Braidyn Sheffield

Dr. Mahoor

ENCE 3100

11/16/2024

Final Project: A Digital Lock

For our final project, we designed and implemented a digital lock using our FPGA board. The digital lock is to be a finite state machine with a few project specifications that we need to implement. When the combination, B-C-A-C, is inputted into the system, an unlock signal is sent. Once the lock is unlocked, any input will cause the lock to lock again. When you are partly through inputting the code for the lock, if an input of, A-A, is inputted, that will bring you back to the initial state regardless of what state you are in. If the wrong code is inputted into the system, the alarm signal will be sent. To prevent ease of figuring out the code to the lock, the system must wait until four inputs are input into the system for the alarm signal to be sent. Once the alarm signal is sent, the only way to turn it off is inputting the sequence, C-A.

To start with this project, we designed a state diagram that would meet all the specifications that need to be met above. The state diagram can be seen in Figure 1. There are a total of thirteen states. Each state has two different outputs. The first output for each state shows the output for the alarm. The second output shows the signal for the unlock signal. On each arrow, there are three bits. These bits show which input will make that change in state, that being either A, B, or C. For example, if an arrow has a value of 011, that means that the input is either B or C, this does not indicate that both B and C need to be inputted at the same time. This was done for simplicity and readability of the state diagram.

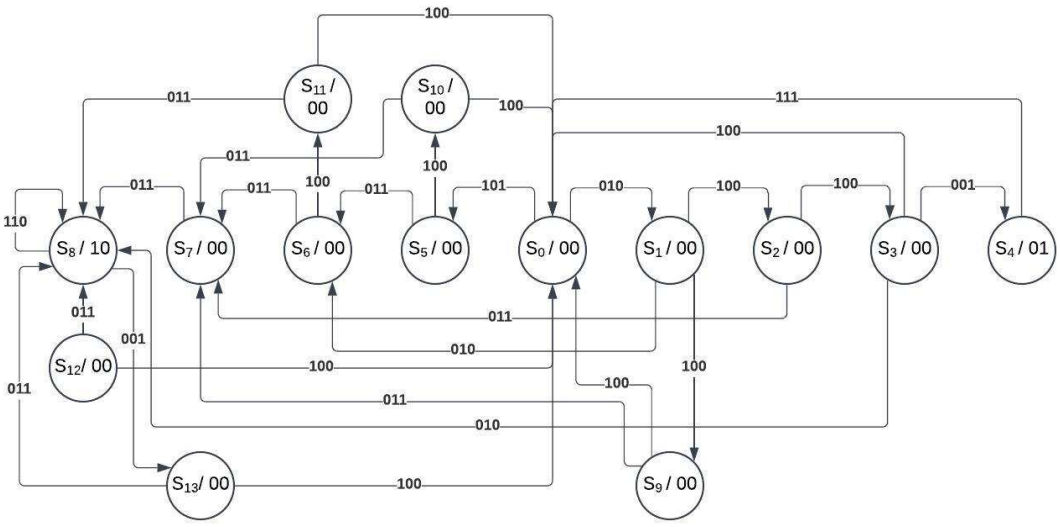

Figure 1: State Diagram of Digital Lock

After we have designed and checked the state diagram for correctness, the state table was built. The state table can be found in Table 1. The state table shows the same information as the state diagram, it was designed for ease of building and writing the Verilog Code.

Table 1: State Table for Digital Lock

Present State	Next State					Output	
	ABC =	000	100	010	001	Alarm	Unlock
S_0		S_0	S_5	S_1	S_5	0	0
S_1		S_1	S_9	S_6	S_2	0	0
S_2		S_2	S_3	S_7	S_7	0	0
S_3		S_3	S_0	S_8	S_4	0	0
S_4		S_4	S_0	S_0	S_0	0	1
S_5		S_5	S_{10}	S_6	S_6	0	0
S_6		S_6	S_{11}	S_7	S_7	0	0
S_7		S_7	S_{12}	S_8	S_8	0	0
S_8		S_8	S_8	S_8	S_{13}	1	0
S_9		S_9	S_0	S_7	S_7	0	0
S_{10}		S_{10}	S_0	S_7	S_7	0	0
S_{11}		S_{11}	S_0	S_8	S_8	0	0
S_{12}		S_{12}	S_0	S_8	S_8	0	0
S_{13}		S_{13}	S_0	S_8	S_8	0	0

After the state table was finalized, we were able to move onto begin writing the Verilog Code. A new project was built in Quartus, and a new Verilog File was created. We used behavioral syntax when writing the Verilog Code, using a case statement block assigning each of the states and describing which state to move too depending on which input is inputted. The Verilog Code can be found in Figure 2 and Figure 3.

```
module Final_Project (Clk, A, B, C, Alarm, Unlock);
   input A, B, C, Clk;
   output Alarm;
   output Unlock;
   reg y;

   parameter S0 = 4'b0000, S1 = 4'b0001, S2 = 4'b0010, S3 = 4'b0011, S4 = 4'b0100, S5 = 4'b0101;
   parameter S6 = 4'b0110, S7 = 4'b0111, S8 = 4'b1000, S9 = 4'b1001, S10 = 4'b1010, S11 = 4'b1011;
   parameter S12 = 4'b1100, S13 = 4'b1101;

   always@(posedge  Clk)
      case(y)
         S0:
            if(A) y<= S5;
            else if (B) y<= S1;
            else if (C) y <= S5;
            else y <= S0;
         S1:
            if(A) y<= S9;
            else if (B) y<= S6;
            else if (C) y <= S2;
            else y <= S1;
         S2:
            if(A) y<= S3;
            else if (B) y<= S7;
            else if (C) y <= S7;
            else y <= S2;
         S3:
            if(A) y<= S0;
            else if (B) y<= S8;
            else if (C) y <= S4;
            else y <= S3;
         S4:
            if(A) y<= S0;
            else if (B) y<= S0;
            else if (C) y <= S0;
            else y <= S4;
         S5:
            if(A) y<= S10;
            else if (B) y<= S6;
            else if (C) y <= S6;
            else y <= S5;
         S6:
            if(A) y<= S11;
            else if (B) y<= S7;
            else if (C) y <= S7;
            else y <= S6;
```

Figure 2: First Part of Verilog Code for Digital Lock

```
48          S7:
49             if(A) y<= S12;
50             else if (B) y<= S8;
51             else if (C) y <= S8;
52             else y <= S7;
53          S8:
54             if(A) y<= S8;
55             else if (B) y<= S8;
56             else if (C) y <= S13;
57             else y <= S8;
58          S9:
59             if(A) y<= S0;
60             else if (B) y<= S7;
61             else if (C) y <= S7;
62             else y <= S9;
63          S10:
64             if(A) y<= S0;
65             else if (B) y<= S7;
66             else if (C) y <= S7;
67             else y <= S10;
68          S11:
69             if(A) y<= S0;
70             else if (B) y<= S8;
71             else if (C) y <= S8;
72             else y <= S11;
73          S12:
74             if(A) y<= S0;
75             else if (B) y<= S8;
76             else if (C) y <= S8;
77             else y <= S12;
78          S13:
79             if(A) y<= S0;
80             else if (B) y<= S8;
81             else if (C) y <= S8;
82             else y <= S13;
83
84       endcase
85
86    always @(posedge Clk) begin
87        Unlock <= (y == S4);
88        Alarm <= (y == S8);
89     end
90
91  endmodule
```

Figure 3: Second Part of Verilog Code for Digital Lock

Once the Verilog Code was written, we were able to compile the code and check for errors and then once there were no errors, we were able to do the pin assignments for the system. For pin assignments, each input had a dedicated switch on the FPGA board. A was set to SW[2], B was set to SW[1], and C was set to SW[0]. For the clock of the system, we used the push button as a manual clock input. LEDR[0] was used as indication that the alarm signal was high, and LEDR[9] was set to be when the unlock signal was high. Once pin assignments were done,

we looked at the state diagram viewer in Quartus. The state diagram from Quartus can be found in Figure 4.

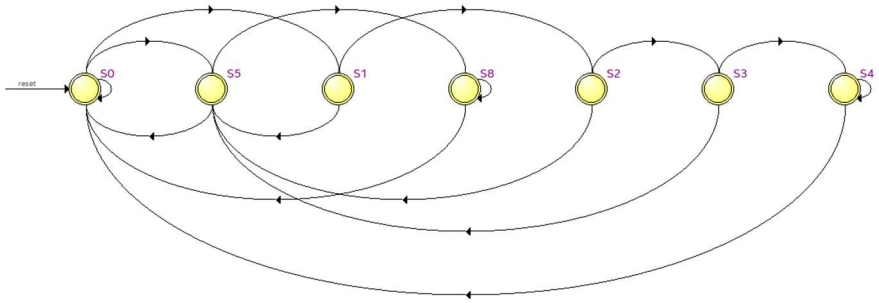

Figure 4: Quartus State Diagram for Digital Lock

We then uploaded the code to our FPGA board and tested our implementation of the digital lock. Two cases of the digital lock is shown below in Figure 5 and Figure 6. In Figure 5, it shows the case when the user inputs the wrong combination, and the alarm signal is high. In Figure 6, it shows the case that the right input sequence was inputted into the system and the unlock signal is high.

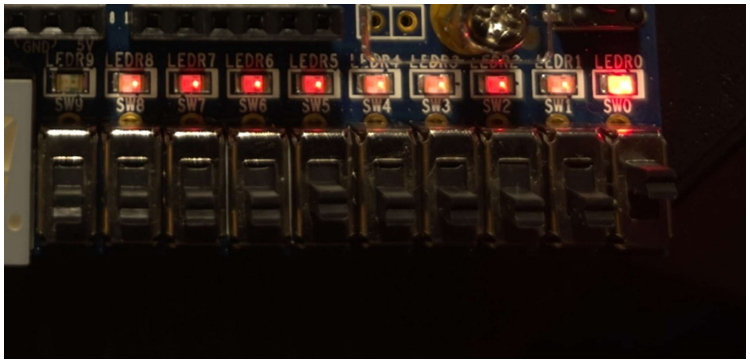

Figure 5: First Implementation of Digital Lock

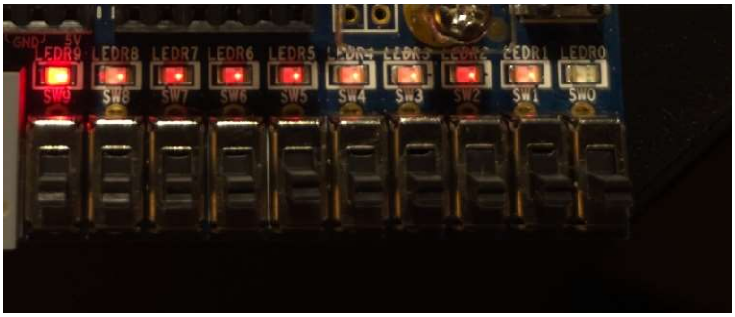

Figure 6: Second Implementation of Digital Lock