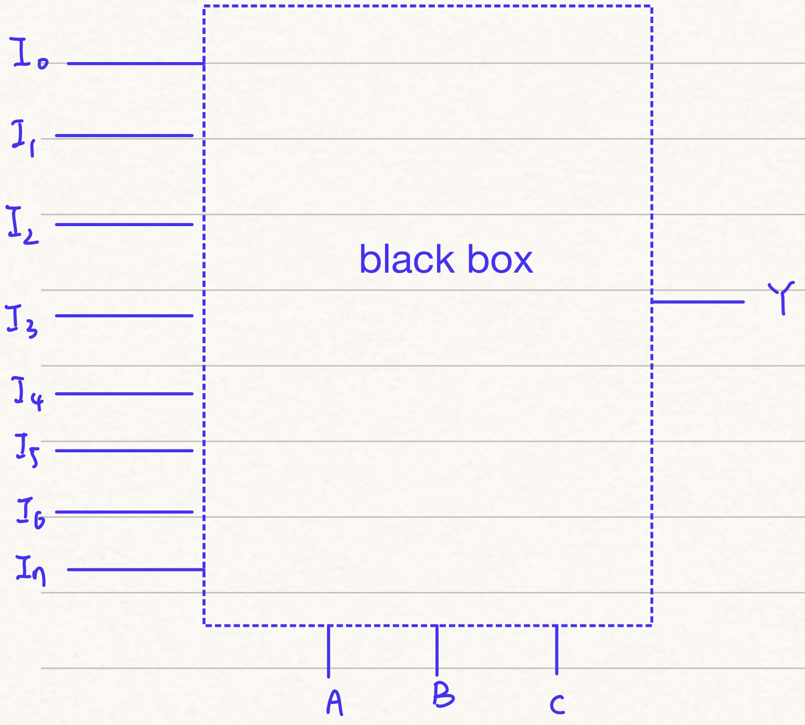
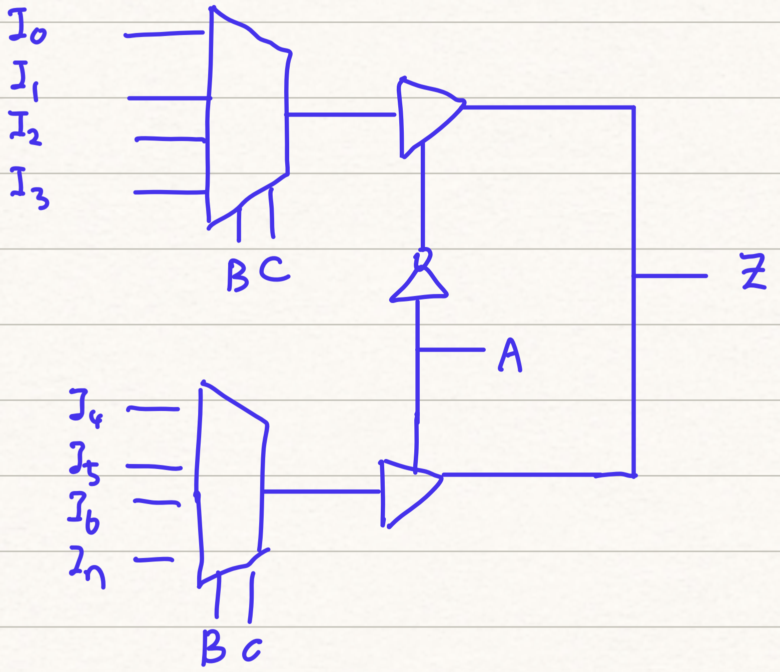
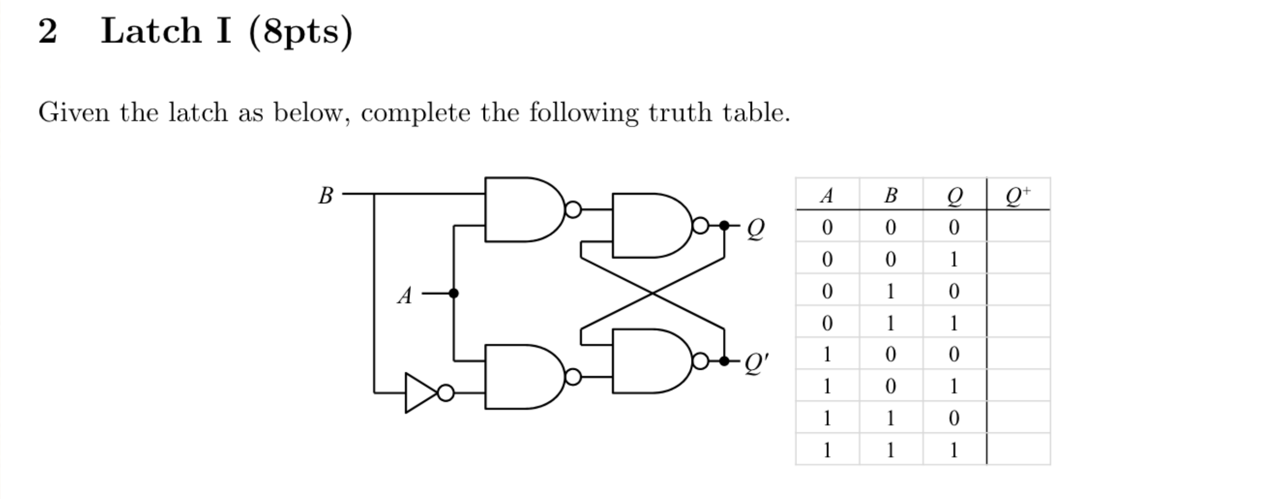
1 MUXes and Three-State Buffers (8pts)

Show how to make an 8-to-1 MUX using two 4-to-1 MUXes, two three-state buffers, and one inverter.



black box : use Shannon expansion $Y = A(\cdots\cdots) + A'(\quad)$

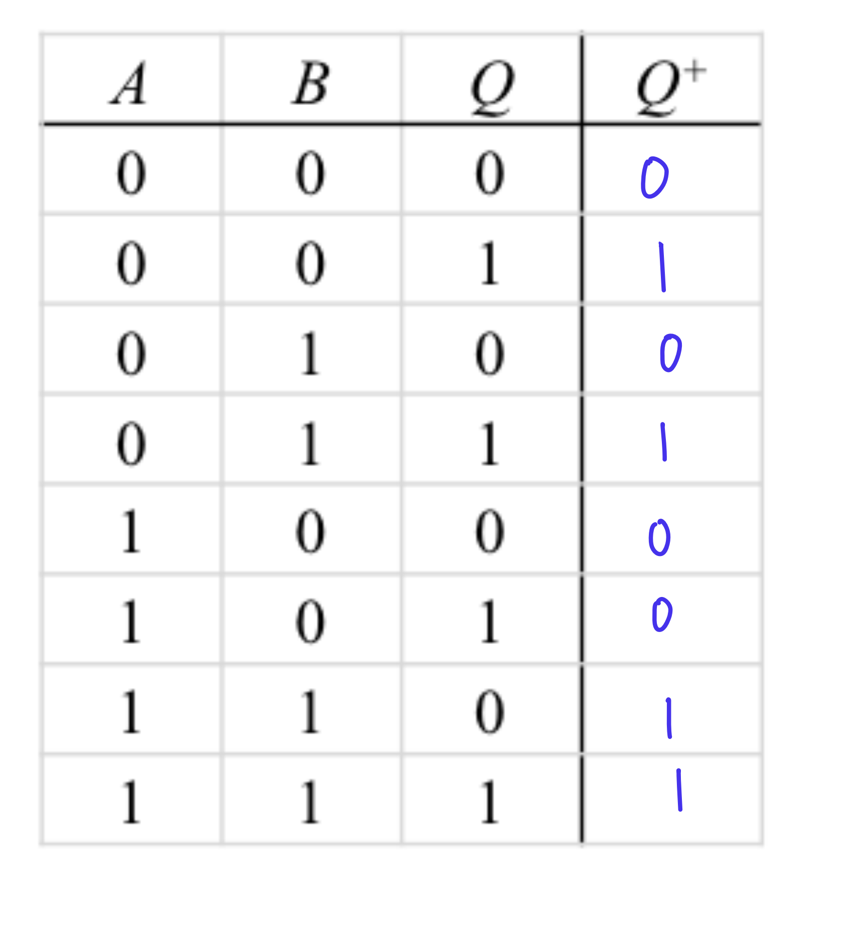




2 Latch I (8pts)

Given the latch as below, complete the following truth table.

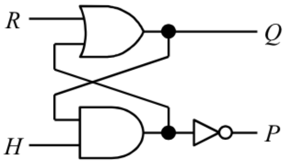
A	B	Q	Q^+
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



A	B	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

3 Latch II (16pts)

A latch can be constructed from an OR gate, an AND gate, and an inverter connected as below.



1. What restriction must be placed on R and H so that P will always equal Q' (under steady-state conditions)?
2. Construct a next-state table and derive the minimum characteristic (next-state) equation for the latch.

1. check $P \neq Q'$ 即 $P = Q$

當 $Q = 1$ 且 $P = 1$ => $H = 0$, $R = 1$

當 $Q = 0$ 且 $P = 0$ => $H = 1$ 且 $Q = 1$ (矛盾, 不存在)

故 restriction: $(R,H) = (1,0)$ #

2.

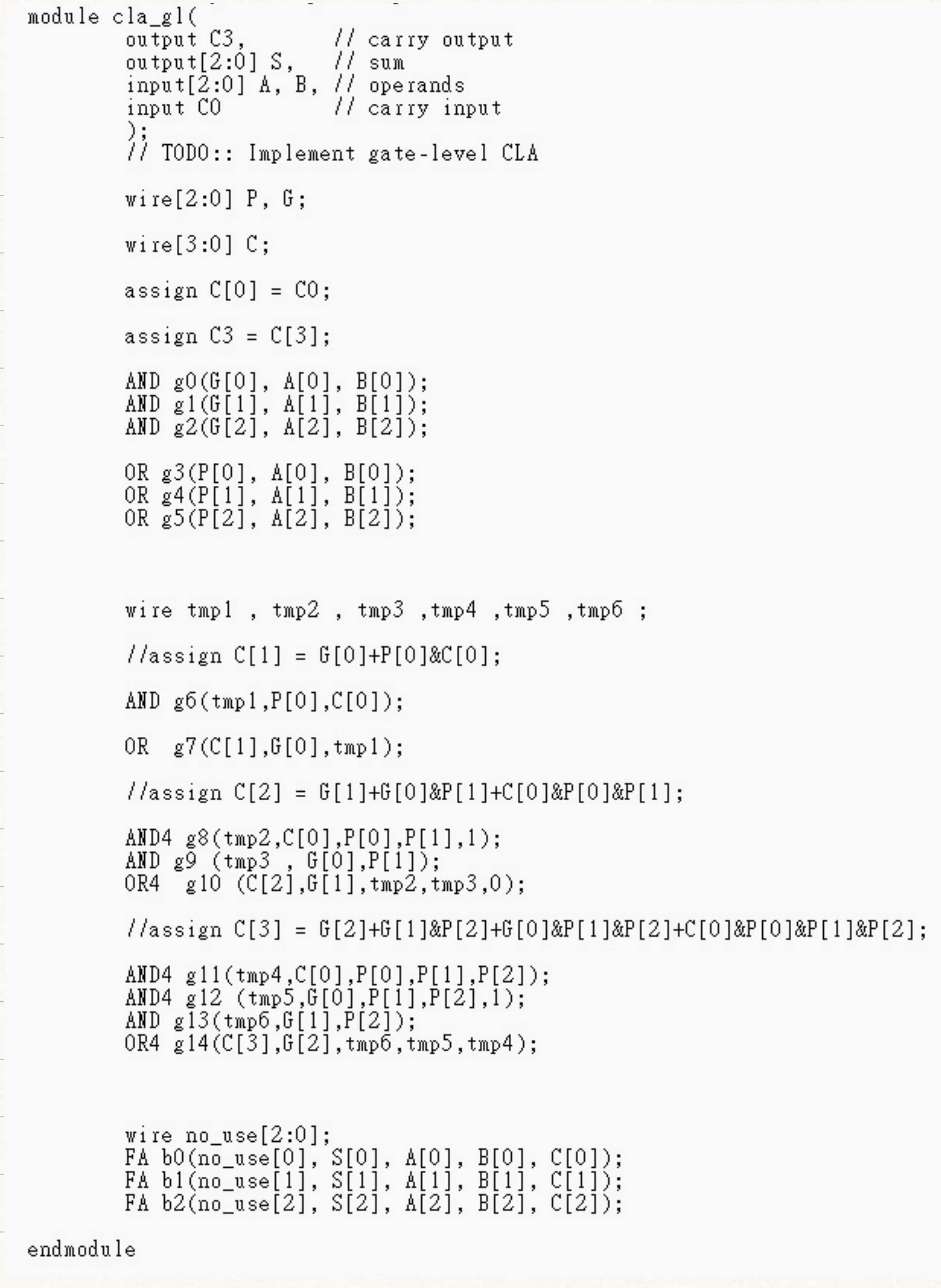
R	H	Q	Q^+
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	x
1	0	1	x
1	1	0	1
1	1	1	1

HQ \ R	0	1
00	0	X
01	0	x
11	1	1
10	0	1

because of 1.

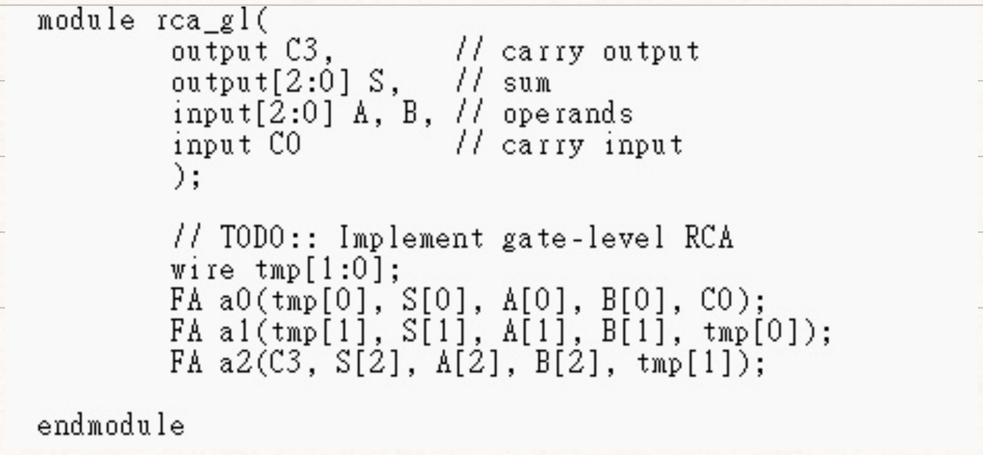
$Q^+ = R + HQ$ #

4(1)



```
module cla_gl(
        output C3,       // carry output
        output[2:0] S,   // sum
        input[2:0] A, B, // operands
        input C0         // carry input
        );
        // TODO:: Implement gate-level CLA

        wire[2:0] P, G;

        wire[3:0] C;

        assign C[0] = C0;

        assign C3 = C[3];

        AND g0(G[0], A[0], B[0]);
        AND g1(G[1], A[1], B[1]);
        AND g2(G[2], A[2], B[2]);

        OR g3(P[0], A[0], B[0]);
        OR g4(P[1], A[1], B[1]);
        OR g5(P[2], A[2], B[2]);



        wire tmp1 , tmp2 , tmp3 ,tmp4 ,tmp5 ,tmp6 ;

        //assign C[1] = G[0]+P[0]&C[0];

        AND g6(tmp1,P[0],C[0]);

        OR  g7(C[1],G[0],tmp1);

        //assign C[2] = G[1]+G[0]&P[1]+C[0]&P[0]&P[1];

        AND4 g8(tmp2,C[0],P[0],P[1],1);
        AND g9 (tmp3 , G[0],P[1]);
        OR4  g10 (C[2],G[1],tmp2,tmp3,0);

        //assign C[3] = G[2]+G[1]&P[2]+G[0]&P[1]&P[2]+C[0]&P[0]&P[1]&P[2];

        AND4 g11(tmp4,C[0],P[0],P[1],P[2]);
        AND4 g12 (tmp5,G[0],P[1],P[2],1);
        AND g13(tmp6,G[1],P[2]);
        OR4 g14(C[3],G[2],tmp6,tmp5,tmp4);



        wire no_use[2:0];
        FA b0(no_use[0], S[0], A[0], B[0], C[0]);
        FA b1(no_use[1], S[1], A[1], B[1], C[1]);
        FA b2(no_use[2], S[2], A[2], B[2], C[2]);

endmodule
```

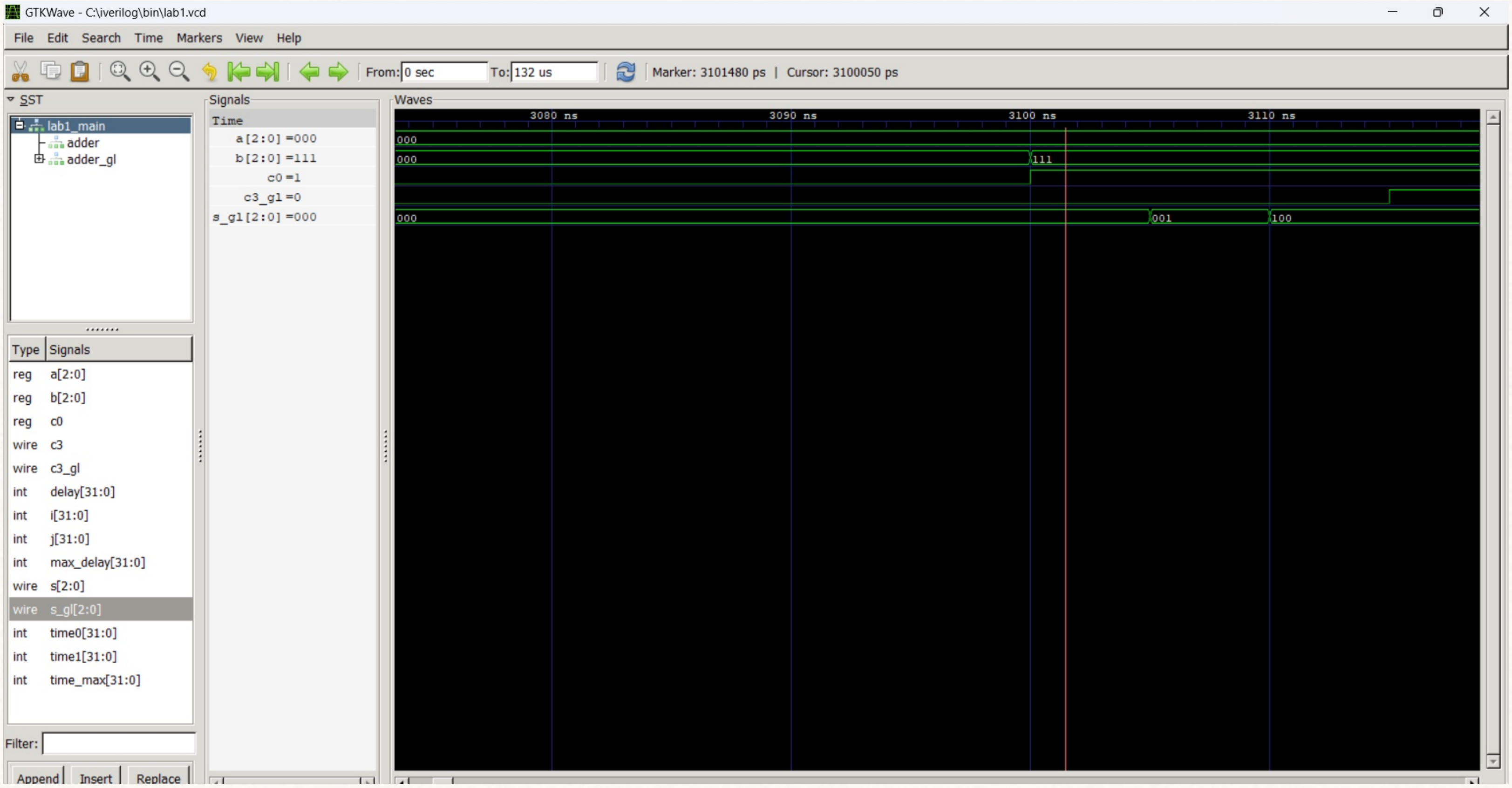
4(2)



```
module rca_gl(
        output C3,       // carry output
        output[2:0] S,   // sum
        input[2:0] A, B, // operands
        input C0         // carry input
        );

        // TODO:: Implement gate-level RCA
        wire tmp[1:0];
        FA a0(tmp[0], S[0], A[0], B[0], C0);
        FA a1(tmp[1], S[1], A[1], B[1], tmp[0]);
        FA a2(C3, S[2], A[2], B[2], tmp[1]);

endmodule
```

5 Lab 1: Waveform (12pts)

Show the waveform of `cla_gl` on input transition from 000 + 000 + 0 to 000 + 111 + 1. You should select all the input and output signals of `cla_gl` module.



GTKWave - C:\iverilog\bin\lab1.vcd

From: 0 sec To: 132 us

Marker: 3101480 ps | Cursor: 3100050 ps

SST

- lab1_main
  - adder
  - adder_gl

Type	Signals
reg	a[2:0]
reg	b[2:0]
reg	c0
wire	c3
wire	c3_gl
int	delay[31:0]
int	i[31:0]
int	j[31:0]
int	max_delay[31:0]
wire	s[2:0]
wire	s_gl[2:0]
int	time0[31:0]
int	time1[31:0]
int	time_max[31:0]

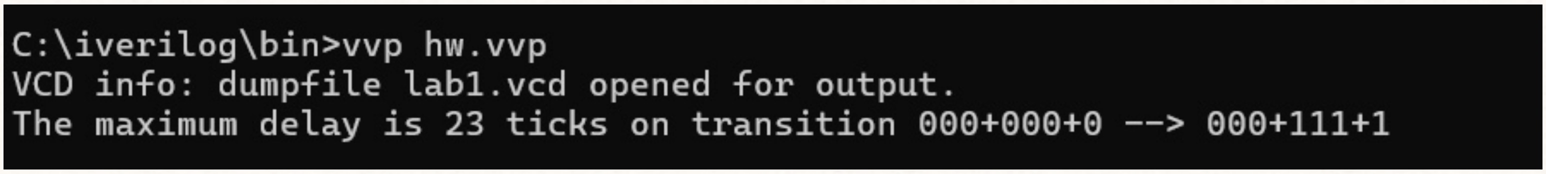
Signals

```
Time
    a[2:0] =000
    b[2:0] =111
        c0 =1
     c3_gl =0
 s_gl[2:0] =000
```

6 Lab 1: Propagation Delays (8pts)

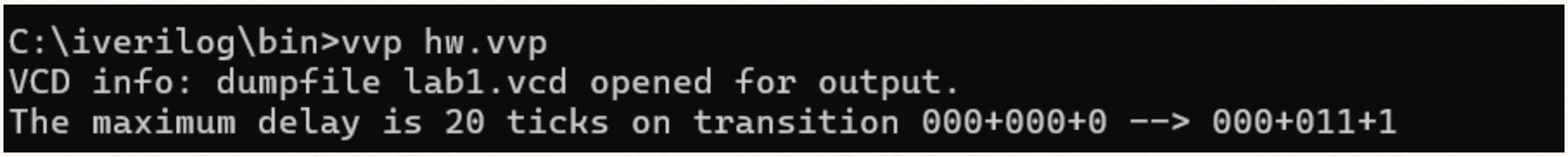
1. (4pts) Find the maximum propagation delay of `rca_gl` and one of the corresponding input transitions.
2. (4pts) Find the maximum propagation delay of `cla_gl` and one of the corresponding input transitions.

1.



```
C:\iverilog\bin>vvp hw.vvp
VCD info: dumpfile lab1.vcd opened for output.
The maximum delay is 23 ticks on transition 000+000+0 --> 000+111+1
```

2.



```
C:\iverilog\bin>vvp hw.vvp
VCD info: dumpfile lab1.vcd opened for output.
The maximum delay is 20 ticks on transition 000+000+0 --> 000+011+1
```

7 Lab 1: Some Derivation (10pts)

Assume that only 2-input gates are used. Derive the number of levels needed in an n-bit carry-lookahead adder as a function of n.

已知 $g_i = A_i + B_i$ 都滿足 2-input : 1 level

$P_i = A_i + B_i$

但 $C_{i+1} = g_i + P_i C_i$

$= g_i + P_i g_{i-1} + P_i P_{i-1} \cdot g_{i-2} + \cdots$ => C_n 要 n+1 的加法

令 $G(i:j) = g_j + P_j g_{j-1} + P_j P_{j-1} g_{j-2} + \cdots (P_j \cdots P_i g_i)$

$P(i:j) = (P_j \cdot P_{j-1} \cdots P_i)$

利用 binary tree 計算

$$\text{需要 2個 gate (level)} \begin{cases} \square = P(k+1:j) \cdot G(i:k) : (\text{AND - gate}) \\ G(i:j) = G(k+1:j) + \square : (\text{OR - gate}) \end{cases}$$

故 $C_n = \underbrace{2}_{\text{2個 gate = 2level}} \cdot \lceil \log_2(n+1) \rceil$

又 $S_i = A_i \oplus B_i \oplus \underbrace{C_i}_{2\lceil \log_2 n \rceil}$ => $2 + 2\lceil \log_2 n \rceil$ 層

故 total 層數 $= \max: \left\{ 2\lceil \log_2(n+1) \rceil + \underset{\text{P.Q}}{\underset{\sim}{1}} \;,\; 2 + 2\lceil \log_2 n \rceil \right\}$ #