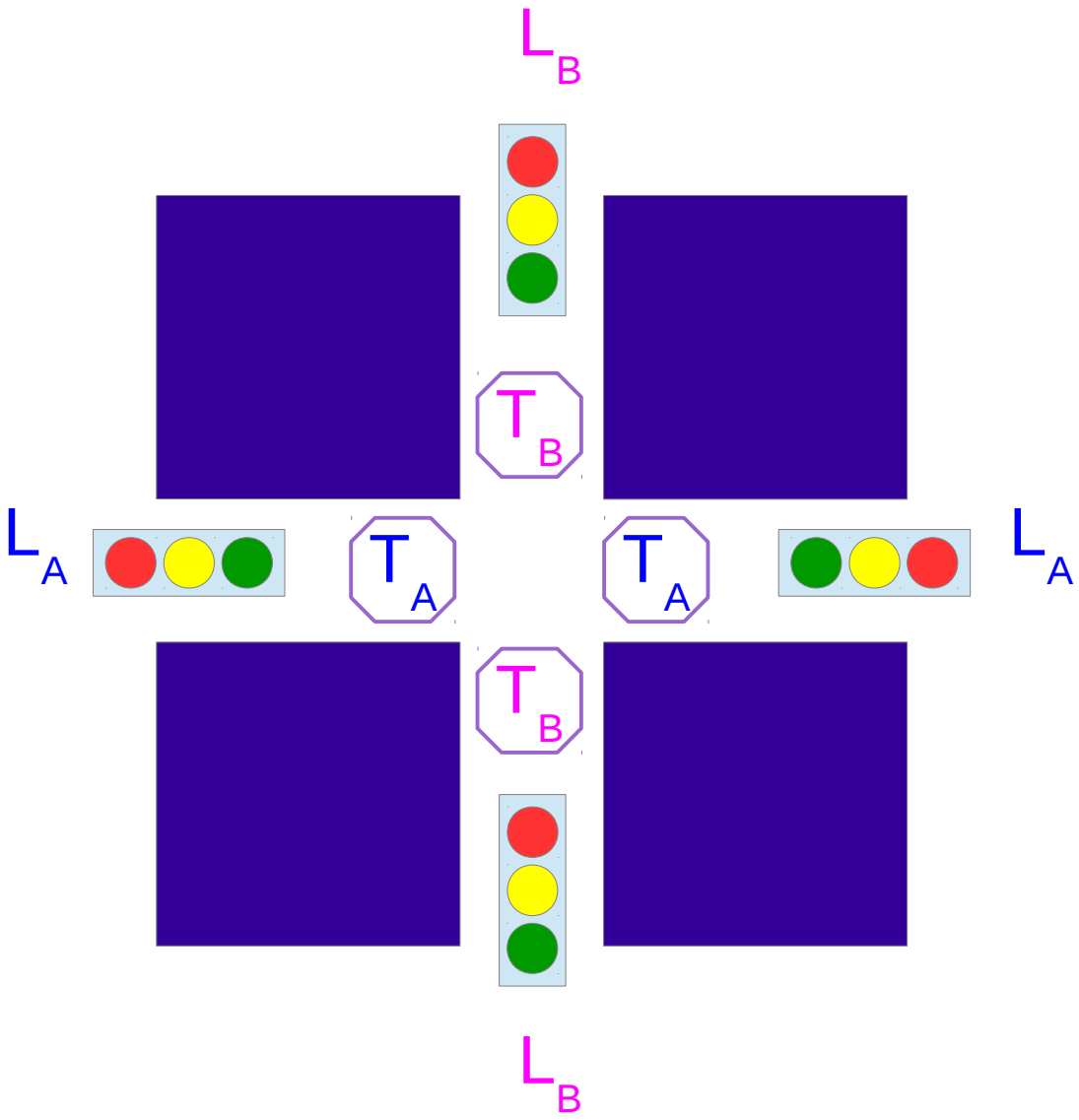
Gate Level Design Example (2A)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

FSM Inputs and Outputs



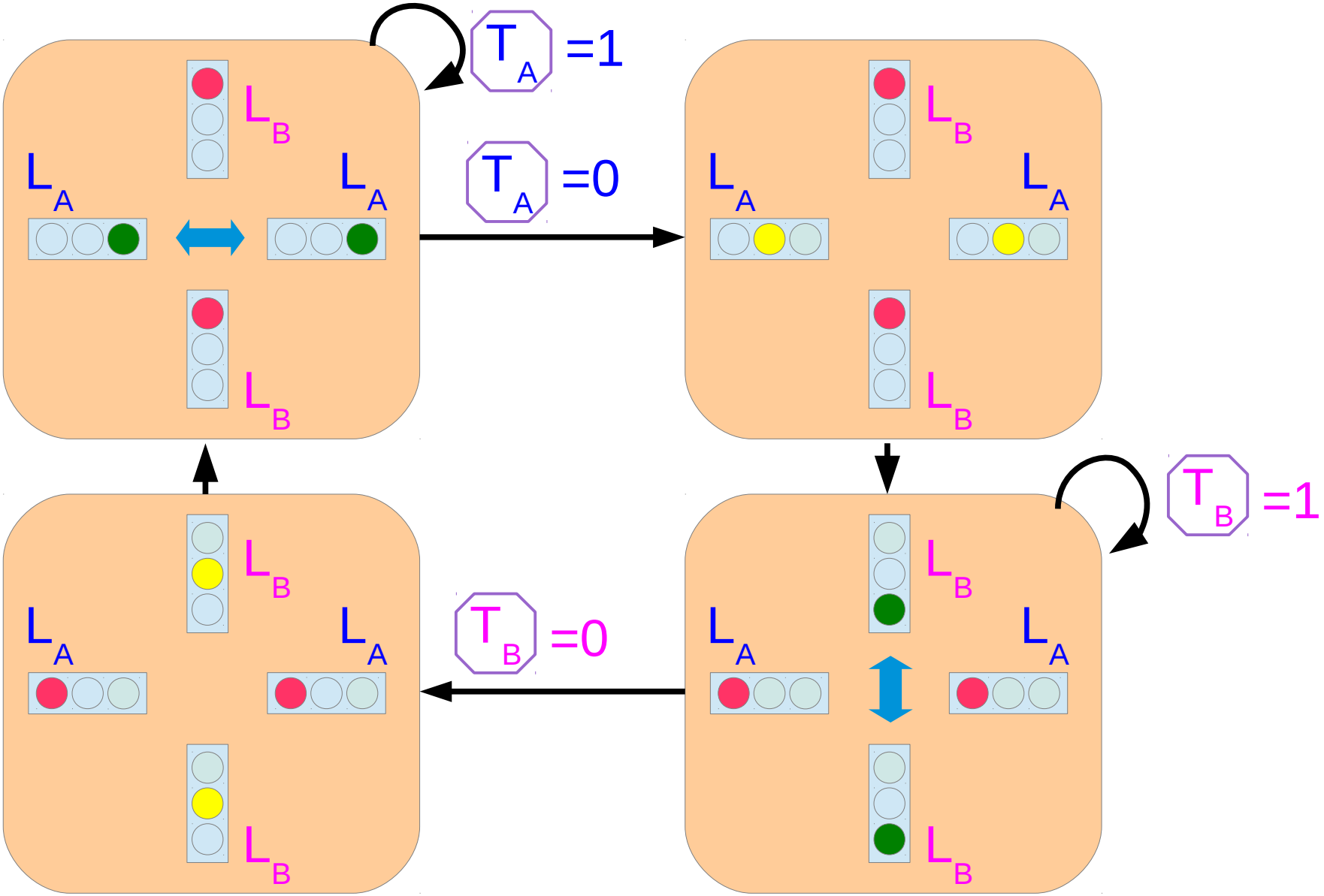
Traffic Lights - Outputs

L_A L_B

Sensor - Inputs

T_A T_B

States



Moore FSM State Transition Table

S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

	S_1	S_0	T_A	T_B	S'_1
	0	0	0	X	0
	0	0	1	X	0
$\overline{S_1}S_0$ ⇨	0	1	X	X	1
$S_1\overline{S_0}\overline{T_B}$ ⇨	1	0	X	0	1
$S_1\overline{S_0}T_B$ ⇨	1	0	X	1	1
	1	1	X	X	0

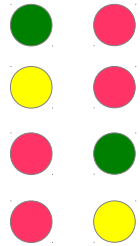
$$S'_1 = \overline{S_1}S_0 + S_1\overline{S_0} = S_1 \oplus S_0$$

	S_1	S_0	T_A	T_B	S'_0
$\overline{S_1}\overline{S_0}\overline{T_A}$ ⇨	0	0	0	X	1
	0	0	1	X	0
	0	1	X	X	0
$S_1\overline{S_0}\overline{T_B}$ ⇨	1	0	X	0	1
	1	0	X	1	0
	1	1	X	X	0

$$S'_0 = \overline{S_1}\overline{S_0}\overline{T_A} + S_1\overline{S_0}\overline{T_B}$$

States

S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1



00
01
10

S_1	S_0	L_{A1}
0	0	0
0	1	0
1	0	1
1	1	1

$$L_{A1} = S_1$$

S_1	S_0	L_{A0}
0	0	0
0	1	1
1	0	0
1	1	0

$$L_{A0} = \overline{S_1} S_0$$

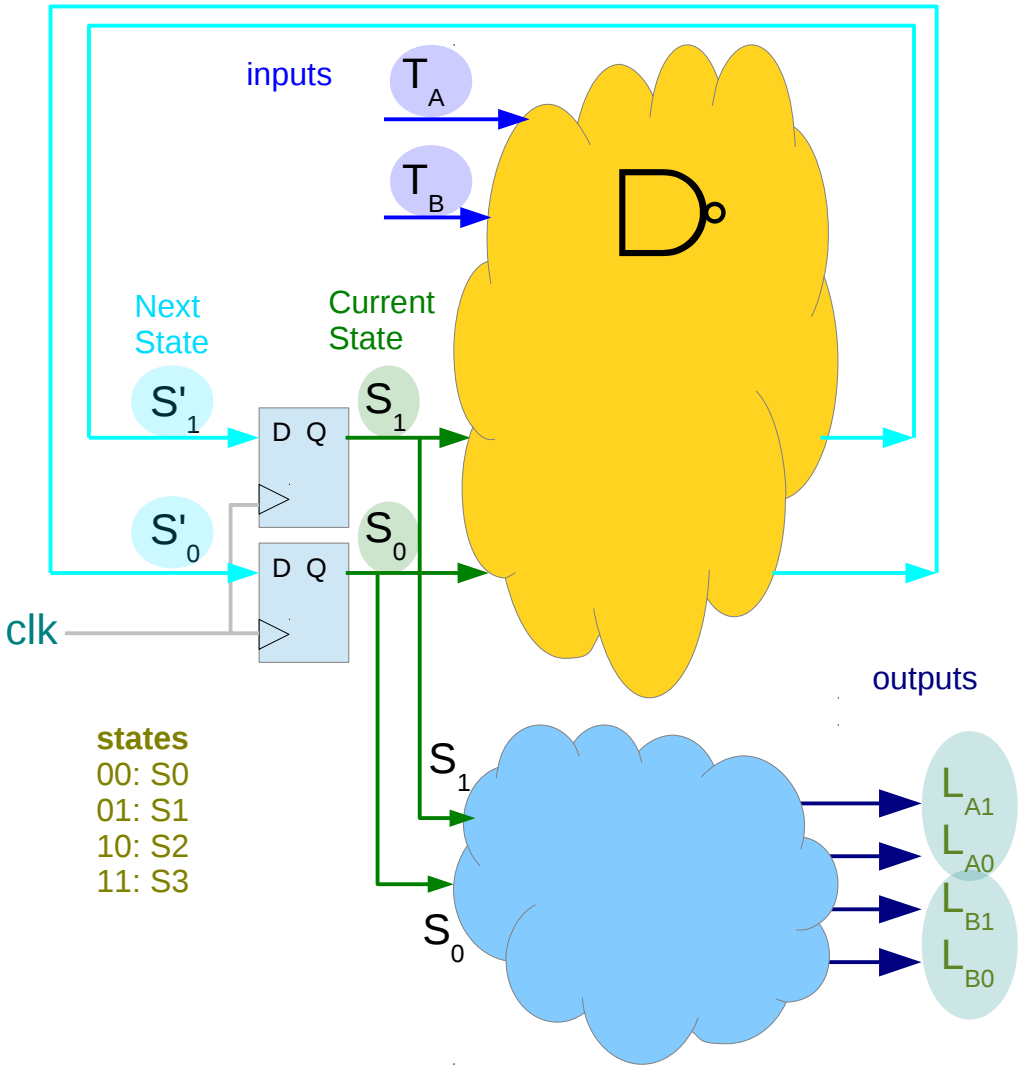
S_1	S_0	L_{B1}
0	0	1
0	1	1
1	0	0
1	1	0

$$L_{B1} = \overline{S_1}$$

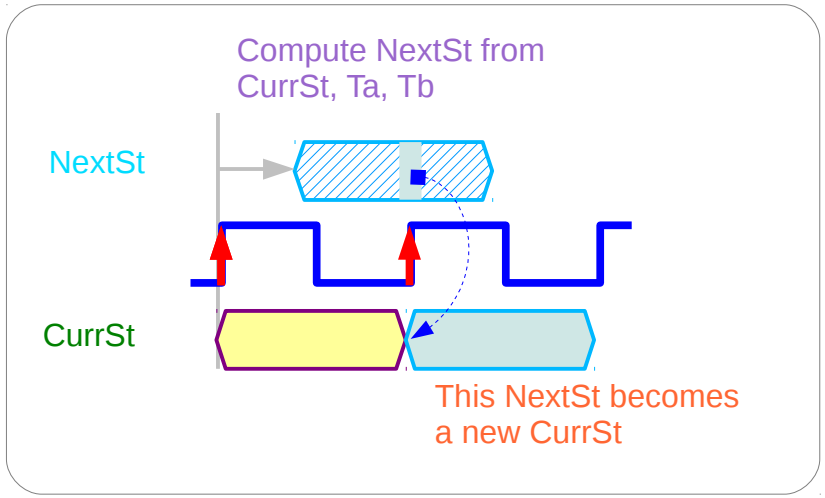
S_1	S_0	L_{B0}
0	0	0
0	1	0
1	0	0
1	1	1

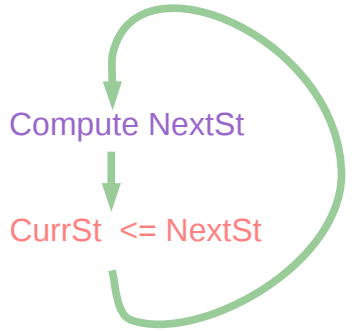
$$L_{B0} = S_1 S_0$$

Moore FSM (1)



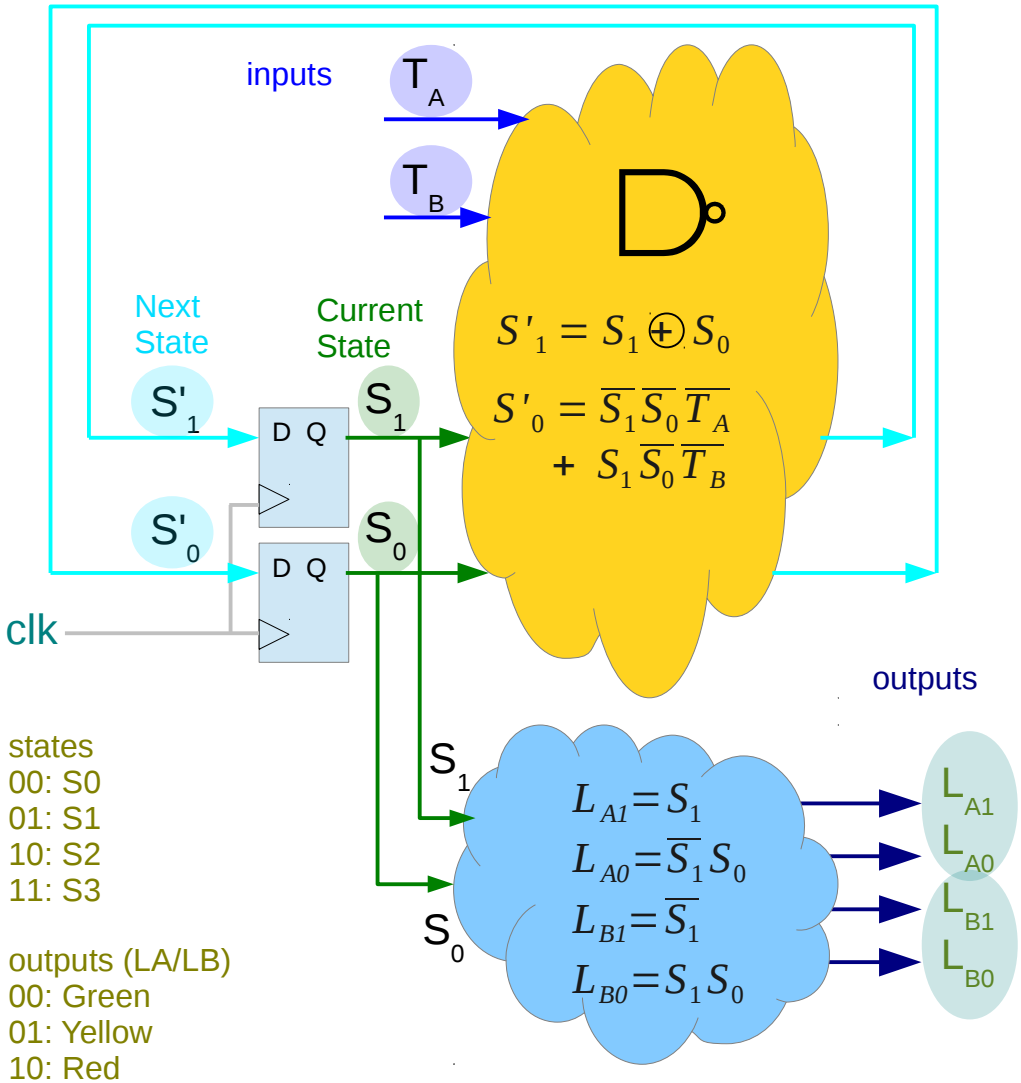
states
00: S0
01: S1
10: S2
11: S3





outputs (LA/LB)
00: Green
01: Yellow
10: Red
11: X

Moore FSM



Inputs T_A T_B

Current State S_1 S_0

Next States

$S'_1 = S_1 \oplus S_0$

$S'_0 = \overline{S_1}\,\overline{S_0}\,\overline{T_A} + S_1\,\overline{S_0}\,\overline{T_B}$

Current State S_1 S_0

Outputs

$L_{A1} = S_1$ $L_{B1} = \overline{S_1}$

$L_{A0} = \overline{S_1}\,S_0$ $L_{B0} = S_1 S_0$

Verilog Gate Level Design - testbench

```
`timescale 1ns/100ps

module traffic_controller_testbench;

  parameter cycle = 40;

  reg clock;

  always
    begin
      #(cycle/2) clock=~clock;
    end


  traffic_controller DUT (.clock(clock),
                          .reset(reset),
                          .TA(TA),
                          .TB(TB),
                          .LA(LA),
                          .LB(LB) );

  reg       reset;
  reg       TA, TB;
  wire [1:0] LA, LB;
```

```
initial
  begin
    clock = 1;
    reset = 1;
    TA = 1;
    TB = 0;

     #1;
    #(cycle)     reset = 0;
    #(cycle)     TB = 1;
    #(cycle)     TA = 0;
    #(cycle*3)   TA = 1;
                 TB = 0;
    #(cycle*3)   TA = 0;
  end

initial
  begin
    $dumpfile("traffic.vcd");
    $dumpvars(0, DUT);
    #(cycle * 10);
    $finish;
  end

endmodule
```

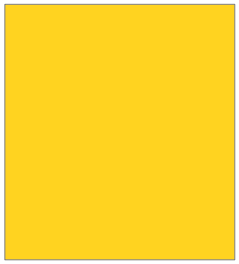
Verilog Gate Level Design – traffic_gate.v

```
module traffic_controller(clock, reset, TA, TB, LA, LB);
  input clock, reset;
  input TA, TB;
  output [1:0] LA, LB;

  reg    [1:0] S;
  wire   [1:0] NextS;

  always @(posedge clock)
    begin: SEQ
      if (reset)
        #8 S = 2'b00;
      else
        #8 S = NextS;
    end
```

```
    not #8 (S1b, S[1]);
    not #8 (S0b, S[0]);
    not #8 (TAb, TA);
    not #8 (TBb, TB);

    xor #8 (NextS[1], S[1], S[0]);
    or  #8 (NextS[0], NS1, NS2);
    and #8 (NS1, S1b,  S0b, TAb);
    and #8 (NS2, S[1], S0b, TBb);


    buf #8 (LA[1], S[1]);
    and #8 (LA[0], S1b, S[0]);
    not #8 (LB[1], S[1]);
    and #8 (LB[0], S[1], S[0]);

endmodule
```

traffic_controller – gate level design (1)

```
module traffic_controller
  (clock, reset, TA, TB, LA, LB);

  input clock, reset;
  input TA, TB;
  output [1:0] LA, LB;


  wire S1, S0, NS1, NS0;
  wire S1b, S0b;
  wire TAb, TBb;
  wire n1, n2;
```



```
endmodule
```

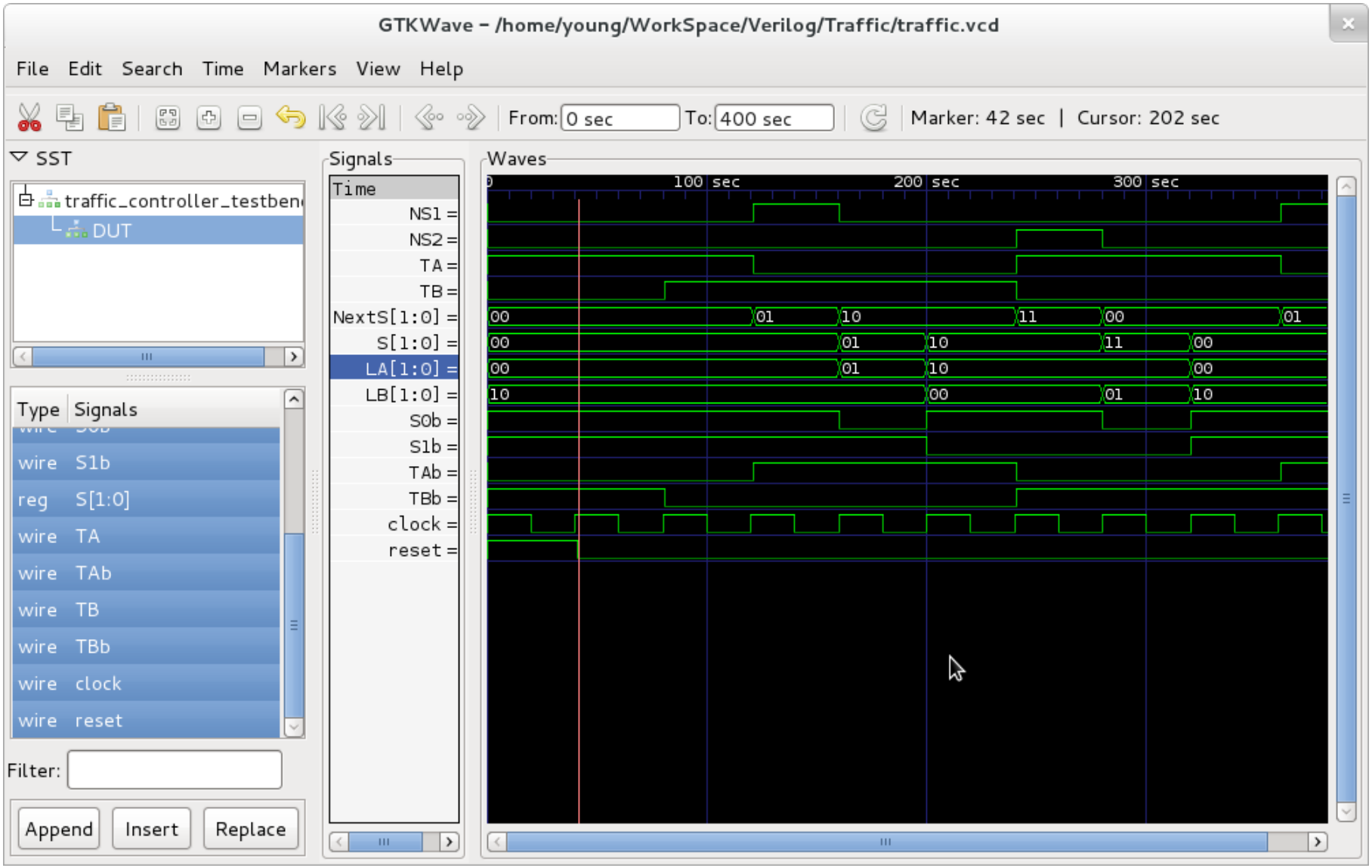
```
not U1 (S1b, S1);
not U2 (S0b, S0);

not U3 (TAb, TA);
not U4 (TBb, TB);

xor U5 (NS1, S1, S0);

and U6 (n1, S1b, S0b, TAb);
and U7 (n2, S1, S0b, TBb);
or  U8 (NS0, n1, n2);

dff U9 (S0, NS0, clock, reset);
dff U10 (S1, NS1, clock, reset);

buf U11 (LA[1], S1);
and U12 (LA[0], S1b, S0);
buf U13 (LB[1], S1b);
and U14 (LB[0], S1, S0);
```

traffic_controller – gate level design (2)

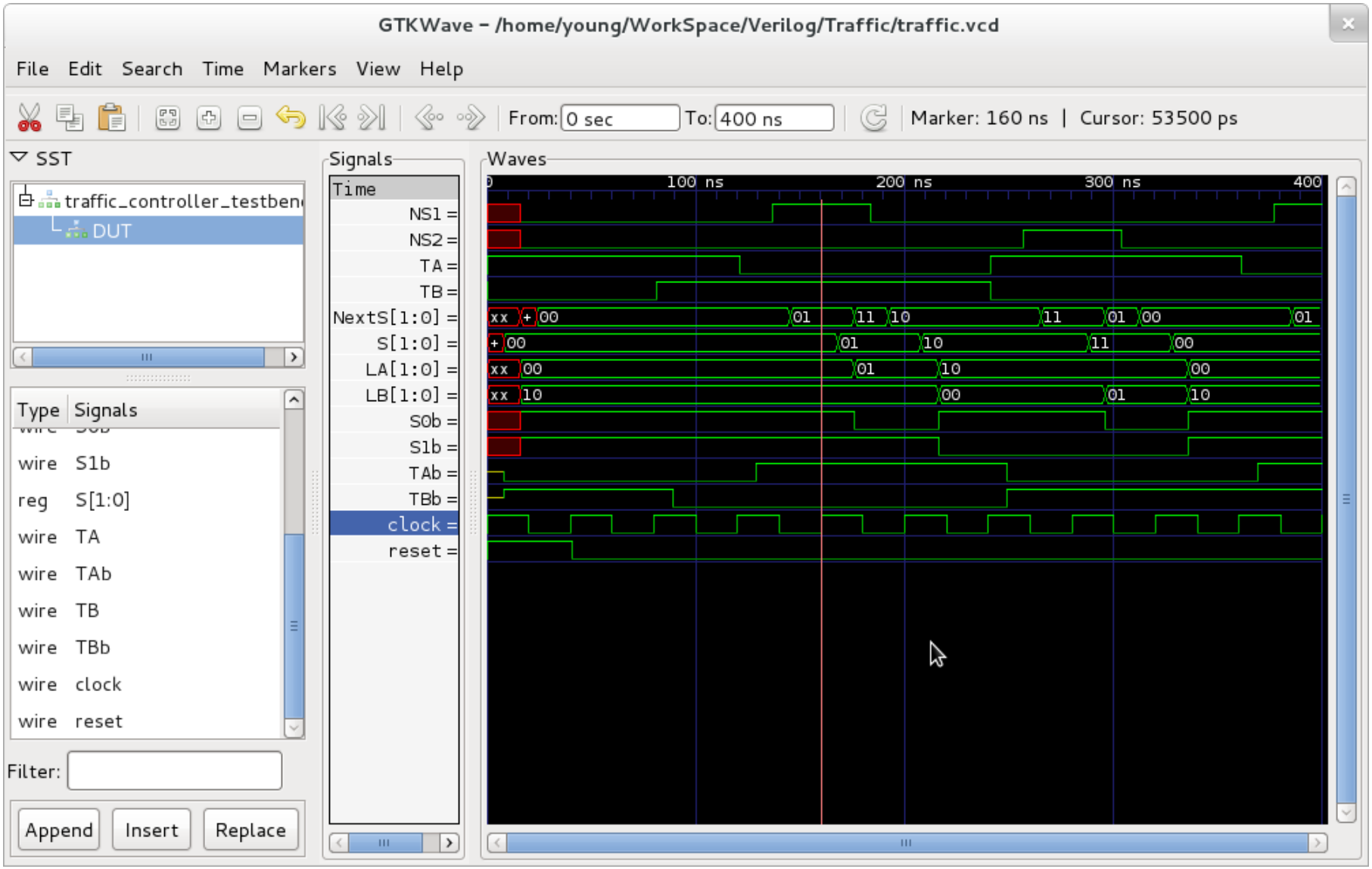
```
primitive dff (q, d, clk, rst);
input d, clk, rst;
output q;
reg    q;

initial   q = 0;

table
//  d    clk    rst :   q :     q+;
//  Handle Rising Edge
    0    (01)  0   :   ? :     0;
    1    (01)  0   :   ? :     1;
    1    (0?)  0   :   1 :     1;
    0    (0?)  0   :   0 :     0;
//  Ignore Falling Edge
    ?    (?0)  0   :   ? :     -;
//  Ignore data changes on Steady Clock
    (??) ?     0   :   ? :     -;
//  RESET Cases
    ?    (??)  1   :   ? :     0;
    ?    ?      1   :   ? :     0;
    ?    ?     (10):   ? :     -;
endtable

endprimitive
```

```
not U1 (S1b, S1);
not U2 (S0b, S0);

not U3 (TAb, TA);
not U4 (TBb, TB);

xor U5 (NS1, S1, S0);

and U6 (n1, S1b, S0b, TAb);
and U7 (n2, S1, S0b, TBb);
or  U8 (NS0, n1, n2);

dff U9 (S0, NS0, clock, reset);
dff U10 (S1, NS1, clock, reset);

buf U11 (LA[1], S1);
and U12 (LA[0], S1b, S0);
buf U13 (LB[1], S1b);
and U14 (LB[0], S1, S0);
```

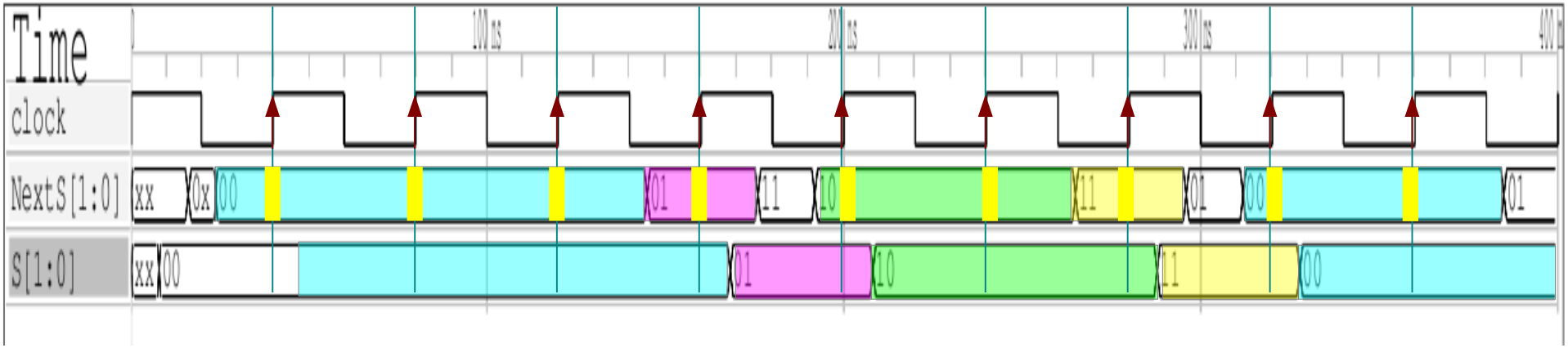
VCD Output with zero delay



VCD Output with gate delays



VCD Output with gate delays



Output Delay

References

[1] http://en.wikipedia.org/
[2] http://www.allaboutcircuits.com/
[3] W. Wolf, “Modern VLSI Design : Systems on Silicon
[4] N. Weste, D. Harris, “CMOS VLSI Design: A Circuits and Systems Perspective”
[5] J. P. Uyemura, “Introduction to VLSI Circuits and Systems”
[6] https://en.wikiversity.org/wiki/The_necessities_in_SOC_Design
[7] https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design
[8] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design
[9] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Architecture
[10] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Organization