Intel or its authorized distributors.  Please
13 // refer to the applicable agreement for further details, at
14 // https://fpgasoftware.intel.com/eula.
15
16 // PROGRAM            "Quartus Prime"
17 // VERSION            "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
18 // CREATED            "Fri Apr 28 17:47:59 2023"
19
20 module S_b(
21   D0,
22   D1,
23   D2,
24   D3,
25   S_b
26 );
27
28
29 input wire    D0;
30 input wire    D1;
31 input wire    D2;
32 input wire    D3;
33 output wire   S_b;
34
35 wire  SYNTHESIZED_WIRE_8;
36 wire  SYNTHESIZED_WIRE_2;
37 wire  SYNTHESIZED_WIRE_3;
38 wire  SYNTHESIZED_WIRE_4;
39 wire  SYNTHESIZED_WIRE_5;
40 wire  SYNTHESIZED_WIRE_6;
41 wire  SYNTHESIZED_WIRE_7;
42
43
44
45
46 assign        SYNTHESIZED_WIRE_4 = SYNTHESIZED_WIRE_8 & D1 & D2;
47
48 assign        SYNTHESIZED_WIRE_7 = D0 & D1 & D3;
49
50 assign        SYNTHESIZED_WIRE_5 = SYNTHESIZED_WIRE_8 & D2 & D3;
51
52 assign        SYNTHESIZED_WIRE_6 = D0 & SYNTHESIZED_WIRE_2 & D2 &
       SYNTHESIZED_WIRE_3;
53
54 assign        S_b = SYNTHESIZED_WIRE_4 | SYNTHESIZED_WIRE_5 | SYNTHESIZED_WIRE_6 |
       SYNTHESIZED_WIRE_7;
```

```
55
56  assign        SYNTHESIZED_WIRE_8 =  ~D0;
57
58  assign        SYNTHESIZED_WIRE_2 =  ~D1;
59
60  assign        SYNTHESIZED_WIRE_3 =  ~D3;
61
62
63  endmodule
```

```
1   // Copyright (C) 2020  Intel Corporation. All rights reserved.
2   // Your use of Intel Corporation's design tools, logic functions
3   // and other software and tools, and any partner logic
4   // functions, and any output files from any of the foregoing
5   // (including device programming or simulation files), and any
6   // associated documentation or information are expressly subject
7   // to the terms and conditions of the Intel Program License
8   // Subscription Agreement, the Intel Quartus Prime License Agreement,
9   // the Intel FPGA IP License Agreement, or other applicable license
10  // agreement, including, without limitation, that your use is for
11  // the sole purpose of programming logic devices manufactured by
12  // Intel and sold by Intel or its authorized distributors.  Please
13  // refer to the applicable agreement for further details, at
14  // https://fpgasoftware.intel.com/eula.
15
16  // PROGRAM            "Quartus Prime"
17  // VERSION            "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
18  // CREATED            "Fri Apr 28 17:47:54 2023"
19
20  module S_c(
21    D0,
22    D1,
23    D2,
24    D3,
25    S_c
26  );
27
28
29  input wire    D0;
30  input wire    D1;
31  input wire    D2;
32  input wire    D3;
33  output wire   S_c;
34
35  wire  SYNTHESIZED_WIRE_7;
36  wire  SYNTHESIZED_WIRE_2;
37  wire  SYNTHESIZED_WIRE_3;
38  wire  SYNTHESIZED_WIRE_4;
39  wire  SYNTHESIZED_WIRE_5;
40  wire  SYNTHESIZED_WIRE_6;
41
42
43
44
45  assign        SYNTHESIZED_WIRE_6 = SYNTHESIZED_WIRE_7 & D2 & D3;
46
47  assign        SYNTHESIZED_WIRE_4 = D1 & D2 & D3;
48
49  assign        SYNTHESIZED_WIRE_5 = SYNTHESIZED_WIRE_7 & D1 & SYNTHESIZED_WIRE_2 &
      SYNTHESIZED_WIRE_3;
```

```
50
51  assign        S_c = SYNTHESIZED_WIRE_4 | SYNTHESIZED_WIRE_5 | SYNTHESIZED_WIRE_6;
52
53  assign        SYNTHESIZED_WIRE_7 =  ~D0;
54
55  assign        SYNTHESIZED_WIRE_2 =  ~D2;
56
57  assign        SYNTHESIZED_WIRE_3 =  ~D3;
58
59
60  endmodule
```

```
1   // Copyright (C) 2020  Intel Corporation. All rights reserved.
2   // Your use of Intel Corporation's design tools, logic functions
3   // and other software and tools, and any partner logic
4   // functions, and any output files from any of the foregoing
5   // (including device programming or simulation files), and any
6   // associated documentation or information are expressly subject
7   // to the terms and conditions of the Intel Program License
8   // Subscription Agreement, the Intel Quartus Prime License Agreement,
9   // the Intel FPGA IP License Agreement, or other applicable license
10  // agreement, including, without limitation, that your use is for
11  // the sole purpose of programming logic devices manufactured by
12  // Intel and sold by Intel or its authorized distributors.  Please
13  // refer to the applicable agreement for further details, at
14  // https://fpgasoftware.intel.com/eula.
15
16  // PROGRAM            "Quartus Prime"
17  // VERSION            "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
18  // CREATED            "Fri Apr 28 17:47:48 2023"
19
20  module S_d(
21    D0,
22    D1,
23    D2,
24    D3,
25    S_d
26  );
27
28
29  input wire    D0;
30  input wire    D1;
31  input wire    D2;
32  input wire    D3;
33  output wire   S_d;
34
35  wire  SYNTHESIZED_WIRE_12;
36  wire  SYNTHESIZED_WIRE_13;
37  wire  SYNTHESIZED_WIRE_14;
38  wire  SYNTHESIZED_WIRE_15;
39  wire  SYNTHESIZED_WIRE_8;
40  wire  SYNTHESIZED_WIRE_9;
41  wire  SYNTHESIZED_WIRE_10;
42  wire  SYNTHESIZED_WIRE_11;
43
44
45
46
47  assign        SYNTHESIZED_WIRE_8 = D0 & D1 & D2;
48
```

```
49    assign        SYNTHESIZED_WIRE_11 = D0 & SYNTHESIZED_WIRE_12 & SYNTHESIZED_WIRE_13
          & SYNTHESIZED_WIRE_14;
50
51    assign        SYNTHESIZED_WIRE_9 = SYNTHESIZED_WIRE_15 & SYNTHESIZED_WIRE_12 & D2 &
           SYNTHESIZED_WIRE_14;
52
53    assign        SYNTHESIZED_WIRE_10 = SYNTHESIZED_WIRE_15 & D1 & SYNTHESIZED_WIRE_13
          & D3;
54
55    assign        SYNTHESIZED_WIRE_15 =  ~D0;
56
57    assign        SYNTHESIZED_WIRE_12 =  ~D1;
58
59    assign        SYNTHESIZED_WIRE_13 =  ~D2;
60
61    assign        SYNTHESIZED_WIRE_14 =  ~D3;
62
63    assign        S_d = SYNTHESIZED_WIRE_8 | SYNTHESIZED_WIRE_9 | SYNTHESIZED_WIRE_10 |
           SYNTHESIZED_WIRE_11;
64
65
66    endmodule
```

```
1     // Copyright (C) 2020  Intel Corporation. All rights reserved.
2   // Your use of Intel Corporation's design tools, logic functions
3   // and other software and tools, and any partner logic
4   // functions, and any output files from any of the foregoing
5   // (including device programming or simulation files), and any
6   // associated documentation or information are expressly subject
7   // to the terms and conditions of the Intel Program License
8   // Subscription Agreement, the Intel Quartus Prime License Agreement,
9   // the Intel FPGA IP License Agreement, or other applicable license
10  // agreement, including, without limitation, that your use is for
11  // the sole purpose of programming logic devices manufactured by
12  // Intel and sold by Intel or its authorized distributors.  Please
13  // refer to the applicable agreement for further details, at
14  // https://fpgasoftware.intel.com/eula.
15
16  // PROGRAM              "Quartus Prime"
17  // VERSION              "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
18  // CREATED              "Fri Apr 28 17:47:43 2023"
19
20  module S_e(
21          D0,
22          D1,
23          D2,
24          D3,
25          S_e
26  );
27
28
29  input wire      D0;
30  input wire      D1;
31  input wire      D2;
32  input wire      D3;
33  output wire     S_e;
34
35  wire    SYNTHESIZED_WIRE_11;
36  wire    SYNTHESIZED_WIRE_12;
37  wire    SYNTHESIZED_WIRE_13;
```

```
wire    SYNTHESIZED_WIRE_5;
wire    SYNTHESIZED_WIRE_6;
wire    SYNTHESIZED_WIRE_7;
wire    SYNTHESIZED_WIRE_8;




assign  SYNTHESIZED_WIRE_5 = D0 & SYNTHESIZED_WIRE_11 & SYNTHESIZED_WIRE_12;

assign  SYNTHESIZED_WIRE_8 = SYNTHESIZED_WIRE_11 & D2 & SYNTHESIZED_WIRE_13;

assign  SYNTHESIZED_WIRE_6 = D0 & D2 & SYNTHESIZED_WIRE_13;

assign  S_e = SYNTHESIZED_WIRE_5 | SYNTHESIZED_WIRE_6 | SYNTHESIZED_WIRE_7 |
    SYNTHESIZED_WIRE_8;

assign  SYNTHESIZED_WIRE_11 =  ~D1;

assign  SYNTHESIZED_WIRE_12 =  ~D2;

assign  SYNTHESIZED_WIRE_13 =  ~D3;

assign  SYNTHESIZED_WIRE_7 = D0 & SYNTHESIZED_WIRE_12 & SYNTHESIZED_WIRE_13;


endmodule
```

```
  // Copyright (C) 2020  Intel Corporation. All rights reserved.
  // Your use of Intel Corporation's design tools, logic functions
  // and other software and tools, and any partner logic
  // functions, and any output files from any of the foregoing
  // (including device programming or simulation files), and any
  // associated documentation or information are expressly subject
  // to the terms and conditions of the Intel Program License
  // Subscription Agreement, the Intel Quartus Prime License Agreement,
  // the Intel FPGA IP License Agreement, or other applicable license
  // agreement, including, without limitation, that your use is for
  // the sole purpose of programming logic devices manufactured by
  // Intel and sold by Intel or its authorized distributors.  Please
  // refer to the applicable agreement for further details, at
  // https://fpgasoftware.intel.com/eula.

  // PROGRAM            "Quartus Prime"
  // VERSION            "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
  // CREATED            "Fri Apr 28 17:47:37 2023"

  module S_f(
    D0,
    D1,
    D2,
    D3,
    S_f
  );


  input wire    D0;
  input wire    D1;
  input wire    D2;
  input wire    D3;
```

```
33  output wire   S_f;
34
35  wire  SYNTHESIZED_WIRE_10;
36  wire  SYNTHESIZED_WIRE_11;
37  wire  SYNTHESIZED_WIRE_5;
38  wire  SYNTHESIZED_WIRE_6;
39  wire  SYNTHESIZED_WIRE_7;
40  wire  SYNTHESIZED_WIRE_8;
41  wire  SYNTHESIZED_WIRE_9;
42
43
44
45
46  assign        SYNTHESIZED_WIRE_6 = D0 & SYNTHESIZED_WIRE_10 & SYNTHESIZED_WIRE_11;
47
48  assign        SYNTHESIZED_WIRE_9 = D1 & SYNTHESIZED_WIRE_10 & SYNTHESIZED_WIRE_11;
49
50  assign        SYNTHESIZED_WIRE_7 = D0 & D1 & SYNTHESIZED_WIRE_11;
51
52  assign        SYNTHESIZED_WIRE_8 = D0 & SYNTHESIZED_WIRE_5 & D2 & D3;
53
54  assign        S_f = SYNTHESIZED_WIRE_6 | SYNTHESIZED_WIRE_7 | SYNTHESIZED_WIRE_8 |
        SYNTHESIZED_WIRE_9;
55
56  assign        SYNTHESIZED_WIRE_5 =  ~D1;
57
58  assign        SYNTHESIZED_WIRE_10 =  ~D2;
59
60  assign        SYNTHESIZED_WIRE_11 =  ~D3;
61
62
63  endmodule
```

```
1   // Copyright (C) 2020  Intel Corporation. All rights reserved.
2   // Your use of Intel Corporation's design tools, logic functions
3   // and other software and tools, and any partner logic
4   // functions, and any output files from any of the foregoing
5   // (including device programming or simulation files), and any
6   // associated documentation or information are expressly subject
7   // to the terms and conditions of the Intel Program License
8   // Subscription Agreement, the Intel Quartus Prime License Agreement,
9   // the Intel FPGA IP License Agreement, or other applicable license
10  // agreement, including, without limitation, that your use is for
11  // the sole purpose of programming logic devices manufactured by
12  // Intel and sold by Intel or its authorized distributors.  Please
13  // refer to the applicable agreement for further details, at
14  // https://fpgasoftware.intel.com/eula.
15
16  // PROGRAM            "Quartus Prime"
17  // VERSION            "Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition"
18  // CREATED            "Fri Apr 28 17:47:15 2023"
19
20  module S_g(
21    D0,
22    D1,
23    D2,
24    D3,
25    S_g
26  );
27
```

```
28
29  input wire    D0;
30  input wire    D1;
31  input wire    D2;
32  input wire    D3;
33  output wire   S_g;
34
35  wire  SYNTHESIZED_WIRE_9;
36  wire  SYNTHESIZED_WIRE_1;
37  wire  SYNTHESIZED_WIRE_10;
38  wire  SYNTHESIZED_WIRE_3;
39  wire  SYNTHESIZED_WIRE_6;
40  wire  SYNTHESIZED_WIRE_7;
41  wire  SYNTHESIZED_WIRE_8;
42
43
44
45
46  assign        SYNTHESIZED_WIRE_8 = SYNTHESIZED_WIRE_9 & SYNTHESIZED_WIRE_1 &
        SYNTHESIZED_WIRE_10;
47
48  assign        SYNTHESIZED_WIRE_6 = SYNTHESIZED_WIRE_3 & SYNTHESIZED_WIRE_9 & D2 &
        D3;
49
50  assign        SYNTHESIZED_WIRE_7 = D0 & D1 & D2 & SYNTHESIZED_WIRE_10;
51
52  assign        S_g = SYNTHESIZED_WIRE_6 | SYNTHESIZED_WIRE_7 | SYNTHESIZED_WIRE_8;
53
54  assign        SYNTHESIZED_WIRE_3 =  ~D0;
55
56  assign        SYNTHESIZED_WIRE_9 =  ~D1;
57
58  assign        SYNTHESIZED_WIRE_1 =  ~D2;
59
60  assign        SYNTHESIZED_WIRE_10 =  ~D3;
61
62
63  endmodule
```