CS221: Digital Design

http://jatinga.iitg.ernet.in/~asahu/cs221

Finite State Machine

A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- Digital Clock Design
- Finite State Machine
- Formal definition
- State machine types
- FSM implementation

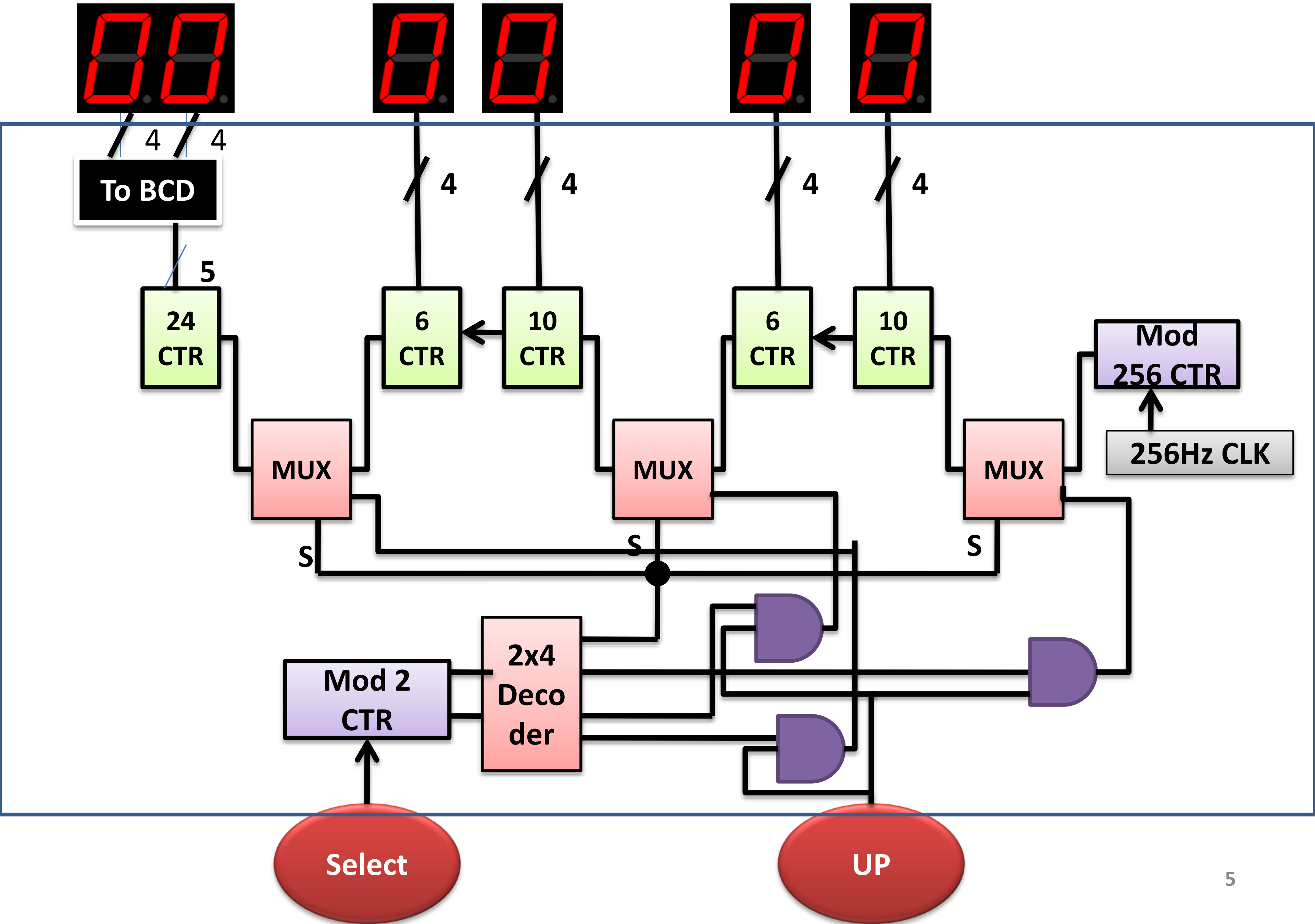
Design of Digital Wall Clock

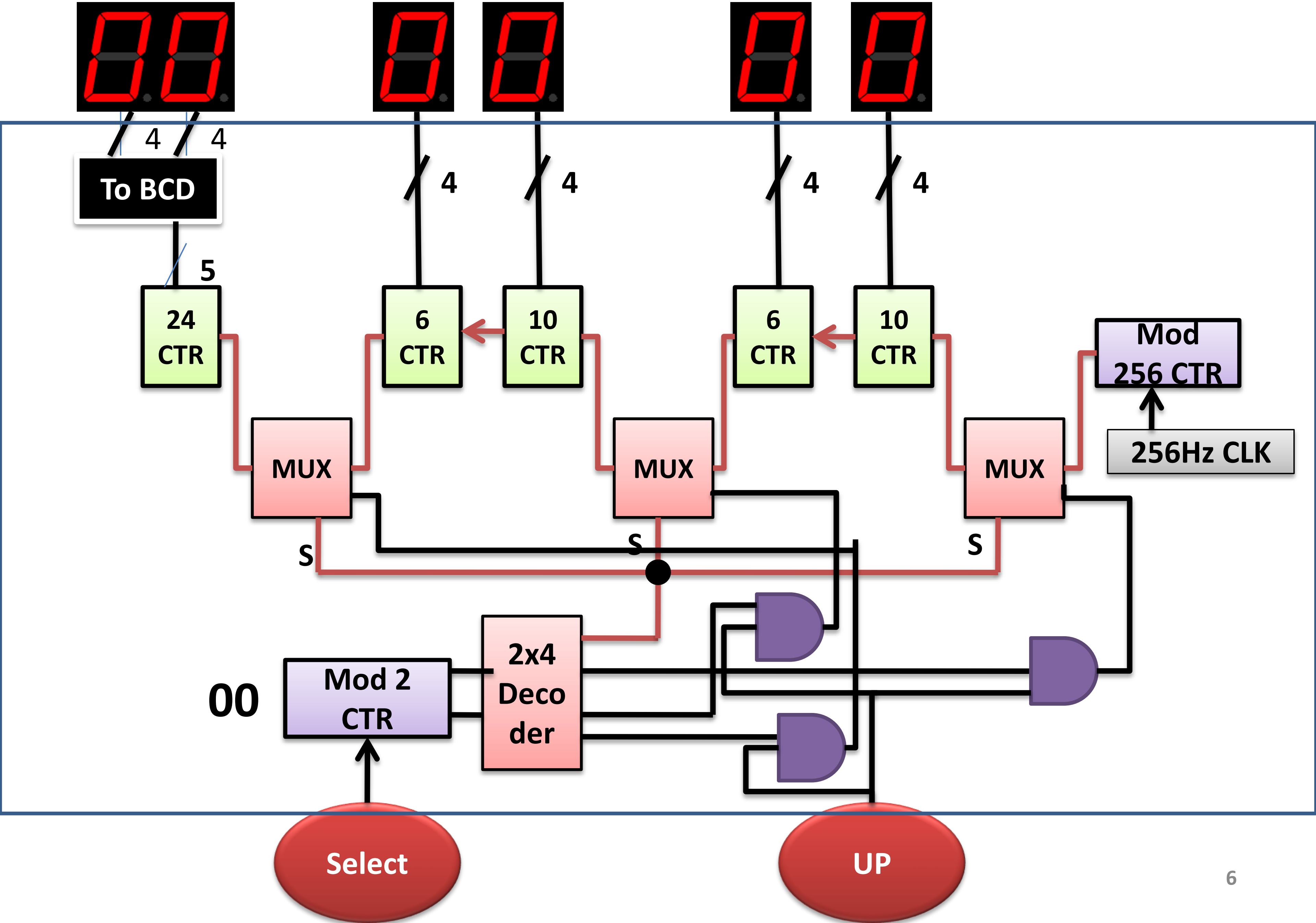
- Given 256Mhz Clock Quartz and other Digital components

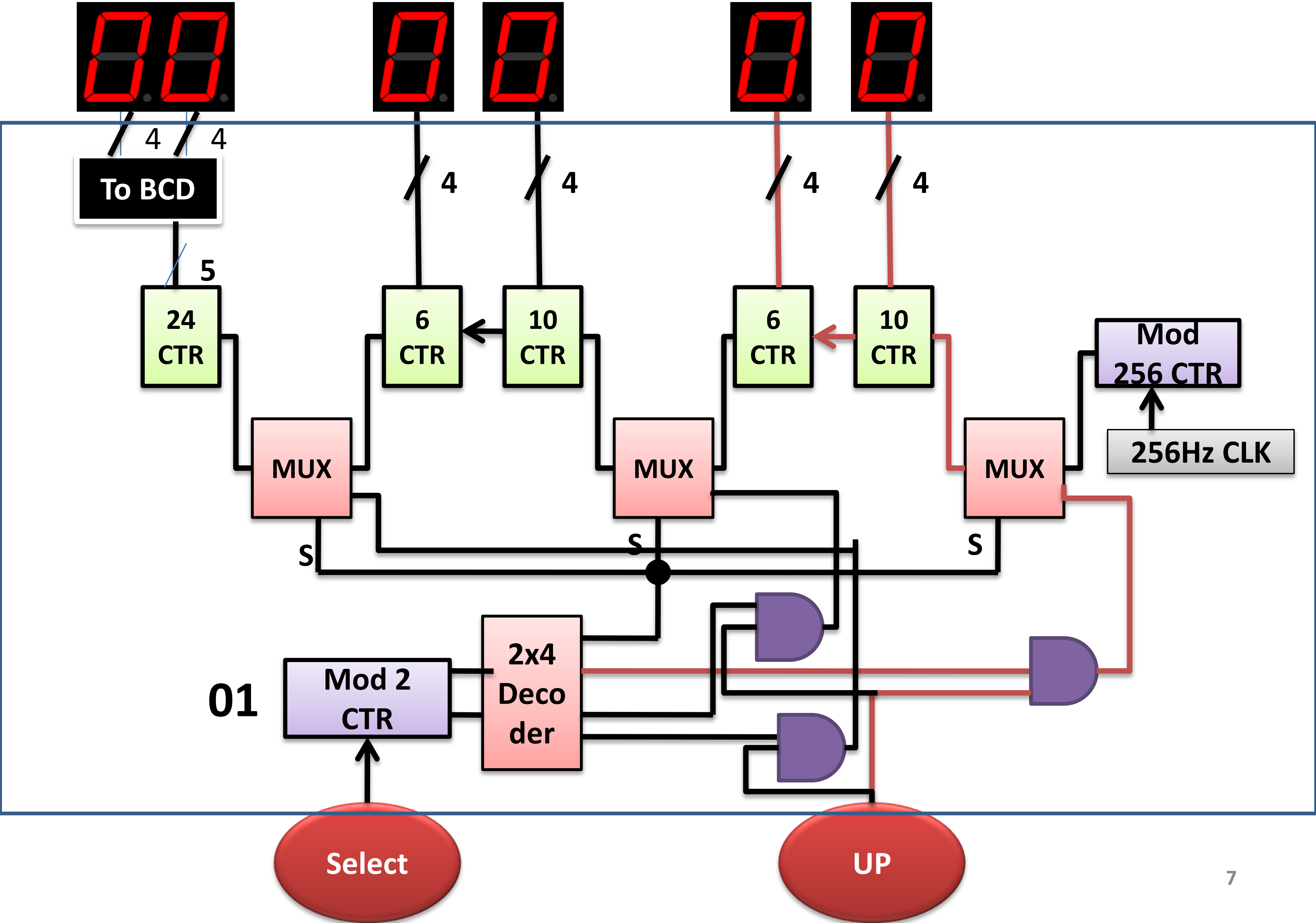
Design of Digital Wall Clock

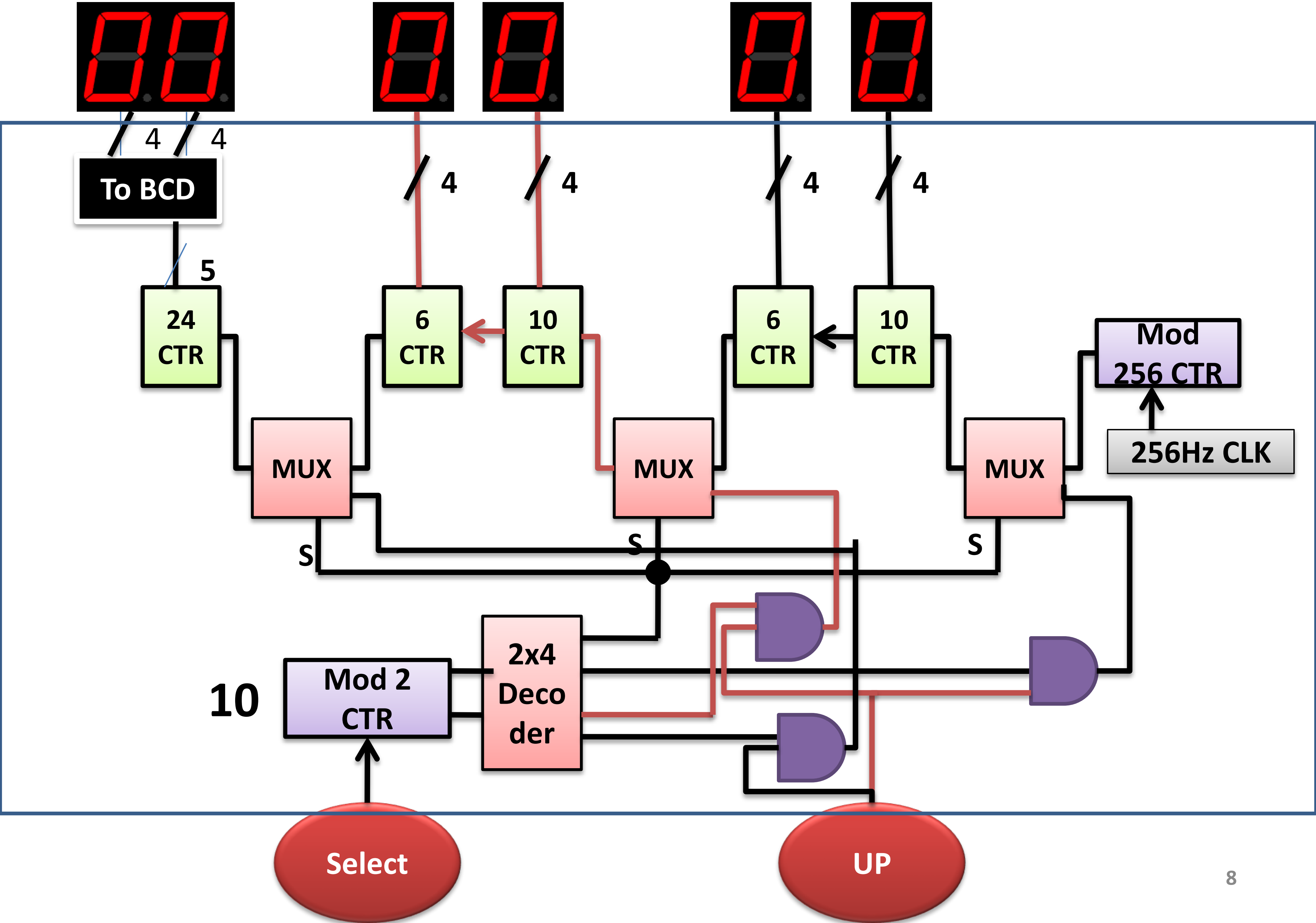
- Given 256Mhz Clock Quartz and other Digital components

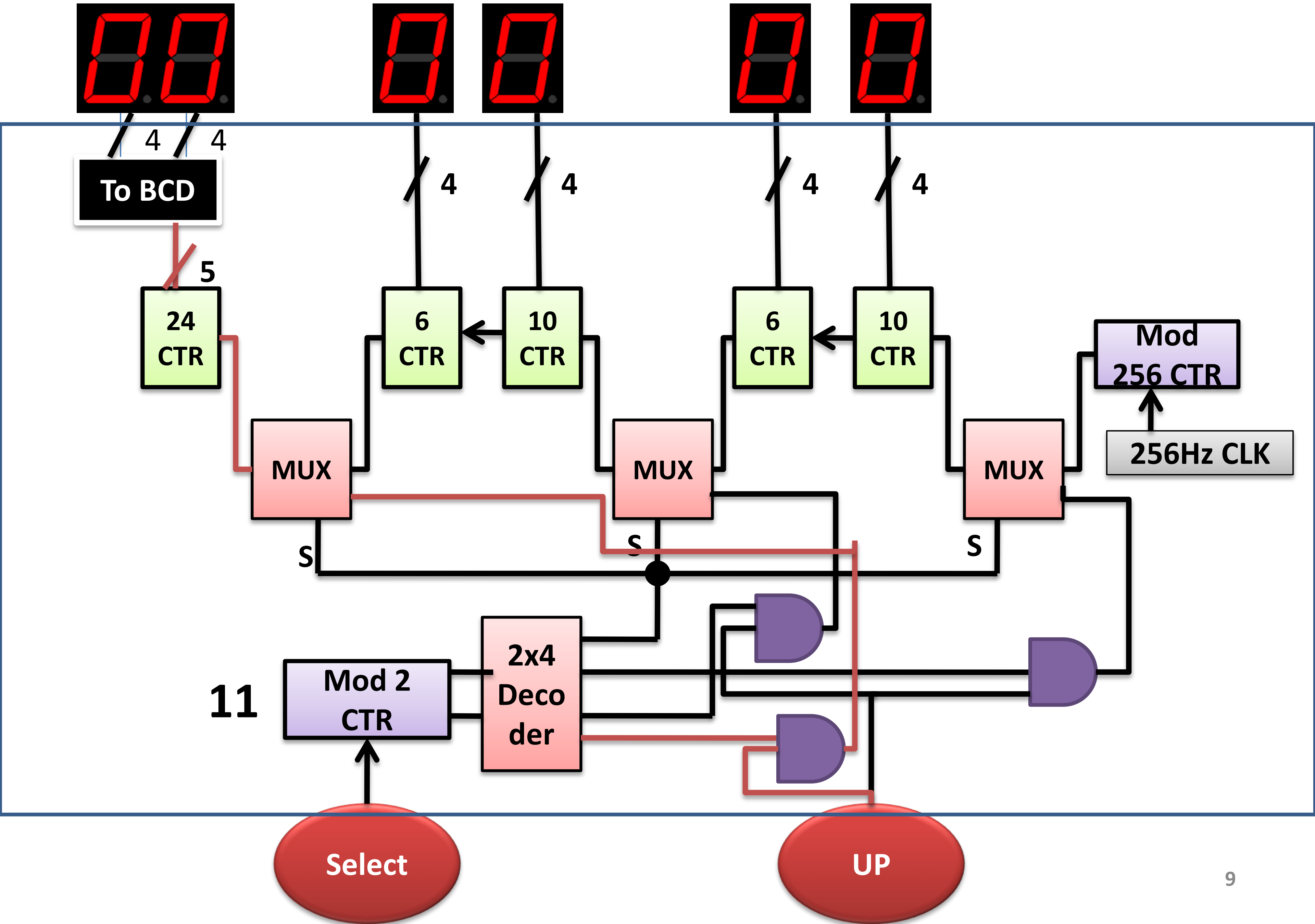
- Design a Wall Clock
  - To display time : HH : MM :SS format
  - Should support Reset/Adjust of time using selectable switch
  - Button 1: for select the Mod Ctr
  - Button 2: increasing select mod Ctr


4
4
To BCD
5
24 CTR
4
4
4
4
6 CTR
10 CTR
6 CTR
10 CTR
Mod 256 CTR
256Hz CLK
MUX
MUX
MUX
S
S
S
2x4 Decoder
Mod 2 CTR
Select
UP


4
4
To BCD
5
24 CTR
4
4
6 CTR
10 CTR
4
4
6 CTR
10 CTR
Mod 256 CTR
256Hz CLK
MUX
MUX
MUX
S
S
S
2x4 Decoder
Mod 2 CTR
00
Select
UP


4
4
To BCD
5
24 CTR
4
6 CTR
4
10 CTR
4
6 CTR
4
10 CTR
Mod 256 CTR
256Hz CLK
MUX
MUX
MUX
S
S
S
2x4 Decoder
Mod 2 CTR
01
Select
UP


4
4
To BCD
5
24 CTR
4
4
6 CTR
10 CTR
4
4
6 CTR
10 CTR
Mod 256 CTR
256Hz CLK
MUX
MUX
MUX
S
S
S
2x4 Decoder
10
Mod 2 CTR
Select
UP


4
4
To BCD
5
24
CTR
4
4
6
CTR
10
CTR
4
4
6
CTR
10
CTR
Mod
256 CTR
256Hz CLK
MUX
MUX
MUX
S
S
S
11
Mod 2
CTR
2x4
Deco
der
Select
UP

Sequential Circuit: FSM

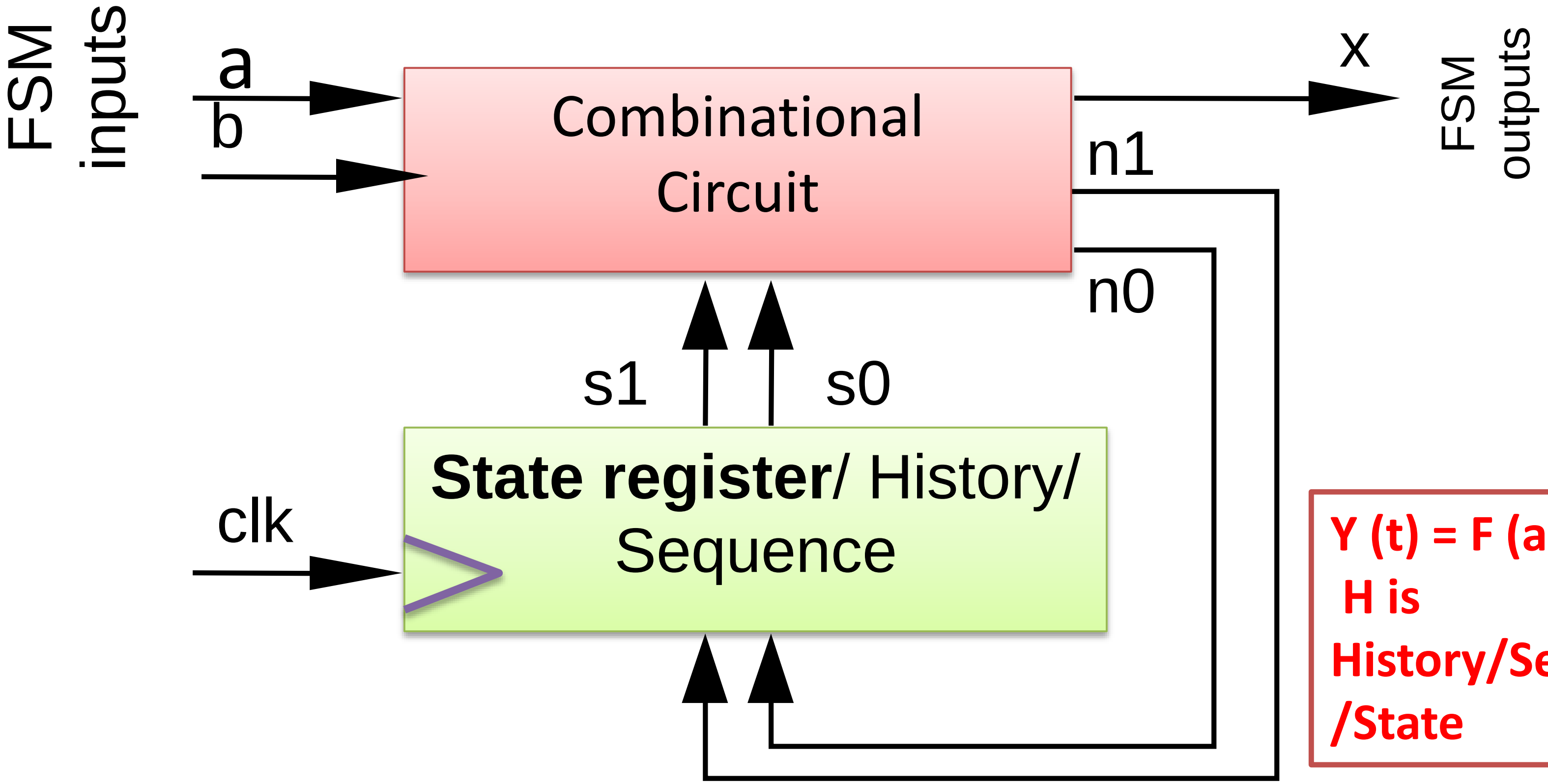
- Combinational Circuit : Formal Approach
  - Boolean Algebra
  - Circuit Minimization (K-Map, Quine McCluskey, Expresso…)
- Sequential Circuit : Formal Approach
  - Finite State Machine

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

Boolean Combinational Circuit

- Can you formally model All Boolean Combinational Circuit using Boolean Algebra?

YES

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?

Finite State Machine (FSM)

Finite State Machine

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

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 $\mathbf{S} \times \mathbf{I} \to \mathbf{S}$

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

Set Theoretic Description

Mealy Machine is an ordered quintuple

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

where

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

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

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

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

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

Clocked synchronous FSM

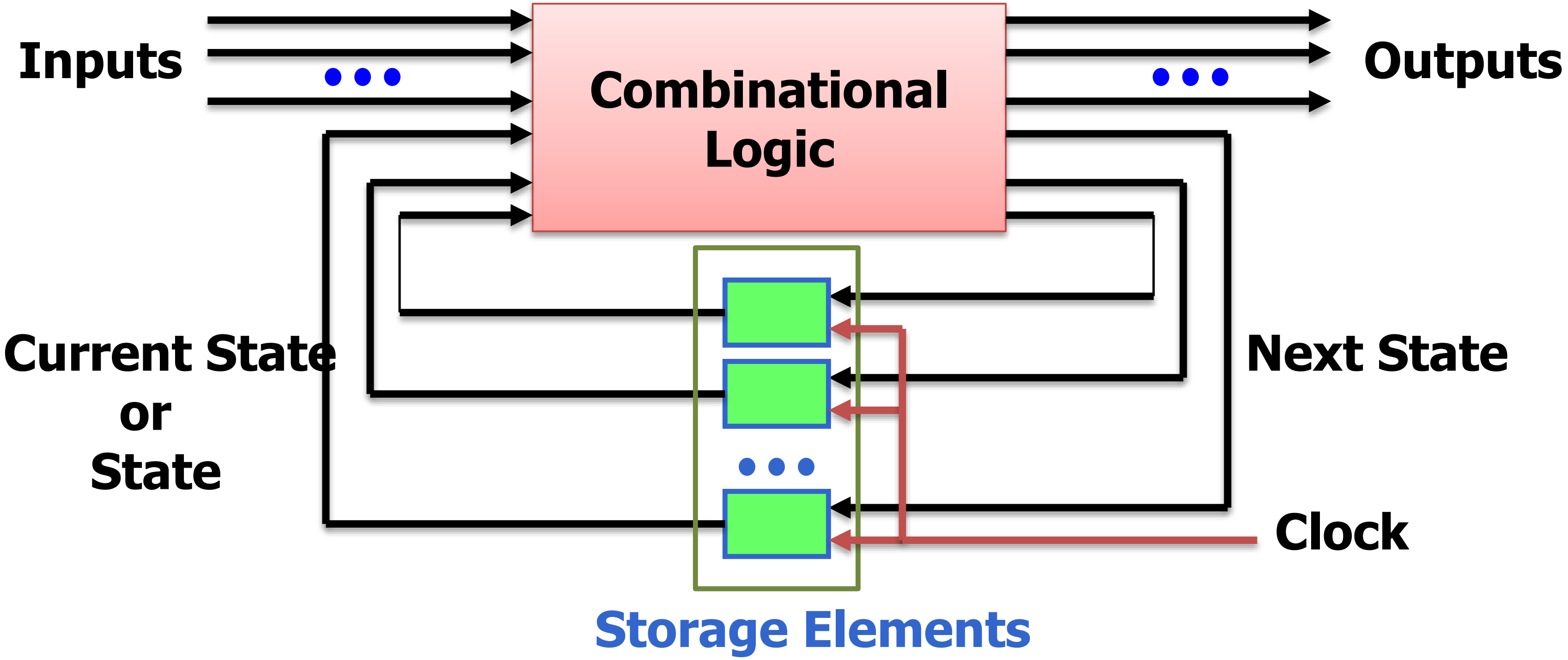
- **Clocked**
  - All storage elements employ a clock input (i.e. all storage elements are flip-flops)
- **Synchronous**
  - All of the flip flops use the same clock signal

Clocked Asynchronous FSM

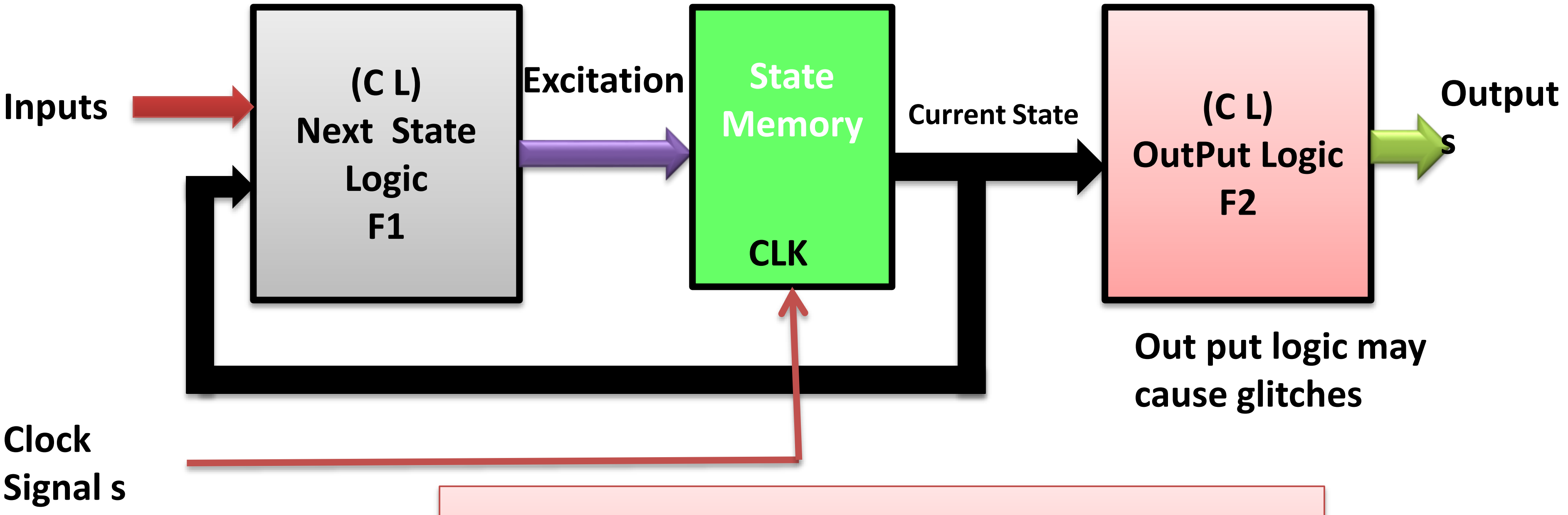
- **FSM**
  - State machine is simply another name for sequential circuits. Finite refers to the fact that the number of states the circuit can assume if finite
- **Async FSM:** A synchronous clocked FSM changes state only when a triggering edge (or tick) occurs on the clock signal

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



Inputs

(C L)
Next State
Logic
F1

Excitation

State
Memory

CLK

Current State

(C L)
OutPut Logic
F2

Output
s

Clock
Signal s

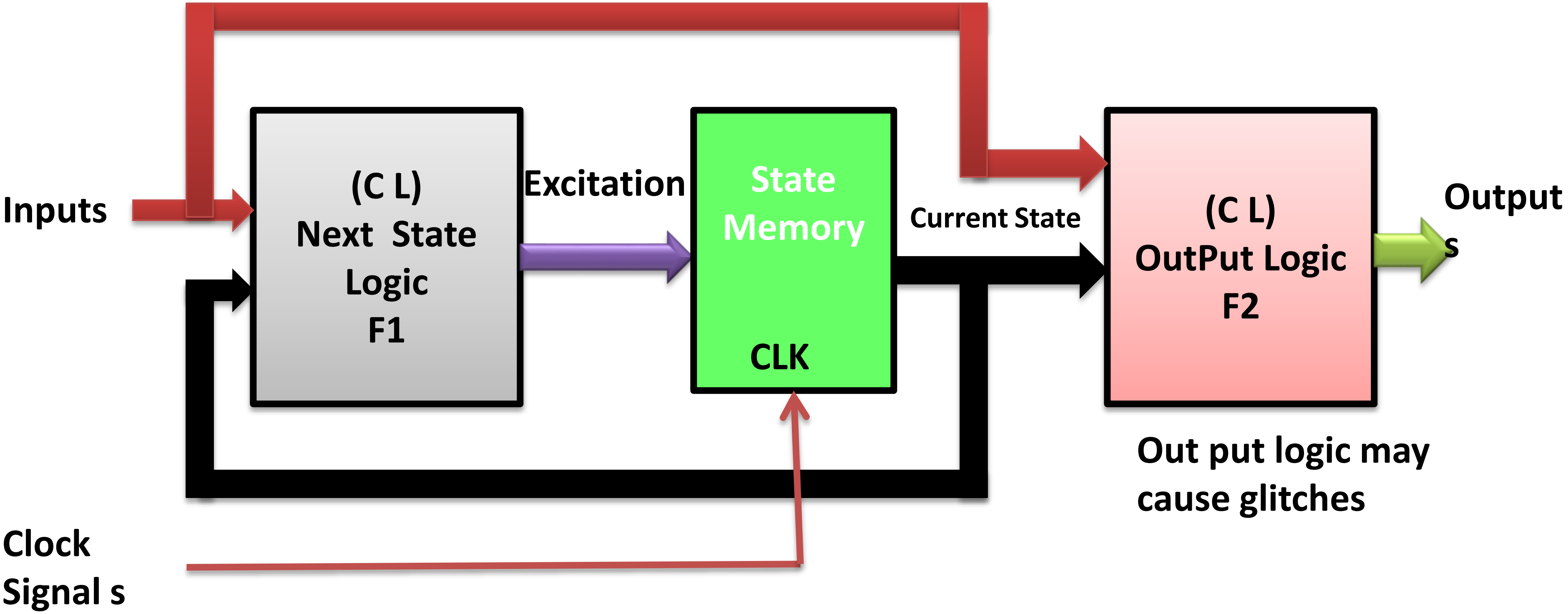
Out put logic may cause glitches

Next state = F_2(current state, inputs)

Output = G_2(current state)

Mealy machine

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



Out put logic may cause glitches

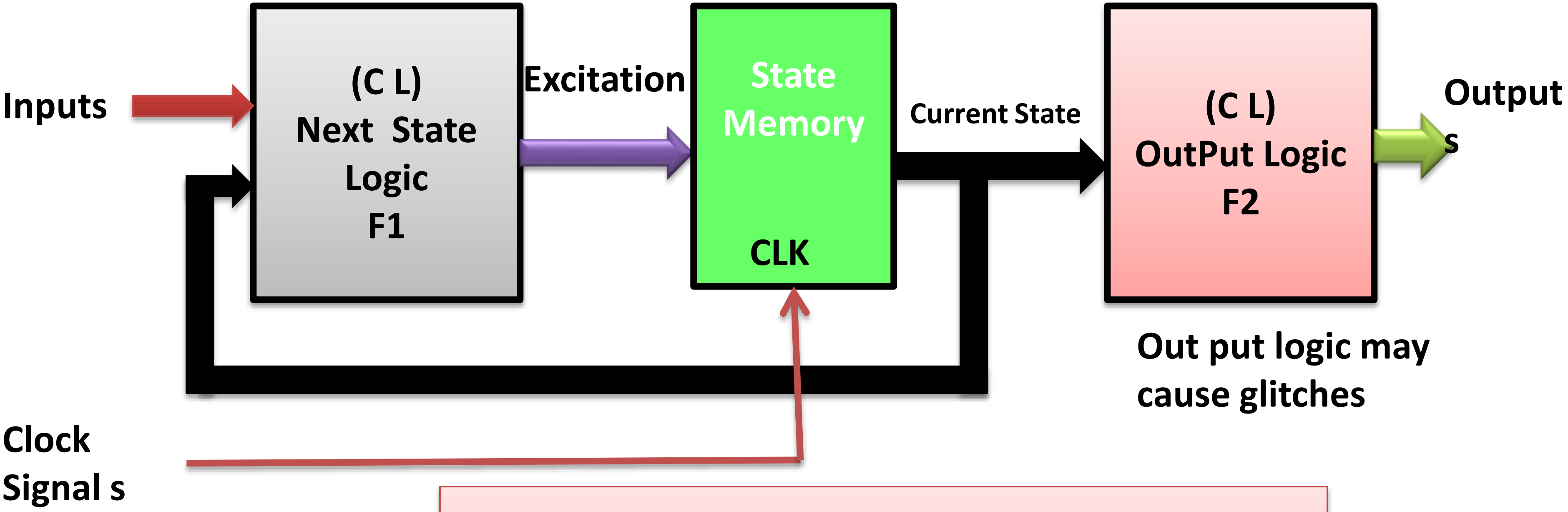
State Storage= Set of n FFs

2^n State can be stored

Next state = F_1(current state, inputs)

Output = F_2(current state, inputs)

Moore machine: For next some examples



Out put logic may cause glitches

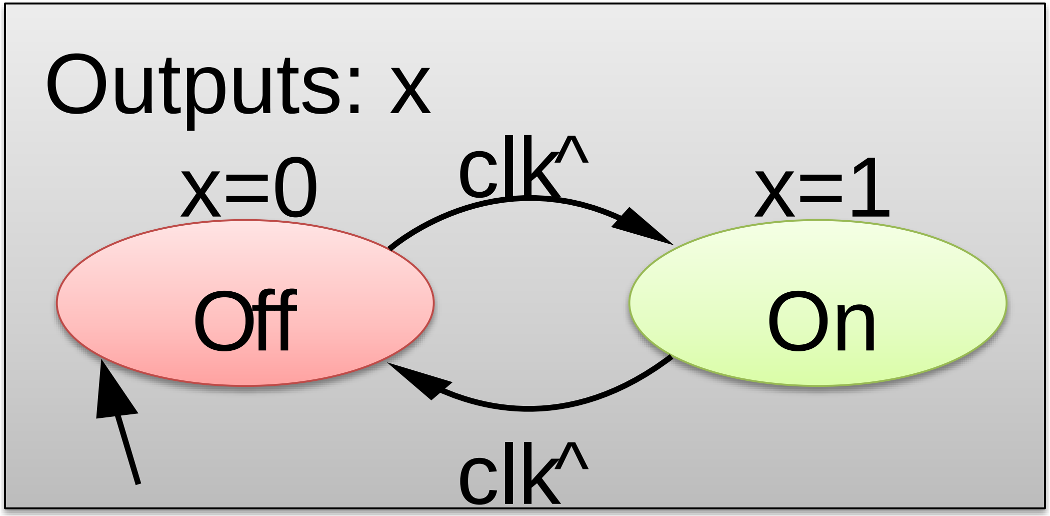
Next state = F2(current state, inputs)

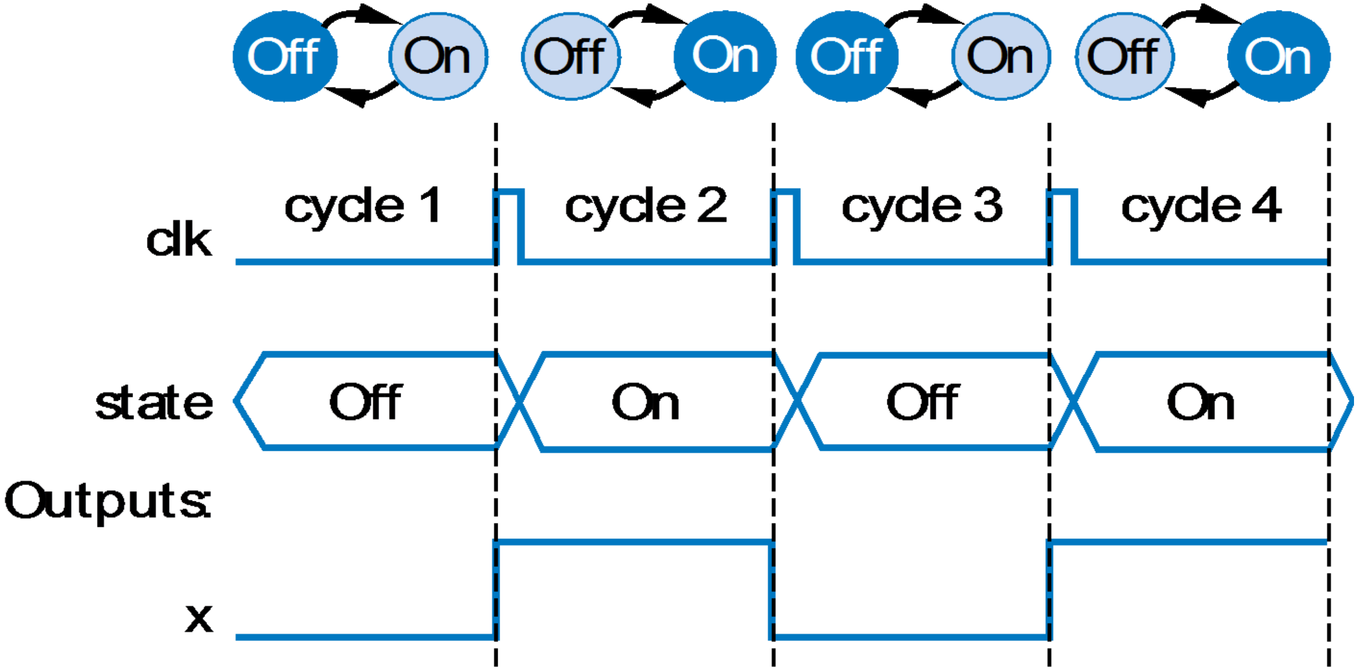
Output = G2(current state)

Finite State Machine: Example

- Example: 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





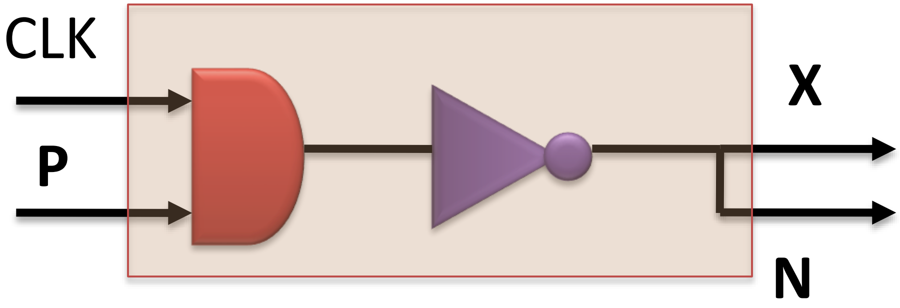
FSM

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

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

Controller for On-Off

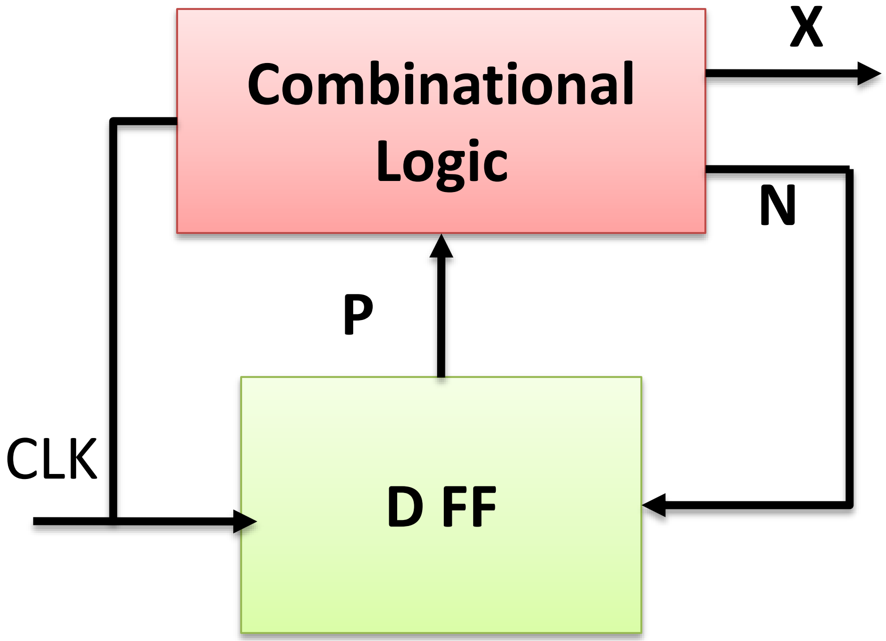
Input		Output	
CLK	P	X	N
RE 1	0	1	1
RE 1	1	0	0

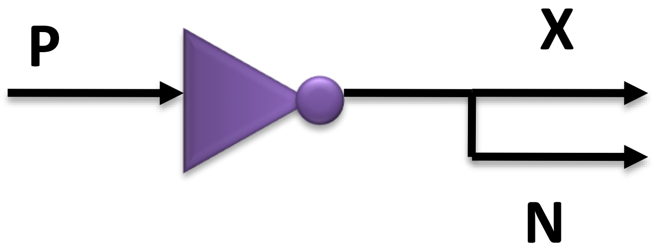


Think of Clock Enable: only **Rising Edge (RE)**
Above one may not work: Level Sensitive

Controller for On-Off

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





Rising Edge: Clock implicit

FSM for D-FF

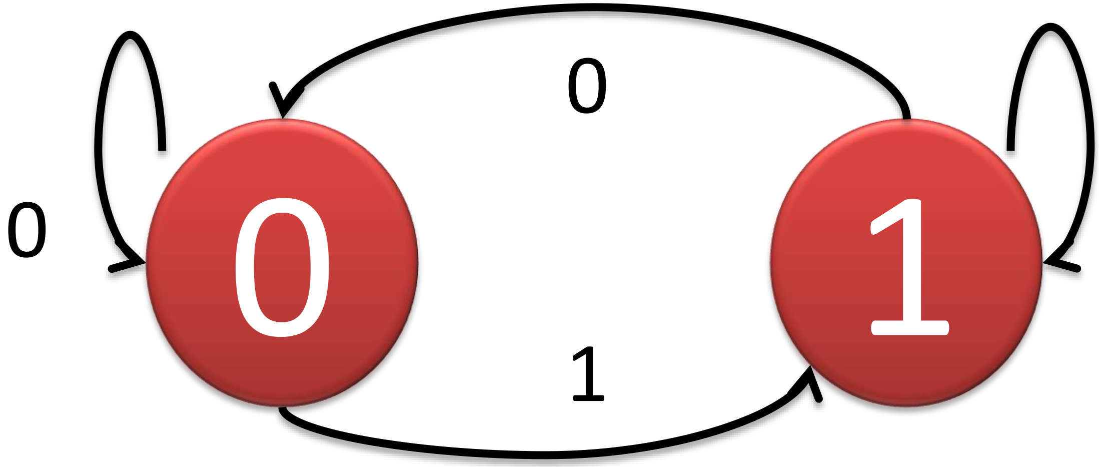
PS= Q(t)	Input	NS =Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Q(t+1)=Q(t)

State: 0, 1

Input : 0, 1

Output: 0,1



FSM Controller for D-FF is D-FF

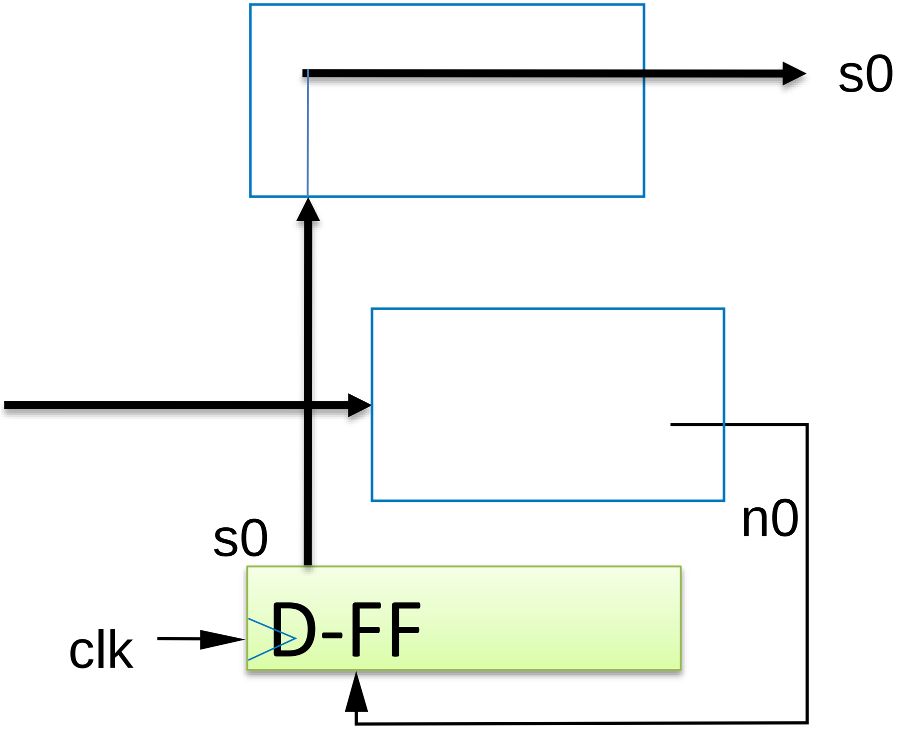
State: 0, 1

Input : 0, 1

Output: 0,1

Q(t+1)=Q(t)

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



FSM for T-FF

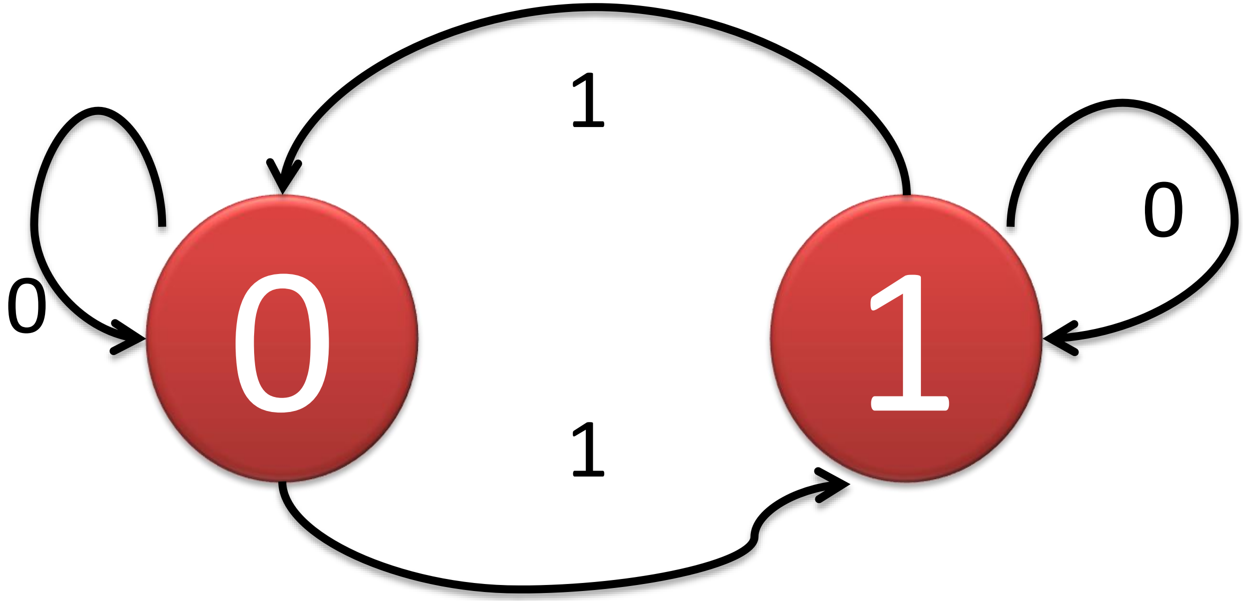
PS= Q(t)	Input	NS =Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

$$Q(t+1)=Q(t)T(t)+Q(t)T(t)$$

State: 0, 1

Input : 0, 1

Output: 0,1



FSM Controller for T-FF

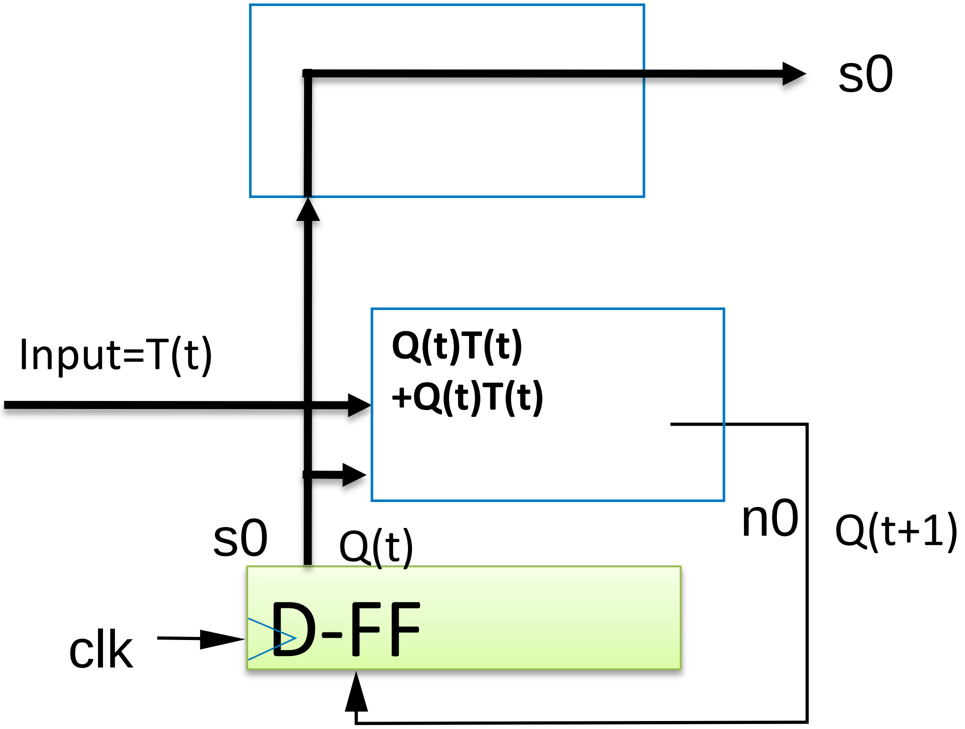
Q(t+1)=Q(t)

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	0
1	1	1

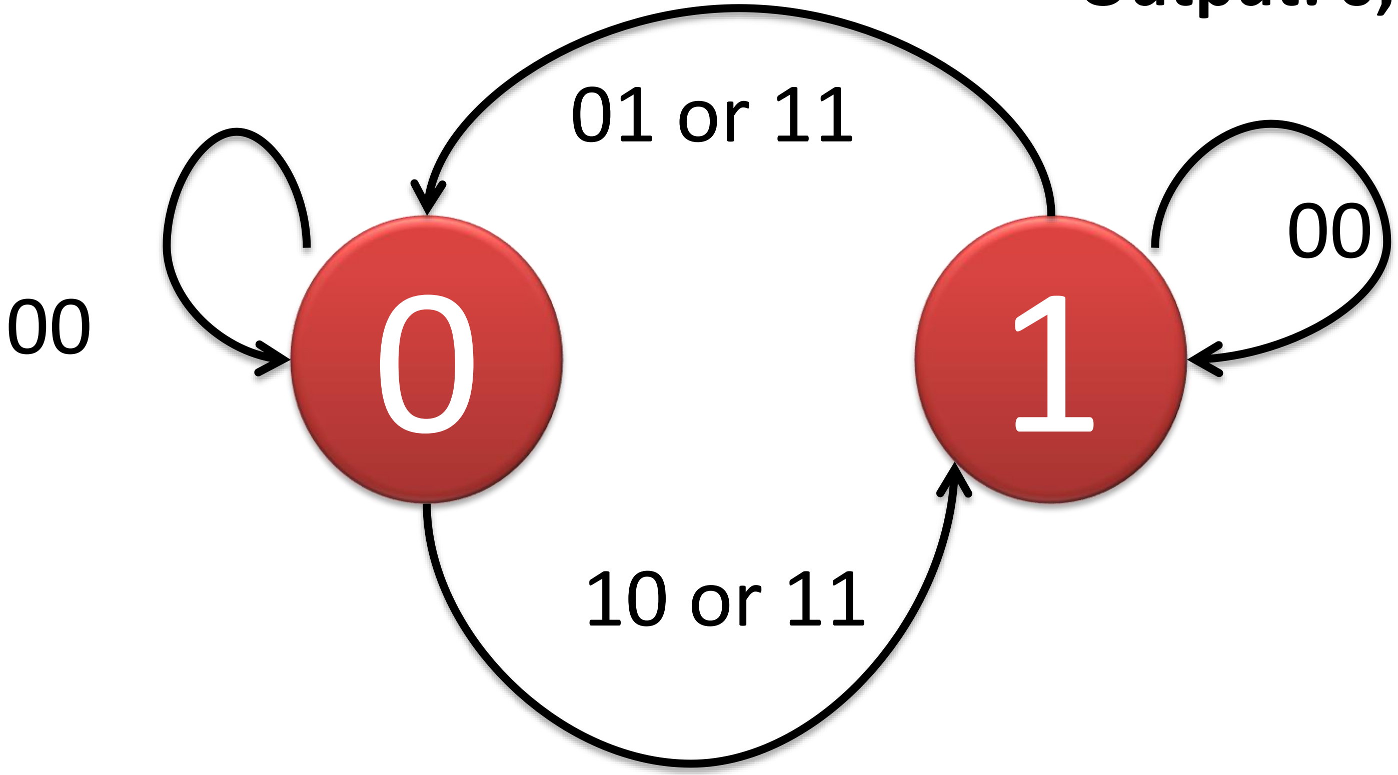


FSM for JK-FF

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

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

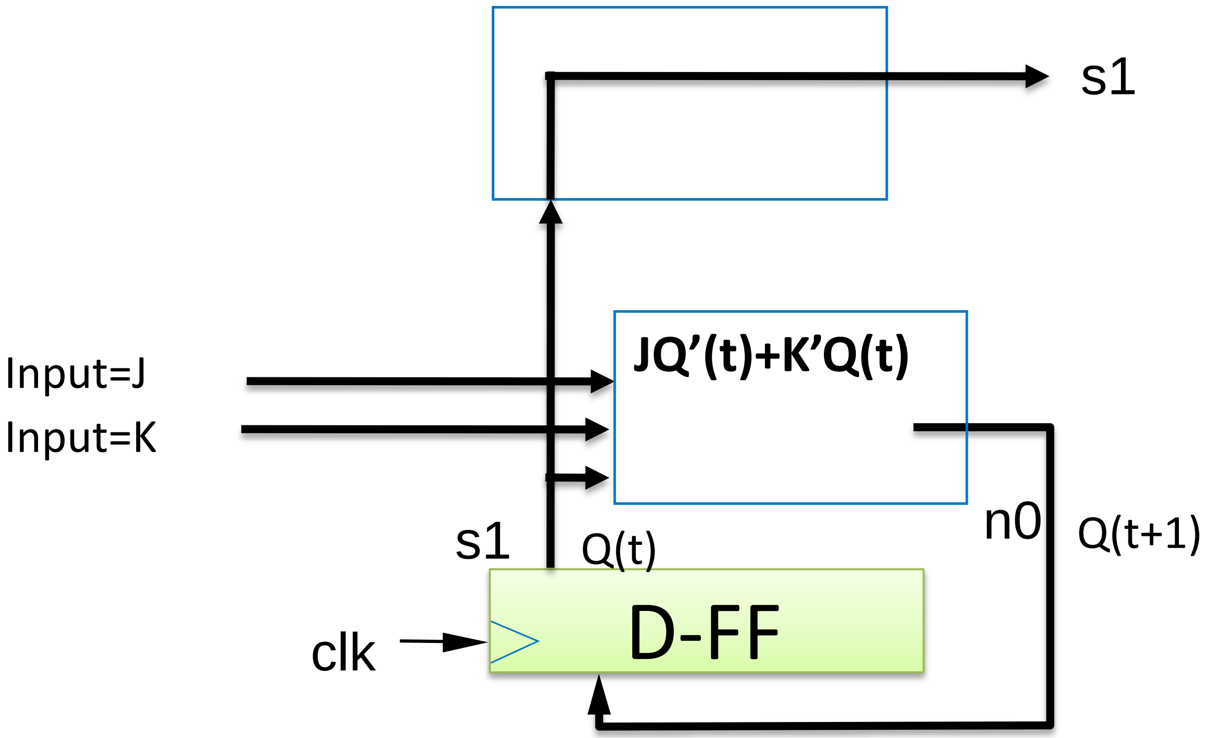
FSM Controller for JK-FF

Q(t+1)=JQ’(t)+K’Q(t)

State: 0, 1

Input : 00, 01,10,11

Output: 0,1



FSM for RS-FF

State: 0, 1

Input : 00, 01,10,11

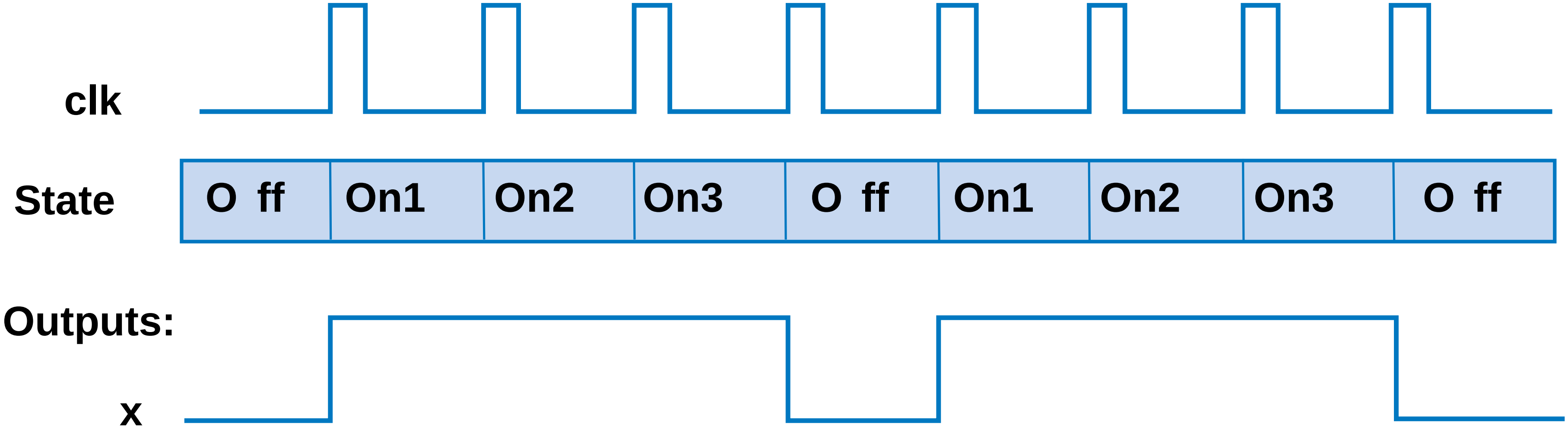
Output: 0,1

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

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

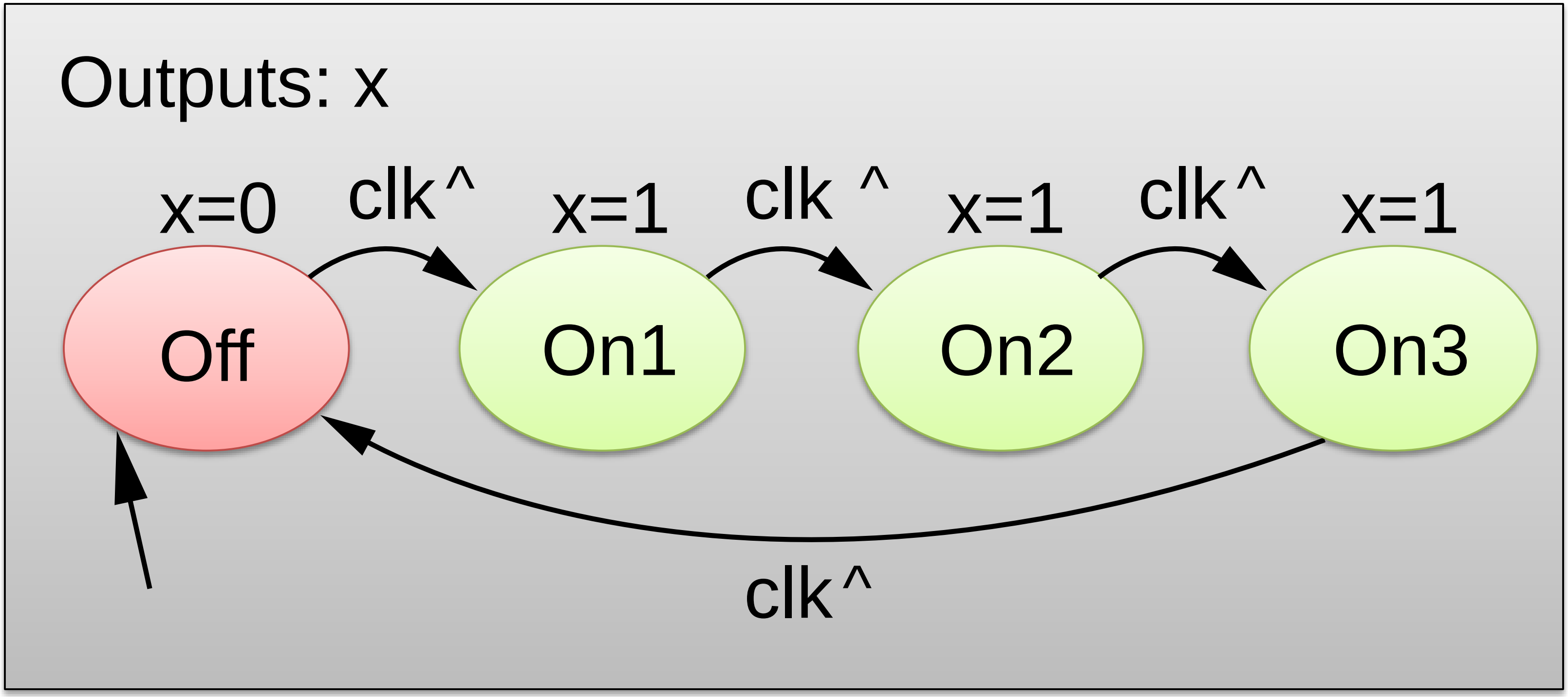
FSM Example: 0,1,1,1,repeat

- Want 0, 1, 1, 1, 0, 1, 1, 1, ...
  - Each value for one clock cycle
- Can describe as FSM: Four states, Transition on rising clock edge to next state



FSM Example: 0,1,1,1,repeat

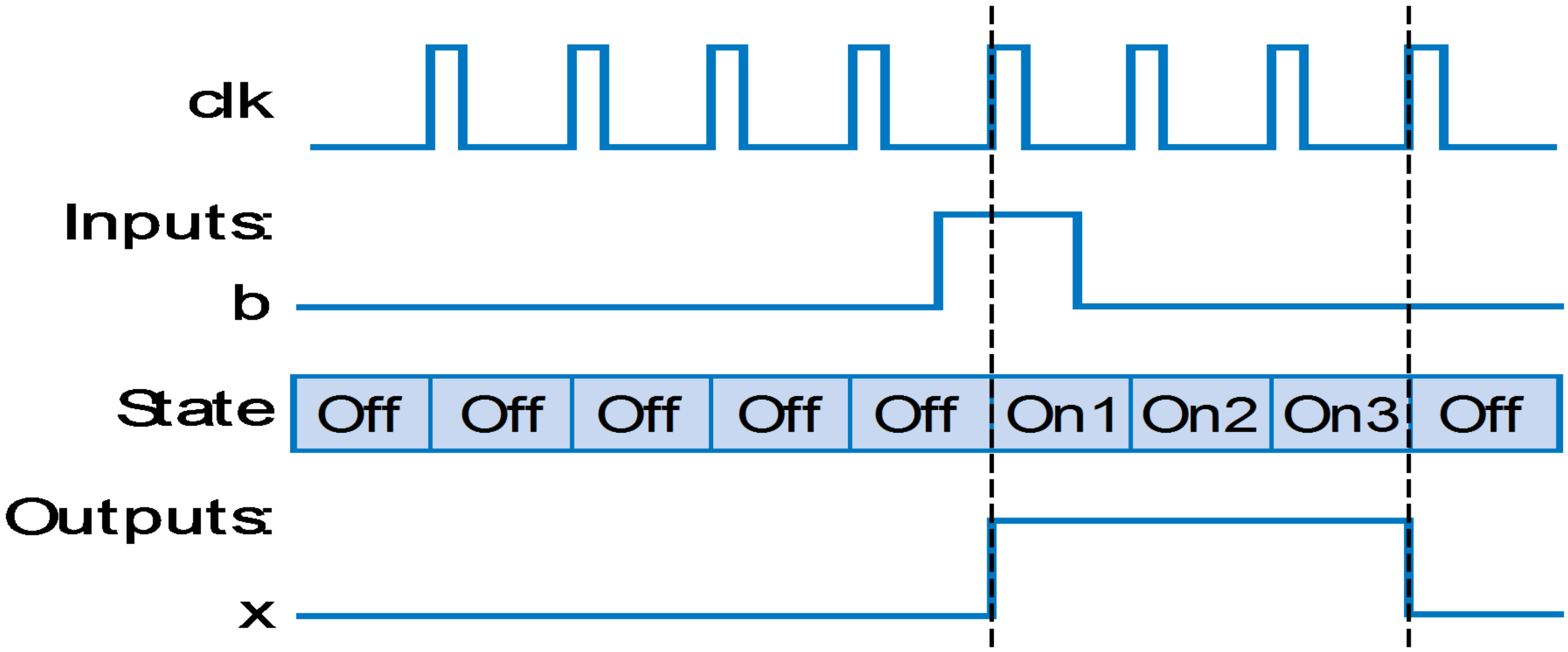
- Want 0, 1, 1, 1, 0, 1, 1, 1, ...
  - Each value for one clock cycle
- Can describe as FSM: Four states, Transition on rising clock edge to next state



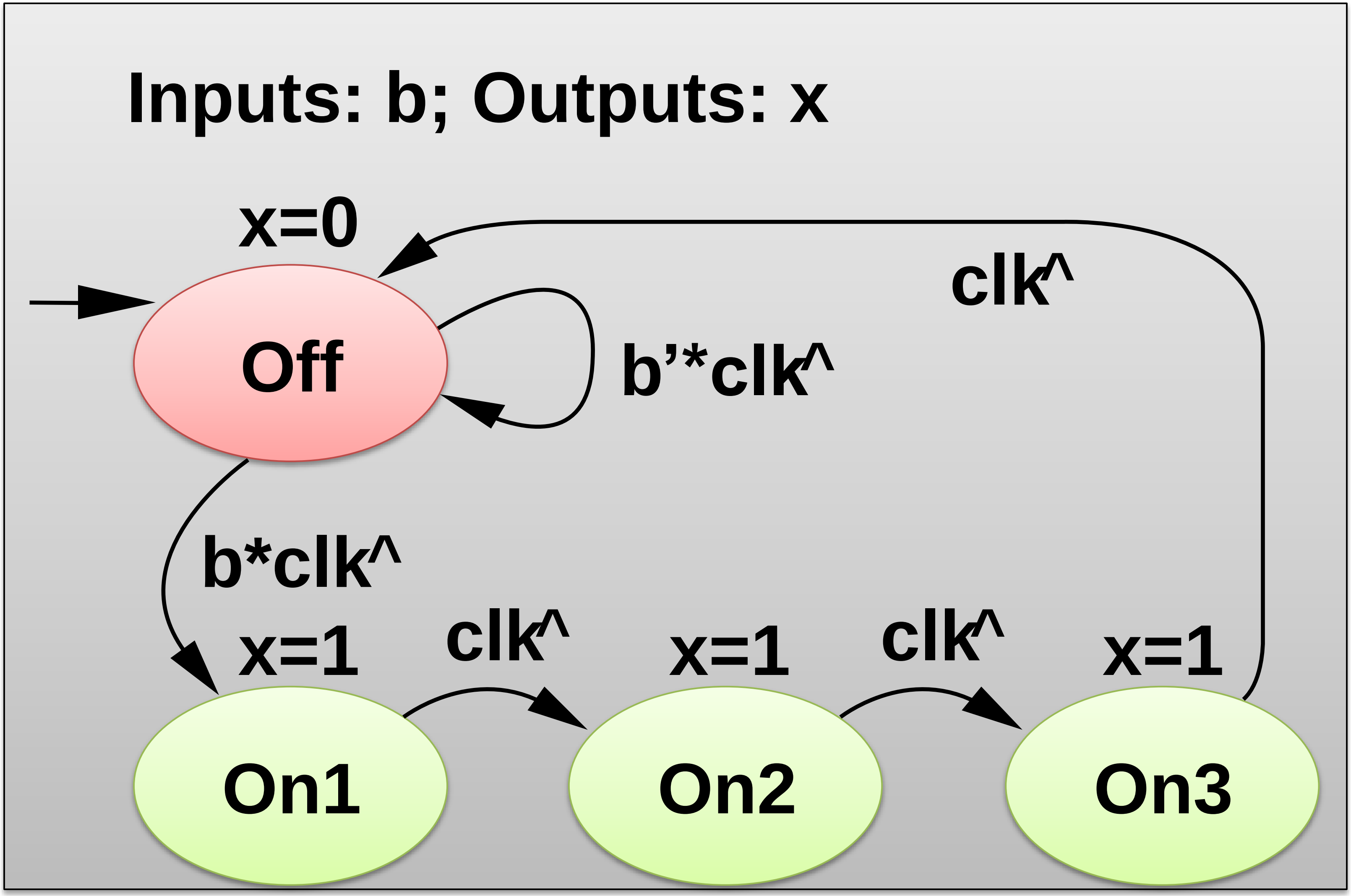
Extend FSM to Three-Cycles High Laser Timer

- Four states: Wait in “Off” state while b is 0 (b’)
- When b=1 (& rising clock edge), transition to On1
  - Sets X=1
  - On next two clock edges, transition to On2, then On3, which also set x=1
- So x=1 for three cycles after button pressed

Extend FSM to Three-Cycles High Laser Timer



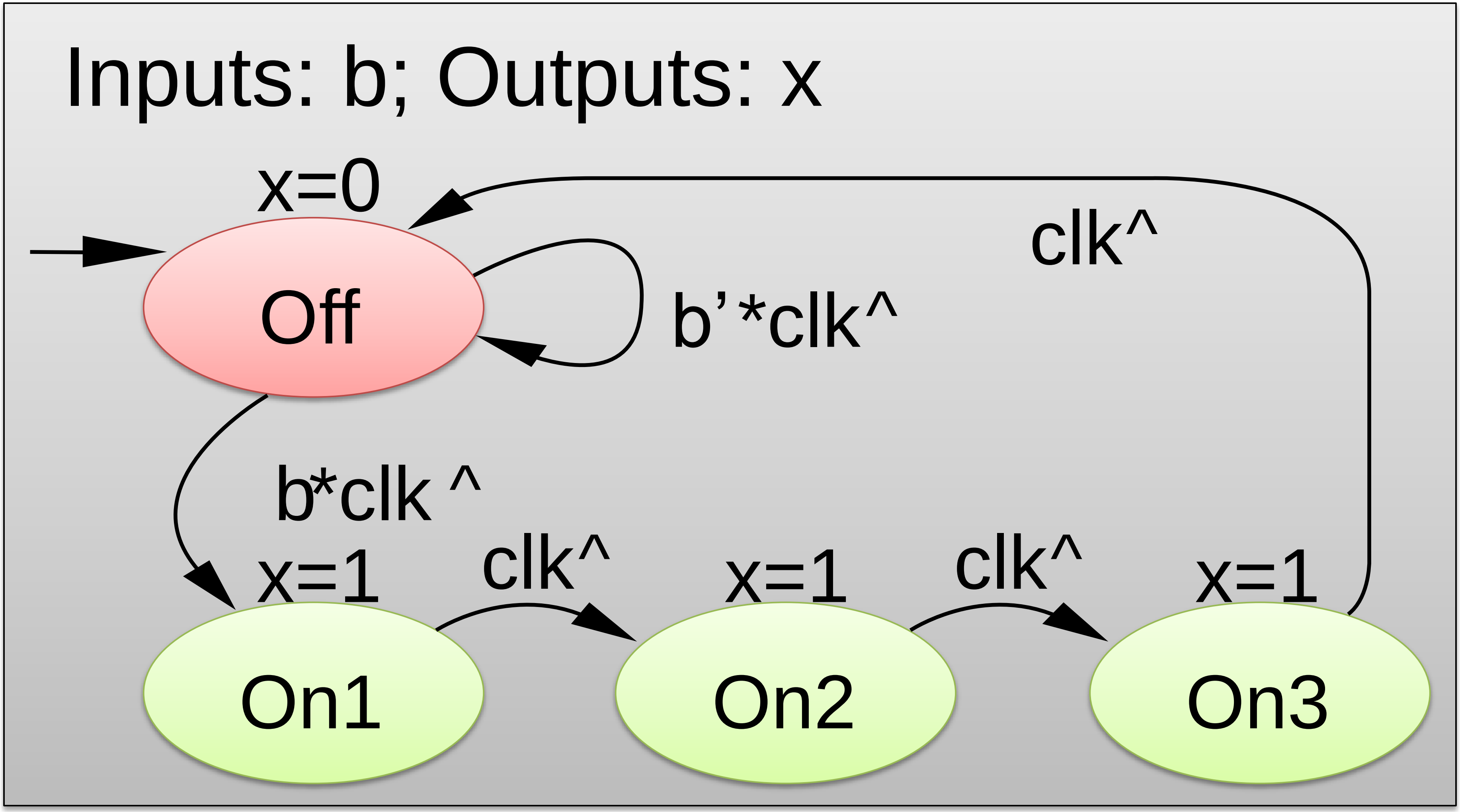
Extend FSM to Three-Cycles High Laser Timer



FSM Simplification: Rising Clock Edges Implicit

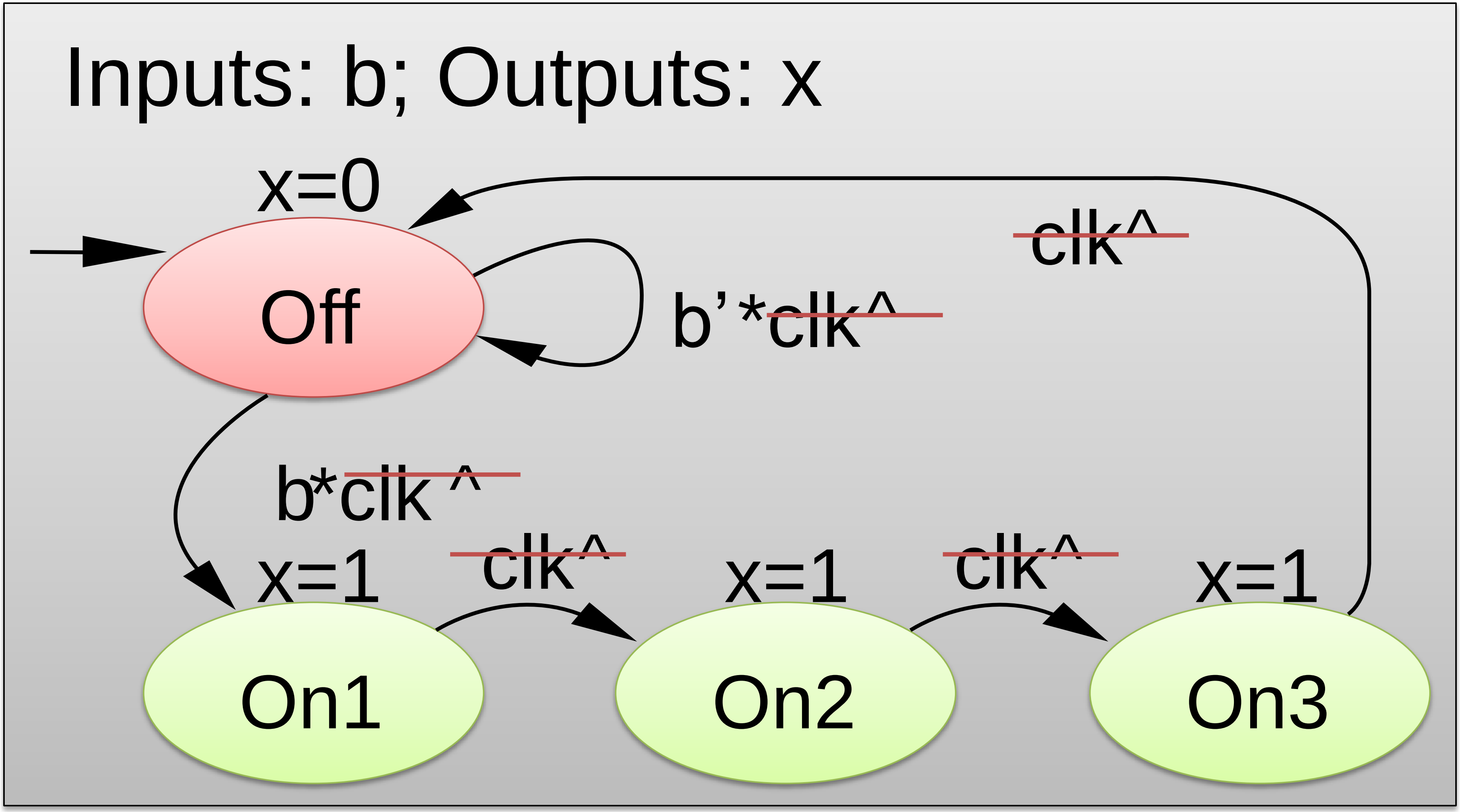
- Showing rising clock on every transition: cluttered
- Make implicit -- assume every edge has rising clock
- What if we wanted a transition *without* a rising edge
  - Asynchronous FSMs -- less common, and advanced topic
  - We consider ***synchronous*** FSMs
  - ***All*** **transition on rising edge**

FSM Simplification: Rising Clock Edges Implicit



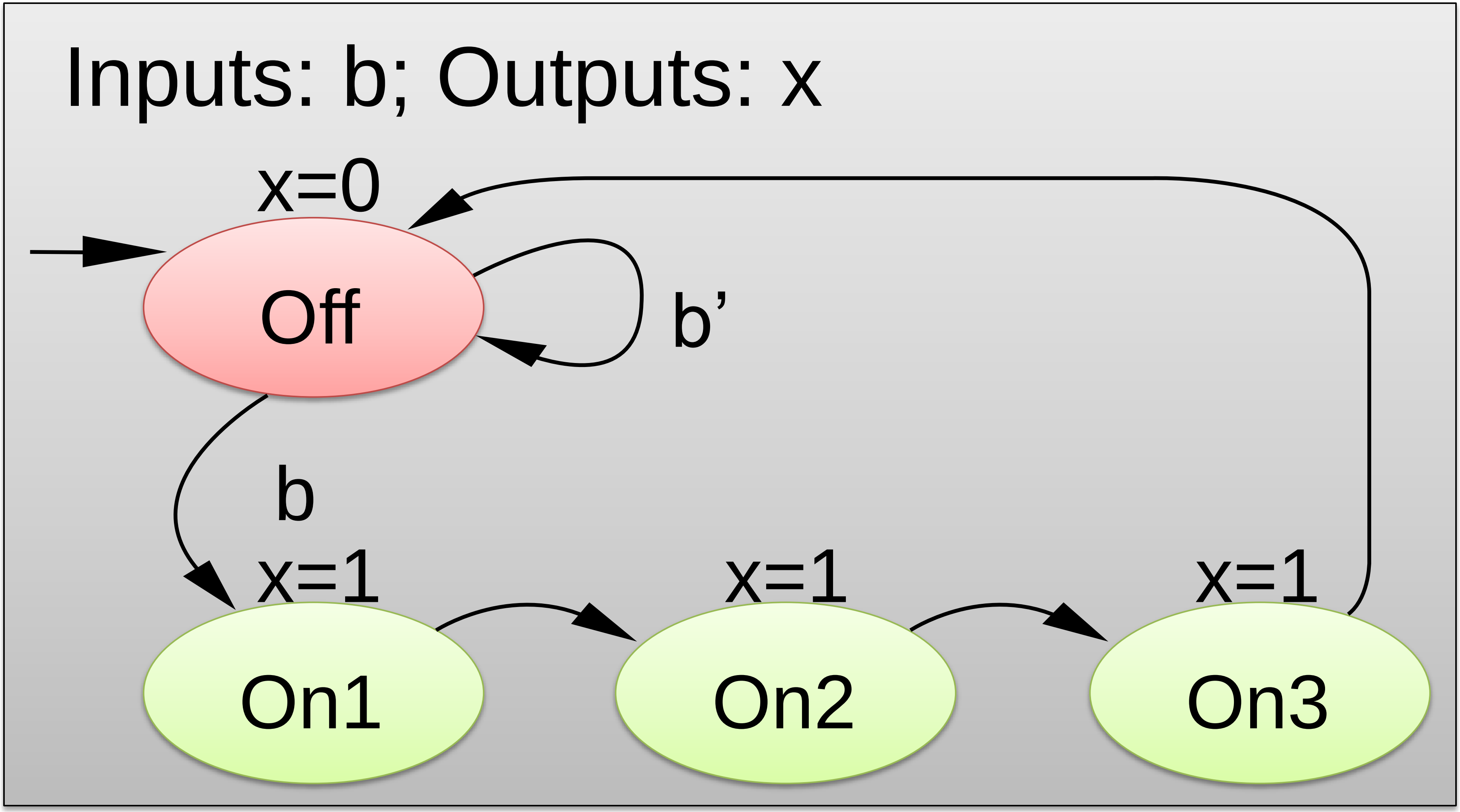
Note: Transition with no associated condition thus transistions to next state on next clock cycle

FSM Simplification: Rising Clock Edges Implicit



Note: Transition with no associated condition thus transistions to next state on next clock cycle

FSM Simplification: Rising Clock Edges Implicit



Note: Transition with no associated condition thus transistions to next state on next clock cycle