State Machine

- It is a concept which can be employed to design any digital system in a generalized way.
- State machine is a system or circuit which has several states in which it can exist and it may has clearly defined a set of inputs which will steer the system from one state to another state and each state also may produce an output.

Characteristics of state machine:

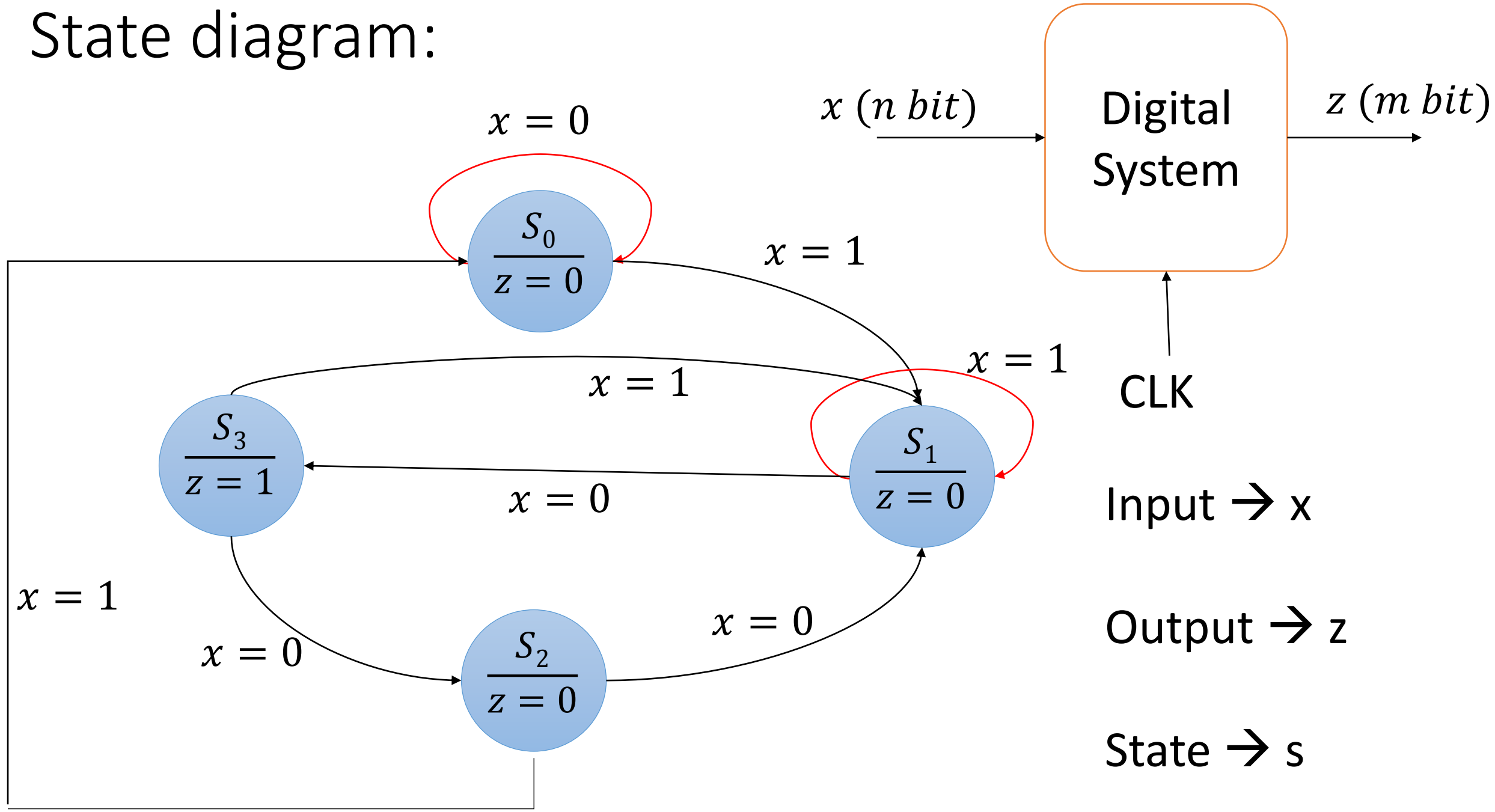
- The next state of state machine always depends on previous state.
- There may be or may not be external inputs
- There will be definitely output. However, the state itself may be the output.
- In general, the behavior of the circuit depends on the present state and external inputs (optional).

Synchronous state machine

- Transition triggered by CLK
- Transition is determined by present state and may be also by external inputs.
- State transition is not necessary in each clock period.
- Output will be determined by the present state and external inputs.

Next_state = func(present_state, ext. inputs)
Output = func(present_state, ext. inputs)

State diagram:



Input → x

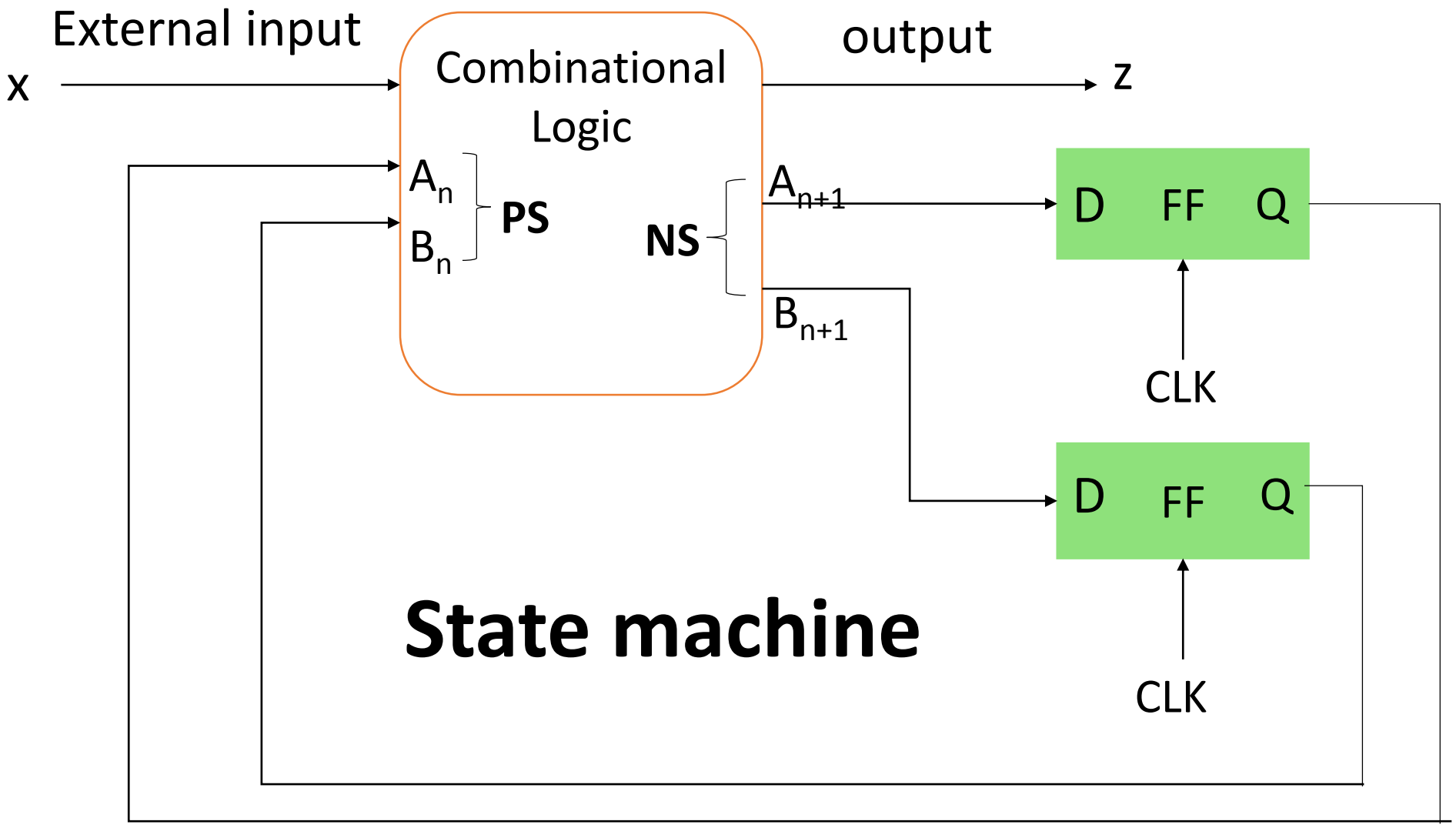
Output → z

State → s

Generalized model of state machine

4 states

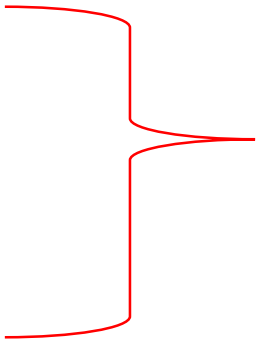
2 state variables → A B



Takeaway Points:

Any digital system can be modelled as a state machine

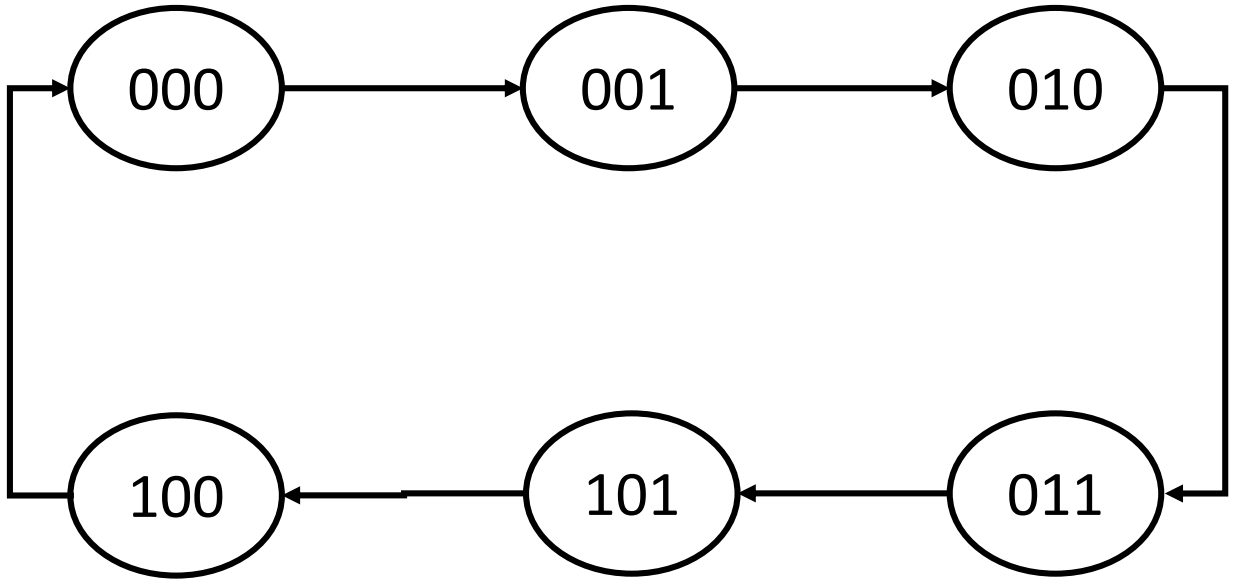
1. Combinational circuit (with one state)
2. Flip-flop (Two states)
3. Counter



State machine

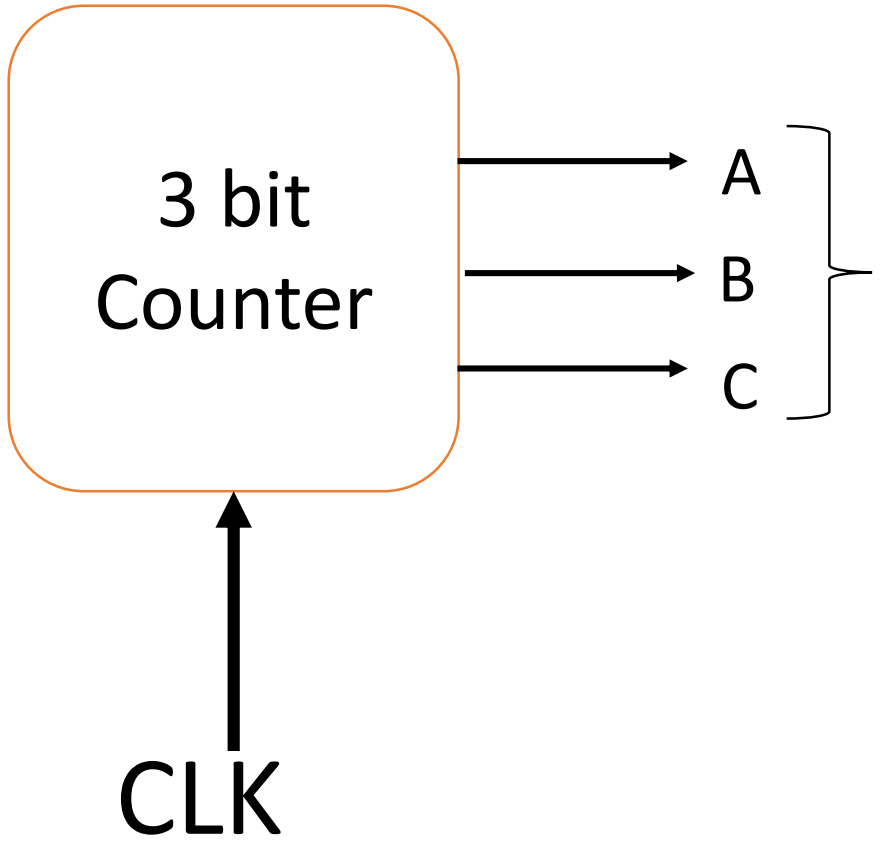
Special scenario: Counter → State machine

- Lets consider a 3 bit (mod-6 counter)



No external inputs

State variables → output



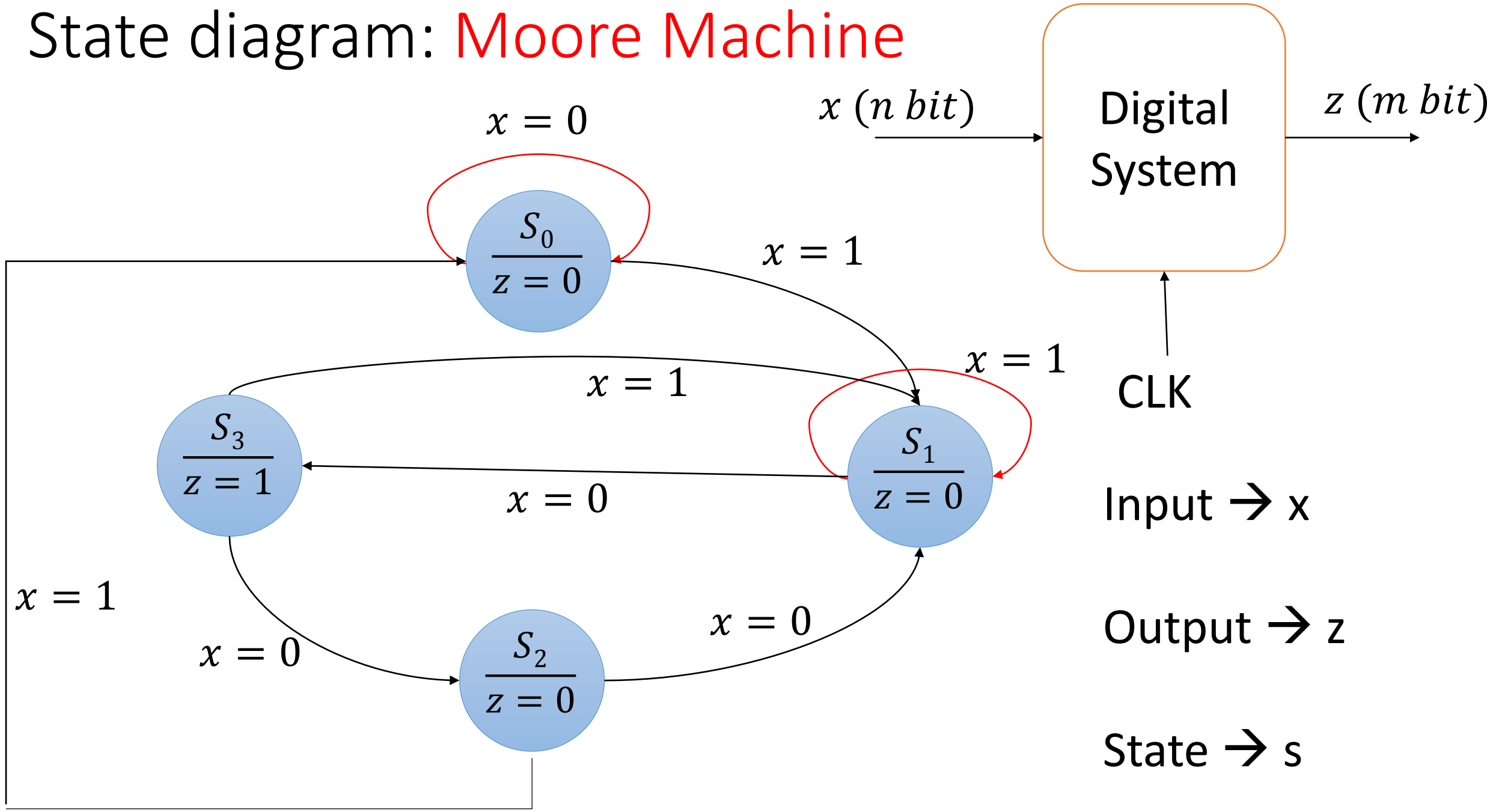
Design strategy: State machine

- Step 1: System/circuit behavior will be given to you as a specification.
- Step 2: Represent the given model specification using a state diagram (graph).
- Step 3: Implement the state graph using hardware.

Summary

- State machine: A design methodology
- Synchronous state machine → triggered by clock. However, circuit may stay at the same state for many clock cycle.
- A state machine may not have external input and output. Eg. Counter
- Asynchronous state machine → triggered by event. A little more difficult to analyze.
- Next we will look at how the given state graph can be translated into a hardware model.

State diagram: Moore Machine



State table of the state-diagram shown in last slide

State variables AB

$S_0 \rightarrow 00$

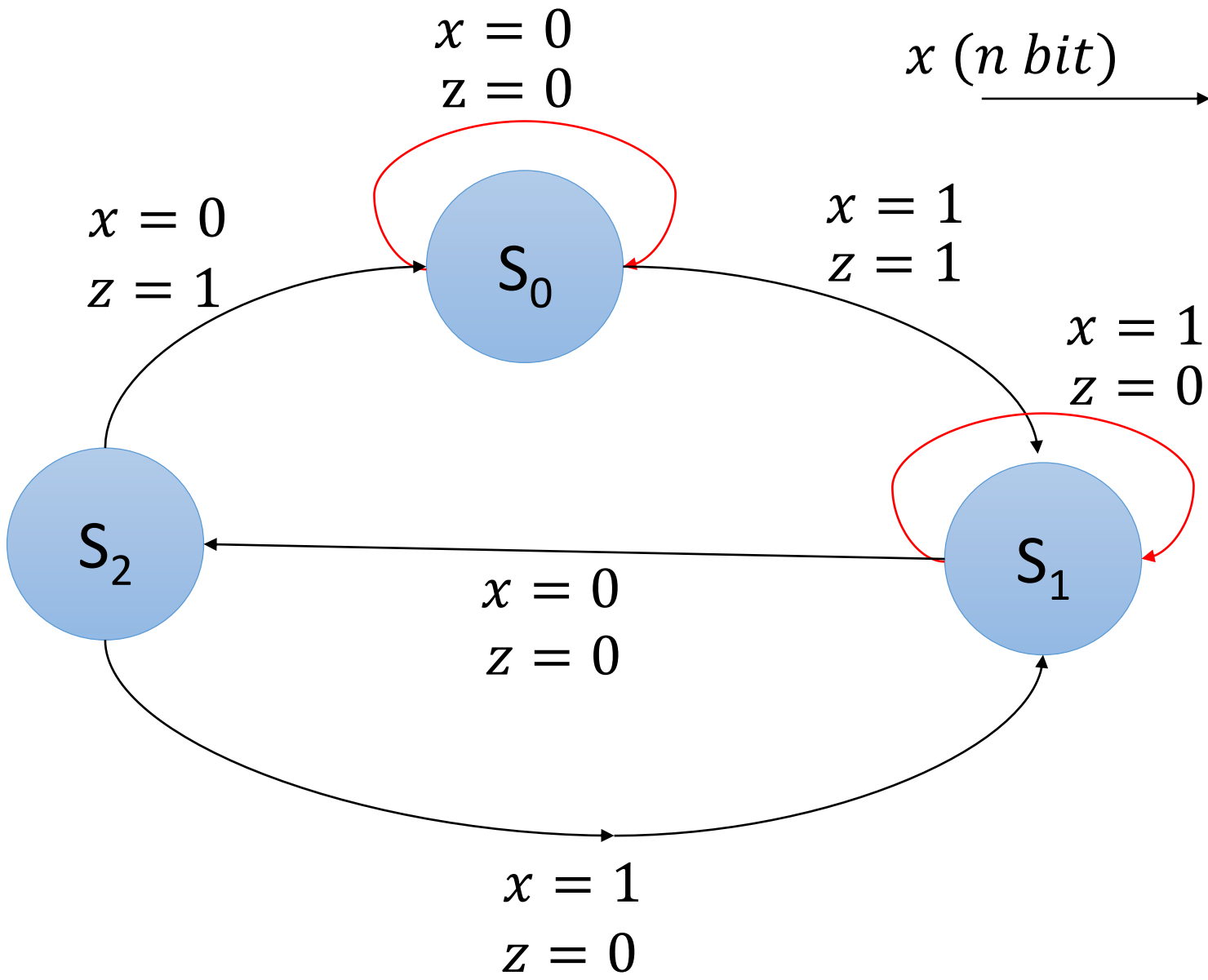
$S_1 \rightarrow 01$

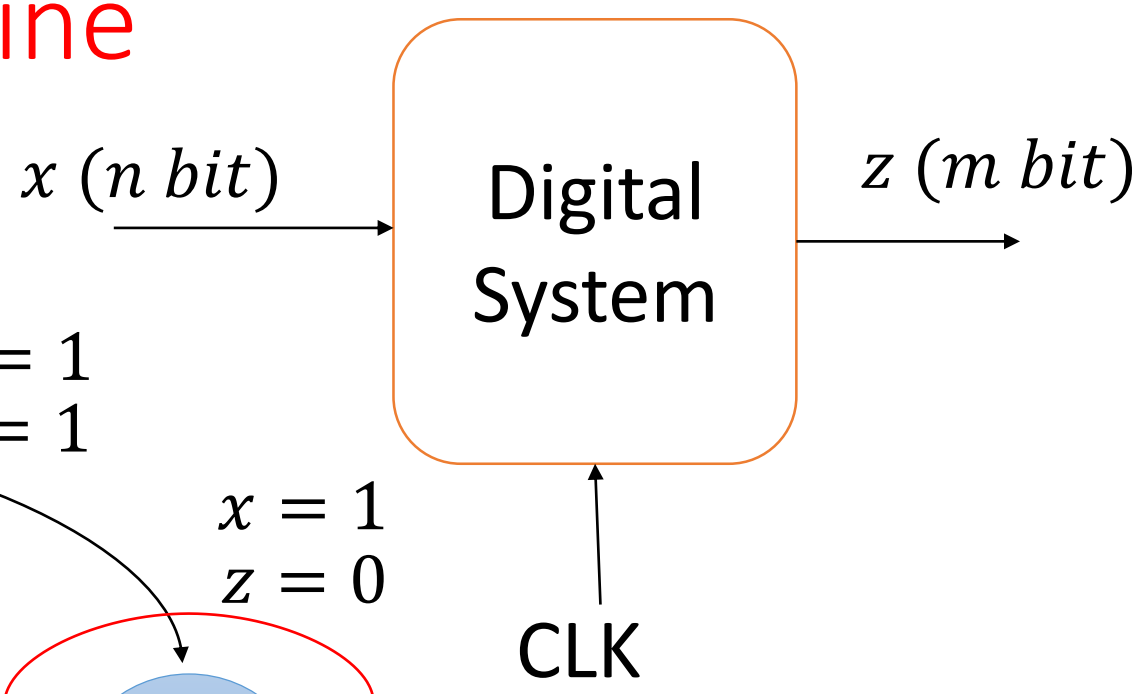
$S_2 \rightarrow 10$

$S_3 \rightarrow 11$

Present state (AB)	Input (X)	Next state (A'B')	Output (Z)
00	0	00	0
00	1	01	0
01	0	11	0
01	1	01	0
10	0	01	0
10	1	00	0
11	0	10	1
11	1	01	1

State diagram: Mealy Machine





Input → x

Output → z

State → S

State table of the state-diagram shown in last slide

State variables AB

$S_0 \rightarrow 00$

$S_1 \rightarrow 01$

$S_2 \rightarrow 10$

Present state (AB)	Input (X)	Next state (A'B')	Output (Z)
00	0	00	0
00	1	01	1
01	0	10	0
01	1	01	0
10	0	00	1
10	1	01	0
11	X	XX	0
11	X	XX	0

Summary

- State machine can be classified into two sub-classes based on the output.
- Steering combinational logic for computing the output will be determined as:

Output = func(present_state, ext. inputs) → Mealy machine

Output = func(present_state) → Moore machine