CS221: Digital Design

Finite State Machine

A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

Outline

- Finite State Machine
- Formal definition
- FSM implementation
- FSM Examples

Combinational Circuit

- Can you formally model all the combinational circuit using Boolean Algebra?

YES

Sequential Circuit: FSM

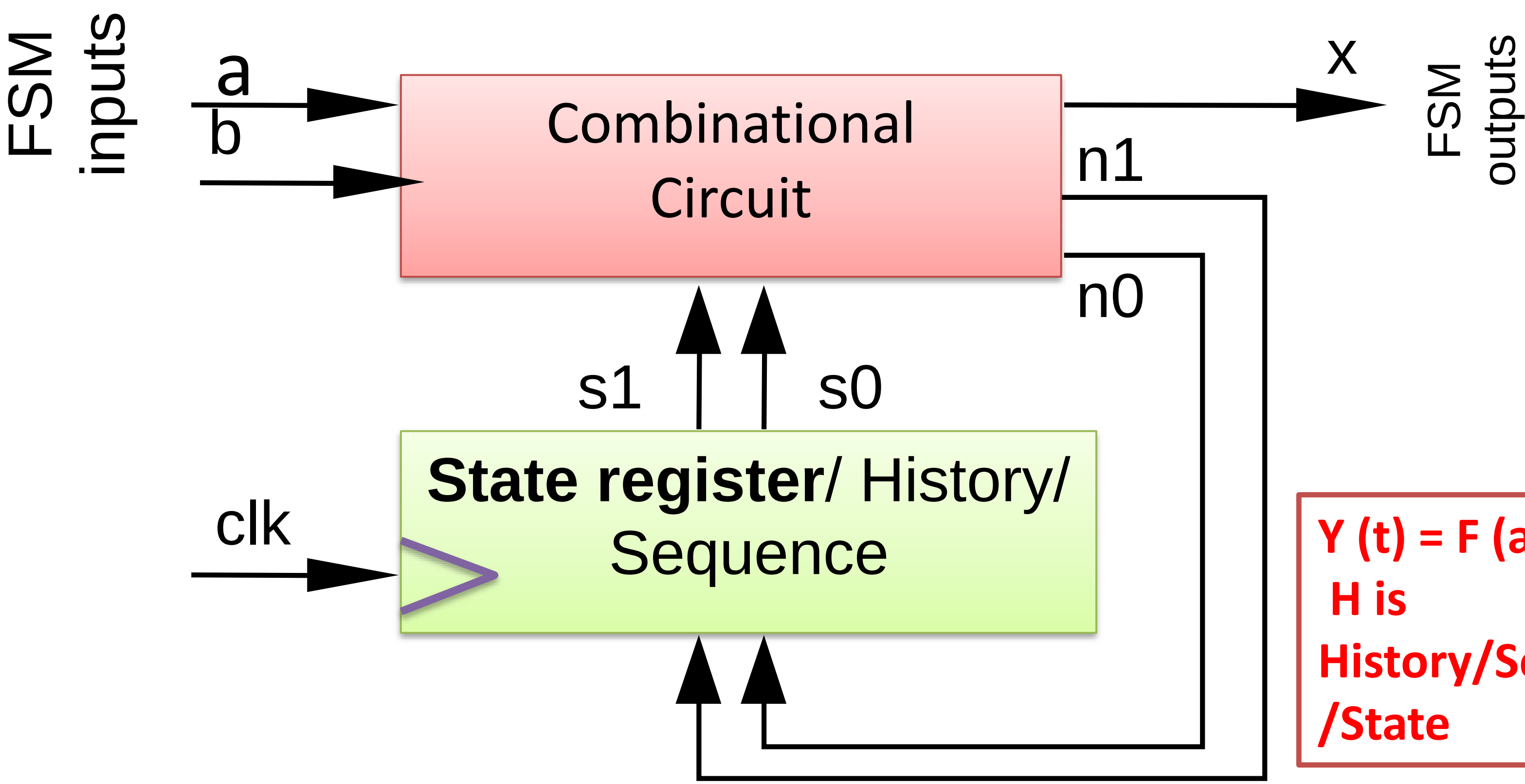
- Combinational Circuit : Formal Approach
  - Boolean Algebra
  - Circuit Minimization (K-Map, Quine McCluskey, Expresso…)
- Sequential Circuit : Formal Approach
  - Finite State Machine

Formally Describe/mathematically Describe

Boolean Algebra: Working Combinational Circuit

Finite State Machine: Working of Sequential Circuit

Sequential Circuit: FSM



Y (t) = F (a(t),b(t), H)
H is History/Sequence /State

Formally Describe/mathematically Describe

Boolean Algebra: Working Combinational Circuit

Finite State Machine: Working of Sequential Circuit

Already Designed Sequential Circuit

- Flip Flops (RS, JK, T, D)
- Register (Shift, PIPO), Memory
- Counter : Async, Sync, Modulo Counter
- Counter using Shift register

Till now we have not used formal approach to design these

Need a Better Way to Design Sequential Circuits

- Combinational circuit design process had two important things
  1. A formal way to describe desired circuit behavior
     - Boolean equation, or truth table
  2. A well-defined process to convert that behavior to a circuit
- We need those things for sequence circuit design

Sequential Circuit

- Can we model all synchronous sequential circuit using some model?

Yes, with Finite State Machine (FSM)

Finite State Machine

- Finite-State Machine (FSM)
  - A way to describe desired behavior of sequential circuit
  - Similar/Akin to Boolean equations for combinational behavior
  - List states, and transitions among states

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 $\mathbf{S} \rightarrow \mathbf{O}$

Clocked synchronous FSM

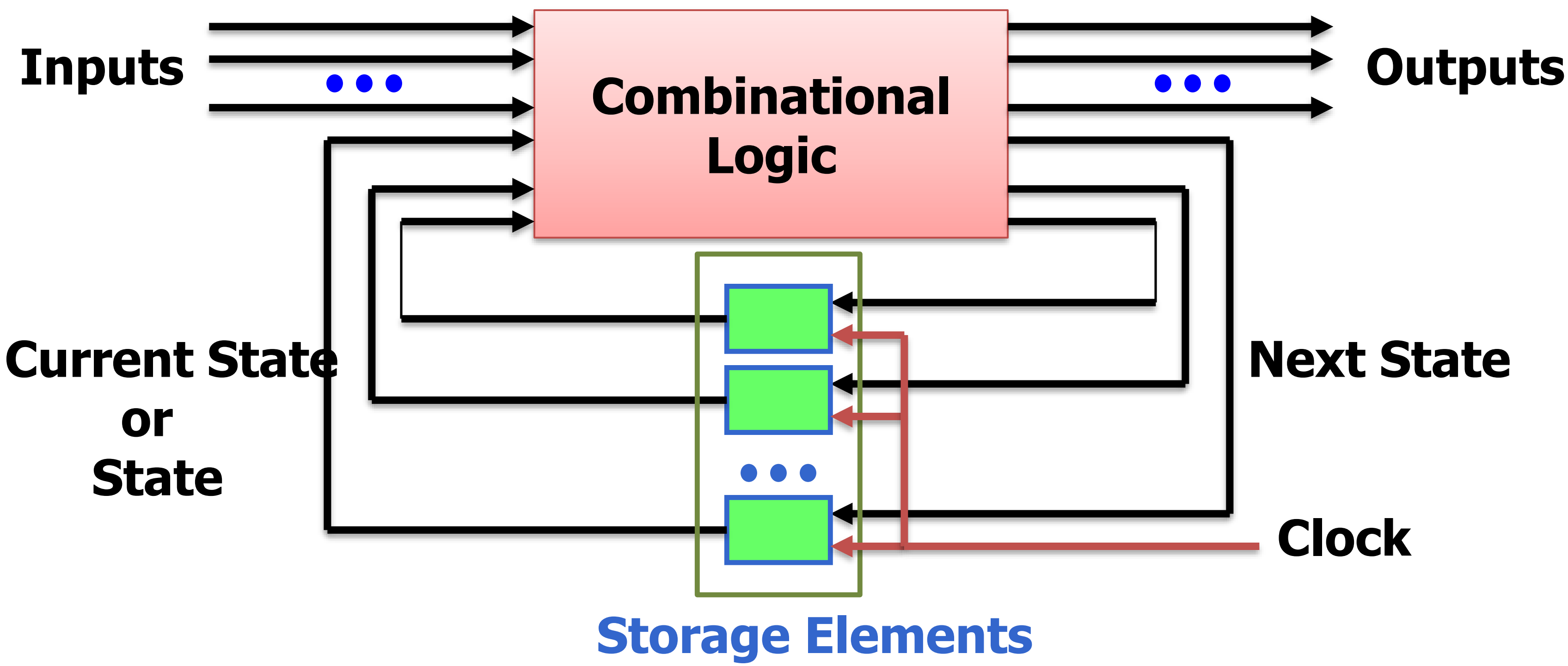
- **Clocked**
  - All storage elements employ a clock input (i.e. all storage elements are flip-flops)
- **Synchronous**
  - All of the flip flops use the same clock signal

Clocked Asynchronous FSM

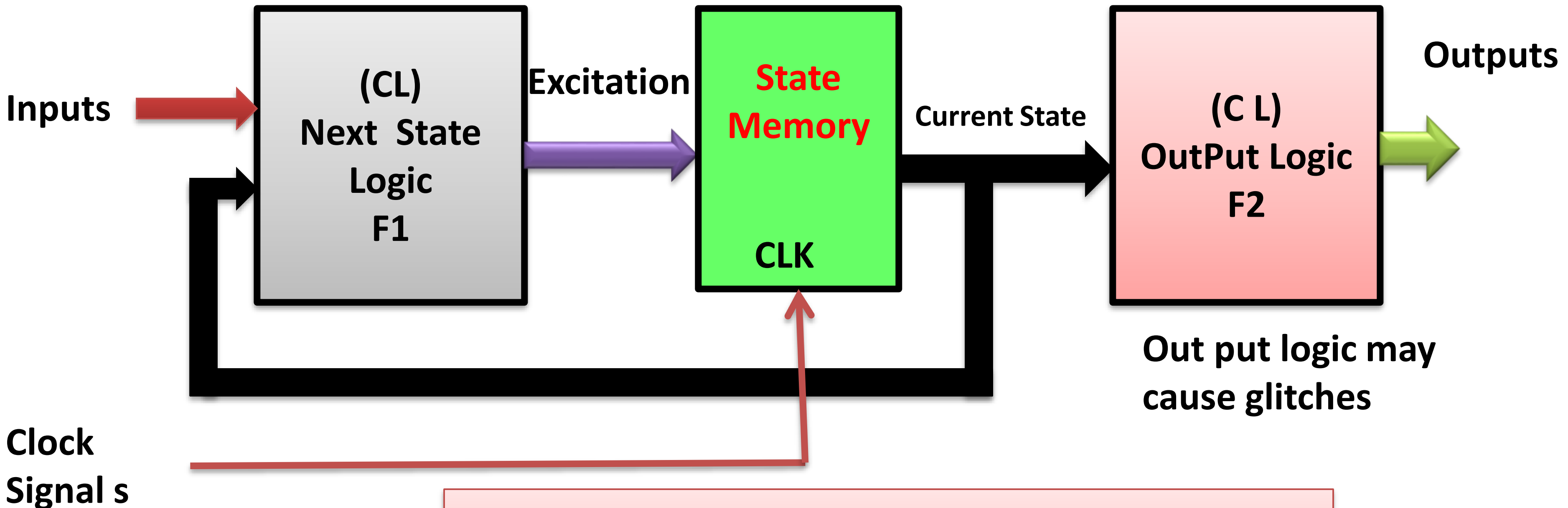
- **FSM**
  - State machine is simply another name for sequential circuits.
  - Finite refers to the fact that the number of states the circuit can assume if finite
- **Async FSM:** A synchronous clocked FSM changes state only when a triggering edge (or tick) occurs on the clock signal
  - **This will not be our focus in this course**

Clocked synchronous FSM structure

- **States**: determined by possible values in sequential storage elements
- **Transitions**: change of state
- **Clock**: controls when state can change by controlling storage elements



Moore machine



Next state = F_2(current state, inputs)

Output = G_2(current state)

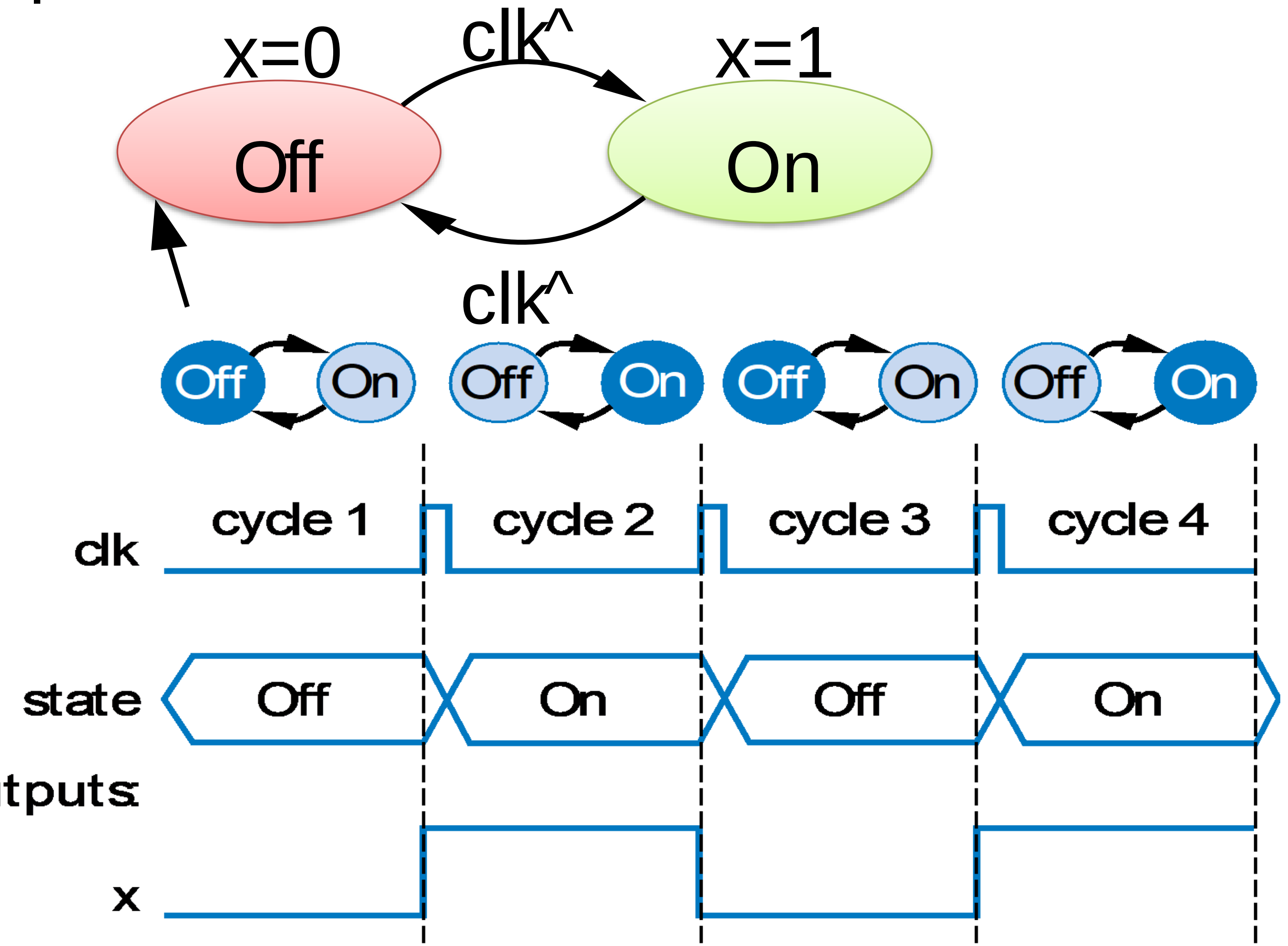
FSM Example 1
ON-OFF FSM

Finite State Machine: Example 1

- **On-off Example FSM**
- Make x change toggle (0 to 1, or 1 to 0) every clock cycle
- Two states: “Off” (x=0), and “On” (x=1)
- Transition from Off to On, or On to Off, on rising clock edge
- Arrow with no starting state points to initial state (when circuit first starts)

Finite State Machine: Example

Outputs: x



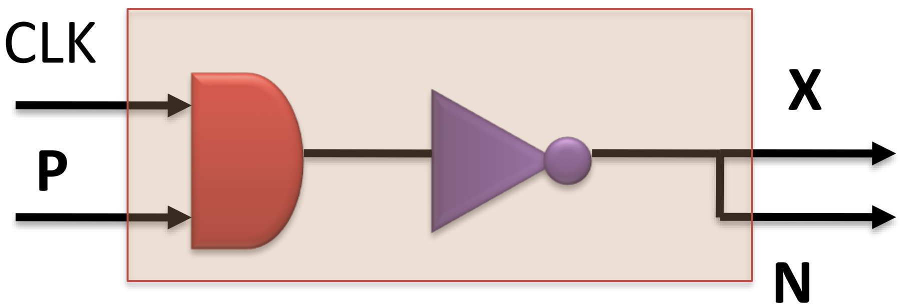
FSM

We often draw FSM graphically, known as ***state diagram***

Can also use table (state table), or textual languages

FSM Controller for On-Off Example

Input		Output	
CLK	P	X	N
RE 1	0	1	1
RE 1	1	0	0

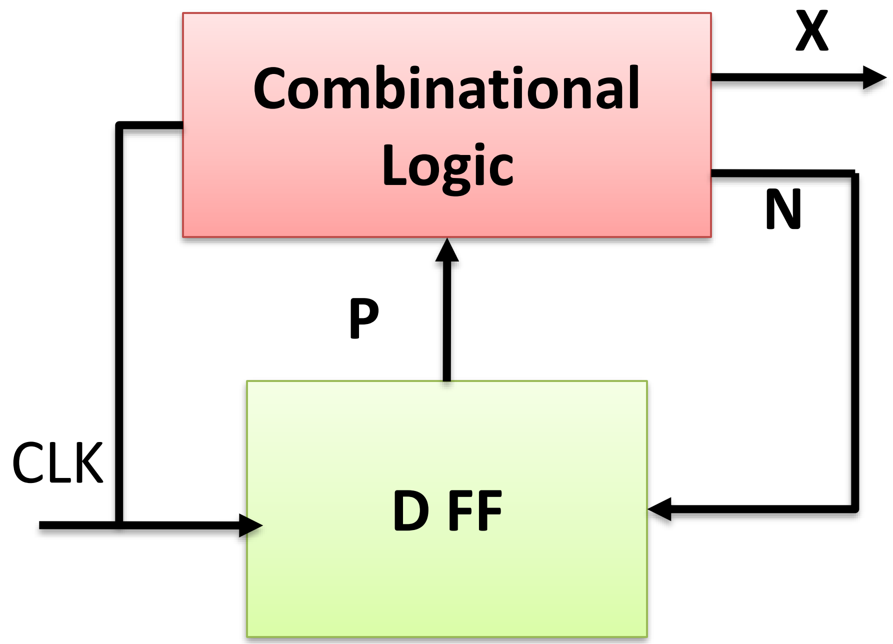


Think of Clock Enable: only **Rising Edge (RE)**
Above one may not work: Level Sensitive

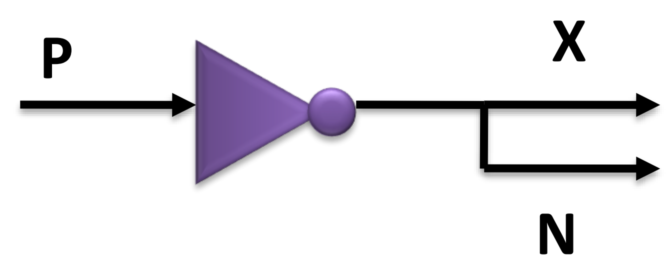
In this example For simplicity:
we are using X, N are function of P and CLK But in FSM X is function of P

Controller for On-Off

Input		Output	
CLK	P	X	N
RE 1	0	1	1
RE 1	1	0	0



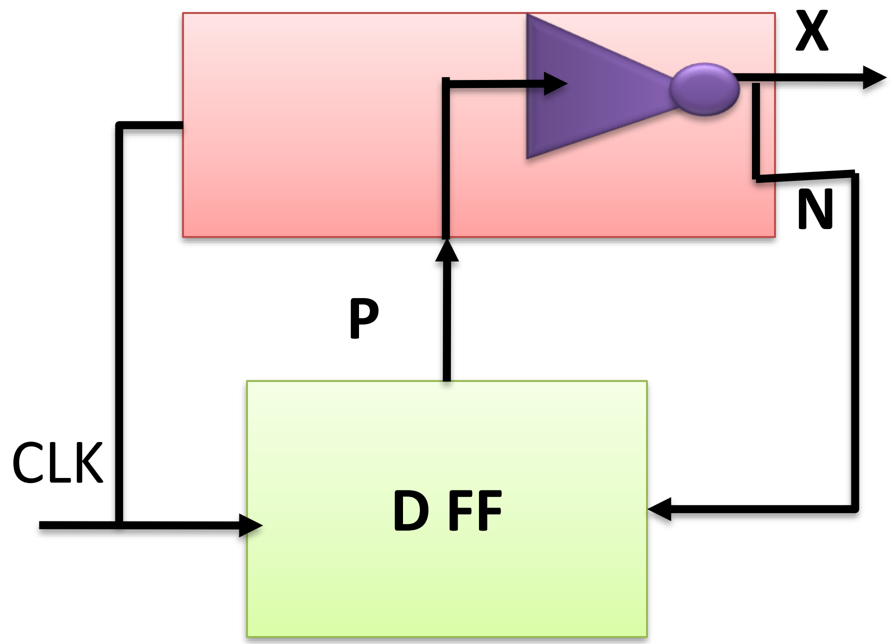
D-FF used to store the Present state



Rising Edge: Clock implicit

Controller for On-Off

Input		Output	
CLK	P	X	N
RE 1	0	1	1
RE 1	1	0	0



This a T-FF design using a D-FF
Where T is always 1
== > T is not an external input

Rising Edge: Clock implicit

FSM Example 2
D-FF

FSM for D-FF

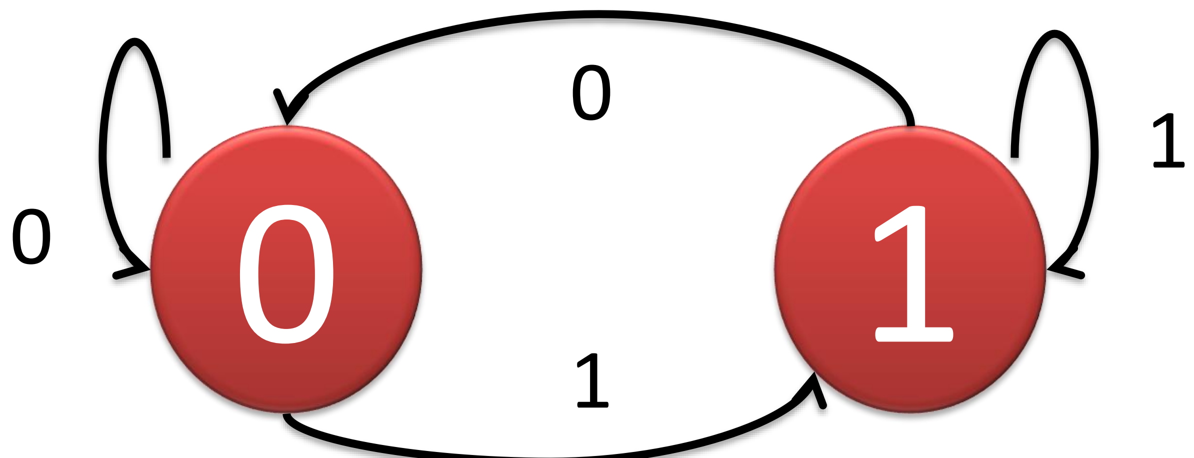
PS= Q(t)	Input	NS =Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Q(t+1)=Input

State: 0, 1

Input : 0, 1

Output: 0,1



FSM Controller for D-FF is D-FF

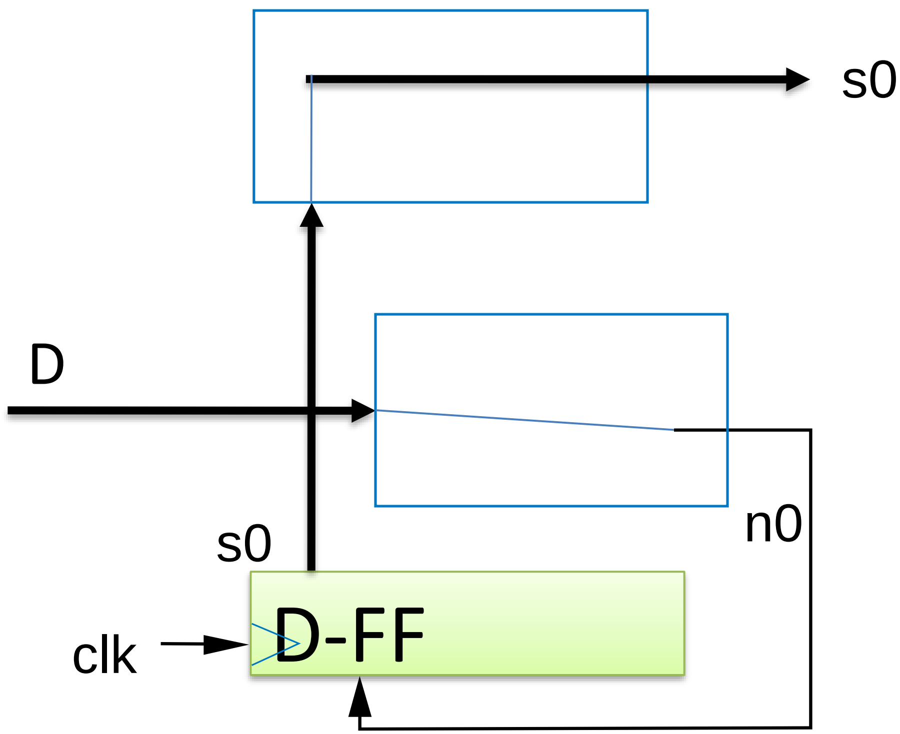
State: 0, 1

Input : 0, 1

Output: 0,1

$Q(t+1)=D$

PS= Q(t)	Input (D)	NS =Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1



FSM Example 3
T-FF

FSM for T-FF

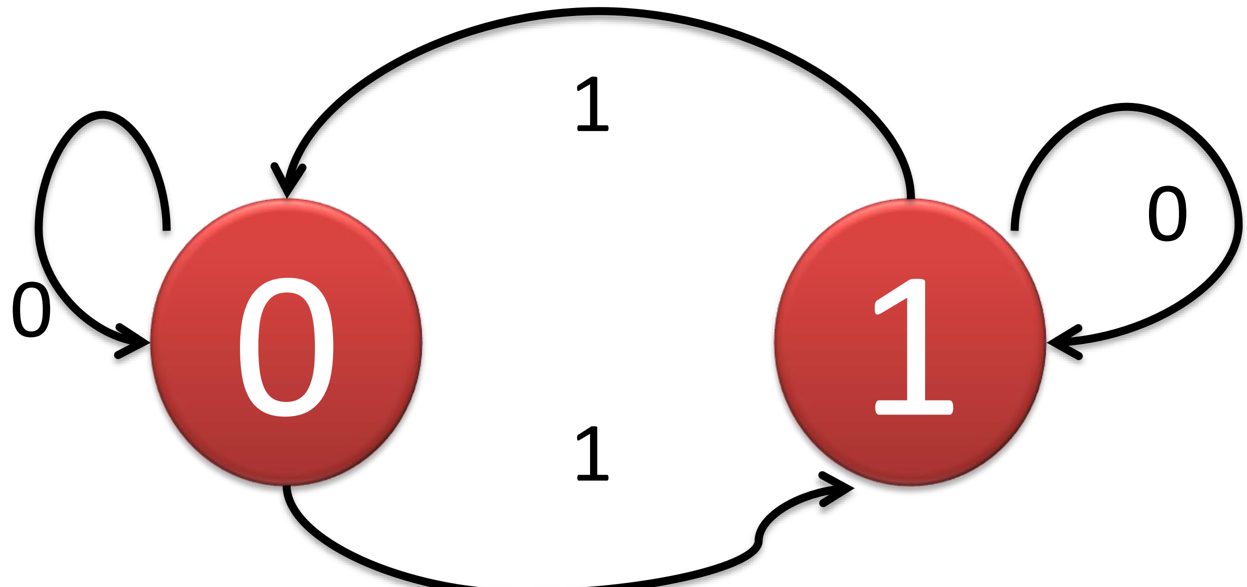
PS= Q(t)	Input(T)	NS =Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

$Q(t+1)=Q'(t)T(t)+Q(t)T'(t)$

State: 0, 1

Input : 0, 1

Output: 0,1



FSM Controller for T-FF

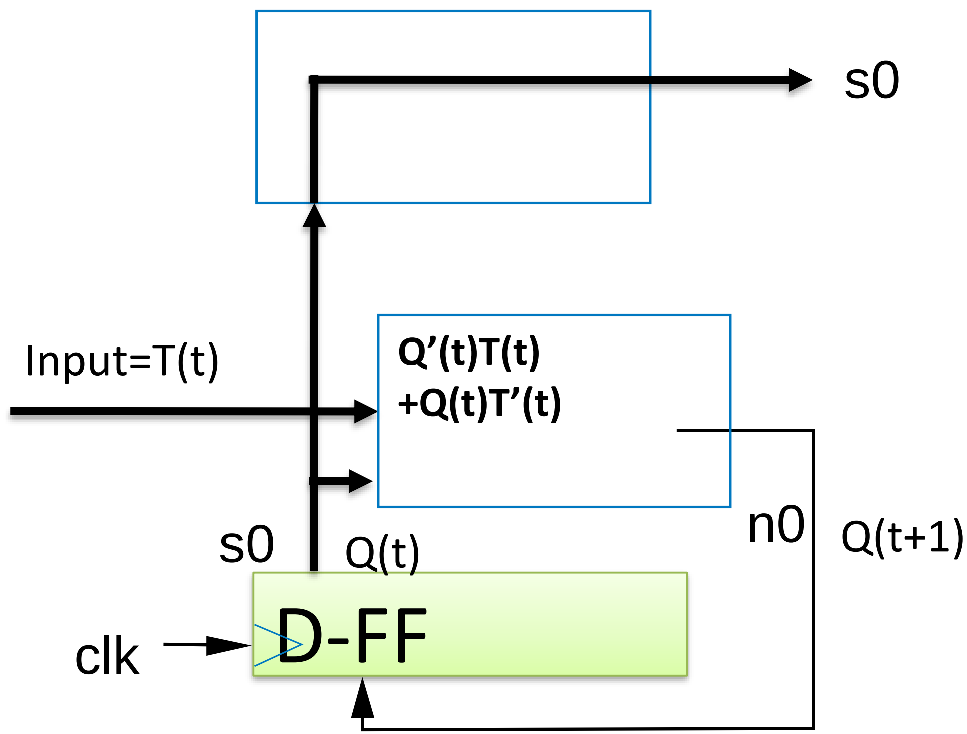
State: 0, 1

Input : 0, 1

Output: 0,1

PS= Q(t)	Input	NS =Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

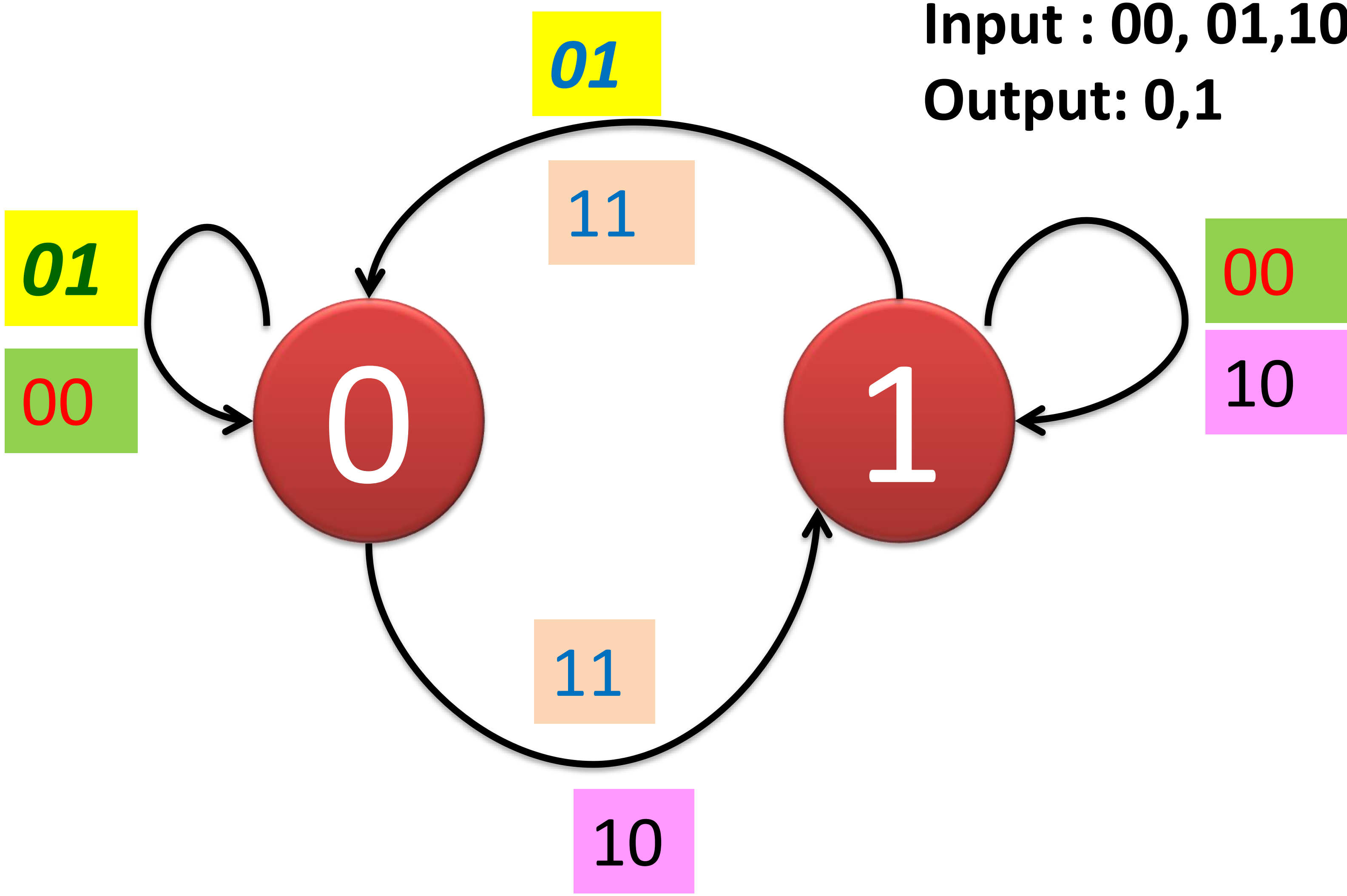
$$Q(t+1) = Q'(t)T(t)+Q(t)T'(t)$$



FSM Example 4
JK-FF

FSM for JK-FF

State: 0, 1
Input : 00, 01,10,11
Output: 0,1



FSM for JK-FF

State: 0, 1

Input : 00, 01,10,11

Output: 0,1

PS= Q(t)	Input(JK)	NS =Q(t+1)
0	00	0
0	01	0
0	10	1
0	11	1
1	00	1
1	01	0
1	10	1
1	11	0

$$Q(t+1)=JQ'(t)+K'Q(t)$$

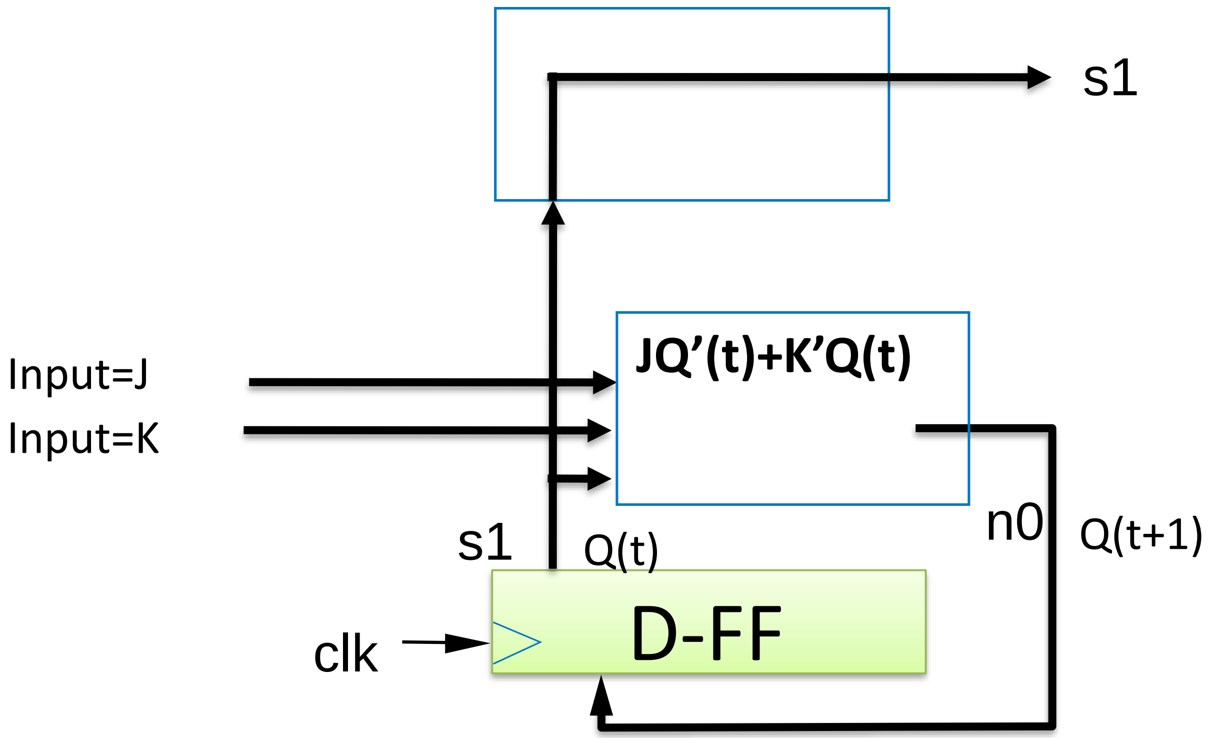
FSM Controller for JK-FF

$Q(t+1)=JQ'(t)+K'Q(t)$

State: 0, 1

Input : 00, 01,10,11

Output: 0,1



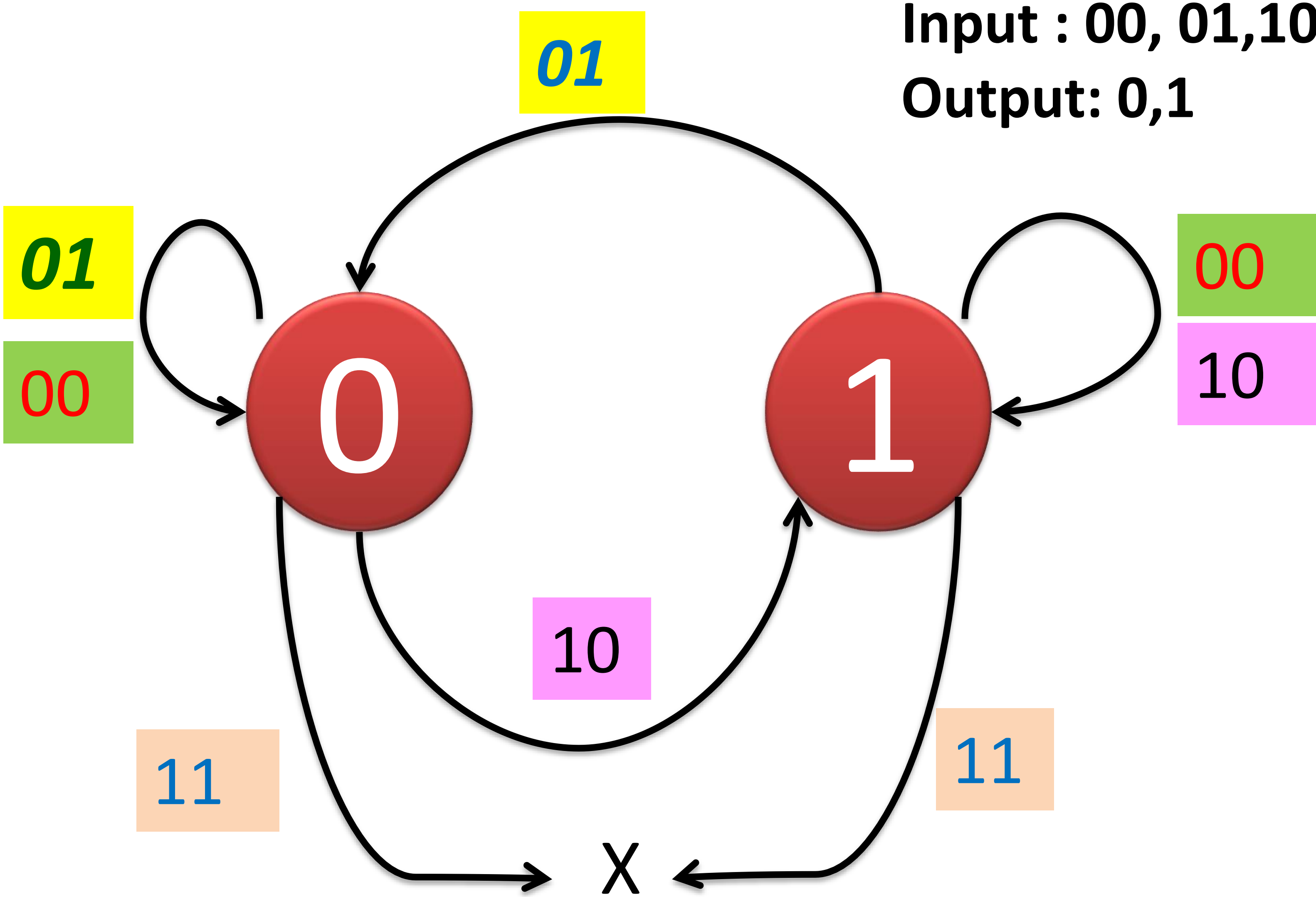
FSM Example 5
RS-FF

FSM for RS-FF

State: 0, 1

Input : 00, 01,10,11

Output: 0,1



FSM for RS-FF

State: 0, 1

Input : 00, 01,10,11

Output: 0,1

PS= Q(t)	Input(RS)	NS =Q(t+1)
0	00	0
0	01	0
0	10	1
0	11	x
1	00	1
1	01	0
1	10	1
1	11	x

$Q(t+1)=R+Q(t).S'$

FSM Controller for RS-FF

$Q(t+1)=Q(t+1)=R+Q(t).S'$

State: 0, 1

Input : 00, 01,10,11

Output: 0,1