

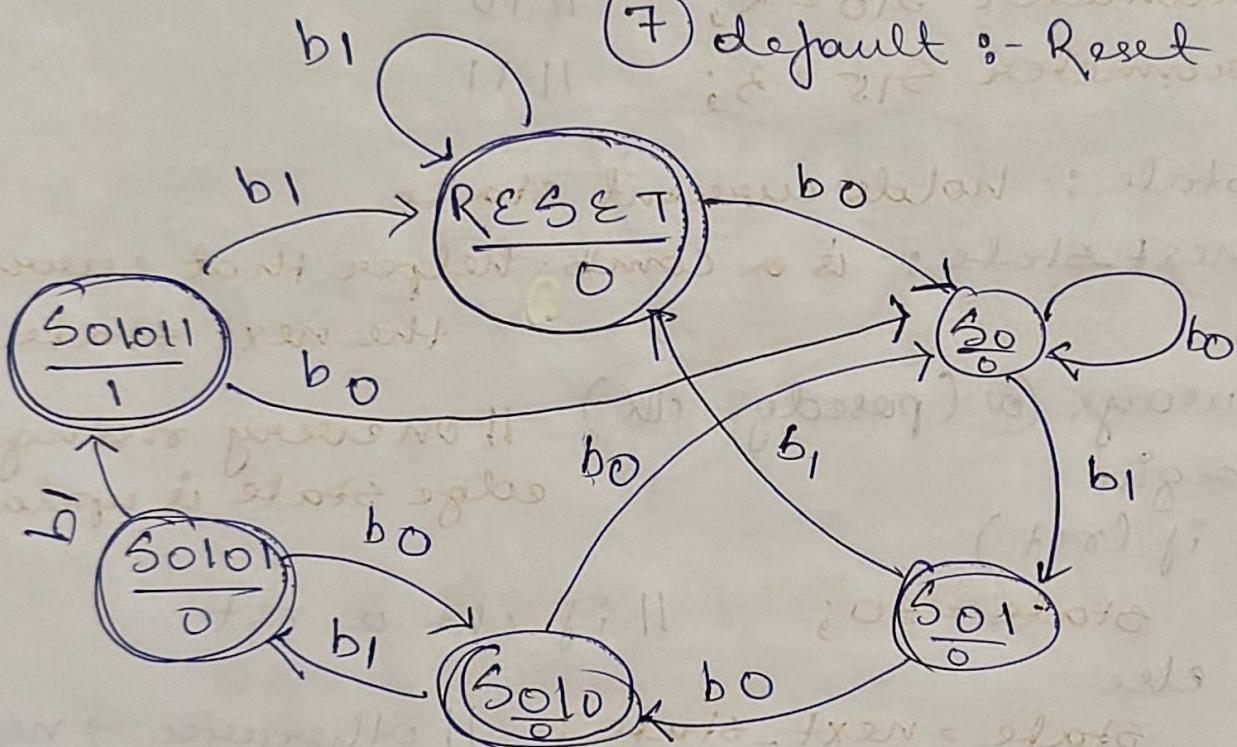
Digital lock :- (01011 → sequence detector)

- Let's say 01011 is the combination to unlock a digital lock.
- Using finite state machine
- State Assign :-

$b_0/b_1 \rightarrow$ user button

① Reset (000)	② S0 (001)
③ S01 (010)	④ S010 (011)
⑤ S0101 (100)	⑥ S01011 (101)

⑦ default :- Reset



- Inputs and signals

① clk

② rst - synch | Asynch ctrl

③ b_0, b_1 - 2 mutually exclusive buttons that encode incoming bit

$b_0 = \text{logical 0}$, $b_1 = \text{logical 1}$

④ unlock - 1 when sequence 01011, otherwise 0

- Next state logic :- on each clk the FSM samples b_0/b_1 .
- ⇒ Allows overlapping patterns to be recognised
- At power-up or after reset, the machine is in S_RESET and unlock = 0
- Verilog code

```
module lock (input clk, rst, b0,b1, output unlock);
parameter S_RESET = 0;
parameter S_0 = 1;
parameter S_01 = 2;
" S_010 = 3;
" S_0101 = 4;
" S_01011 = 5;
reg [2:0] state, next-state;
```

- ⇒ State transitions (using Ternary operator)

always @ (*) begin

if (rst)

next-state = S_RESET

else

case (state)

S_RESET :

next-state = $b_0 ? S_0 : b_1 ? S_{RESET} : state;$

$S_0 :$

next-state = $b_0 ? S_0 : b_1 ? S_{RESET} : state;$

* Remember ($next_state = b_0 ? S_0$) ⇒ if $b_0 \Rightarrow S_0$

$(b_0 ? S_0 : b_1 ? S_{RESET}) \Rightarrow$ if $b_0 \Rightarrow S_0$

$(S_{RESET} : state) \Rightarrow$ stay in state else $b_1 \Rightarrow S_{RESET}$

\Rightarrow always @ (posedge clk)

state = next_state;

assign unlock = state == 5'b01011 ? 1'b0

{ // if (state == 5'b01011)

unlock = 1

else

unlock = 0

endmodule

• Testbench

module digital_lock_tb;

reg clk, b0, b1, rst;

wire unlock;

integer i;

reg [4:0] data;

always #10 clk = ~clk

lock until (clk, rst, b0, b1, unlock);

initial begin

clk = 0; rst = 1; // initialize

#50 reset = 0; // ckt is on

data = 5'b01011; // 20; 10

#130 \$finish

end

always @ (posedge clk) begin

b0 = data[5-i] ? 0:1;

b1 = data[5-i] ? 1:0;

end

endmodule

* index i ↑ with each clk

{ if selected data = 0 \Rightarrow b0 = 1, b1 = 0

{ if selected data = 1 \Rightarrow b1 = 0, b0 = 0