CS221: Digital Design

Sequential Design Based on Mealy FSM

A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- FSM Types : Mealy Machine, Moore Machine
  - Till now we studied Moore Machine
- Example of Mealy Machine
- Comparison of Mealy Vs Moore FSM

FSM Types

- Two main types of FSMs
  - Moore FSM : output is only function of state
  - Mealy FSM: output is function of state and inputs

Set Theoretic Description

Moore Machine is an ordered quintuple

$$\text{Moore} = (\mathbf{S}, \mathbf{I}, \mathbf{O}, \delta, \lambda)$$

where

$\mathbf{S}$ = Finite set of states $\neq \Phi, \{s_1, s_2, \cdots, s_n\}$

$\mathbf{I}$ = Finite set of inputs $\neq \Phi, \{i_1, i_2, \cdots, i_m\}$

$\mathbf{O}$ = Finite set of outputs $\neq \Phi, \{o_1, o_2, \cdots, o_l\}$

δ = Next state function which maps $\mathbf{S} \times \mathbf{I} \rightarrow \mathbf{S}$

λ = Output function which maps $\mathbf{S} \rightarrow \mathbf{O}$

Set Theoretic Description

Mealy Machine is an ordered quintuple

$$\text{Mealy} = (\mathbf{S}, \mathbf{I}, \mathbf{O}, \delta, \beta)$$

where

$$\mathbf{S} = \text{Finite set of states} \neq \Phi, \{s_1, s_2, \cdots, s_n\}$$

$$\mathbf{I} = \text{Finite set of inputs} \neq \Phi, \{i_1, i_2, \cdots, i_m\}$$

$$\mathbf{O} = \text{Finite set of outputs} \neq \Phi, \{o_1, o_2, \cdots, o_l\}$$

$$\delta = \text{Next state function which maps} \quad \mathbf{S} \times \mathbf{I} \rightarrow \mathbf{S}$$

$$\beta = \text{Output function which maps} \quad \mathbf{S} \times \mathbf{I} \rightarrow \mathbf{O}$$

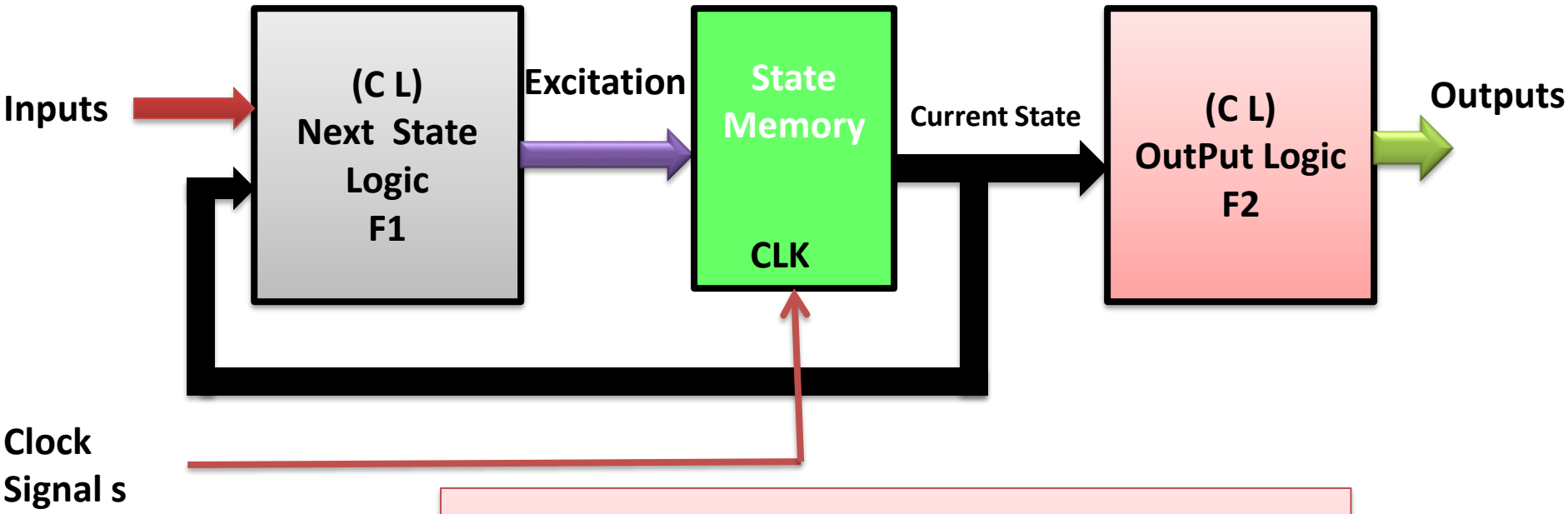
Clocked synchronous FSM

- **Clocked**
  - All storage elements employ a clock input (i.e. all storage elements are flip-flops)
- **Synchronous**
  - All of the flip flops use the same clock signal

Clocked synchronous FSM

- **FSM**
  - State machine is simply another name for sequential circuits. Finite refers to the fact that the number of states the circuit can assume if finite
- **Async FSM:** A synchronous clocked FSM changes state only when a triggering edge (or tick) occurs on the clock signal

Moore machine

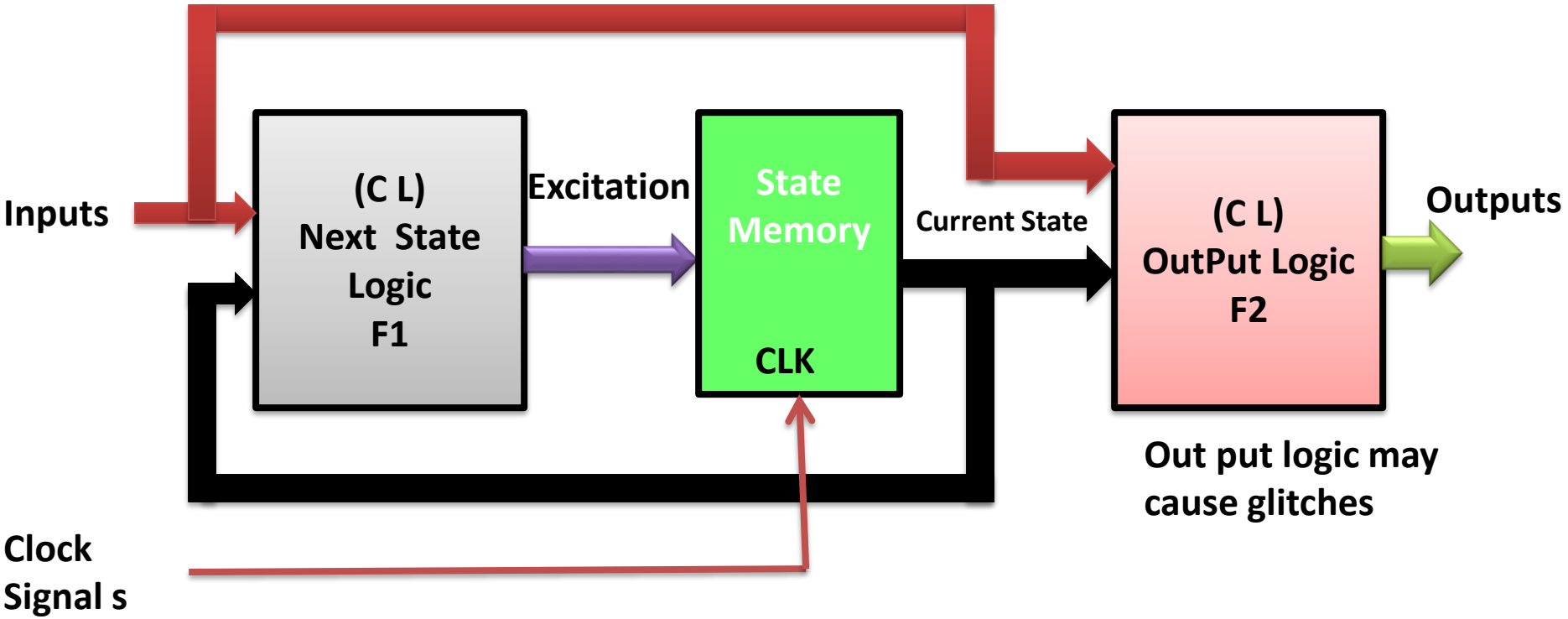


$$\text{Next state} = F_1(\text{current state, inputs})$$

$$\text{Output} = F_2(\text{current state})$$

Mealy machine

- A Much better name of State “Memory” is State Storage



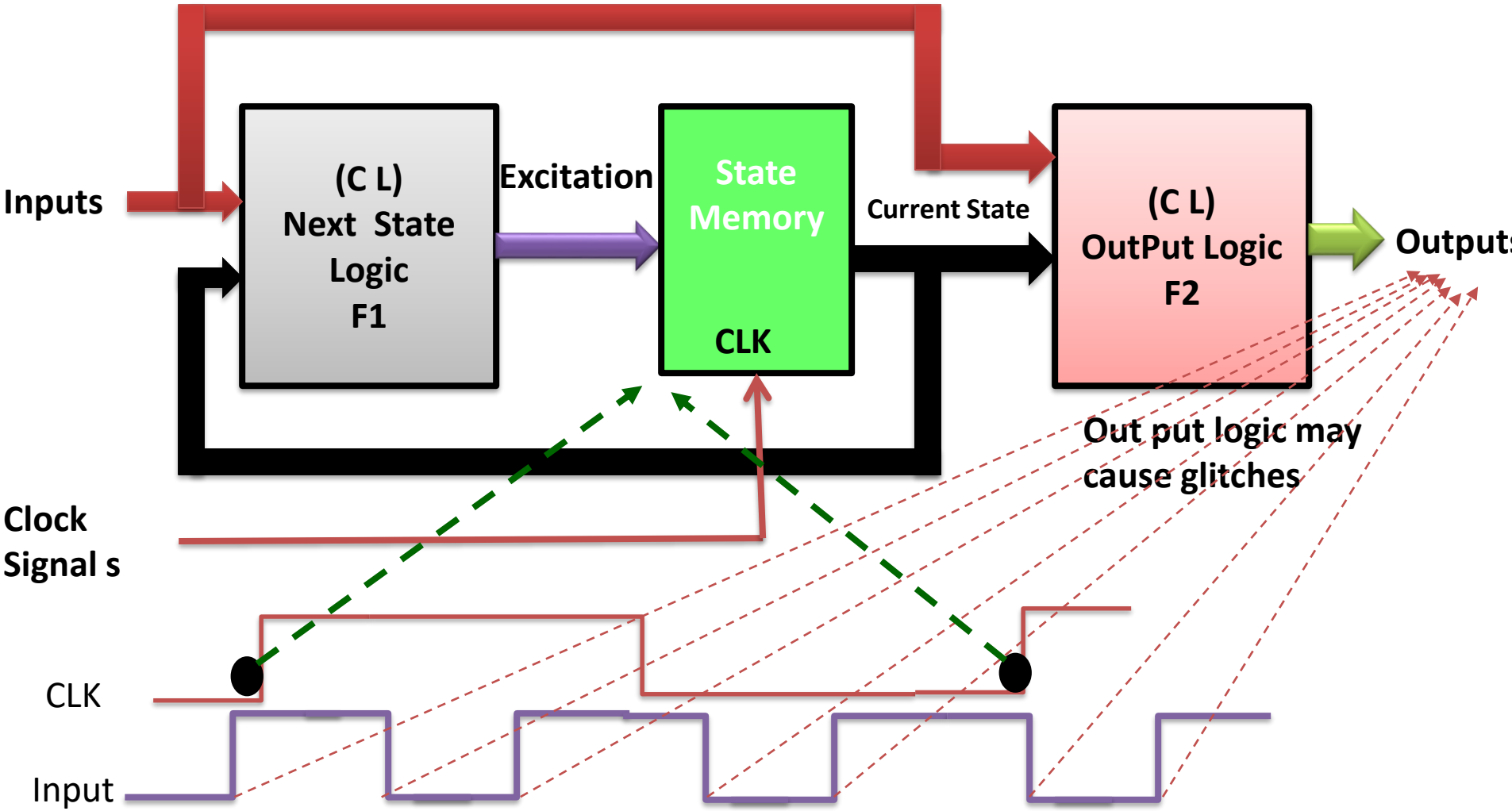
State Storage= Set of n FFs
2^n State can be stored

Next state = F_1(current state, inputs)

Output = F_2(current state, inputs)

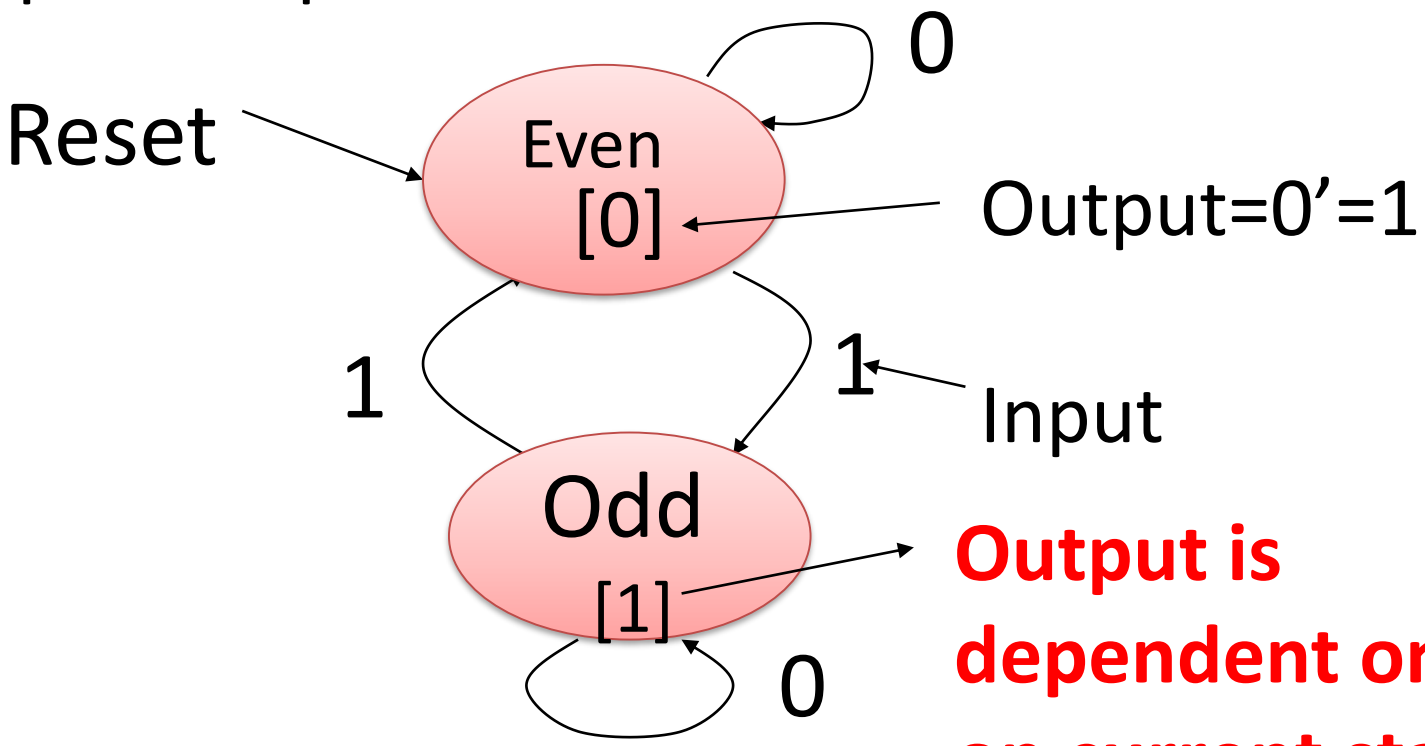
Mealy machine

- State change VS Output change: State at Clk^, Output change whenever input changes



Moore FSM : Parity Encoder

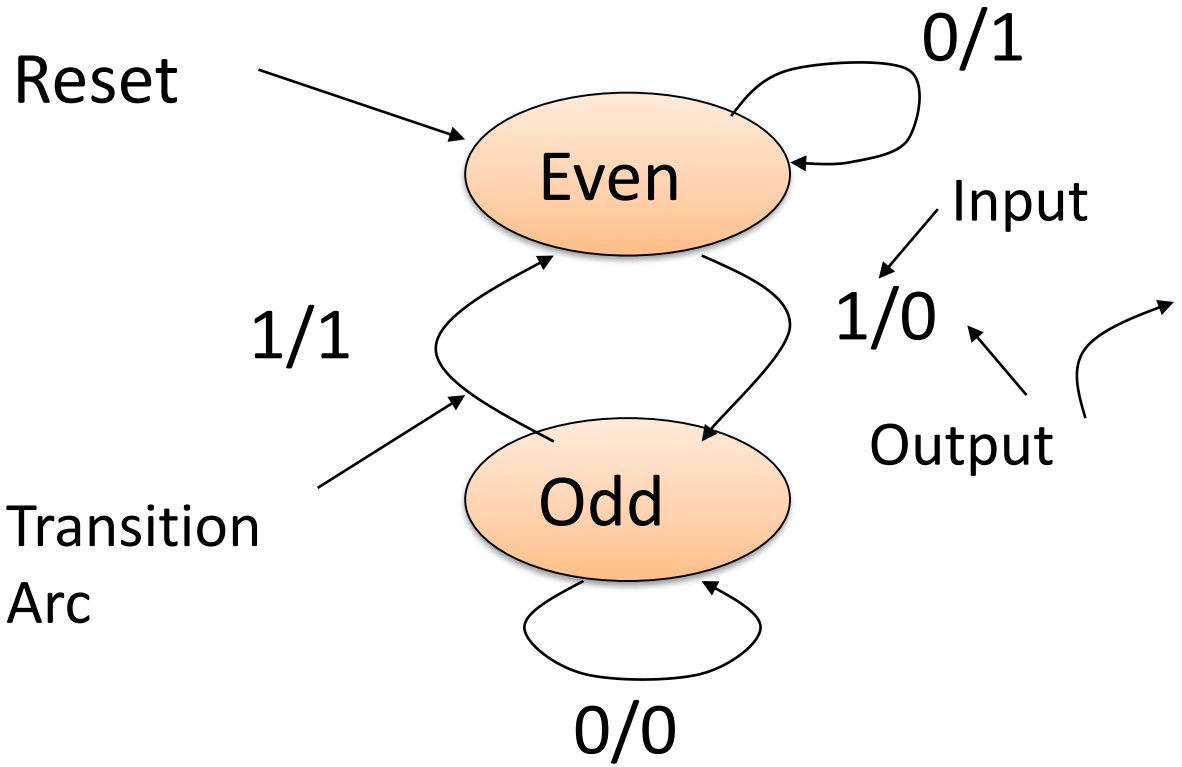
- Input: 1 or 0 // entering as stream
- Out put: output a 1 when total number of 1 is even



Output is associated with the state and hence appears after the state transition take place.

Example of Moore & Mealy Machine

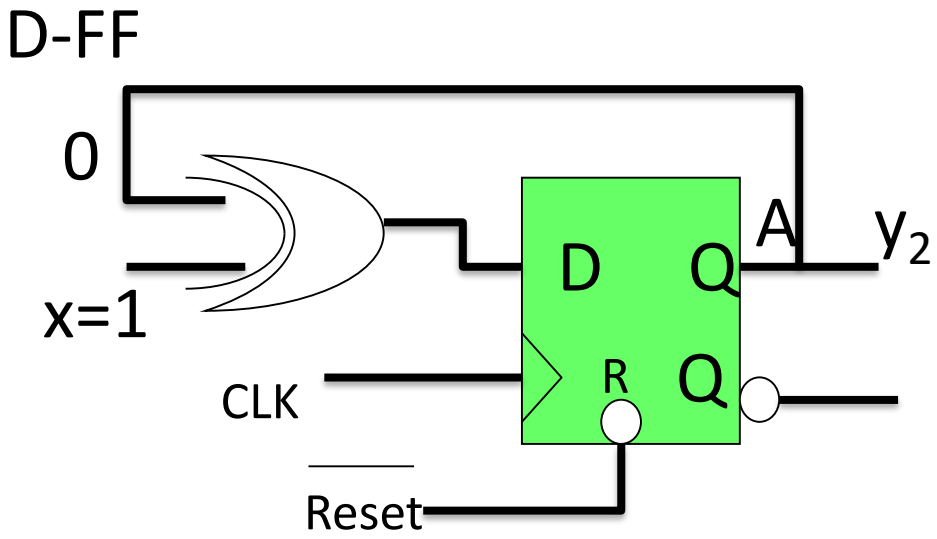
- Parity Checker Solution : (Mealy)

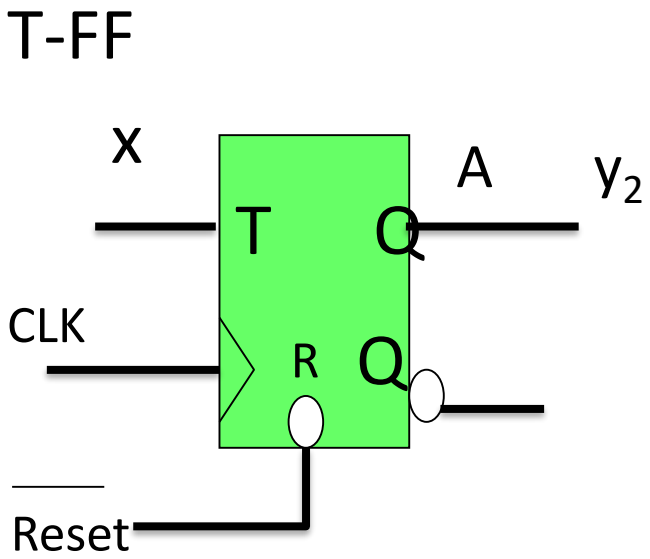


O/P is dependent on current state and input in Mealy

Mealy Machine: Output is associated with the state transition, and appears before the state transition is completed (by the next clock pulse).

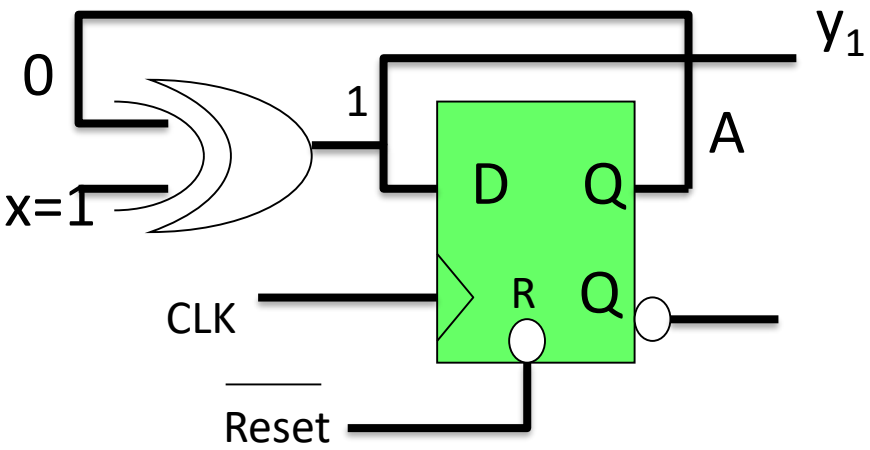
Parity Checker: Moore M/C Implementation



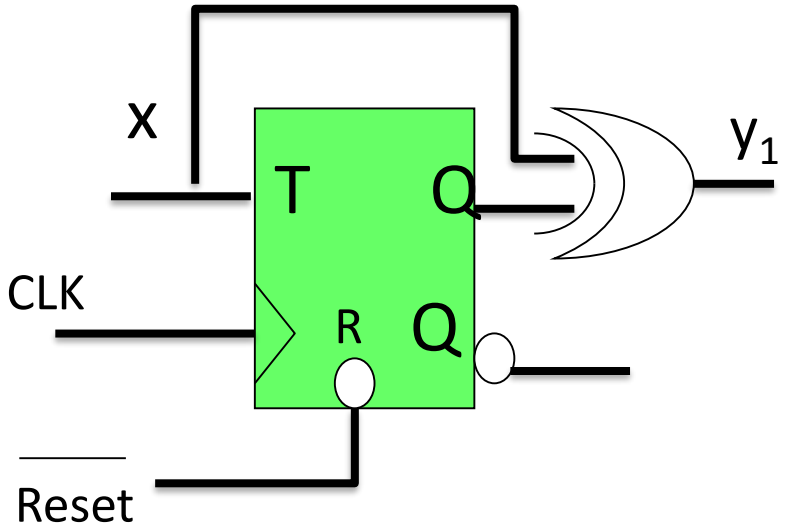


Moore O/P is synchronized with clock.

Parity Checker: Mealy M/C Implementation



D-FF



T-FF

Mealy O/P is not synchronized with clock.

Comparison : Mealy FSM Vs Moore FSM

- **Mealy** machines have **less states**
  - Outputs are on transitions (SxI) rather than states (n)
- **Moore** machines are **safer** to use
  - Outputs change at clock edge (always one cycle later)
- Mealy machines: input change can cause output change as soon as logic is done
  - A **big problem** when two machines are interconnected
  - Asynchronous feedback may occur if one isn't careful

Comparison : Mealy FSM Vs Moore FSM

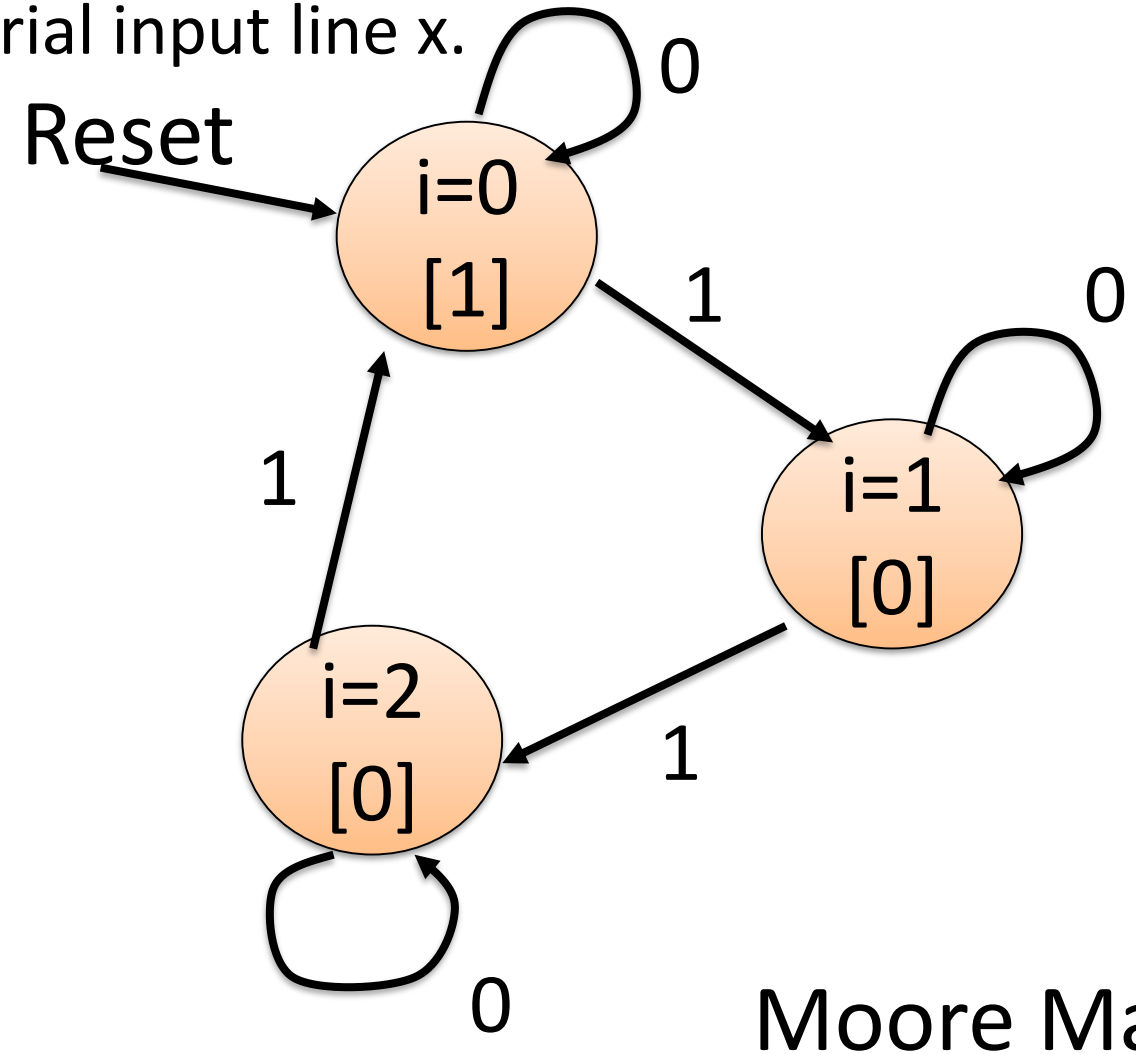
- **Mealy** machines **react faster** to inputs
  - react in same cycle – don't need to wait for clock
  - outputs may be considerably shorter than the clock cycle
  - in Moore machines, more logic may be necessary to decode state into outputs – there may be more gate delays after clock edge

FSM Example : Mealy vs Moore

- Design a system that outputs a ‘1’ whenever it receives a multiple of 3 # of 1’s (i.e., 0, 3, 6, 9, etc. # of 1’s) on a serial input line x.

FSM Example: Moore Machine

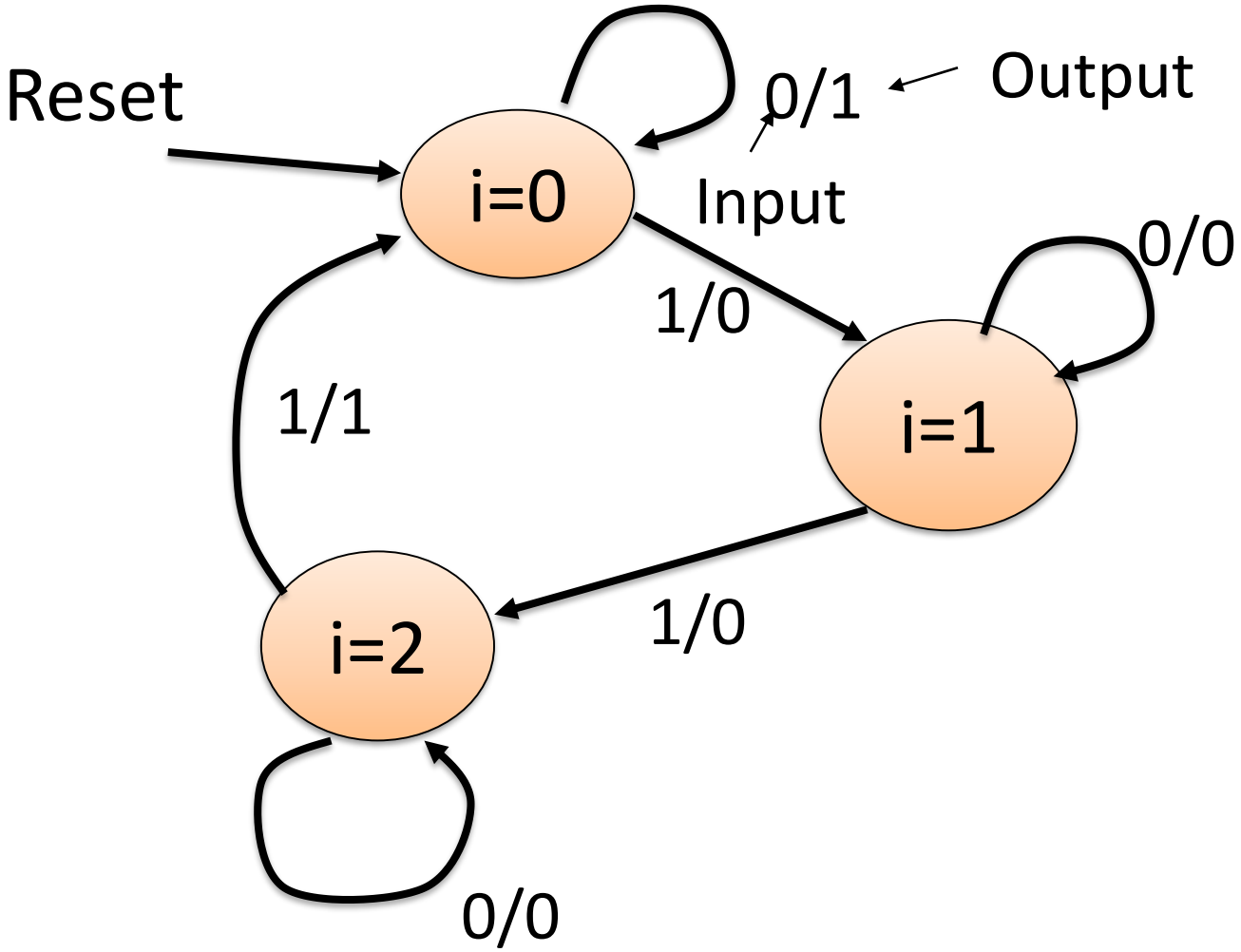
- Design a system that outputs a '1' whenever it receives a multiple of 3 # of 1's (i.e., 0, 3, 6, 9, etc. # of 1's) on a serial input line x.



Moore Machine

FSM Example: Moore Machine

- Design a system that outputs a '1' whenever it receives a multiple of 3 # of 1's (i.e., 0, 3, 6, 9, etc. # of 1's) on a serial input line x.



Mealy Machine

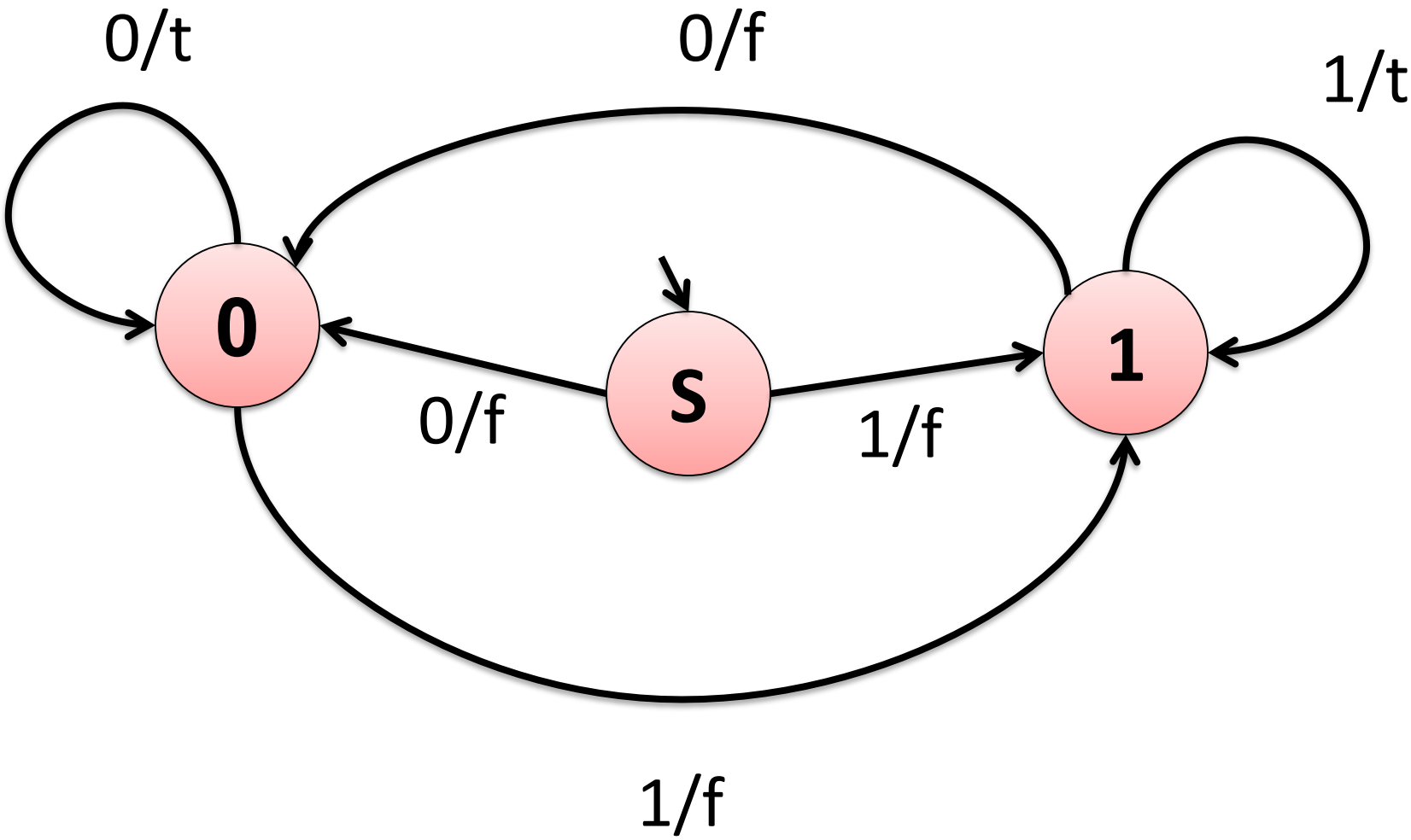
Mealy Vs Moore : Problem

- $\Sigma_O = \{t, f\}$ $\Sigma_I = \{0, 1\}$
- Outputs t $\Leftrightarrow$ at least 2 last input digits are same

Solutions to Problem

- Mealy implementation
  - –Needs only 3 states
- Moore implementation
  - –Needs 2 more almost duplicated states just in order to the output value t there.

FSM: Mealy Machine



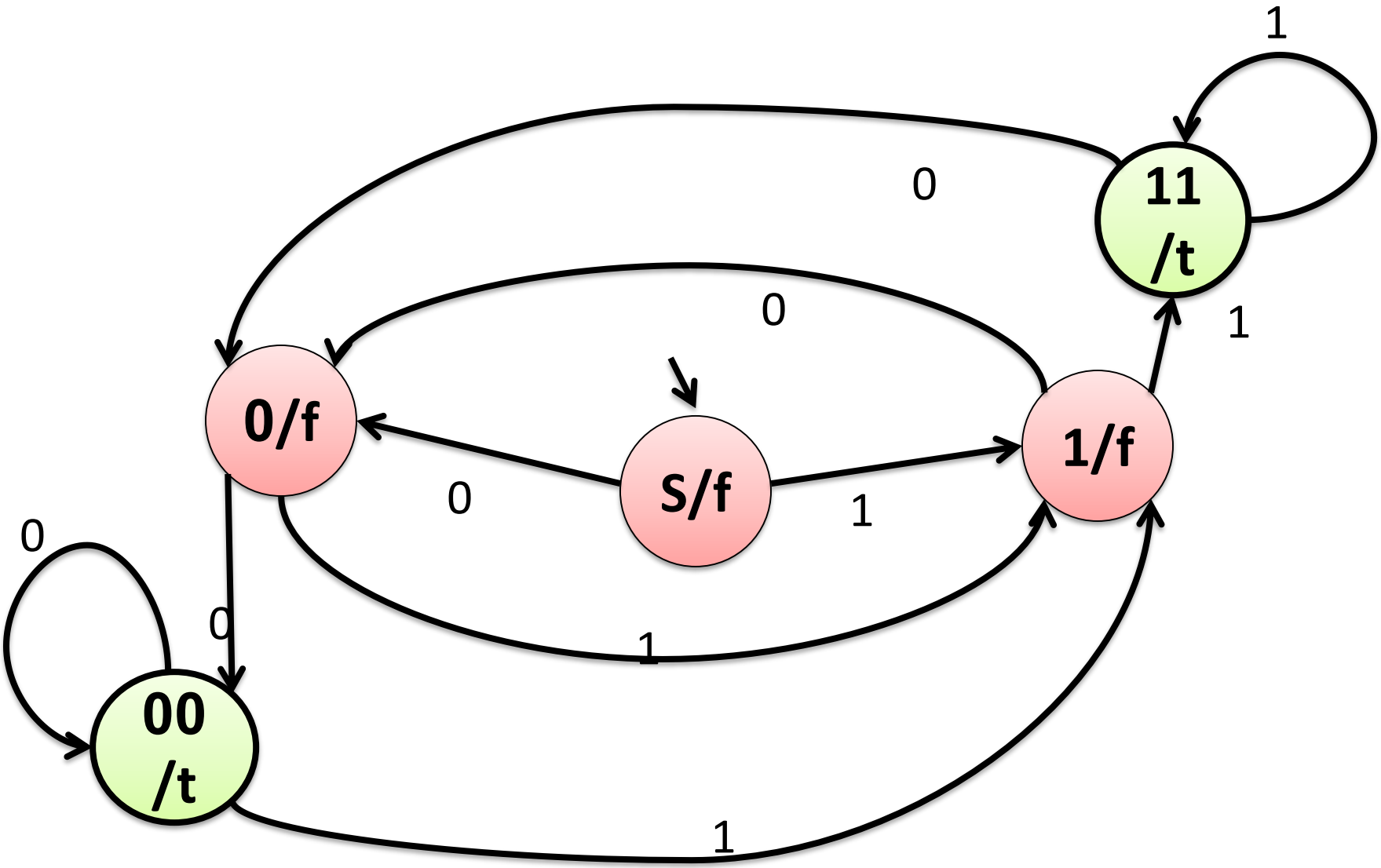
FSM: Moore Machine

Outputs **true** if at least 2 last input digits are same

Is it possible to design Moore FSM of the same problem with 3 state ?

You can try now..

FSM: Moore Machine



Next Class: FSM Optimization

- How to reduce the number of states in a FSM if it is specified with redundant states.

- Both type of FSM will be used
  - Some Examples are with Moore FSM
  - Some Examples are with Mealy FSM