

Finite State Machines

Introduction

- State Machines are an integral part of software programming.
- State machines make code more efficient, easier to debug and help organize the program flow.
- State machines are not limited to just firmware, they can be used to streamline any system.

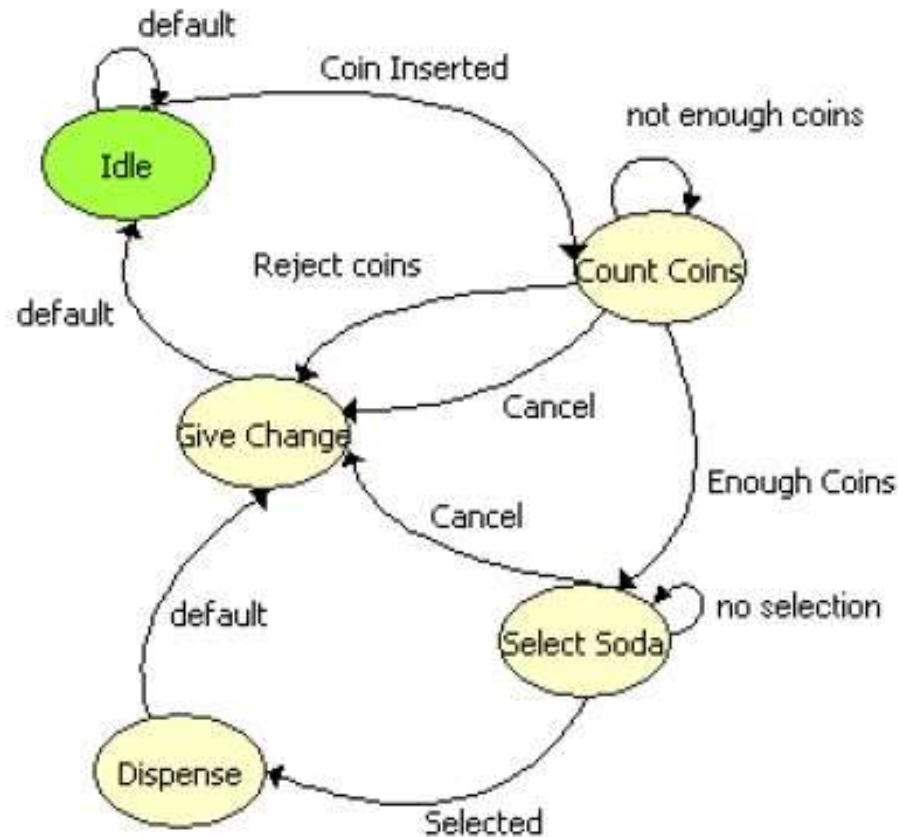
State variable

- State machines require a State Variable (SV).
- The SV is essentially a pointer that keeps track of the state that the microcontroller is in, and directs the program flow to the corresponding software module.
- The SV can be modified in the software modules (or states) themselves or by an outside function.
- The example firmware uses an outside function which detects a button press to advance through the states.

Benefits

- The first advantage is using state machines inherently promotes good design techniques.
- When beginning to implement an application, think about what states are necessary to make the application work. List all the pieces, or states, of an application and then explore how they tie to one another. This will help prevent developing bugs in the code.
- This line of thinking also leads to the development of a very useful engineering tool – the flow chart.

State Diagram of a Simple Soda Vending Machine



What is a FSM

- A Finite State Machine (FSM) is based on the idea of there being finite number of states for a given system.
- For instance, when an application turns an LED on and off, two states exist; one state is when the LED is on and the other is when it is off.
- The example will turn on eight LEDs sequentially.
- Only one LED is on at a time, therefore eight states exist.
- Each state consists of one LED being turned on while all the rest are off.

What is a FSM?

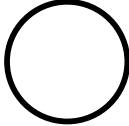
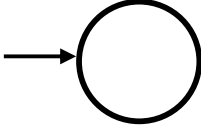
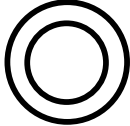
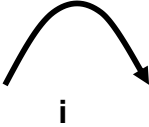

- A theoretical tool used by computer scientists
- Attempts to describe a protocol or algorithm
- Virtual machine that progresses through a series of stages of operation in response to various events
- State – status of a algorithm
- Transition – act of moving from one state to another
- Event – something that causes a transition to occur between states
- Action – response to an event before a transition

Why FSM are useful?

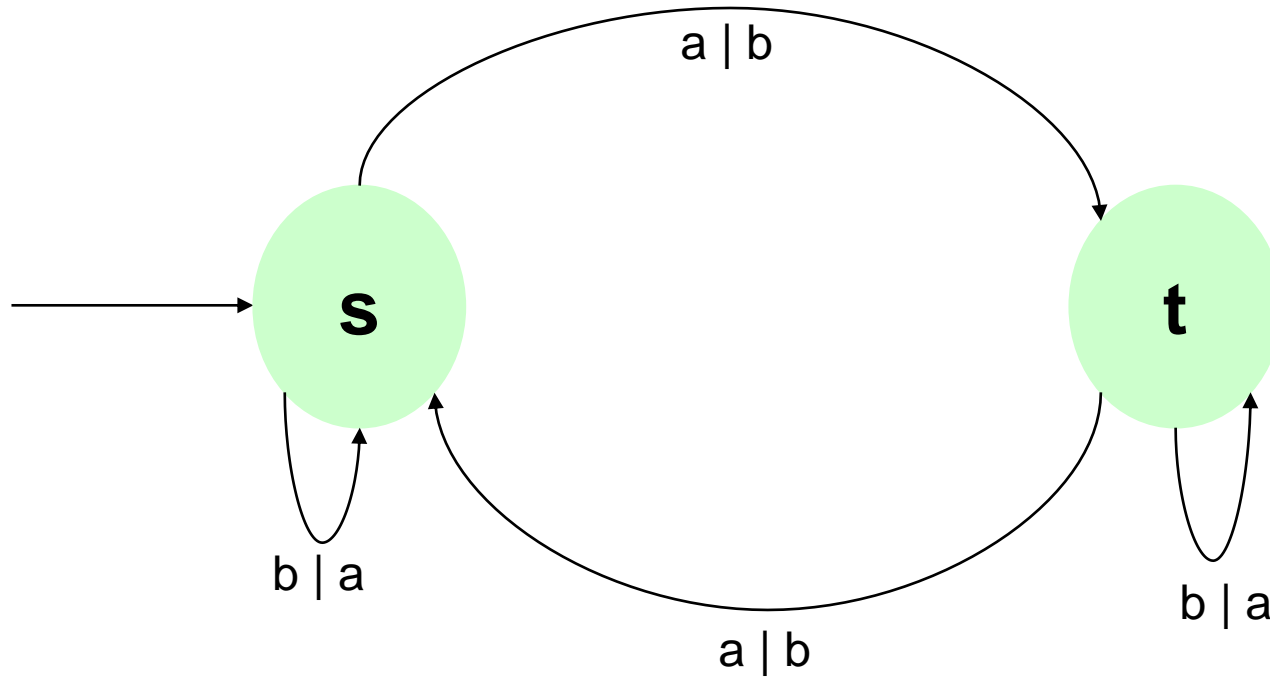
- Modelling and designing
 - Vending machines
 - Lifts
 - Traffic light controllers
- Specifying a language
- Programs for
 - Spell checking
 - Grammar checking
 - Processing HTML and XML

State transition diagrams

A state transition diagram represents a FSM

| Symbol | Meaning |
|---|---|
|  | State |
|  | Starting state – this usually has an arrow going towards it |
|  | Accept (or end) state |
|  | Connector (where i is the input) |
|  | Connector (where i is the input and k is the output) |

State transition diagram for an FSM with outputs



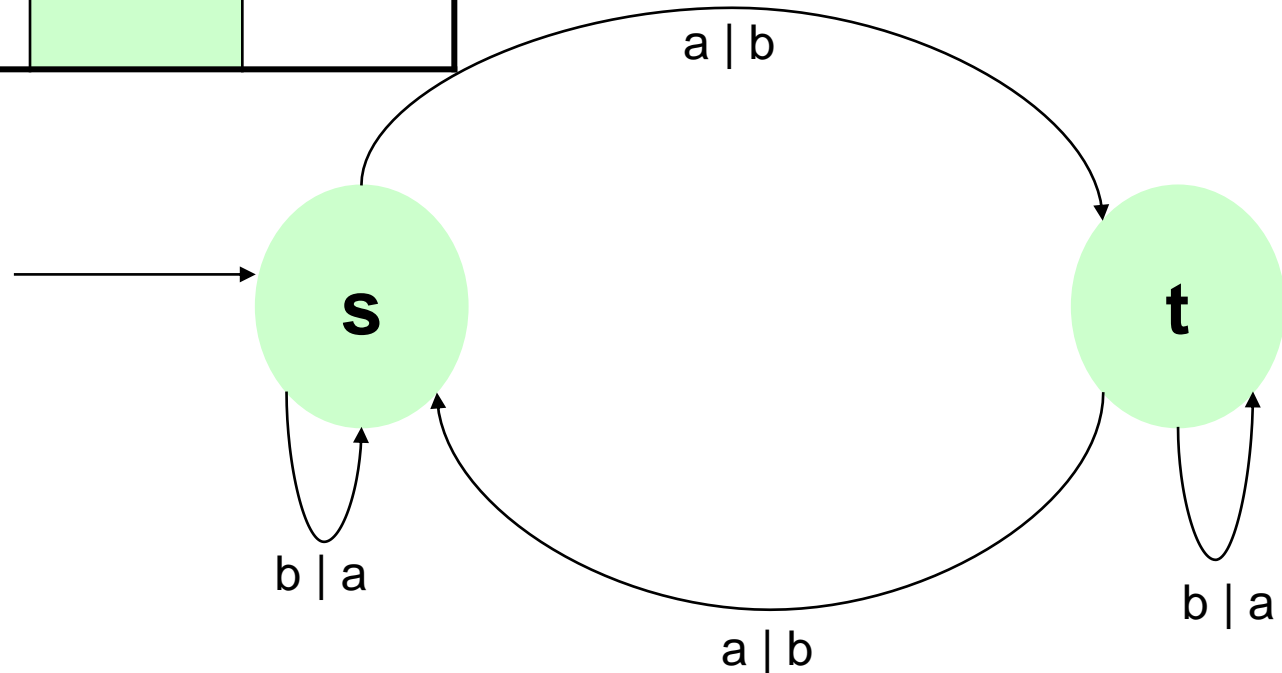
Transition table for the FSM

| | | | | |
|---------------|---|---|---|---|
| Current state | s | s | t | t |
| Input symbol | a | b | a | b |
| Next state | t | s | s | t |
| Output symbol | b | a | b | a |

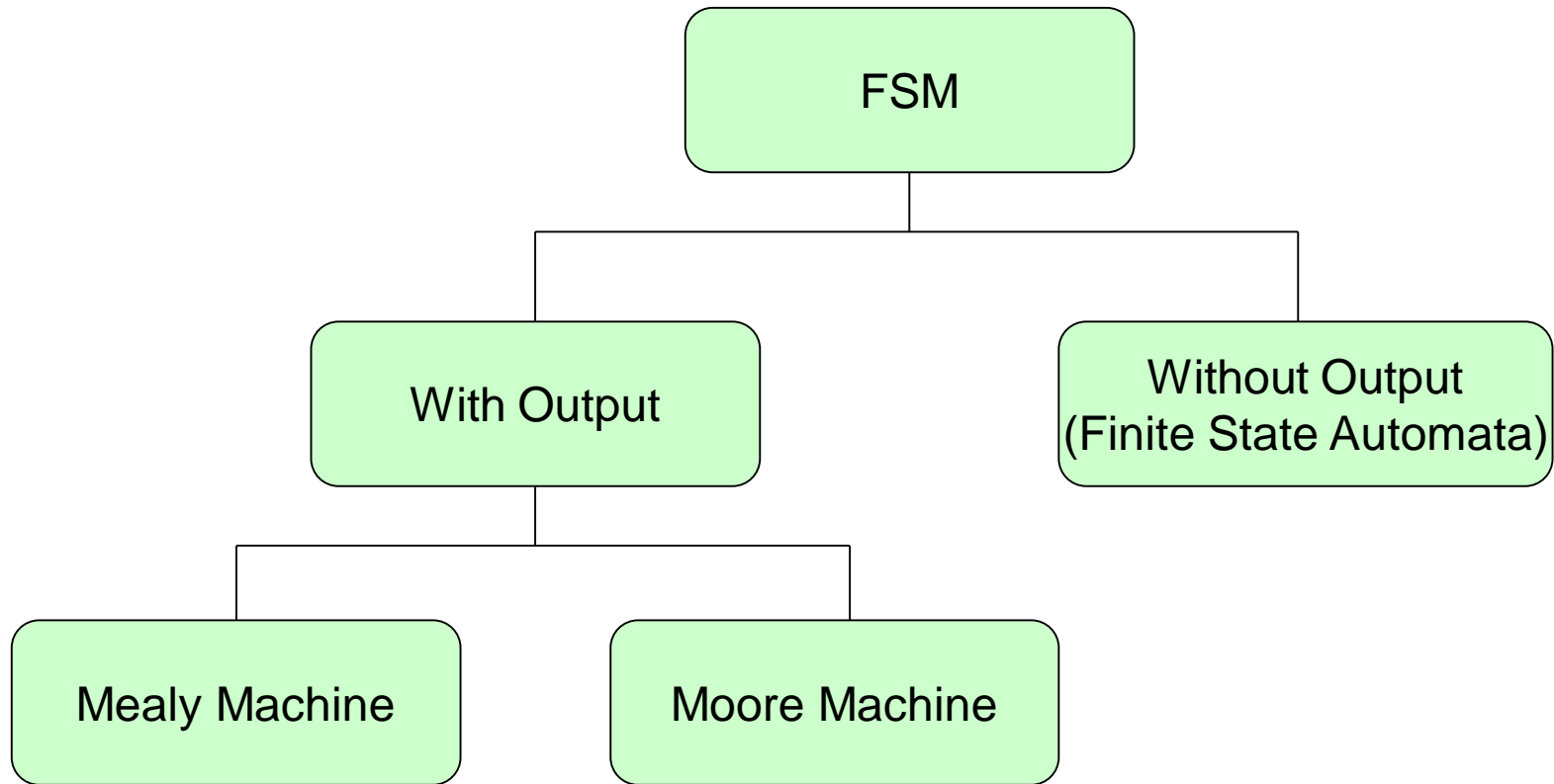
The FSM has a **transition function** that maps a state-symbol pair to a state and may generate an output.

This table represents the transition function.

What does this FSM do?

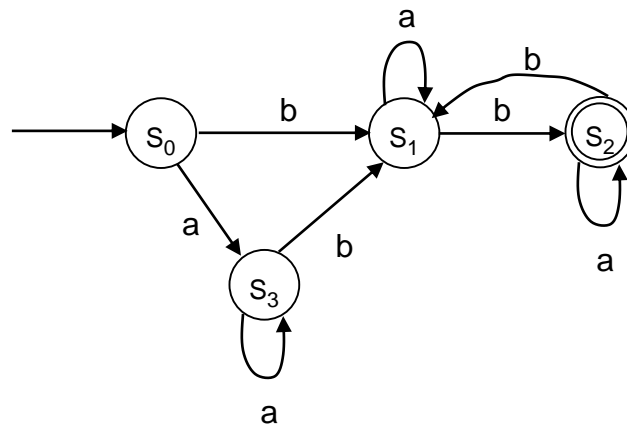


Types of FSM



Finite State Automata

- FSM without output
- Restricted to decision problems
- Problems that are solved by answering Yes or No
- e.g. language syntax, parity operations



FSMs with output

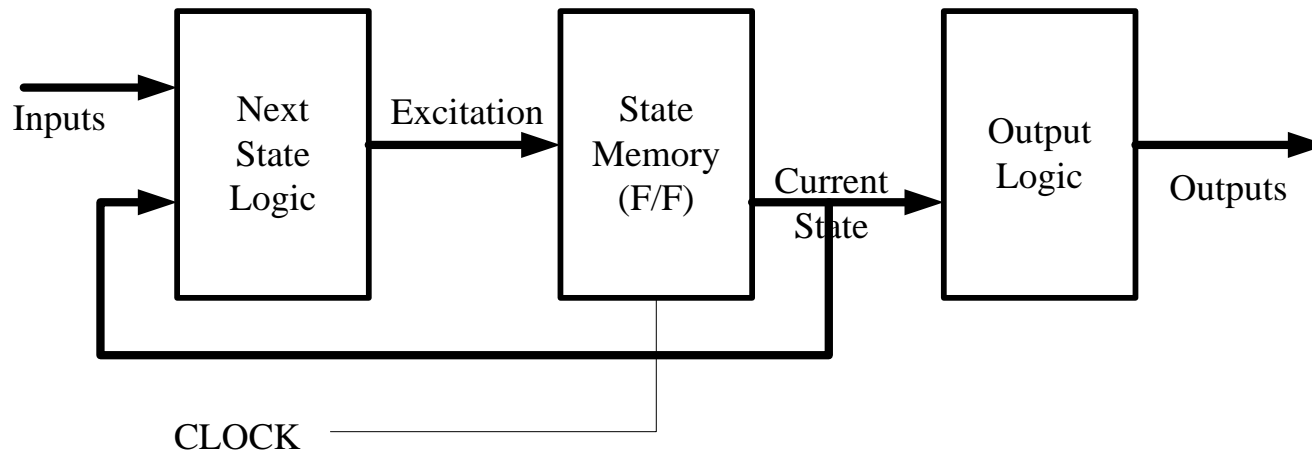
- Most common controller of machines
- E.g. lifts, traffic lights, washing machines
- Mealy machines
- Moore machines

Moore machine

- A FSM that determines its outputs from the present state only
- The current state of a Moore FSM is determined by the sequence of inputs that it has processed
- Any electronic circuit with a memory is a Moore FSM

State Machine Types

- Moore machine



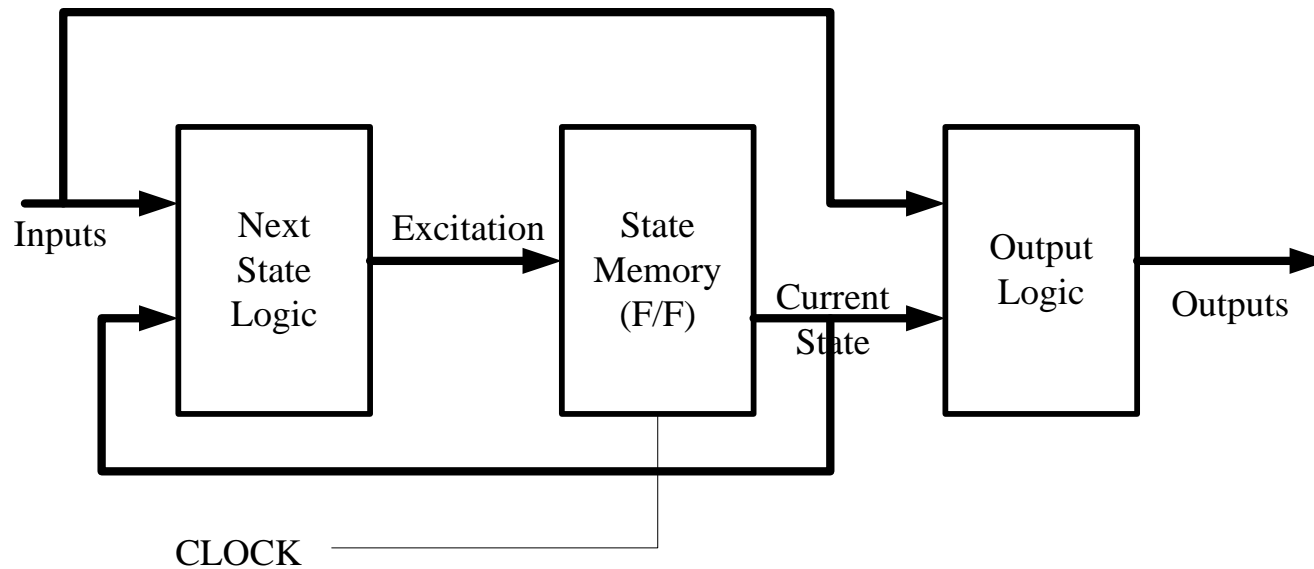
- Characterized by – Outputs are a function current state only

Mealy machines

- A FSM that determines its outputs from the present state and from the inputs

Types of state machines

- Mealy Machine



- Characterized by – Outputs are a function of both inputs and current state

Non-deterministic FSMs

- For some states there is more than one transition labelled with the same trigger
- It is an FSM with several possible next states for each pair of input symbol and state
- E.g. searching documents

