A Study of different Finite state machine design and efficient coding techniques

[1]Bhaumik Vaidya [2]Devani Anupam

**[1 2]Department of Electronics Engineering,
Gujarat Technological University,
Gandhinagar, Gujarat, India.**

[1]vaidya.bhaumik@gmail.com
[2]anupam_devani@yahoo.com

ABSTRACT: Finite State Machines (FSM), are the mostcomplex structures found in almost all digital systems today. There are mainly two types of finite state machines Mealy and Moore state machines. So in this paper comparison between the two is done using an example. Hardware Description Languages are used for high-leveldigital system design. Verilog and VHDL provide the capability of different coding styles for FSMs. Therefore, a choice of a coding style is needed to achieve specific performance goals and to minimize resource utilization on a chip. This paper is a study of the trade-offs that can be made by changing coding styles. A comparative study on different FSM encoding styles is shown to address their impact on performance and resource utilization.

KEYWORDS:*Finite State Machine, Mealy Machine, Moore Machine, Binary and One hot Encoding*

1. Introduction

In any programming languages, there will always be more than one way to code the same problem. Hardware description languages like Verilog and VHDL are no different. It also provides several alternatives to the designer as to how to accomplish a same task.

So It is up to the designer to code in an efficient way that results in optimal performance and minimum resource utilization.

In this paper, first the comparison between the Mealy and Moore state machine is provided. Both has its own advantages and disadvantages so it is up to designer to choose appropriate design based on application. Same way comparison is done between different State encoding schemes like binary and one hot encoding. The effect on area and performance by using these schemes is also discussed in the paper.

2. Types of state machines:

There are mainly two types of state machine.

1. Mealy state machine
2. Moore state machine

In Mealy state machine output depends on present input and present state while in Moore state machine output only depend upon present state. Figure 1 shows the general flow chart of mealy state machine while figure 2 shows the general flow chart of Moore state machine.

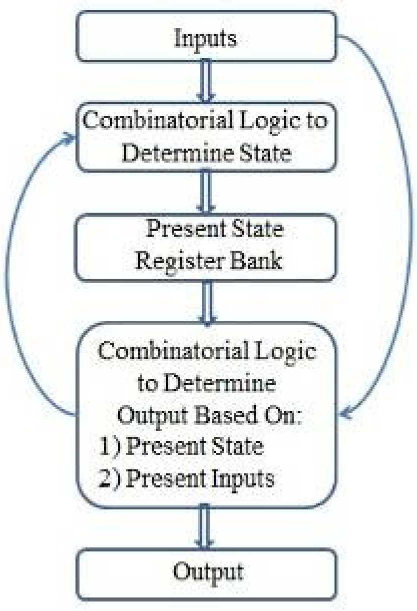
Inputs
Combinatorial Logic to Determine State
Present State Register Bank
Combinatorial Logic to Determine Output Based On:
1) Present State
2) Present Inputs
Output

Figure 1 Mealy State Machine

As can be seen from Figure 1 in Mealy state machine combinatorial logic is there to determine the value of next state depending upon the value of input. Output is also determined by combinatorial logic based on input and present state value. Sequential logic is only used for storing states value.Same is the case in Figure 2 for Moore machine but here output only depend upon the present state value.

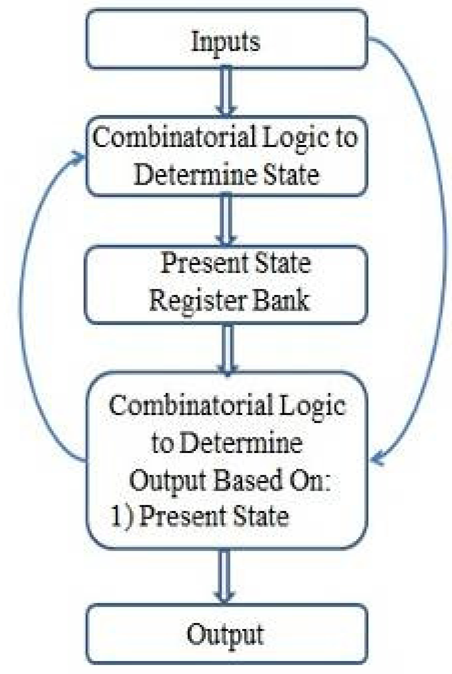
Inputs
Combinatorial Logic to Determine State
Present State Register Bank
Combinatorial Logic to Determine Output Based On: 1) Present State
Output

Figure 2 Moore State Machine

Now to compare Mealy and Moore machine one example is taken. Consider the case of a circuit to detect a pair of 1's or 0's in the single bit input.
If two one's or two zero's comes one after another, output should go high. Otherwise output should be low.
Here is a Moore type state transition diagram for the circuit:
•When reset, state goes to 00
•If input is 1, state will be 01
•If input is 0, state goes to 10
•State will be 11 if input repeats
•After state 11, goes to 10 states or 01 depending on the input. Once the state reaches the state 11 then assign 1 to outsince it is an asynchronous we have to check the reset before making output high.

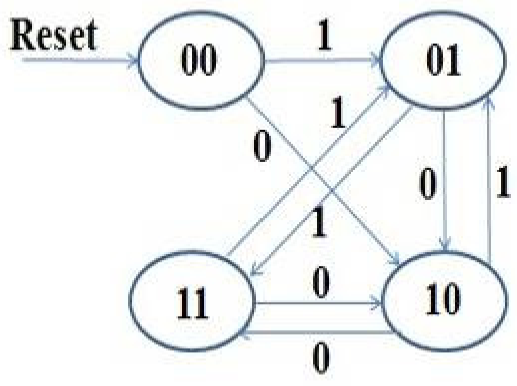

Figure 3State Diagram of Moore Machine

Now, let us re-design the above circuit using Mealy style state machine.
Output depends on both state and input
State transition diagram is as follows:

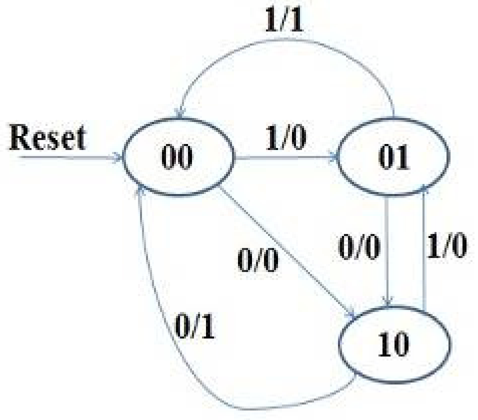

Figure 4State Diagram of Mealy Machine

3. Comparison between two state machines:
Moore state machine is easier to design than Mealy. First design the states depending on the previous state and input. Then design output only depending on state. Whereas in Mealy, you have to consider both state and input while designing the output.
In Moore four states are used to design the circuit and mealy three states are used so there are more number of states in mealy machine then Moore machine. So sometimes number of flip flops will be more in the Moore machine.
In Mealy, output changes immediately when the input changes. We can observe this point when you simulate the codes above. In Moore example, output becomes high in the clock next to the clock in which state goes 11. So, Mealy is faster than Moore. Mealy gives immediate response to input and Moore gives response in the next clock.
Now, most of the designer use only one always block to design state machine so combinational and sequential logic is mixed in this block so it might create a problem when you synthesize the code and then verify gate level netlist. So It is more efficient if you separate combinational and sequential logic in the Mealy and Moore machines.

Modelling Mealy state machine
```
always@(in or pres_state) //Next State Decoder
always@(posedge clock) //Memory
always@(in or pres_state) //Output Decoder
```

Modelling Moore state machine
```
always@(in or pres_state) //NS Decoder
always@(posedge clock) //Memory
always@(pres_state) //Output Decoder
```

4. Results:

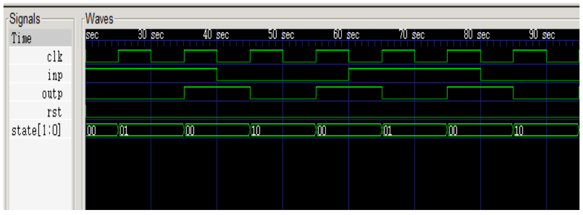

Figure 5 Mealy Machine waveforms

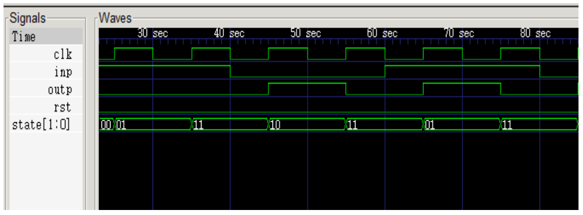

Figure 6Moore Machine waveforms

From above results we can conclude that Mealy machine will give output on the same clock edge while Moore machine will give output on the next clock edge.
So, that Mealy state machine is faster than Moore state machine.

5. FSM state Encoding Schemes:

Each state in a finite state machine can be represented with a unique pattern of one's and zeros and it is called state encoding. Two most popular encoding schemes are binary and one hot encoding. In this paper, will briefly discuss both of them and discuss how to select the best encoding scheme that suits your design, so efficient performance and resource usage can be ensured.
In binary encoding the relationship between the number of state bits and number of states is represented by the following equation.

$$B = \log_2 (S)$$

So to implement the state machine with four states with a binary encoding scheme, two flip flops or state bits can be used to uniquely encode four states as follows:

State1 = "00"
State2 = "01"
State3 = "10"
State4 = "11"

The Gray code binary encoding scheme can also be used where one bit change at a time. Gray code binary scheme is useful when the outputs of the state bits are used asynchronously. For example , if state machine switches from state '10' to '01' as it does in sequential binary encoding and the registers do not switch the outputs as exactly the same time , temporary outputs of either '11' or '00' can exist. This type of fluctuation can cause unpredictable results throughout the circuit.

A one-hot encoding scheme uses one register for every state. For example four registers are used for a 4-state machine- with only one state bit high at a time. State encoding can be done as follows in one hot encoding:

State1 = "0001"
State2 = "0010"
State3 = "0100"
State4 = "1000"

6. Selecting an Encoding Scheme:

The encoding schemes are choose based on the complexity of the state machine, the target application and the requirement for recovering from illegal states.

For comparison between binary and one hot encoding scheme, one sample state machine was taken with six states. Verilog code was developed using binary and one hot encoding scheme and then synthesize it to evaluate area and number of gates used in it.

The report in design compiler [Synthesis engine from synthesis] shows that more number of gates was used in one hot encoding scheme and as a result more area was utilizedin it. The area utilized by one hot machine was 350 units and area utilized by binary encoding was 200 units so one hot encoding scheme utilized almost double area.

Although one-hot encoding scheme requires more registers, the logic is much simpler. In binary encoded machine, the logic that controls the transitions from one state to other depends on all state bits as well as the input to the state machine. This logic requires high fan-in functions to the inputs of the state bits. In one hot scheme the inputs to the state bits are often simply the function of other state bits. So binary encoding is a preferred approach in ASIC designs unless the timing in the output path is critical.

In FPGA, different types of architectures also favour certain types of encoding. For example, Altera FLEX 8000 device is register intensive, state machines can be implemented using one hot encoding scheme.

When choosing encoding method, number of potential illegal states should also be taken in to consideration. The design can enter in to an illegal state if there is a setup or hold violation. The one hot encoding scheme has more number of illegal states. For example , 14 state machine encoded with binary state machine has only 2 illegal state while coded using one hot encoding scheme has 16370 illegal states. So this factor should also be taken into consideration while choosing the encoding scheme.

7. Conclusion:

In this paper different types of state machines and types of encoding schemes are discussed. Comparison among them is also made. When there is need for faster state machine then mealy machine is used but it makes design complex. In complex design, to make state machine design simpler Moore machine is preferred. Encoding schemes also play a crucial role in determining the area and performance of the state machine. Binary encoding scheme uses less area but can lead to unpredictable performance if asynchronous outputs are used. One hot encoding uses more area but has better timing performance.

8. Reference:

1. IEEE standard Verilog Reference Manual , IEEE standard 2001
2. Verilog FAQ by Shivkumar Chanod , Needamangalam Balachandar
3. Design Compiler User Guide from Synopsys
4. Verilog HDL by Samir Palnitkar