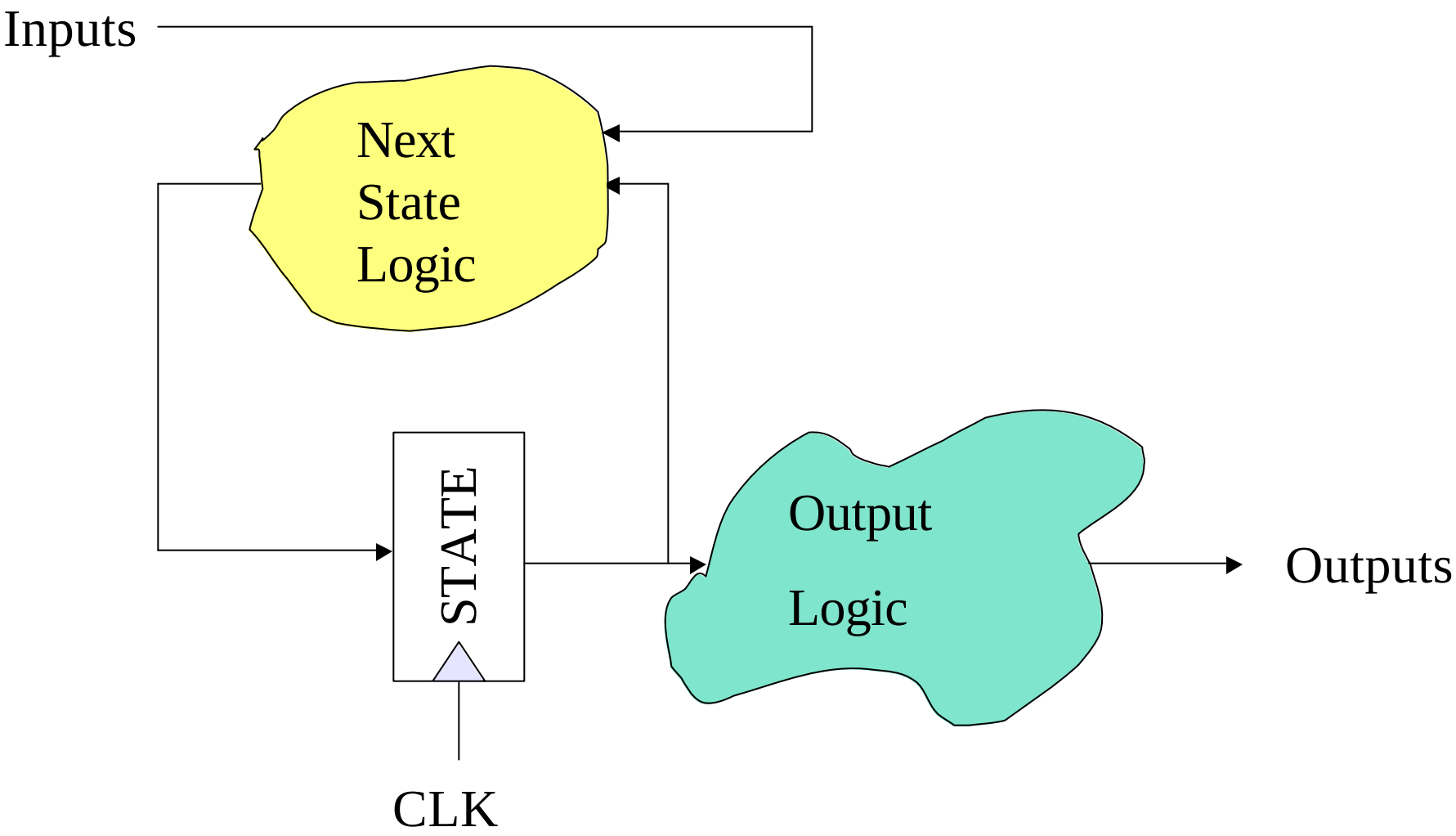
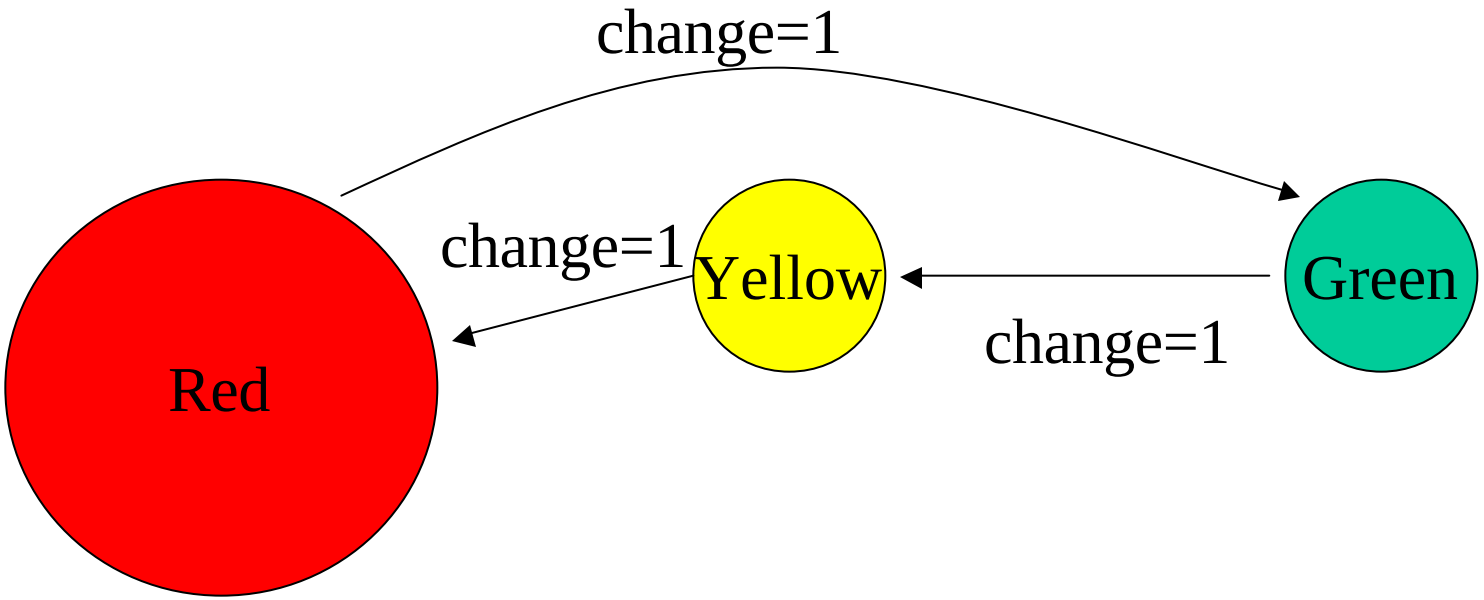
FSMs in Verilog
and other random things

9/27/02

FSM structure




change=1
change=1
Yellow
Green
change=1
Red

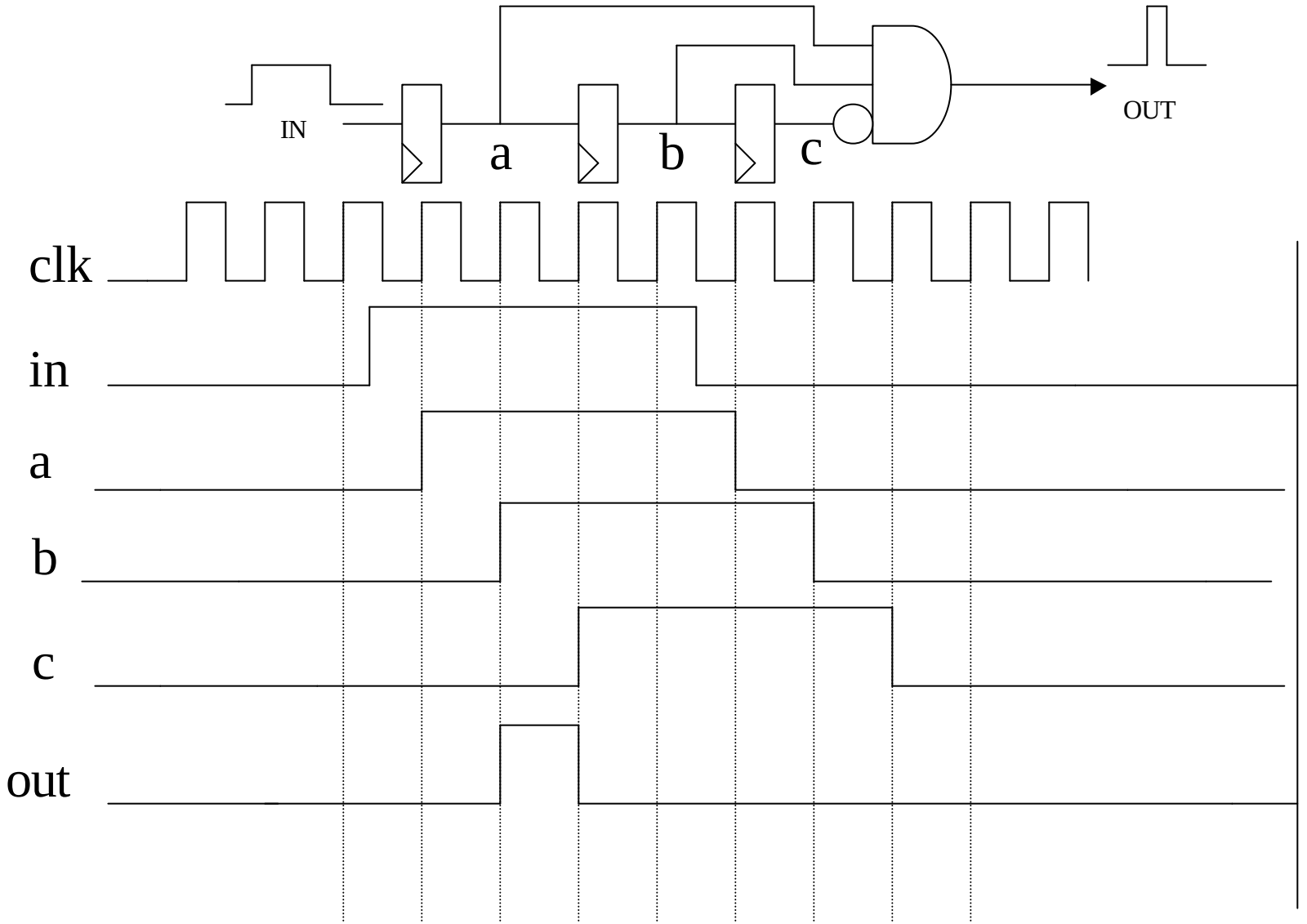
```
module trafficLightFSM(clk,reset,change,red,yellow,green);
        input clk,reset,change;
        output red,yellow,green;
        reg red,yellow,green;
        reg [1:0] curState,nextState;
        parameter showRed = 2'b00, showYellow = 2'b01;
        parameter showGreen = 2'b10;


// state register
always @(posedge clk)
        if (reset==1'b1) curState <= 2'b0;
        else curState <= nextState;
```

```
// next state logic
// dependent only on the current state and input
always @(curState or change)
begin
	nextState = showRed; // default state
	case (curState)
		showRed: if (change) nextState=showGreen;
		showYellow: if (change) nextState=showRed;
			else nextState=showYellow;
		showGreen: if (change) nextState=showYellow;
			else nextState=showGreen;
	endcase
end
```

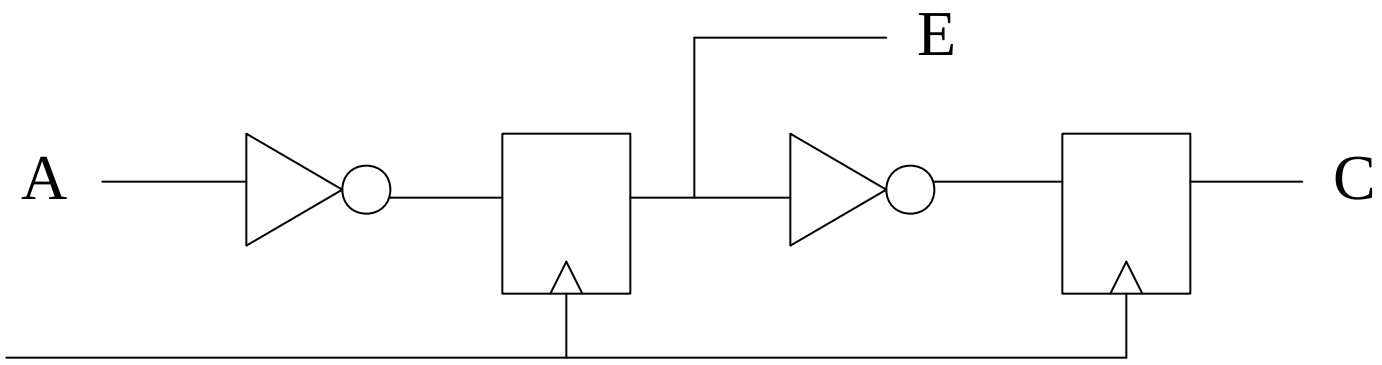
```
// Output Logic: dependent ONLY on state
always @(curState) begin
   // *ALWAYS* put default output values
   red=1'b0; green=1'b0; yellow=1'b0;
   case (curState)
        showRed: red=1'b1;
        showYellow: yellow=1'b1;
        showGreen: green=1'b1;
   endcase
end
endmodule
```

Edge Detector



Non-blocking

```
always @(posedge clk)
   begin
       E <= ~A;
       C <= ~E;
   end
```



Blocking

```
always @(posedge clk)
   begin
       E = ~A;
       C = ~E;
   end
```