CMPE 160 Laboratory Exercise 9

Design and Simulation of a Moore State Machine

Andrei Tumbar
Performed: April 2nd
Submitted: April 9th

Lab Section: 4
Instructor: Mr. Byers
TA: Sam Myers
Kobe Balin
Georgi Thomas

Lecture Section: 1
Professor: Mr. Cliver

Abstract

In this laboratory exercise a Moore state machine was implemented to create a sequence detector. There are two input pins meaning the sequence is one of a two bit number that is denoted, `AB`. An output `Z` will go to high when the sequence `(11)`, `(10)`, `(00)` is detected. `Z` will also stay high if the sequence is detected and the input remains `(00)`. To implement this a state diagram was defined which then yielded a state table followed by a K-map to define the functionality of the state machine. A circuit was implemented in Quartus and simulated in Modelsim. Two different sequences were tested, one was given and the other was created to fully test the circuit.

Design Methodology

A Moore state machine is a function of only the state. This means that the output must not include any input terms. The output in this case is simply indicating whether the sequence was reached. The state can be represented by a two bit number denoted $Q_1 Q_0$. This state is held in two D flip-flops. The flow of the state given inputs can be mapped.

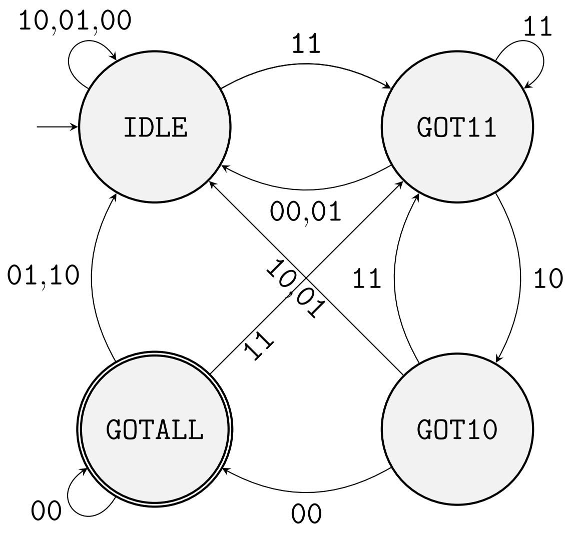


Figure 1: State diagram of sequence detecting Moore state machine.

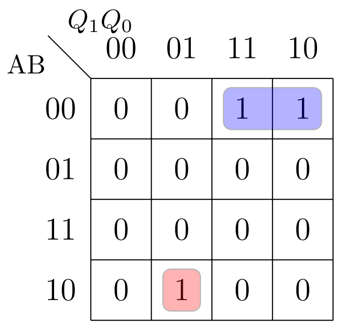
Figure 1 shows the flow of the state machine given certain states and inputs. The circles represent the four different states that the machine can be in. `IDLE` represents the state where the detector is waiting for the first pair in the sequence. The arrow to left of this state denotes that this is the starting state. The machine is put into the `IDLE` state initially by the active-low asynchronous reset signal `RST`. The arrows on this graph represent how the state would change given a set of input. For example, the bottom arrow labeled `00` is pointing from the `GOT10` state to the `GOTALL` indicating that if the machine is presently in the `GOT10` state and recieves an input of `00`, on the next rising clock edge it would be put into the `GOTALL` state.

After a state diagram is created, a transition table is created to quantitatively describe the functionality of the circuit.

Table 1: Truth Table for mathematical operations N^2 and $5N$

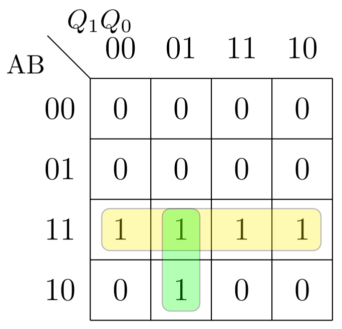
A B	IDLE	GOT11	GOT10	GOTALL	Q
0 0	IDLE	IDLE	GOTALL	GOTALL	
0 1	IDLE	IDLE	IDLE	IDLE	Q^*
1 1	GOT11	GOT11	GOT11	GOT11	
1 0	IDLE	GOT10	IDLE	IDLE	

Table 1 shows the next state given current state and input A and B. This table can be used to create a Karnaugh Map to define the functionality. Using the 2-bit state definitions previously described, two K-maps are created for the next value of each bit in the state.



AB \ Q_1Q_0	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	0	0	0	0
10	0	1	0	0

(a) K-Map ${Q_1}^*$ function



AB \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	1	0	0

(b) K-Map ${Q_0}^*$ function

Figure 2: K-Maps for next state functions.

Using the K-maps shown in Figure 2 a set of min-terms can be extracted from the groupings shown. An equation for each next-state bit is required as well as an equation to describe the output `Z`.

$$
\begin{aligned}
{Q_1}^* &= \bar{A}\bar{B}Q_1 + A\bar{B}\overline{Q_1}Q_0 && (1)\\
&= \bar{B}(\bar{A}Q_1 + A\overline{Q_1}Q_0) && (2)\\
{Q_0}^* &= AB + A\overline{Q_1}Q_0 && (3)\\
Z &= Q_1\overline{Q_0} && (4)
\end{aligned}
$$

Equations 1-3 were extracted directly from the K-maps. Equation 3 was not fully simplified so that the output of one of the gates could be reused in both the ${Q_1}^*$ and ${Q_0}^*$ functions. The Z function is a function of only the state as per the definition of a Moore function.

Given the next-state functions as boolean expressions of the input and the current state, a circuit schematic can be implemented.

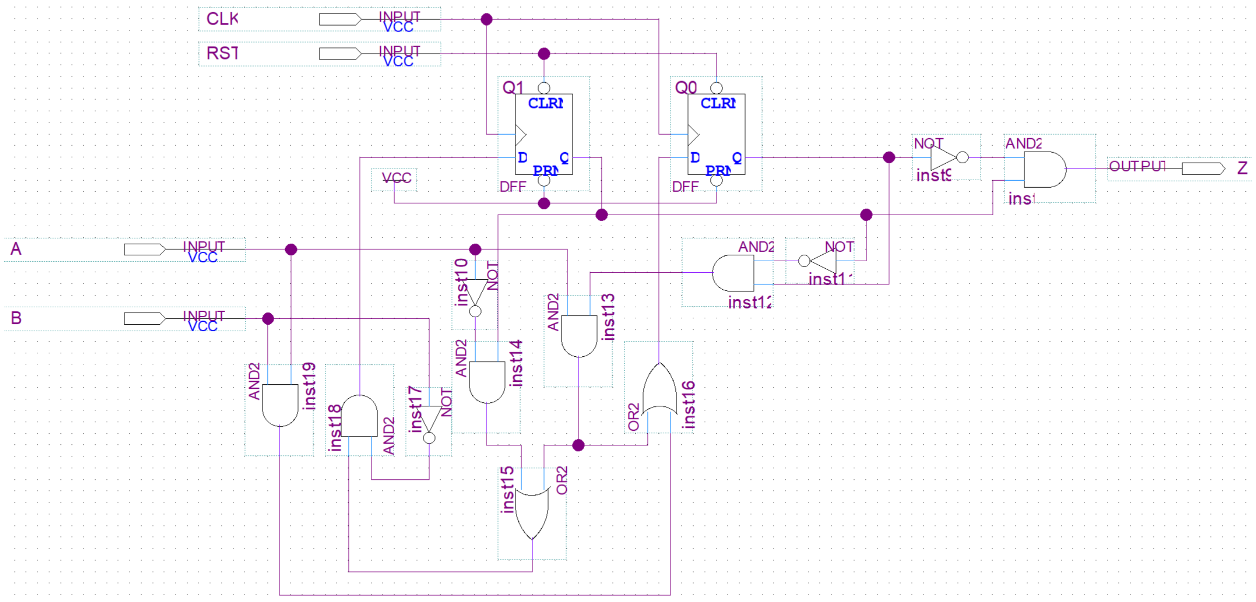


Figure 3: State machine circuit schematic.

Figure 3 shows the circuit implementation of the functions defined in the equations above. An asynchronous reset input pin was added to reset the state before taking an input sequence. The asynchronous set pins on the two D flip-flops are connected to 5 V source to keep the circuit from using this.

Results and Analysis

The circuit implemented in Quartus was tested using Modelsim to verify the integrity of the functions. Two different cases were used to test the functionality. The function test sequence used was given while the second was created to illustrate the full use of the state machine. The following two sequences were used:

1. (A,B) = (1,1),(1,0),(0,0),(1,0),(0,0),(1,1),(1,0),(0,0),(0,0)
2. (A,B) = (0,1),(0,0),(1,0),(1,1),(1,1),(1,0),(1,1),(1,0),(0,0),(0,0),(1,1)

The second sequence will test every state and transition as well as hold the output to `HIGH` before moving back to a low state. Both sequences were tested using Modelsim and the wave captures were recorded.

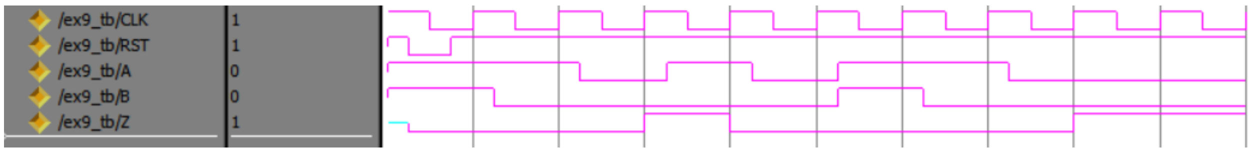


Figure 4: Wave capture of the first sequence.

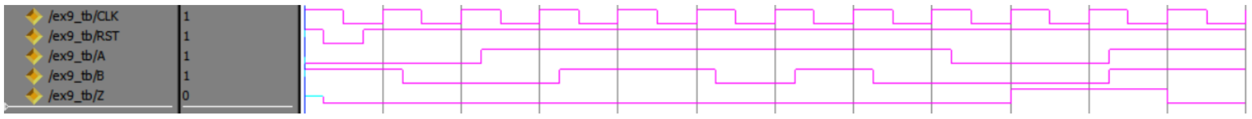


Figure 5: Wave capture of the second sequence.

The wave captures yielded the correct set of outputs. This indicates that the circuit schematic was correct.

The timing of this circuit is important when finding the constraints of it. The minimum clock period can be described as the sum of all of the delays measured in the circuit.

$$T_{min} = t_{co} + t_{pd} + t_s = 6.232\, ns + 0\, ns + 2.004\, ns = 8.236\, ns \tag{5}$$

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{8.236 \cdot 10^{-6}} = 121.4\, kHz \tag{6}$$

The Equation 5 indicated the minimum clock period which can be used to calculate the maximum clock frequency of 121.4 kHz.

Conclusion

In this laboratory exercise, the synthesis of a state machine was reviewed. A state diagram was created which then generated a state table. The state table allowed the creation of a K-map. From the K-map the boolean functions describing the next state could be extracted and a circuit could be implemented in Quartus. The circuit was then tested using two different input sequences in Modelsim. Finally, maximum frequency was calculated using measured delays from quartus. This exercise was successful as the desired functionality described in the lab prompt was achieved by the state machine.

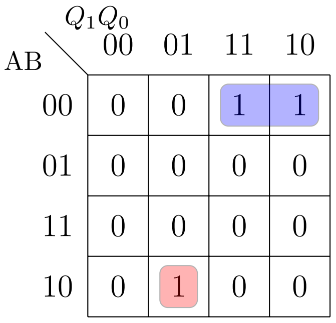
Questions

1. Repeat the synthesis of the process of your state machine for the following state encoding style.

Table 2: Alternative states and encodings for the Moore state machine

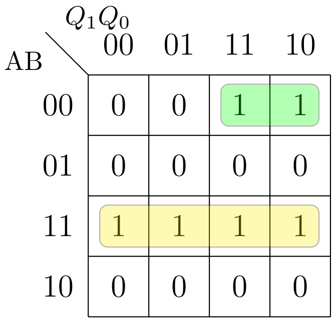
State name	State	
	Q1	Q0
Idle	0	0
Got11	0	1
Got10	1	0
GotAll	1	1

The state diagram and next-state table can be reused from the previous design. A new set of K-maps and equations however must be created.



AB \ Q_1Q_0	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	0	0	0	0
10	0	1	0	0

(a) K-Map $Q_1{}^*$ function



AB \ Q_1Q_0	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	1	1
10	0	0	0	0

(b) K-Map $Q_0{}^*$ function

Figure 6: K-Maps for next state functions.

The first K-map (Figure 6a) is identical to the one in Figure 2a because the first bit definition is same across all of the states. The second K-map is slightly different to accomodate the change. The equations for the next states can now be extracted.

$$Q_1{}^* = \bar{A}\bar{B}Q_1 + A\bar{B}\overline{Q_1}Q_0 \tag{7}$$
$$= \bar{B}(\bar{A}Q_1 + A\overline{Q_1}Q_0) \tag{8}$$
$$Q_0{}^* = AB + \bar{A}\bar{B}Q_1 \tag{9}$$
$$Z = Q_1Q_0 \tag{10}$$

$Q_1{}^*$ is the same function as before. $Q_0{}^*$ is slightly different from the old function. Z was changed to still only be high when the state is `GOTALL`.

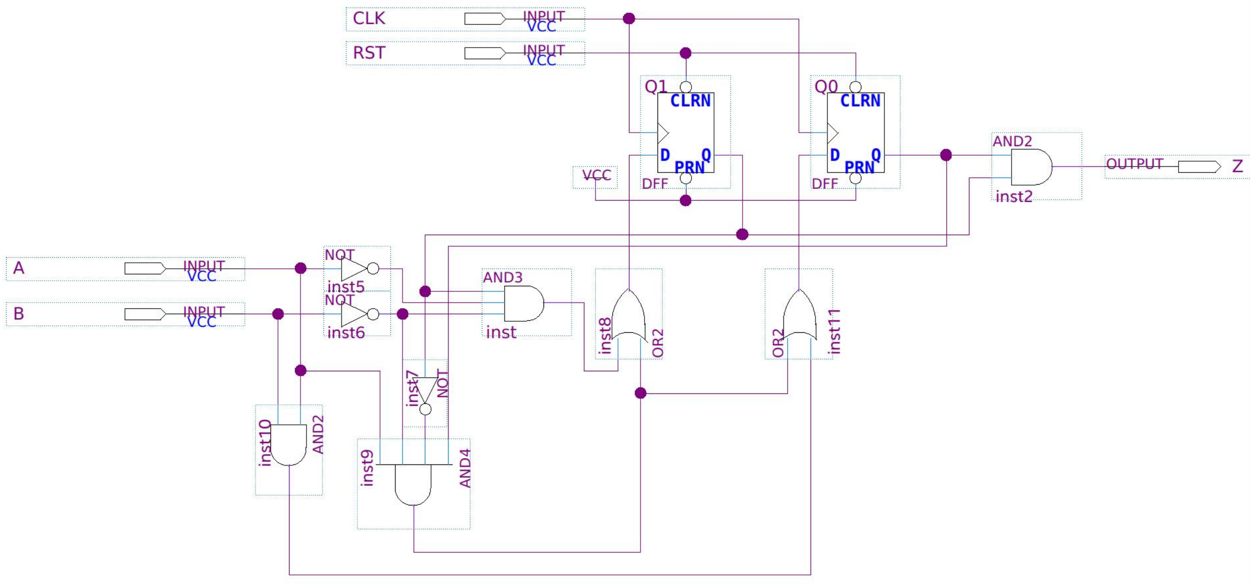


Figure 7: State machine circuit schematic with altered encoding.

Figure 7 shows the circuit schematic of the new encoding state machine.

2. How many different ways are there to encode a state machine with four states that is implemented using two D flip-flops? Note that the answer should be a number, not "one-hot" or "gray".

   There are 4! or 24 different combinations of encodings that can be used to a encode a state machine with four states.

3. What is "one-hot" encoding style?

   One hot encoding is useful because it is easier to implement than other types of encoding. In "one-hot" encoding, if there are N states, there must be N flip-flops. Each flip-flop will represent a single state. All other flip-flops must be in the low state when one is high.