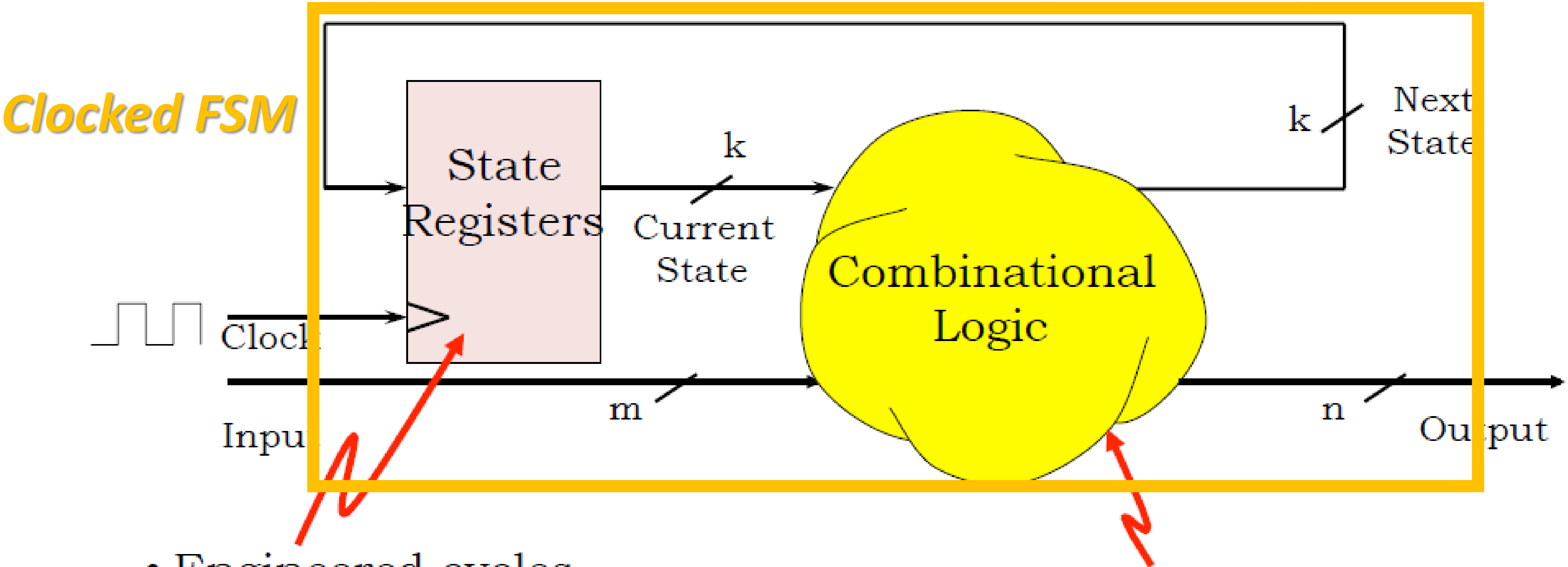
Synchronous FSM
(Finite State Machine)
Rehearsal

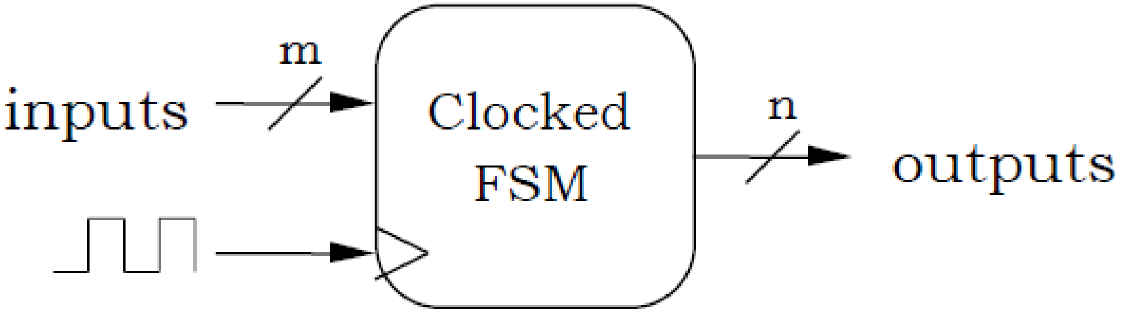
Synchronous FSM definition



- Engineered cycles
- Works only if dynamic discipline obeyed
- Remembers k bits for a total of 2^k unique combinations

- Acyclic graph
- Obeys static discipline
- Can be exhaustively enumerated by a truth table of 2^{k+m} rows and k+n output columns

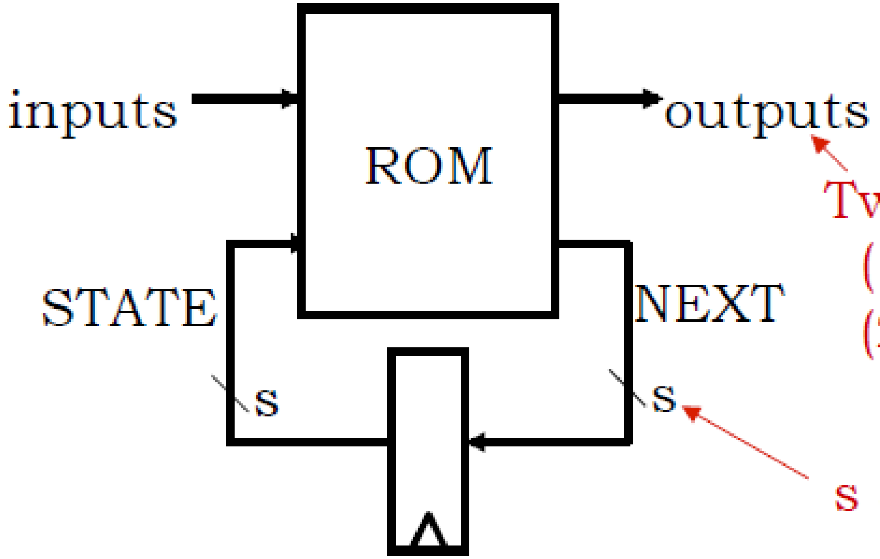
Synchronous FSM Abstraction



A FINITE STATE MACHINE has

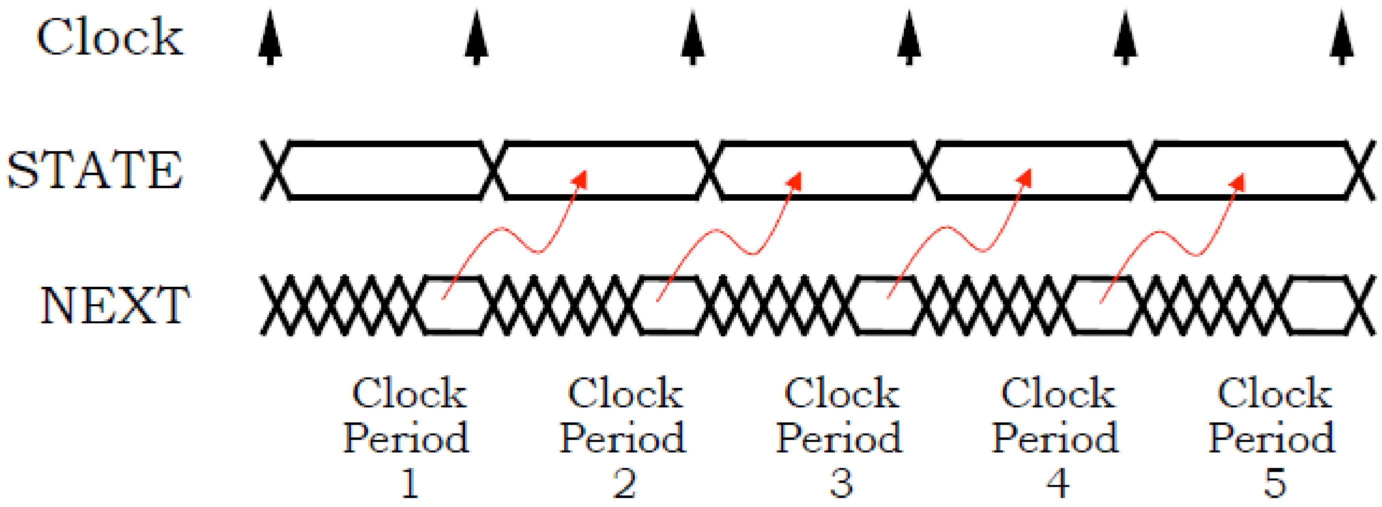
- k STATES: S_1 … S_k (one is “initial” state)
- m INPUTS: I_1 … I_m
- n OUTPUTS: O_1 … O_n
- Transition Rules: s’(s, I) for each state s and input I
- Output Rules: Out(s) or Out(s, I) for each state s and input I

Synchronous FSM - Discrete State, Discrete Time

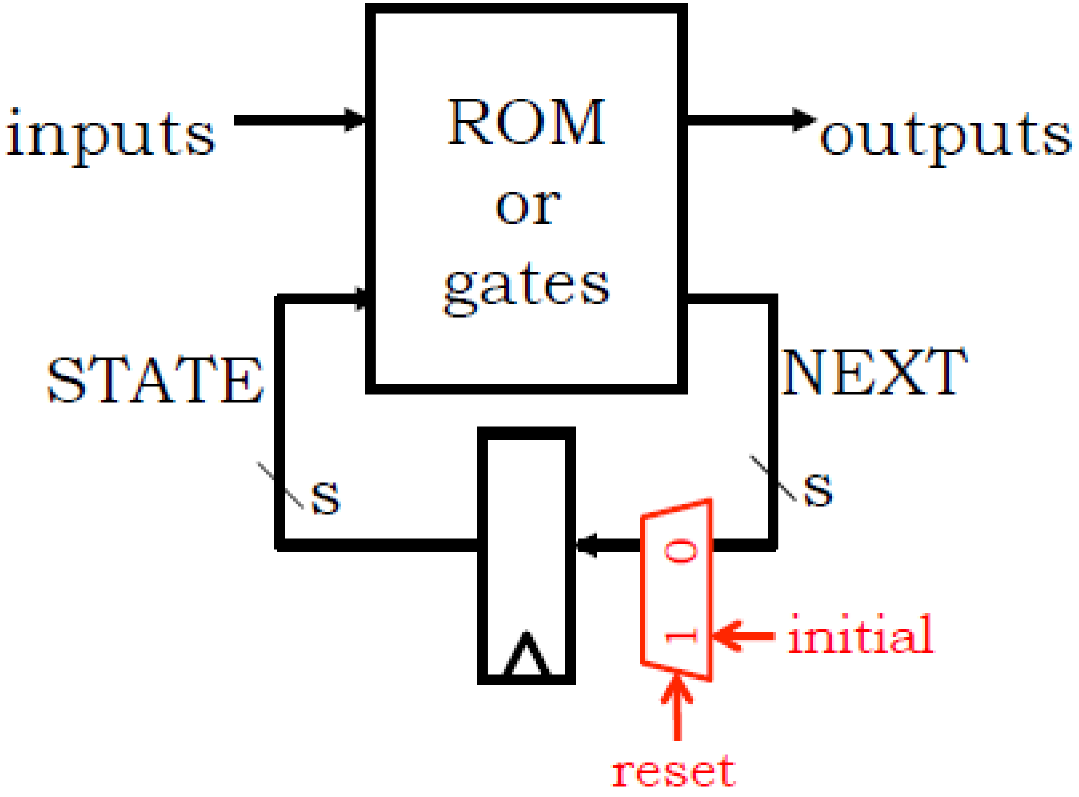


Two design choices:
(1) outputs only depend on state (Moore)
(2) outputs depend on inputs + state (Mealy)

s state bits ⇒ 2^s possible states

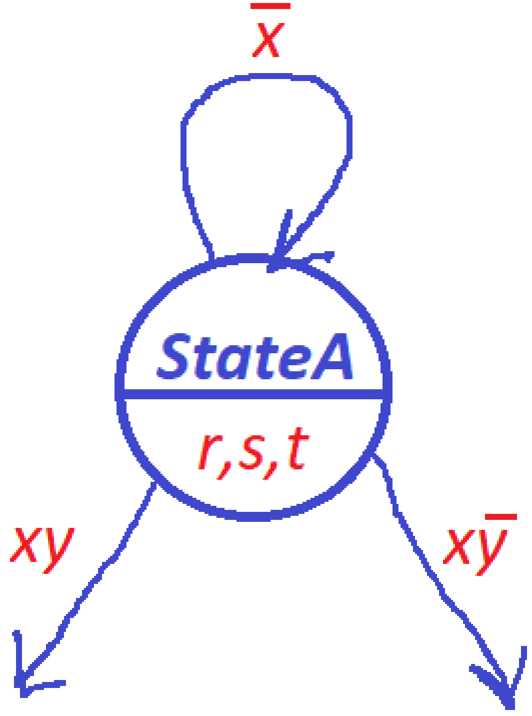


Synchronous FSM - Initialization

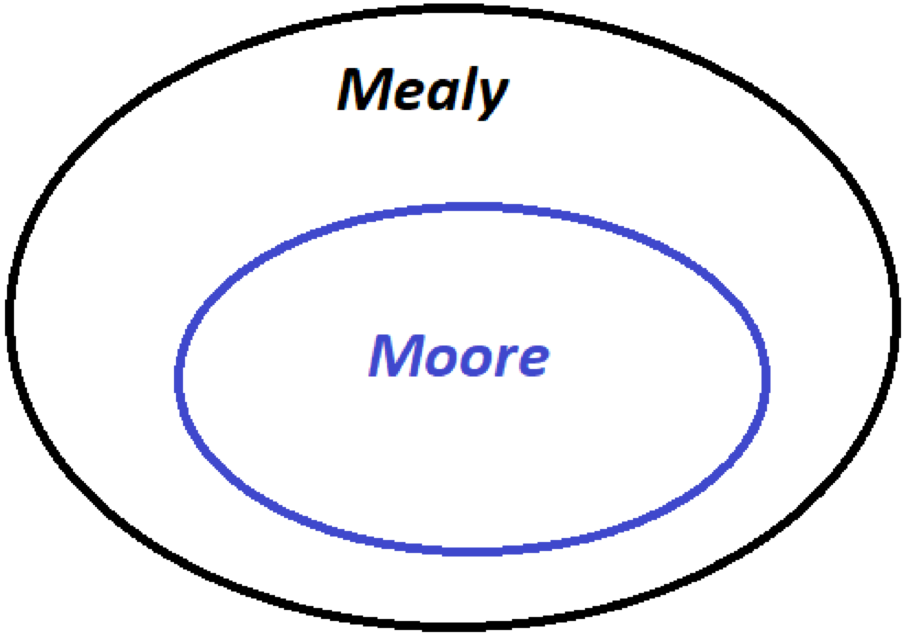


Mealy vs. Moore Machines

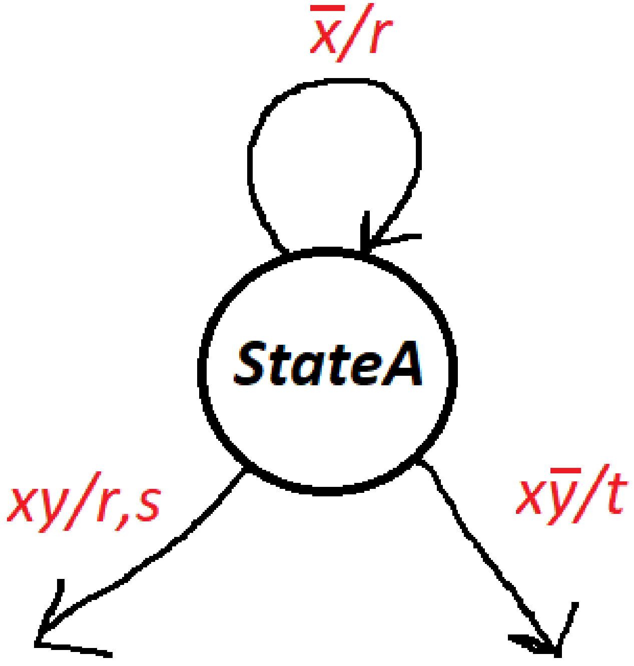
Moore



input=(x,y)
output=(r,s,t)



Mealy

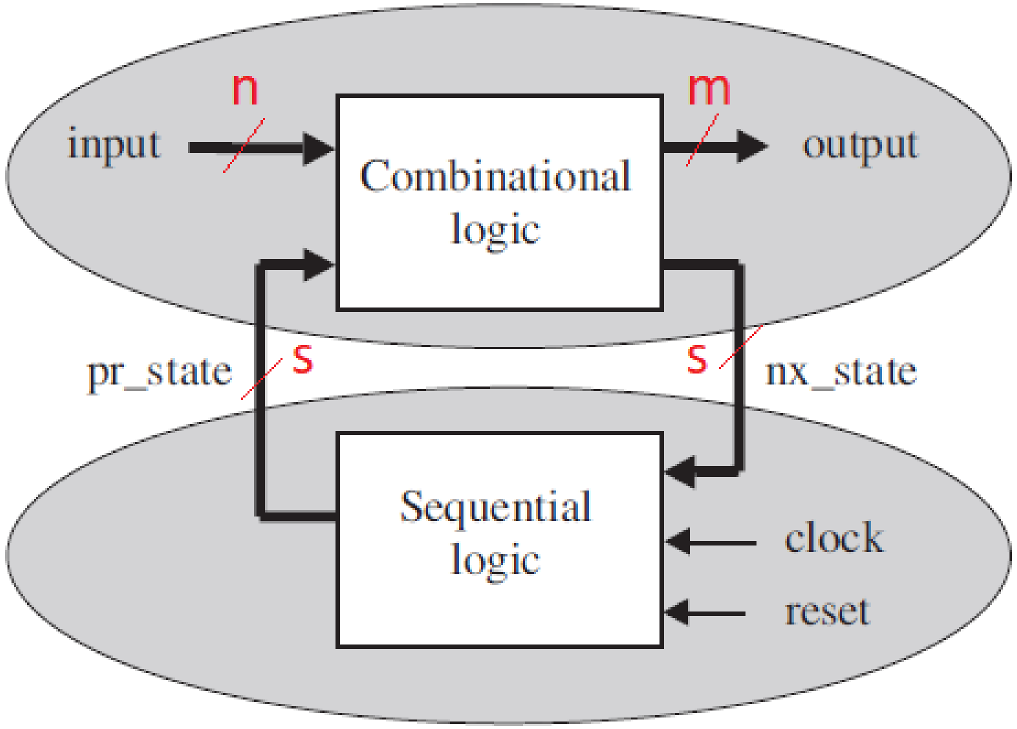


input=(x,y)
output=(r,s,t)

©Hanan Ribo

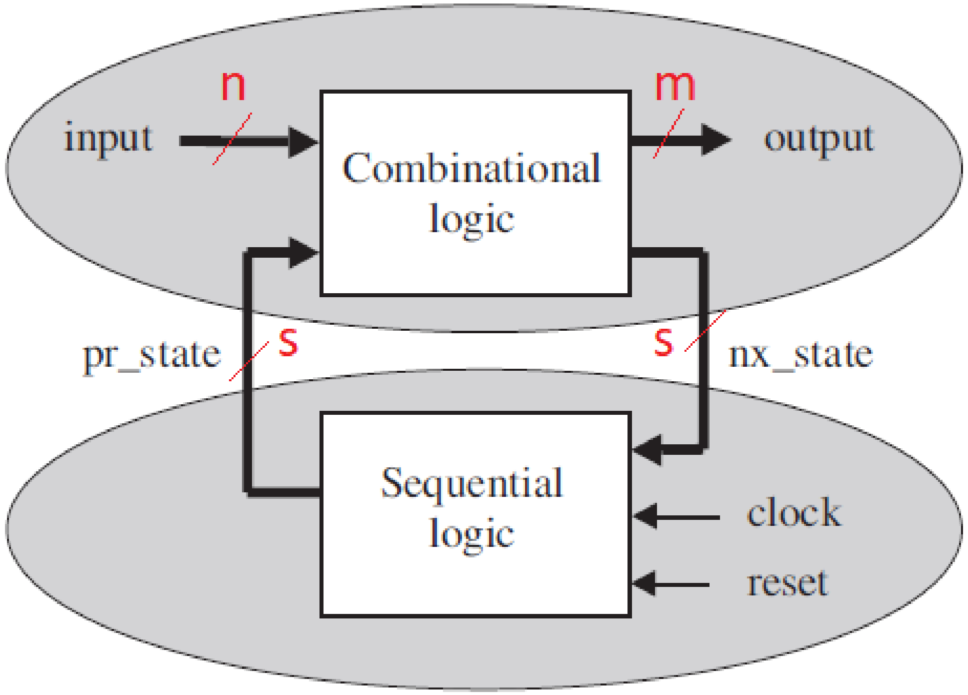
Mealy vs. Moore Machines

Moore



$$output = f(PS)$$
$$NS = g(input\,, PS)$$

Mealy



$$output = f(input\,, PS)$$
$$NS = g(input\,, PS)$$

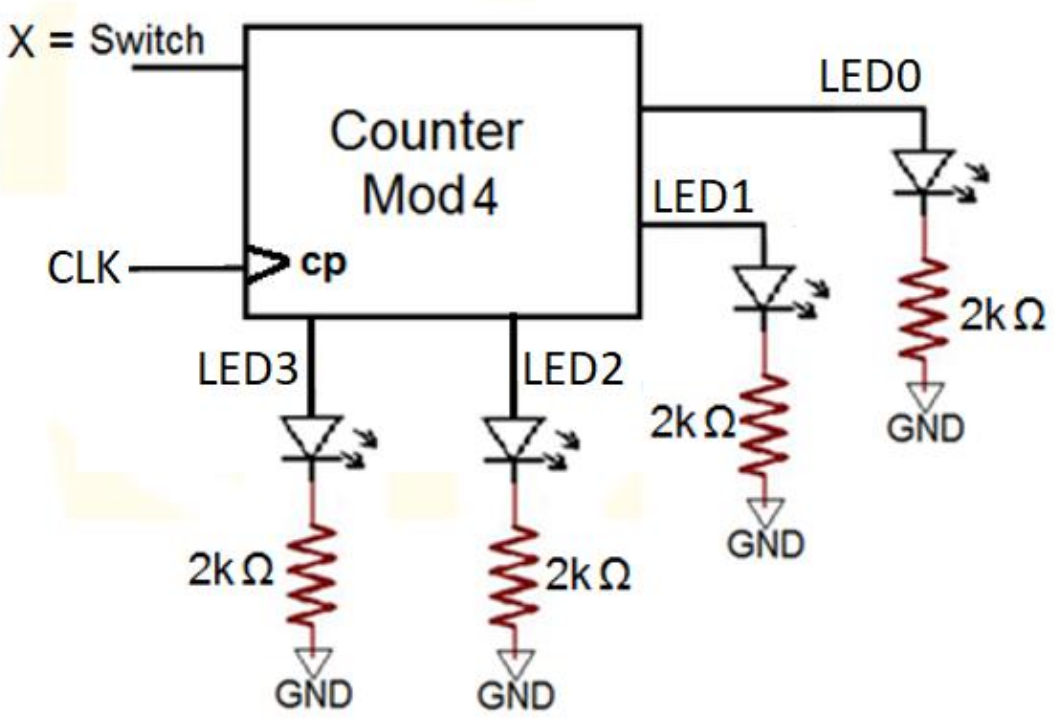
©Hanan Ribo

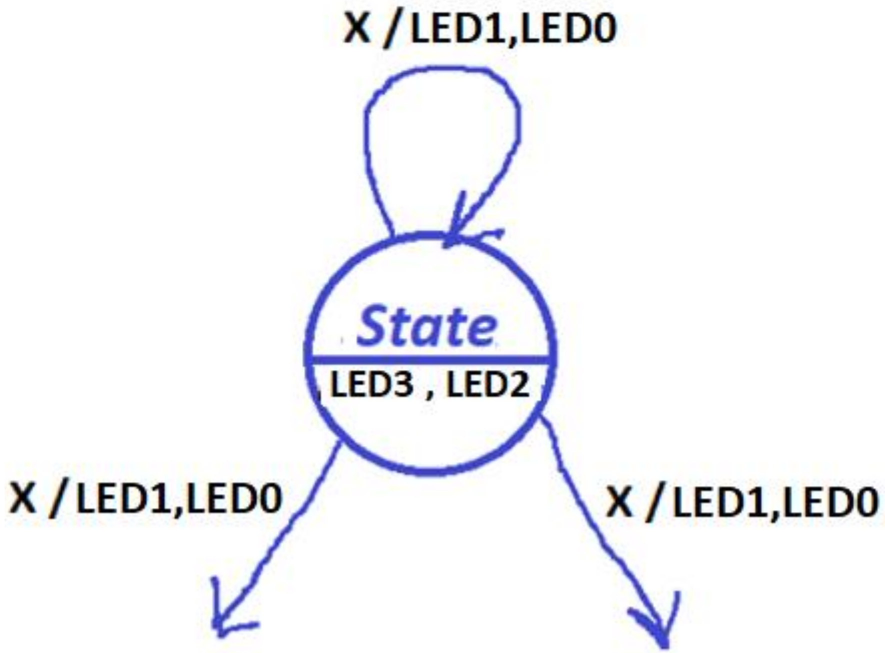
FSM design Example

FSM design Example

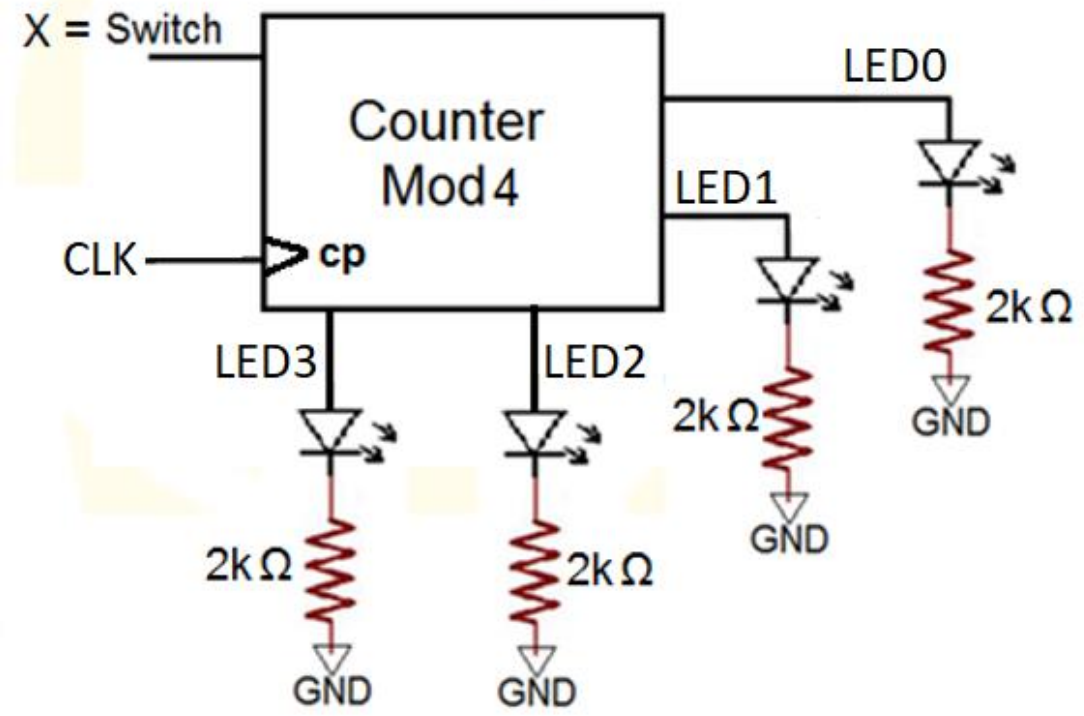
You are required to design Up/Down modulo 4 counter with the next spec:

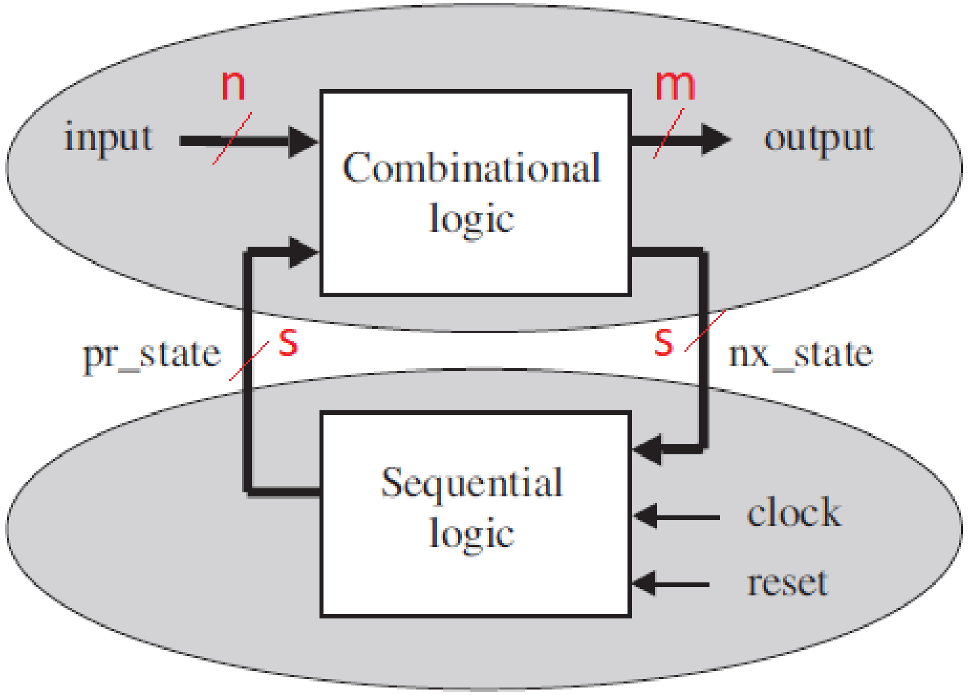
- Input X uses as Up/Down selector (X=‘0’ Up counter, X=‘1’ Down counter).
- Output LED3,LED2 show the counter value.
- Output LED0 is set only on counter value transition from “00” to “11”.
- Output LED1 is set only on counter value transition from “11” to “00”.

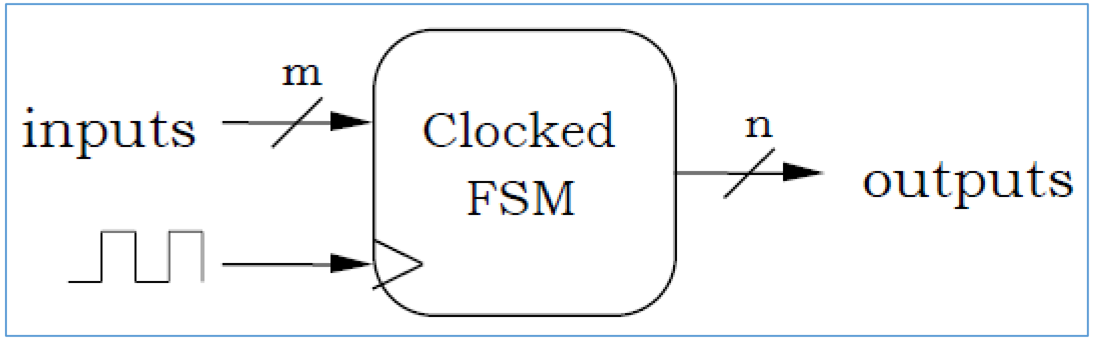




FSM design Example







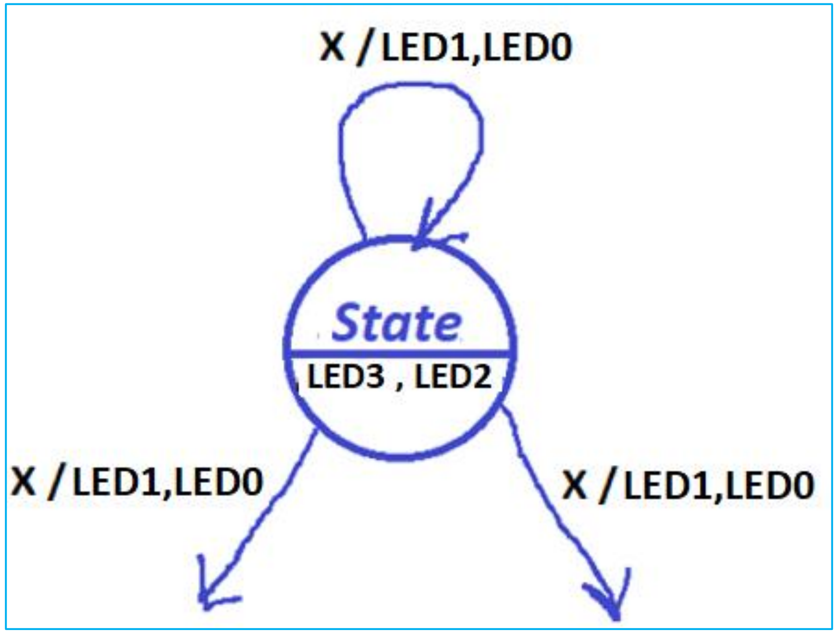
$$(LED3,LED2) = f(PS)$$

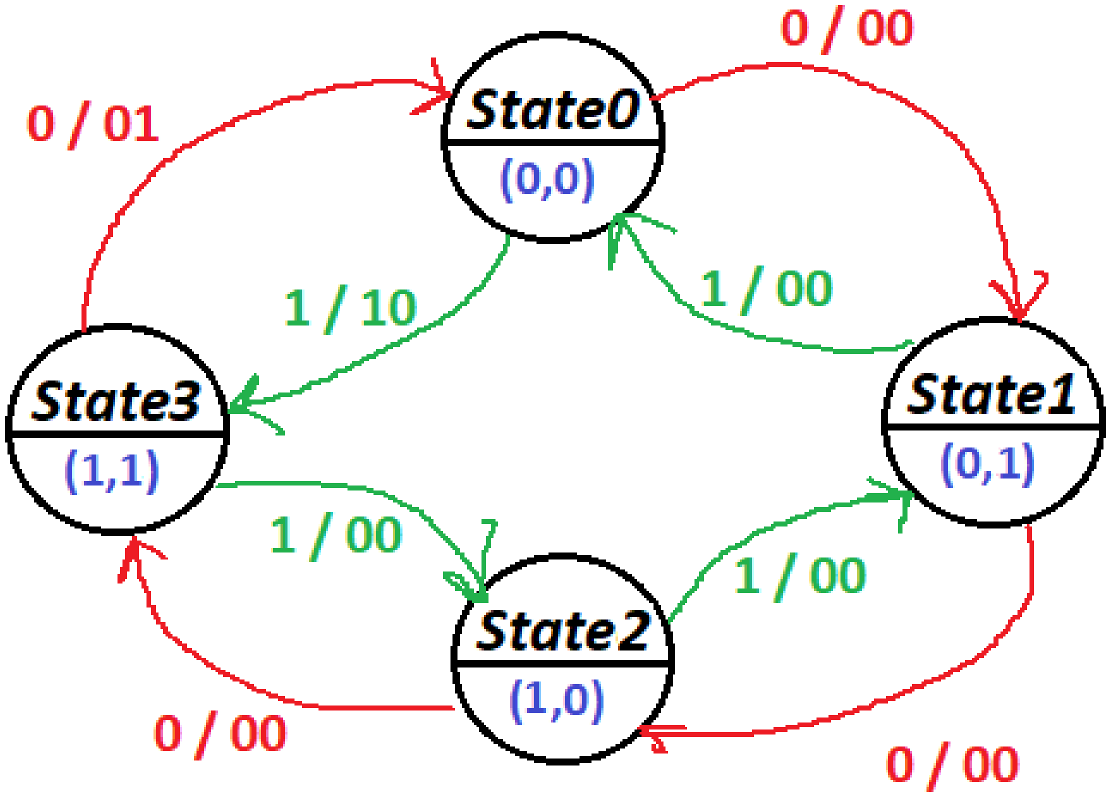
$$(LED1,LED0) = f(X\,,PS)$$

$$NS = g(X\,,PS)$$

Phase 1 – Drawing of States Diagram

Symbolic Legend:





***Thumb rule**: The transitions condition from each state out are **disjoint** and **complement**.*

For example:

State1 transitions condition out are: x=1/00 , x=0/00 therefore $x \cdot \overline{x} = 0$ *(disjoint)* and $x + \overline{x} = 1$ *(complement)*

©Hanan Ribo

Phase 2 – Primitive Transitions Table

Input X PS	X=0	X=1
State0	NS/LED1,LED0 = *State1/00*	NS/LED1,LED0 = *State3/10*
State1	NS/LED1,LED0 = *State2/00*	NS/LED1,LED0 = *State0/00*
State2	NS/LED1,LED0 = *State3/00*	NS/LED1,LED0 = *State1/00*
State3	NS/LED1,LED0 = *State0/01*	NS/LED1,LED0 = *State2/00*

$$PS \underbrace{\rightarrow}_{input\ X} NS\ /\ Output$$

This primitive table cannot be reduced (*recall that two states are identical iff their NS and Output are identical for the same input value*).

Phase 3 – States Encoding and Transitions table

State	Binary
State0	00
State1	01
State2	10
State3	11

$$PS \triangleq Present\ State = (y_2, y_1)$$

$$NS \triangleq Next\ State = (Y_2, Y_1)$$

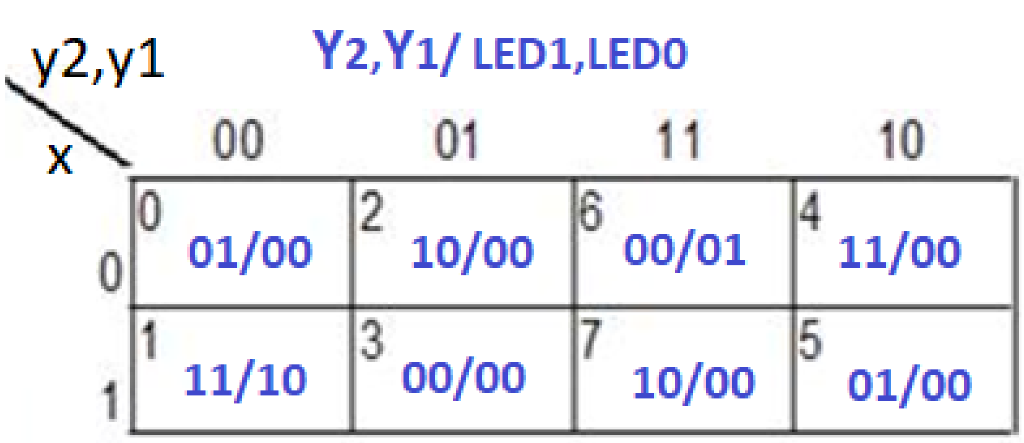
Input X PS	X=0	X=1
State0	NS/LED1,LED0 = *State1/00*	NS/LED1,LED0 = *State3/10*
State1	NS/LED1,LED0 = *State2/00*	NS/LED1,LED0 = *State0/00*
State2	NS/LED1,LED0 = *State3/00*	NS/LED1,LED0 = *State1/00*
State3	NS/LED1,LED0 = *State0/01*	NS/LED1,LED0 = *State2/00*

X PS	X=0	X=1
00	*01/00*	*11/10*
01	*10/00*	*00/00*
10	*11/00*	*01/00*
11	*00/01*	*10/00*

Phase 4 – Implementation

Input X PS	X=0	X=1
State0	NS/LED1,LED0 = *State1/00*	NS/LED1,LED0 = *State3/10*
State1	NS/LED1,LED0 = *State2/00*	NS/LED1,LED0 = *State0/00*
State2	NS/LED1,LED0 = *State3/00*	NS/LED1,LED0 = *State1/00*
State3	NS/LED1,LED0 = *State0/01*	NS/LED1,LED0 = *State2/00*

X PS	X=0	X=1
00	*01/00*	*11/10*
01	*10/00*	*00/00*
10	*11/00*	*01/00*
11	*00/01*	*10/00*



Y2,Y1/ LED1,LED0

x \ y2,y1	00	01	11	10
0	0 01/00	2 10/00	6 00/01	4 11/00
1	1 11/10	3 00/00	7 10/00	5 01/00

$$PS \triangleq Present\ State = (y_2, y_1)$$

$$NS \triangleq Next\ State = (Y_2, Y_1)$$

Phase 4 – Implementation

$PS \triangleq (y_2, y_1)$

$NS \triangleq (Y_2, Y_1)$

Y2,Y1 / LED1 LED0

x \ y2,y1	00	01	11	10
0	0 01/00	2 10/00	6 00/01	4 11/00
1	1 11/10	3 00/00	7 10/00	5 01/00

$LED0 = y_2 \cdot y_1 \cdot \bar{x}$

x \ y2,y1	00	01	11	10
0	0 0	2 0	6 1	4 0
1	1 0	3 0	7 0	5 0

$LED1 = \overline{y_2} \cdot \overline{y_1} \cdot x$

x \ y2,y1	00	01	11	10
0	0 0	2 0	6 0	4 0
1	1 1	3 0	7 0	5 0

$Y_2 = x \oplus y_2 \oplus y_1$

x \ y2,y1	00	01	11	10
0	0 0	2 1	6 0	4 1
1	1 1	3 0	7 1	5 0

$Y_1 = \overline{y_1}$

x \ y2,y1	00	01	11	10
0	0 1	2 0	6 0	4 1
1	1 1	3 0	7 0	5 1

©Hanan Ribo

Phase 4 – Implementation

Tradeoff:

#DFFs

vs.

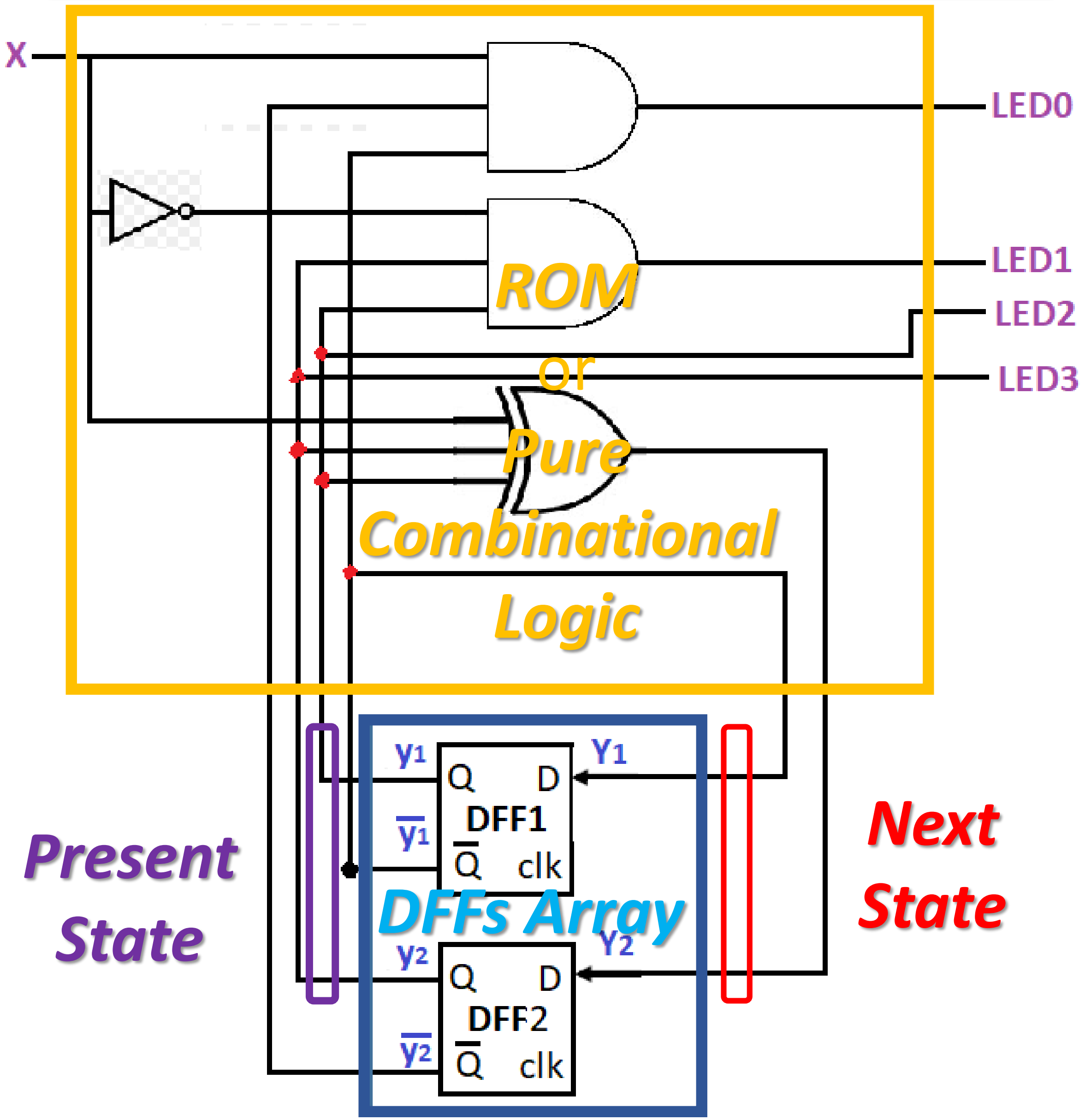
FANin and FANout size

Explanation:

Depending the States Encoding Allocation (Binary Encoding or Direct Encoding)

and

The target HW (LUT based with low FANin but DFFs abundantly as with FPGA or the opposite with ASIC).


X
LED0
LED1
LED2
LED3
ROM
or
Pure
Combinational
Logic
y1
Y1
Q
D
DFF1
y̅1
Q̅
clk
DFFs Array
y2
Y2
Q
D
DFF2
y̅2
Q̅
clk
Present
State
Next
State