Mohammad Amanlou
Student of computer engineering
SID:810100084
Logical circuits(DSD)
Teacher: Dr.Navabi

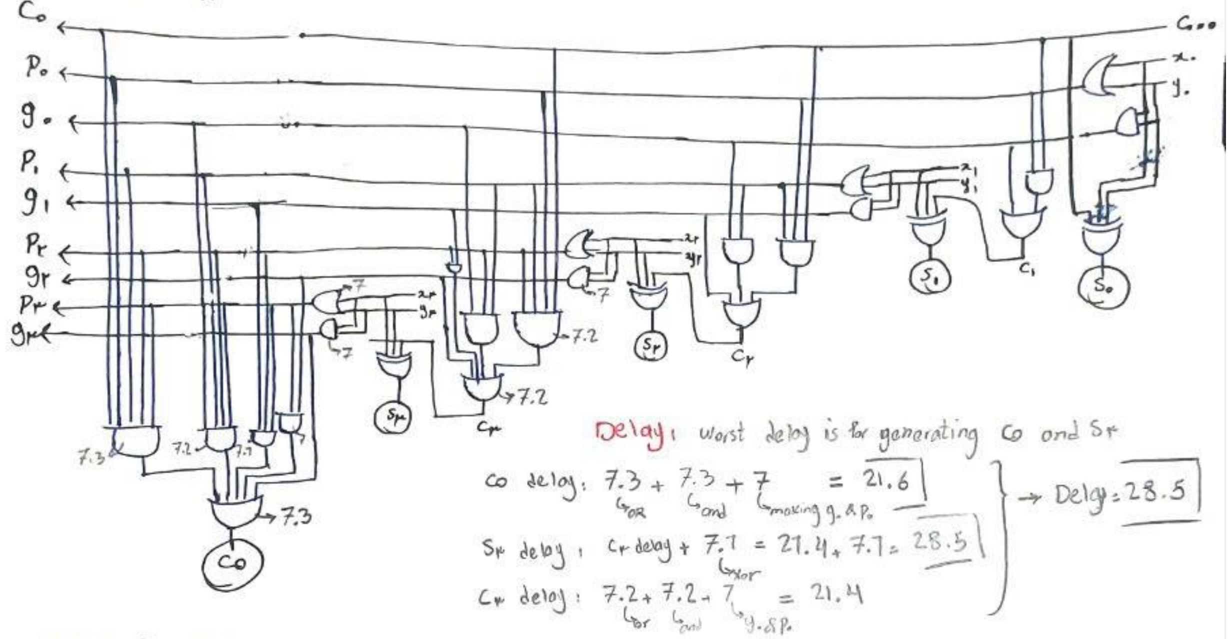
Q1:

The diagram below shows a 4-bit CLA. In addition to that, the delays related to the last bit have been calculated. The reason why it is calculated for the last bit is that the output of the last bit or the most valuable bit has the biggest delay, so to find the worst case delay, the relevant delays for the last bit must be calculated.

Mohammad Amanlou   logical circuits (DSD)   Dr. Zain navabi

SID: 810100084

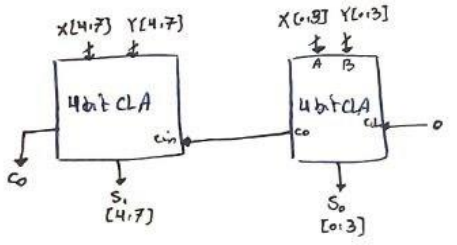
4bit CLA



Delay: worst delay is for generating Co and S_3

$$\left.\begin{array}{l} \text{Co delay: } \underbrace{7.3}_{\text{OR}} + \underbrace{7.3}_{\text{and}} + \underbrace{7}_{\text{making } g_0 \,\&\, P_0} = \boxed{21.6} \\ S_3 \text{ delay: } C_3 \text{ delay} + \underbrace{7.1}_{\text{Xor}} = 21.4 + 7.1 = \boxed{28.5} \\ C_3 \text{ delay: } \underbrace{7.2}_{\text{or}} + \underbrace{7.2}_{\text{and}} + \underbrace{7}_{g_0 \,\&\, P_0} = 21.4 \end{array}\right\} \rightarrow \text{Delay} = \boxed{28.5}$$

8 bit CRA

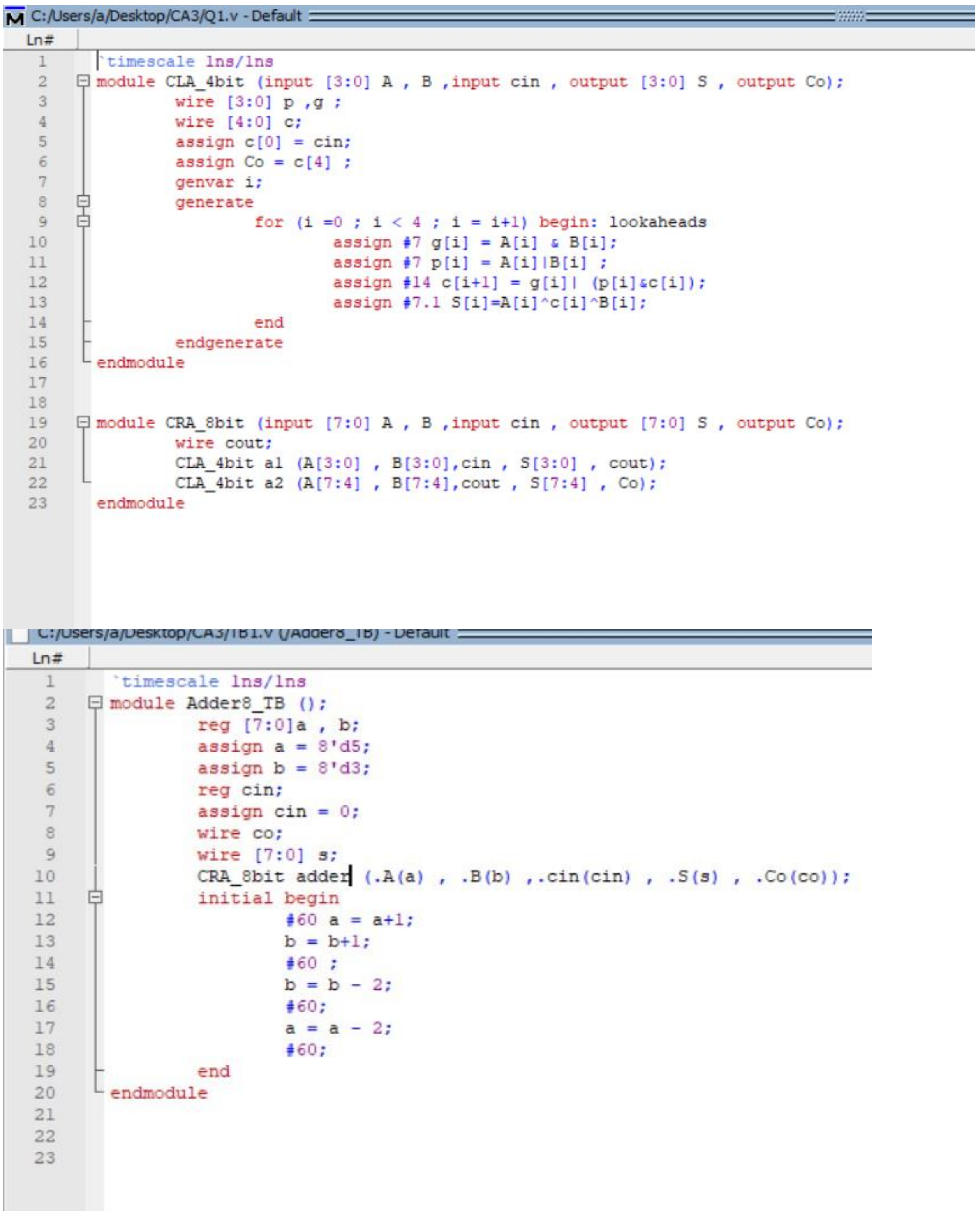


Delay: because of the starting of second CLA is after finishing of first CLA the total worst case delay is:

$2 * 28.5 = \boxed{57}$

- Answer is $\{S_1, S_0\}$ with carry out "Co".

The test bench, Verilog code and wave lines can be seen below:



C:/Users/a/Desktop/CA3/Q1.v - Default

Ln#

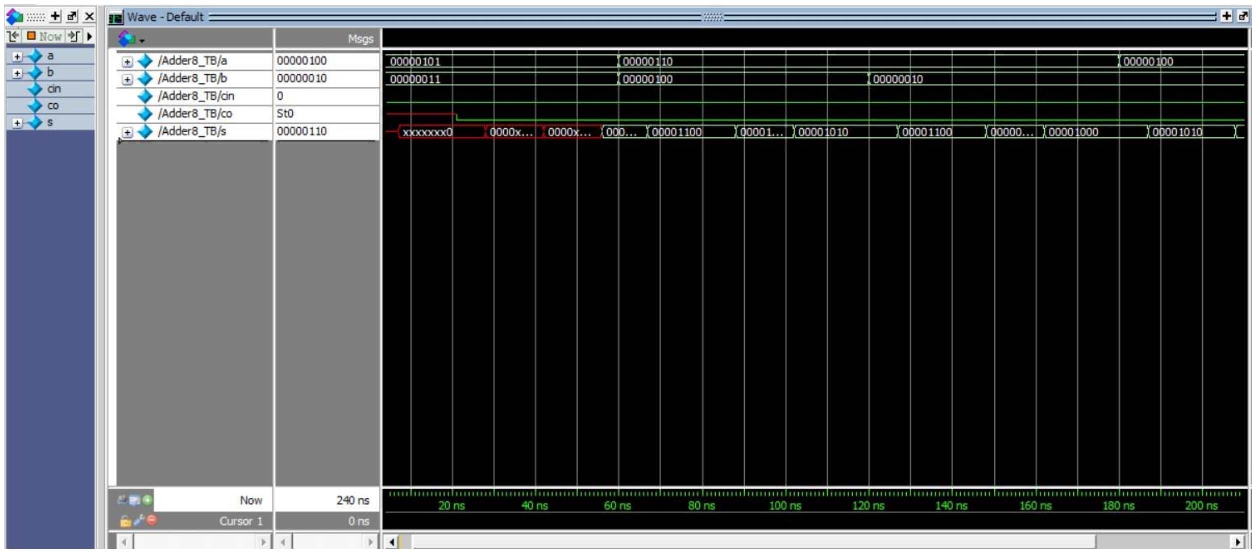
```verilog
`timescale 1ns/1ns
module CLA_4bit (input [3:0] A , B ,input cin , output [3:0] S , output Co);
        wire [3:0] p ,g ;
        wire [4:0] c;
        assign c[0] = cin;
        assign Co = c[4] ;
        genvar i;
        generate
                for (i =0 ; i < 4 ; i = i+1) begin: lookaheads
                        assign #7 g[i] = A[i] & B[i];
                        assign #7 p[i] = A[i]|B[i] ;
                        assign #14 c[i+1] = g[i]| (p[i]&c[i]);
                        assign #7.1 S[i]=A[i]^c[i]^B[i];
                end
        endgenerate
endmodule


module CRA_8bit (input [7:0] A , B ,input cin , output [7:0] S , output Co);
        wire cout;
        CLA_4bit a1 (A[3:0] , B[3:0],cin , S[3:0] , cout);
        CLA_4bit a2 (A[7:4] , B[7:4],cout , S[7:4] , Co);
endmodule
```

C:/Users/a/Desktop/CA3/TB1.v (/Adder8_TB) - Default

Ln#

```verilog
`timescale 1ns/1ns
module Adder8_TB ();
        reg [7:0]a , b;
        assign a = 8'd5;
        assign b = 8'd3;
        reg cin;
        assign cin = 0;
        wire co;
        wire [7:0] s;
        CRA_8bit adder (.A(a) , .B(b) ,.cin(cin) , .S(s) , .Co(co));
        initial begin
                #60 a = a+1;
                b = b+1;
                #60 ;
                b = b - 2;
                #60;
                a = a - 2;
                #60;
        end
endmodule
```



```
Wave - Default                          Msgs
/Adder8_TB/a      00000100     00000101                00000110                                   00000100
/Adder8_TB/b      00000010     00000011                00000100            00000010
/Adder8_TB/cin    0
/Adder8_TB/co     St0
/Adder8_TB/s      00000110     xxxxxxx0  0000x...  0000x...  000...  00001100  00001...  00001010  00001100  00000...  00001000  00001010

Now       240 ns     20 ns  40 ns  60 ns  80 ns  100 ns  120 ns  140 ns  160 ns  180 ns  200 ns
Cursor 1  0 ns
```

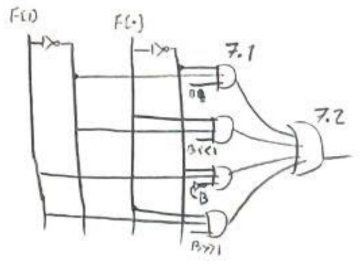
Q3:

Below, an ALU with certain delays is designed with the help of the adder of the previous section and with the help of MUX. The 4 bits on the left are prepared by pre-prepared gates and the 4 gates on the right are designed with the help of shift, matching, then adding and checking the sign to find the smallest number and add two numbers. To find the value of CARRY, for the cases that have it, the value is received from the adder and transferred out. Also, to check whether the numbers are zero or not, all the bits are first compared and then ORed together. Finally, the Verilog code and WAVE Forms have also arrived.

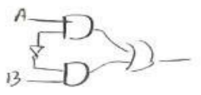
■ ALu

Delay : ● 4to1 Mux :

7.1 + 7.2 = 14.3

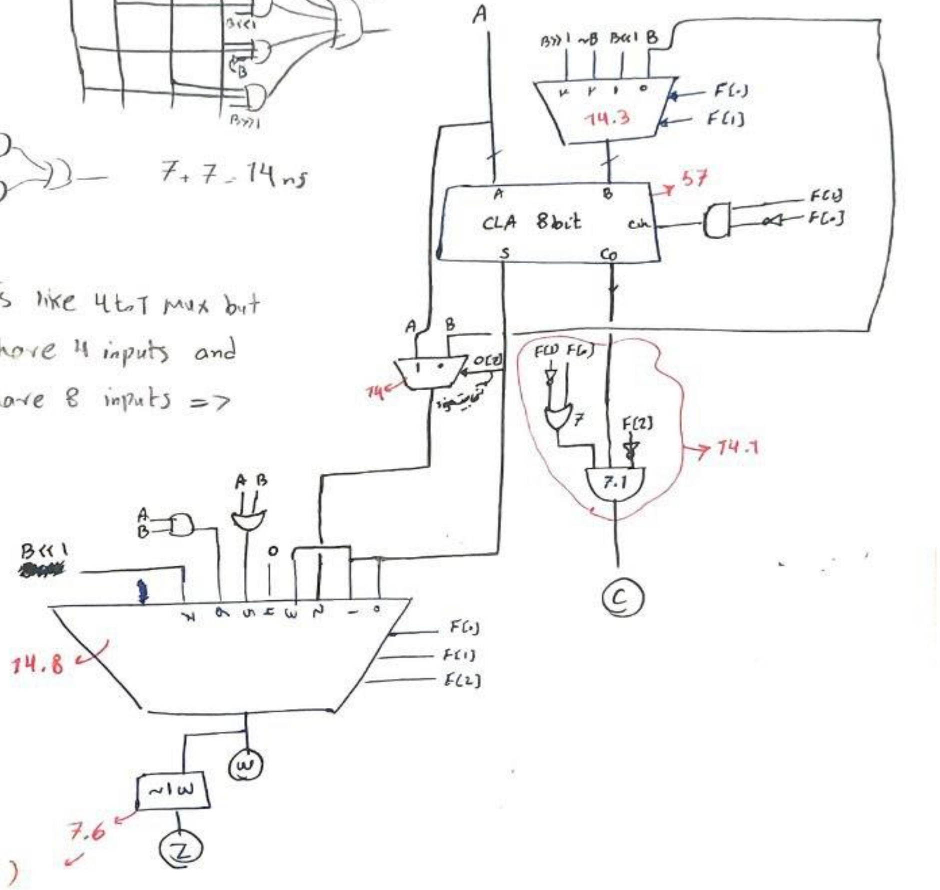


● 2to1 Mux : 7 + 7 = 14 ns



• 8to1 Mux : that is like 4to1 Mux but First And gates have 4 inputs and Final or gates have 8 inputs =>

7.2 + 7.6 = 14.8



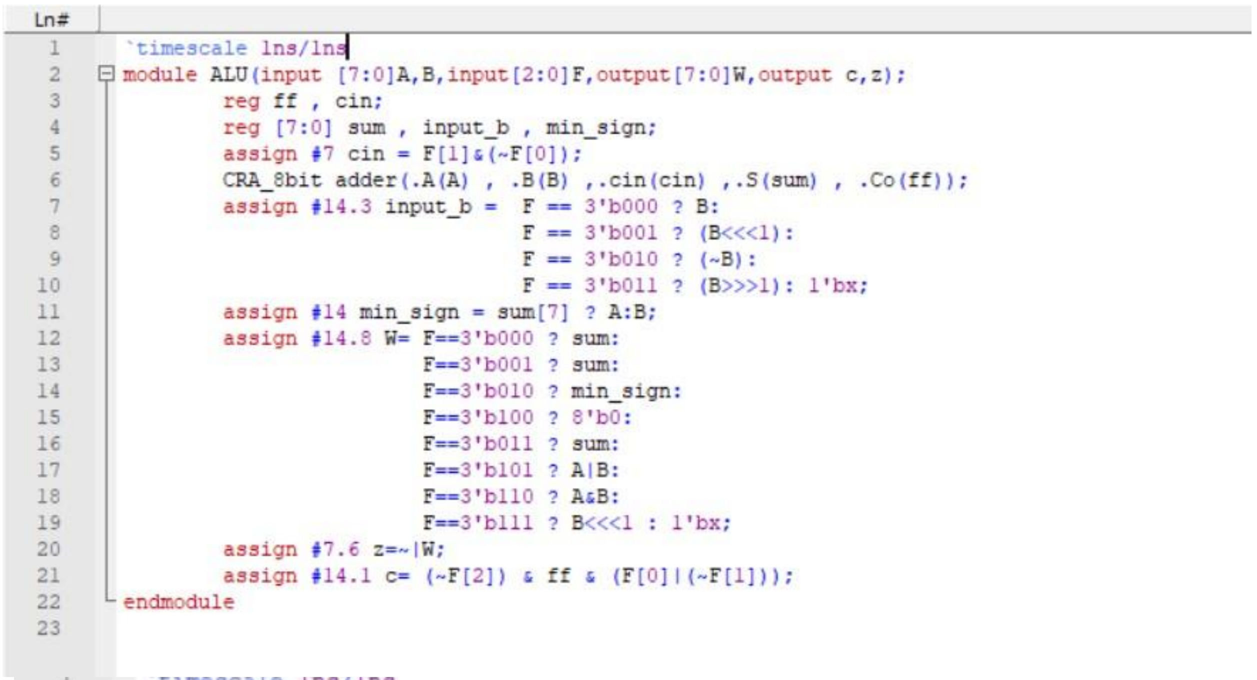
(A 8 input or)

=>

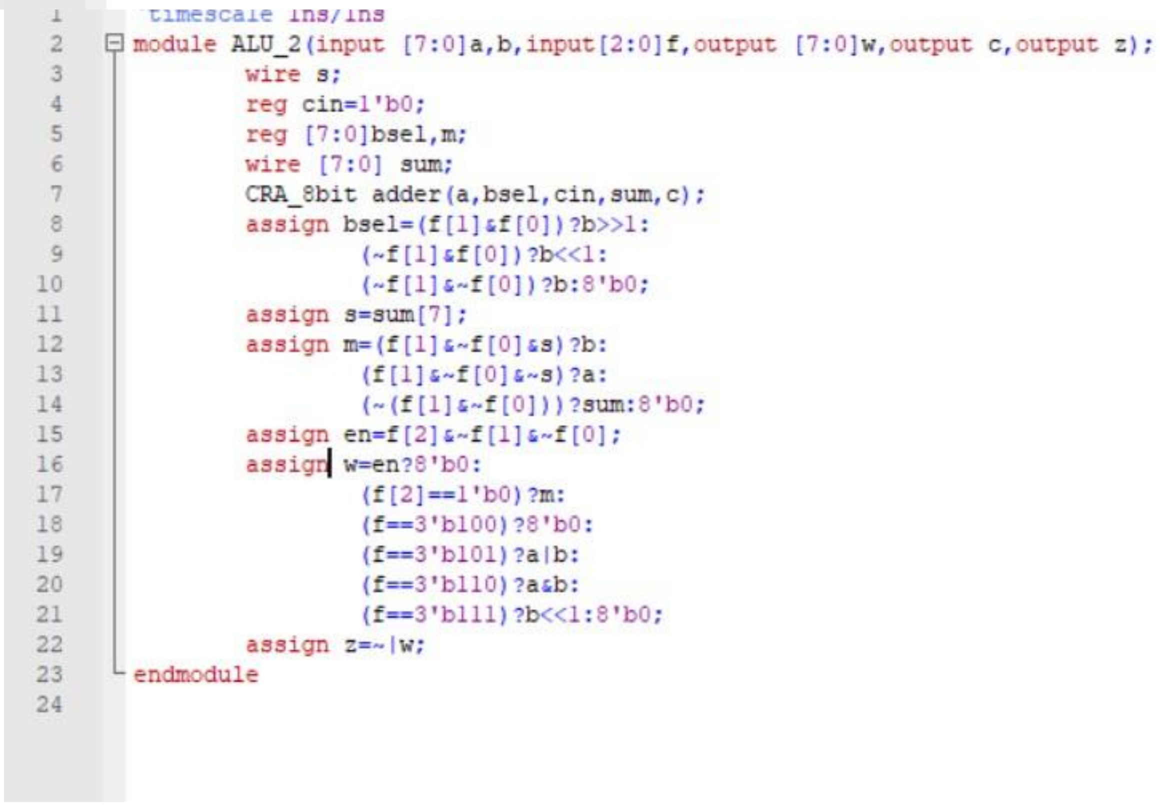
opcode	Function	delay W	delay C	delay Z
000	W = A+B	86.1	85.4	93.7
001	W = A+~B	86.1	86.4	93.7
010	W = Min(A,B)	100.1	86.4	107.7
011	W = A+1 ⊕ B	86.1	86.4	93.7
100	W = 0	14.8	14.1	22.4
101	W = A|B	21.8	14.1	29.4
110	W = A&B	21.8	14.1	29.4
111	W = ~B	14.8	14.1	22.4

→ All delay worst cases

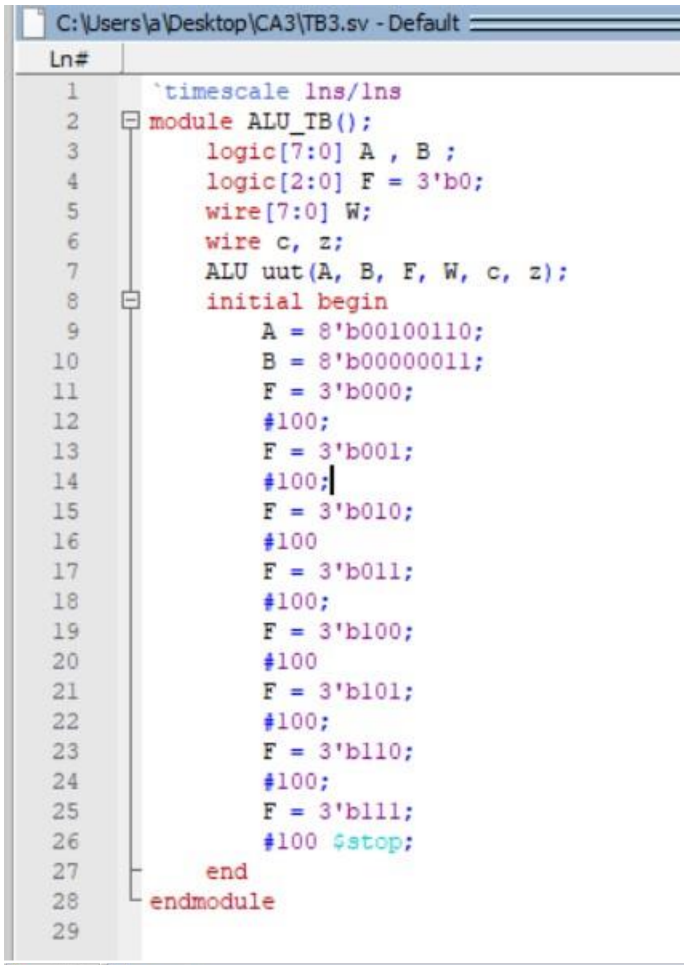
Designed with two ALU methods, ALU_2 does not match the design on paper, but it correctly associates the values



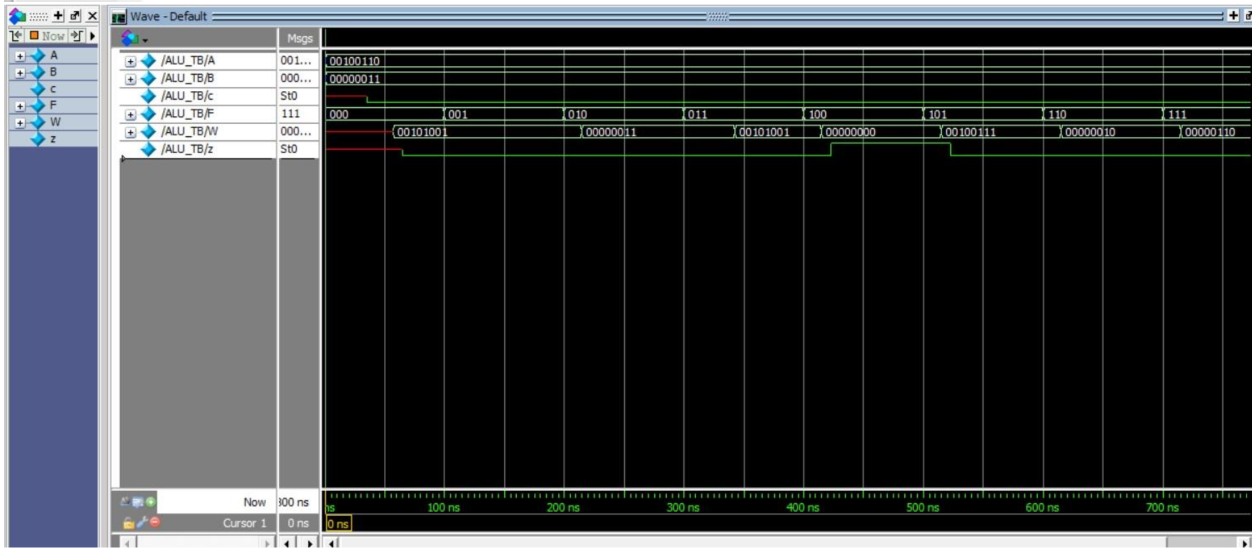
```
`timescale 1ns/1ns
module ALU(input [7:0]A,B,input[2:0]F,output[7:0]W,output c,z);
        reg ff , cin;
        reg [7:0] sum , input_b , min_sign;
        assign #7 cin = F[1]&(~F[0]);
        CRA_8bit adder(.A(A) , .B(B) ,.cin(cin) ,.S(sum) , .Co(ff));
        assign #14.3 input_b =  F == 3'b000 ? B:
                                F == 3'b001 ? (B<<<1):
                                F == 3'b010 ? (~B):
                                F == 3'b011 ? (B>>>1): 1'bx;
        assign #14 min_sign = sum[7] ? A:B;
        assign #14.8 W= F==3'b000 ? sum:
                        F==3'b001 ? sum:
                        F==3'b010 ? min_sign:
                        F==3'b100 ? 8'b0:
                        F==3'b011 ? sum:
                        F==3'b101 ? A|B:
                        F==3'b110 ? A&B:
                        F==3'b111 ? B<<<1 : 1'bx;
        assign #7.6 z=~|W;
        assign #14.1 c= (~F[2]) & ff & (F[0]|(~F[1]));
endmodule
```



```
`timescale 1ns/1ns
module ALU_2(input [7:0]a,b,input[2:0]f,output [7:0]w,output c,output z);
        wire s;
        reg cin=1'b0;
        reg [7:0]bsel,m;
        wire [7:0] sum;
        CRA_8bit adder(a,bsel,cin,sum,c);
        assign bsel=(f[1]&f[0])?b>>1:
                (~f[1]&f[0])?b<<1:
                (~f[1]&~f[0])?b:8'b0;
        assign s=sum[7];
        assign m=(f[1]&~f[0]&s)?b:
                (f[1]&~f[0]&~s)?a:
                (~(f[1]&~f[0]))?sum:8'b0;
        assign en=f[2]&~f[1]&~f[0];
        assign w=en?8'b0:
                (f[2]==1'b0)?m:
                (f==3'b100)?8'b0:
                (f==3'b101)?a|b:
                (f==3'b110)?a&b:
                (f==3'b111)?b<<1:8'b0;
        assign z=~|w;
endmodule
```


C:\Users\a\Desktop\CA3\TB3.sv - Default

```
`timescale 1ns/1ns
module ALU_TB();
    logic[7:0] A , B ;
    logic[2:0] F = 3'b0;
    wire[7:0] W;
    wire c, z;
    ALU uut(A, B, F, W, c, z);
    initial begin
        A = 8'b00100110;
        B = 8'b00000011;
        F = 3'b000;
        #100;
        F = 3'b001;
        #100;
        F = 3'b010;
        #100
        F = 3'b011;
        #100;
        F = 3'b100;
        #100
        F = 3'b101;
        #100;
        F = 3'b110;
        #100;
        F = 3'b111;
        #100 $stop;
    end
endmodule
```



```
/ALU_TB/A   001...  00100110
/ALU_TB/B   000...  00000011
/ALU_TB/c   St0
/ALU_TB/F   111     000       001       010       011       100       101       110       111
/ALU_TB/W   000...    00101001          00000011      00101001  00000000  00100111  00000010  00000110
/ALU_TB/z   St0
```