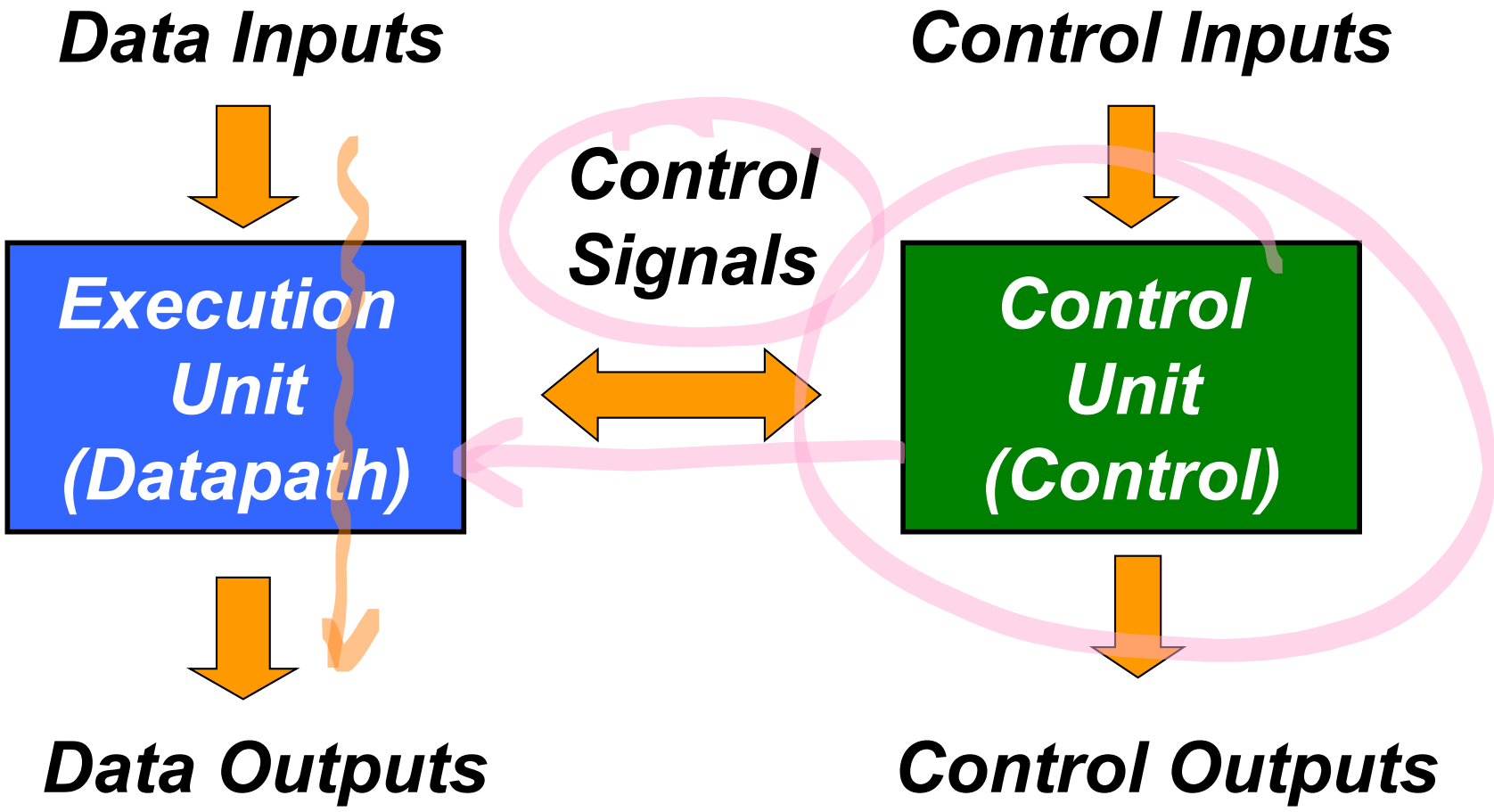
Finite State Machine

Structure of a Typical Digital System



Execution Unit (Datapath)

- Provides All Necessary Resources and Interconnects Among Them to Perform Specified Task
- Examples of Resources
  - Adders, Multipliers, Registers, Memories, etc.

Control Unit (Control)

- Controls Data Movements in an Operational Circuit by Switching Multiplexers and Enabling or Disabling Resources
- Follows Some 'Program' or Schedule
- Often Implemented as Finite State Machine

  or collection of Finite State Machines

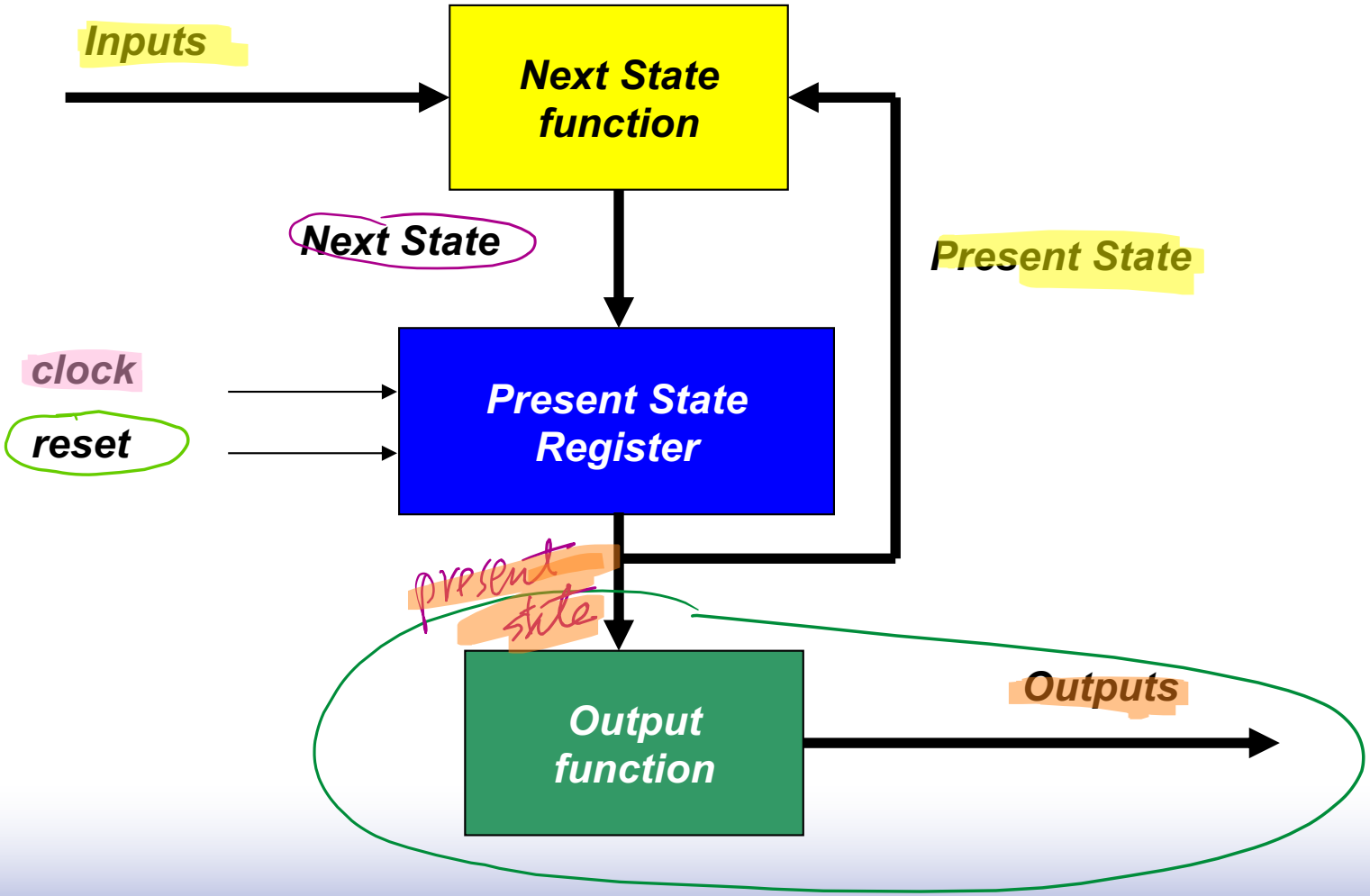
Finite State Machines (FSMs)

- Any Circuit with Memory Is a Finite State Machine
  - Even computers can be viewed as huge FSMs
- Design of FSMs Involves
  - Defining states
  - Defining transitions between states
  - Optimization / minimization
- Above Approach Is Practical for Small FSMs Only

Moore FSM

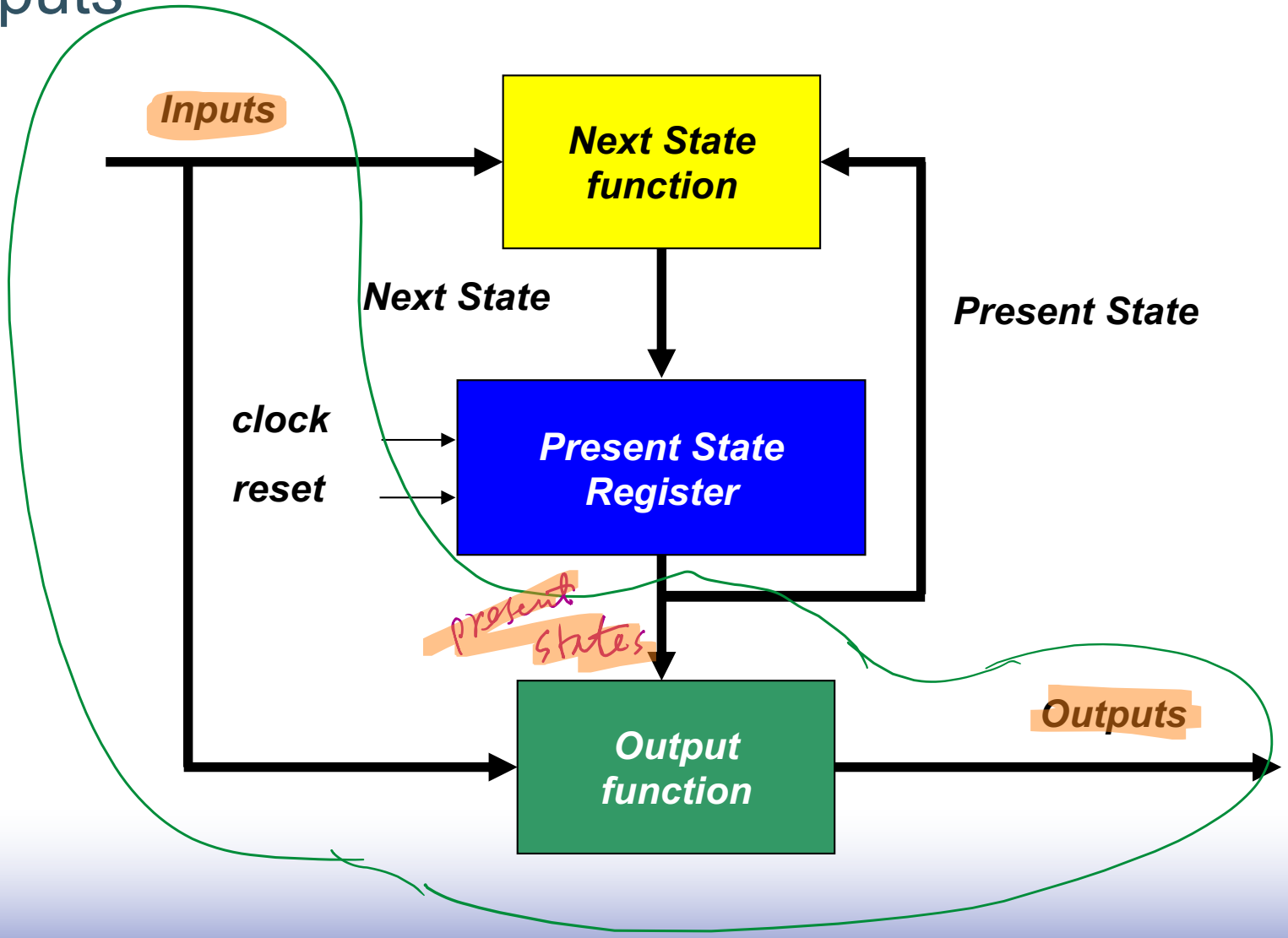
FSM < Moore (Faster) / Mealy (less hardware)

- Output Is a Function of a Present State Only



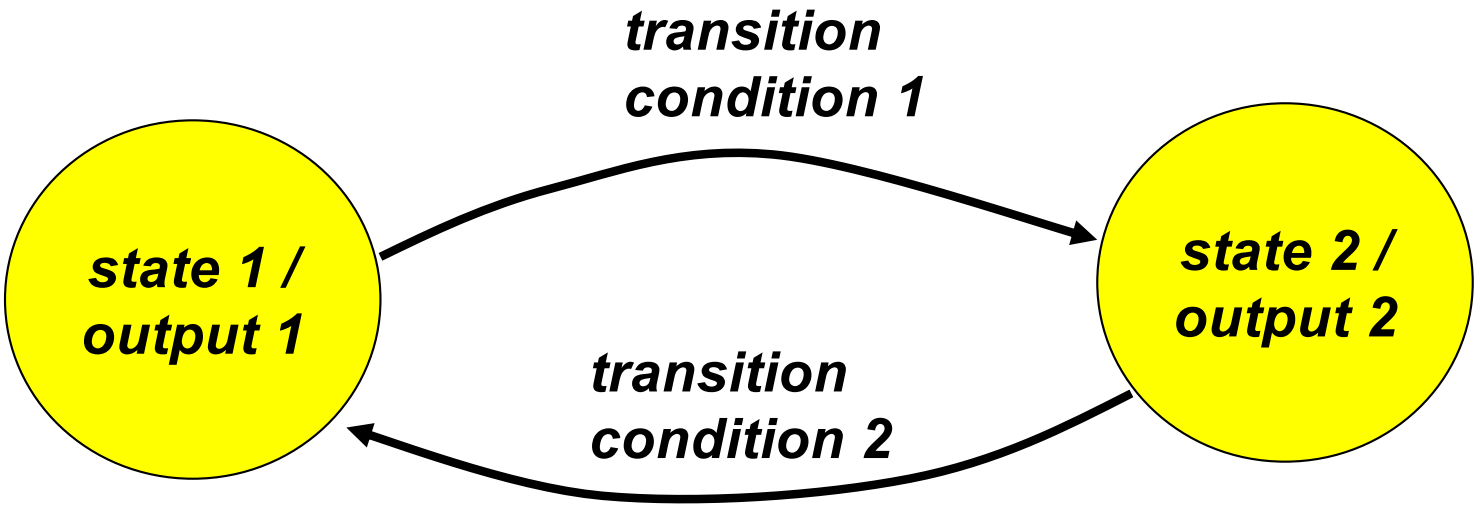
Mealy FSM

- Output Is a Function of a Present State and Inputs

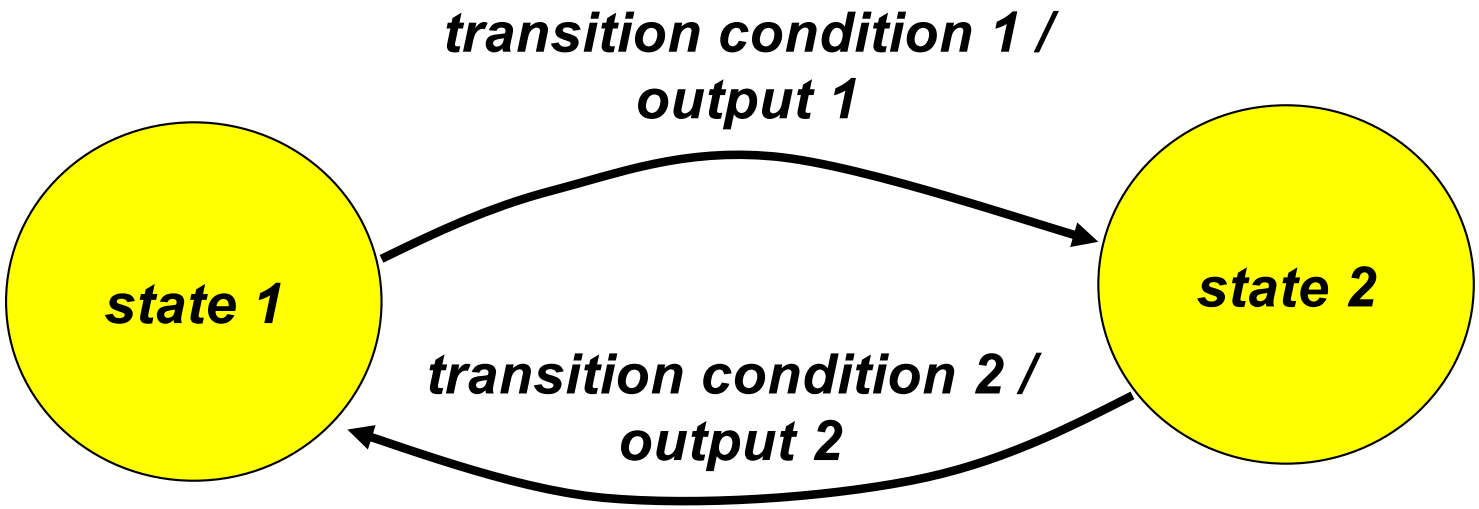


Moore Machine

STG (state Transition Graph)
- define states (# of FF's)
- define transition min. # of states



Mealy Machine



Moore vs. Mealy FSM (1)

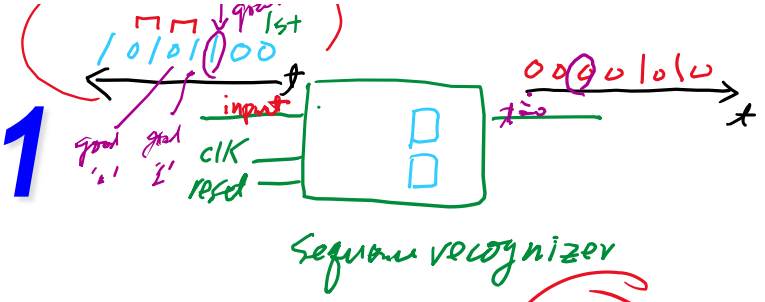
- Moore and Mealy FSMs Can Be Functionally Equivalent
  - Equivalent Mealy FSM can be derived from Moore FSM and vice versa
- Mealy FSM Has Richer Description and Usually Requires Smaller Number of States
  - Smaller circuit area

less hardware

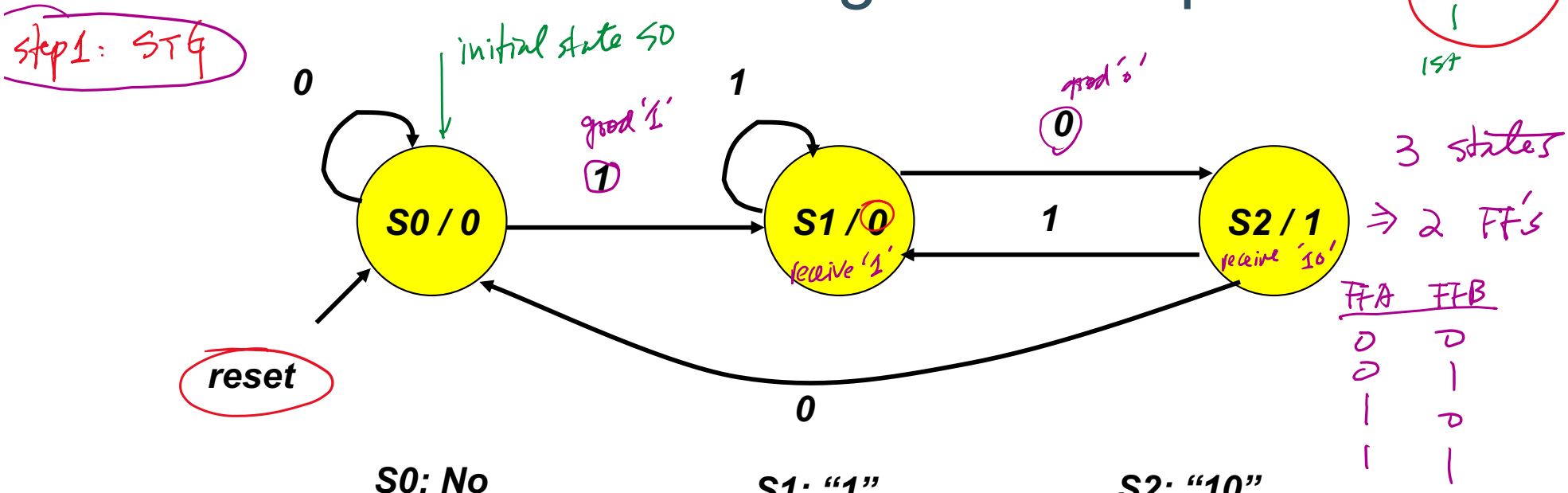
Moore vs. Mealy FSM (2)

- Mealy FSM Computes Outputs as soon as Inputs Change
  - Mealy FSM responds one clock cycle sooner than equivalent Moore FSM
- Moore FSM Has No Combinational Path Between Inputs and Outputs
  - Moore FSM is more likely to have a shorter critical path faster

Moore FSM - Example 1



- Moore FSM that Recognizes Sequence "10"



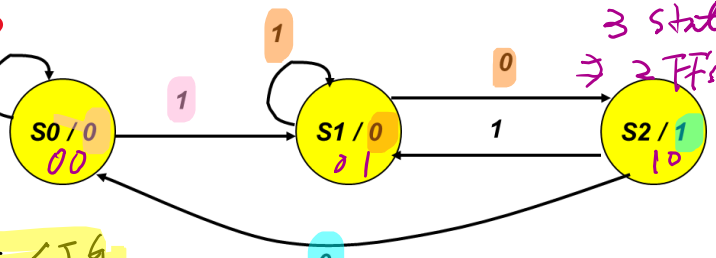
Meaning of states:

S0: No elements of the sequence observed

S1: "1" observed

S2: "10" observed

Moore



3 states ⇒ 2 FFs : A, B

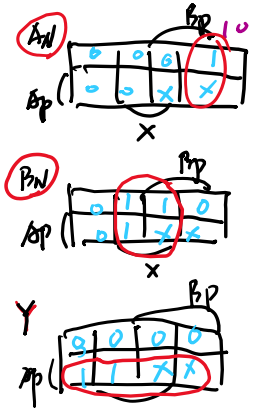
step 1: STG

step 2: state Table (truth table)

- state encoding, i.e. S0 = (00), S1 = (01), S2 = (10)

A_P	B_P	X	A_N	B_N	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	X	X	X
1	1	1	X	X	X

S0 = 00, S1 = 01, S2 = 10



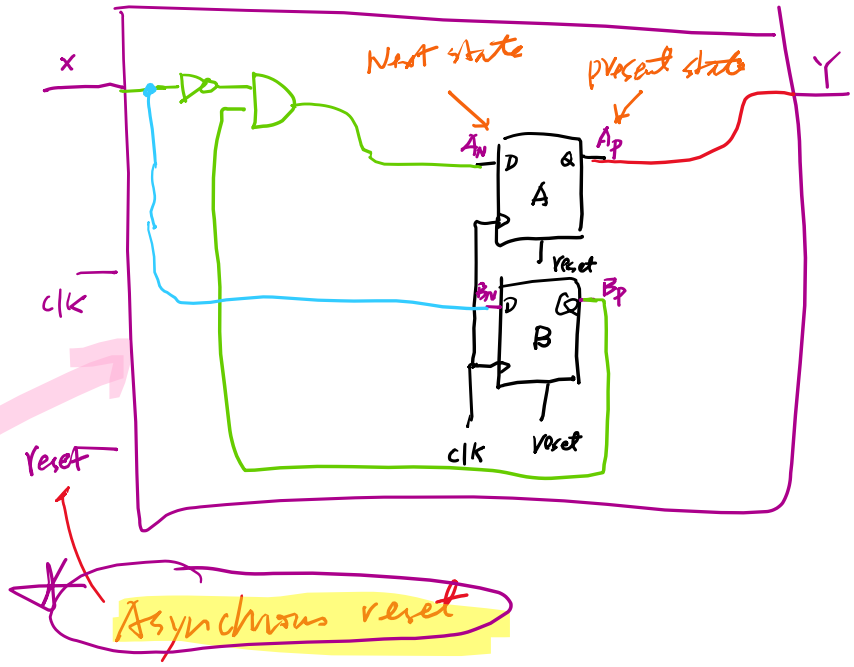
$B_N = B_P \cdot x'$

$B_N = x$

$Y = A_P$

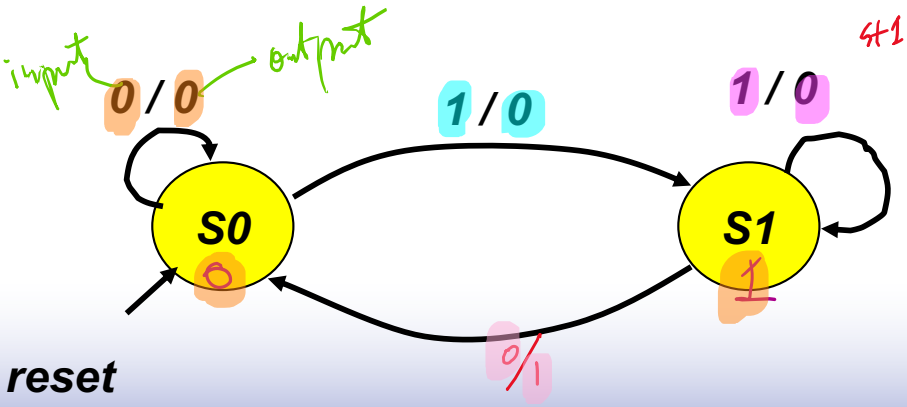
step 3: next state eq. & output eq

step 4: FSM logic Design



2 states ⇒ 1 FF — A

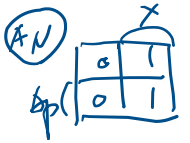
Mealy : state encoding S0 = (0), S1 = (1)



A_P	x	A_N	Y
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	0

S0

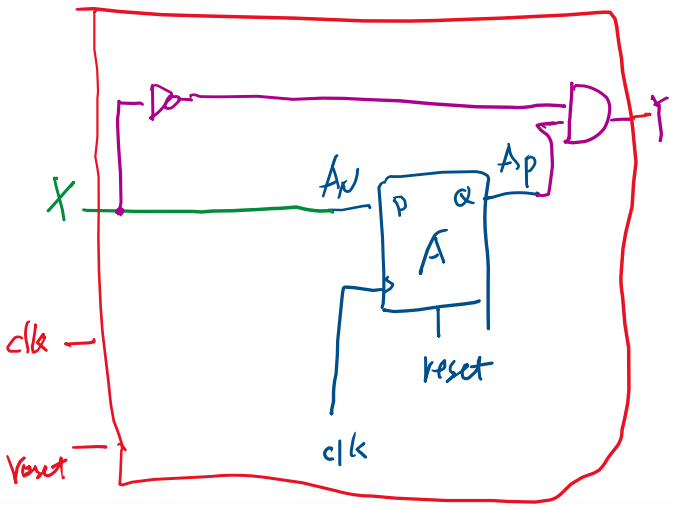
S1



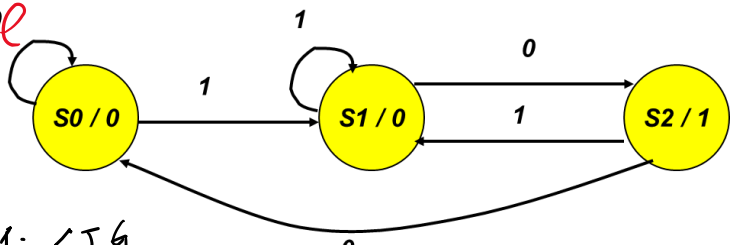
$A_N = Y$

$Y = A_P x'$

Less Hardware



Moore

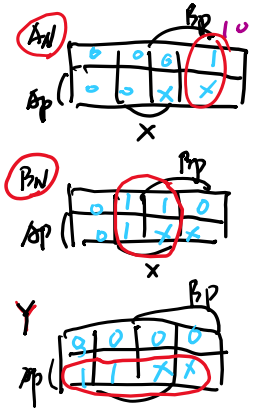


step1: STG

step2: state Table (truth table)

- state encoding, i.e. S0 = (00), S1 = (01), S2 = (10)

A_P	B_P	X	A_N	B_N	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	X	X	X
1	1	1	X	X	X



$B_N = B_P \cdot X'$

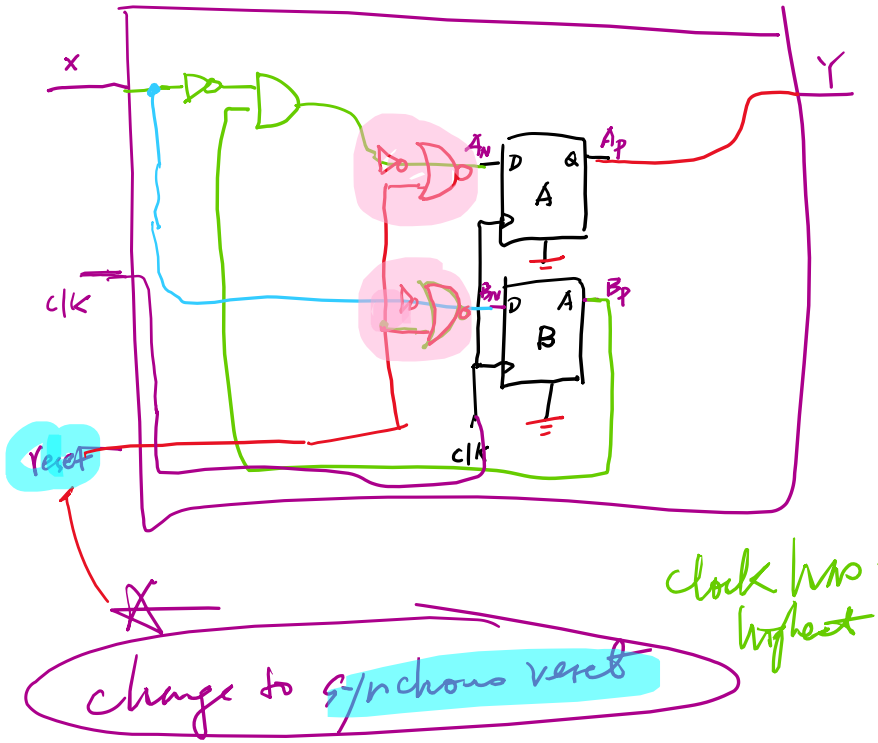
$B_N = X$

$Y = A_P$

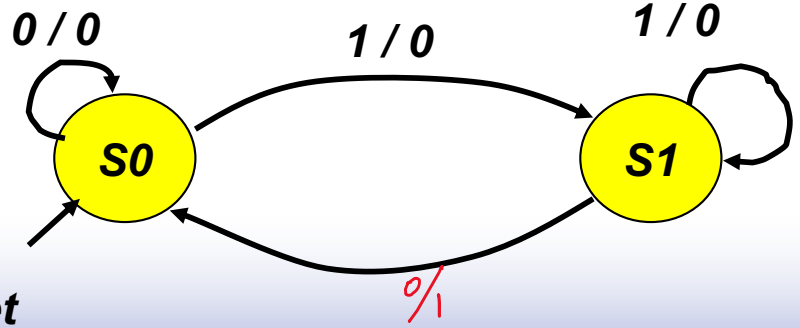
next

step3: V state eq. & output eq

step4: FSM logic Design



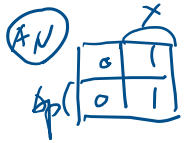
Mealy: state encoding S0 = (0), S1 = (1)



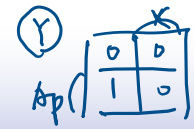
A_P	X	A_N	Y
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	0

S0

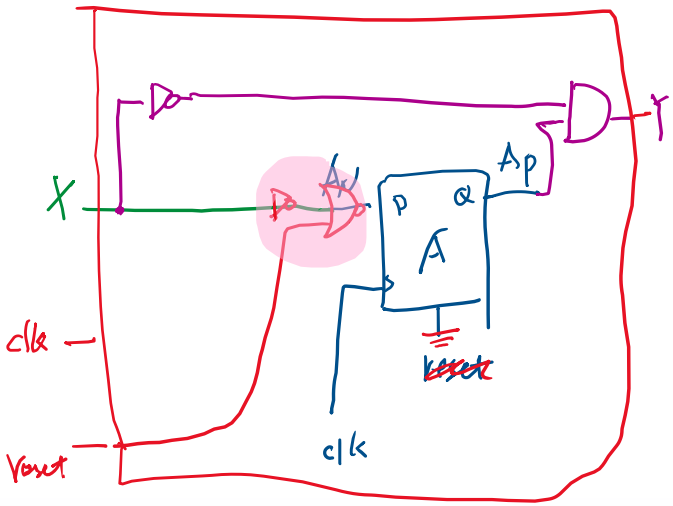
S1



$A_N = X$

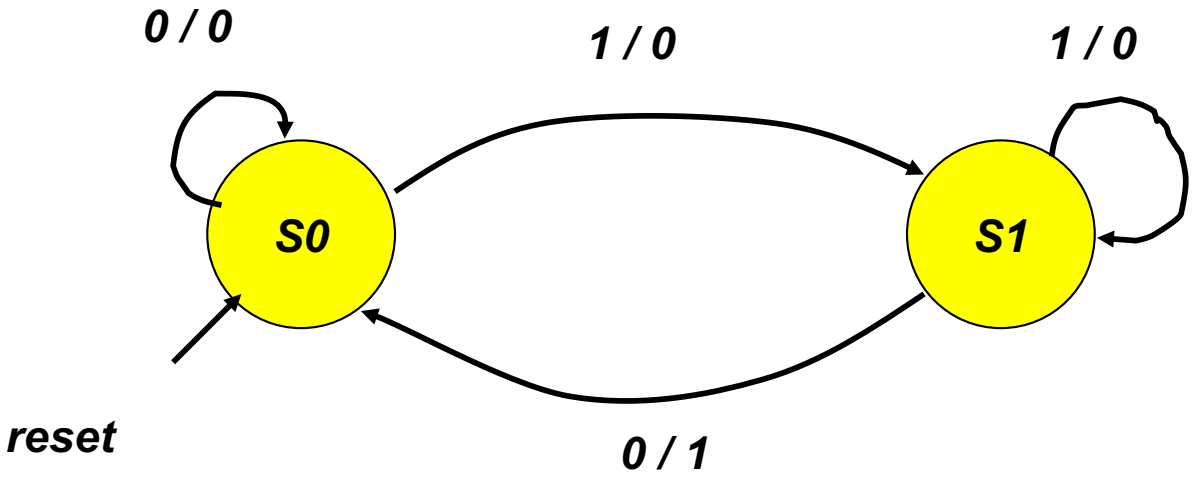


$Y = A_P X'$



Mealy FSM - Example 1

- Mealy FSM that Recognizes Sequence “10”

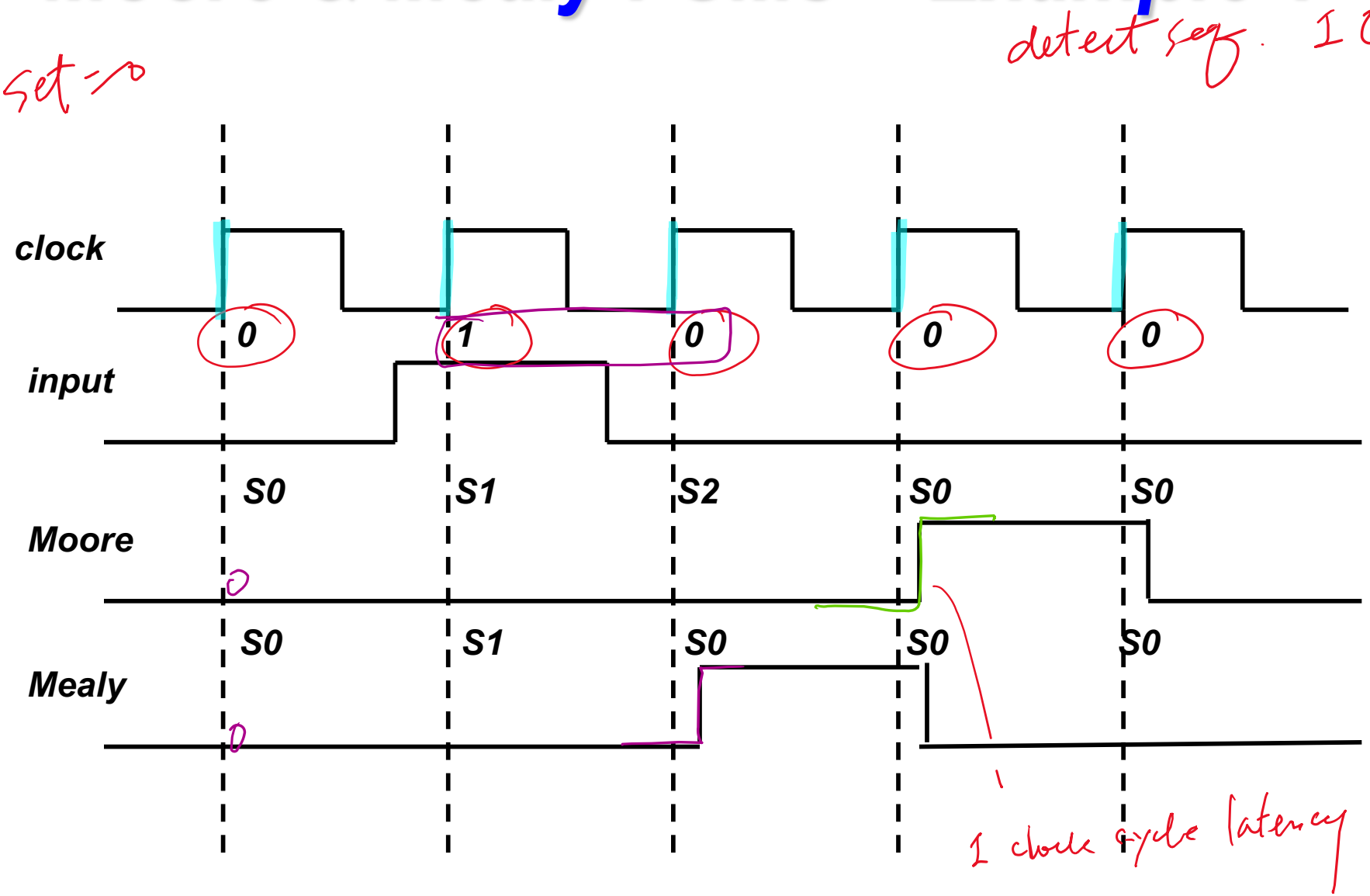


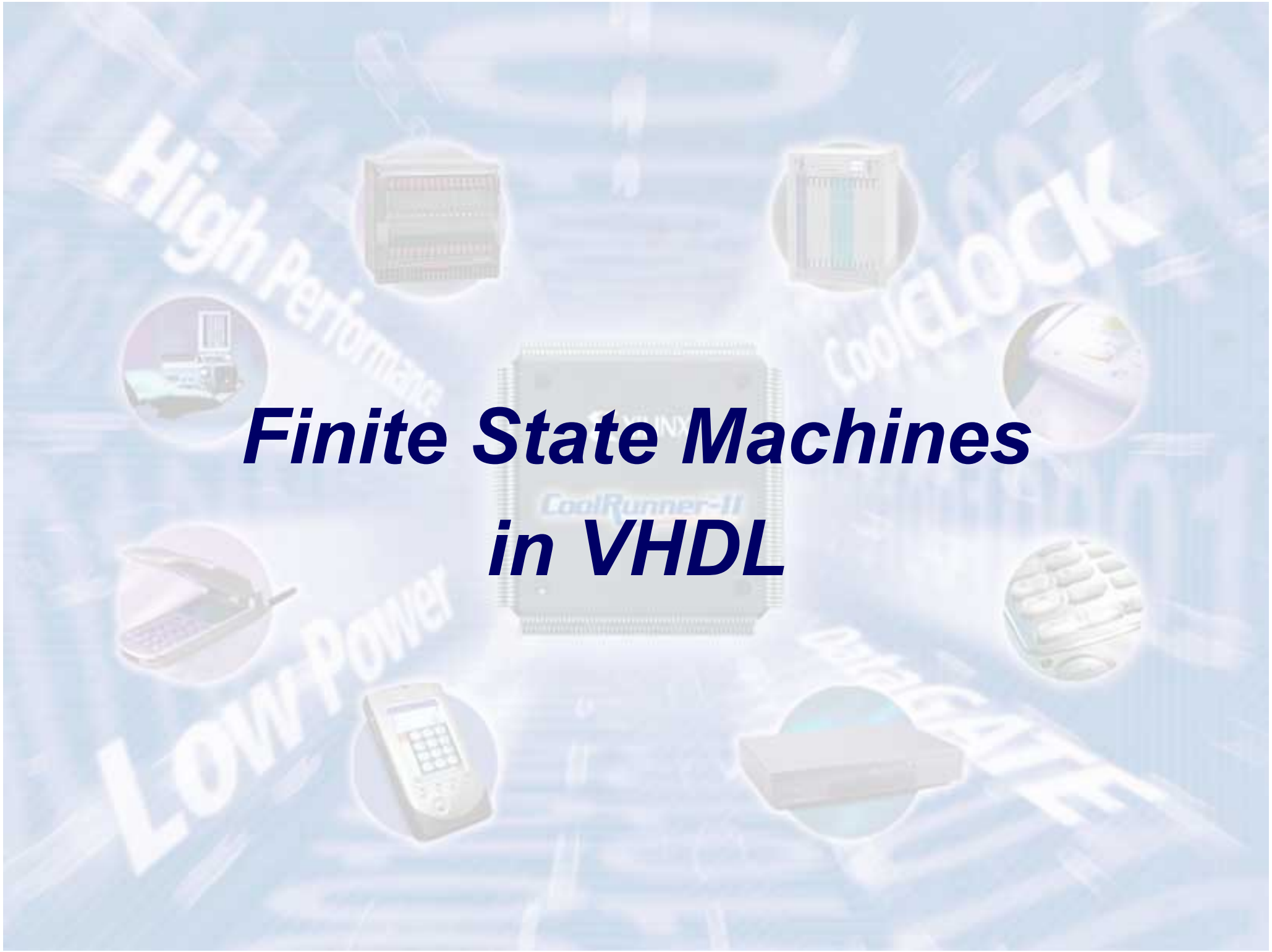
Meaning of states:

S0: No elements of the sequence observed

S1: “1” observed

Moore & Mealy FSMs – Example 1





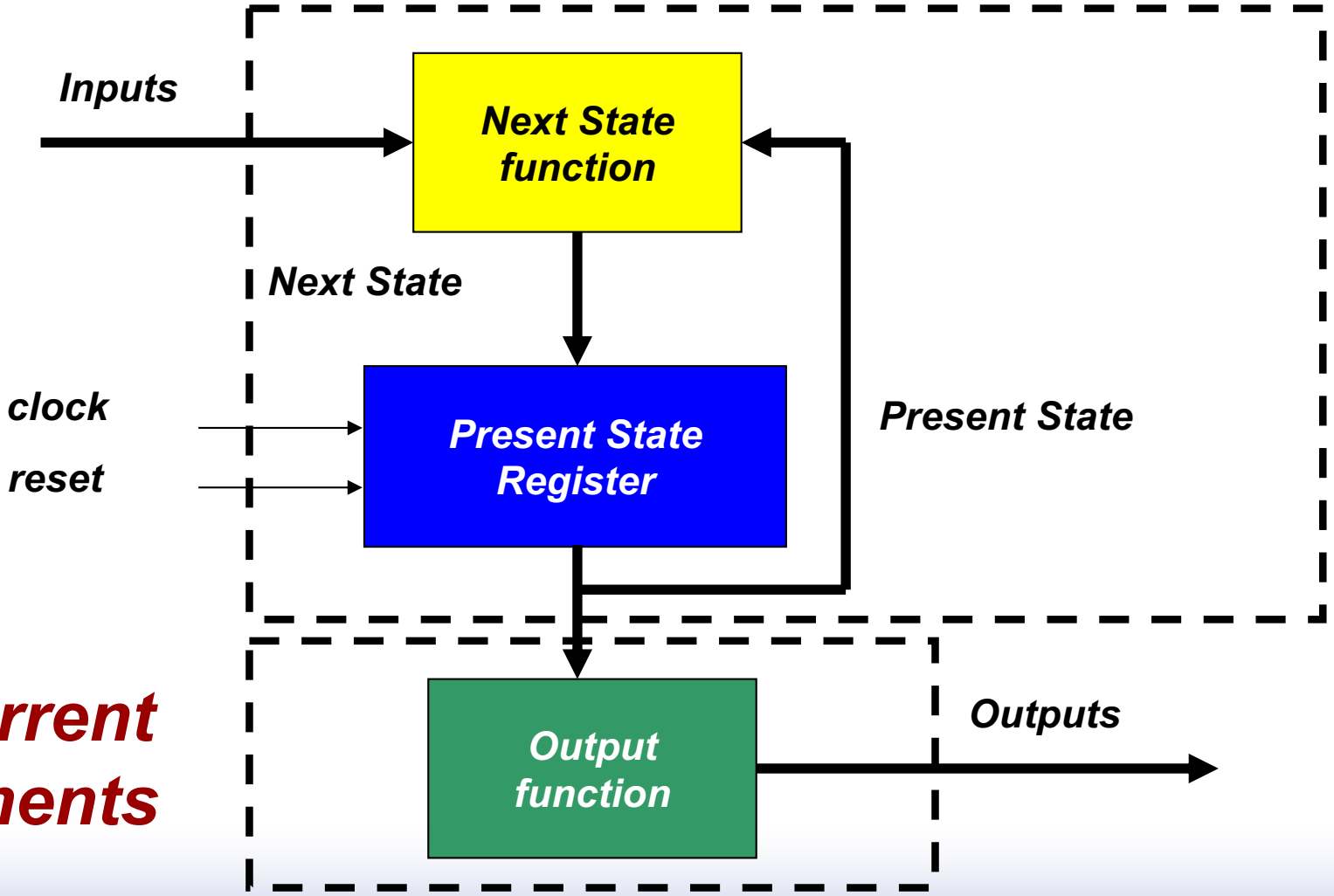
Finite State Machines in VHDL

FSMs in VHDL

- Finite State Machines Can Be Easily Described With Processes
- Synthesis Tools Understand FSM Description If Certain Rules Are Followed
  - State transitions should be described in a process sensitive to *clock* and *asynchronous reset* signals only
  - Outputs described as concurrent statements outside the process

Moore FSM

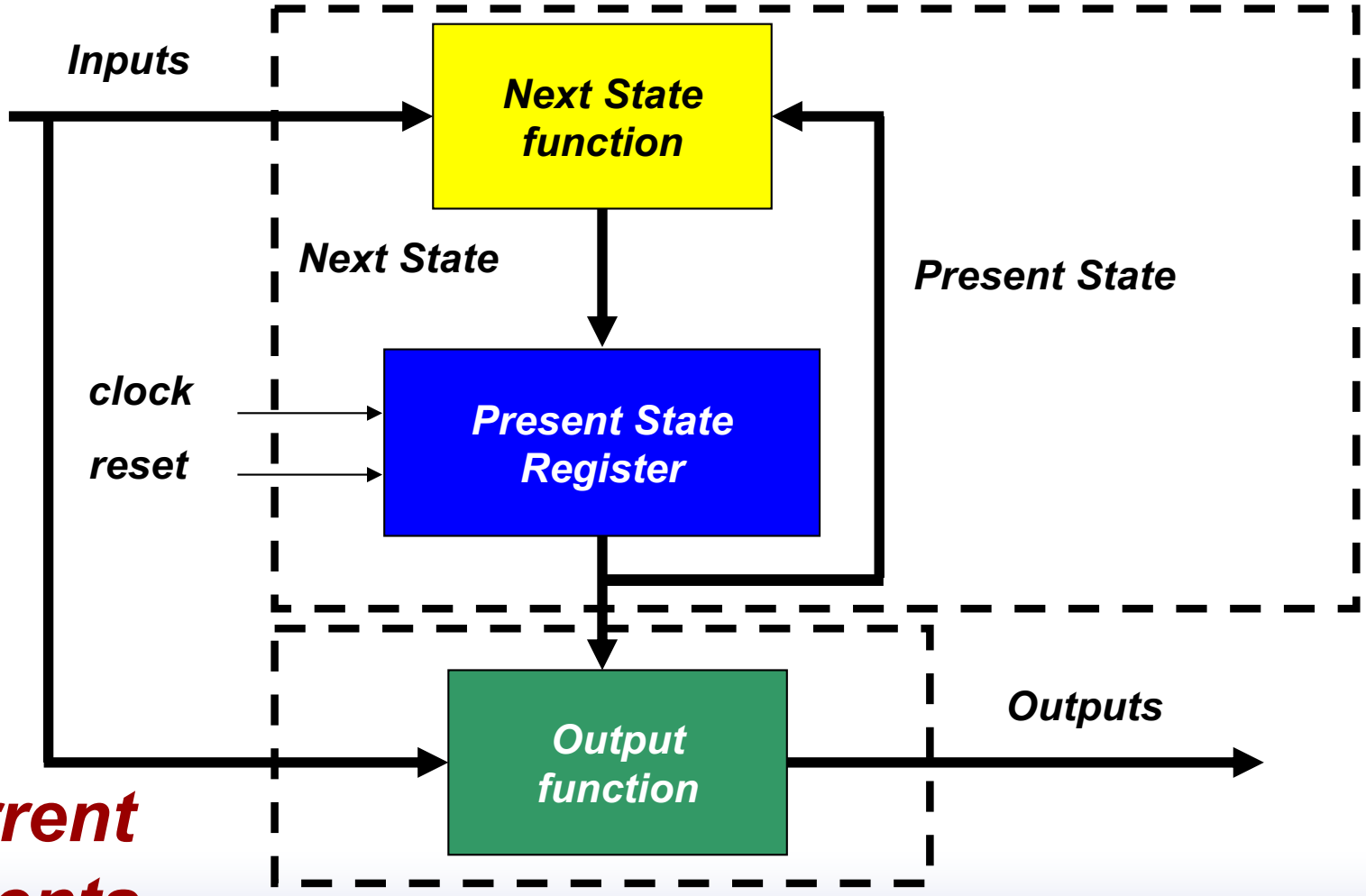
process(clock, reset)



concurrent statements

Mealy FSM

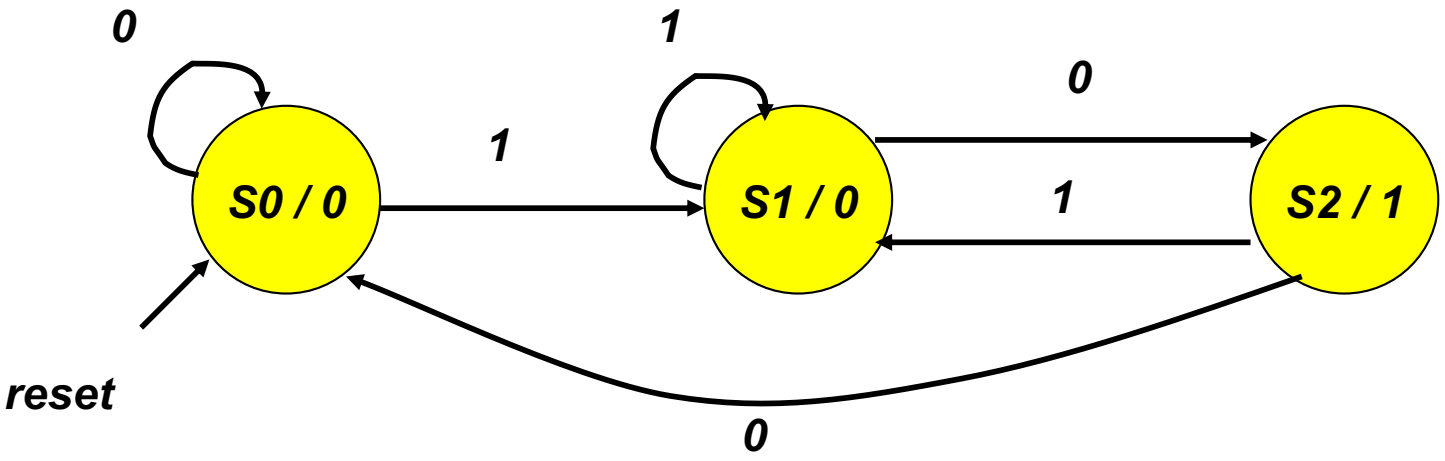
process(clock, reset)



concurrent statements

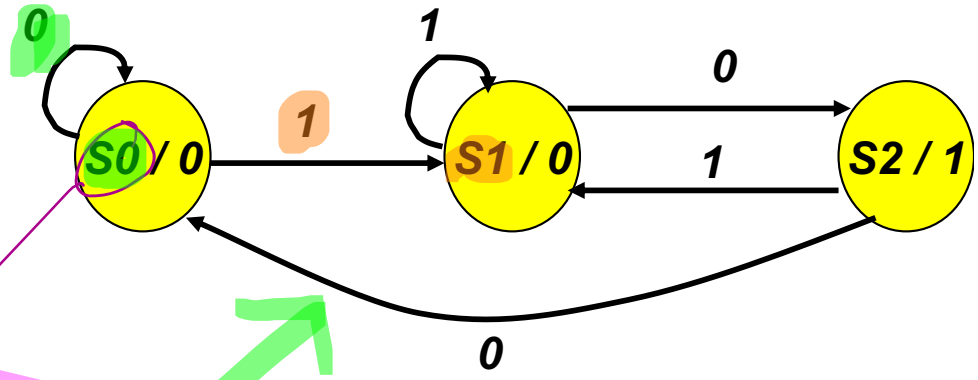
Moore FSM - Example 1

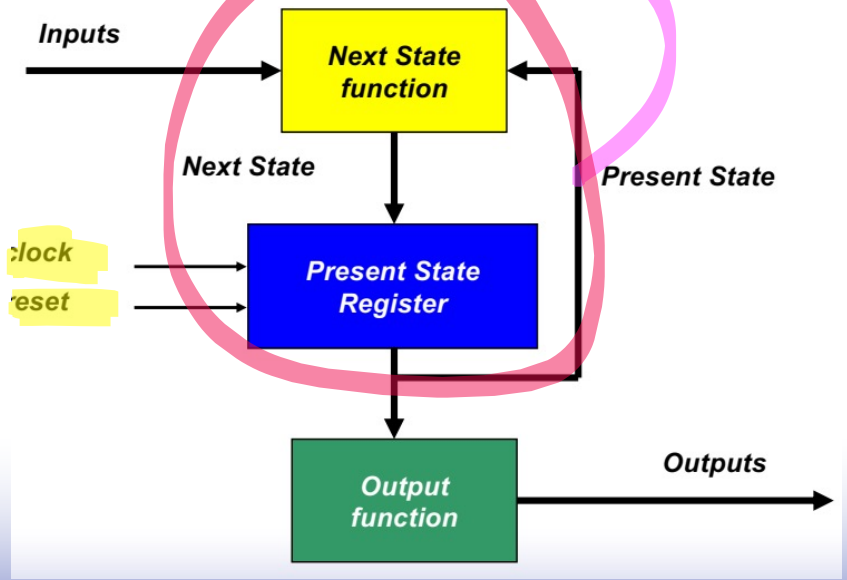
- Moore FSM that Recognizes Sequence “10”



Moore FSM in VHDL (1)

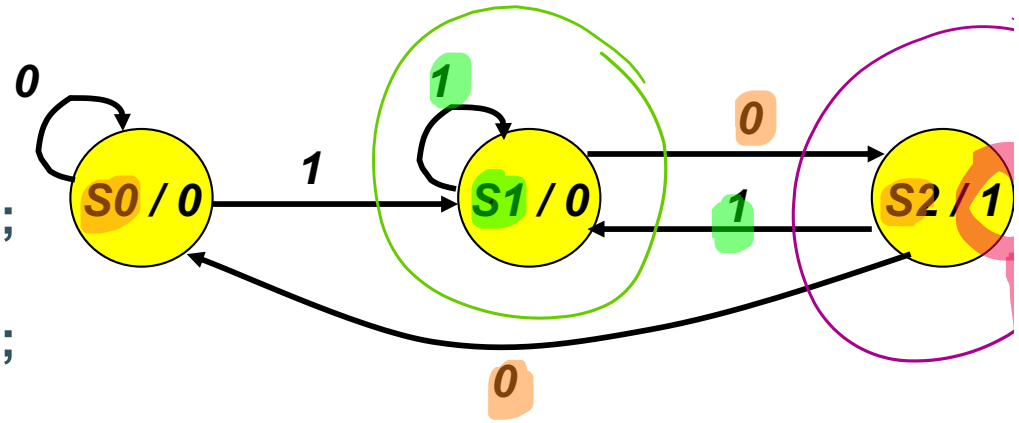
```
TYPE state IS (S0, S1, S2);
SIGNAL Moore_state: state;

U_Moore: PROCESS (clock, reset)
BEGIN
    IF(reset = '1') THEN
        Moore_state <= S0;
    ELSIF (clock = '1' AND clock'event) THEN
        CASE Moore_state IS
            WHEN S0 =>
                IF input = '1' THEN
                    Moore_state <= S1;
                ELSE
                    Moore_state <= S0;
                END IF;
```

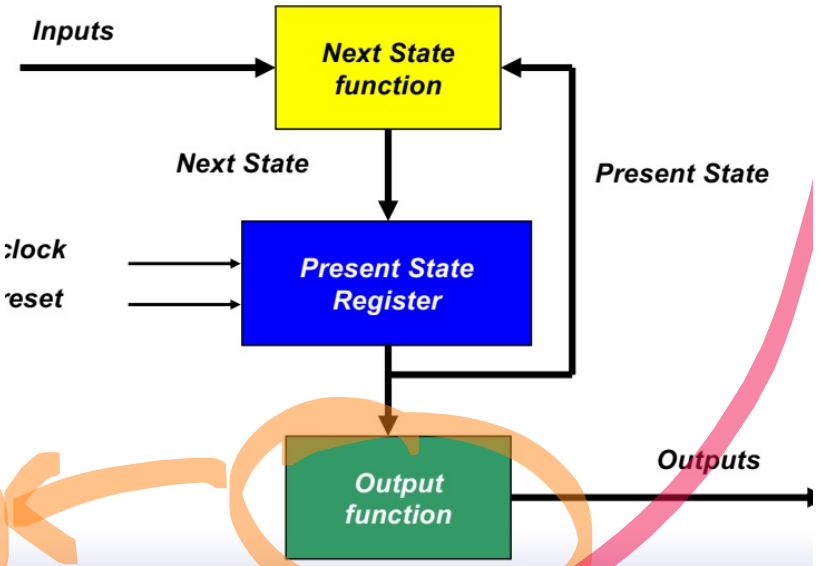




Moore FSM in VHDL (2)

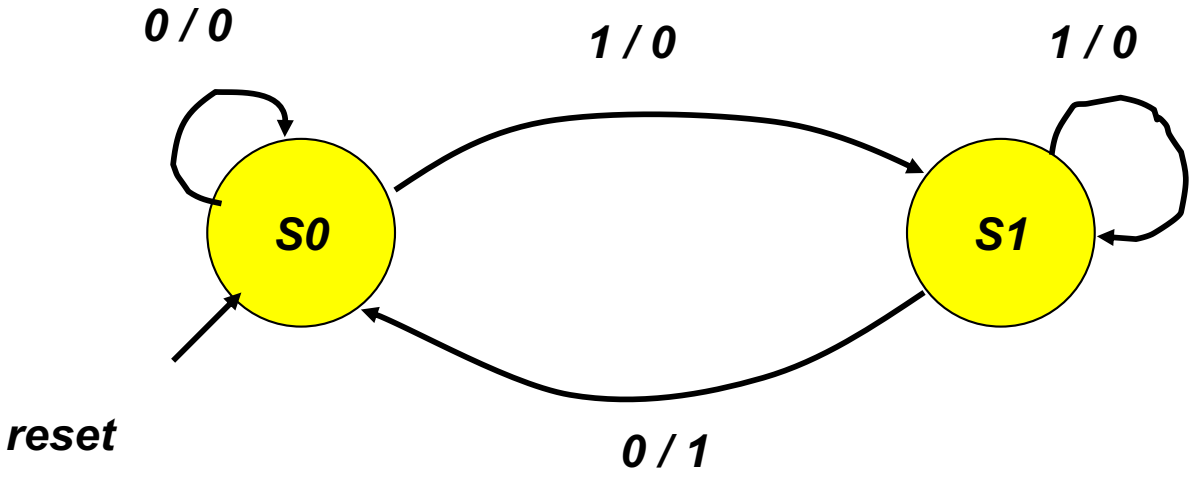
```
        WHEN S1 =>
            IF input = '0' THEN
                Moore_state <= S2;
            ELSE
                Moore_state <= S1;
            END IF;
        WHEN S2 =>
            IF input = '0' THEN
                Moore_state <= S0;
            ELSE
                Moore_state <= S1;
            END IF;
    END CASE;
END IF;
END PROCESS;

Output <= '1' WHEN Moore_state = S2 ELSE '0';
```



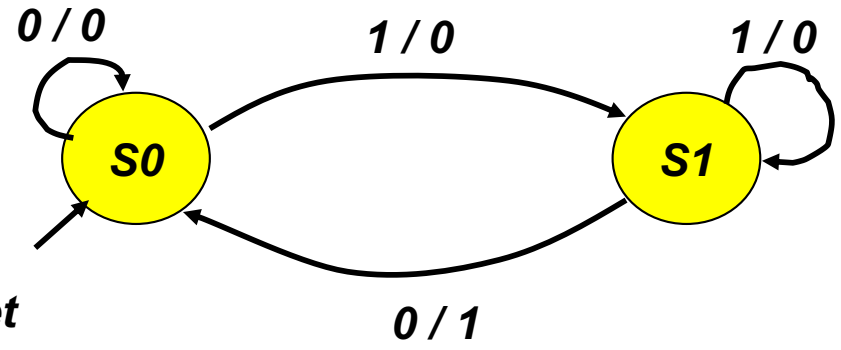


Mealy FSM - Example 1

- Mealy FSM that Recognizes Sequence "10"



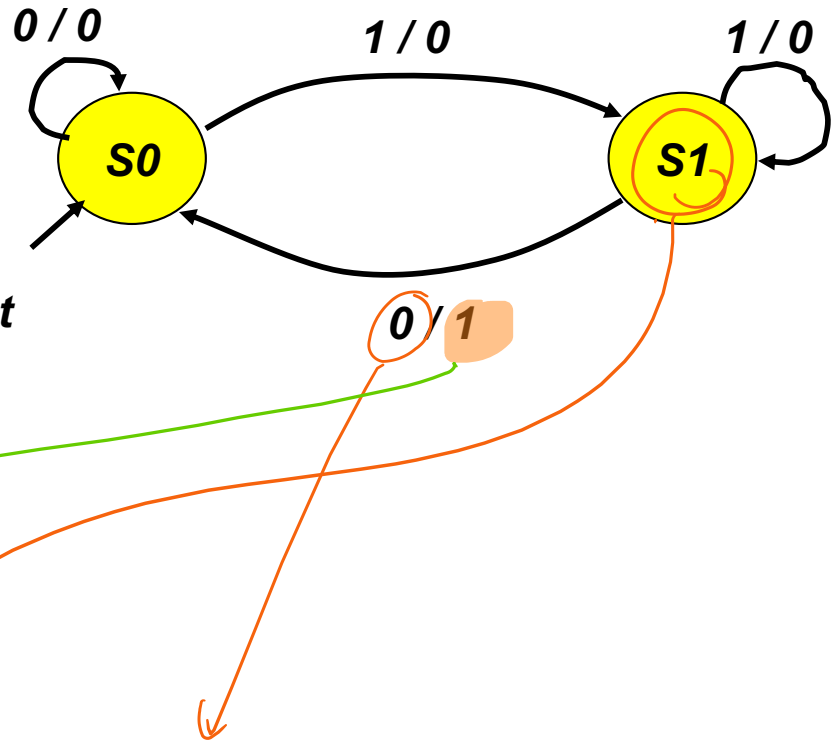
Mealy FSM in VHDL (1)



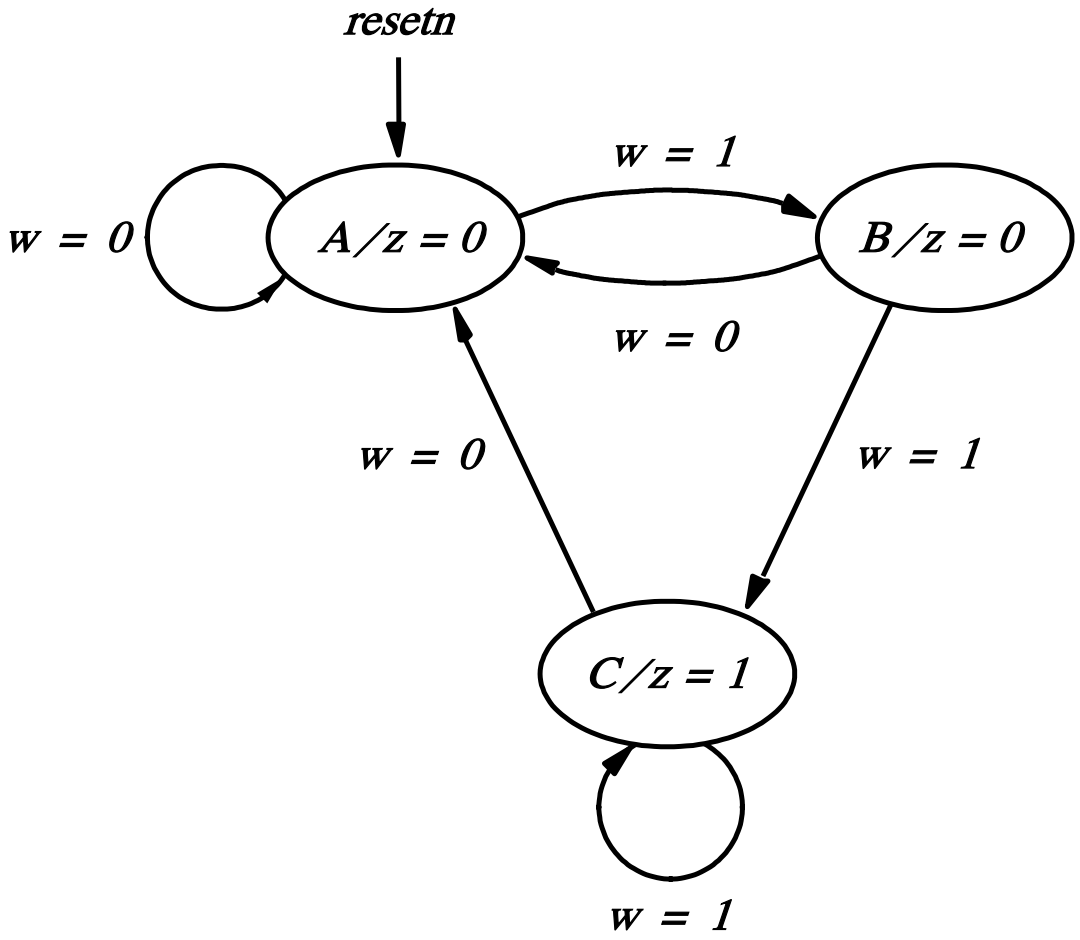

```
TYPE state IS (S0, S1);
SIGNAL Mealy_state: state;

U_Mealy: PROCESS(clock, reset)
BEGIN
      IF(reset = '1') THEN
         Mealy_state <= S0;
      ELSIF (clock = '1' AND clock'event) THEN
         CASE Mealy_state IS
               WHEN S0 =>
                  IF input = '1' THEN
                       Mealy_state <= S1;
                  ELSE
                       Mealy_state <= S0;
                  END IF;
```

Mealy FSM in VHDL (2)

```
            WHEN S1 =>
              IF input = '0' THEN
                  Mealy_state <= S0;
              ELSE                          reset
                  Mealy_state <= S1;
              END IF;
        END CASE;
    END IF;
END PROCESS;

Output <= '1' WHEN (Mealy_state = S1 AND input = '0') ELSE '0';
```



Moore FSM – Example 2: State diagram

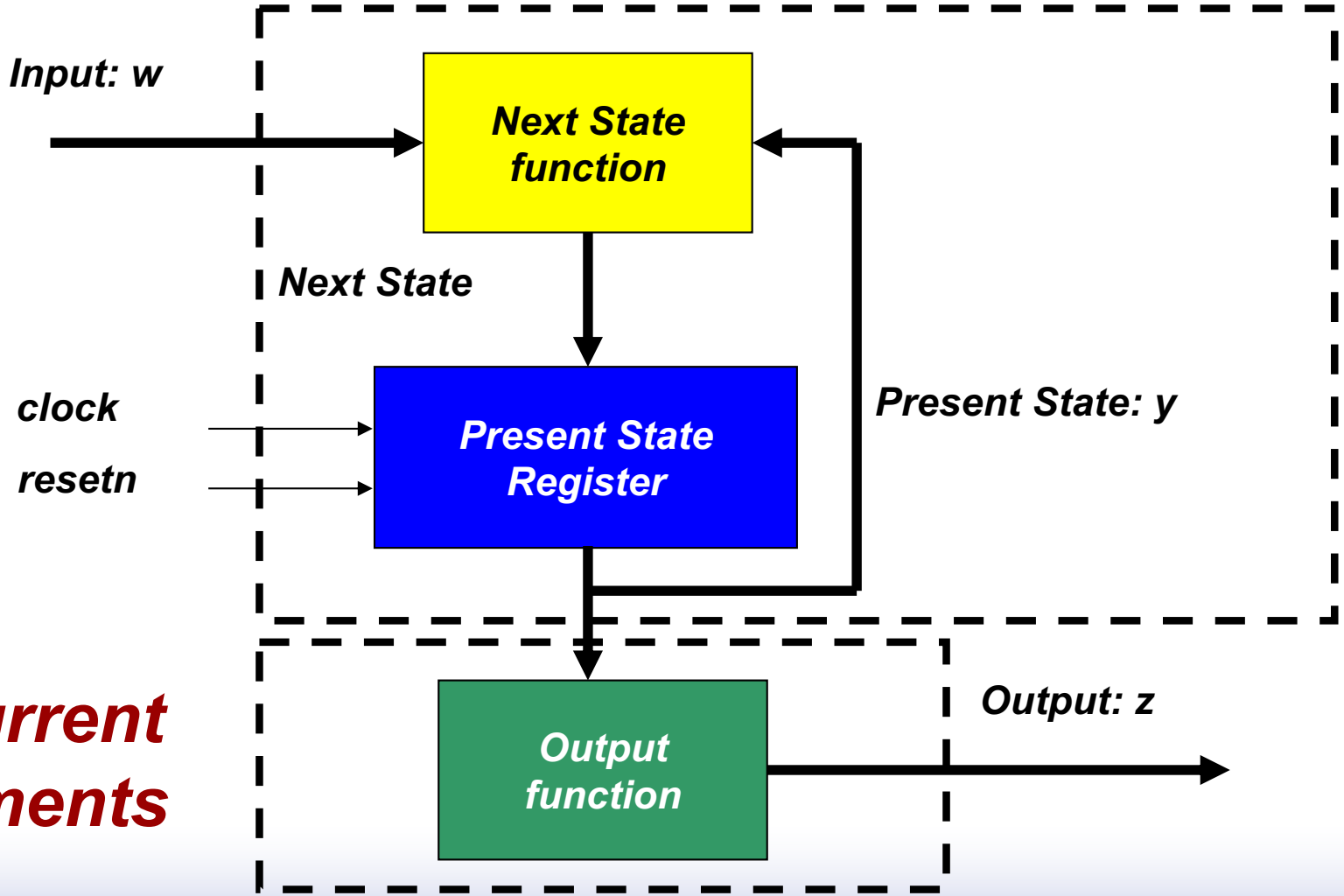


Moore FSM – Example 2: State table

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	C	1

Moore FSM

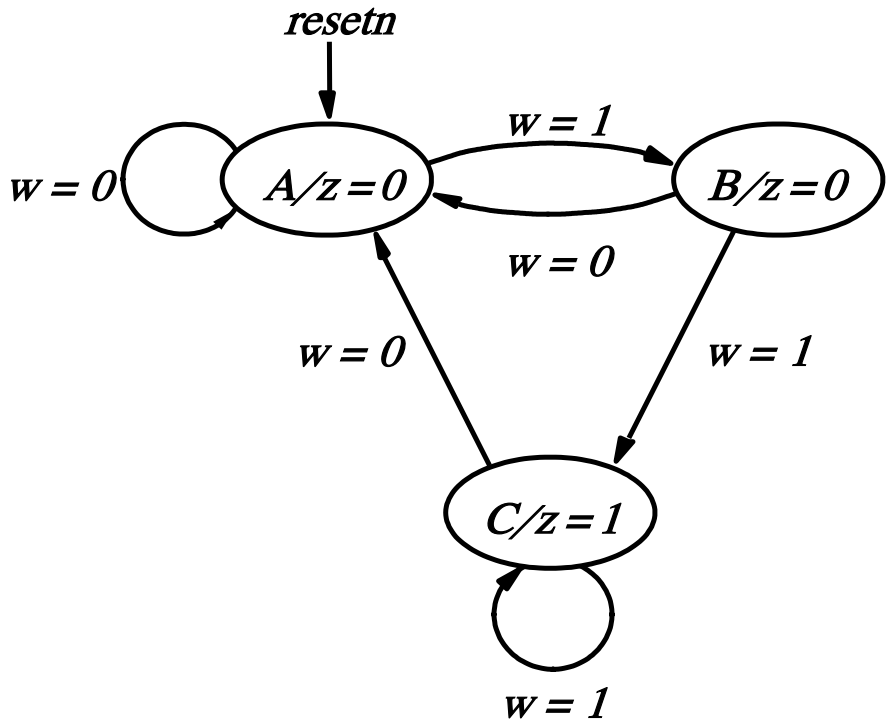
process(clock, reset)



concurrent statements

Moore FSM – Example 2: VHDL code (1)

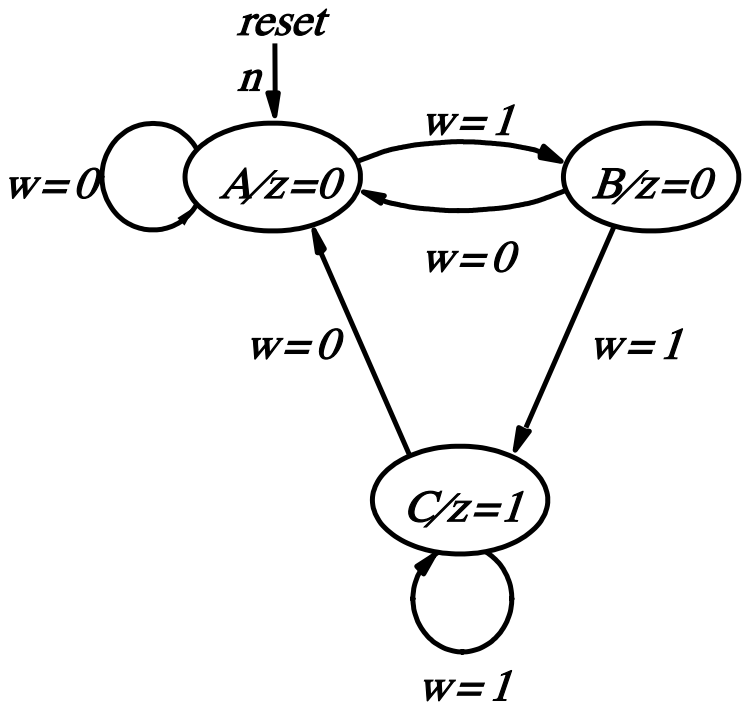
```
USE ieee.std_logic_1164.all ;

ENTITY simple IS
      PORT (  clock   : IN STD_LOGIC ;
              resetn  : IN STD_LOGIC ;
              w       : IN STD_LOGIC ;
              z       : OUT STD_LOGIC ) ;
END simple ;

ARCHITECTURE Behavior OF simple IS
      TYPE State_type IS (A, B, C) ;
      SIGNAL y : State_type ;
BEGIN
      PROCESS ( resetn, clock )
      BEGIN
            IF resetn = '0' THEN
                  y <= A ;
            ELSIF (Clock'EVENT AND Clock = '1') THEN
```



* sequence detector detects sequence of 2 or more consecutive 1's.

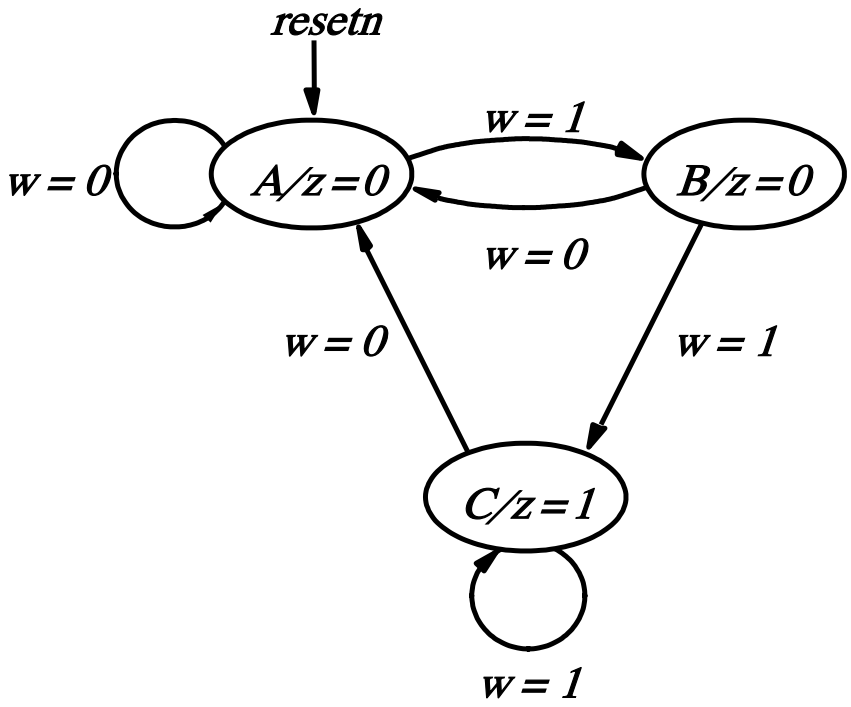
Moore FSM – Example 2: VHDL code (2)

```
CASE y IS
      WHEN A =>
            IF w = '0' THEN
                  y <= A ;
            ELSE
                  y <= B ;
            END IF ;
      WHEN B =>
            IF w = '0' THEN
                  y <= A ;
            ELSE
                  y <= C ;
            END IF ;
      WHEN C =>
            IF w = '0' THEN
                  y <= A ;
            ELSE
                  y <= C ;
            END IF ;
END CASE ;
```



Moore FSM – Example 2: VHDL code (3)

```
        END IF ;
    END PROCESS ;

    z <= '1' WHEN y = C ELSE '0' ;

END Behavior ;
```

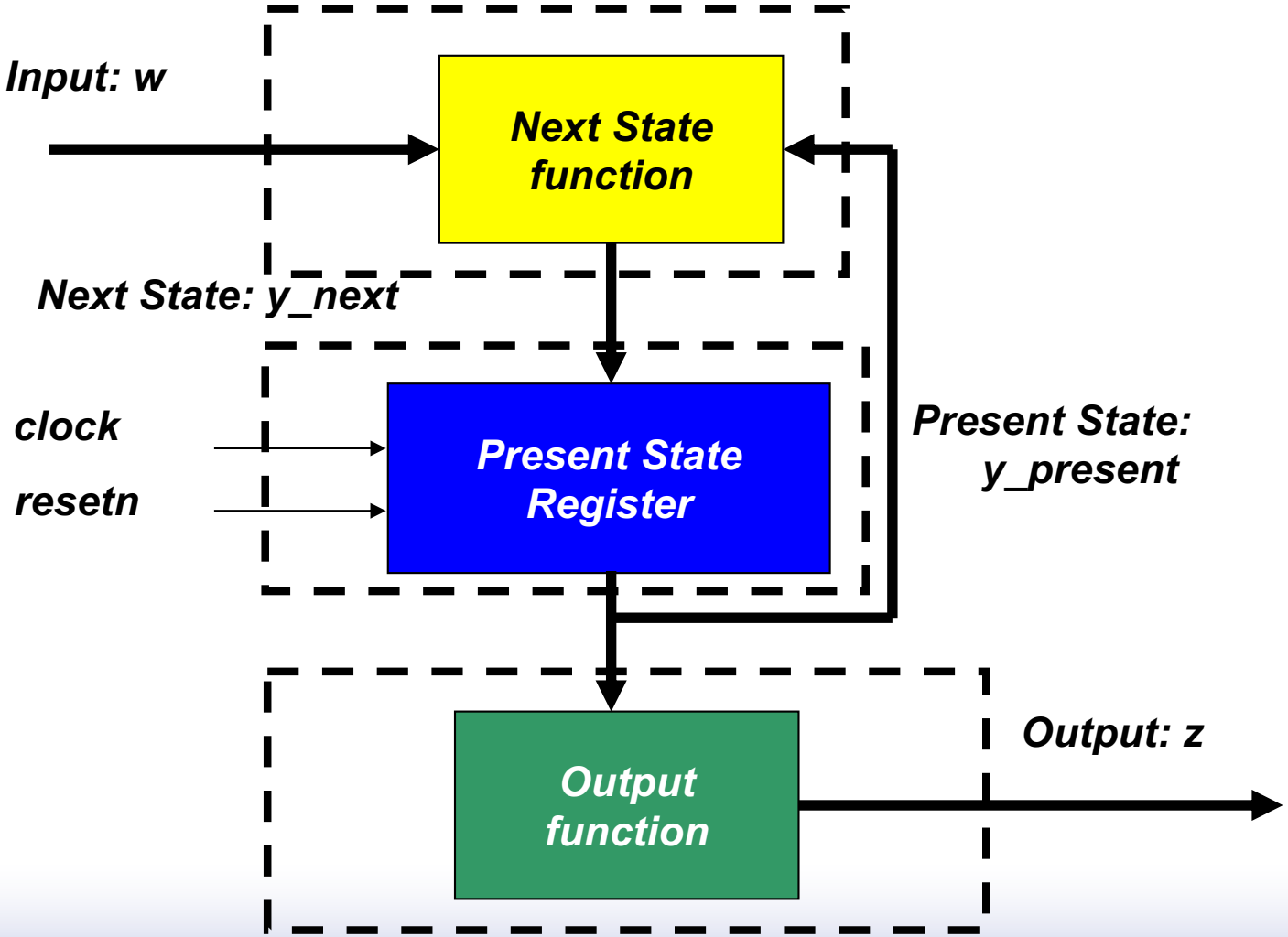


Moore FSM

process
(w,
y_present)

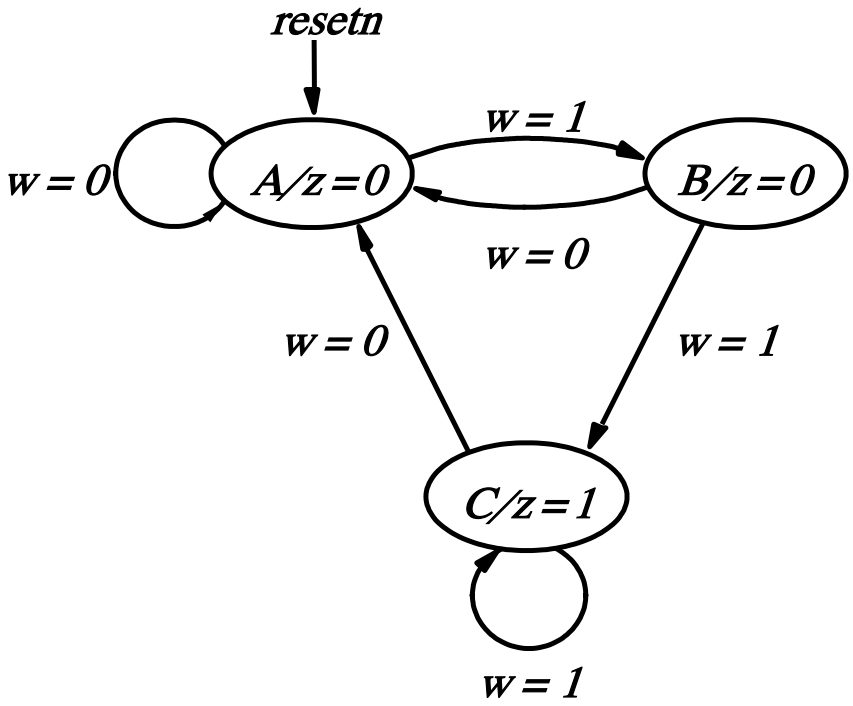
process
(clock,
resetn)

concurrent
statements



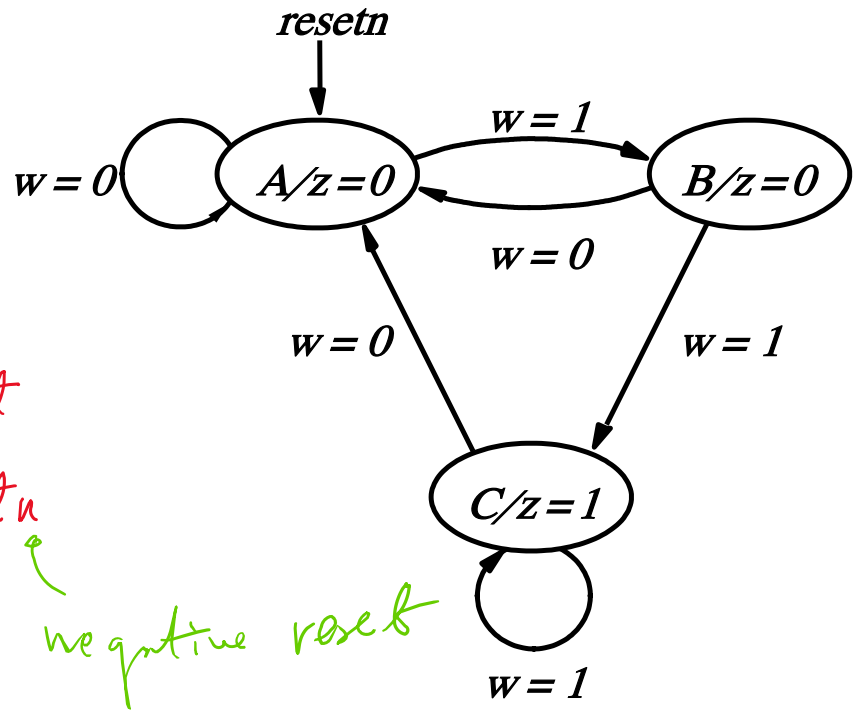
Alternative VHDL code (1)

```
ARCHITECTURE Behavior OF simple IS
    TYPE State_type IS (A, B, C) ;
    SIGNAL y_present, y_next : State_type ;
BEGIN
    PROCESS ( w, y_present )
    BEGIN
        CASE y_present IS
            WHEN A =>
                IF w = '0' THEN
                    y_next <= A ;
                ELSE
                    y_next <= B ;
                END IF ;
            WHEN B =>
                IF w = '0' THEN
                    y_next <= A ;
                ELSE
                    y_next <= C ;
                END IF ;
```



Alternative VHDL code (2)

```
            WHEN C =>
                IF w = '0' THEN
                    y_next <= A ;
                ELSE
                    y_next <= C ;
                END IF ;
        END CASE ;
    END PROCESS ;

    PROCESS (clock, resetn)
    BEGIN
        IF resetn = '0' THEN
            y_present <= A ;
        ELSIF (clock'EVENT AND clock = '1') THEN
            y_present <= y_next ;
        END IF ;
    END PROCESS ;

        z <= '1' WHEN y_present = C ELSE '0' ;
END Behavior ;
```

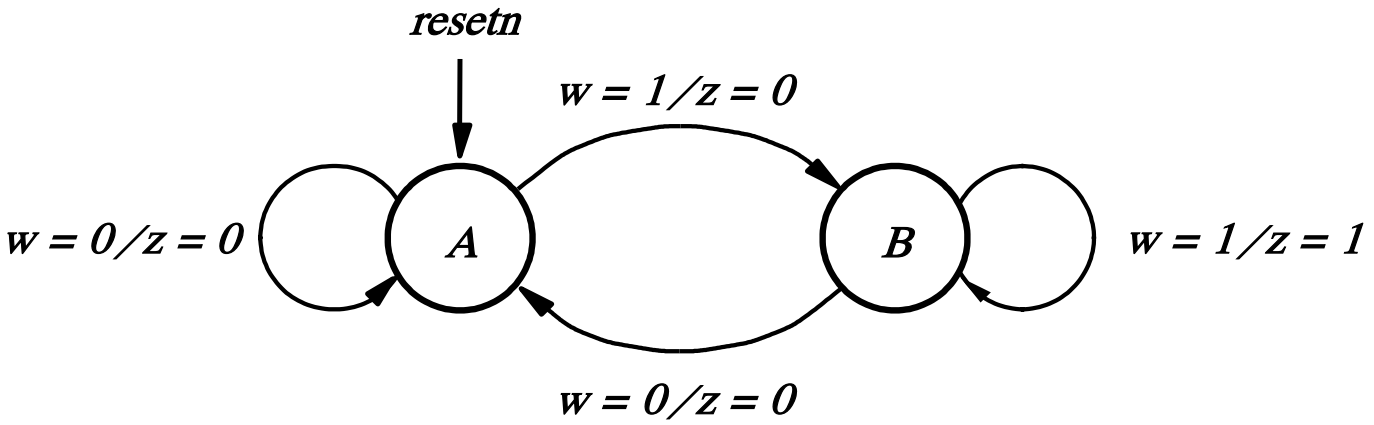


reset

resetn

negative reset

Mealy FSM – Example 2: State diagram

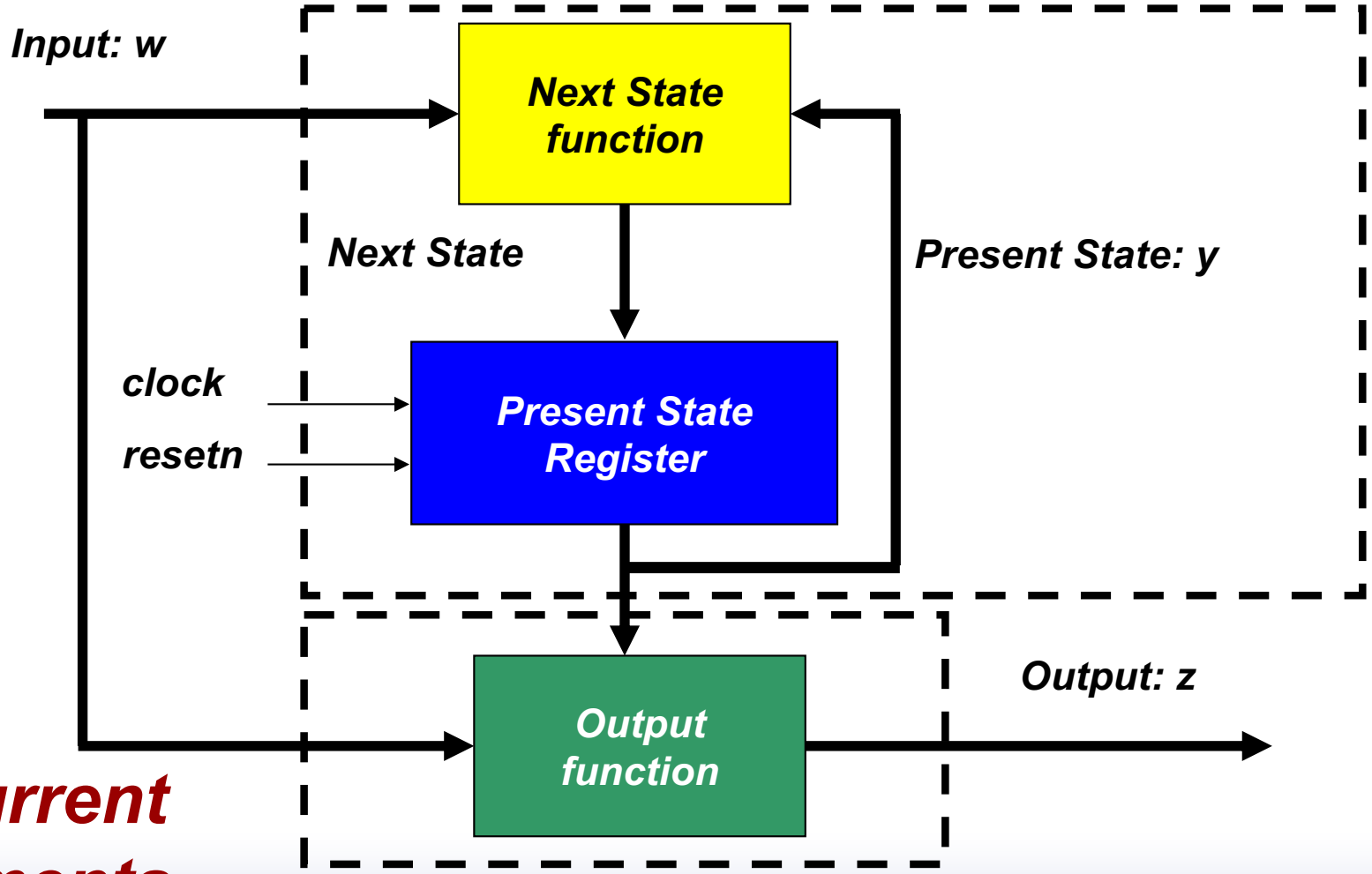


Mealy FSM – Example 2: State table

Present state	Next state		Outputz	
	w = 0	w = 1	w = 0	w = 1
A	A	B	0	0
B	A	B	0	1

Mealy FSM

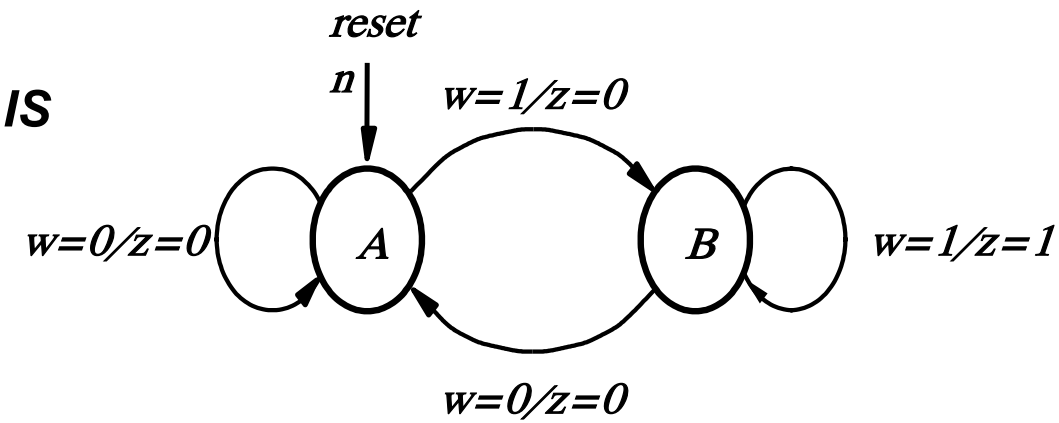
process(clock, reset)



concurrent statements

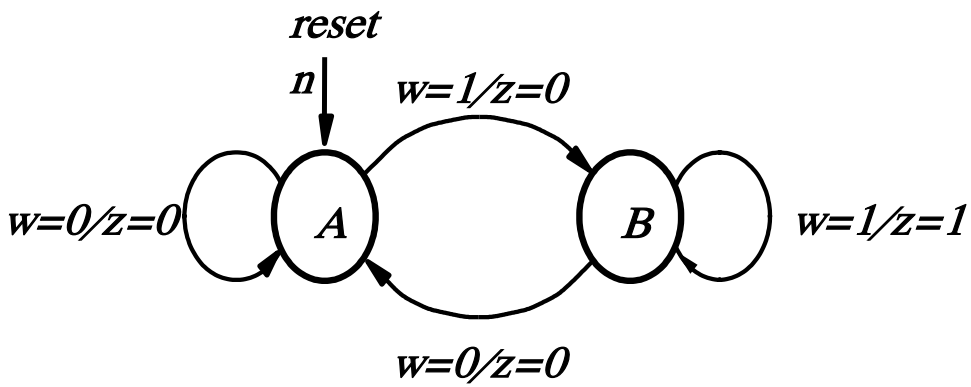
Mealy FSM – Example 2: VHDL code (1)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY Mealy IS
      PORT ( clock    : IN     STD_LOGIC ;
             resetn   : IN     STD_LOGIC ;
             w        : IN     STD_LOGIC ;
             z        : OUT    STD_LOGIC ) ;
END Mealy ;

ARCHITECTURE Behavior OF Mealy IS
      TYPE State_type IS (A, B) ;
      SIGNAL y : State_type ;
BEGIN
      PROCESS ( resetn, clock )
      BEGIN
          IF resetn = '0' THEN
              y <= A ;
          ELSIF (clock'EVENT AND clock = '1') THEN
```



Mealy FSM – Example 2: VHDL code (2)

```
CASE y IS
    WHEN A =>
        IF w = '0' THEN
            y <= A ;

        ELSE
            y <= B ;

        END IF ;
    WHEN B =>
        IF w = '0' THEN
            y <= A ;

        ELSE
            y <= B ;

        END IF ;
END CASE ;
```



Mealy FSM – Example 2: VHDL code (3)

```
        END IF ;
END PROCESS ;

WITH y SELECT
        z <= w WHEN B,
        z <= '0'  WHEN others;

END Behavior ;
```