APPLICATION OF FINITE AUTOMATA IN DIGITAL CIRCUIT DESIGN

16010421119 – AARYA TIWARI , 16010421120 – SUMIT TIWARI , 16010421121 – VIRINCHI TONGE

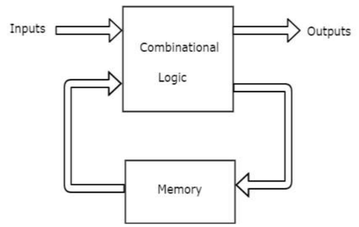
SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Somaiya TRUST

INTRODUCTION TO FINITE MACHINES

MEALY STATE MACHINE

A Finite State Machine is said to be Mealy state machine, if outputs depend on both present inputs & present states.

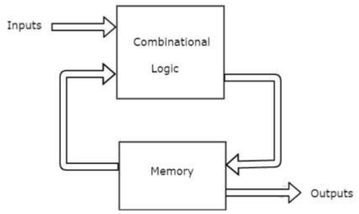


Those are combinational logic and memory. Memory is useful to provide some or part of previous outputs present states as inputs of combinational logic.

So, based on the present inputs and present states, the Mealy state machine produces outputs. Therefore, the outputs will be valid only at positive or negative
transition of the clock signal.

MOORE STATE MACHINE

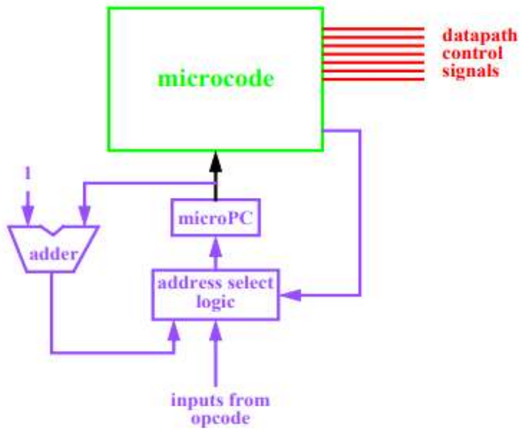
A Finite State Machine is said to be Moore state machine, if outputs depend only on present states.



there are two parts present in Moore state machine. Those are combinational logic and memory. In this case, the present inputs and present states determine the next states. So, based on next states, Moore state machine produces the outputs. Therefore, the outputs will be valid only after transition of the state.

APPLICATIONS OF FINITE AUTOMATA IN DCD

1. CONTROL UNIT DESIGN

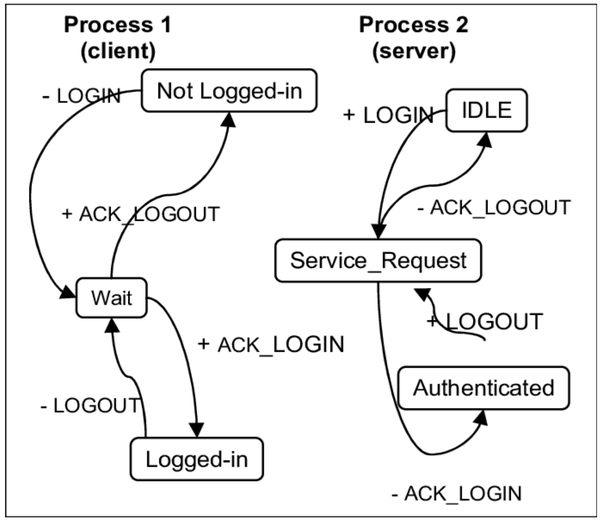


Control unit design using finite automata is a method of designing the control unit of a computer using finite state machines. Finite state machines are mathematical models that can be used to represent a system that can be in a finite number of states and can transition between states based on a set of inputs.

The finite state machine has four states: FETCH, DECODE, EXECUTE, and WAIT. The control unit would start in the FETCH state. When the control unit receives a signal from the processor that a new instruction is available, it would transition to the DECODE state. In the DECODE state, the control unit would decode the instruction and determine which instruction it is. Based on the instruction, the control unit would transition to the EXECUTE state. In the EXECUTE state, the control unit would execute the instruction. Once the instruction has been executed, the control unit would transition back to the FETCH state and start the process over again.

APPLICATIONS OF FINITE AUTOMATA IN DCD

2. PROTOCOL DESIGN AND VERIFICATION



Protocol design and verification using finite automata is a method of designing and verifying communication protocols using finite state machines. Finite state machines are mathematical models that can be used to represent a system that can be in a finite number of states and can transition between states based on a set of inputs.

Communication protocols are sets of rules that govern how devices communicate with each other over a network. Protocol design and verification using finite automata can be used to ensure that communication protocols are correct and that they will work as expected.

The finite state machine represents a simple three-way handshake protocol. The protocol starts in the CONNECT state. When the sender sends a SYN packet, the receiver transitions to the SYN+ACK state. When the sender receives the SYN+ACK packet, it transitions to the ESTABLISHED state. Once both devices are in the ESTABLISHED state, they can start sending data packets to each other.

SOME OTHER APPLICATIONS

State Machines: Finite state machines (FSMs) are a type of finite automaton used extensively in digital circuit design. FSMs are used to model and control the behavior of sequential circuits, such as counters, memory controllers, and communication protocols. They help ensure proper sequencing and synchronization of operations within the circuit.

Hardware Acceleration: Finite automata can be implemented in hardware to accelerate certain computations. For example, they can be used to perform tasks like string searching, text compression, and protocol parsing efficiently in networking hardware.

Pattern Recognition: Finite automata can be employed in digital circuitry for pattern recognition tasks, such as detecting specific sequences of bits or characters. This can be useful in applications like error detection and correction, data compression, and regular expression matching.

CONCLUSION

From this poster, we have learned about the diverse applications of finite automata in digital circuit design, ranging from control unit optimization to advanced pattern recognition. This knowledge empowers us to harness the potential of finite automata for creating more efficient and innovative digital circuits.

REFERENCES

Computer Architecture: A Quantitative Approach by John L. Hennessy and David A. Patterson, Chapter 5.

Digital Computer Electronics by Morris Mano and Charles R. Kime, Chapter 6.

Computer Organization and Design: The Hardware/Software Interface by David A. Patterson and John L. Hennessy, Chapter 3.

Computer Systems: A Programmer's Perspective by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne, Chapter 5.