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

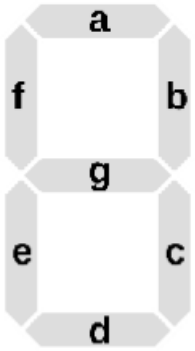

Figure 2: Digits to Hex

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).

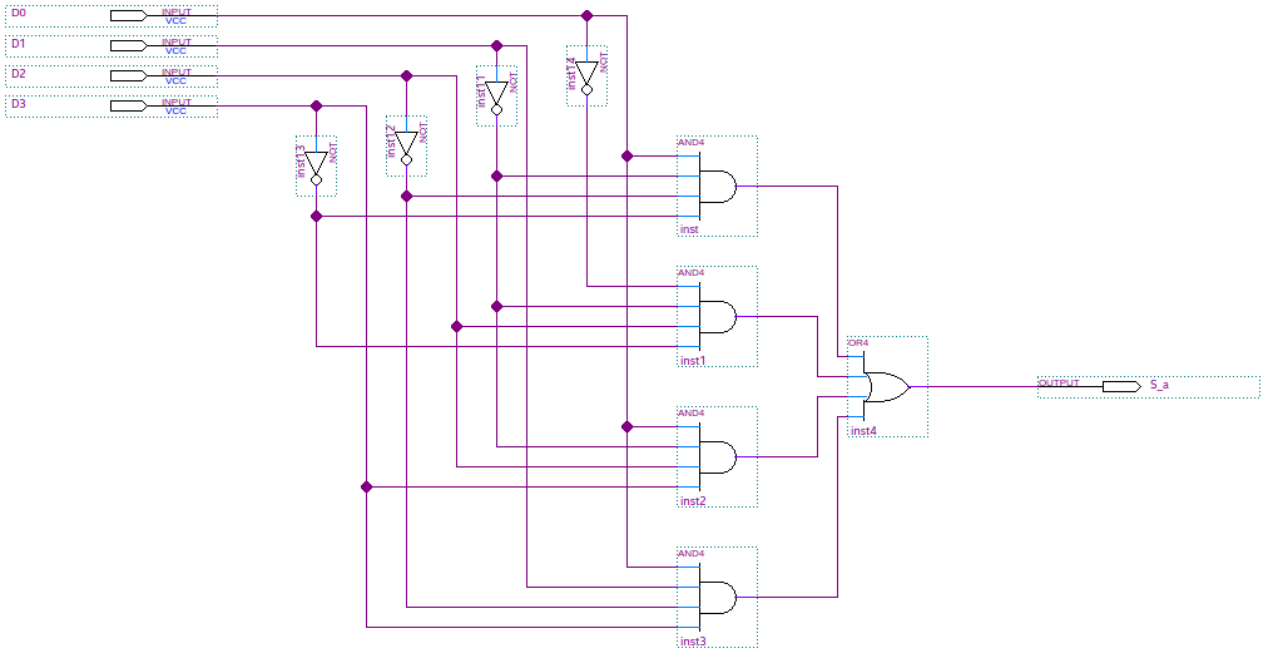

Figure 3: S_a Digital Logic Schematic

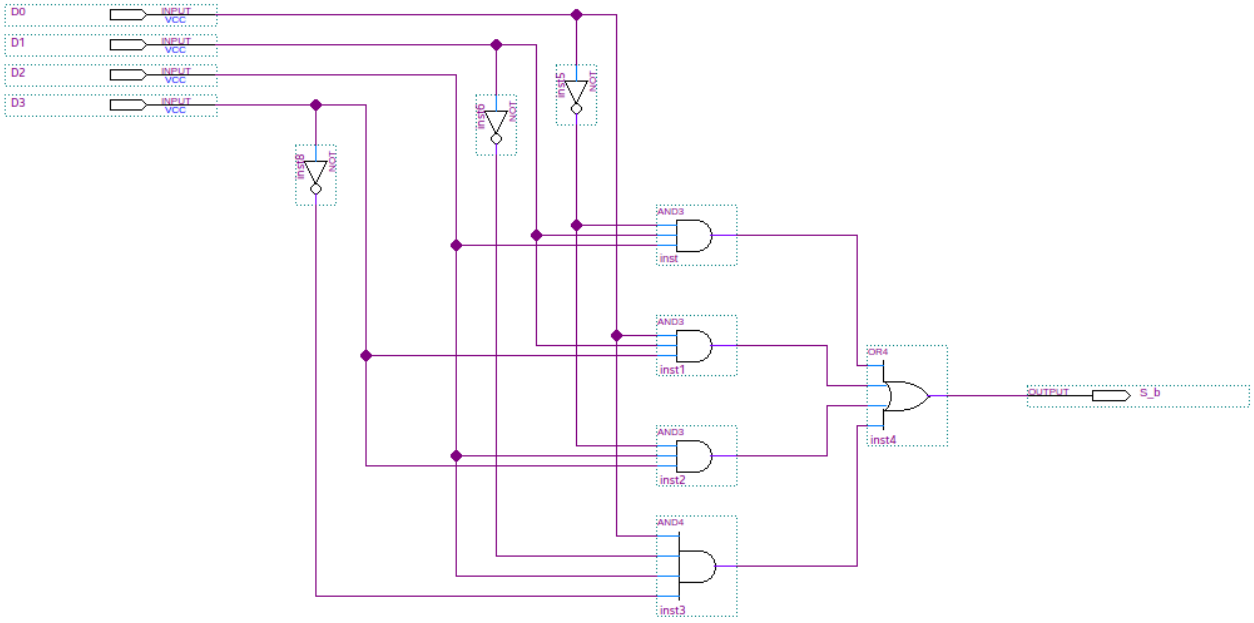

Figure 4: S_b Digital Logic Schematic

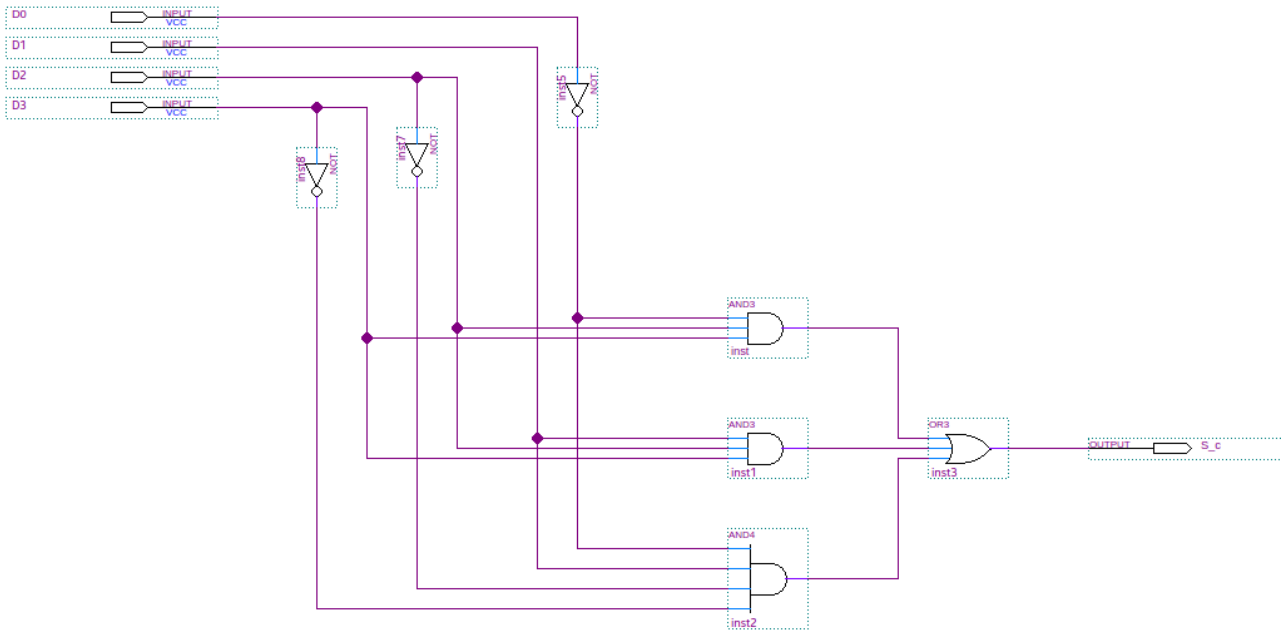

Figure 5: S_c Digital Logic Schematic

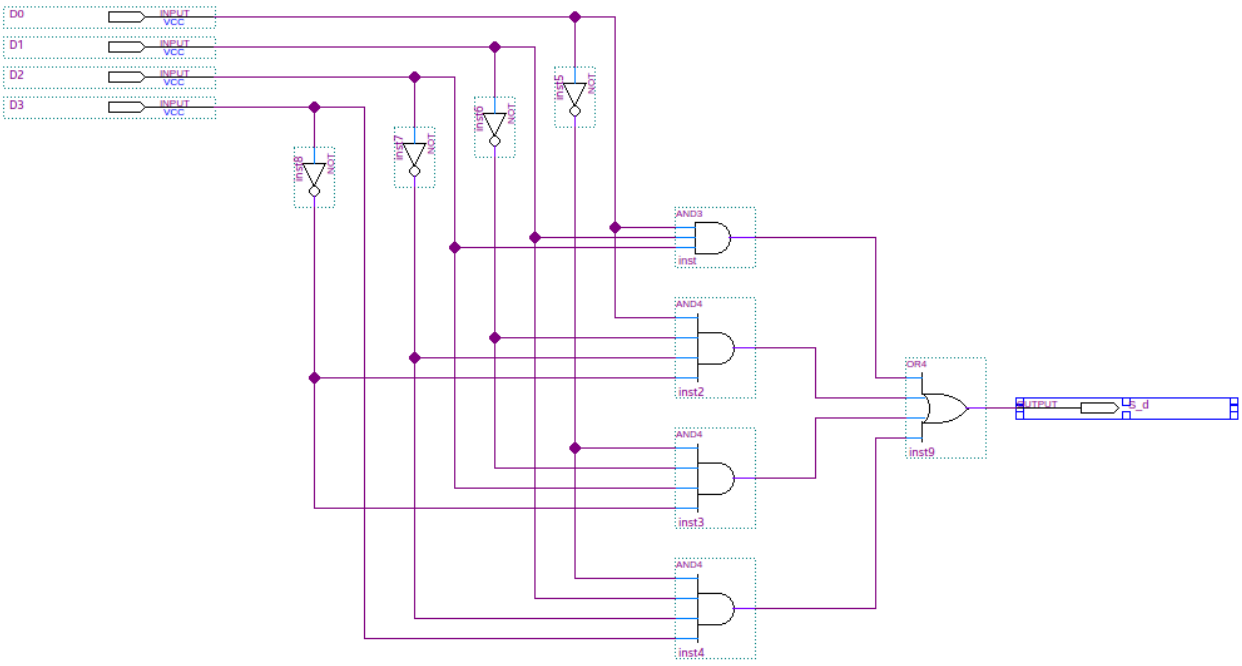

Figure 6: S_e Digital Logic Schematic

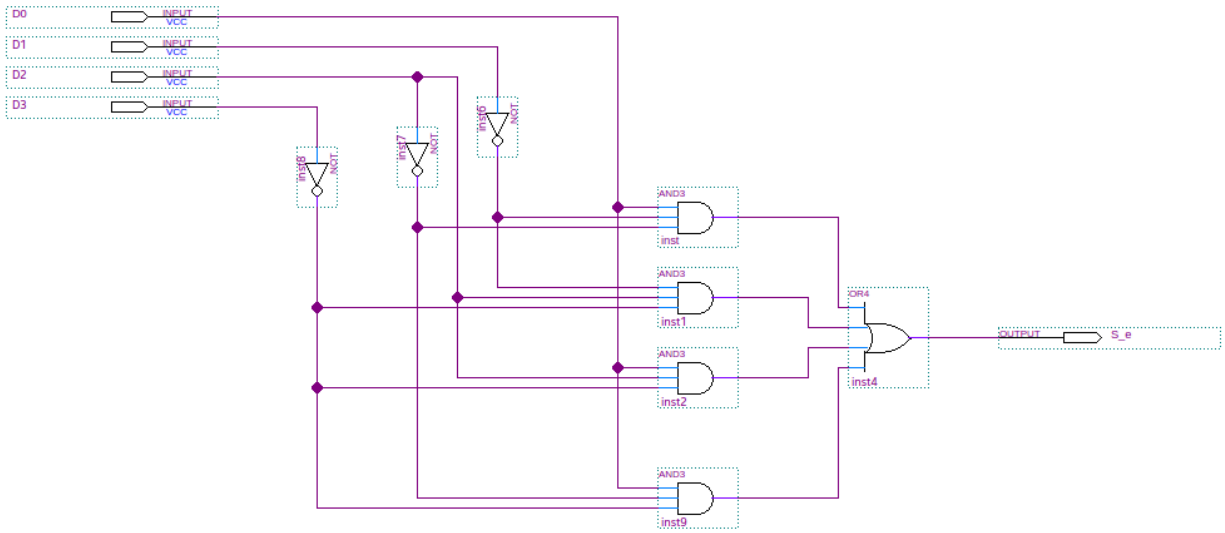

Figure 7: S_f Digital Logic Schematic

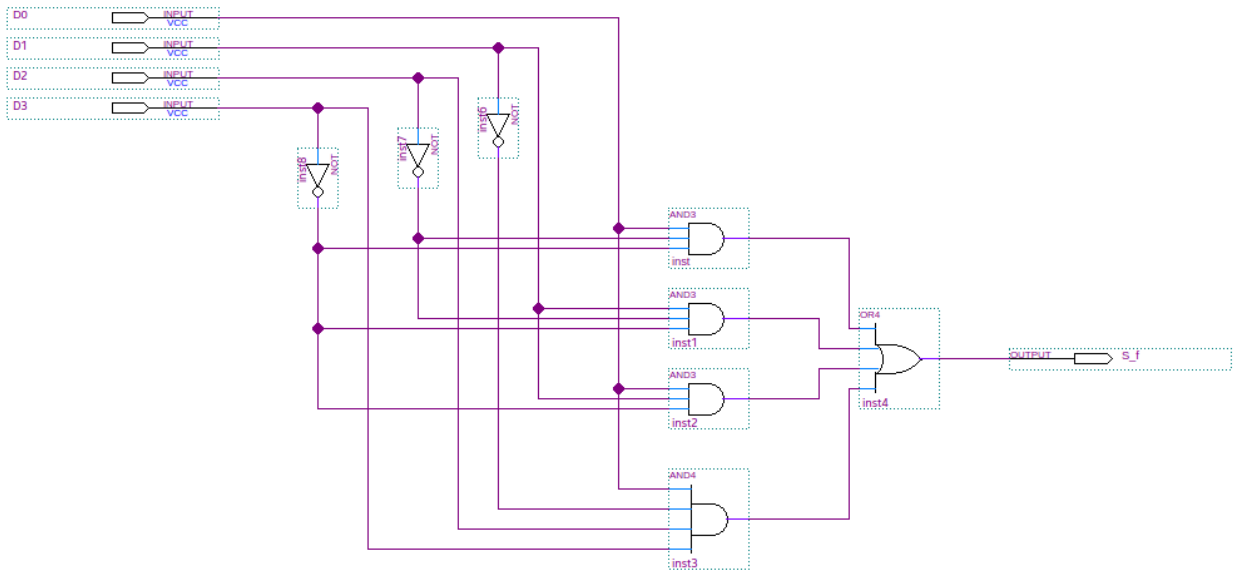

Figure 8: S_g Digital Logic Schematic

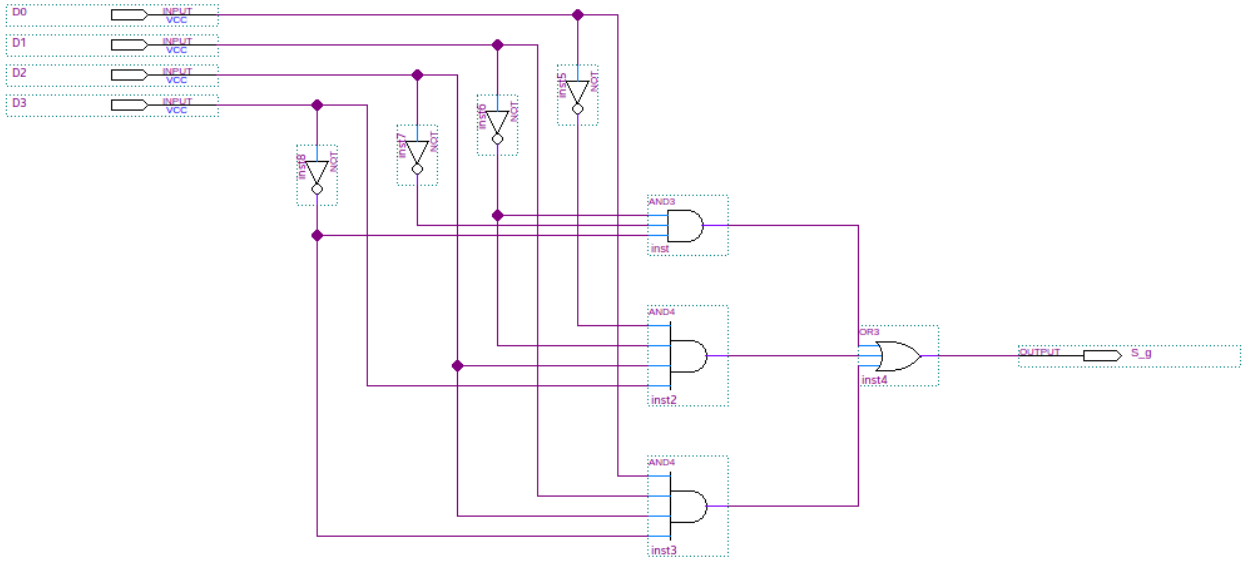

Figure 9: S_g Digital Logic Schematic

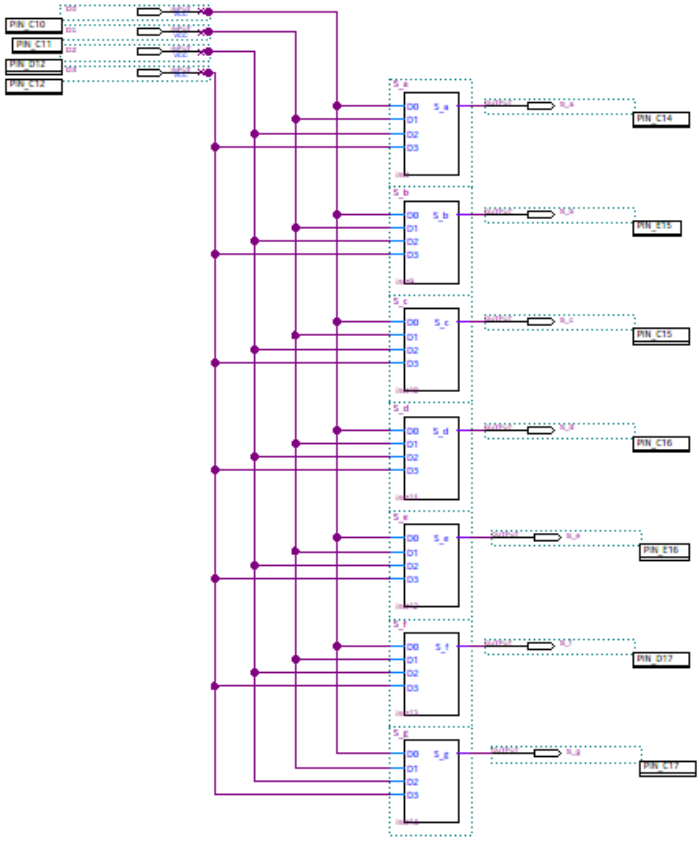

Figure 10: Full Binary To Hex Display Digital Logic Schematic

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.

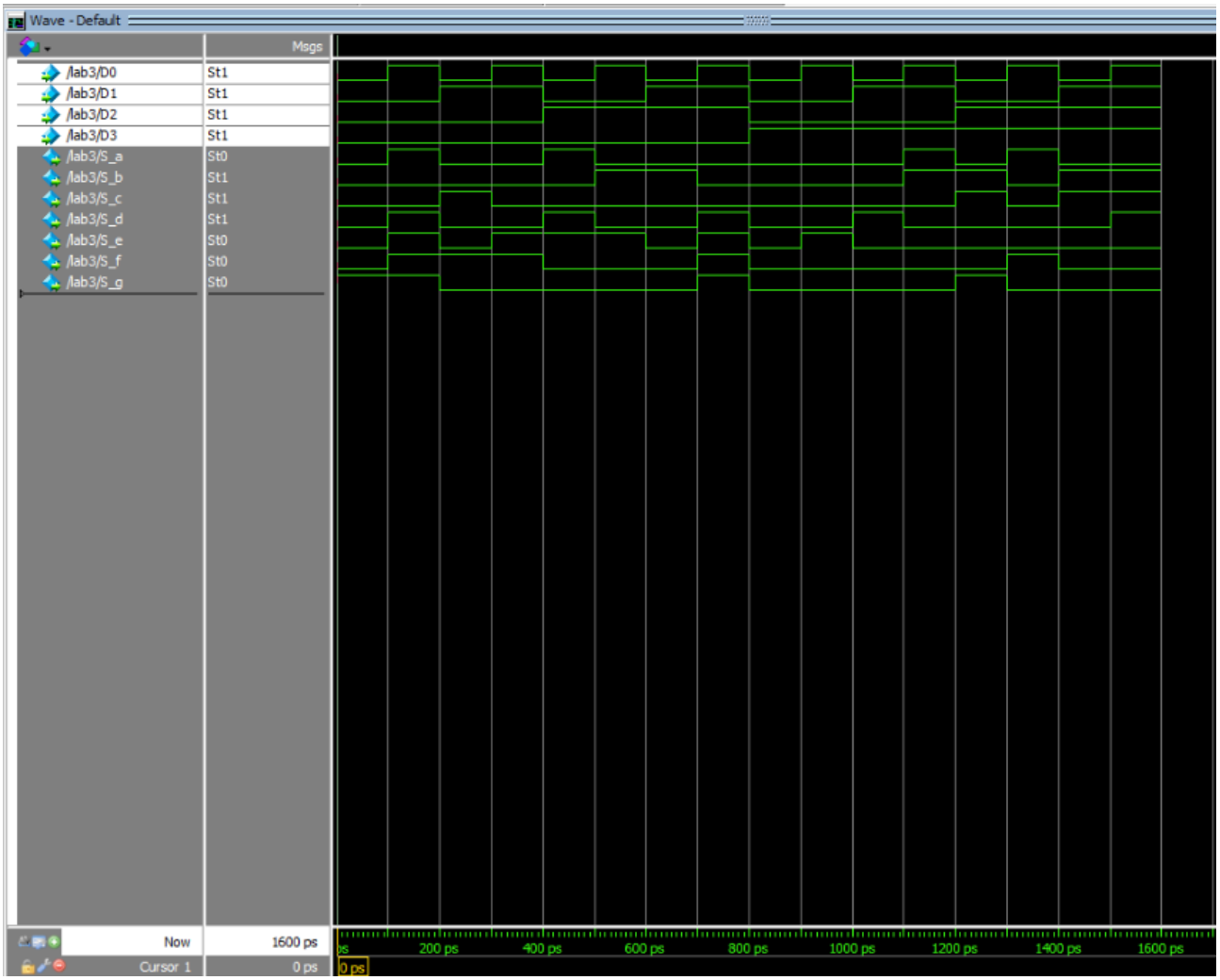

Figure 11: Full Binary To Hex Display Digital Logic Schematic

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.