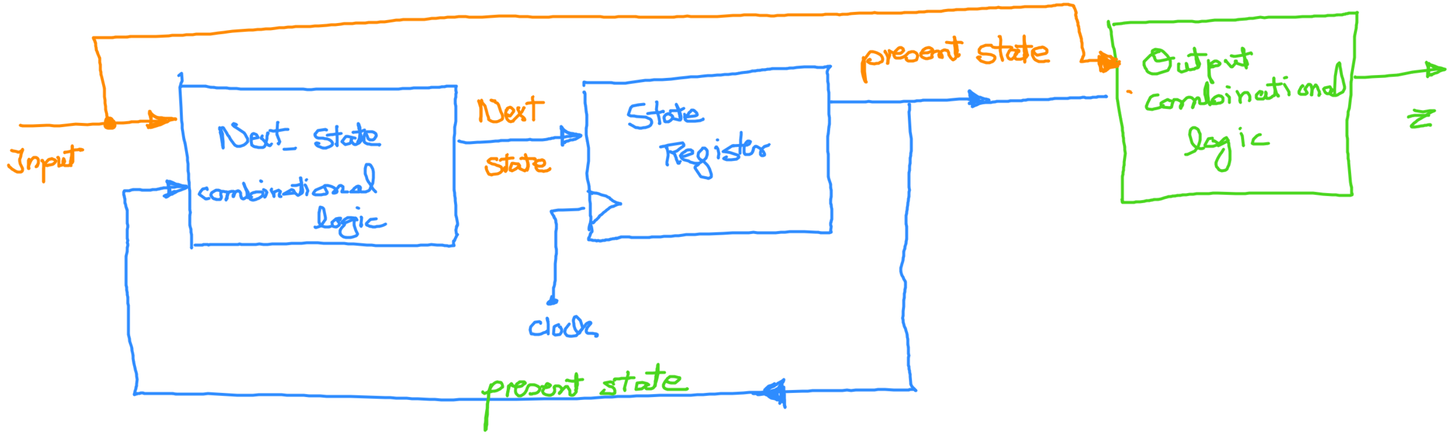
Finite State Machines

Mealy machine



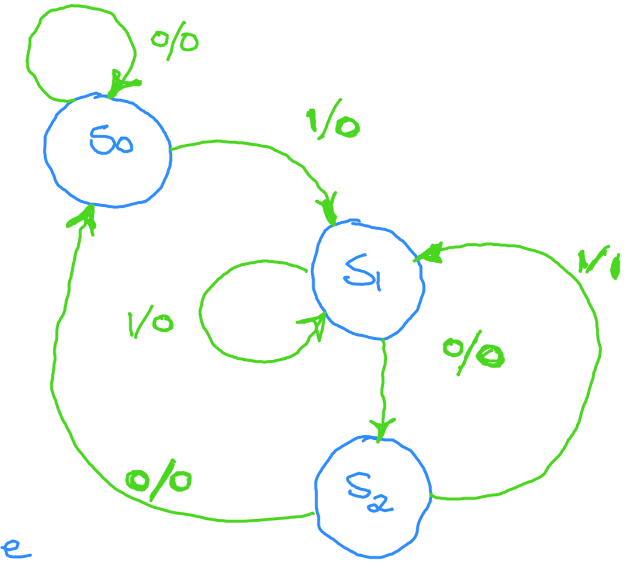
To detect
a code (101)

Z = 1

S_0 Reset state
none of the bits
of the code were received

S_1 The first bit of the
code was received (1)

S_2 The first two bits of the code
were received (10)

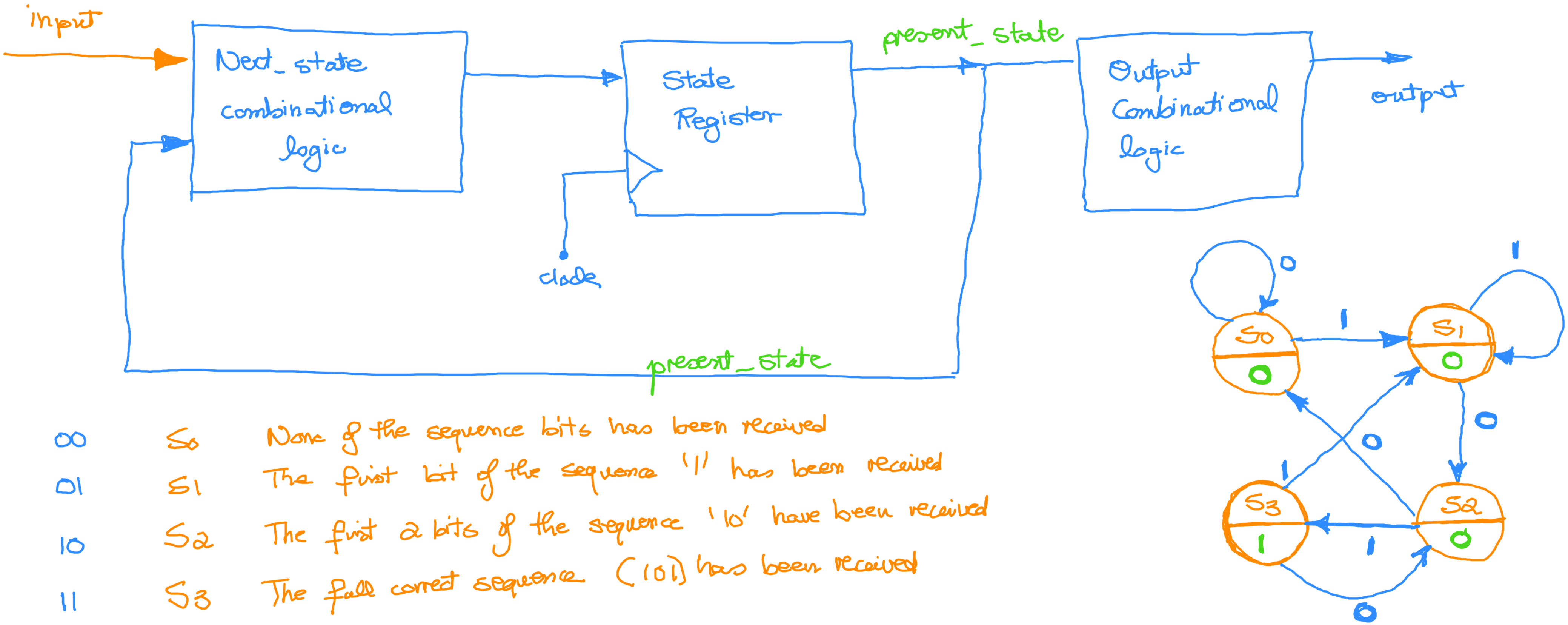


```
module   Mealy_Sequence_detector ( input clk , x ;
                                   output z         );

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

   always @ (posedge clk)
       state <= next_ state ;      // sequential logic

   //combinational logic :

   assign  next_ state [0]  = x ;
   assign  next_ state [1]  = (~x) & state [0] ;

   // combinational logic for the output z :

   assign  z = x & state [1] ;

endmodule
```

```
module  Mealy_case  ( input clk , x ;
                      output reg z    );

   reg    [1:0]   state ;
   reg    [1:0]   next_state ;   // to use it in the always
                                    block
   parameter  S0 = 2'b00 , S1 = 2'b01 ,
              S2 = 2'b10 ;
   always @ (posedge clk)
       state <= next_state ;

   always @ (*)
     case ( state )
   S0   : begin   z = 1'b0 ;
                if  x = 1'b1
                   next_state  =  S1 ;
                else next_ state  =  S0 ;  end

   S1   : begin   z = 1'b0 ;
                 if  x = 1'b1
                    next_state  =  S1 ;
                 else next-state = S2; end

   S2   :   if x = 1'b1   begin
                 z = 1'b1 ; next_state  = S1 ;
            else begin  z = 1'b0 ; next_state = S0 ;
   default : begin  z = 1'b0 ; next_state  = S0 ; end

          endcase

   endmodule
```

Moore Machine

101 Sequence detector



00	S0	None of the sequence bits has been received
01	S1	The first bit of the sequence '1' has been received
10	S2	The first 2 bits of the sequence '10' have been received
11	S3	The full correct sequence (101) has been received

State Table

	Present state		Next_state x = 0	Next_state x = 1	Output z
✓	00	S0	S0 00	S1 01	0
✓	01	S1	S2 10	S1 01	0
	10	S2	S0 00	S3 11	0
✓	11	S3	S2 10	S1 01	1

z = state [0] & state [1] ;

next_state [0] = X ;

$$\text{next\_state [1]} = \underbrace{((\sim x) \,\&\, \text{state [0]}) \mid (x \,\&\, \text{state [1]} \,\&\, (\sim \text{state [0]}))}_{} \;;$$

State / X	S0 00	S1 01	S3 11	S2 10
0	0	0	0	0
1	1	1	1	1

next_state [0] = x ;

State / X	S0 00	S1 01	S3 11	S2 10
0	0	1	1	0
1	0	0	0	1

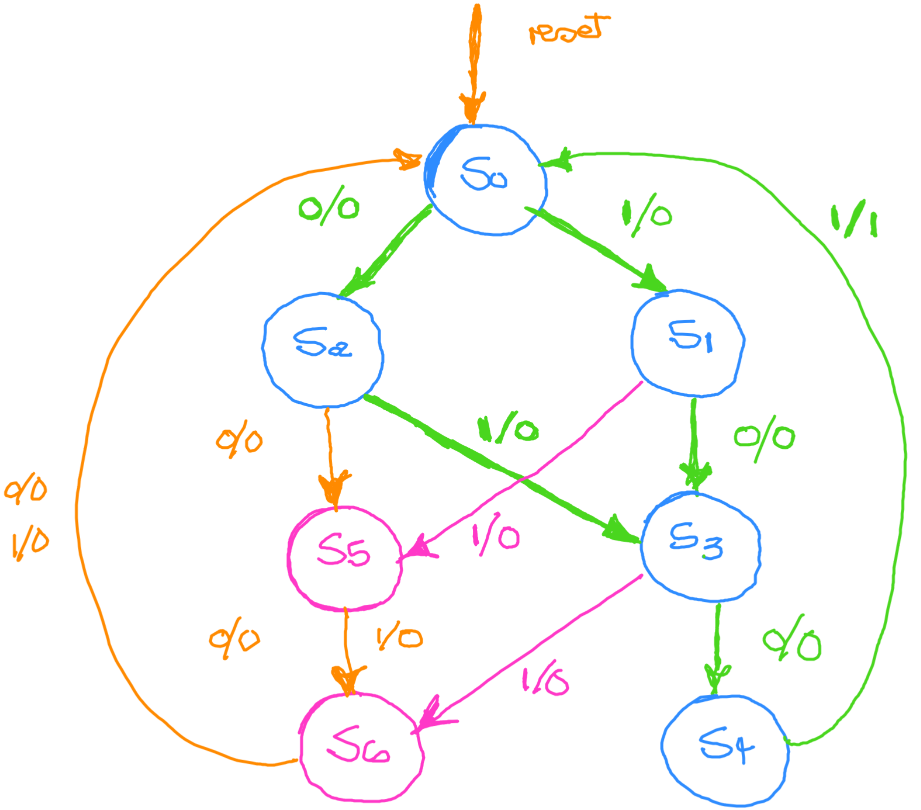
next_state [1] = (~x) & state [0]
| (x & state[1]
& (~state[0])

```
module   Moore_machine  ( input   clk , x
                          output  z        ) ;

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

  always @  (posedge  clk )
        State  <=  next_state    ;

assign   z  =  next_state [0]  &  next_state [1]  ;

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


endmodule
```

Example :

A sequential circuit has one input (x) and one output (z). The circuit examines groups four consecutive inputs and produces an output z = 1 if the input sequence 0101 or 1001 occurs. The circuit resets after every four inputs. Find the Mealy state graph.

X	=	0101	0010	1001	0100
Z	=	0001	0000	0001	0000



S0 reset state – none of the bits have been received

S1 The first bit of 1001 has been received

S2 The " " " 0101 has " "

S3 The first 2 bits of either 1001 or 0101 have been received

S4 The " 3 " " 1001 or 0101 have been received

S5 Two bits of a wrong sequence have been received.

S6 3 bits of a wrong sequence have been received