International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 2, Issue 3, pp: (174-181), Month: July - September 2014, Available at: www.researchpublish.com

VHDL Implementation of Moore and Mealy State Machine

Md. Mehedi Hasan[1], Prajoy Podder[2], Jag Mohan Thakur[3], Anamul Haque[4], Mursalin Sayeed[5], Md. Rafiqul Islam[6]

[1,2,3] Department of ECE , [4,5,6] Department of EEE
[1,2,3,4,5,6]Khulna University of Engineering & Technology, Khulna, Bangladesh

Abstract: **Field Programmable Gate Array (FPGA) provides breakout performance capacity and system integration while optimizing to improve FPGA devices based on CAD tools in the Verilog Hardware Description Language (VHDL), which demonstrate the logic, function and behaviors of system hardware. State machines are often the backbone of FPGA development. Choosing the right architecture and implementation methods will ensure that an optimal solution can be obtained. For a designer, the best way to address these actions and sequences is by using a state machine. State machines are logical constructs that transition among a finite number of states. A state machine will be in only one state at a particular point in time. It will however, move between states depending upon a number of triggers. This paper mainly focuses on the design of Moore and Mealy state machine in FPGA. RTL logic has been adopted for designing purposes. The minimum time period, maximum delay, less number of flip flops, registers etc. have been considered to design the proposed Moore and Mealy state machines more effectively.**

Keywords: **Finite state machine, Moore state machine, Mealy state machine, State transition, Sequential systems, Time delay, Test bencher, Xilinx ISE simulator.**

I. INTRODUCTION

A Finite-state machine (FSM) or finite-state automaton is a mathematical model of computation used to design both computer programs and sequential logic circuits. It is considered as an abstract machine of a finite number of states. The machine is in only one state at a time is called the current state. It can change from one state to another state when initiated by a triggering event or condition, this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition [1] [6].

State machines are a method of modelling systems. Normally the machine's output depends on not only the recent input but also the entire history of inputs. The output is purely determined by their inputs .In digital logic system, state machines play an important role [4].

Finite-state machines can model a large number of problems. These problems include electronic design automation, communication protocol design, language parsing and other engineering applications. State machines or hierarchies of state machines have been used to describe neurological systems and in linguistics—to describe the grammars of natural languages in the field of biology and artificial intelligence research [3].

The rest of the paper is organized as follows. The overview of state machine is described on section-2. Under this section, Moore and Mealy state machines are explained clearly. State machine design process is explained in section-3.The basic difference of Moore and Mealy state machine are described in section- 4.The experimental results are discussed in section-5, where the pin diagram, RTL schematic diagram and the timing diagram of a Moore and Mealy state machine. Finally, Section-6 concludes the paper.

II. STATE MACHINE

Most digital electronic systems are designed as clocked sequential systems. Clocked sequential systems are a constrained form of Moore machine, where the state changes only when the global clock signal changes. Typically the current state is stored in flip-flops, and a global clock signal is connected to the "clock" input of the flip-flops. Clocked sequential systems are one way to solve met stability problem. A typical electronic Moore machine contains a combinational logic chain to decode the current state into the outputs (lambda) [2].

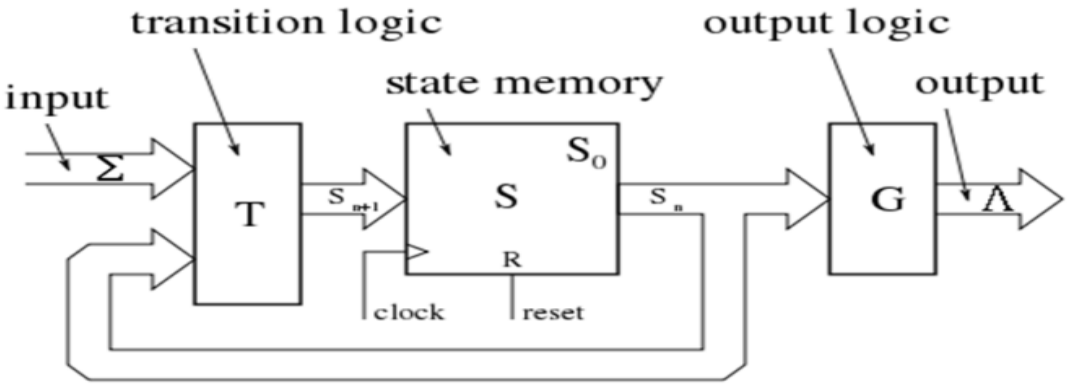


Fig.1: Finite State machine system

A. Moore State Machine

Moore state machine is a reading and writing memory device. The state machine begins in the IDLE state after a reset. It remains in the IDLE state until it receives a read or a write signal (rd or wr). It goes to the write state when receiving a write signal, assigning the write enable (write_en) and the acknowledged signal (ack) for one cycle. When the write operation is completed in a single cycle then it backs to the IDLE state. It goes to the READ1 state when receiving a read signal, assigning the output enable (out_en).It waits for the ready signal (ready) from the memory device, signaling that valid data is being output. It then goes to the READ2 state, keeping the output asserted and giving an acknowledgment before returning to the IDLE state. It is implied that the state machine remains in the current state for any condition other than ones. If the ready signal is not asserted, the machine stays in the READ state.
Simple Moore machine have one input and one output:

- Edge detector using XOR
- Binary adding machine
- Clocked sequential systems (a restricted form of Moore machine where the state changes only when the global clock signal changes)

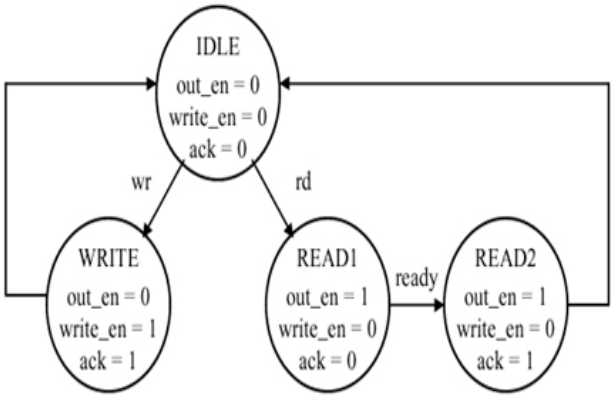


Fig.2: Moore state machine for reading and writing a memory device

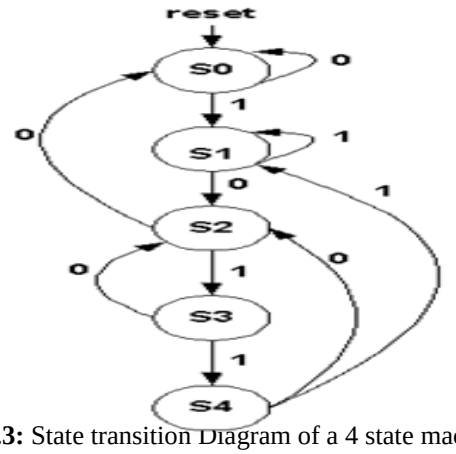


Fig.3: State transition Diagram of a 4 state machine

There are design techniques to confirm that no glitches occur on the outputs during that brief period whereas those changes are rippling through the chain, but most systems are designed so that glitches during that brief transition time are ignored or are irrelevant [6].

The outputs then stay the same indefinitely (LEDs stay bright, power stays connected to the motors, solenoids stay energized, etc.), until the Moore machine changes state again.

B. Mealy State Machine

A mealy state machine is one in which the output changes on the translations of the device. In other the output of the next state are determined by the current state and the current inputs. This type of state machine is in contrast to the Moore state machine in which each value of an output is associated with a specific current state.in a mealy machine, the during a particular state can be different at different times, depending on how the machine entered the current state. While either method Moore or mealy, can be used to describe any state machine.

Fig.4 describes a very simple state machine for reading and writing a memory device. The state machine begins in the IDLE state after a reset. If then wait in IDLE until it receives a read or a write signal (rd or wr).if it receives a write signal, it goes to the write state, assert the write enable (write_en) and the acknowledged signal (ack) for one cycle. If then return to the IDLE state, having complete the write operation in a signal cycle. If it receives a read signal while in the IDLE state, it then goes to the READ1 state and asserts the output enable (out_en).it wait for the ready signal (ready) from the memory device, signaling that valid data is being output. It then goes to the READ2 state, keeping the memory device signaling the valid data is being output. It then goes to READ2 state, keeping the output enable asserted and giving an acknowledgment before returning to the IDLE state.

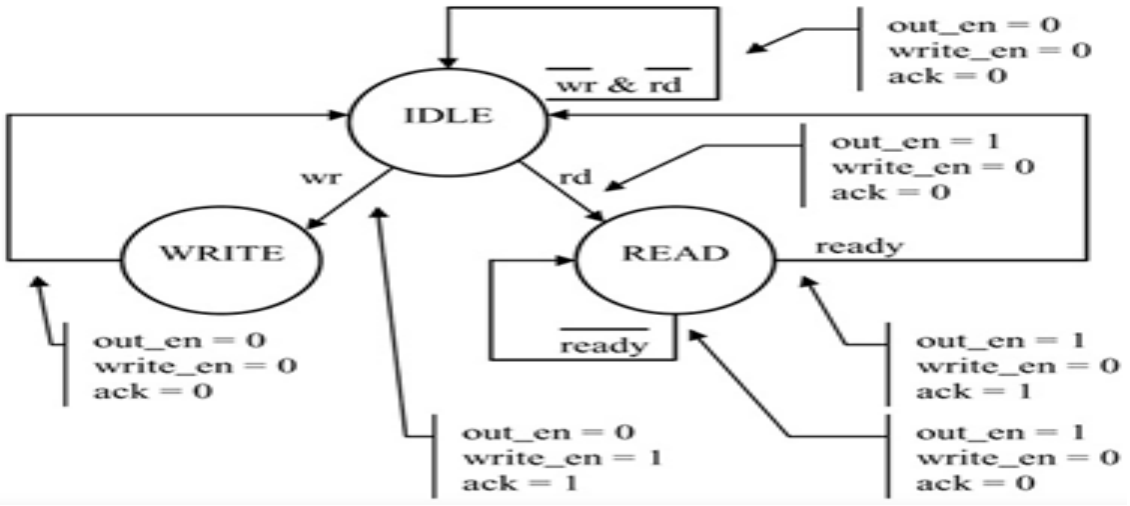


Fig.4: Mealy state machine for reading and writing a memory device

III. STATE MACHINE DESIGN PROCESS

Steps have been discussed below [6] [7],

1. Identify state variables S.
2. Identify output decoder & Next state decoder.
3. Build state transition diagram.
4. Minimize states.
5. Choose appropriate type of flip flops.
6. Choose state assignment (Assignment of Binary codes to machine states).
7. Design next state decoder& output decoder-Use combinational logic structured design methods.

IV. DIFFERENCES BETWEEN MEALY & MOORE STATE MACHINE

1) Mealy Machines tend to have less states

a) Different outputs on arcs (n^2) rather than states (n).

2) Moore Machines are safer to use

a) Outputs change at clock edge (always one cycle later).

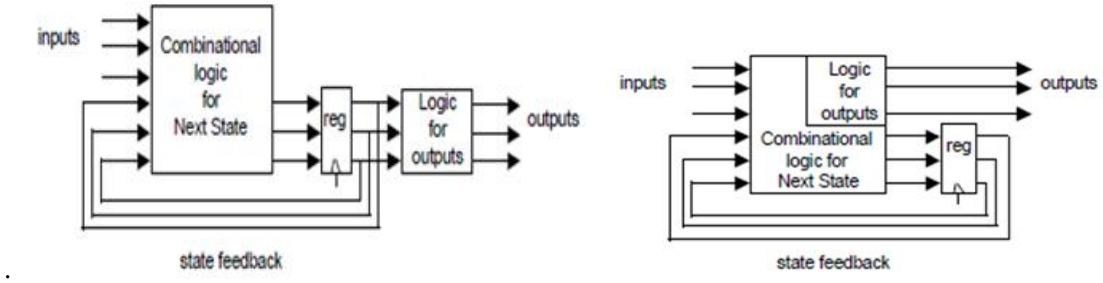
b) In Mealy machines, input change can cause output change as soon as logic is done – a big problem when two machines are interconnected asynchronous feedback.

3) Mealy Machines react faster to inputs

b) React in same cycle – don't need to wait for clock.

c) In Moore machines, more logic may be necessary to decode state into outputs – more gate delays after.

Fig.5 describes a Moore and Mealy state machine with state feedback



(a) **(b)**

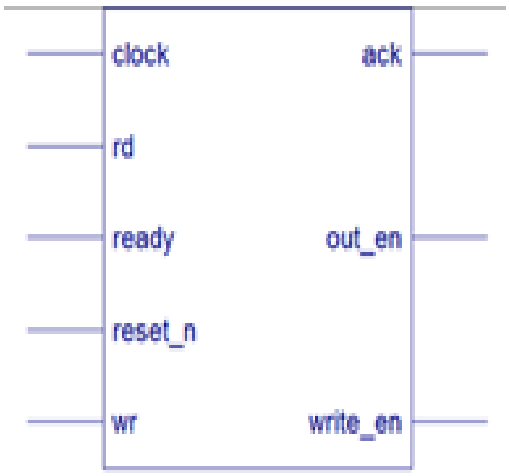
Fig.5: (a) Mealy machine (b) Moore machine

V. EXPERIMENTAL RESULTS

A. MOORE STATE MACHINE

a) Circuit Schematic

Figure 6 & 7 shows a very simple state machine for reading and writing a memory device. The state machine begins in the IDLE state after a reset. It then waits in IDLE until it receives a read or a write signal (rd or wr). If it receives a write signal, it goes to the WRITE state, asserts the write enable (write_en) and the acknowledge signal (ack) for one cycle. It then returns to the IDLE state, having completed the write operation in a single cycle. If it instead receives a read signal while in the IDLE state, it then goes to the READ1 state and asserts the output enable (out_en).



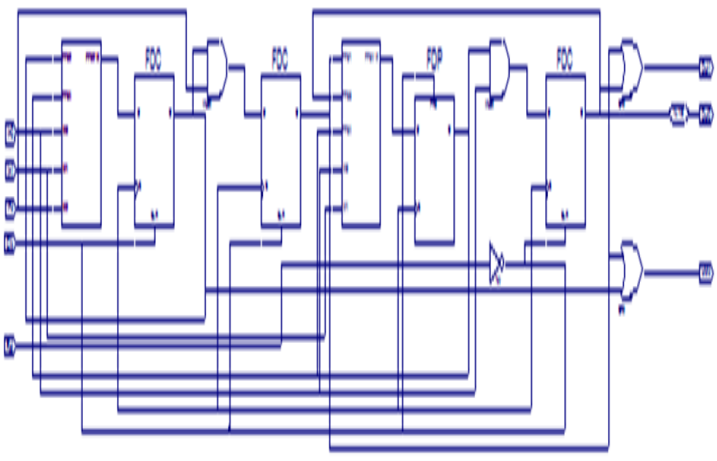


Fig.6: Pin diagram of simple state machine

Fig.7: Schematic diagram of simple state machine

It waits for the ready signal (ready) from the memory device, signaling that valid data is being output. It then goes to the READ2 state, keeping the output enable asserted, and giving acknowledge, before returning to the IDLE state. Note that for any condition other than ones explicitly shown, it is implied that the state machine remains in the current state. For example, in the READ1 state, if the ready signal is not asserted, the machine stays in the READ1 state.

TABLE 1: TIMING SUMMARY OF MOORE STATE MACHINE

Parameters	Seconds
Minimum period	3.675ns
Minimum input arrival time before clock	4.950ns
Maximum output required time after clock	8.669ns

The proposed state machine model is designed in HDL, using Xilinx 6.3i with SPARTAN 3 family (XC3S1500), FG320 package logically tested and then synthesized in XST synthesis tool.

TABLE 2: DEVICE UTILIZATION SUMMARY OF MOORE STATE MACHINE

Name	Used Blocks	Percentages (%)
Number of Slices	5 out of 192	2
Number of Slice Flip Flops	4 out of 384 (FDC:3,FDP:1)	1
Number of 4 input LUTs	9 out of 384	2
Number of bonded IOBs	7 out of 90 IBUF:6 OBUF:1	7
Number of GCLKs	1 out of 4	25

The timing summary, device utilization summary are specified in Table 1, Table 2 respectively. From the timing summary, it has been observed that the minimum period is 3.675 ns and the maximum output required time after clock is 8.669 ns. From the table 2, it is seen that 4 flip flops are used among 384.

In the schematic diagram from fig.8 and fig.9, the outputs of the each state determine that state due to having a unique set of outputs. This is an optimization of the Moore state machine. In this way, no flip-flops or extra logic are allocated to state variables, which results in more compact logic.

Fig.8: Pin diagram of Moore state machine with compact logic

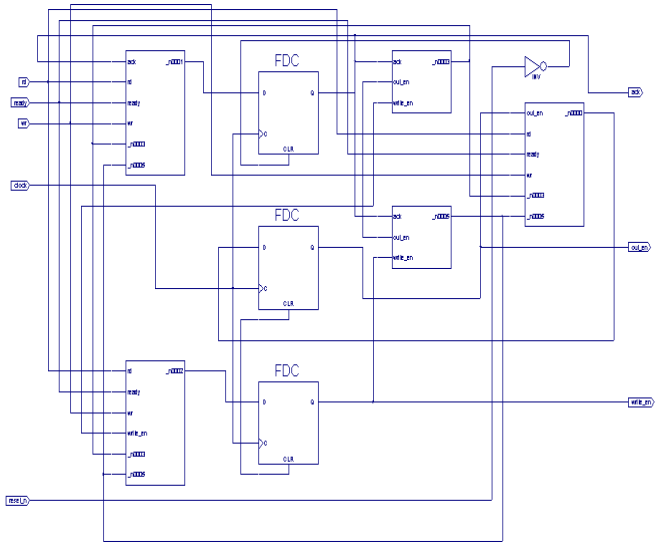


Fig.9: RTL Schematic of Moore state machine with compact logic

The timing summary, device utilization summary are specified in Table 3 and Table 4 respectively. The maximum delay is 4.432 ns.

b) Simulation Results

The timing diagram is achieved from Verilogger Test-bencher pro 6.5. The state machine simulation code (from fig.10) uses an on-location memory.

The simulation instantiates the memory controller state machine and performs a number of writes and reads to test that the memory is being written and read by the state machine. First, it checks that outputs are reset to the correct values after asserting the reset signal.

The simulation then writes a data value to memory, reads it back, and checks that the correct value was read. Reading the memory a second time it checks the value to ensure that the memory did not change after the previous read. This tests back-to-back read operations.

Finally, the simulation writes a value to memory and immediately writes a new value to memory. It then reads a value from memory and checks the second read back value. This tests back-to-back write operations.

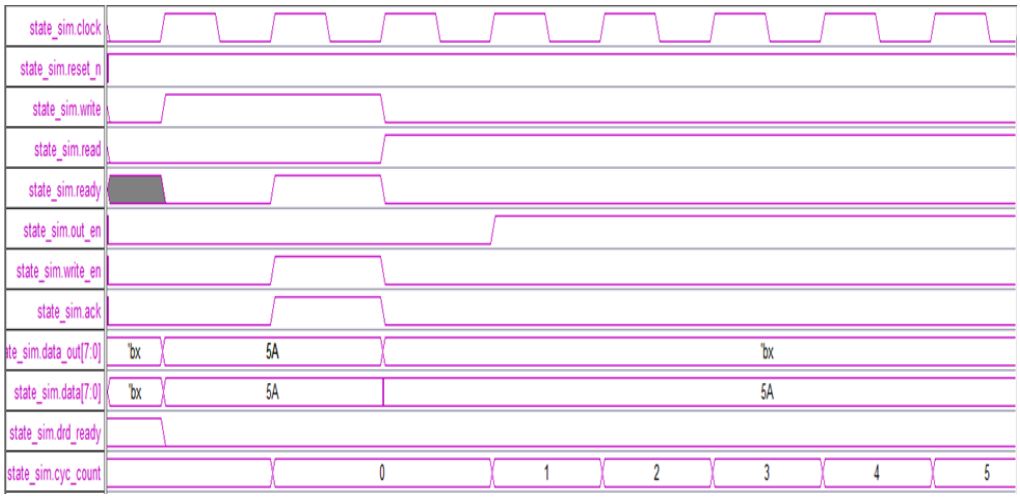


Fig.10: Timing diagram of Moore state machine

B. MEALY STATE MACHINE

a) Circuit Schematic

Fig.11 & Fig.12 show the pin diagram and RTL schematic diagram of Mealy state machine. The outputs are defined by the current state. The proposed state machine model is designed in HDL, using Xilinx 6.3i with SPARTAN 3 family (XC3S1500), FG320 package logically tested and then synthesized in XST synthesis tool. The timing summary, device utilization summary and HDL synthesis report are specified in Table 3, Table 4 respectively. From the timing summary, the maximum output required time after clock is 7.319 ns. From the table 2 it is seen that 5 flip flops are used among 384.

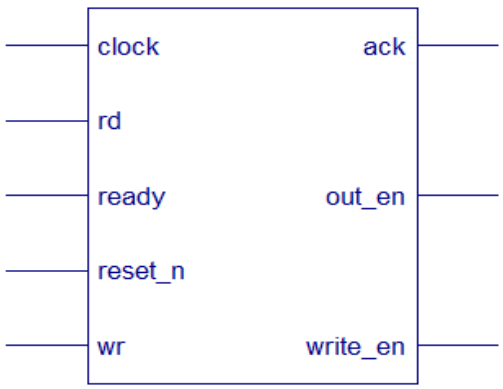


Fig.11: Pin Diagram of Mealy state machine

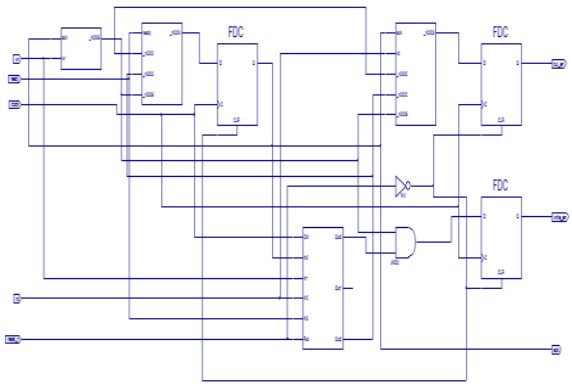


Fig.12: RTL schematic of Mealy state machine

TABLE 3: TIMING SUMMARY OF MEALY STATE MACHINE

Parameters	Seconds
Minimum period	4.588ns
Minimum input arrival time before clock	5.058ns
Maximum output required time after clock	7.319ns
Maximum delay	4.588 ns

TABLE 4: DEVICE UTILIZATION SUMMARY OF MEALY STATE MACHINE

Name	Used Blocks	Percentages (%)
Number of Slices	5 out of 192	2
Number of Slice Flip Flops	5 out of 384 (FDC:4 FDP:1)	1
Number of 4 input LUTs	9 out of 384	2
Number of bonded IOBs	7 out of 90 (IBUF:4 BUF:3)	7
Number of GCLKs	1 out of 4	25

The timing diagram is achieved from Verilogger Test-bencher pro 6.5.Fig.11 shows the timing diagram of Mealy state machine (fig, 13).

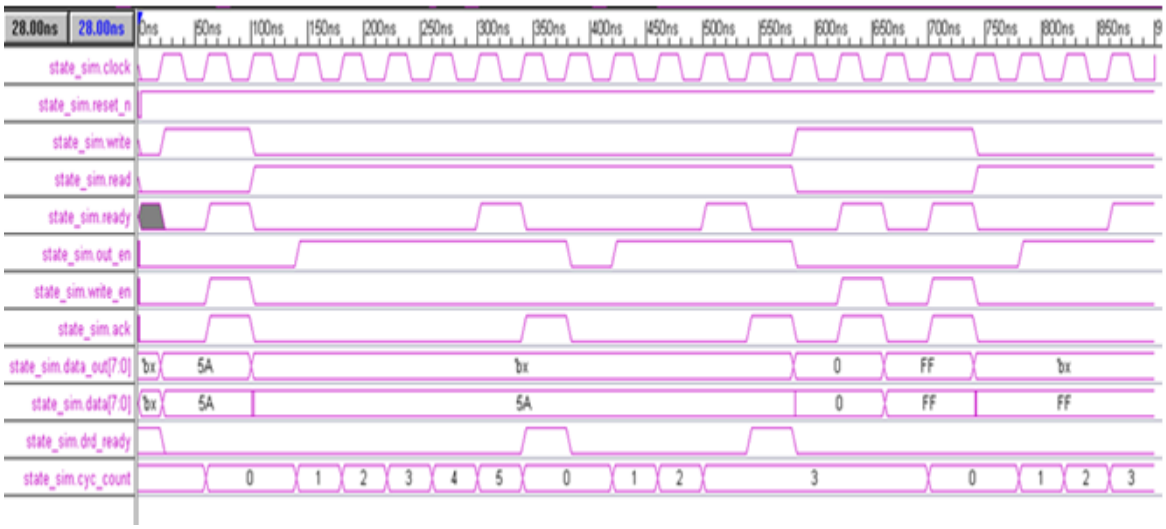


Fig.13: Timing diagram of Mealy State machine

VI. CONCLUSION

There is greater necessity for synthesis optimization for proper resource utilization and logic that are embedded in the targeted device in FPGA. The focus has been to minimize the active area, speed and resources in a FPGA target device. Moore and Mealy state machine have been designed here using the Xilinx ISE software. RTL code has been adopted here. Timing summary (Minimum period, maximum delay etc.) and Device utilization summary have also been discussed. Timing diagram of Moore and Mealy state machines have been observed using Verilogger Test bench pro 6.5.

REFERENCES

[1] R. Uma, "Qualitative Analysis of Hardware Description Languages: VHDL and Verilog" (IJCSIS), International Journal of Computer Science and Information Security, Vol. 9, No. 4, pp-127-135, April 2011.

[2] Alberto Sangiovanni, Vincentelli, Abbas El Gamal and Jonathan Rose, "Synthesis Methods for Field Programmable Gate Arrays" Proceedings of the IEEE, VOL. 81, NO. 7, July, 1993.

[3] Maico Cassel, Fernanda Lima Kastensmidt "Evaluating One-Hot Encoding Finite State Machines for SEU Reliability in SRAM-based FPGAs" Proceedings of the 12th IEEE International On-Line Testing Symposium (IOLTS'06).

[4] Bharat, K.; Broder, A. (1998): A technique for measuring the relative size and overlap of public Web search engines. Computer Networks, 30(1–7), pp. 107–117.

[5] Stephen D Brown, "Fundamentals of digital logic with Verilog design", Tata McGraw-Hill Education, 2007.

[6] Zainalabedin Navabi, "Digital System Test and Testable Design: Using HDL Models and Architectures", Springer Science & Business Media, Dec 10, 2010.

[7] Wai-Kai Chen, "Design Automation, Languages, and Simulations", CRC Press, Mar 26, 2003.

[8] William J. Eccles, "Pragmatic Logic", Morgan & Claypool Publishers, Jun 1, 2007.

[9] Joseph Cavanagh, "Verilog HDL: Digital Design and Modeling", CRC Press, Feb 20, 2007.

[10] Bob Zeidman, "Verilog Designers library".

[11] Zainalabedin Navabi, "Verilog Digital System Design: Register Transfer Level Synthesis, Testbench, and Verification: Register Transfer Level Synthesis, Testbench, and Verification", McGraw Hill Professional, Oct 3, 2005.