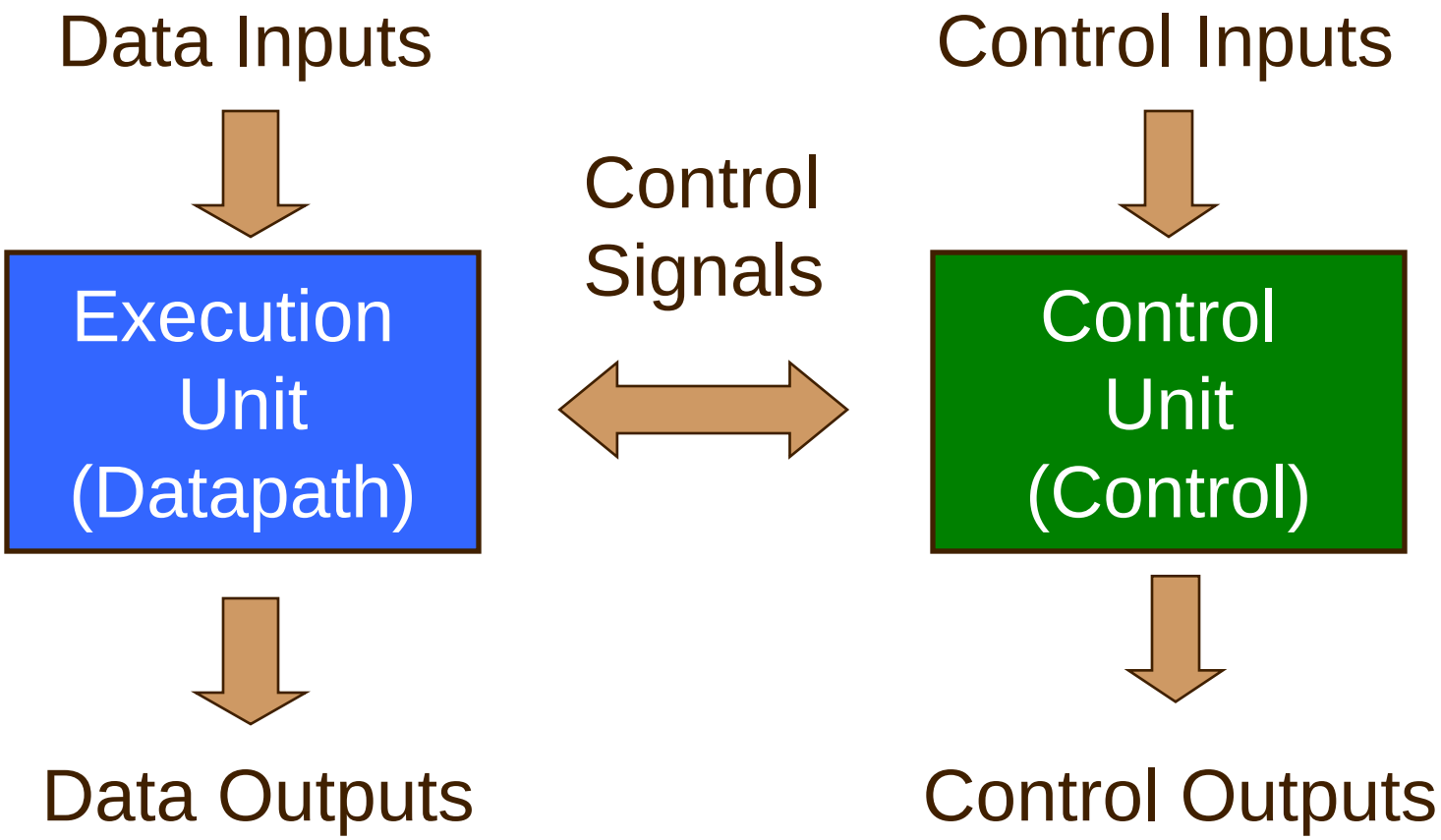
Μηχανές Πεπερασμένων Καταστάσεων σε VHDL

Δομή ενός Τυπικού Ψηφιακού Συστήματος



Finite State Machines (FSMs)

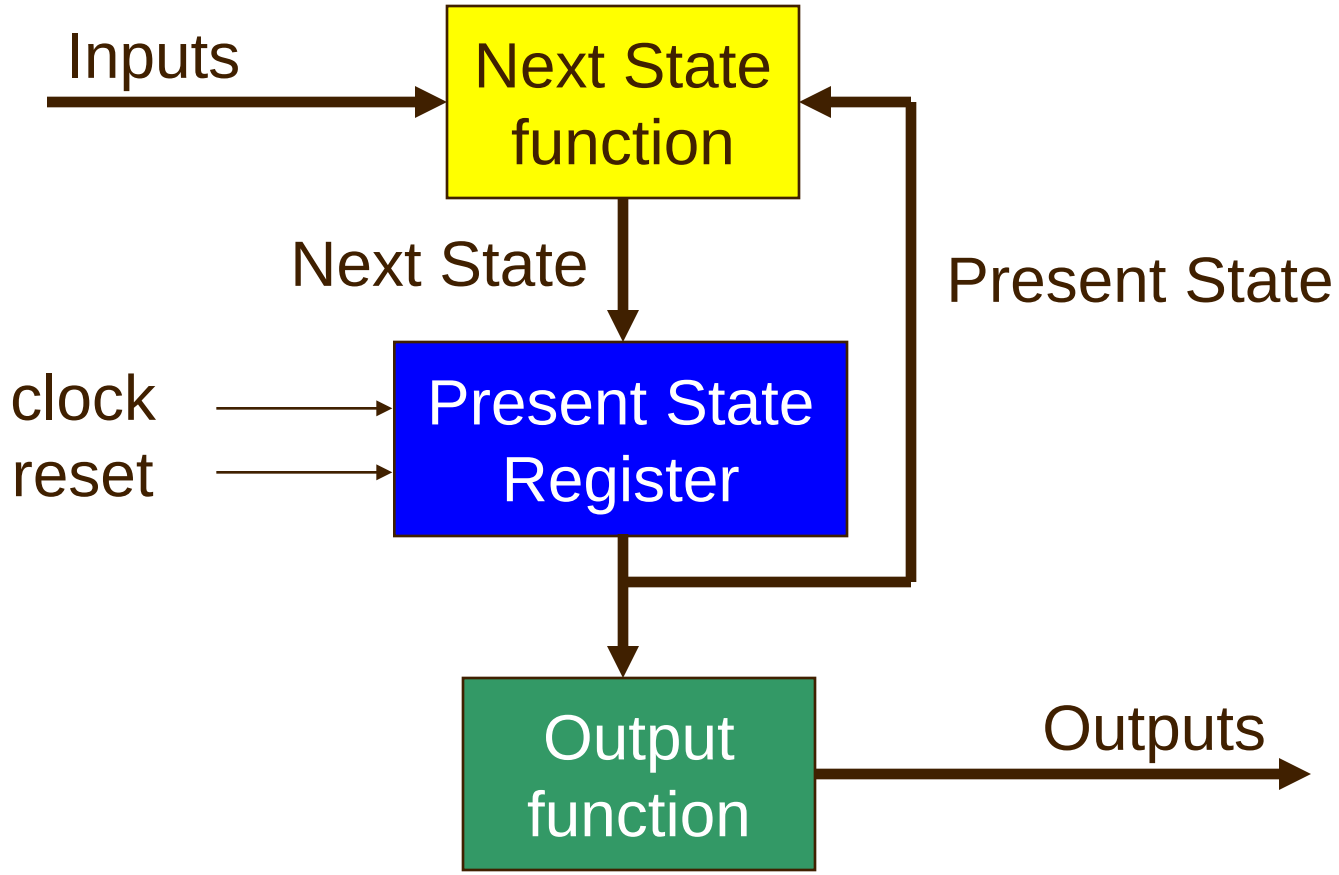
- Οποιοδήποτε κύκλωμα με μνήμη μπορεί να αναπαρασταθεί ως Finite State Machine

- Η Σχεδίαση FSMs περιλαμβάνει
  - Ορισμός καταστάσεων
  - Ορισμός μεταβάσεων μεταξύ καταστάσεων
  - Βελτιστοποίηση / ελαχιστοποίηση

- Above Approach Is Practical for Small FSMs Only

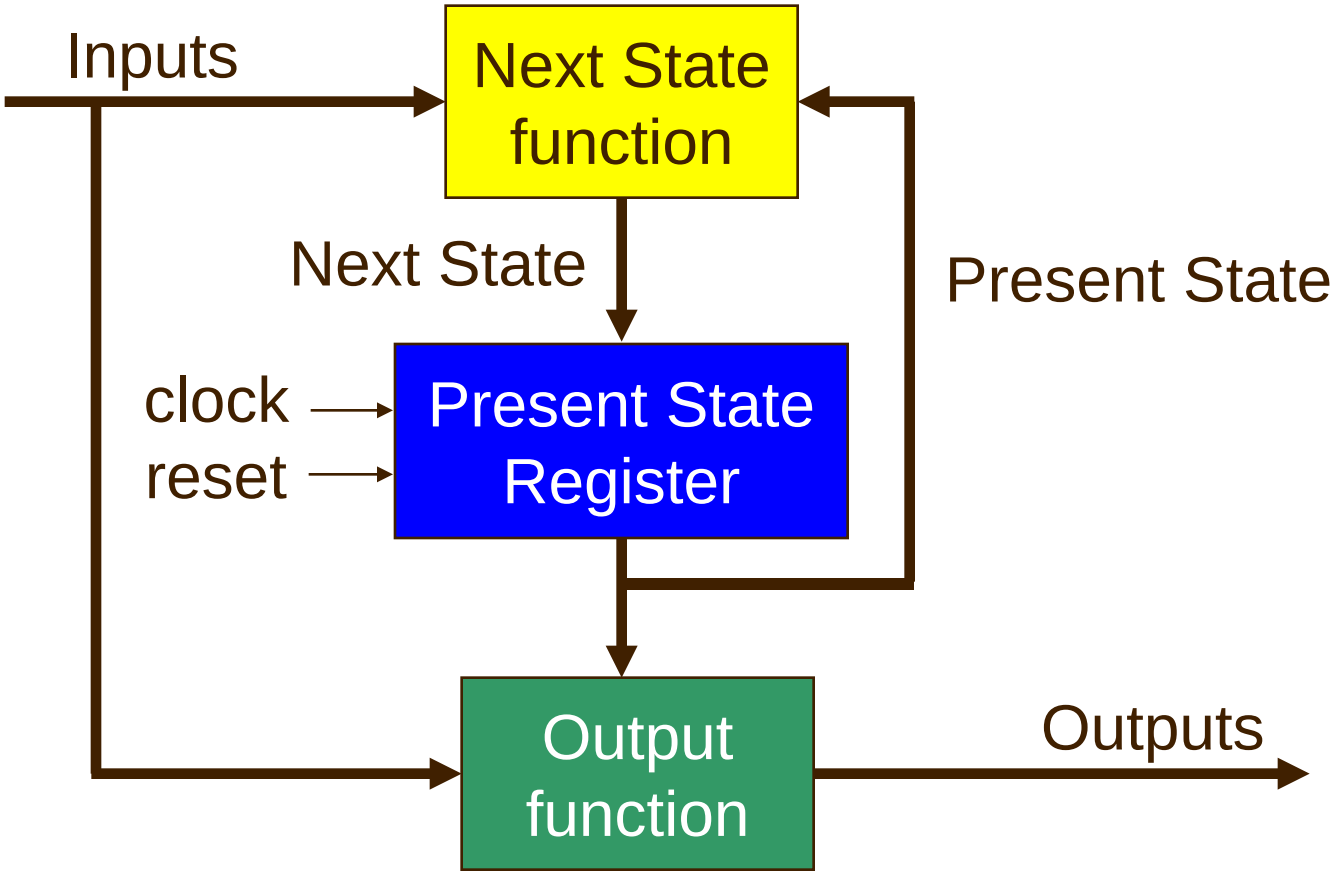
Moore FSM

- Η έξοδος είναι συνάρτηση μόνο της παρούσας κατάστασης

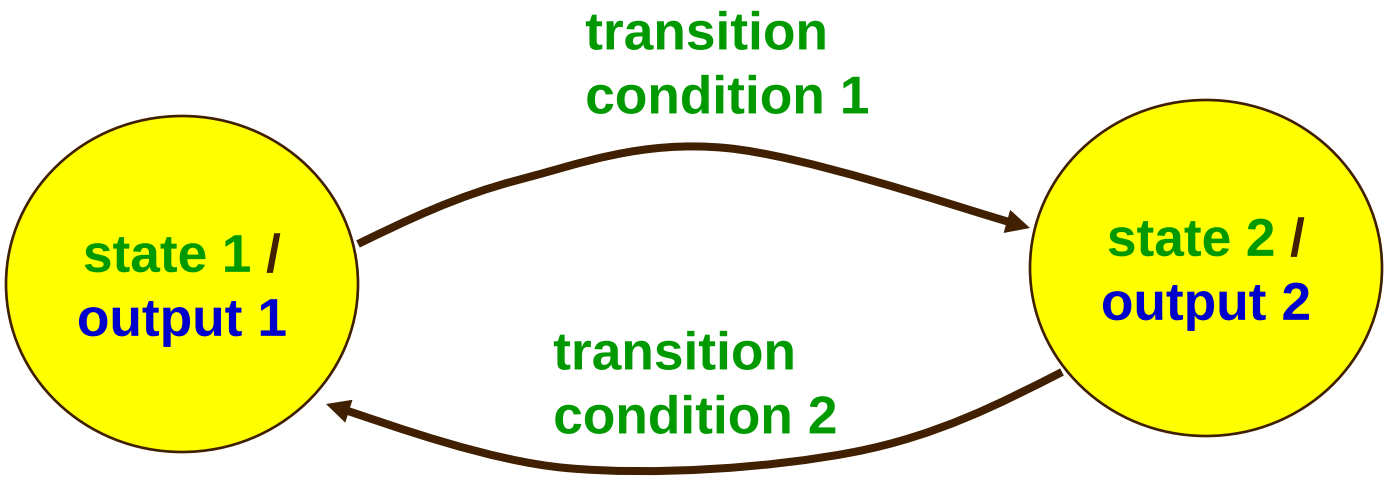


Mealy FSM

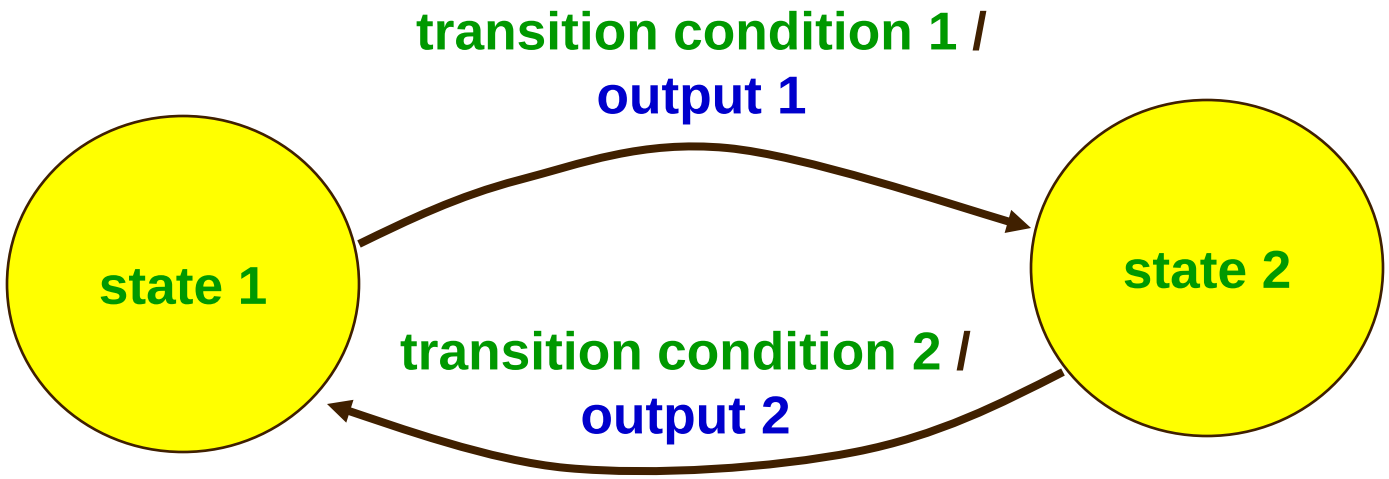
- Η έξοδος είναι συνάρτηση της παρούσας κατάστασης και των εισόδων



Moore Machine



Mealy Machine



Moore vs. Mealy FSM (1)

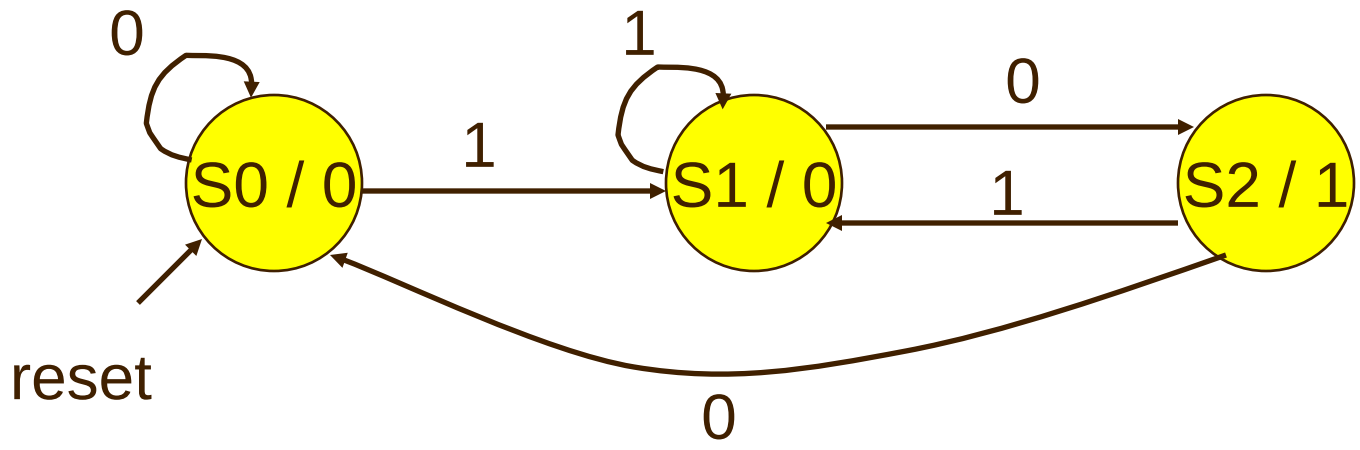
- Moore and Mealy FSMs μπορεί να είναι λειτουργικά ισοδύναμες
  - Μία ισοδύναμη Mealy FSM μπορεί να παραχθεί από μία Moore FSM και αντίστροφα
- Η τύπου Mealy FSM συνήθως απαιτεί λιγότερο αριθμό καταστάσεων
  - Το κύκλωμα απαιτεί λιγότερο χώρο

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

Moore FSM - Example 1

- Σχεδιάστε μία Moore FSM να αναγνωρίζει την ακολουθία “10”



Ερμηνεία των states:

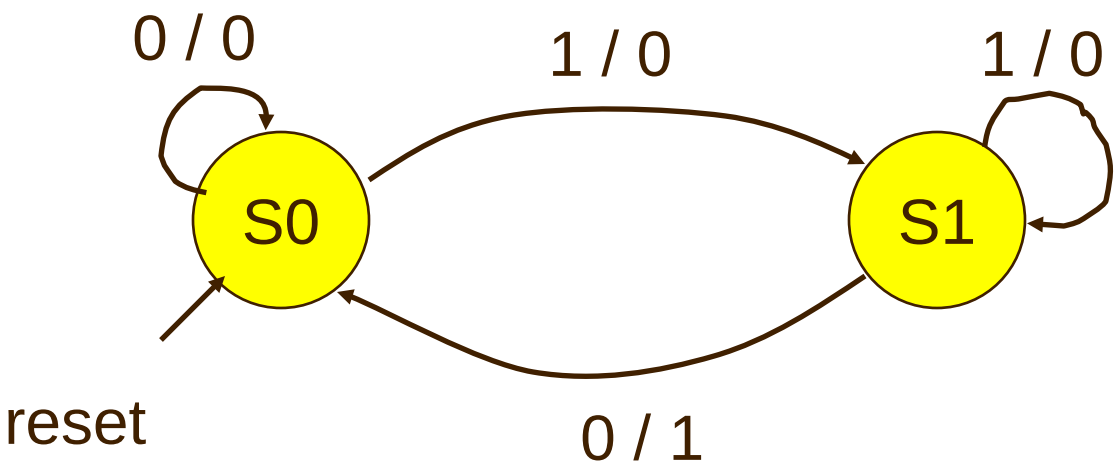
S0: κανένα ψηφίο της ακολουθίας δεν αναγνωρίστηκε ακόμα

S1: “1” Αναγνώριση του «1»

S2: “10” Αναγνώριση !!

Mealy FSM - Example 1

- Mealy FSM that Recognizes Sequence “10”

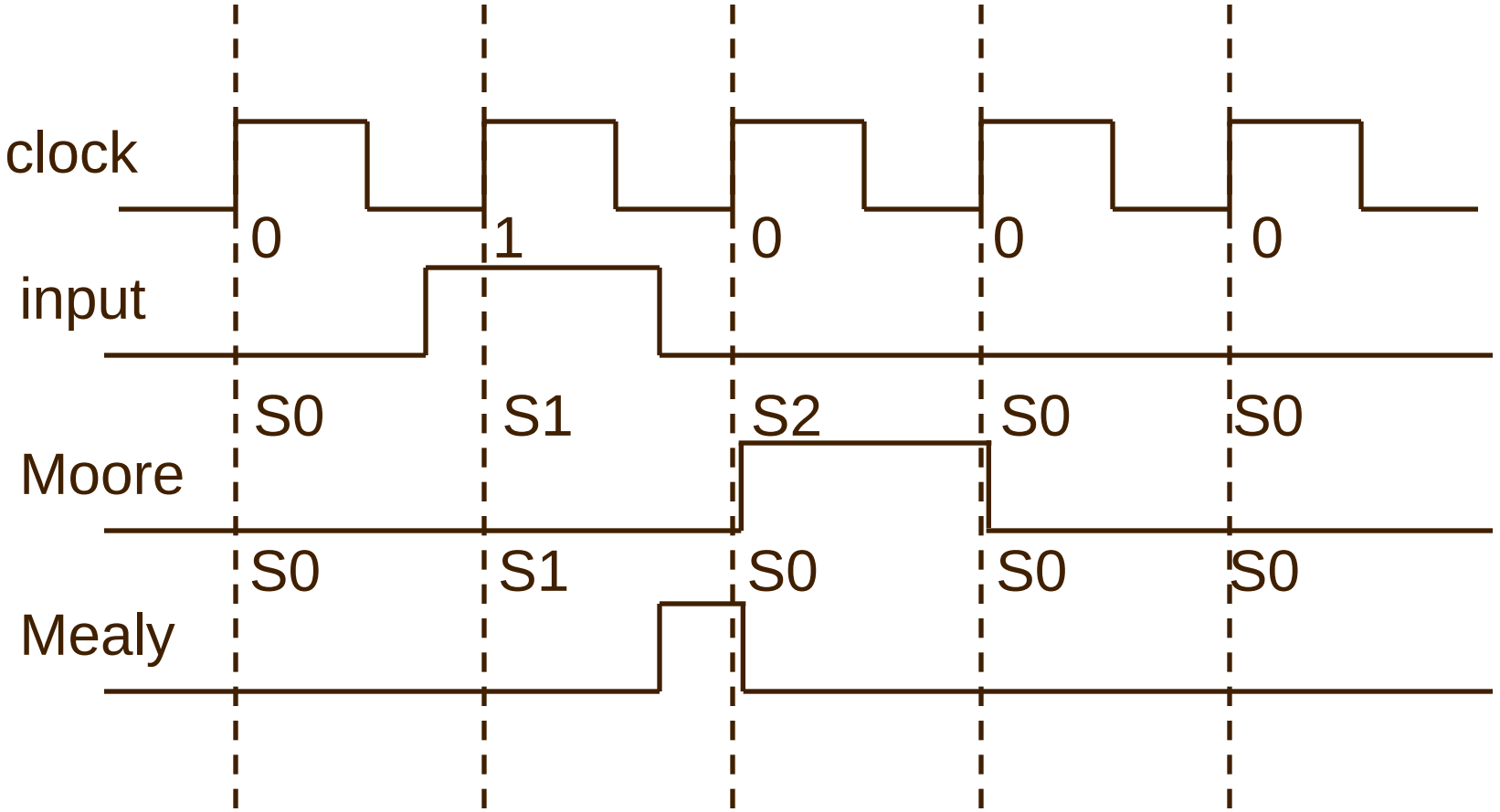


Meaning of states:

S0: No elements of the sequence observed

S1: “1” observed

Moore & Mealy FSMs – Example 1



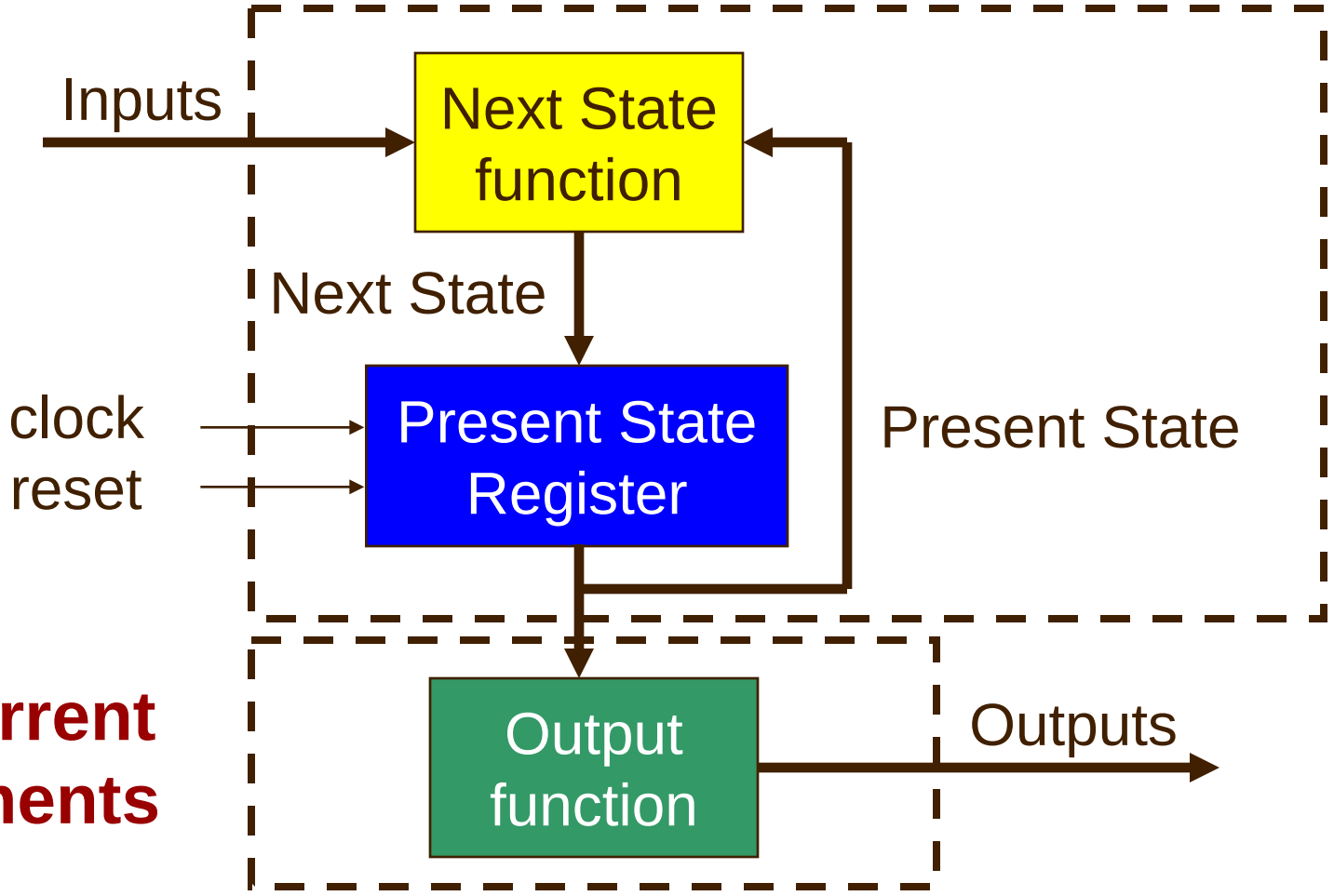
FSMs σε VHDL

- Οι μηχανές πεπερασμένων καταστάσεων περιγράφονται με την βοήθεια processes.

- Τα εργαλεία Synthesis καταλαβαίνουν μία περιγραφή FSM αν ακολουθούνται κάποιοι κανόνες:
  - Οι μεταβάσεις καταστάσεων πρέπει να περιγράφονται μέσα σε ένα process sensitive στο *clock* and *asynchronous reset* signals only
  - Οι έξοδοι περιγράφονται ως παράλληλες εντολές (statements) έξω από ένα process

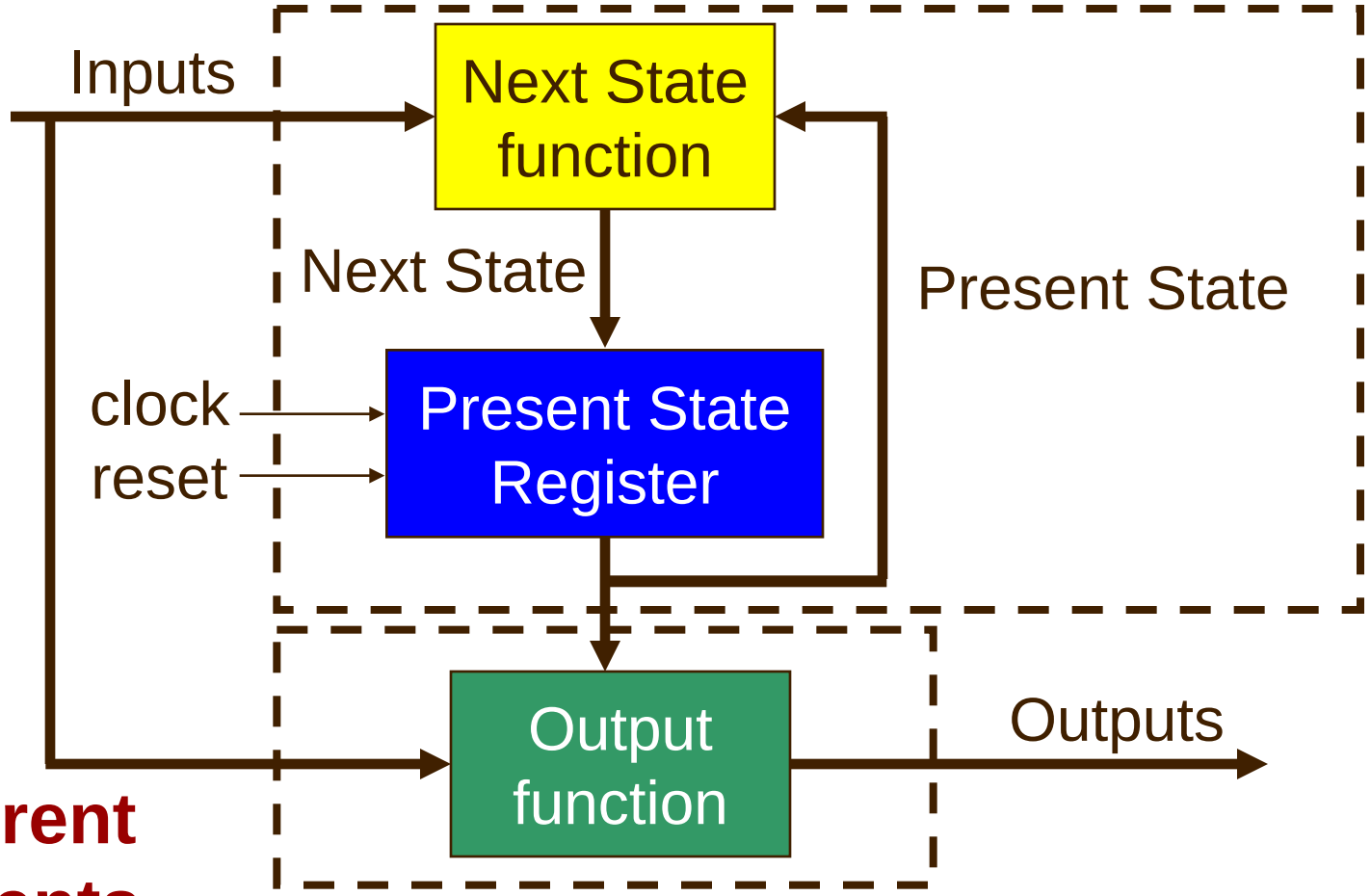
Moore FSM

process(clock, reset)



Mealy FSM

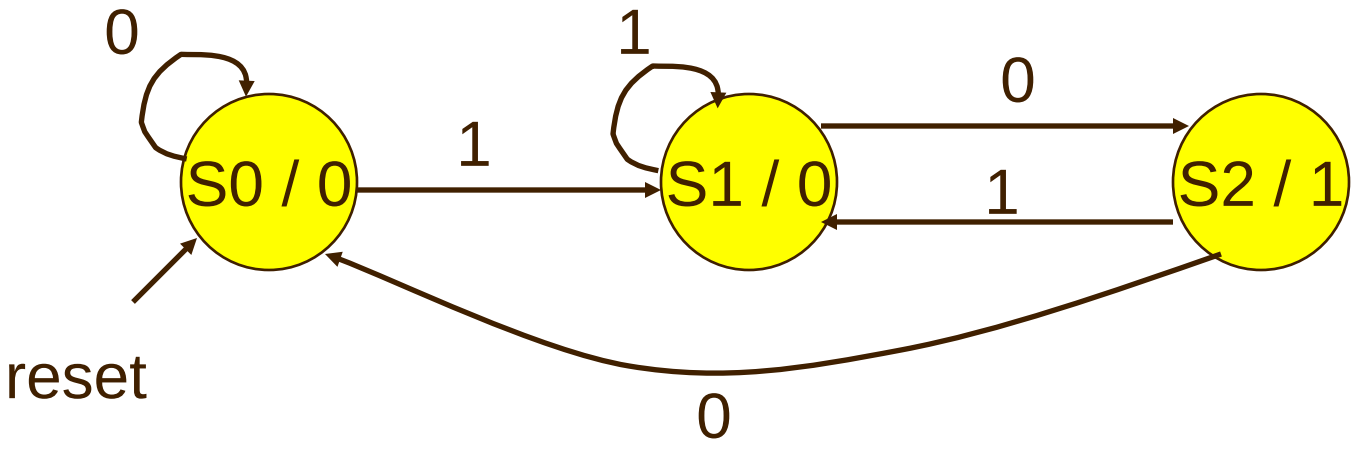
process(clock, reset)



concurrent statements

Moore FSM - Example 1

- Moore FSM that Recognizes Sequence “10”



Moore FSM in VHDL (1)

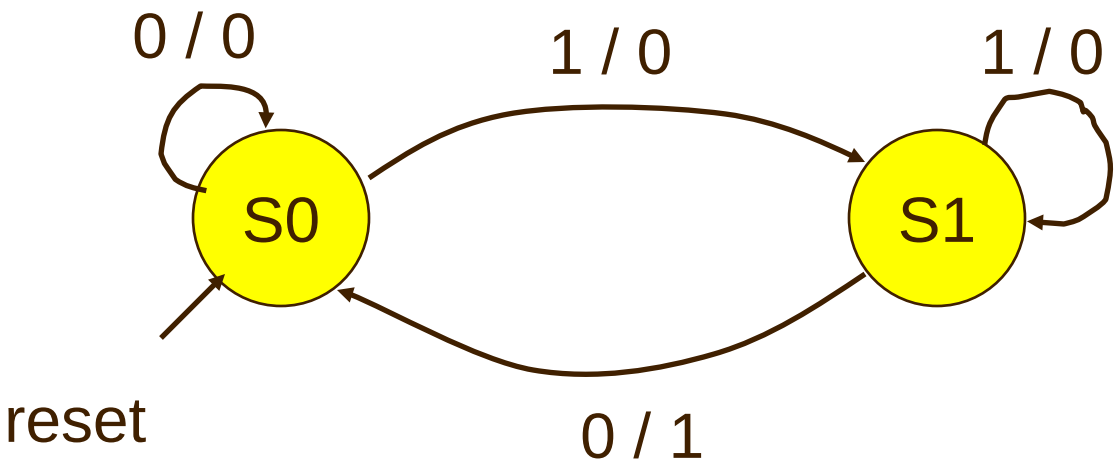
```
TYPE state IS (S0, S1, S2);
SIGNAL Moore_state: state;

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”



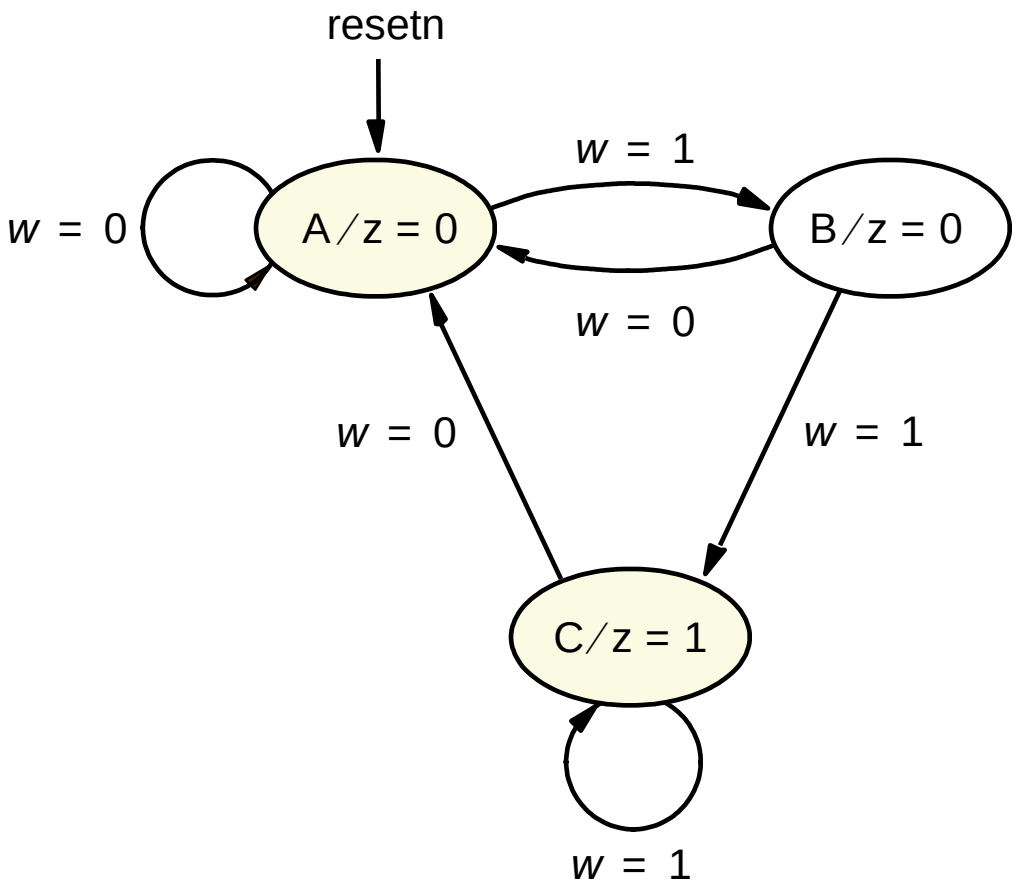
Mealy FSM in VHDL (1)

```
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
                           Mealy_state <= S1;
                      END IF;
            END CASE;
      END IF;
END PROCESS;

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

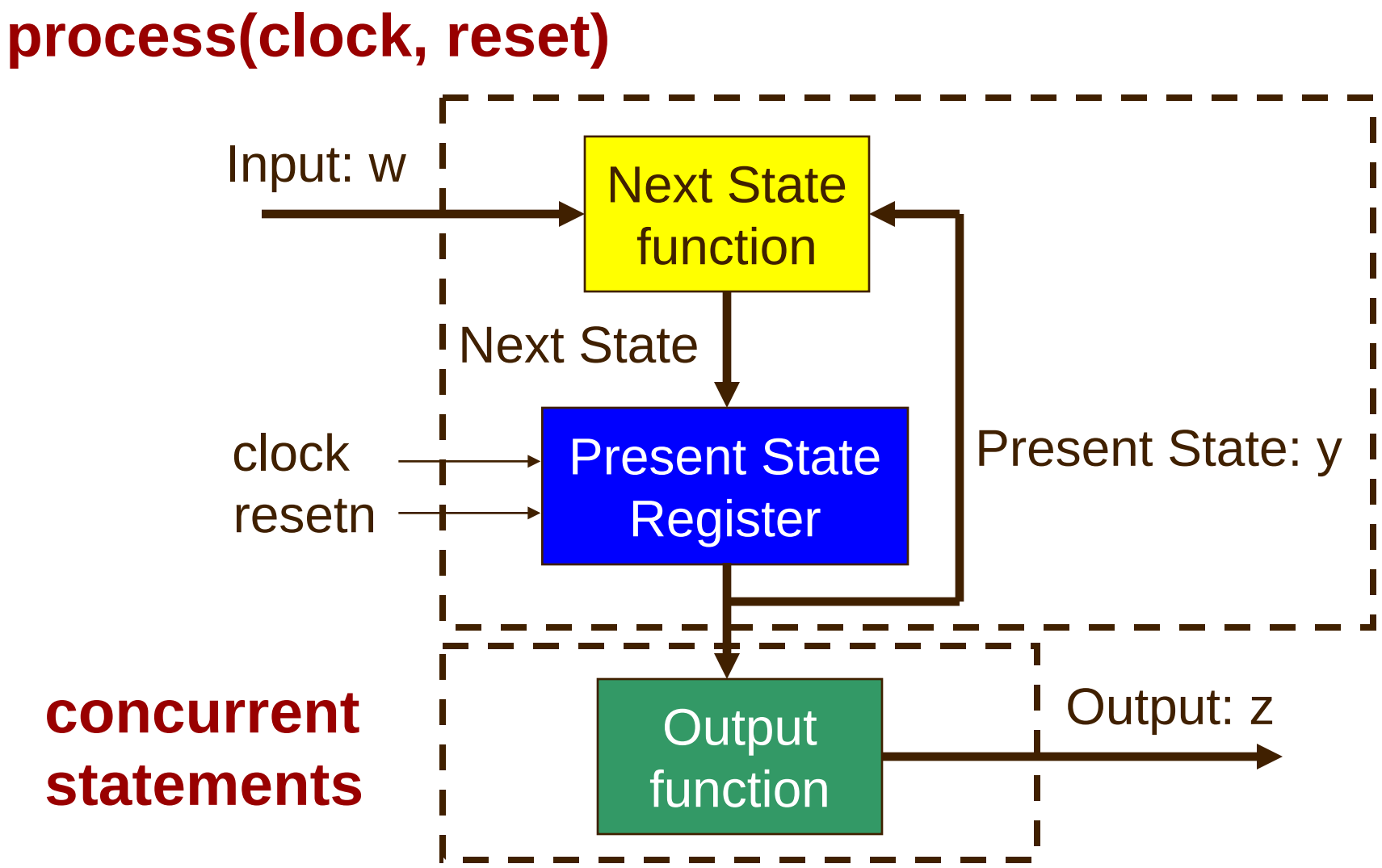
Moore FSM – Example 2: State diagram



Moore FSM – Example 2: State table

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

Moore FSM


process(clock, reset)
Input: w
Next State function
Next State
clock
resetn
Present State Register
Present State: y
concurrent statements
Output function
Output: z

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
```

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 ;
```

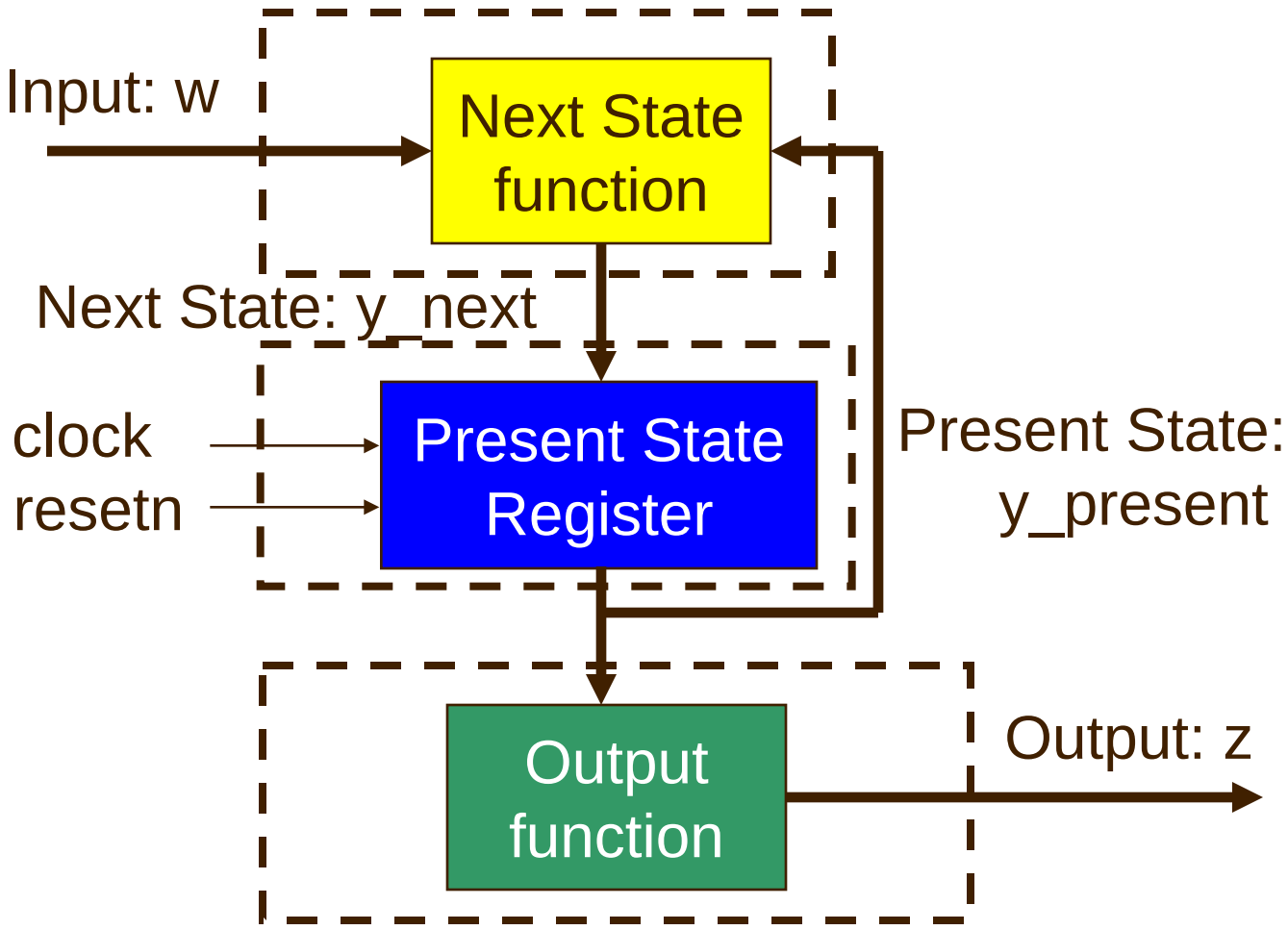
```
        END IF ;
    END PROCESS ;

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

END Behavior ;
```

Moore FSM

process (w, y_present)

process (clock, resetn)

concurrent statements



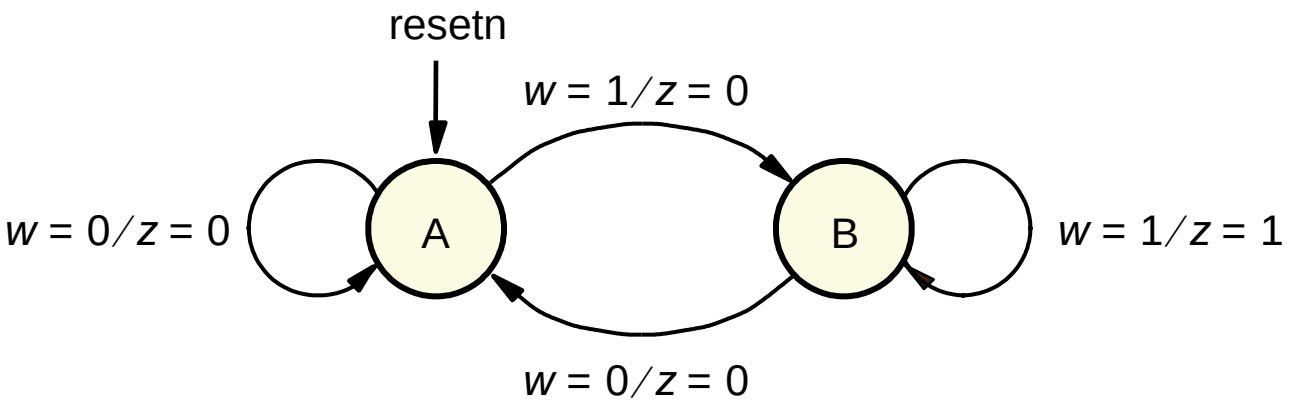
Alternative VHDL code (1)

```
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 ;
```

Mealy FSM – Example 2: State diagram

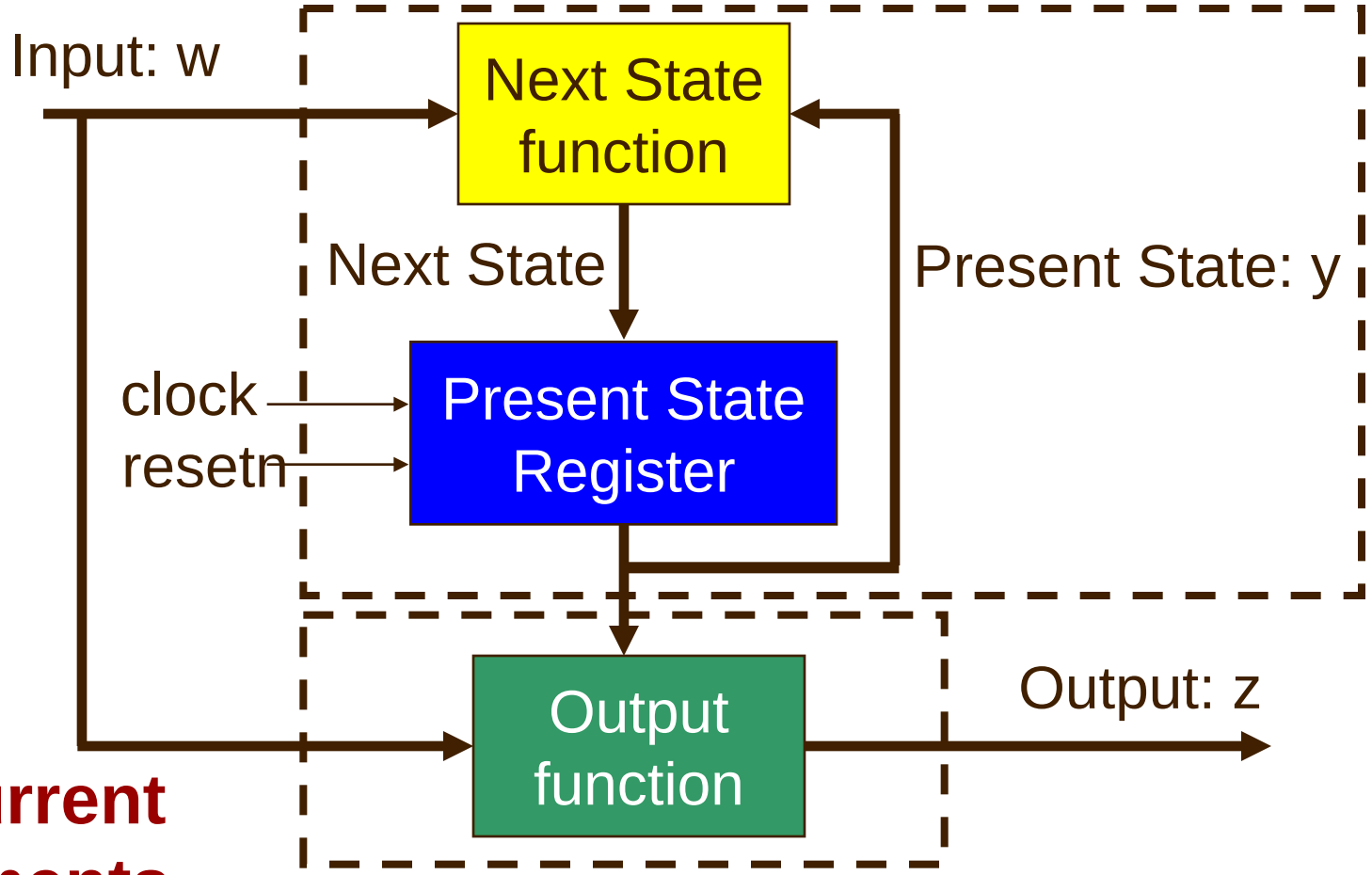


Mealy FSM – Example 2: State table

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
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 ;
```

State Encoding

State Encoding Problem

- State Encoding Can Have a Big Influence on Optimality of the FSM Implementation
  - No methods other than checking all possible encodings are known to produce optimal circuit
  - Feasible for small circuits only
- Using Enumerated Types for States in VHDL Leaves Encoding Problem for Synthesis Tool

Types of State Encodings (1)

- Binary (Sequential) – States Encoded as Consecutive Binary Numbers
  - Small number of used flip-flops
  - Potentially complex transition functions leading to slow implementations
- One-Hot – Only One Bit Is Active
  - Number of used flip-flops as big as number of states
  - Simple and fast transition functions
  - Preferable coding technique in FPGAs

Types of State Encodings (2)

State	Binary Code	One-Hot Code
S0	000	10000000
S1	001	01000000
S2	010	00100000
S3	011	00010000
S4	100	00001000
S5	101	00000100
S6	110	00000010
S7	111	00000001

A user-defined attribute for manual state assignment

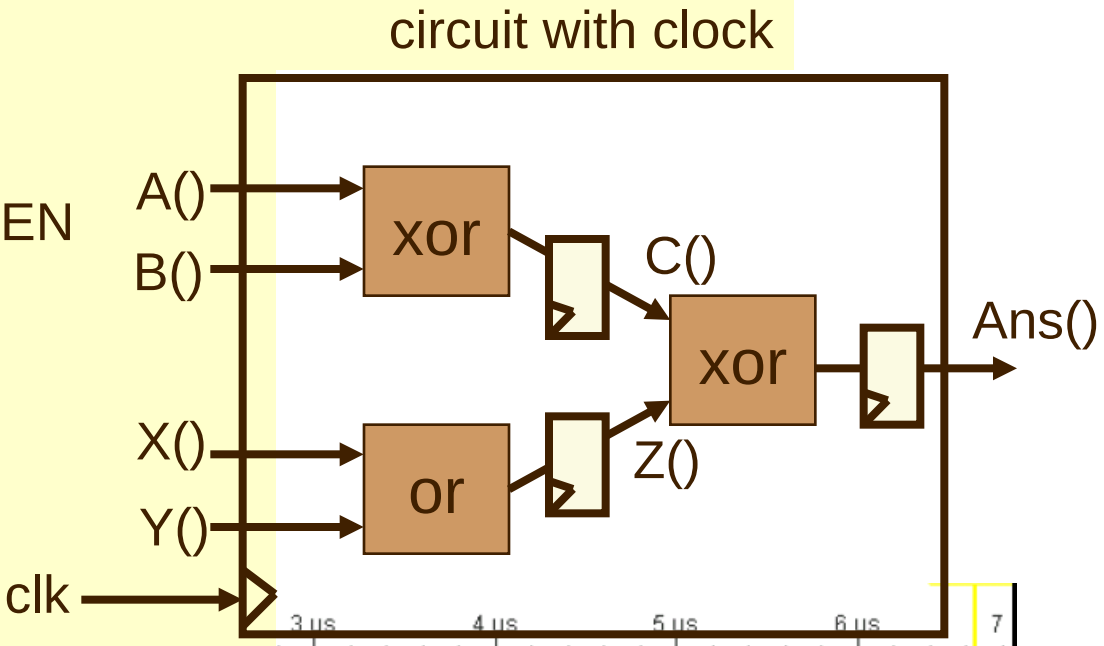
```
(ENTITY declaration not shown)

ARCHITECTURE Behavior OF simple IS
        TYPE State_type IS (A, B, C) ;
        ATTRIBUTE ENUM_ENCODING                    : STRING ;
        ATTRIBUTE ENUM_ENCODING OF State_type      : TYPE IS "00 01 11" ;
        SIGNAL y_present, y_next : State_type ;
BEGIN

con't ...
```

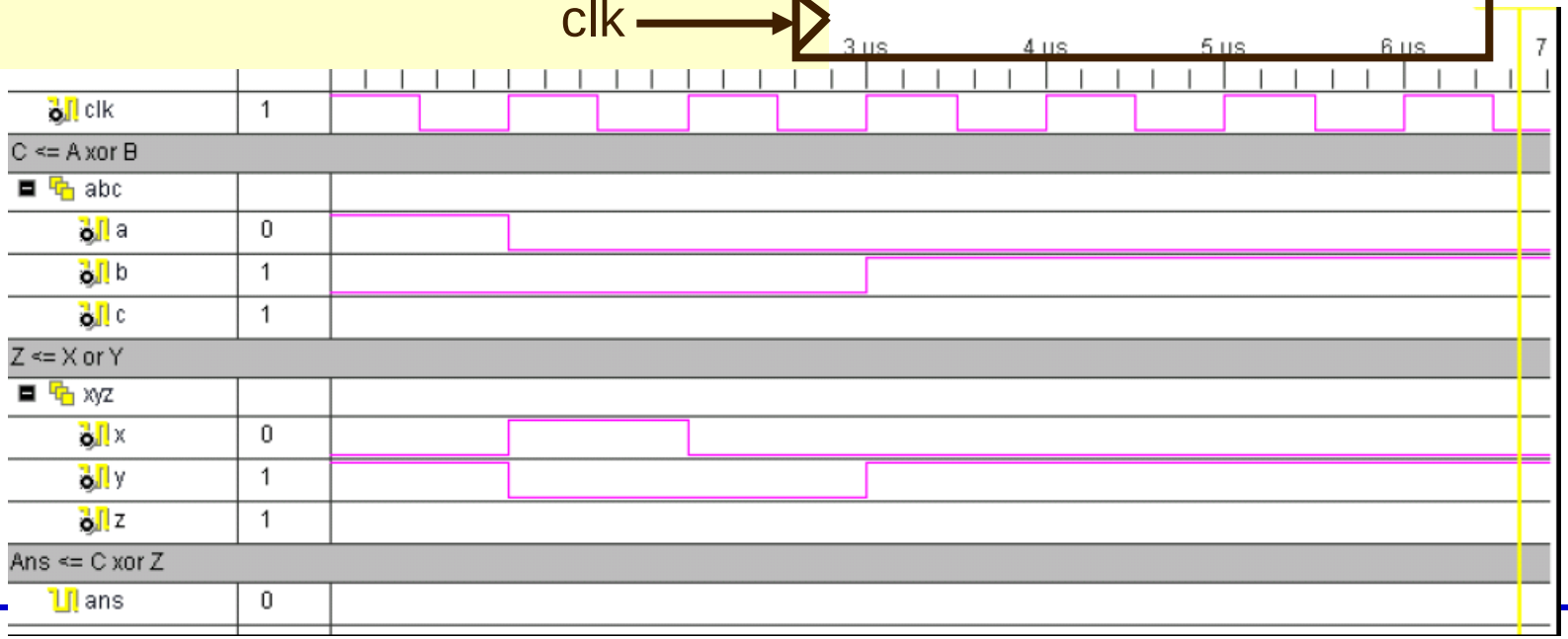
Using constants for manual state assignment (1)

```
ARCHITECTURE Behavior OF simple IS
      SUBTYPE  ABC_STATE is STD_LOGIC_VECTOR(1 DOWNTO 0);

      CONSTANT A  : ABC_STATE := "00" ;
      CONSTANT B  : ABC_STATE := "01" ;
      CONSTANT C  : ABC_STATE := "11" ;

      SIGNAL y_present, y_next : ABC_STATE;
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 ;
… con't
```

Clock Process Example

```
BEGIN
  My_process_1 : process (clk)
  Begin
    IF (clk'event and clk = '1') THEN
      C <= A xor B;
      Z <= X or Y;
      Ans <= C xor Z;
    END IF;
  End My_process_1;
END;
```





Clock Process Example (Answer)

```
BEGIN
  My_process_1 : process (clk)
  Begin
    IF (clk'event and clk = '1') THEN
      C <= A xor B;
      Z <= X or Y;
      Ans <= C xor Z;
    END IF;
  End My_process_1;
END;
```