[Introduction to Computer Architecture'21] Lab: Verilog Syntax 4

Question: Design a sequential synchronous circuit of two traffic lights at crossroad, each of which receives a signal whether a car is waiting to cross or not (Ta, Tb)

Answer:

First, we draw a state diagram of our possible states. Let La and Lb be our traffic lights

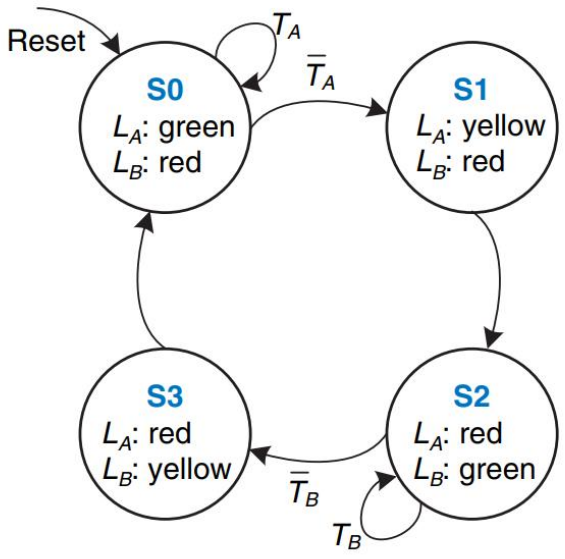
S0: La=green and Lb=red

S1: La=yellow and Lb=red

S2: La=red and Lb= green

S3: La=red and Lb=yellow

The following state diagram illustrates the transition of the traffic lights



As in the diagram. we can see that the traffic light remains at S0 until all the cars waiting to pass from the La Traffic light have passed and then it transitions to the following state S1 which in turn transitions to state S2 and remain there as long as there are still cars waiting to pass from traffic light Lb (as long as it receives signal) finally once all cars have passed it transitions to the following state S3 which in turn transition to S0 and the cycle is repeated.

From the diagram we can draw the State transition table.

But first we must encode the states:

State	Signal (S1 and S0)
S0	00
S1	01
S2	10
S3	11

The above table is called the state encoding table

Now we can make our state transition table

Current State		Ta	Tb	Next State	
S1	S0			S1'	S0'
0	0	1	X	0	0
0	0	0	X	0	1
0	1	X	X	1	0
1	0	X	1	1	0
1	0	X	0	1	1
1	1	X	X	0	0

Now from the above truth table we can deduce our next state logic

S1'= ((!S1) & S0) | (S1 & (!S0) & Tb) | (S1 & S0 & (!Tb))

S0'= (((!S0)& (!S1) & (!Ta)) + (S1 & (!S0) &(!Tb))

The above expressions can be reduced to

S1'= $S1 \oplus S0$

S0'= ((!S0)& (!S1) & (!Ta)) + (S1 & (!S0) & (!Tb))

Our next step is to create our current state logic (the relation between the states and the traffic lights)

We first must encode our outputs

Output encoding

Output	Signal
green	00
yellow	01
red	10

Our output table is

Current State		La		Lb	
S1	S0	La1	La0	Lb1	Lb0
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

From the truth table we can deduce that our current state logic

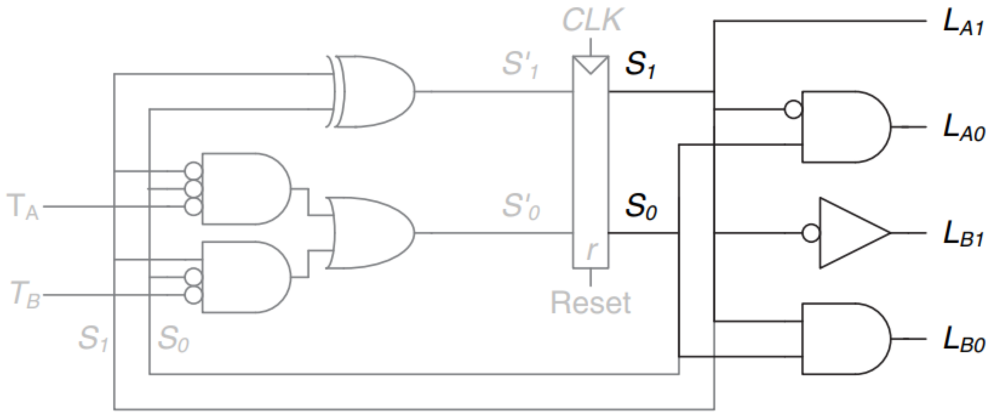
La1 = S1

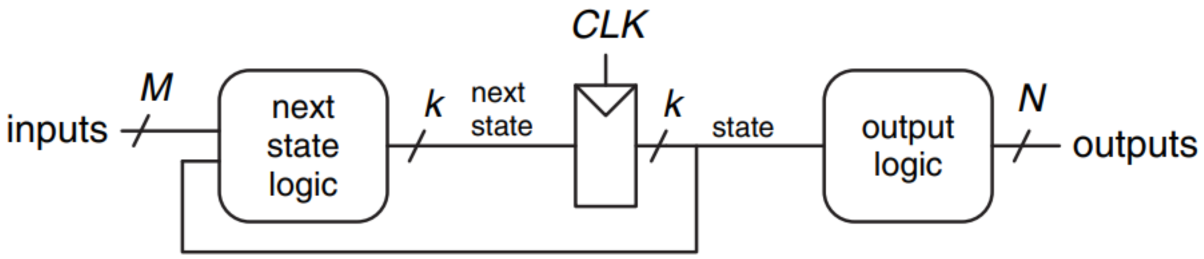
La0 = (!S1)&S0

Lb1 = (!S1)

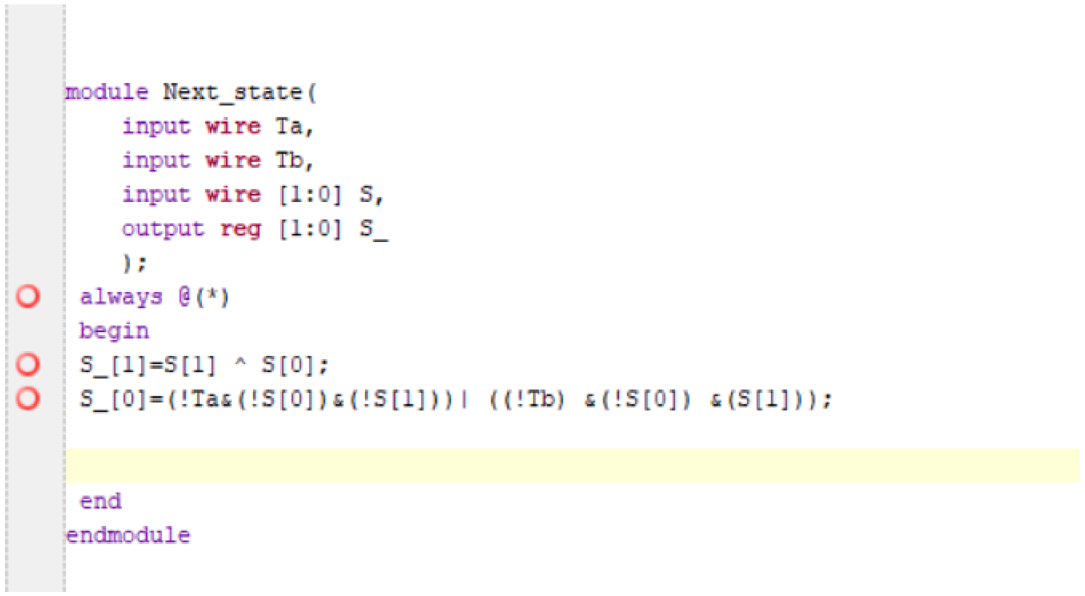
Lb0 = S1&S0

Our schematic:

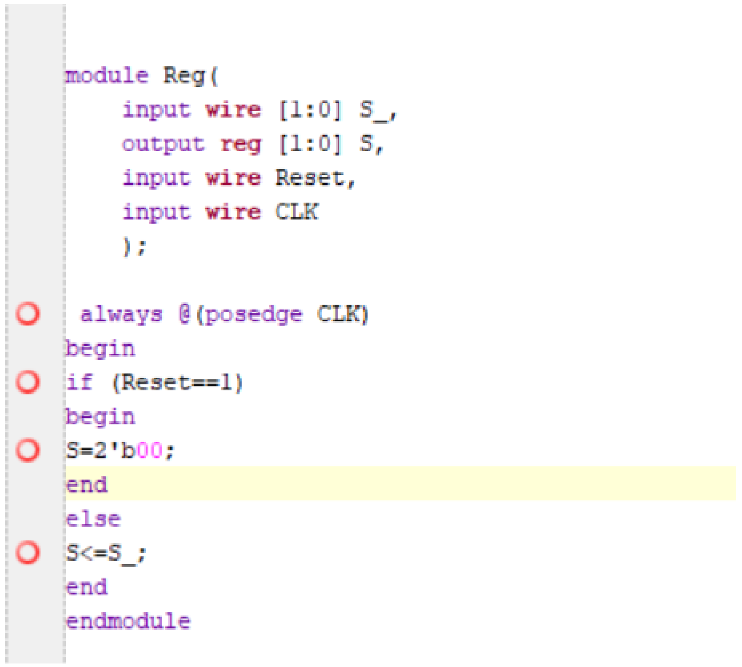




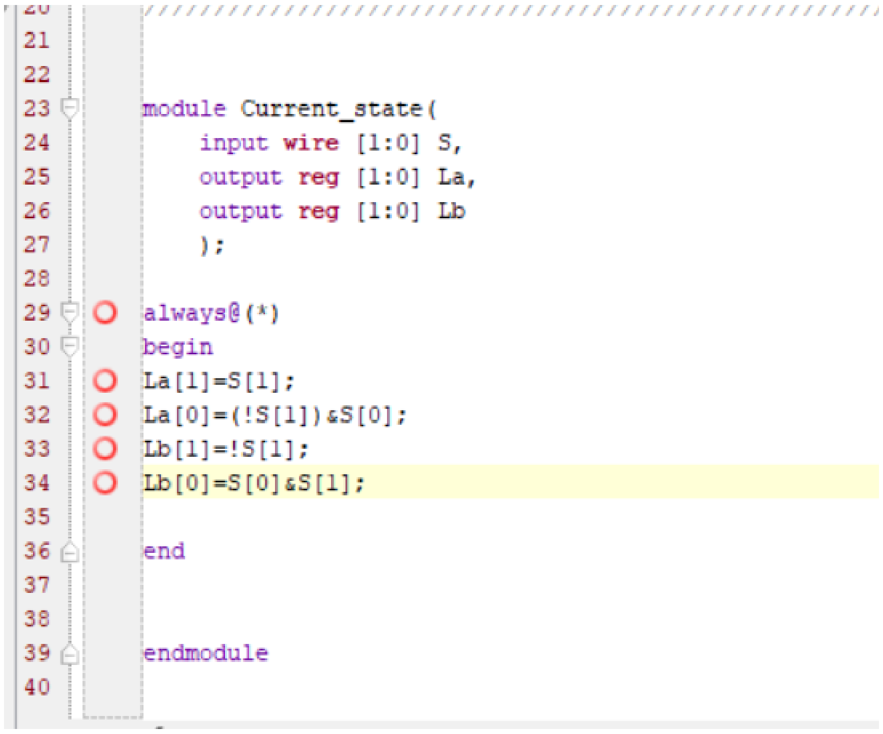
Verilog code:



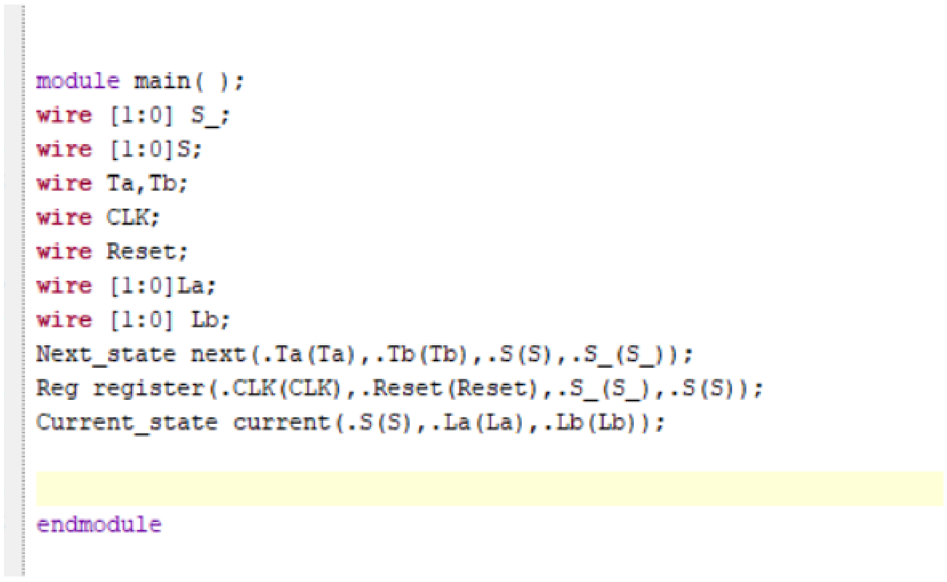
```
module Next_state(
    input wire Ta,
    input wire Tb,
    input wire [1:0] S,
    output reg [1:0] S_
    );
 always @(*)
 begin
 S_[1]=S[1] ^ S[0];
 S_[0]=(!Ta&(!S[0])&(!S[1]))| ((!Tb) &(!S[0]) &(S[1]));


 end
endmodule
```



```
module Reg(
    input wire [1:0] S_,
    output reg [1:0] S,
    input wire Reset,
    input wire CLK
    );

 always @(posedge CLK)
begin
if (Reset==1)
begin
S=2'b00;
end
else
S<=S_;
end
endmodule
```

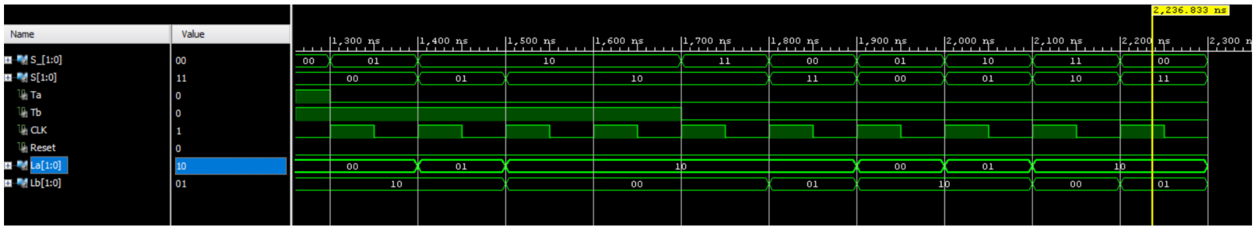


```
/////////////////////////////////////////////////////


module Current_state(
    input wire [1:0] S,
    output reg [1:0] La,
    output reg [1:0] Lb
    );

always@(*)
begin
La[1]=S[1];
La[0]=(!S[1])&S[0];
Lb[1]=!S[1];
Lb[0]=S[0]&S[1];

end


endmodule

```



```
module main( );
wire [1:0] S_;
wire [1:0]S;
wire Ta,Tb;
wire CLK;
wire Reset;
wire [1:0]La;
wire [1:0] Lb;
Next_state next(.Ta(Ta),.Tb(Tb),.S(S),.S_(S_));
Reg register(.CLK(CLK),.Reset(Reset),.S_(S_),.S(S));
Current_state current(.S(S),.La(La),.Lb(Lb));


endmodule
```

Waveform:


Name
Value
S_[1:0]
S[1:0]
Ta
Tb
CLK
Reset
La[1:0]
Lb[1:0]
2,236.833 ns
1,300 ns
1,400 ns
1,500 ns
1,600 ns
1,700 ns
1,800 ns
1,900 ns
2,000 ns
2,100 ns
2,200 ns
2,300 ns