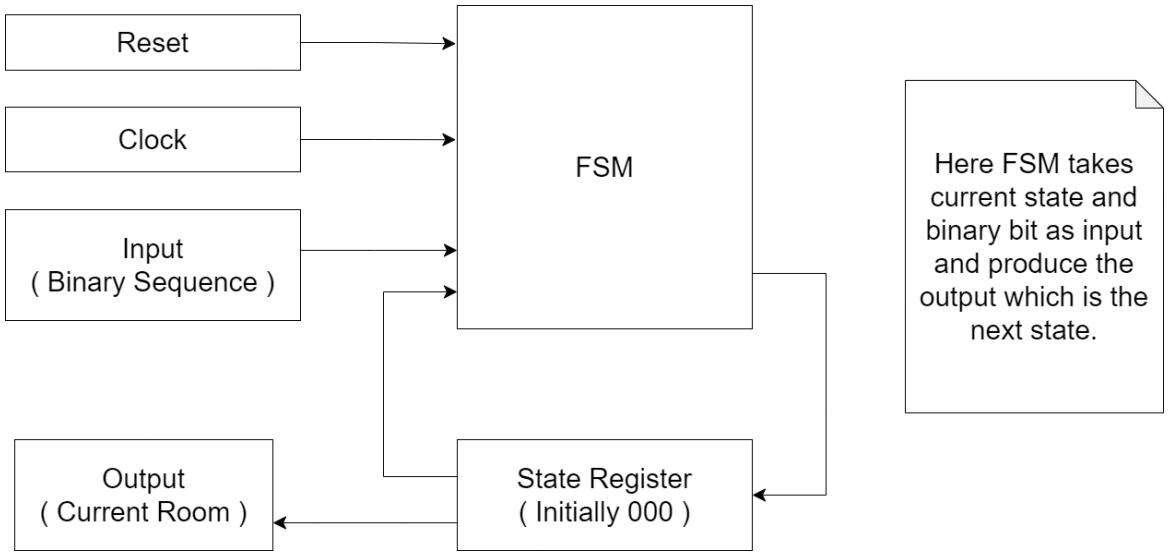
APPROACH

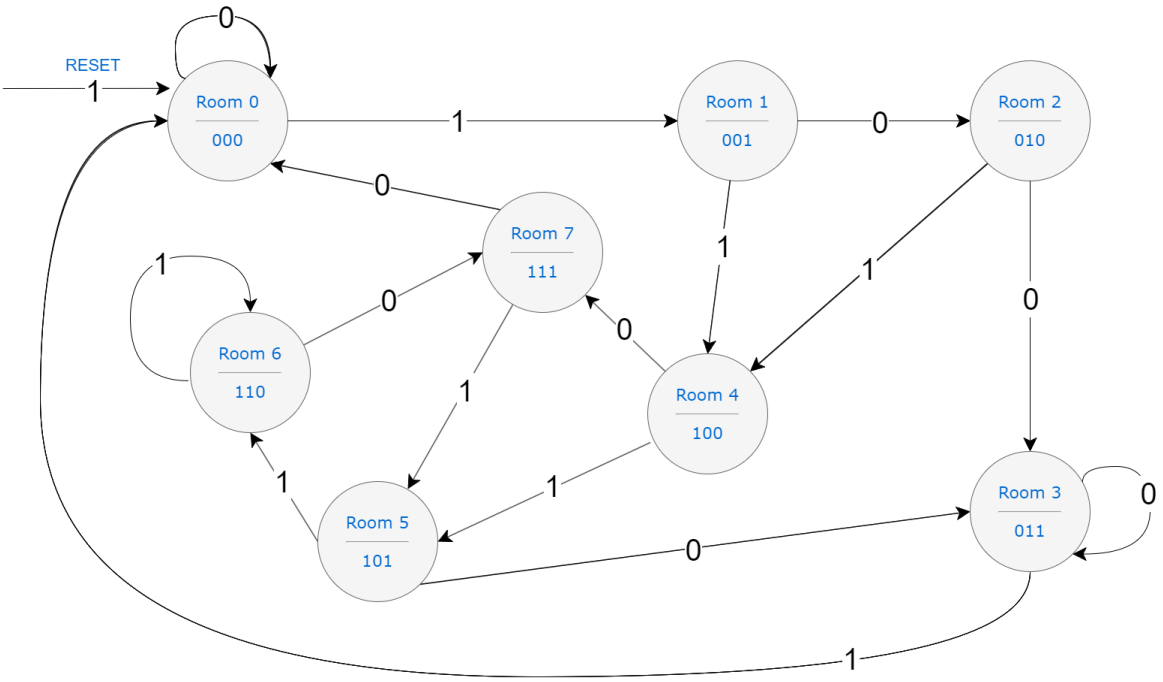
Topics:

1. Block Diagram.
2. State Diagram.
3. Moore or Mealy.
4. Identifying the Number of flip flops.
5. State Transition Table with Excitation.
6. Using Karnaugh-map.
7. Verilog Code.

1. Block Diagram



2. State Diagram



3. Moore or Mealy

For this project a Mealy Machine is chosen, because the outputs depend on both current state and input .

The output is also a state which can be determined by the current state and binary sequence which is considered as an input.

4. Number of Flip Flops

Since there are 8 states,

$$2^n = 8$$

$$n = 3$$

Number of flip flops required = 3

5. State Transition Table with Excitation

D Flip flop is used for Excitation.

S.No	Present State			Input	Next State			Excitation		
	A	B	C	X	A	B	C	DA	DB	DC
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1	0	0	1
3	0	0	1	0	0	1	0	0	1	0
4	0	0	1	1	1	0	0	1	0	0
5	0	1	0	0	0	1	1	0	1	1
6	0	1	0	1	1	0	0	1	0	0
7	0	1	1	0	0	1	1	0	1	1
8	0	1	1	1	0	0	0	0	0	0
9	1	0	0	0	1	1	1	1	1	1
10	1	0	0	1	1	0	1	1	0	1
11	1	0	1	0	0	1	1	0	1	1
12	1	0	1	1	1	1	0	1	1	0
13	1	1	0	0	1	1	1	1	1	1
14	1	1	0	1	1	1	0	1	1	0
15	1	1	1	0	0	0	0	0	0	0
16	1	1	1	1	1	0	1	1	0	1

6. Karnaugh - Map

For DA,

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

DA = B' CX + BC' X + AC' + AX

= X (B' C + BC') + A (C' + X)

DA = X (B ^ C) + A (C' + X)

For DB,

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

DB = A' CX' + AB' C + ABC' + AC' X' + BC' X'

= A' CX' + A (B' C + BC') + (A + B) C' X'

DB = A' CX' + A (B ^ C) + (A + B) C' X'

For DC,

	CX			
AB	00	01	11	10
00	0	1	0	0
01	1	0	0	1
11	1	0	1	0
10	1	1	0	1

DC = B' C' X + A' B X' + A B' X' + ABCX + BC' X'

= B' C' X + X' (A ^ B) + ABCX + BC' X'

DC = B' C' X + X' ((A ^ B) + BC')) + ABCX

7. Verilog Code

```
module landrover (
  input wire X, clk, reset,
  output reg [2:0] state
);

  reg A,B,C;
  reg [2:0] next_state;

  always @(posedge clk or posedge reset) begin

        if(reset == 1)   begin
              state <= 3'b000;
        end
        else   begin
              A = state[2];
              B = state[1];
              C = state[0];

              next_state[2] = ( A & ( !C | X ) ) | ( (B ^ C) & X );
              next_state[1] = ( !A & C & !X ) | ( A & (B ^ C) ) | ( (A |
B) & !C & !X );
              next_state[0] = ( !B & !C & X ) | ( !X & ( ( A ^ B ) | ( B &
!C ) ) ) | ( A & B & C & X );
              state <= next_state;
        end
  end

  initial begin
    state <= 3'b000; // Initialize state to 000
  end

endmodule
```