Finite State Machines in VHDL

Dr. M. M. Shiple

http://Drshiple-courses.weebly.com/

Quote of the Day

http://Drshiple-courses.weebly.com/

Definitions

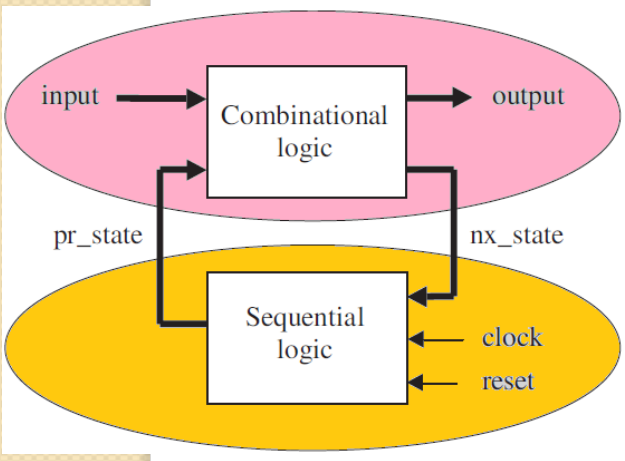
- **A** finite state machine (FSM) is a sequential circuit with "random" next-state logic.

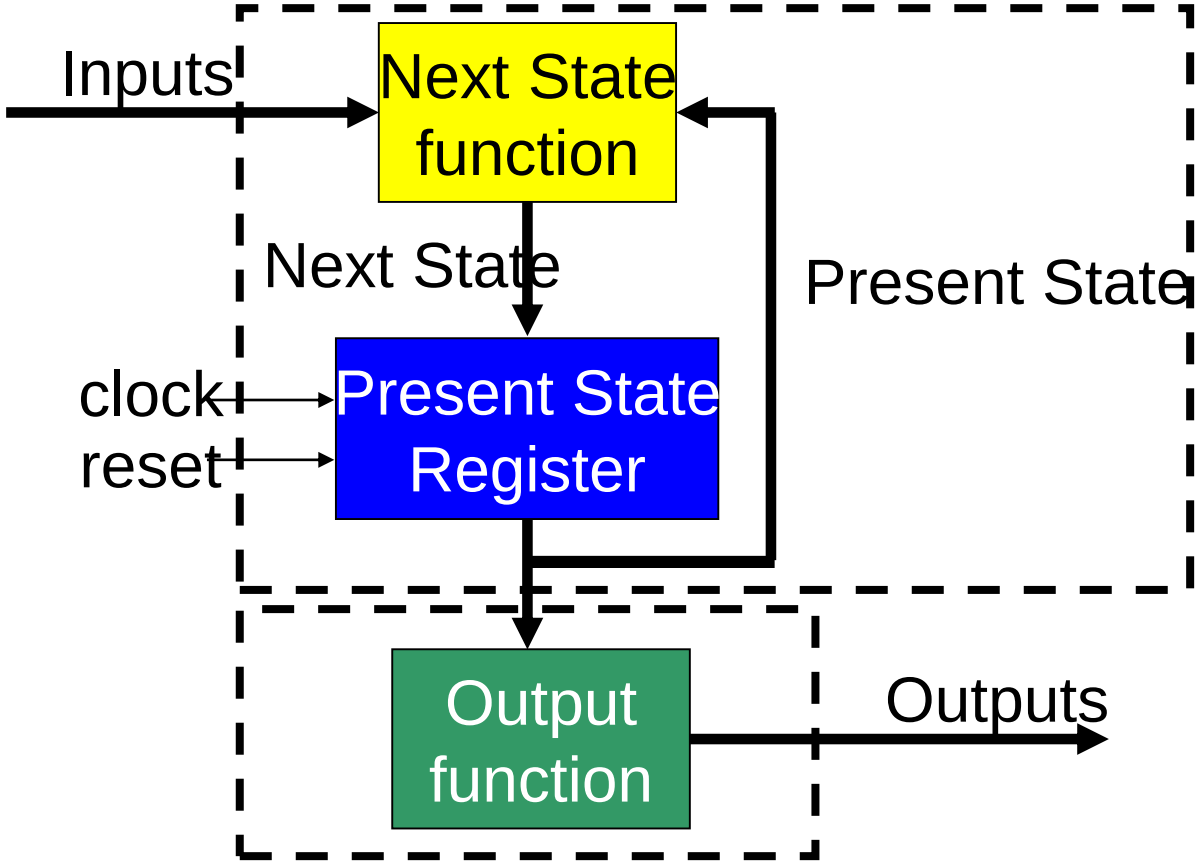
Introduction

- The models selected will influence a design but there are no general indications as to which model is better.

Moore FSM

Process (clock, reset)

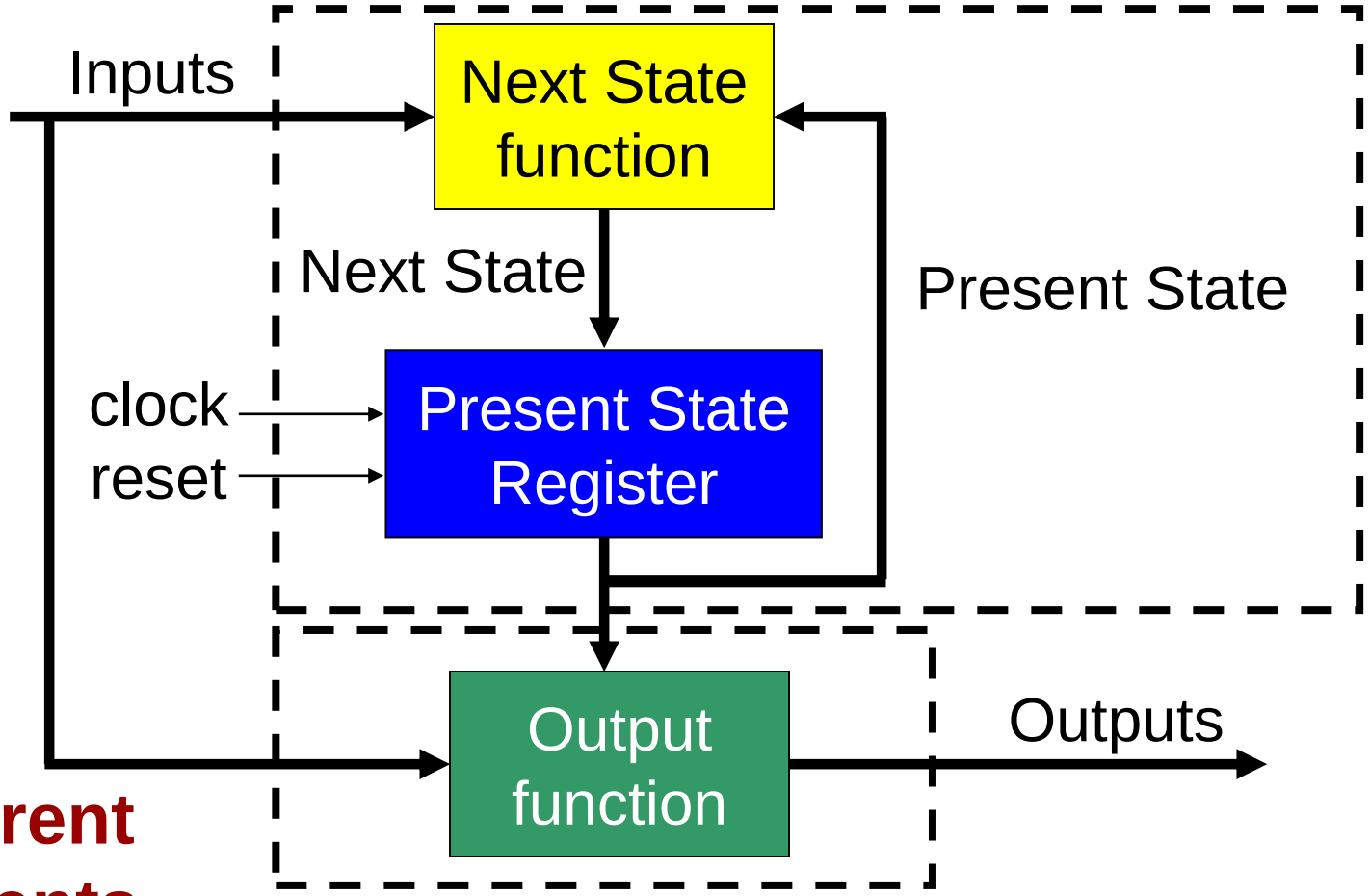




concurrent statements

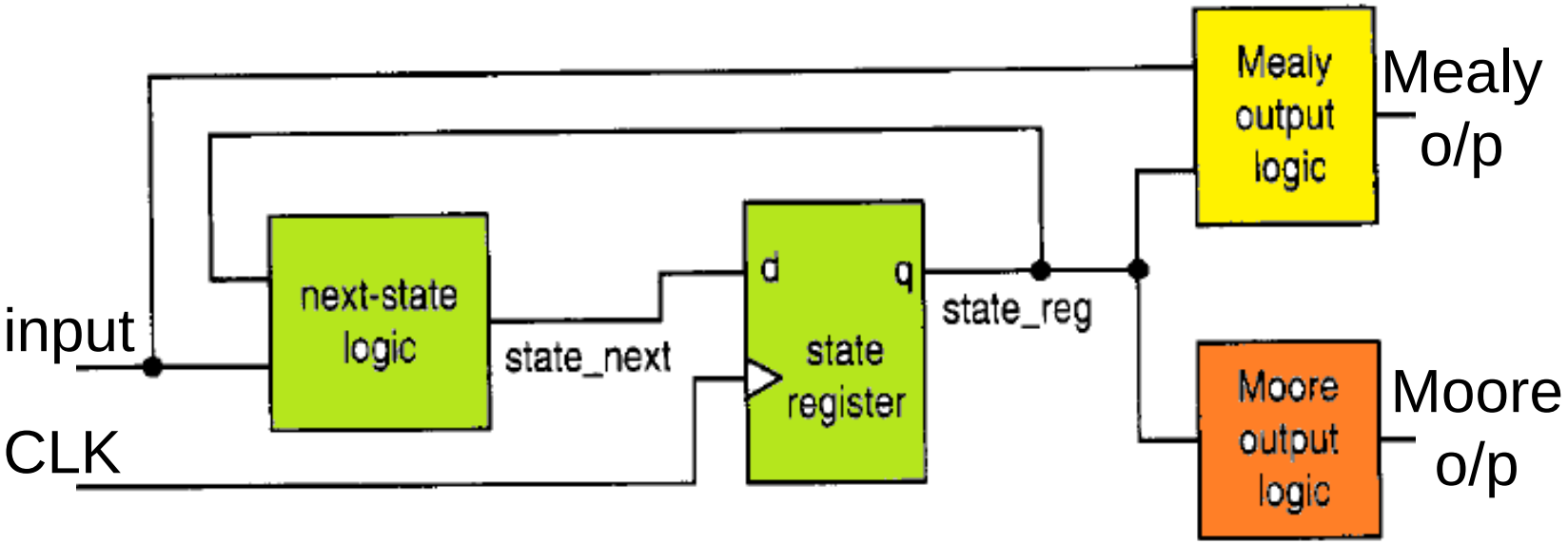
Mealy FSM

Process (clock, reset)



concurrent statements

Moore vs. Mealy

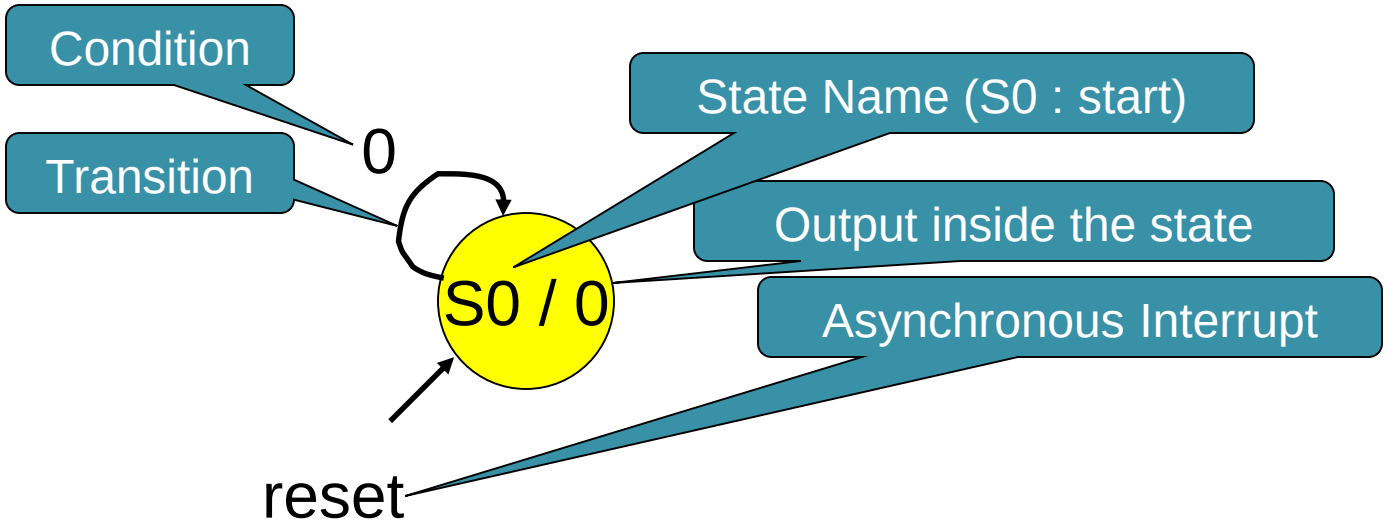


State Diagram by example

http://Drshiple-courses.weebly.com/

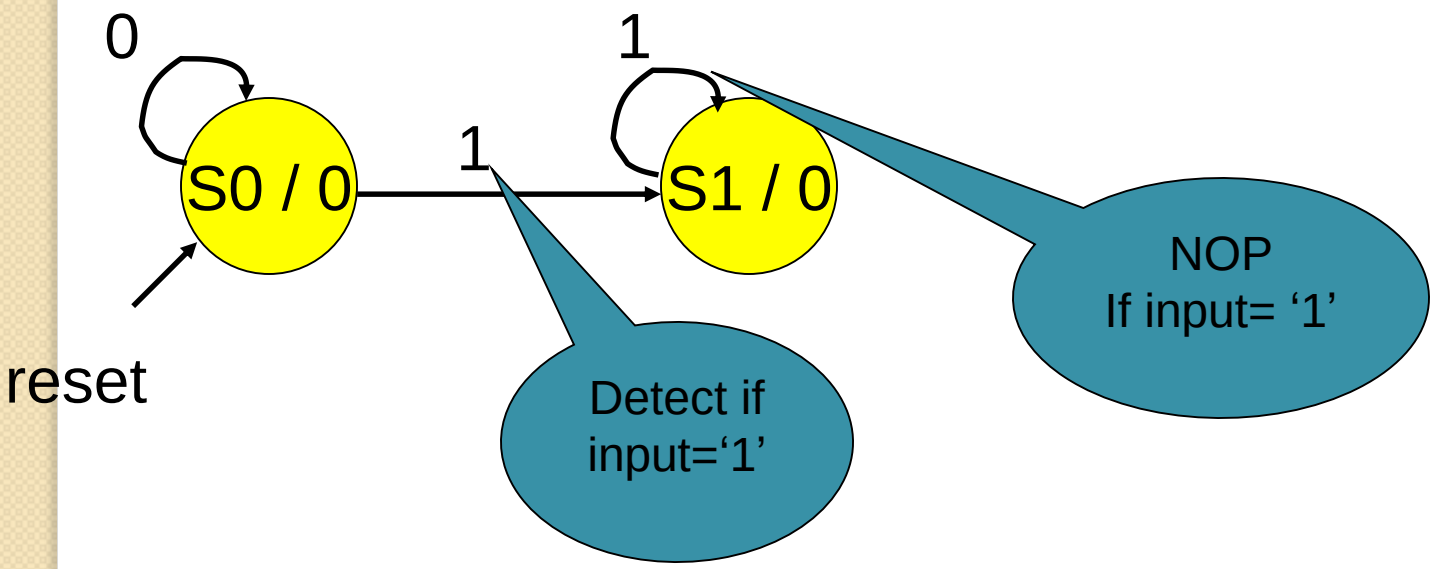
Moore FSM - Example 1

- Draw Moore Recognizes Sequence "1 then 0"



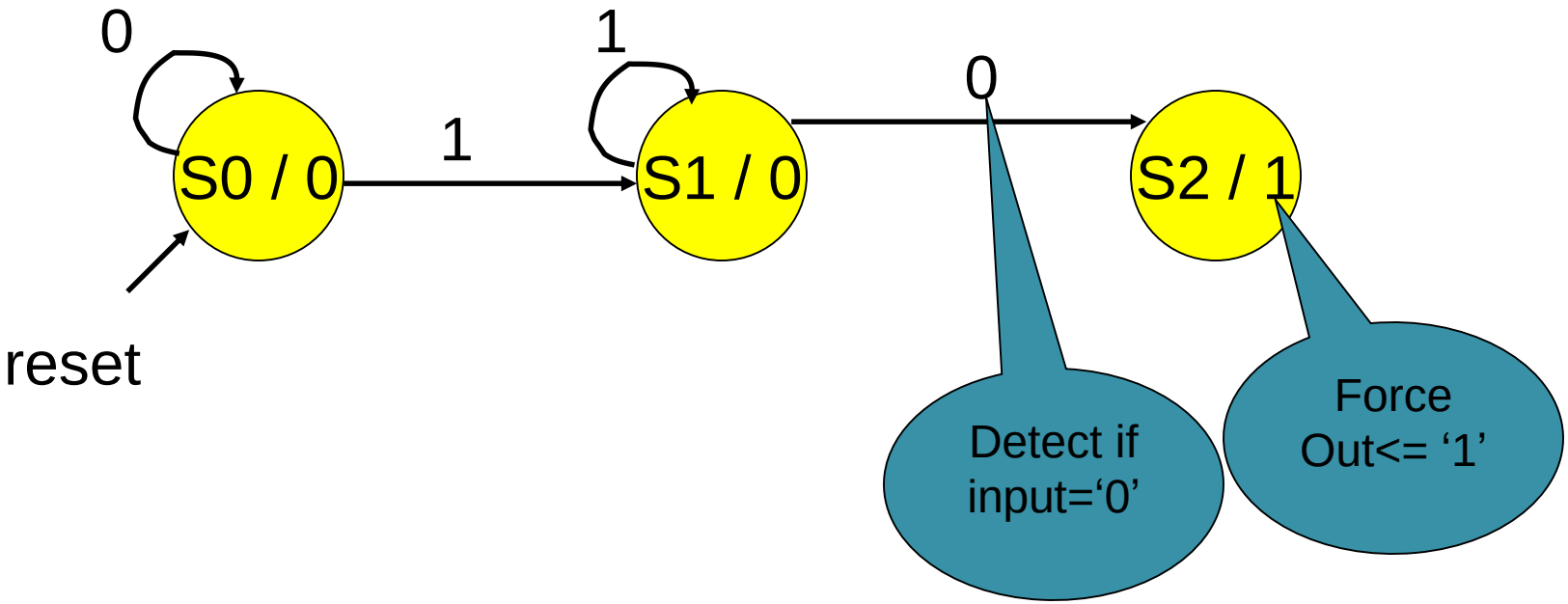
Moore FSM - Example 1

- Define the output if the input='1';

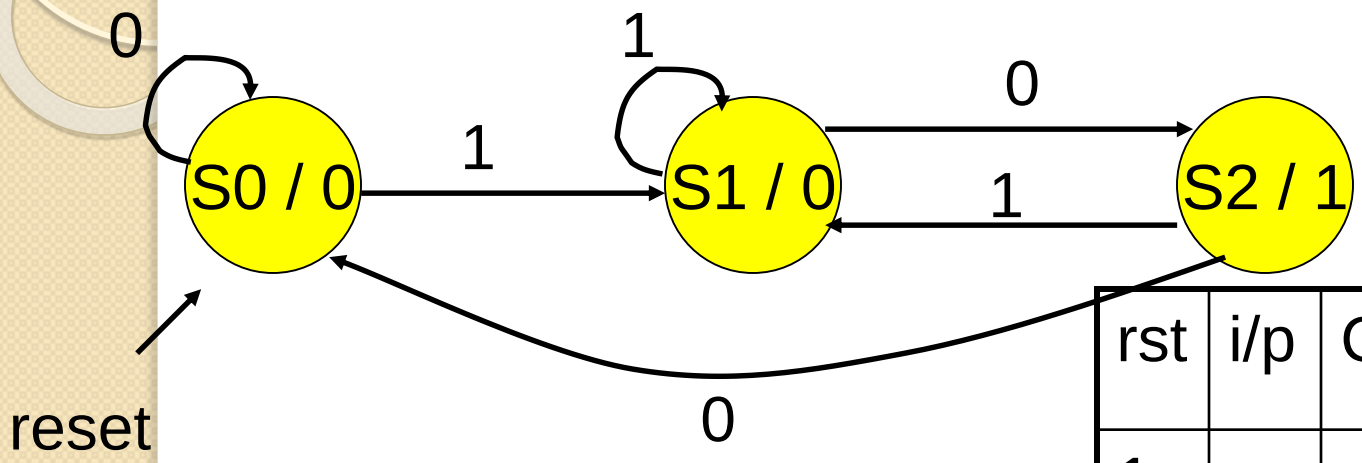


Moore FSM - Example 1

- The sequence complete perfectly



Moore FSM - Example 1

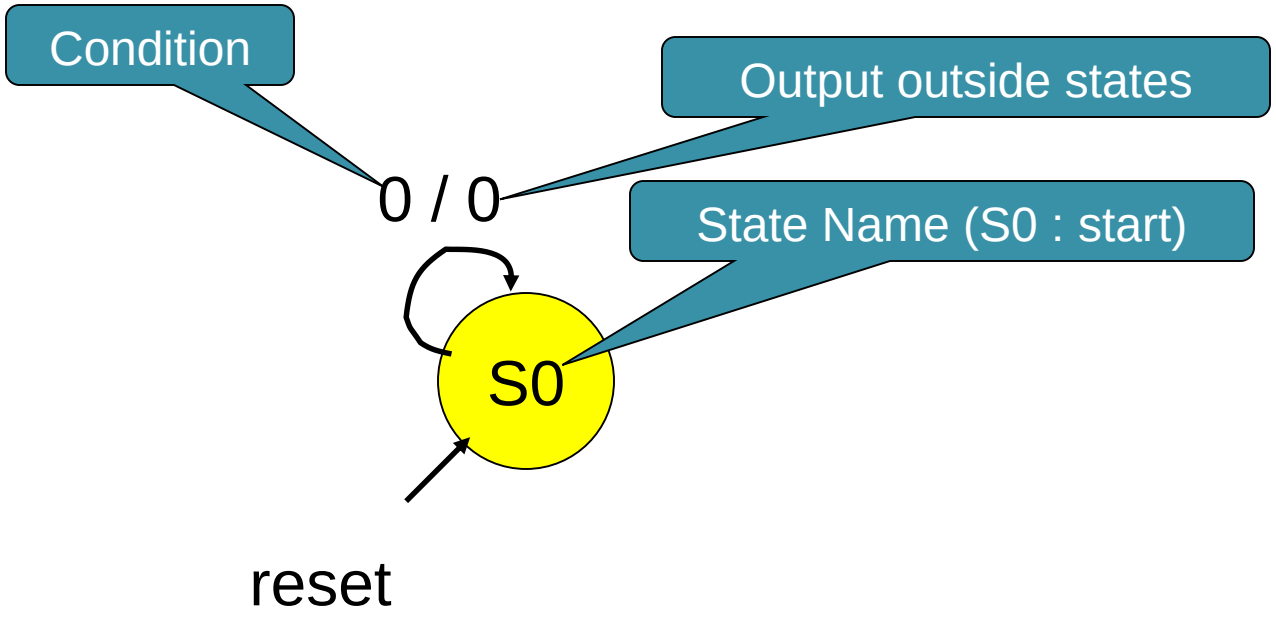


rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S2	1
0	1	S1	S1	0
0	0	S2	S0	0
0	1	S2	S1	0

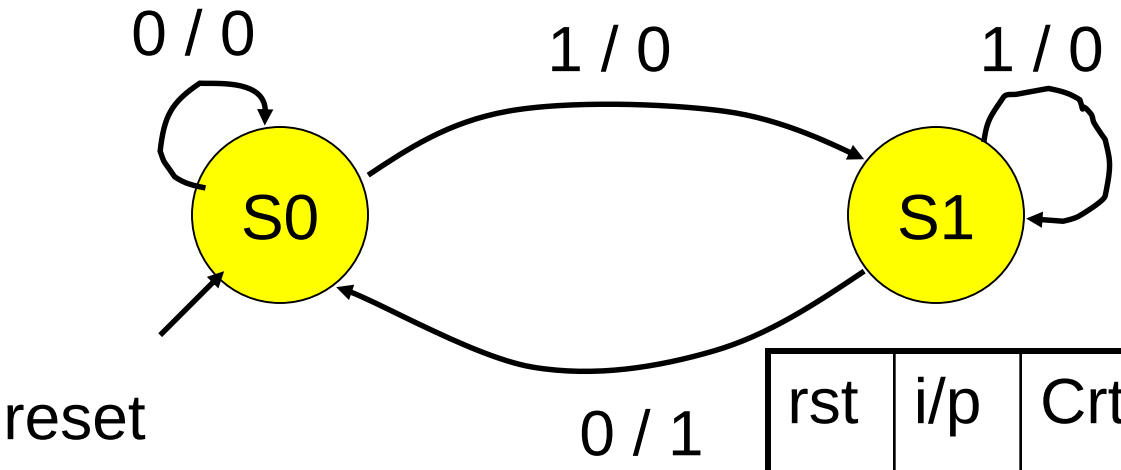
Each output cases need unique state

Mealy FSM - Example 1

- Draw sequence recognition with mealy



Mealy FSM - Example 1



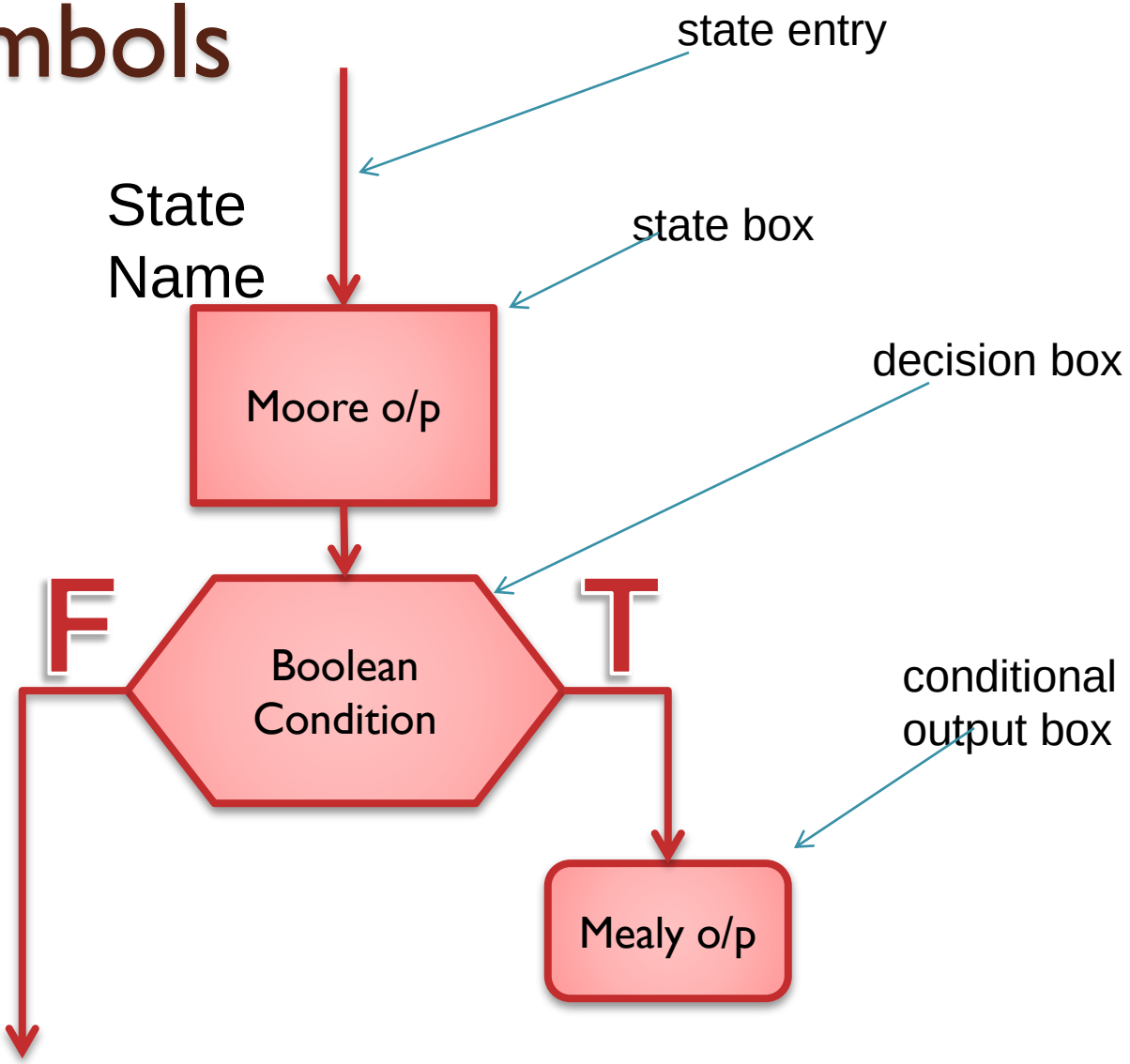
rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S0	1
0	1	S1	S1	0

Algorithmic State Machine (ASM)

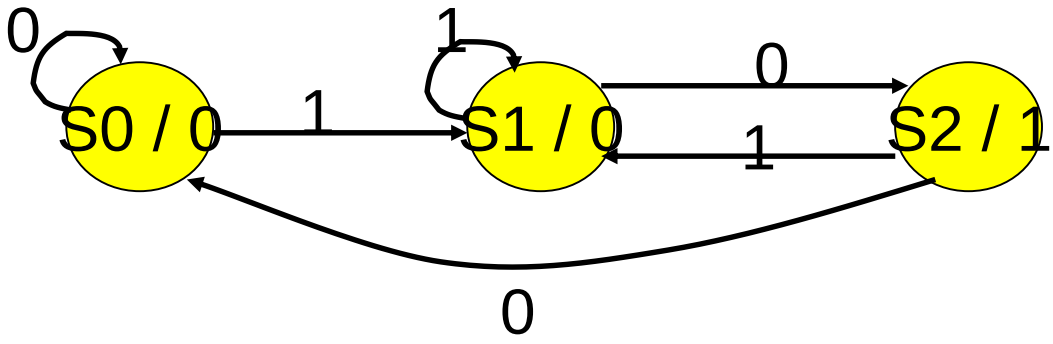
by example

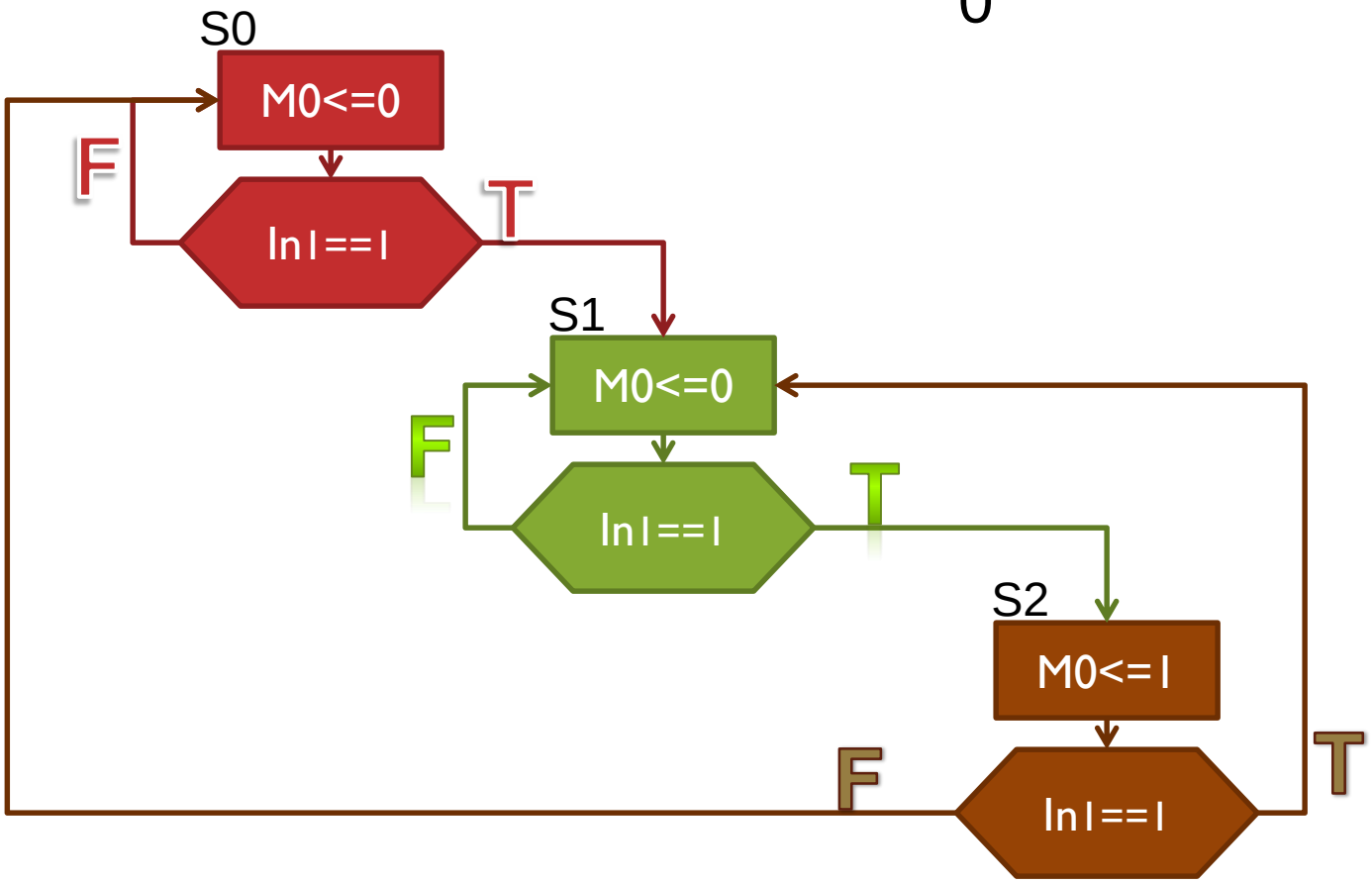
http://Drshiple-courses.weebly.com/

ASM Symbols


state entry
State
Name
state box
Moore o/p
decision box
F
Boolean
Condition
T
conditional
output box
Mealy o/p

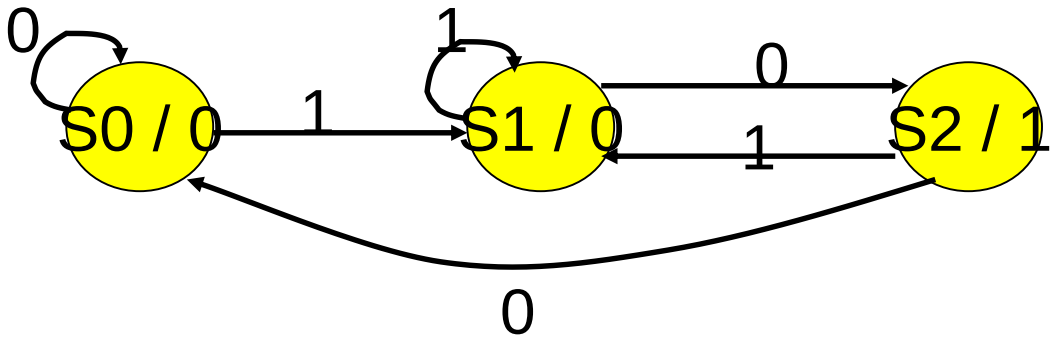
Moore ASM

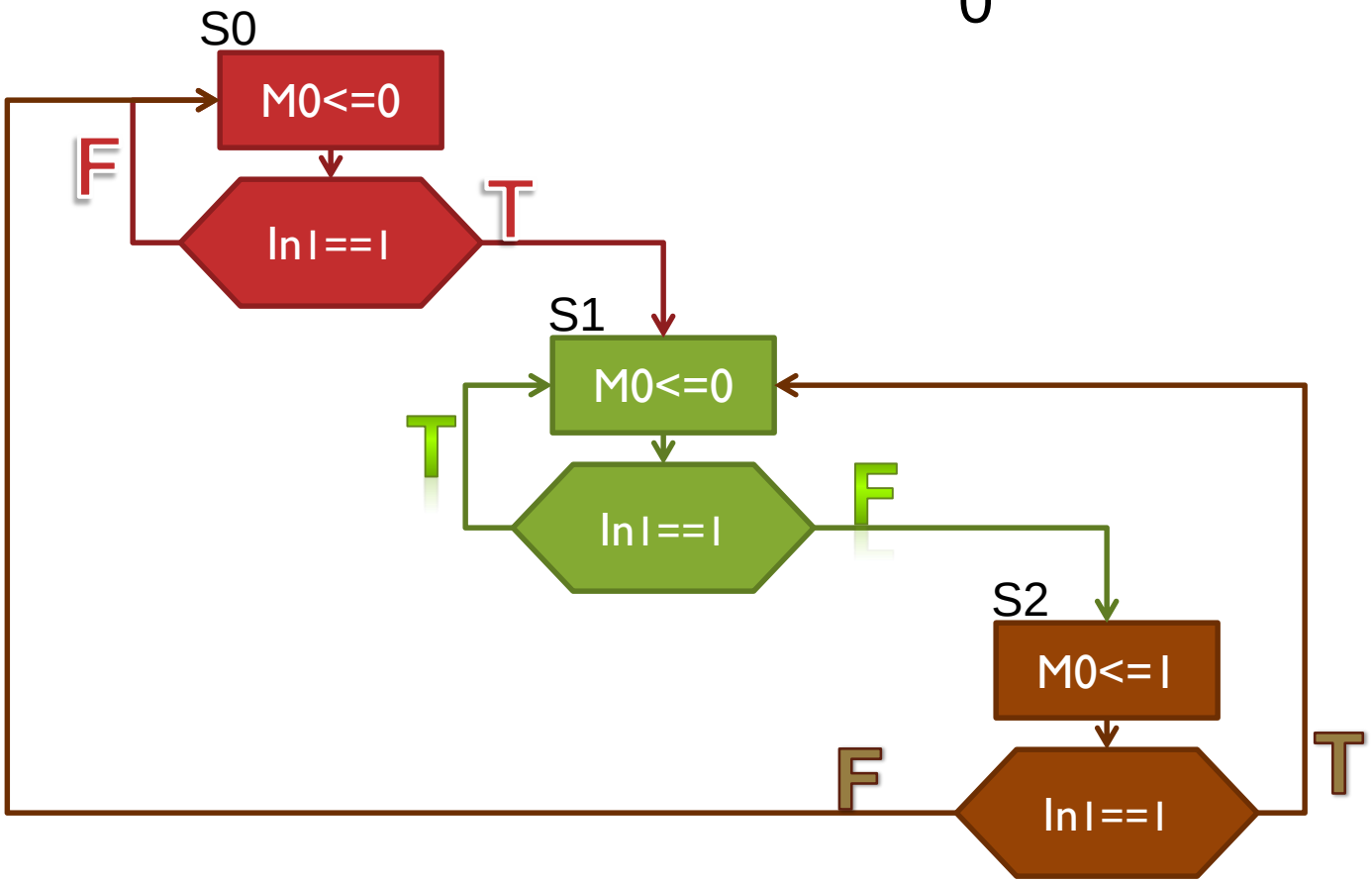




http://Drshiple-courses.weebly.com/

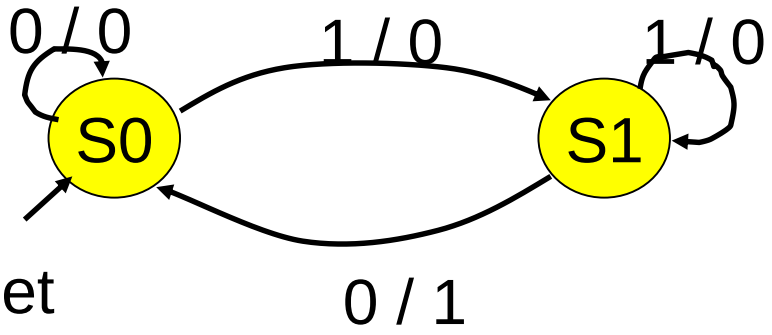
Moore ASM

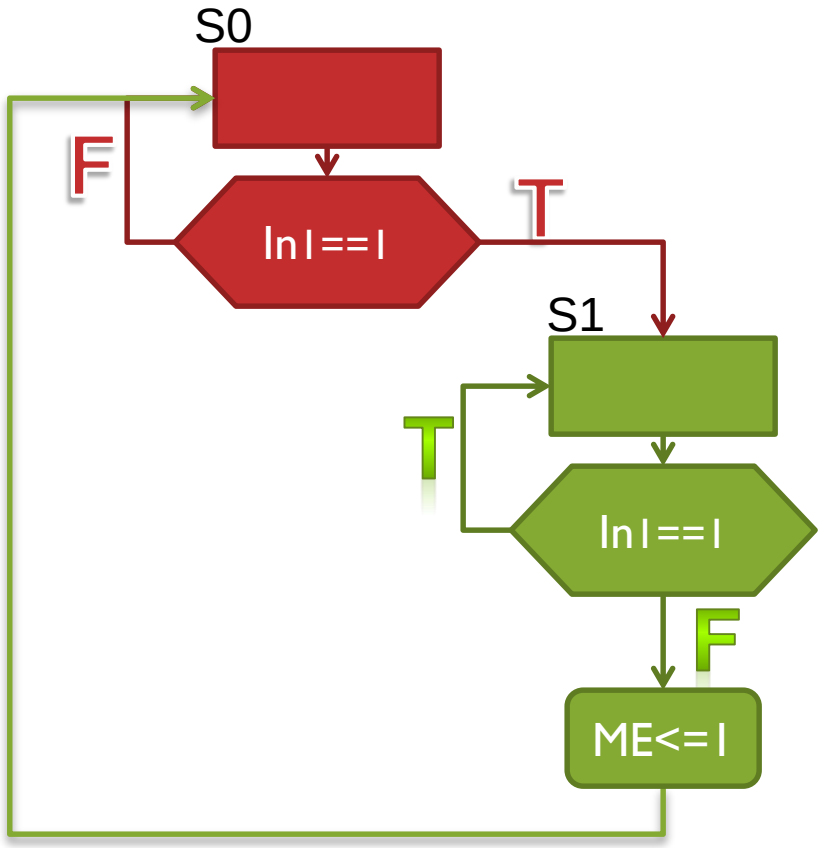




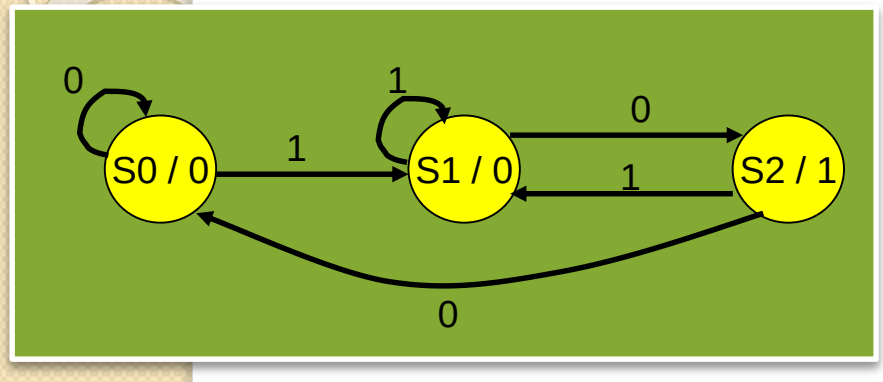
http://Drshiple-courses.weebly.com/

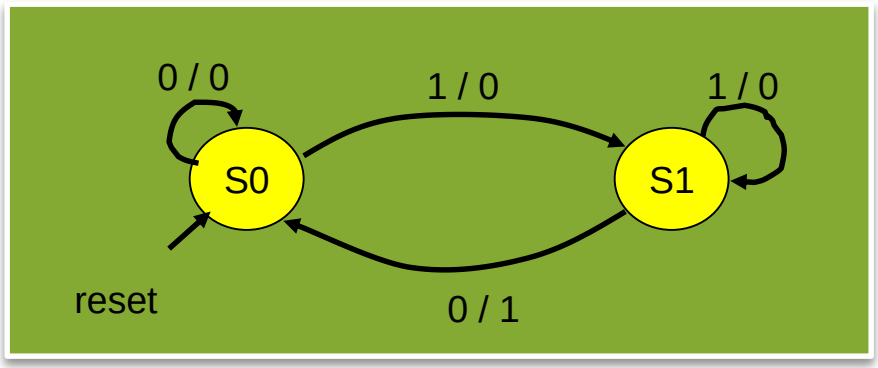
Mealy ASM





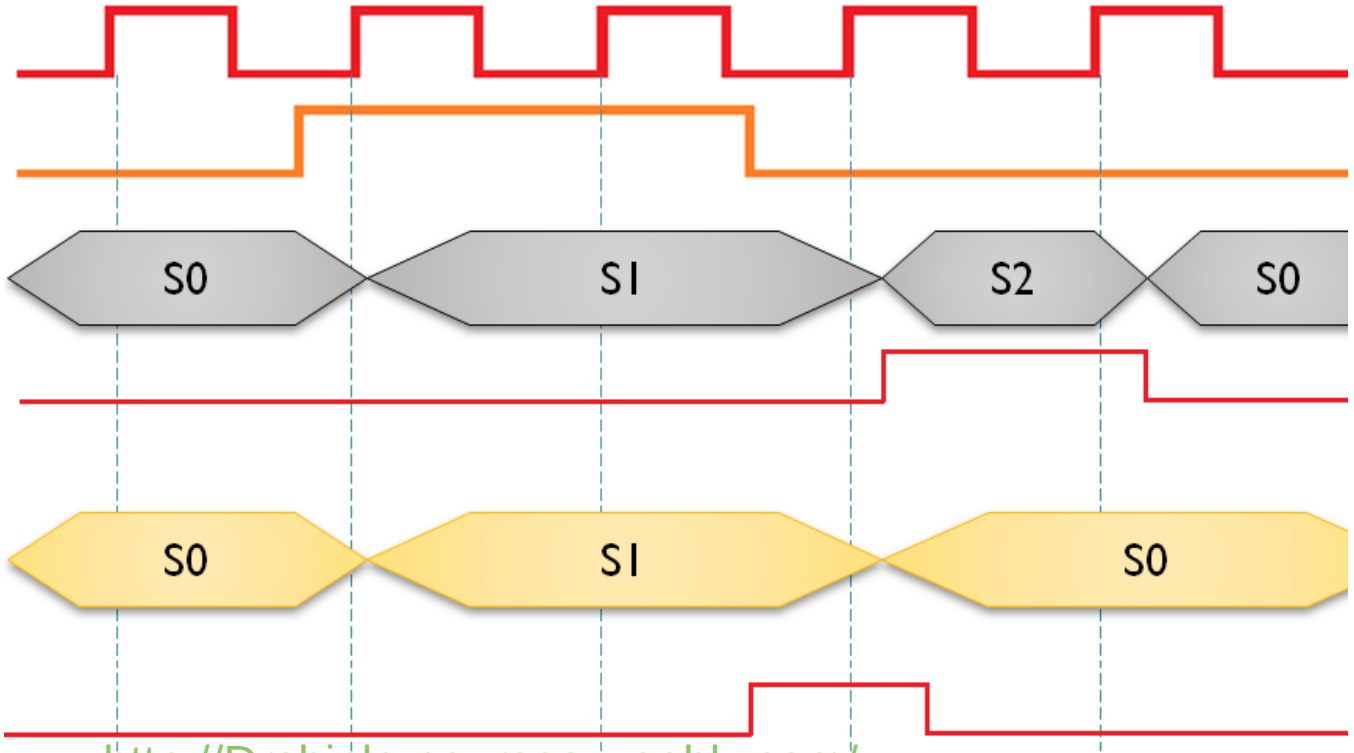
Moore vs. Mealy





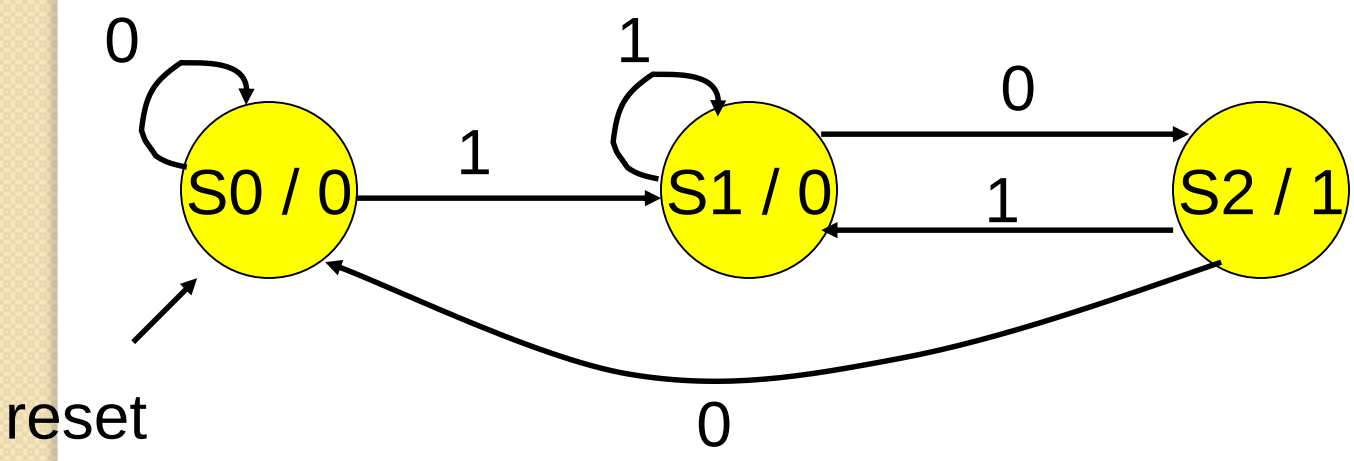
Mo

Me

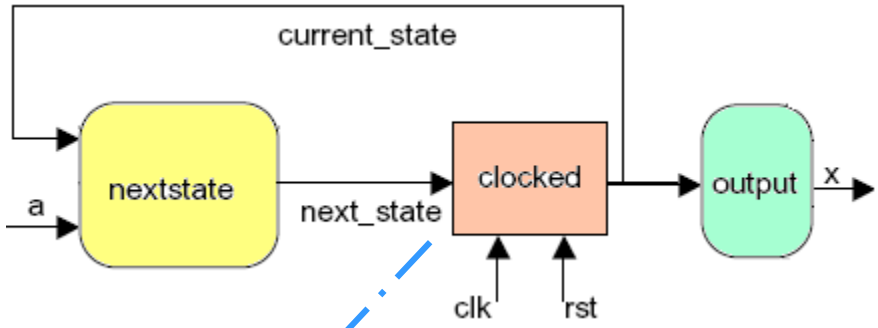


Moore FSM - Example 1

- VHDL of Moore: Again



Moore FSM in VHDL (1)

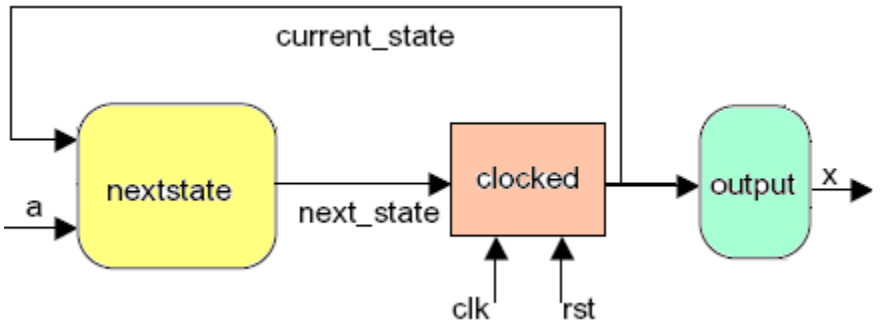



```
-------------------------------------------------------
clocked : PROCESS ( clk, rst )
-------------------------------------------------------
BEGIN
  IF (rst = '1') THEN
      current_state <= s0;
      -- Reset Values
  ELSIF (clk'EVENT AND clk = '1') THEN
      current_state <= next_state;
      -- Default Assignment To Internals
  END IF;
END PROCESS clocked;
```

rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S2	1
0	1	S1	S1	0
0	0	S2	S0	0
0	1	S2	S1	0

Moore FSM in VHDL (1)

```
nextstate : PROCESS ( a, b, current_state )
-----------------------------------------------------
BEGIN
CASE current_state IS
        WHEN S0 =>IF input = '1' THEN
                    Next_state <= S1;
              ELSE
                    Next_state <= S0;
              END IF;
        WHEN S1 =>IF input = '0' THEN
                    Next_state <= S2;
              ELSE
                    Next_state <= S1;
              END IF;
        WHEN S2 =>IF input = '0' THEN
                    Next_state <= S0;
               ELSE
                    Next_state <= S1;
              END IF;
             END CASE;
END PROCESS nextstate
```



rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S2	1
0	1	S1	S1	0
0	0	S2	S0	0
0	1	S2	S1	0

Moore FSM in VHDL (2)

Output <= ‘1’ WHEN current_state = S2 ELSE ‘0’;

```
-------------------------------------------------------
output : PROCESS ( current_state )
-------------------------------------------------------
BEGIN
-- Default Assignment
 o/p <= '0';
-- Default Assignment To Internals
-- State Actions
  CASE current_state IS
            WHEN s2 => o/p <= '1';
            WHEN OTHERS => NULL;
  END CASE;

END PROCESS output;
```

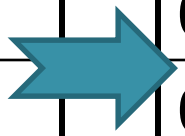
rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S2	1
0	1	S1	S1	0
0	0	S2	S0	0
0	1	S2	S1	0

FSM Implementation

Moore paradigm

Enumerate Data type

rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S2	1
0	1	S1	S1	0
0	0	S2	S0	0
0	1	S2	S1	0

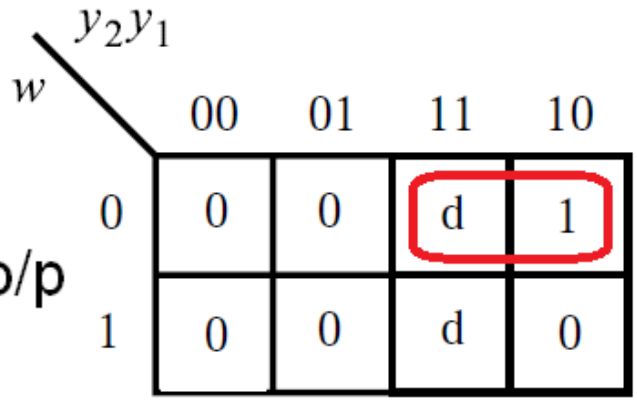


rst	w	y_1y_2	y_1y_2	o/p
1	-	-	00	0
0	0	00	00	0
0	1	00	01	0
0	0	01	10	1
0	1	01	01	0
0	0	10	00	0
0	1	10	01	0

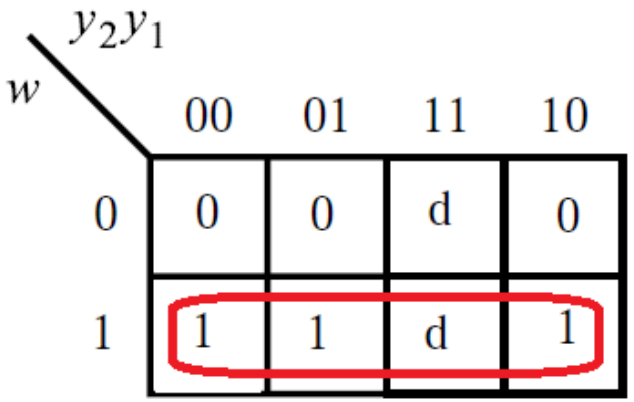
Enumerate Data type

rst	w	y_1y_2	Y_1Y_2	o/p
1	-	-	00	0
0	0	00	00	0
0	1	00	01	0
0	0	01	10	1
0	1	01	01	0
0	0	10	00	0
0	1	10	01	0

$Y_1=\overline{w}$ y_2=o/p



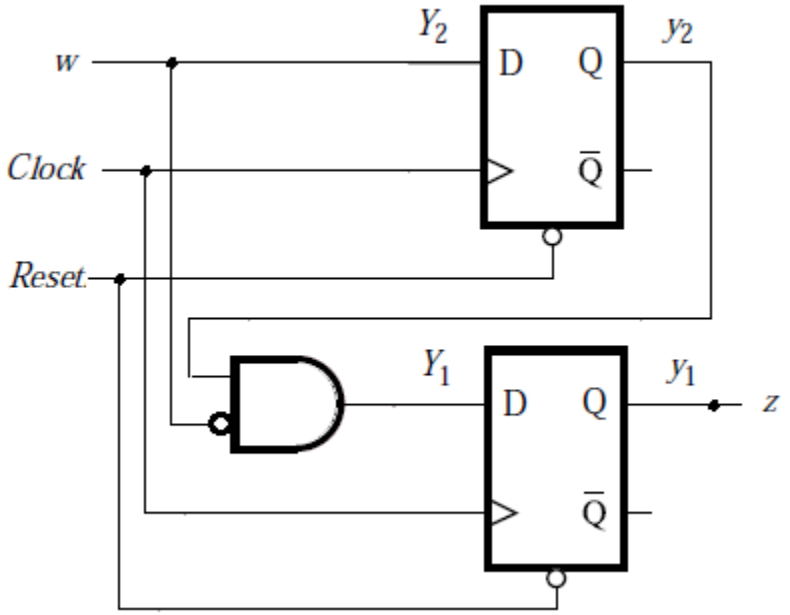
Y_2=w



Realization

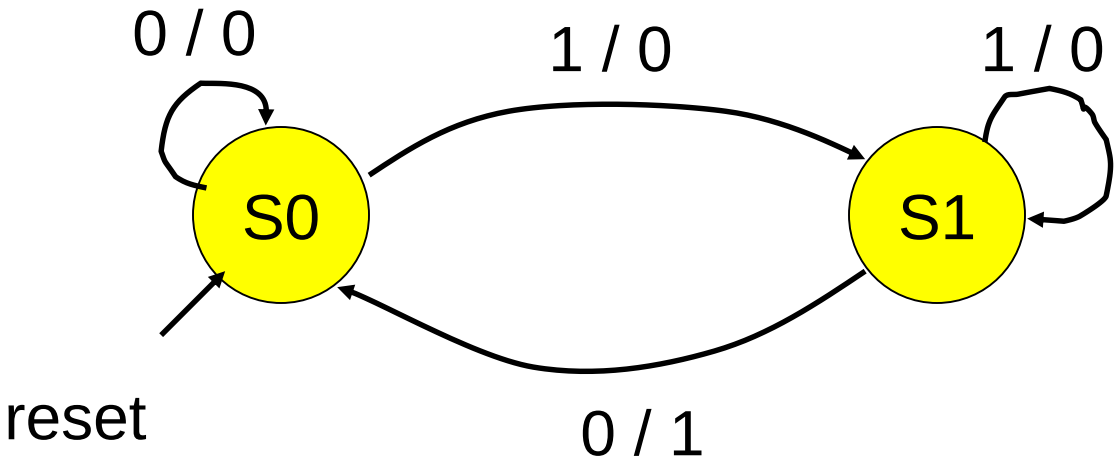
$Y_2 = w$

$Y_1 = \overline{w}\, y_2 = o/p$



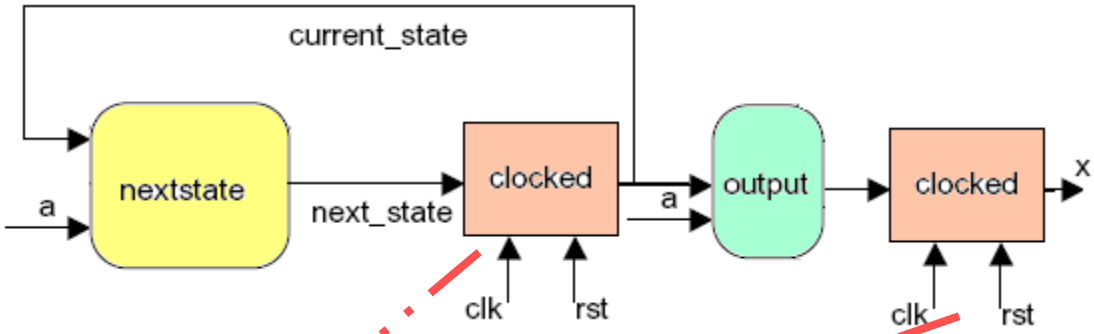
Mealy FSM - Example 1

- Mealy FSM that Recognizes Sequence "10"



Mealy FSM in VHDL (1)

```
-------------------------------------
clocked : PROCESS ( clk, rst )
-------------------------------------
BEGIN
  IF (rst = '1') THEN
          current_state <= s0;
          -- Reset Values
          x <= '0';
  ELSIF (clk'EVENT AND clk = '1') THEN
          current_state <= next_state;
          -- Registered output assignments
          x <= x_int;
          -- Default Assignment To Internals
  END IF;
END PROCESS clocked;
```

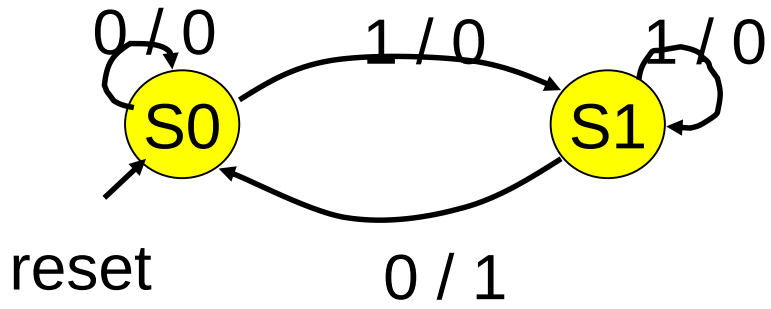


rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S0	1
0	1	S1	S1	0

Mealy FSM in VHDL (2)

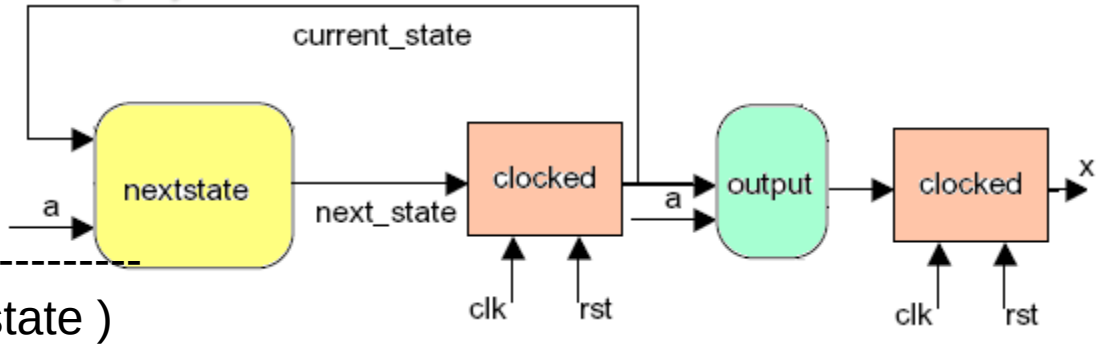
```
-------------------------------------------------------
nextstate : PROCESS ( a, b, current_state )
-------------------------------------------------------
BEGIN
CASE current_state IS
        WHEN S0 =>
            IF input = '1' THEN
              next_state <= S1;
            ELSE
              next_state <= S0;
                  END IF;
  WHEN S1 =>
            IF input = '0' THEN
              next_state <= S0;
            ELSE
              next_state <= S1;
           END if;
  WHEN OTHERS => next_state <= s0;

END CASE;
END PROCESS;
```

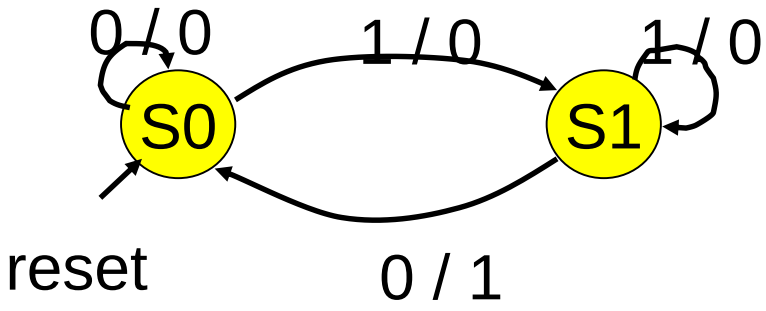


rst	i/p	Crt st	Nxt st	o/p
1	-	-	S0	0
0	0	S0	S0	0
0	1	S0	S1	0
0	0	S1	S0	1
0	1	S1	S1	0

Mealy FSM in VHDL (3)



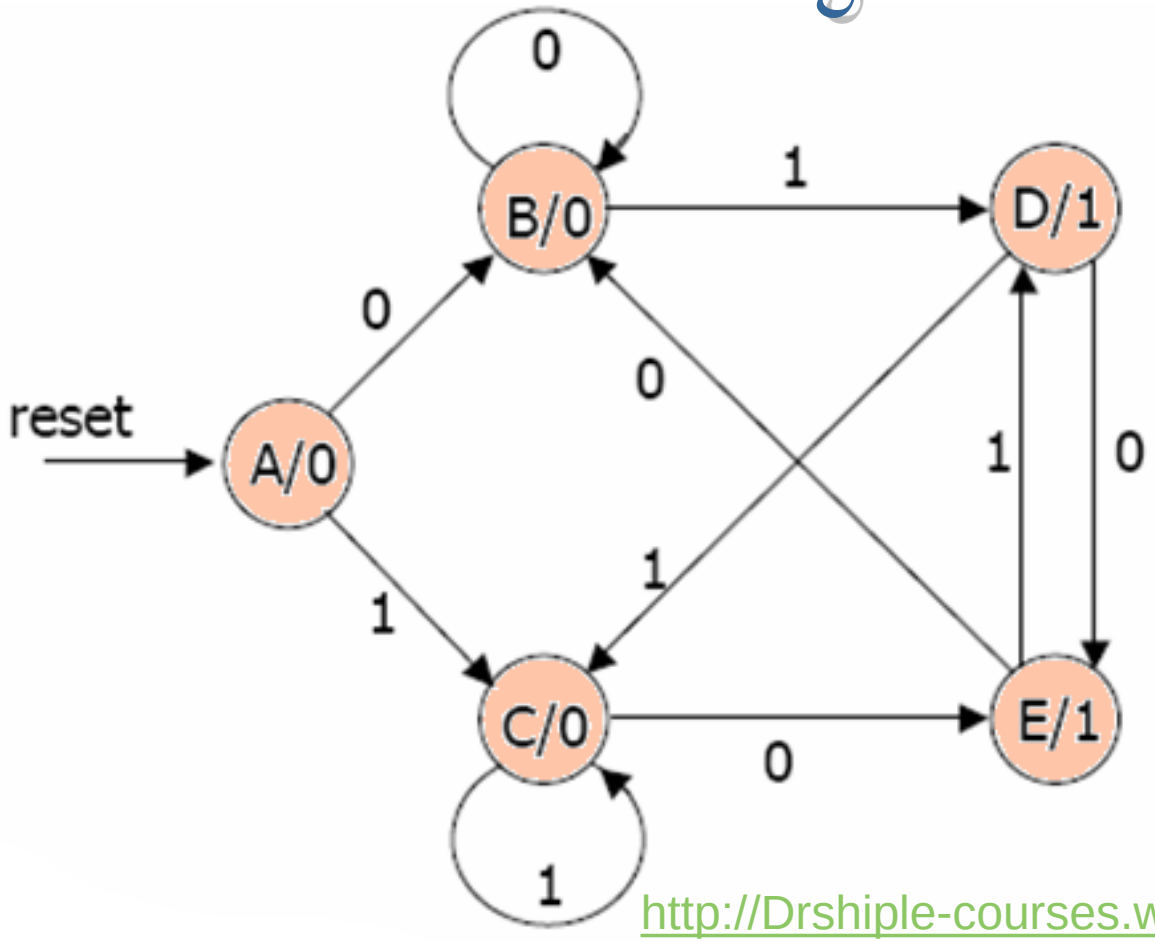

```
-------------------------------------------------------
output : PROCESS ( a, b, current_state )
-------------------------------------------------------
BEGIN
x_int <= '0';
CASE current_state IS
        WHEN s0 => IF (a = '1') THEN
          x_int <= ‘0';
          END IF;
        WHEN s1 => IF (a = '1') THEN
          x_int <= '1';
          END IF;
        WHEN OTHERS => NULL;
END CASE;
END PROCESS output;
```



Challenge your mind

Moore or Mealy? Function?

State the table of the next diagram?

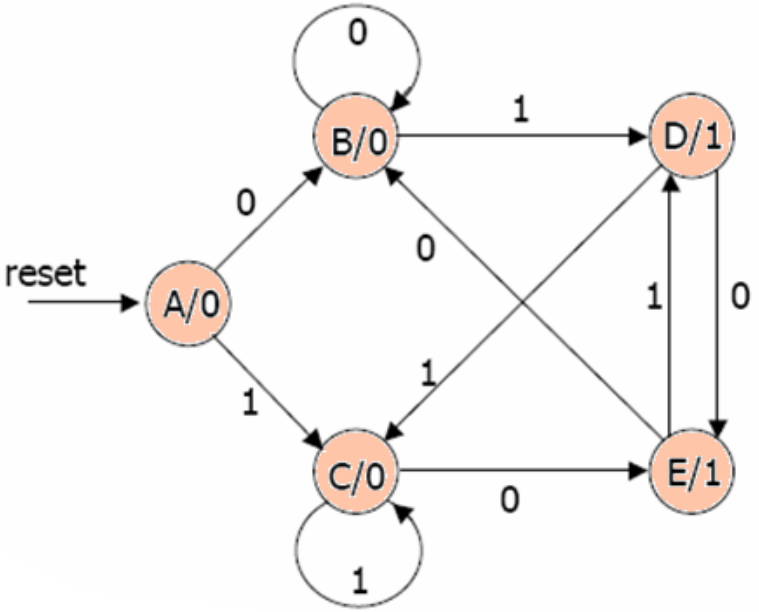


The answer is : Moore

Function : sequence detector for 01 or 10

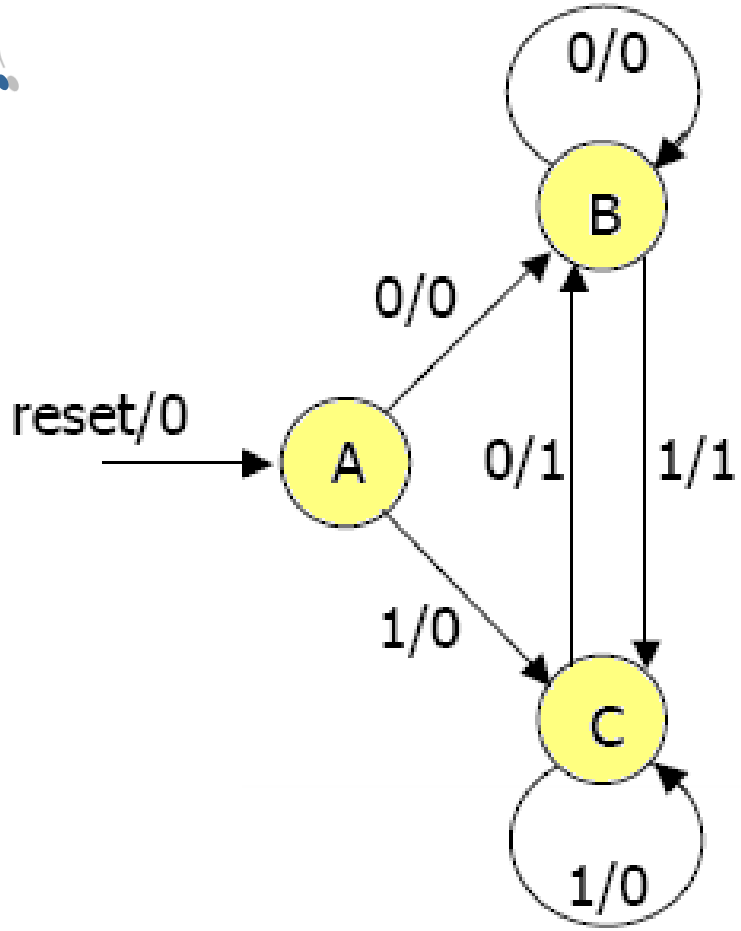
Switch the diagram to be Mealy?

reset	input	current state	next state	output
1	–	–	A	
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	D	0
0	0	C	E	0
0	1	C	C	0
0	0	D	E	1
0	1	D	C	1
0	0	E	B	1
0	1	E	D	1



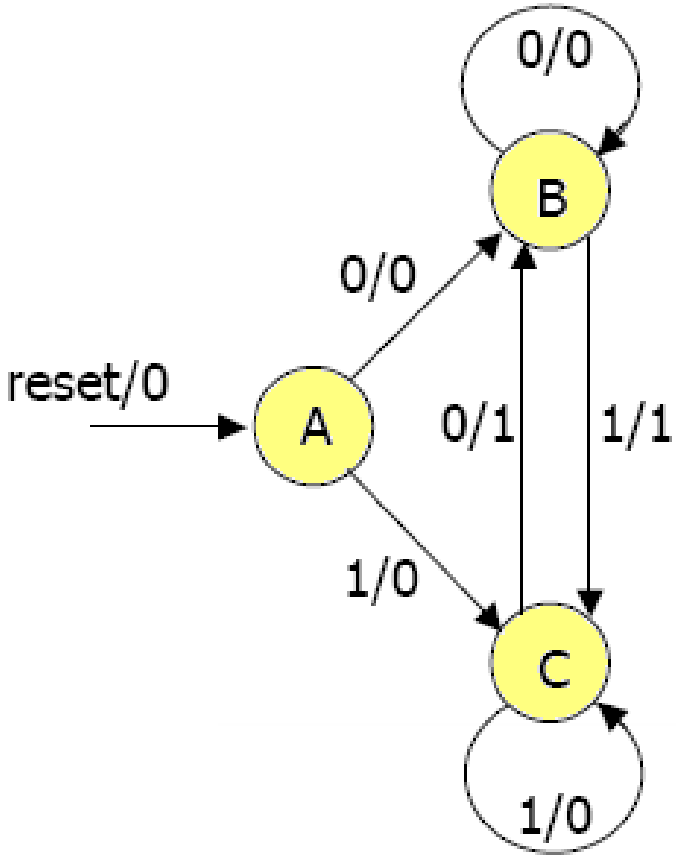
Function : sequence detector for 01 or 10

Stete FSM table?



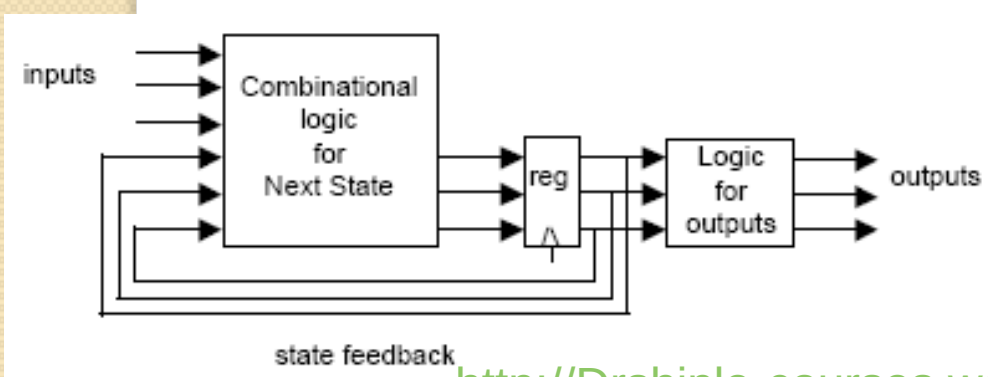
Function : sequence detector for 01 or 10

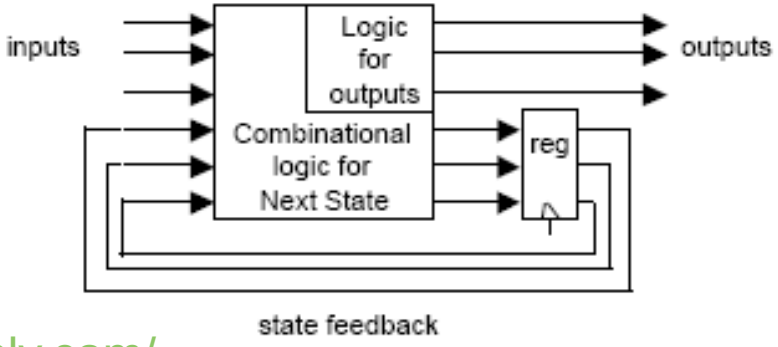
reset	input	current state	next state	output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	C	1
0	0	C	B	1
0	1	C	C	0



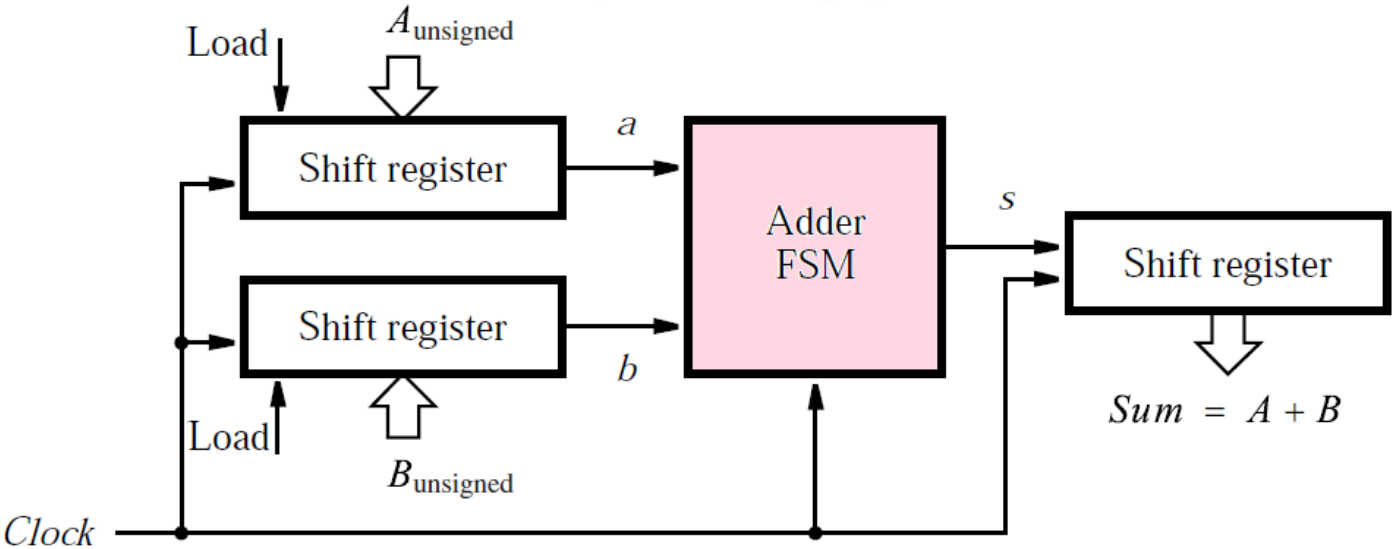
Comparison of Mealy and Moore Machines

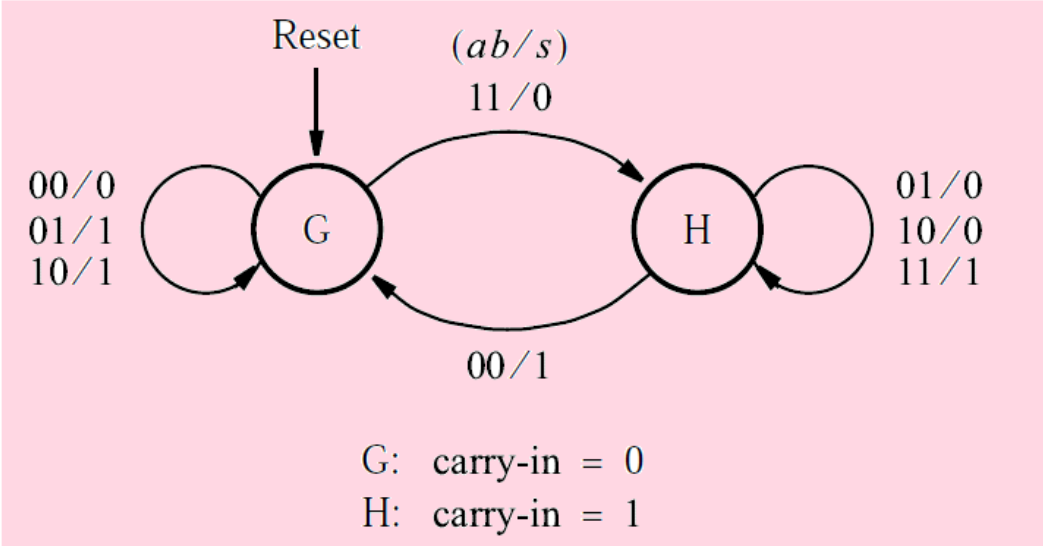
- Moore Machines are safer to use
  - Outputs change at clock edge (always one cycle later)
  - In Mealy machines, input change can cause output change as soon as logic is done – a big problem when two machines are interconnected – asynchronous feedback
- Mealy Machines react faster to inputs
  - React in same cycle – don't need to wait for clock
  - In Moore machines, more logic may be necessary to decode state into outputs – more gate delays after





Serial Adder (Mealy)



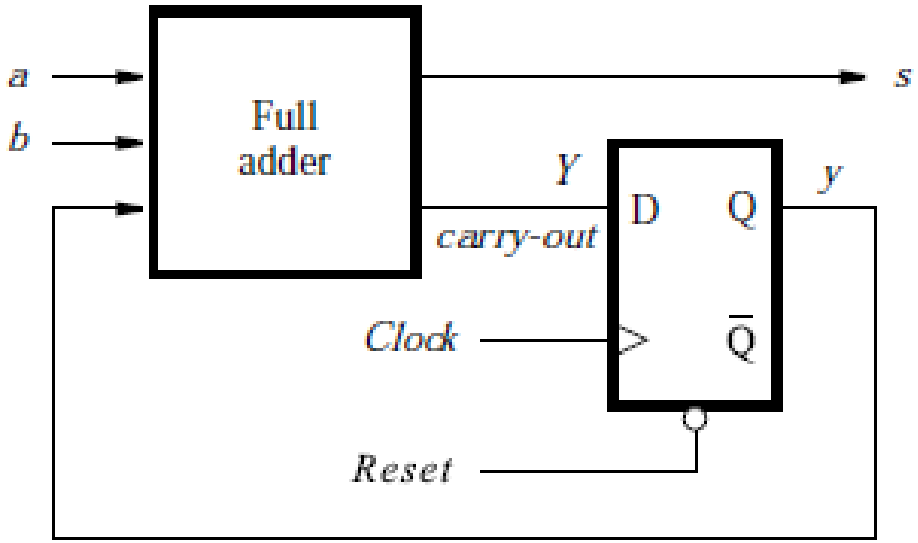


Truth Table

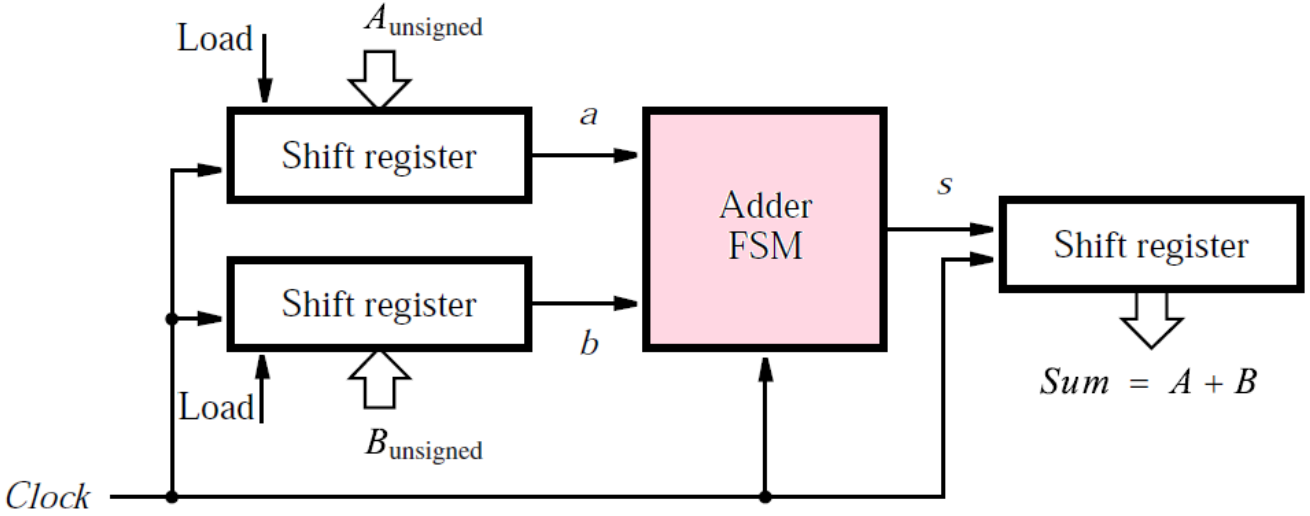
Present state	Next state				Output s			
	$ab = 00$	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

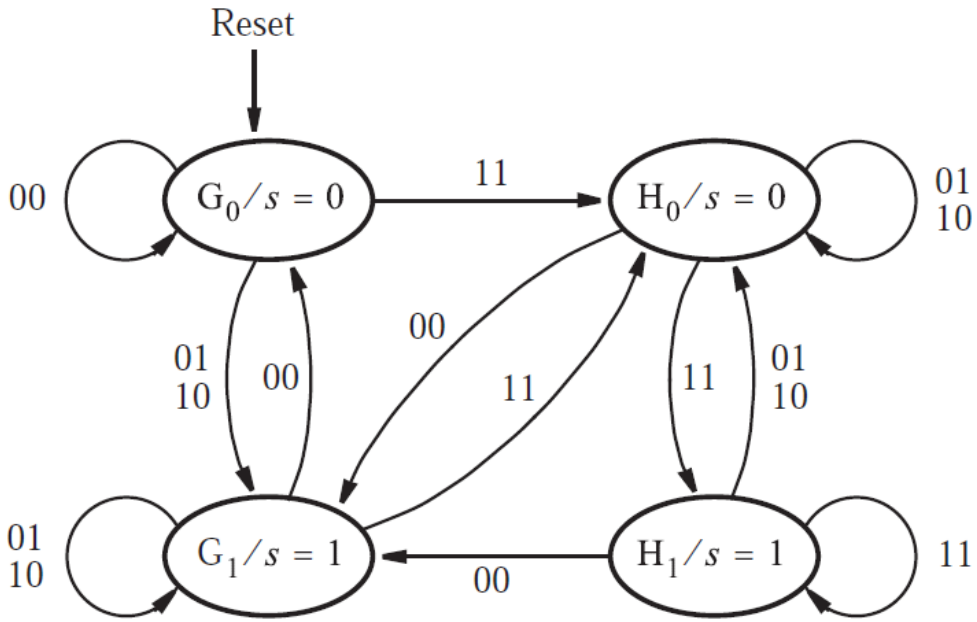
Present state	Next state				Output			
	$ab = 00$	01	10	11	00	01	10	11
y	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Realization



Serial Adder (Moore)



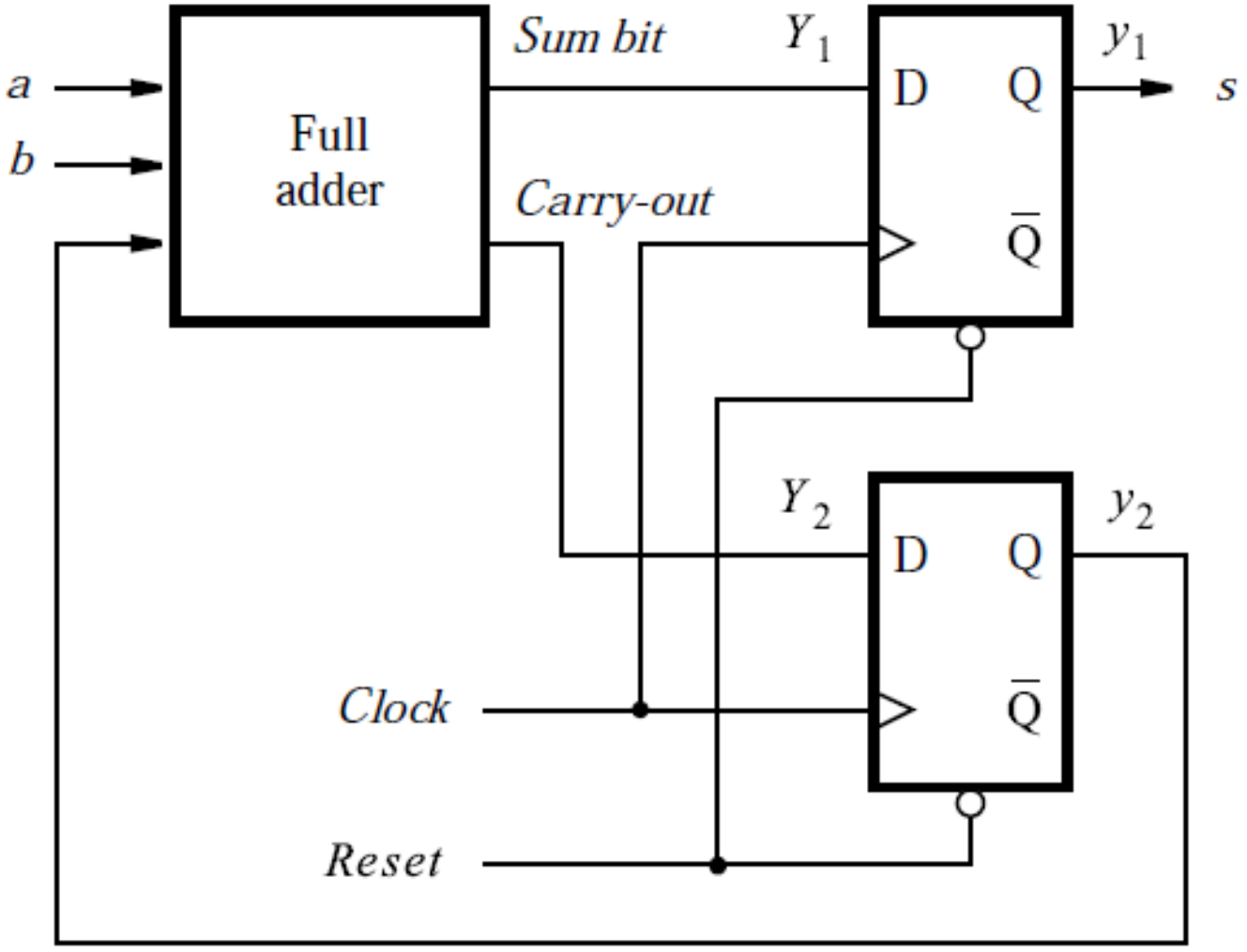


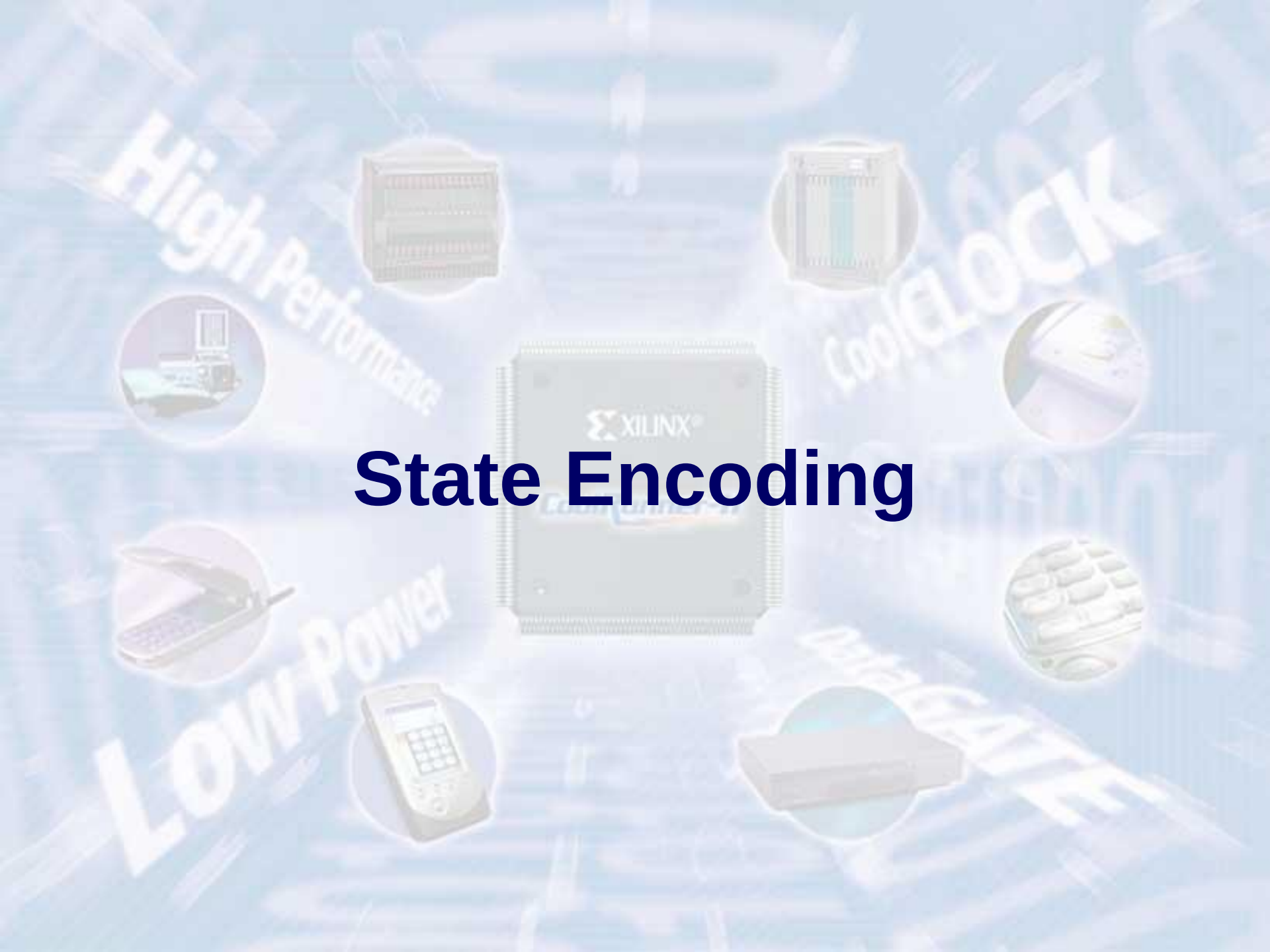
Truth Table

Present state	Next state				Output
	$ab = 00$	01	10	11	s
G_0	G_0	G_1	G_1	H_0	0
G_1	G_0	G_1	G_1	H_0	1
H_0	G_1	H_0	H_0	H_1	0
H_1	G_1	H_0	H_0	H_1	1

Present state	Next state				Output
	$ab = 00$	01	10	11	
$y_2 y_1$	$Y_2 Y_1$				s
0 0	0 0	0 1	0 1	1 0	0
0 1	0 0	0 1	0 1	1 0	1
1 0	0 1	1 0	1 0	1 1	0
1 1	0 1	1 0	1 0	1 1	1

Realization





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

State	Binary Code	Gray Code	One-Hot Code
S0	000	000	10000000
S1	001	001	01000000
S2	010	011	00100000
S3	011	010	00010000
S4	100	110	00001000
S5	101	111	00000100
S6	110	101	00000010
S7	111	100	00000001
# states=n FF	2^n	2^n	n

Types of State Encodings (1)

- One-Hot – Only One Bit Is Active
  - Number of used flip-flops as big as number of states (n)
  - Simple and fast transition functions
  - Preferable coding technique in FPGAs

Types of State Encodings (1)

- Binary (Sequential) – States Encoded as Consecutive Binary Numbers
  - Small number of used flip-flops ($\log_2 n$)
  - Potentially complex transition functions leading to slow implementations.
  - High Power Consumption .
  - Preferable coding technique in CPLD
  - Could generate Hazard circuits

Types of State Encodings (1)

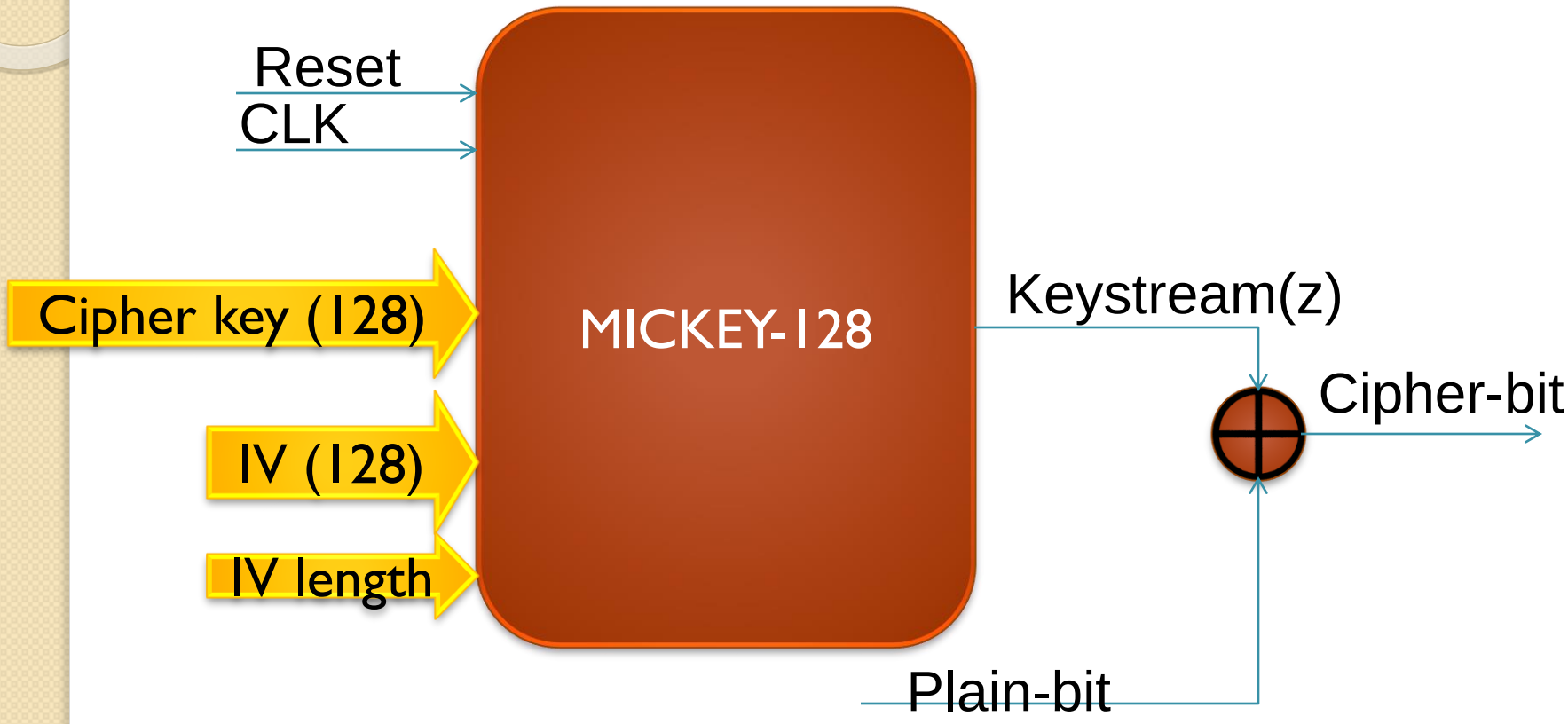
- Grey Code
  - Small number of used flip-flops ($\log_2 n$)
  - Potentially complex transition functions leading to slow implementations (more than Binary).
  - Low Power Consumption .
  - Preferable coding technique in CPLD
  - Could generate Hazard circuits

The Stream Cipher

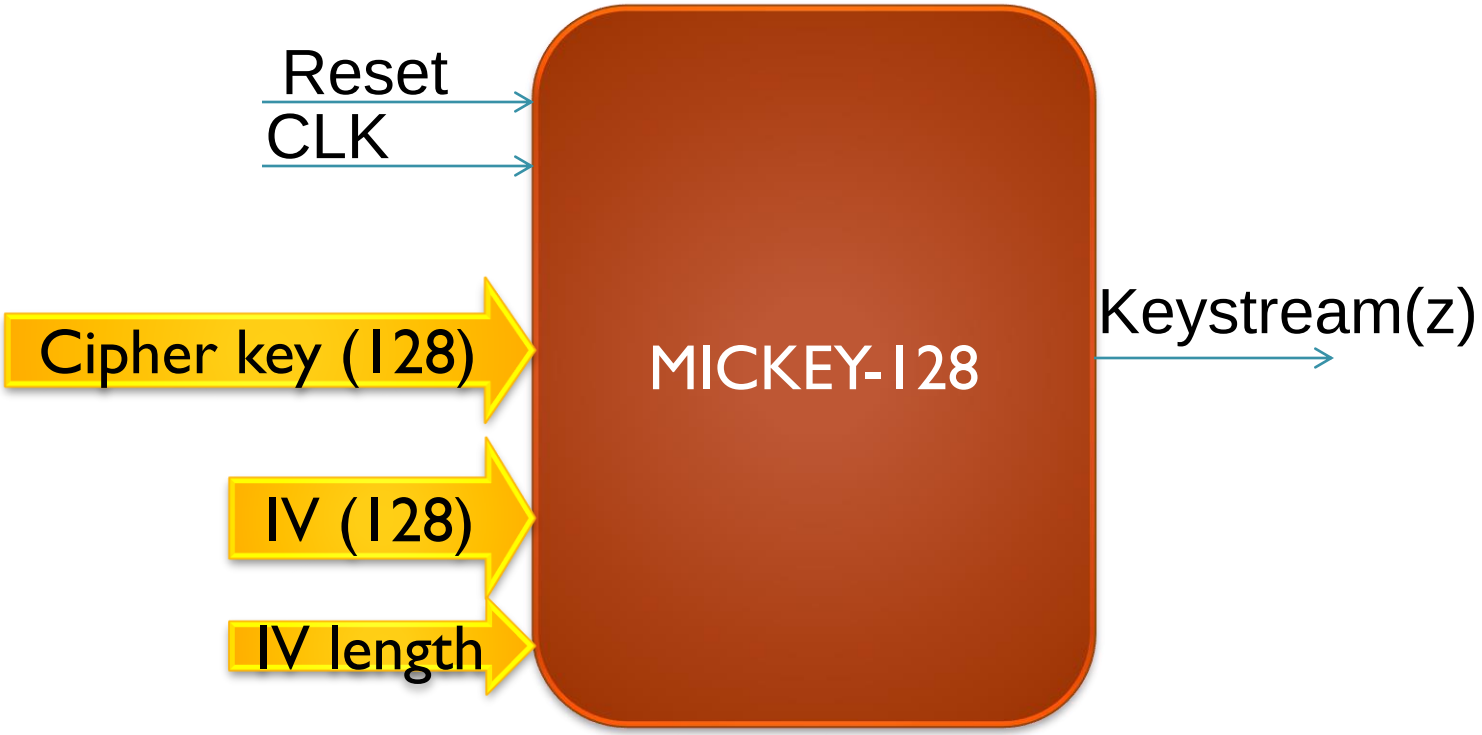
MICKEY-128

http://Drshiple-courses.weebly.com/

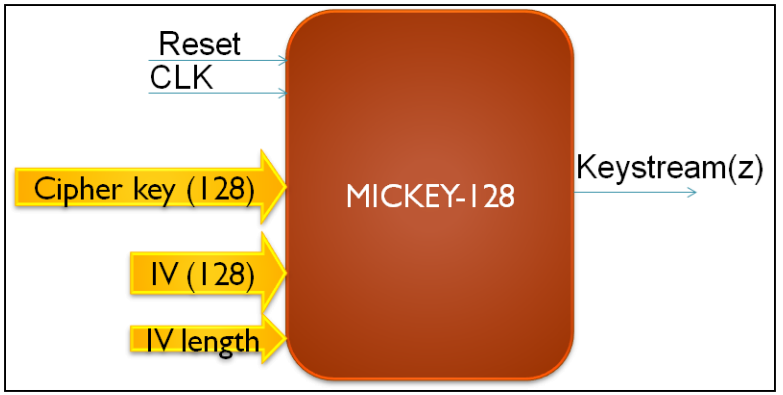
Cipher System

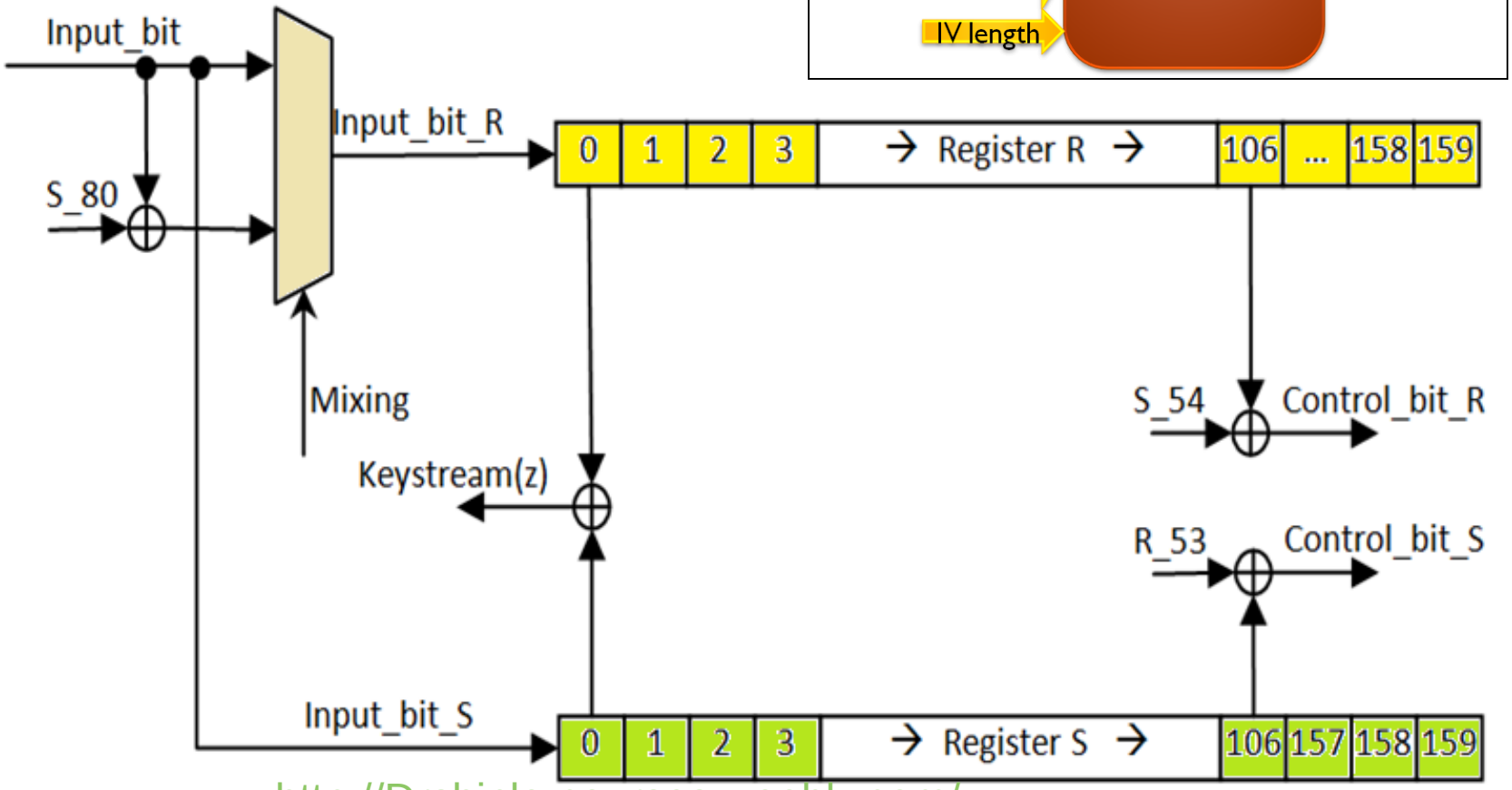


Mickey 128


Reset
CLK
Cipher key (128)
IV (128)
IV length
MICKEY-128
Keystream(z)

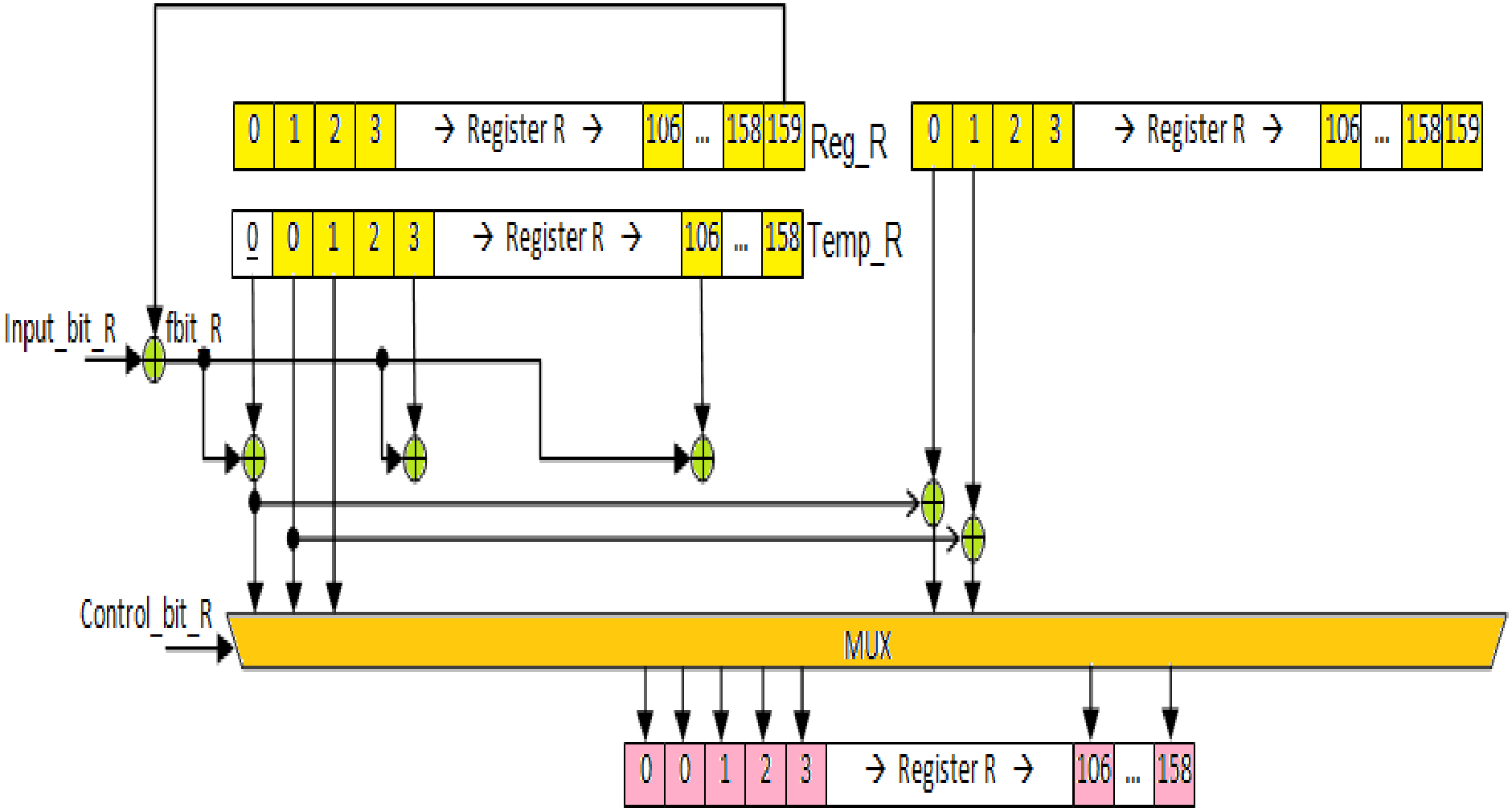
Clocking and overall generator





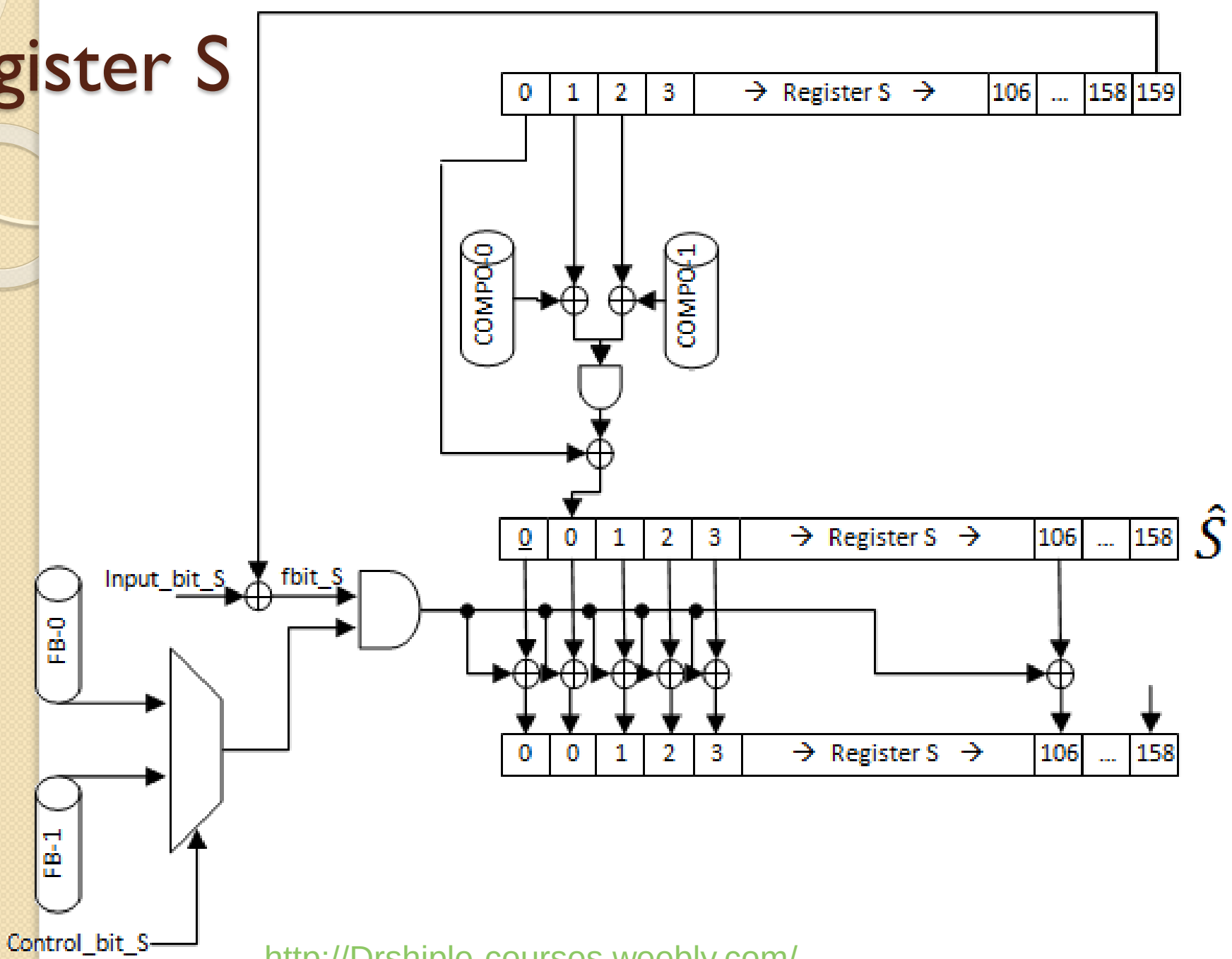
Register R Entity

Register R


0 1 2 3 → Register R → 106 ... 158 159
Reg_R
0 1 2 3 → Register R → 106 ... 158 159
0 0 1 2 3 → Register R → 106 ... 158
Temp_R
Input_bit_R
fbit_R
Control_bit_R
MUX
0 0 1 2 3 → Register R → 106 ... 158

Register S Entity

Register S



End of Presentation

http://Drshiple-courses.weebly.com/