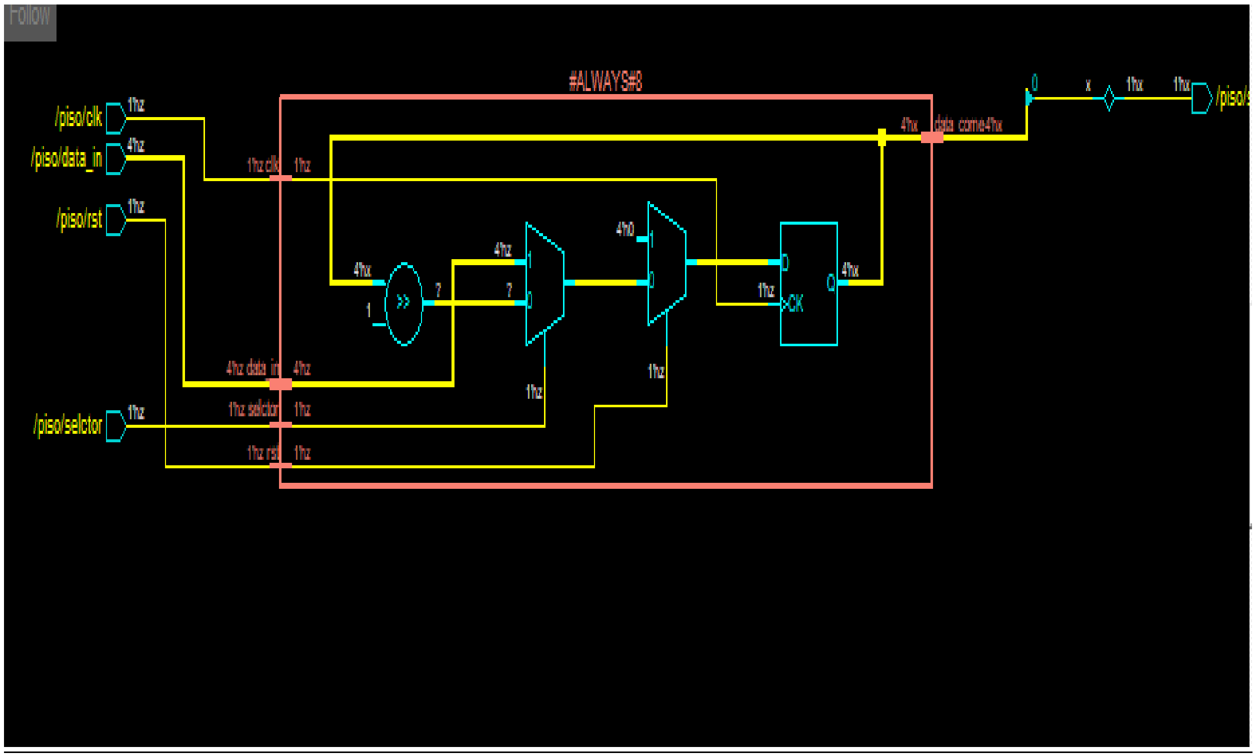
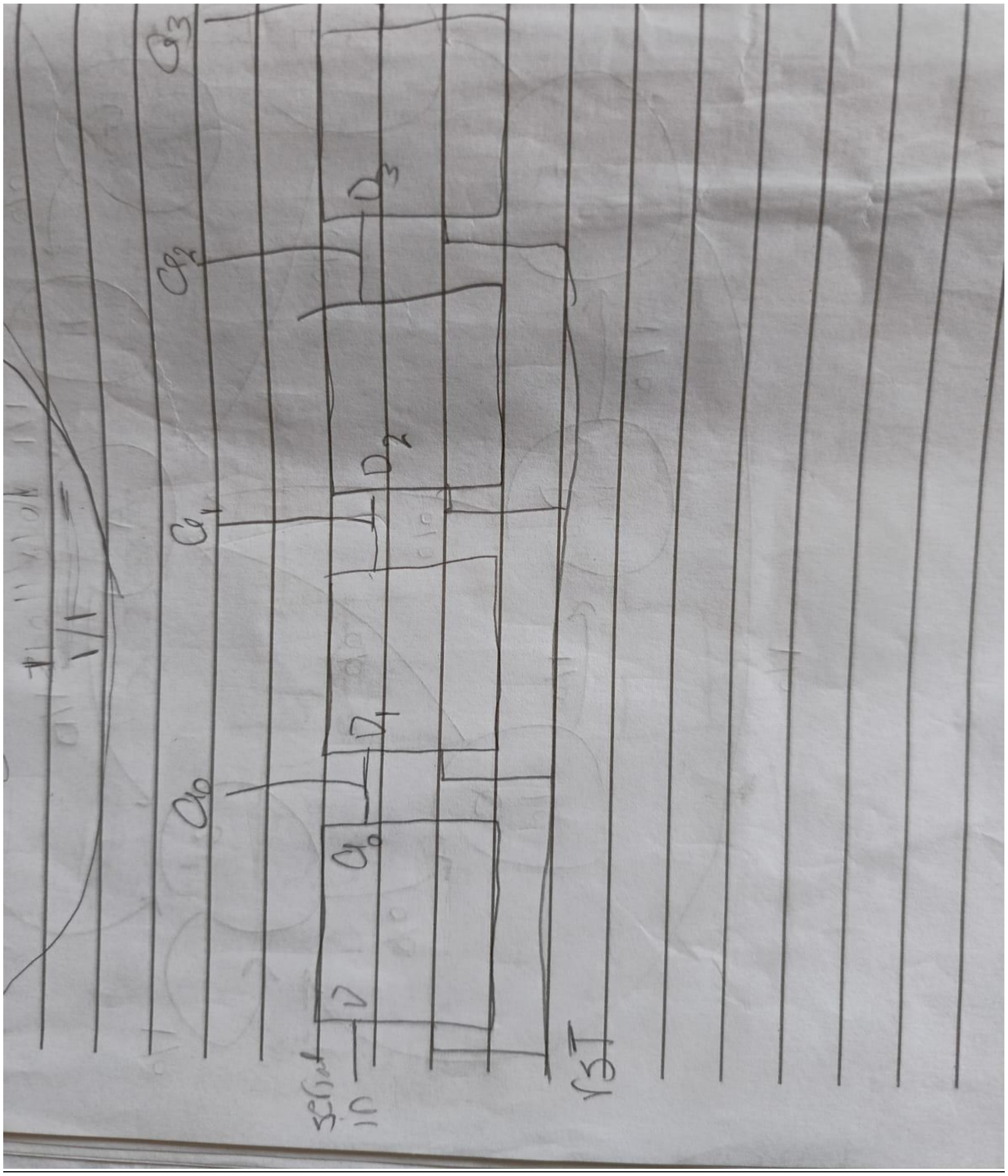
Continuo on lap2

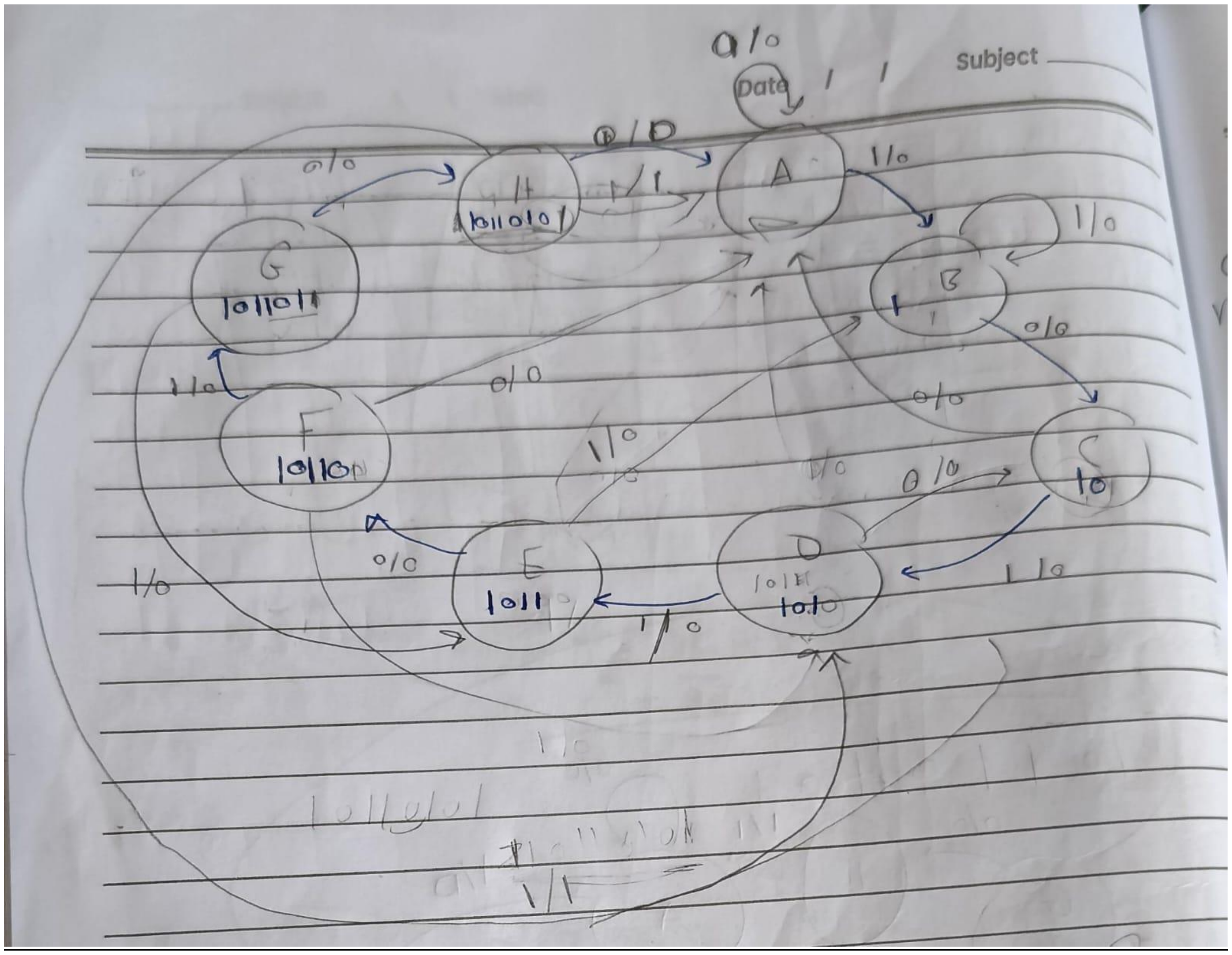
PISO Schmatic



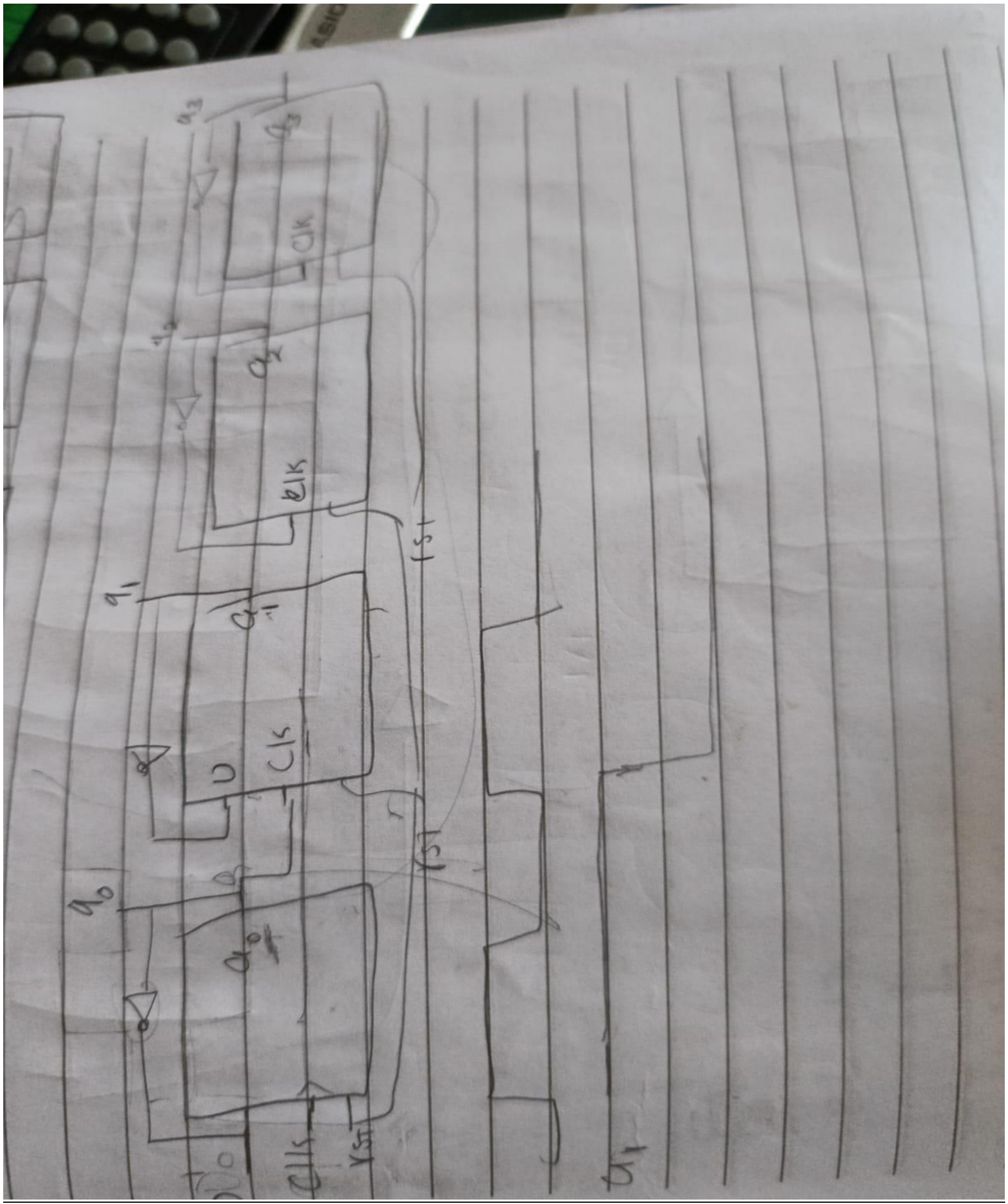
SIPO schem



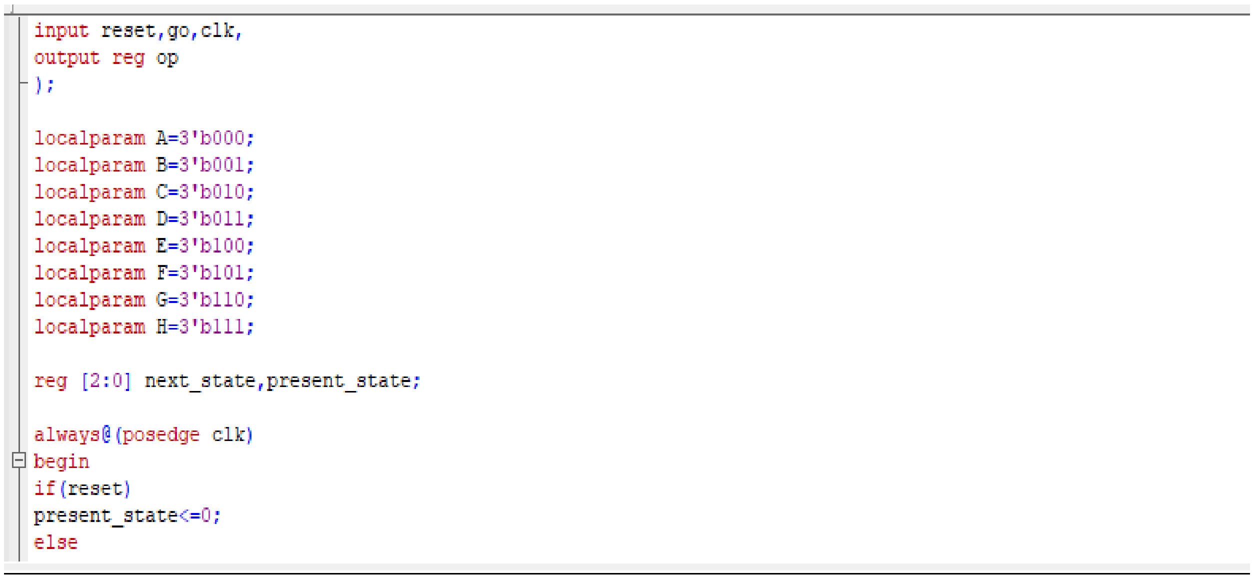
Fsm2


0/0
A
1/0
B
1/0
0/0
C
10
1/0
0/0
D
1010
0/0
1/0
E
1011
0/0
F
10110
1/0
G
101101
0/0
H
10110101
0/0
1/1
1/0
1/0
1/1
Subject
Date

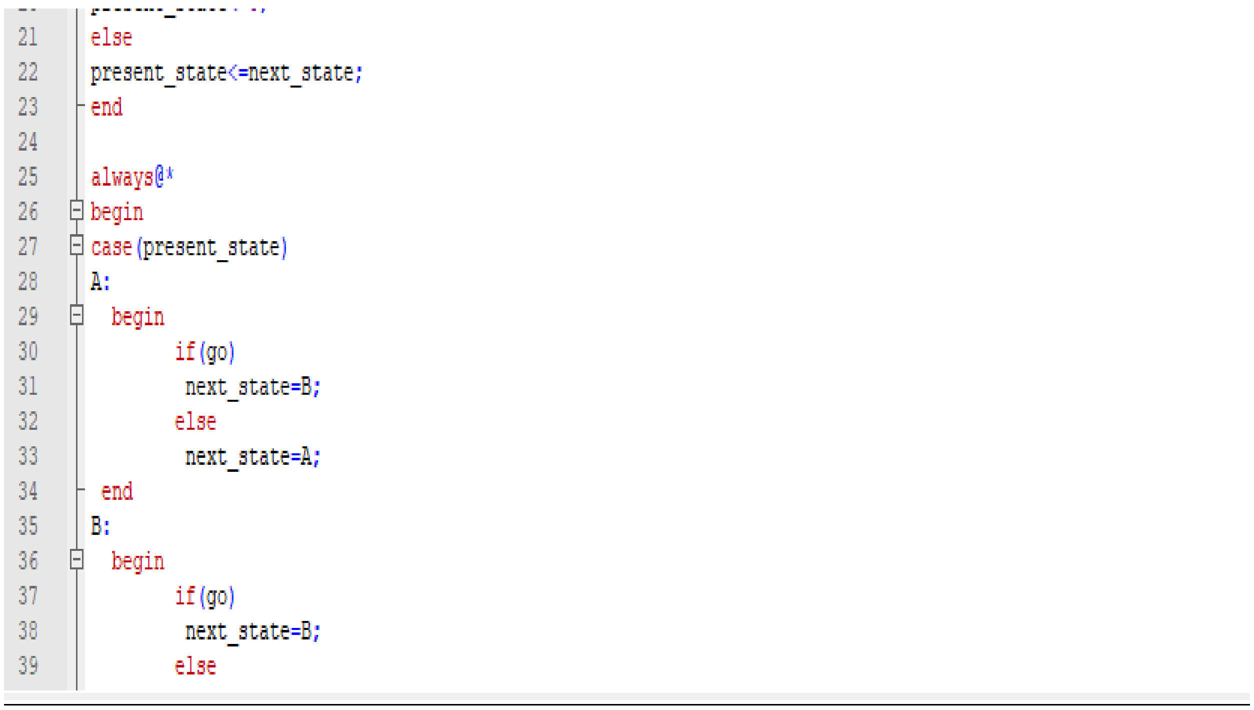
Divide by 2,4,8,16



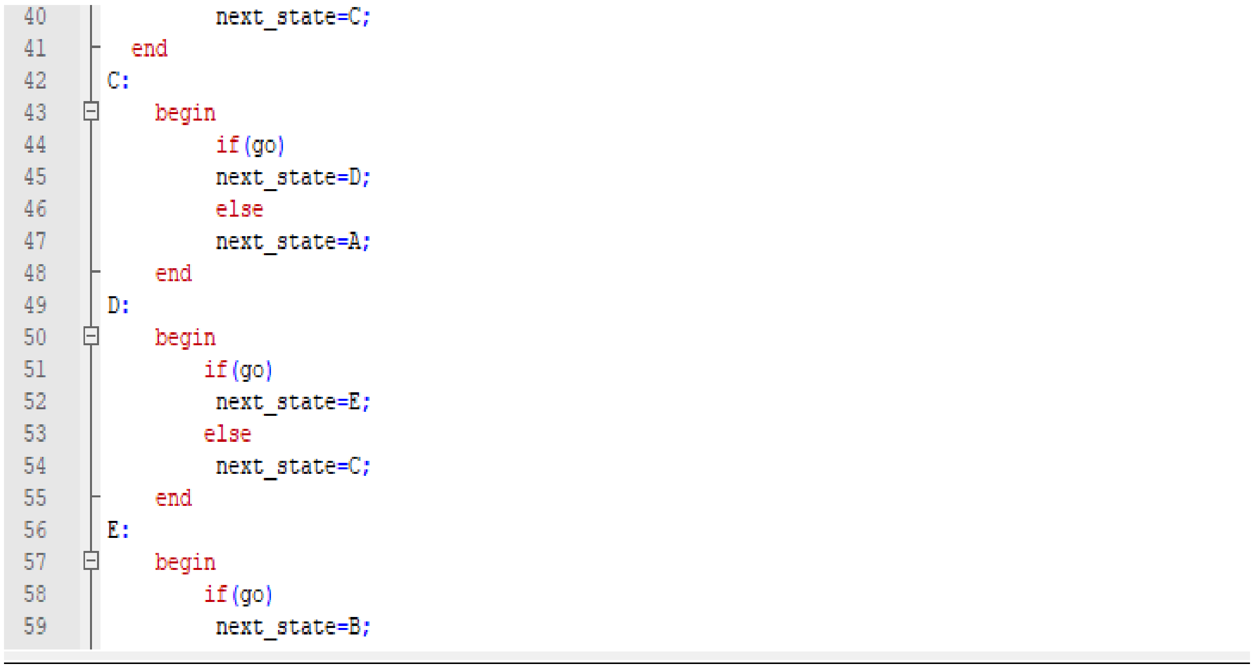
Fsm2_style



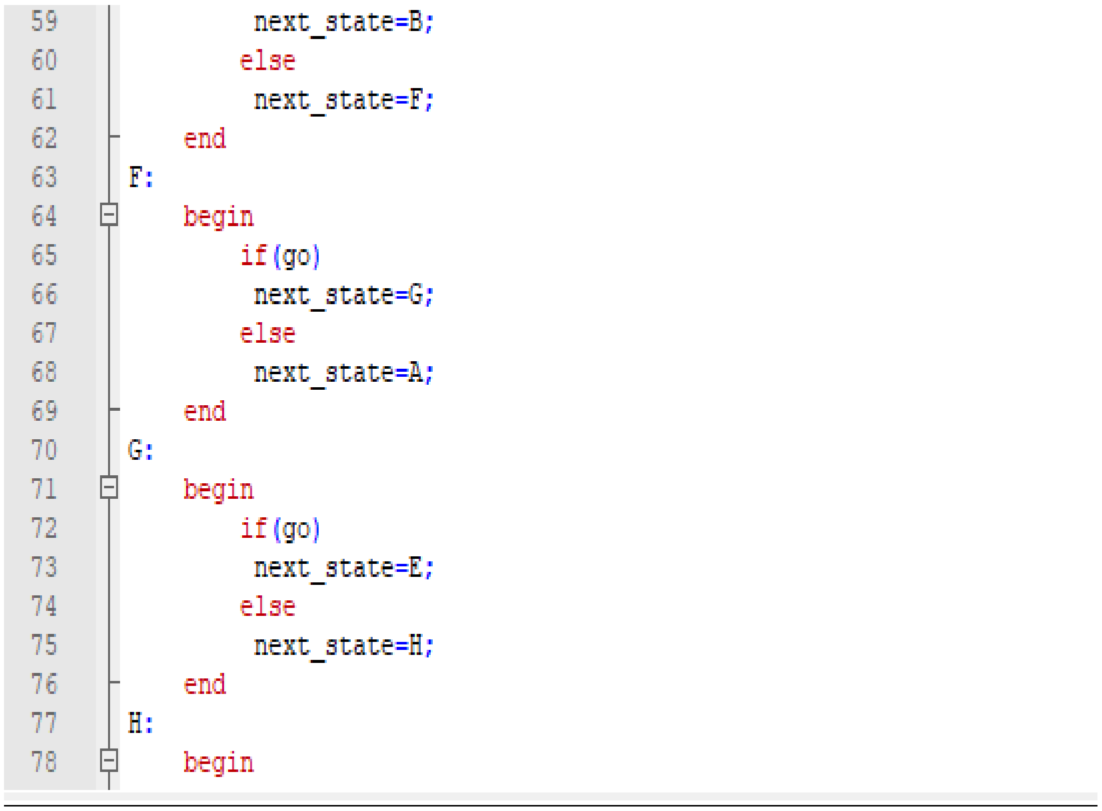
```
input reset,go,clk,
output reg op
);

localparam A=3'b000;
localparam B=3'b001;
localparam C=3'b010;
localparam D=3'b011;
localparam E=3'b100;
localparam F=3'b101;
localparam G=3'b110;
localparam H=3'b111;

reg [2:0] next_state,present_state;

always@(posedge clk)
begin
if(reset)
present_state<=0;
else
```



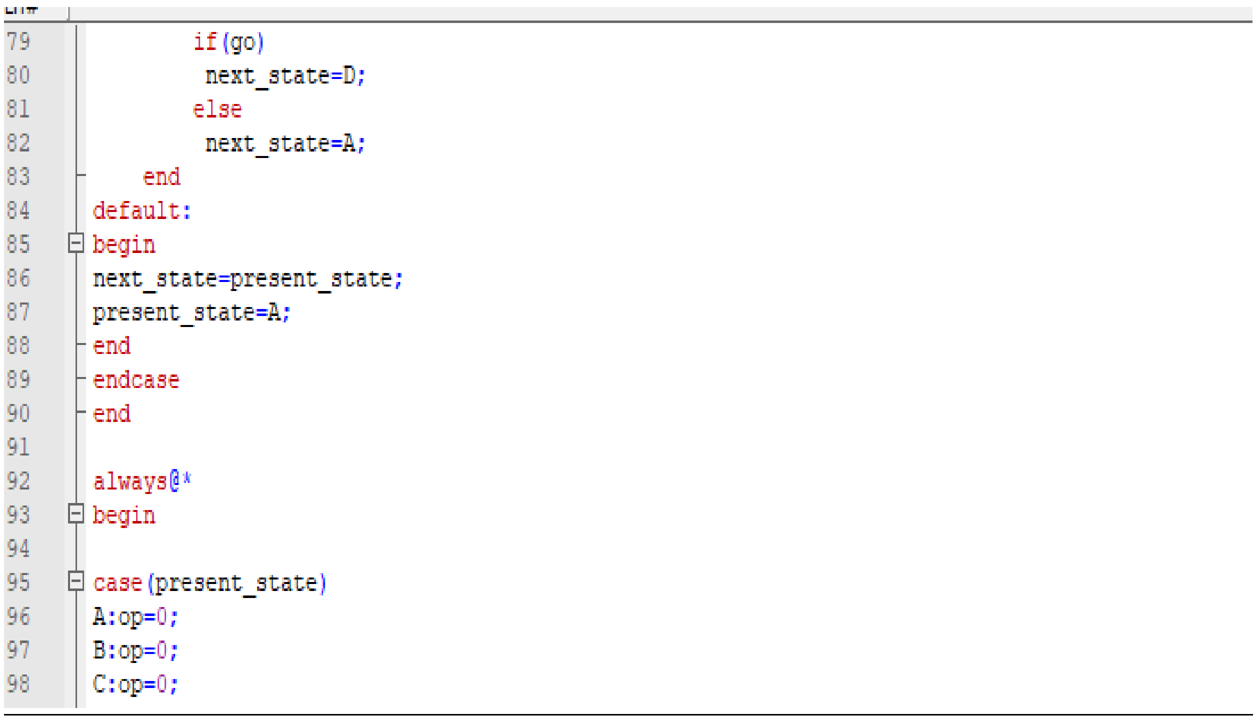
```
else
present_state<=next_state;
end

always@*
begin
case(present_state)
A:
  begin
        if(go)
         next_state=B;
        else
         next_state=A;
 end
B:
  begin
        if(go)
         next_state=B;
        else
```



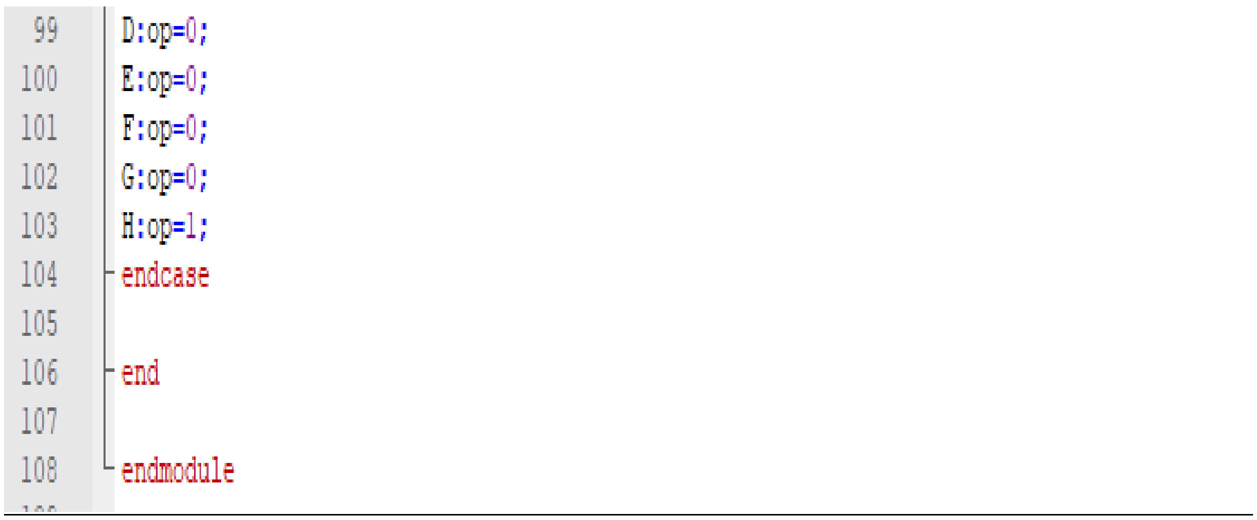
```
         next_state=C;
  end
C:
    begin
         if(go)
         next_state=D;
         else
         next_state=A;
    end
D:
    begin
        if(go)
         next_state=E;
        else
         next_state=C;
    end
E:
    begin
        if(go)
         next_state=B;
```



```
         next_state=B;
        else
         next_state=F;
    end
F:
    begin
        if(go)
         next_state=G;
        else
         next_state=A;
    end
G:
    begin
        if(go)
         next_state=E;
        else
         next_state=H;
    end
H:
    begin
```

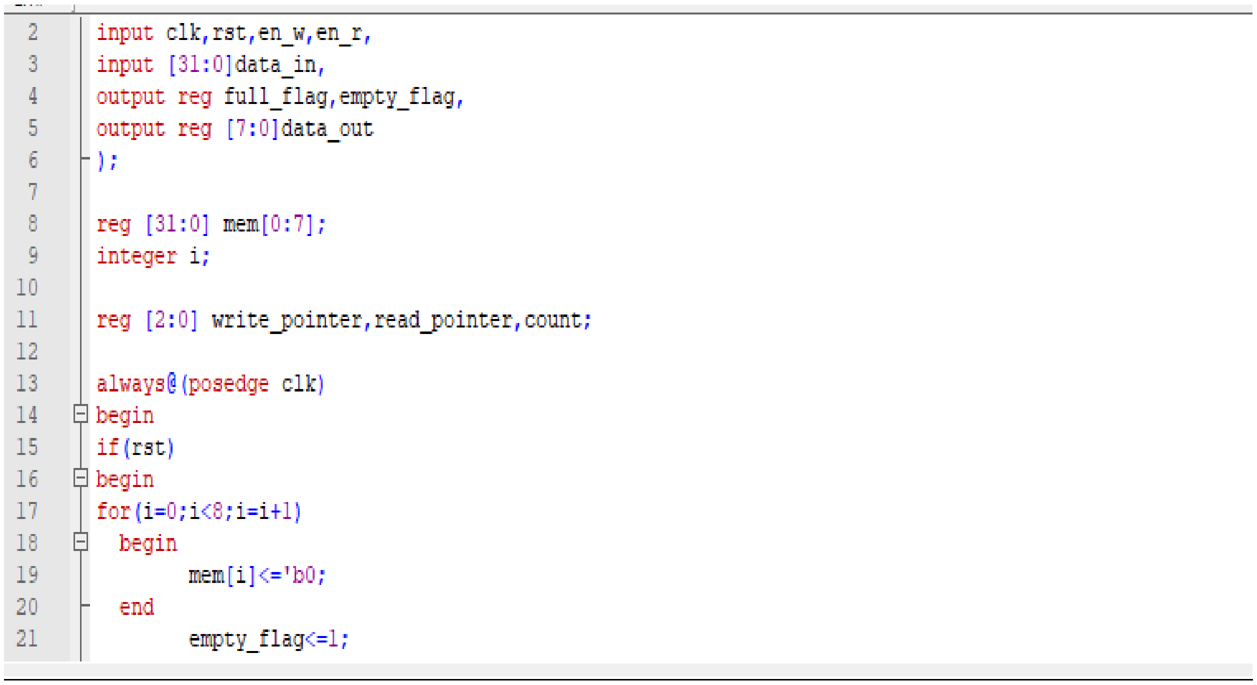


```
        if(go)
         next_state=D;
        else
         next_state=A;
    end
default:
begin
next_state=present_state;
present_state=A;
end
endcase
end

always@*
begin

case(present_state)
A:op=0;
B:op=0;
C:op=0;
```

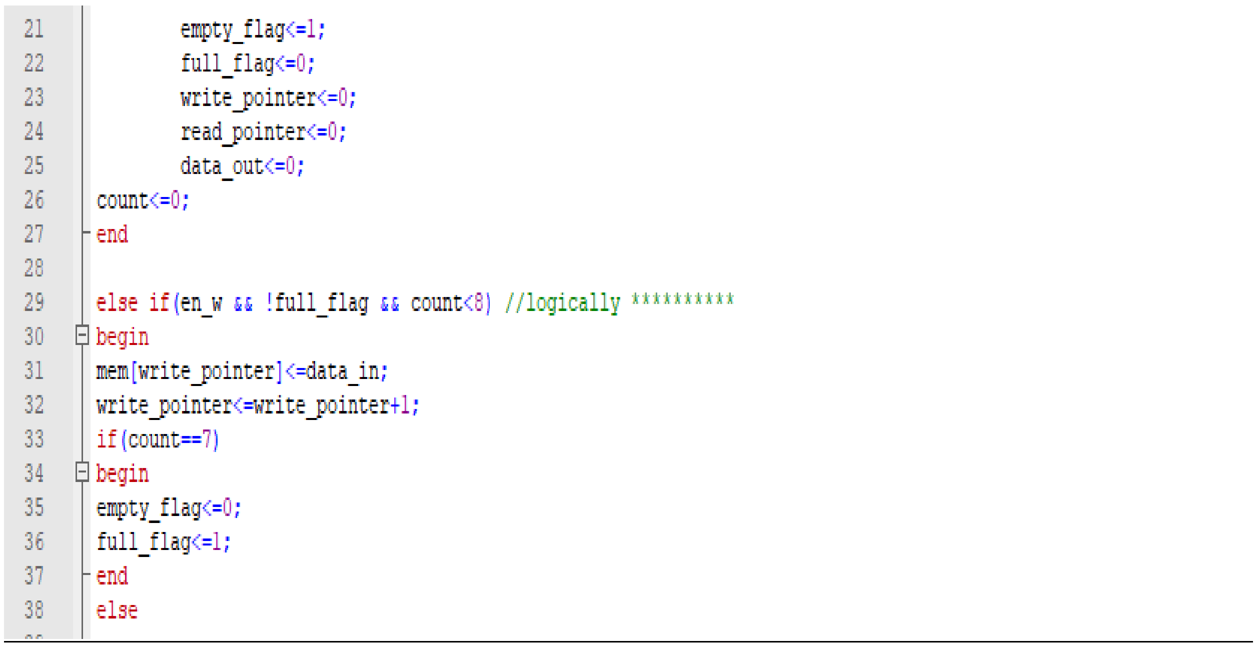


```
D:op=0;
E:op=0;
F:op=0;
G:op=0;
H:op=1;
endcase

end

endmodule
```

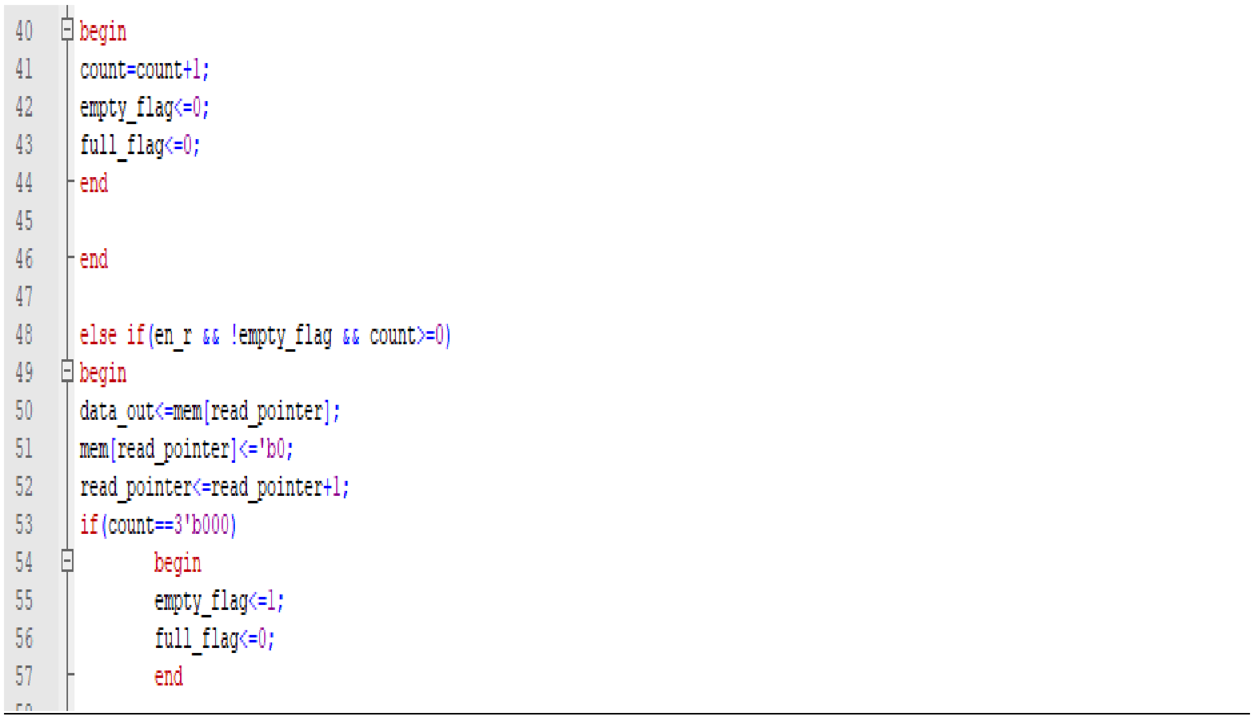
Edit for fifo code



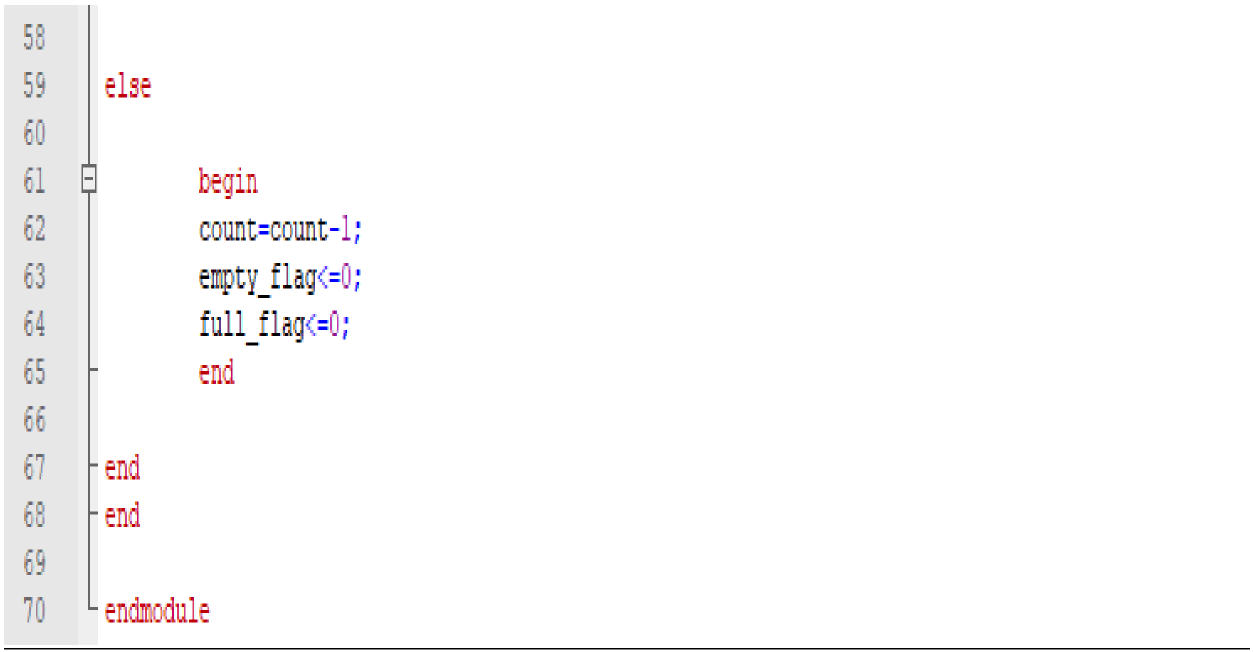
```
input clk,rst,en_w,en_r,
input [31:0]data_in,
output reg full_flag,empty_flag,
output reg [7:0]data_out
);

reg [31:0] mem[0:7];
integer i;

reg [2:0] write_pointer,read_pointer,count;

always@(posedge clk)
begin
if(rst)
begin
for(i=0;i<8;i=i+1)
  begin
        mem[i]<='b0;
  end
        empty_flag<=1;
```



```
        empty_flag<=1;
        full_flag<=0;
        write_pointer<=0;
        read_pointer<=0;
        data_out<=0;
count<=0;
end

else if(en_w && !full_flag && count<8) //logically **********
begin
mem[write_pointer]<=data_in;
write_pointer<=write_pointer+1;
if(count==7)
begin
empty_flag<=0;
full_flag<=1;
end
else
```



```
begin
count=count+1;
empty_flag<=0;
full_flag<=0;
end

end

else if(en_r && !empty_flag && count>=0)
begin
data_out<=mem[read_pointer];
mem[read_pointer]<='b0;
read_pointer<=read_pointer+1;
if(count==3'b000)
        begin
        empty_flag<=1;
        full_flag<=0;
        end
```



```

else

        begin
        count=count-1;
        empty_flag<=0;
        full_flag<=0;
        end

end
end

endmodule
```