

ELL318 Digital Hardware Design: Lecture 7

Lecturer: Debanjan Bhowmik, Indian Institute of Technology Delhi
Scribed by: Vardaan Taneja (2019EE10061)
and Shambhav Tewari (2019EE10657)

Contents: DFA, Mealy and Moore Machines

1 Introduction

In previous courses when we studied digital electronics, we studied finite state machines. In this course we studied these machines using the theory of computation template and subsequently named them Discrete Finite Automata (DFA) and Non-Discrete Finite Automata (NFA). A NFA is a theoretical tool for analysis, we have already proved equivalence between DFA and NFA. Thus any analysis performed through a NFA will automatically apply to the DFA which we actually construct. Our aim in this lecture is to show that a DFA is equivalent to a Moore machine with binary output and then convert this DFA to a Mealy machine.

2 DFA, Moore and Mealy Machines

2.1 Example 1:

Suppose we have a DFA (A) which accepts the language

$$L(A) = \{w \mid w \text{ has an odd number of } 1s\}$$

Note that, a single 0 is not part of the language. The DFA for the language is shown below, in form of the 5-tuple and in the form of a transition diagram.

$$A = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, q_2)$$

$$\delta : \{q_1, q_2\} \times \{0, 1\} \rightarrow \{q_1, q_2\}$$

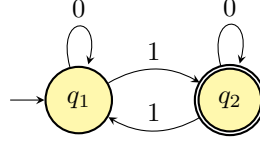


Figure 1: Given $DFA(A)$

This same information above can be written in a different form which will resemble a Moore Machine. We shall construct a new DFA, in this new DFA the accept states will be associated with 1 and the reject states will be associated with the 0 state. We can define a new machine: A_{moore} which is defined as shown below.

$$A_m \equiv (\{q1, q2\}, \{0, 1\}, \Sigma_{out}, \delta, q_1, \delta_{out}) \quad (1)$$

$$\text{where, } \delta_{out} : \{q1, q2\} \rightarrow \{0, 1\}, \quad \Sigma_{out} \equiv \{0, 1\}$$

In this new machine we make the following additions, we add a new alphabet Σ_{out} which is the set $\{0, 1\}$ and for this new alphabet we define a new transition function δ_{out} which maps the states $\{q1, q2\}$ to 0 or 1.

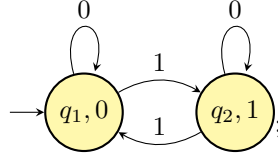


Figure 2: New Finite State Machine (A_{moore})

Now we look into the Mealy machine. In case of a Mealy machine, the output is associated with a transition, and not a state. When we reach q_1 upon input 0 from q_1 itself, the output is 0. Also when we reach q_1 upon input 1 from q_2 , then the output is 0 too. Similarly, when we reach q_2 upon input 1 from q_1 , the output is 1, and when we reach q_2 upon input 0 from q_2 , the output is still 1. The intuition is that if via a transition we reach a state that is an accept state then our output will be 1 else 0 (*so the output associated with the next state in the Moore machine determines the output associated with the related transition in the Mealy machine*).

In some examples, when we covert from Moore to Mealy machine, the number of states goes down, but in this particular case we looked at (Figure 2: Moore, Figure 3: Mealy), it does not. To define the equivalent Mealy Machine we do the following, we associate the outputs of $\{0, 1\}$ with transitions rather than the states. In this particular example, A_{mealy} is quite similar to A_{moore} the only difference being, δ_{out} connects $\{q_1, q_2\} \times \{0, 1\} \rightarrow \Sigma_{out}$ in Mealy and δ_{out} connects $\{q_1, q_2\} \rightarrow \Sigma_{out}$ in Moore.

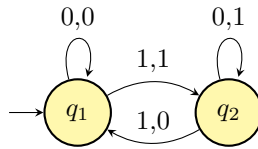


Figure 3: New Finite State Machine (A_{mealy})

2.2 Example 2:

In the previous section we took a DFA and changed it to a Moore machine and subsequently a Mealy machine. The motivation behind the exercise is to show that the DFA and the Moore machine are practically the same and carry the same information. What's called 'reject' state in DFA is a state with output 0 (say) in Moore machine, and what's called 'accept' state in DFA is a state with output 1 (say).

But in the Mealy machine, we have to make slight modifications. In the previous example, we were not able to expand much upon this point. Now we further explore this facet of automata theory. The number of states in the Mealy Machine may be less than that in a Moore machine. The reason for this is that in the Moore machine the output is linked to a state, whereas in Mealy it is linked to a transition, thus one state in Mealy can do that job of two or more states in Moore. This will be clear from the example below.

Consider the DFA that accepts the strings which end in '1011', the input alphabet Σ is 0,1. The DFA for this is described below:

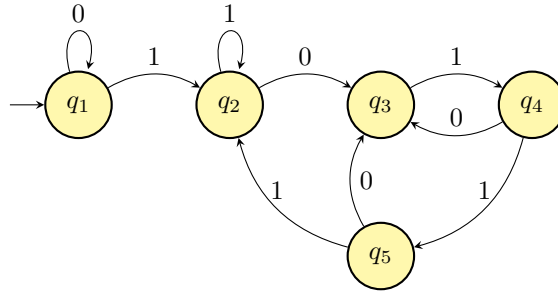


Figure 4: A DFA that accepts strings that end with '1011' (q_5 is the accept state)

Converting this DFA to a Moore machine should now be trivial; as discussed previously the states that are not accept states are associated with 0 and the accept states are associated with 1. The Moore machine looks as follows:

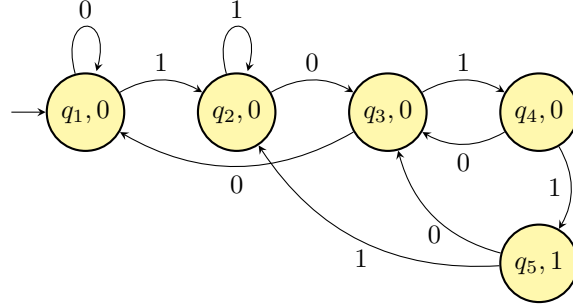


Figure 5: Converted Moore Machine that outputs '1' for input strings that end with '1011'

Now, we develop a systematic method to convert this Moore machine to a Mealy machine. The principle for reduction of the states is as follows: “*For any input, if for two states the next state is the same, then we club the two states together.*”

δ	
$q_1, 0 \rightarrow q_1$	$q_1, 1 \rightarrow q_2$
$q_2, 0 \rightarrow q_3$	$q_2, 1 \rightarrow q_2$
$q_3, 0 \rightarrow q_1$	$q_3, 1 \rightarrow q_4$
$q_4, 0 \rightarrow q_3$	$q_4, 1 \rightarrow q_5$
$q_5, 0 \rightarrow q_3$	$q_5, 1 \rightarrow q_2$

Table 1: Transition Table for Mealy Machine

δ_{out}
$q_1 \rightarrow 0$
$q_2 \rightarrow 0$
$q_3 \rightarrow 0$
$q_4 \rightarrow 0$
$q_5 \rightarrow 1$

Table 2: δ_{out} for Moore Machine

Referring to Table 1, note that $q_2, 0 \rightarrow q_3$ and $q_2, 1 \rightarrow q_2$ and if we look at q_5 , note that $q_5, 0 \rightarrow q_3$ and $q_5, 1 \rightarrow q_2$. Thus from the principle of reduction of states: for any input q_2 and q_5 have the same output thus we can club these two states together. If we look at our state transition diagram we can see that in the DFA/Moore machine the only difference between q_2 and q_5 was the output state associated with it. Once we make the switch to a Mealy machine the output is decoupled from the state and coupled to the transition, hence q_2 and q_5 become the same state. Also note that once we club q_2 and q_5 , we cannot further reduce the number of states in this case because this kind of reduction cannot be done for the other states here.

For states q_1, q_2, q_3, q_4 we can convert the Moore to Mealy machine exactly how we did in example 1. We will now focus on how to deal with eliminating state q_5 . Consider that from state q_4 when you go to q_5 we have input 1. We can convert this to the following statement, from q_4 on input 1 we go to q_2 with output 1. We can make this change because states q_2 and q_5 are equivalent. Once we make this change, we can see that we have defined all transitions and we have a complete Mealy machine. One important point to note is that if in the transition table we had $q_5, 1 \rightarrow q_5$ then we wouldn't have been able to club

q_2 and q_5 . Always remember that we can only club two states when final states exactly match. We will do this reduction/ state minimization formally in the subsequent lectures.

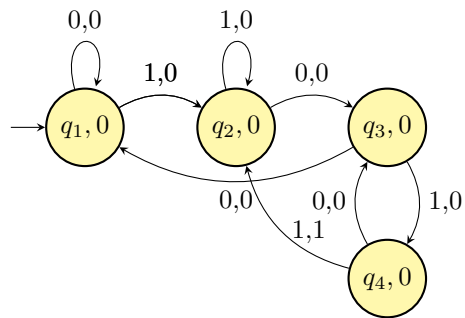
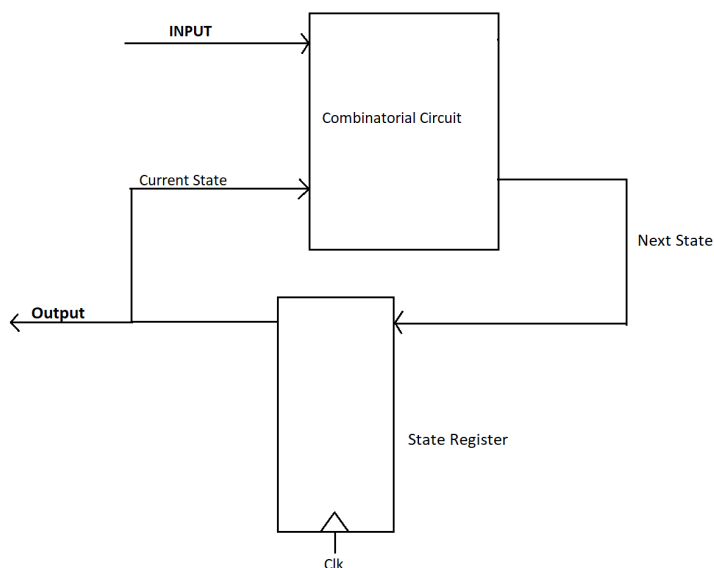


Figure 6: Reduced Mealy Machine

3 Formal Definition and Implementation Details

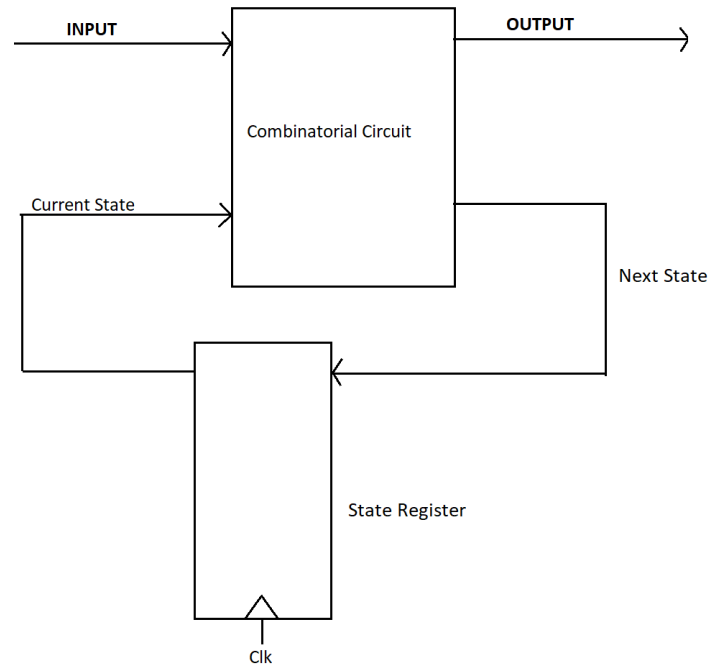
Figure 7: General schematic regarding implementation of a Mealy machine



Now that we have discussed Mealy and Moore machines in an abstract/theoretical way, we conclude this lecture with a broad overview of the implementation of these machines at a high level abstraction, followed by their formal definitions. Assume we have a combinational circuit, the next state depends on current state and input, thus these two are the inputs to our circuit as shown in Figure 7. In a Moore machine we will get the next state as the output (The circuit models the state transition table). So far in the course we have assumed some form of memory to store the value of the next state and then further input it as the next state. In physical hardware this is where the concept of a flip-flop/register comes in; this is also shown in Figure 7 above. We will also have a trivial combinational circuit that will take the next state and generate an output to decipher if the state is an accept state or not.

To implement a Mealy machine, we have to implement: $\delta_{out} : Q \times \Sigma \rightarrow \Sigma_{out}$. Thus the combinational circuit block will also generate the output value, as shown in Figure 8. Now, suppose we have inputs coming in and we have a clock that runs this machine, the state transition occurs at, say, the negative edge. Thus the input state will only update on the negative edges of the clock. The output of the Moore machine is tied with the next state (Figure 7), but that is not the case with the Mealy machine. Thus Moore machine will lag behind the Mealy machine. This is one implementation-level difference between a Mealy

Figure 8: General schematic regarding implementation of a Mealy machine



machine and an equivalent Moore machine.

4 References

‘Digital Hardware Design’ lecture series on Youtube by Prof. Balakrishnan, Department of Computer Science and Engineering, IIT Delhi