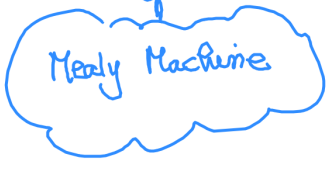
Sequential logic



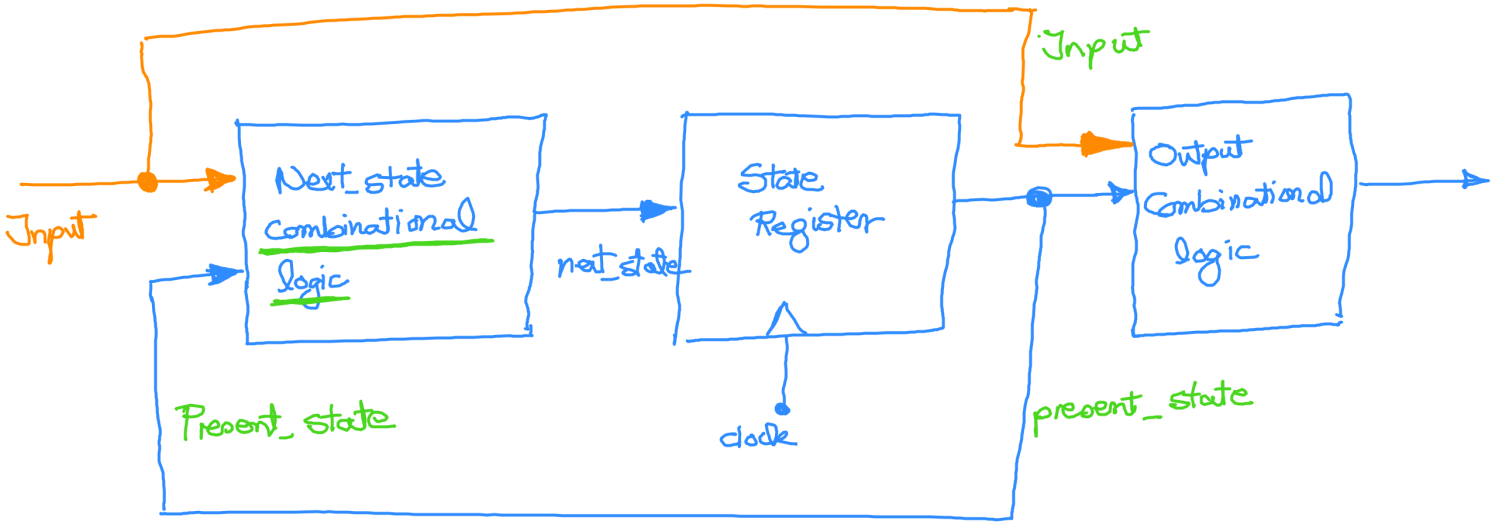
- Block diagram
- State Transition graph
  STG
- Verilog design
- Design examples

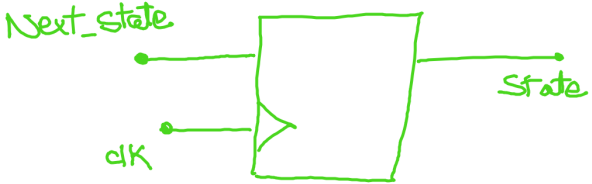
- Block diagram
- State Transition graph
  STG
- Verilog design
- Design examples

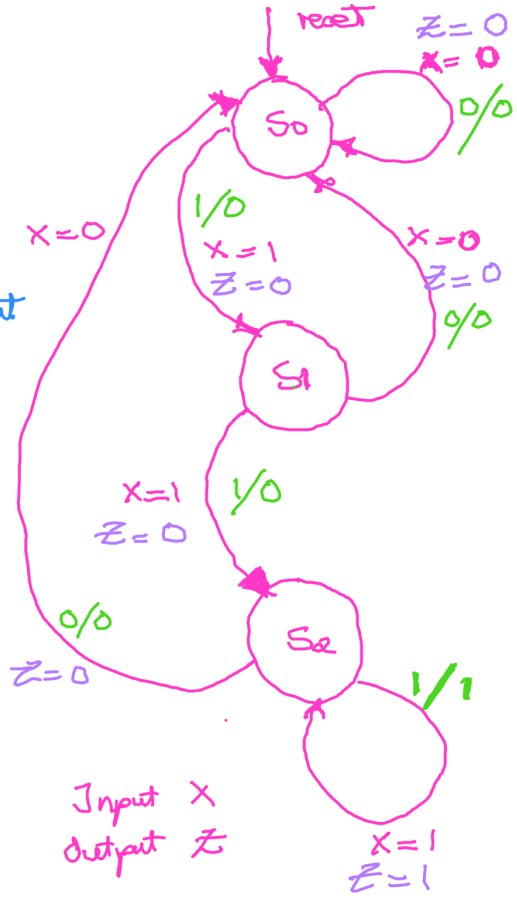
Mealy Machine

Block Diagrams

State graph



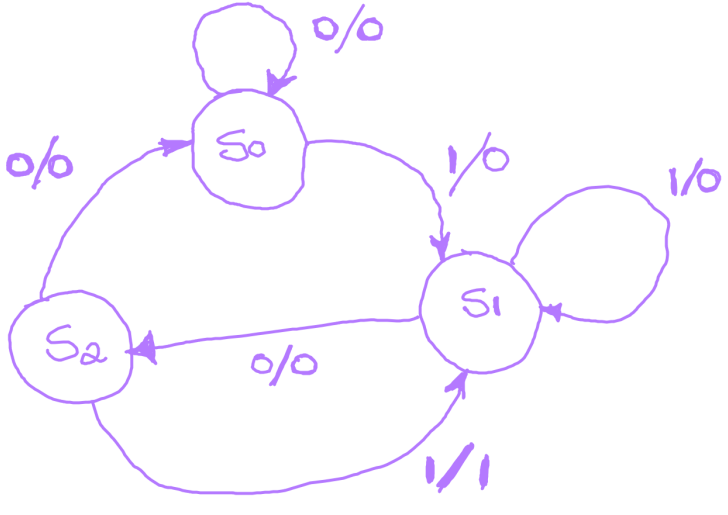




time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0
Z	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0

Detecting the sequence 101. The output $Z=1$ coincid with the last 1. The machine does not reset after receiving the code.

State Transition graph



S_0 none of the bits of the required sequence has been received.

S_1 The first bit of the required sequence has been received.

S_2 The first two bits of the required sequence have been received

State Transition table

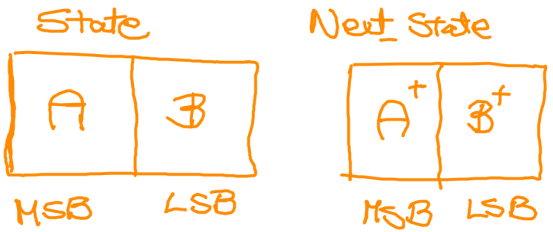
Present State	Next State X=0	Next State X=1	Present Output X=0	Present Output X=1
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

Present State	Next State x=0	Next State x=1	Present Output x=0	Present Output x=1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

$S_0 = 00$

$S_1 = 01$

$S_2 = 10$



K-map for A^+

X \ AB	00	01	11	10
0	0	1	X	0
1	0	0	X	0

$A^+ = \bar{X}B$

K-map for B^+

X \ AB	00	01	11	10
0	0	0	X	0
1	1	1	X	1

$B^+ = X$

K-map for Z

X \ AB	00	01	11	10
0	0	0	X	0
1	0	0	X	1

$Z = XA$

```
module Mealy_Sequence_detector ( input clk, X ;
                                 output Z ) ;

reg  [1:0] state ;
wire [1:0] next_state ;

always @ ( posedge clk )          // Flip-flop description
    state <= next_state ; sequential logic

// Combinational for the next_state :

assign next_state [0] = X ;
assign next_state [1] = (~X) & state [0] ;

// Combinational logic for the output Z :

assign Z = X & state [1] ;

endmodule
```

Combinational logic