

Name: Xinyu Hadrian Hu, and Duan Wei Zhang

Student Number: 500194233, and 500824903

Date: June 26, 2020

COE 608: Computer Architecture and Design

COE 608: Lab 3, Part 1 Report

Purpose

The purpose of this lab is to generate the components of a 32-bit ALU, with six different operations. The addition and subtraction units must be done in structural form of VHDL.

Design and Implementation

The goal of the 32-bit ALU is to perform the following operations:

32-bit ALU				
OP Name	Neg/Tsel	ALU-Select	Operation Performed	
AND (logical)	0	0 0	Result $\leq a \text{ AND } b$	
OR (logical)	0	0 1	Result $\leq a \text{ OR } b$	
ADD	0	1 0	Result $\leq a + b$	
SUB	1	1 0	Result $\leq a - b$	
ROL	1	0 0	Result $\leq a \ll 1$	
ROR	1	0 1	Result $\leq a \gg 1$	

Figure 1: ALU Operations

The first step is to create the adder, since it is not possible to use behavioral model in this lab to create the adder circuit.

Full Adder Truth Table						
A	B	Cin		Sum		Cout
0	0	0		0		0
0	0	1		1		0
0	1	0		1		0
0	1	1		0		1
1	0	0		1		0
1	0	1		0		1
1	1	0		0		1
1	1	1		1		1
			Sum	A xor B xor C		
			Cout	(A and (B or C)) or (B and C)		

Figure 2: Full Adder Circuit with logical functions

Next, the operation of the two-to-one multiplexer is illustrated below:

2 to 1 Multiplexer						
A	B	X	A and not X	B and X		A and not X or (B and X)
0	0	0	0	0		0
0	0	1	0	0		0
0	1	0	0	0		0
0	1	1	0	1		1
1	0	0	1	0		1
1	0	1	0	0		0
1	1	0	0	0		0
1	1	1	0	1		1

Figure 3: Two-to-one multiplexer

Next, the binary-coded-decimal to seven-segment converter is shown below. It is assumed that the seven-segment display is cathode-active.

BCD to 7 segment display		
BCD Code	7seg Code	Symbol
0 0 0 0	0 0 0 0 0 0 1	0
0 0 0 1	1 0 0 1 1 1 1	1
0 0 1 0	0 0 1 0 0 1 0	2
0 0 1 1	0 0 0 0 1 1 0	3
0 1 0 0	1 0 0 1 1 0 0	4
0 1 0 1	0 1 0 0 1 0 0	5
0 1 1 0	0 1 0 0 0 0 0	6
0 1 1 1	0 0 0 1 1 1 1	7
1 0 0 0	0 0 0 0 0 0 0	8
1 0 0 1	0 0 0 0 1 0 0	9
1 0 1 0	0 0 0 1 0 0 0	A
1 0 1 1	1 1 0 0 0 0 0	B
1 1 0 0	0 1 1 0 0 0 1	C
1 1 0 1	1 0 0 0 0 1 0	D
1 1 1 0	0 1 1 0 0 0 0	E
1 1 1 1	0 1 1 1 0 0 0	F

Figure 4: BCD to 7 segment display Truth Table

One-bit Adder: Functional and Timing Waveforms

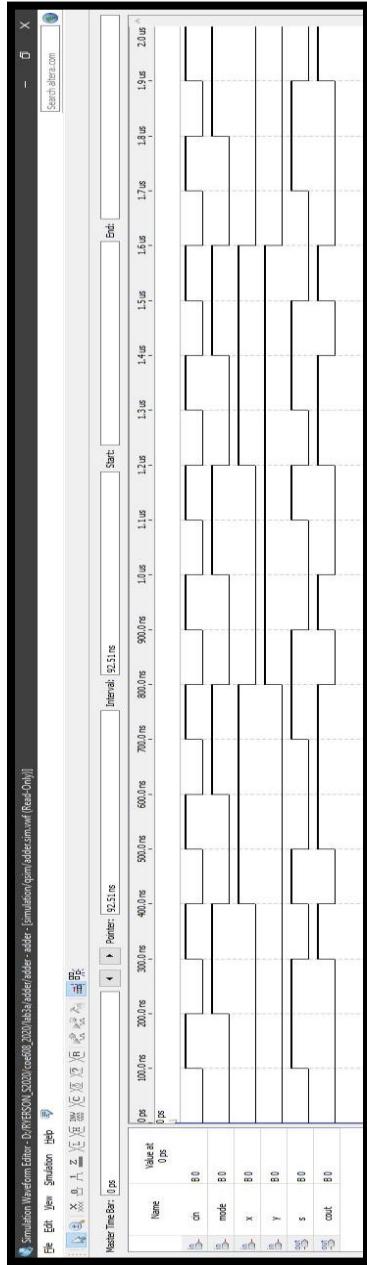


Figure 5: 1-bit Adder Functional Waveform

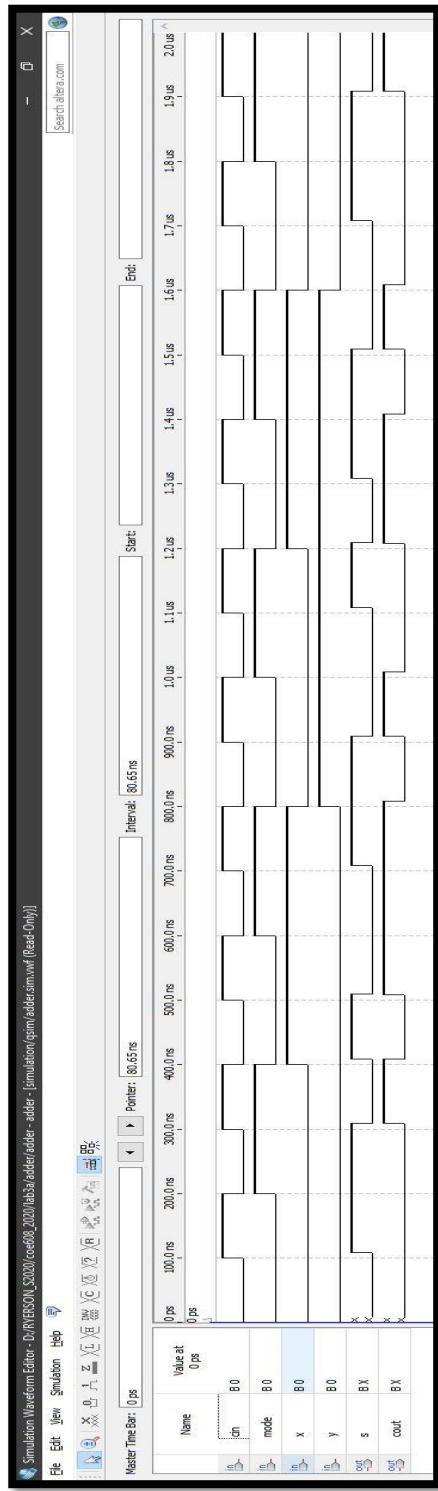


Figure 6: 1-bit Adder Timing Waveform

32-bit Adder: Functional and Timing Waveforms

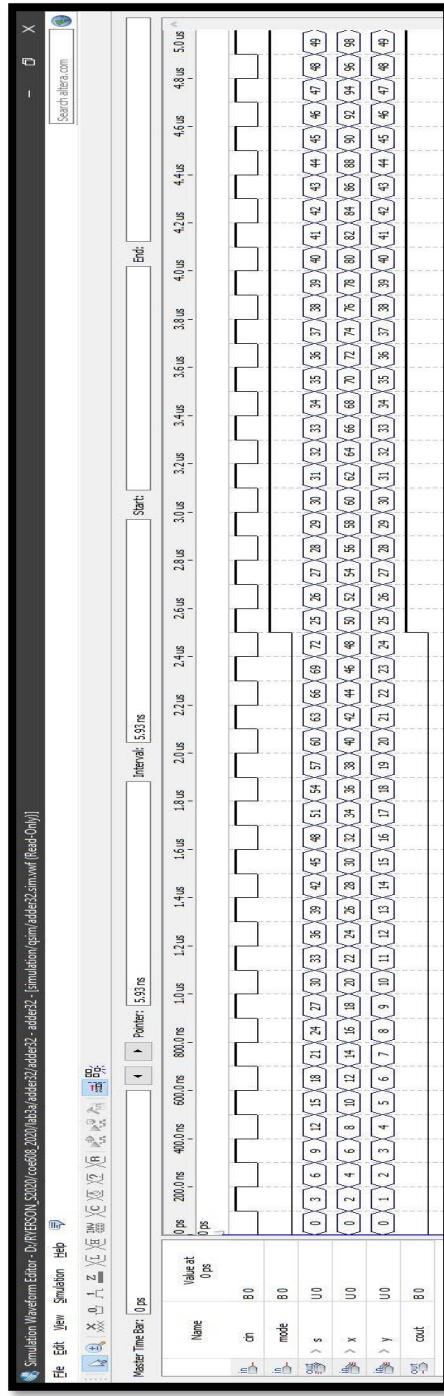


Figure 7: 32-bit Adder Functional Waveform

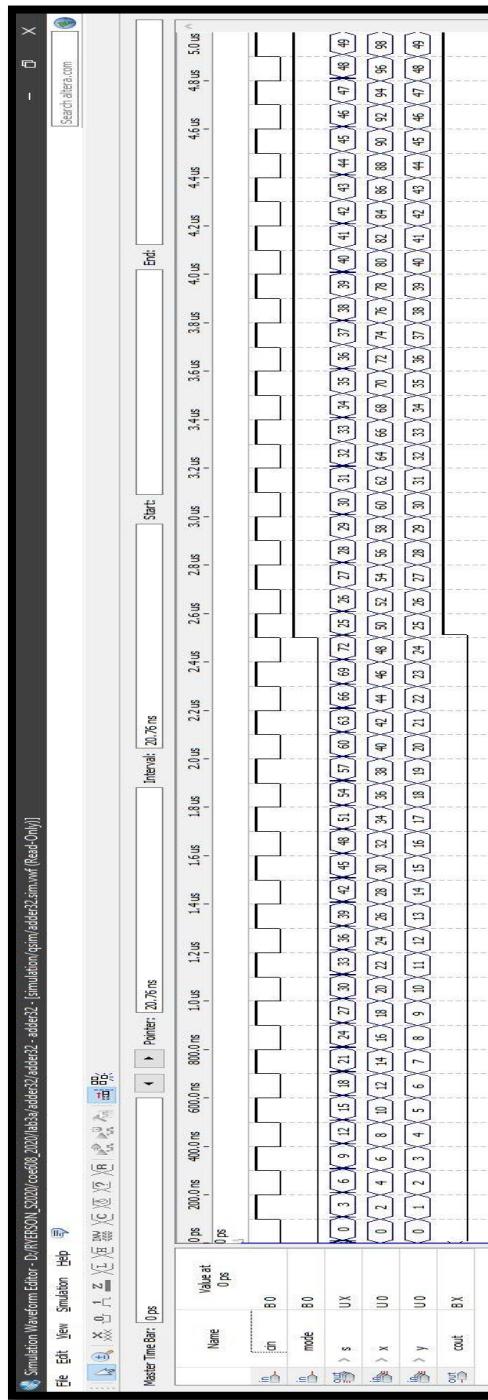


Figure 8: 32-bit Adder Timing Waveform

2 to 1 Multiplexer: Functional and Timing Waveforms

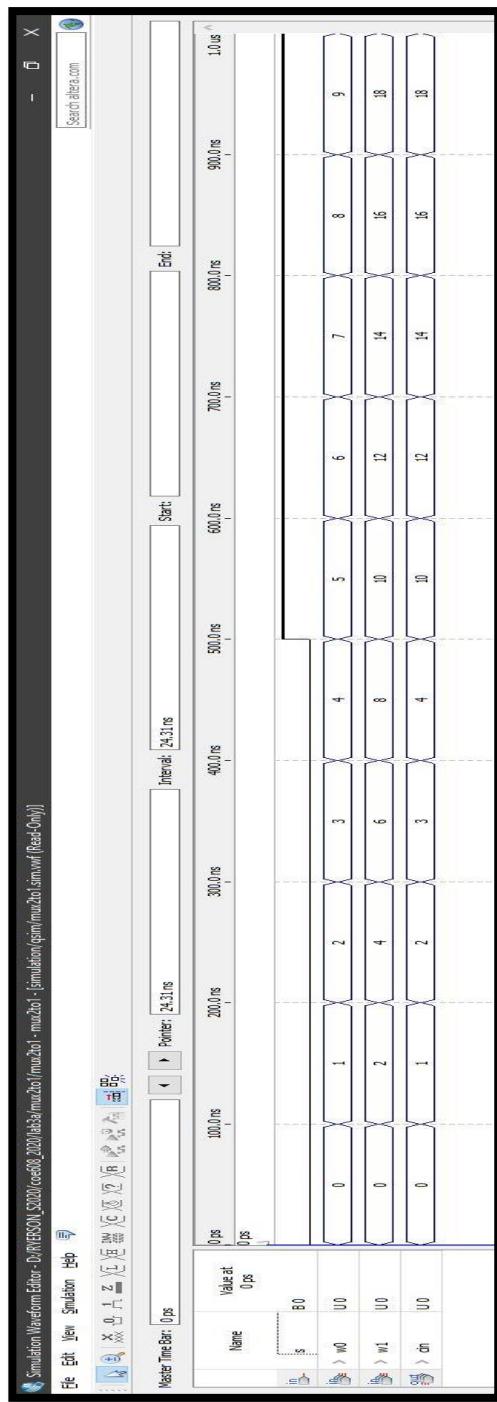


Figure 9: 2-to1 Multiplexer Functional Waveform

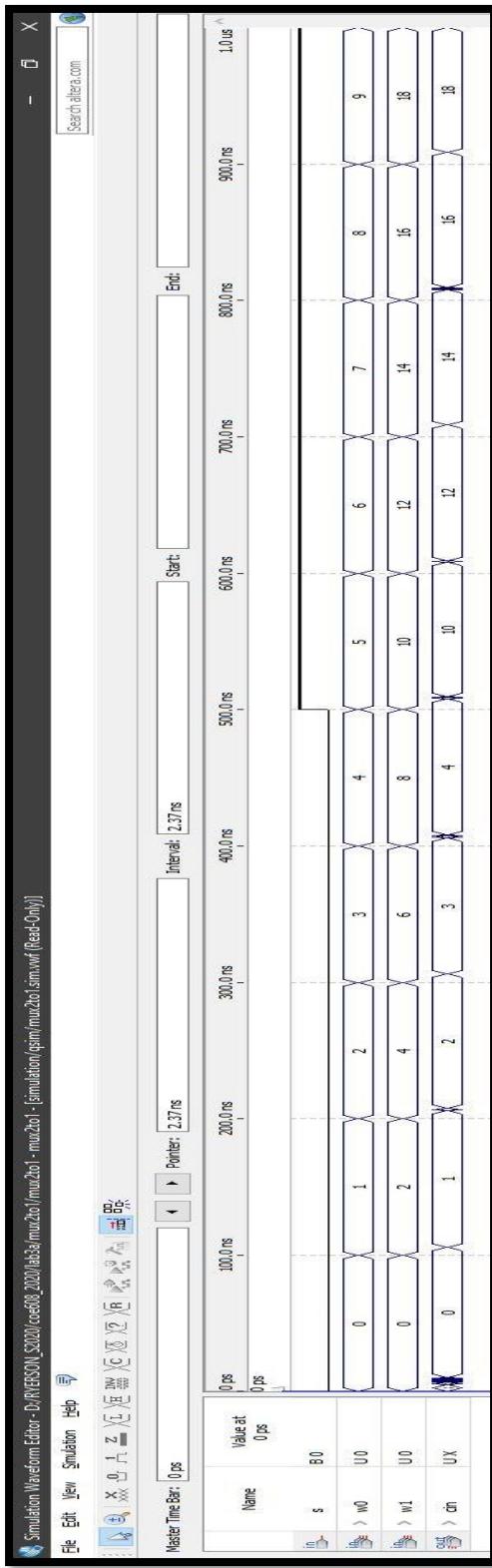


Figure 10: 2-to-1 Multiplexer Timing Waveform

32-bit ALU: Functional and Timing Waveforms

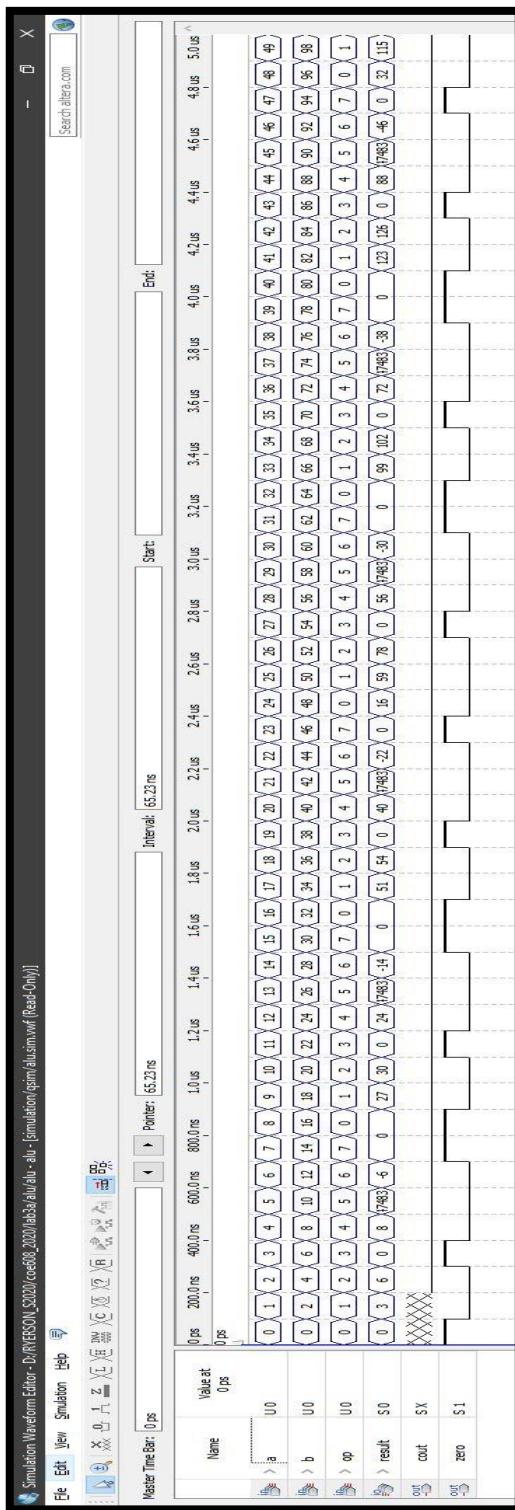


Figure 11: 32-bit ALU Functional Waveform

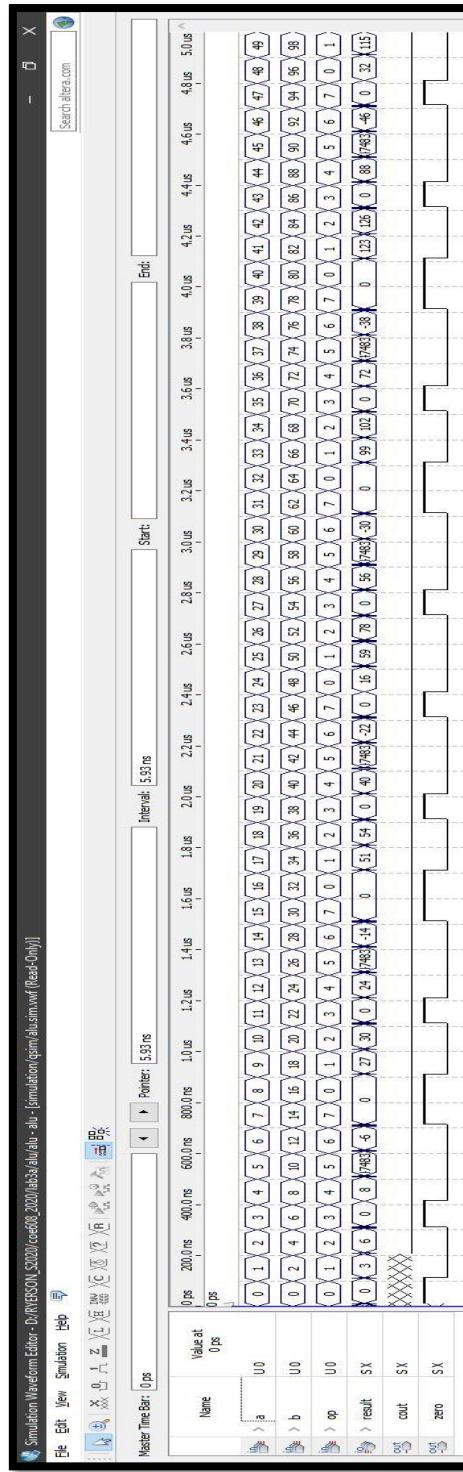


Figure 12: 32-bit ALU Timing Waveform

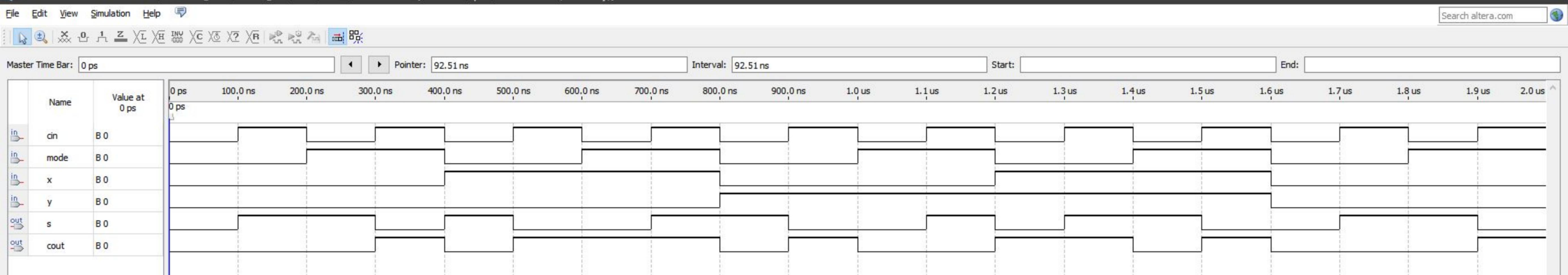
Discussions and Conclusions

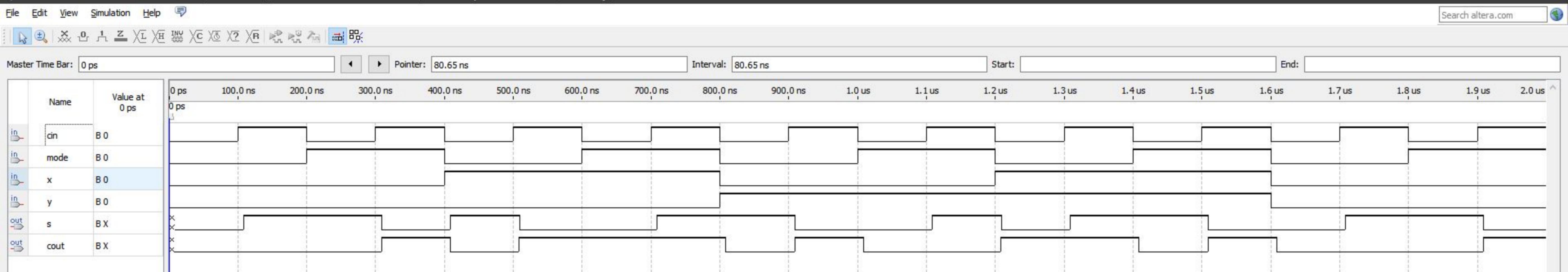
All of the above devices work as expected. The worst-case delay of the 32-bit ALU is 1 ns, for most outputs. The longest delay occurs at the positive or negative signs of the output. The worst delay for the negative output symbol is 40 ns.

Appendices: VHDL Codes for the ALU and other Supporting Devices

The files are attached as PDF to make this document easier to read.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity adder is
7     port (
8         x, y: in std_logic;
9         mode: in std_logic;
10        cin : in std_logic;
11        cout : out std_logic;
12        s : out std_logic);
13 end adder;
14
15 architecture behavior of adder is
16     signal temp : std_logic;
17 begin
18     process(mode)
19     begin
20         if (mode = '1') then
21             temp <= (y xor mode);
22         else
23             temp <= y;
24         end if;
25         s <= (x xor temp xor cin) or (x and temp and cin);
26         cout <= (x and temp) or (x and cin) or (cin and temp);
27     end process;
28 end behavior;
29
30
31
```





```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  ENTITY adder32 IS
7      PORT(cin      :IN STD_LOGIC;
8            mode     :IN STD_LOGIC; -- 1 = sub , 0 = add
9            x        : in std_logic_vector(31 downto 0);
10           y        : in std_logic_vector(31 downto 0);
11           s        : out std_logic_vector(31 downto 0);
12           cout    :OUT STD_LOGIC
13       );
14 end adder32;
15
16 ARCHITECTURE description of adder32 IS
17     SIGNAL c31,c30,c29,c28,c27,c26,c25,c24,c23,c22,c21,c20,c19,c18,c17,c16,c15,c14,c13,c12,
18 c11,c10,c9,c8,c7,c6,c5,c4,c3,c2,c1,c0 : STD_LOGIC;
19     COMPONENT adder
20         port(
21             x : in std_logic;
22             y : in std_logic;
23             mode : in std_logic;
24             cin : in std_logic;
25             cout : out std_logic;
26             s : out std_logic
27         );
28     END COMPONENT;
29 BEGIN
30     --st0: adder port map (x(0), y(0), mode, cin, c1, s(0));
31     st0: adder port map (x(0), y(0), mode, mode, c1, s(0));
32     st1: adder port map (x(1), y(1), mode, c1, c2, s(1));
33     st2: adder port map (x(2), y(2), mode, c2, c3, s(2));
34     st3: adder port map (x(3), y(3), mode, c3, c4, s(3));
35     st4: adder port map (x(4), y(4), mode, c4, c5, s(4));
36     st5: adder port map (x(5), y(5), mode, c5, c6, s(5));
37     st6: adder port map (x(6), y(6), mode, c6, c7, s(6));
38     st7: adder port map (x(7), y(7), mode, c7, c8, s(7));
39     st8: adder port map (x(8), y(8), mode, c8, c9, s(8));
40     st9: adder port map (x(9), y(9), mode, c9, c10, s(9));
41     st10: adder port map (x(10), y(10), mode, c10, c11, s(10));
42     st11: adder port map (x(11), y(11), mode, c11, c12, s(11));
43     st12: adder port map (x(12), y(12), mode, c12, c13, s(12));
44     st13: adder port map (x(13), y(13), mode, c13, c14, s(13));
45     st14: adder port map (x(14), y(14), mode, c14, c15, s(14));
46     st15: adder port map (x(15), y(15), mode, c15, c16, s(15));
47     st16: adder port map (x(16), y(16), mode, c16, c17, s(16));
48     st17: adder port map (x(17), y(17), mode, c17, c18, s(17));
49     st18: adder port map (x(18), y(18), mode, c18, c19, s(18));
50     st19: adder port map (x(19), y(19), mode, c19, c20, s(19));
51     st20: adder port map (x(20), y(20), mode, c20, c21, s(20));
52     st21: adder port map (x(21), y(21), mode, c21, c22, s(21));
53     st22: adder port map (x(22), y(22), mode, c22, c23, s(22));
54     st23: adder port map (x(23), y(23), mode, c23, c24, s(23));
55     st24: adder port map (x(24), y(24), mode, c24, c25, s(24));
56     st25: adder port map (x(25), y(25), mode, c25, c26, s(25));
57     st26: adder port map (x(26), y(26), mode, c26, c27, s(26));
58     st27: adder port map (x(27), y(27), mode, c27, c28, s(27));
59     st28: adder port map (x(28), y(28), mode, c28, c29, s(28));
60     st29: adder port map (x(29), y(29), mode, c29, c30, s(29));
61     st30: adder port map (x(30), y(30), mode, c30, c31, s(30));
62     st31: adder port map (x(31), y(31), mode, c31, cout, s(31));

```

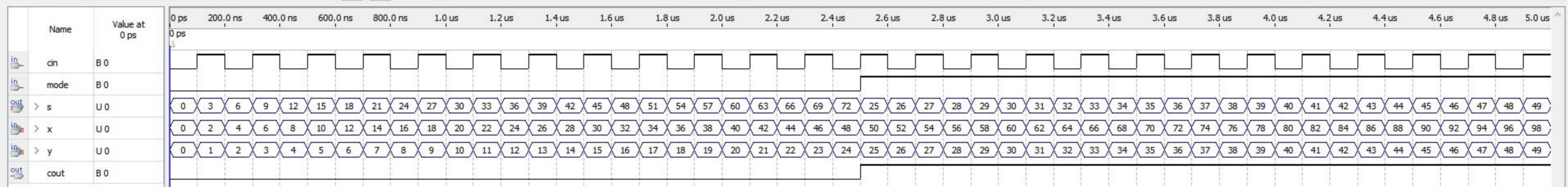
```
62      end description;  
63
```

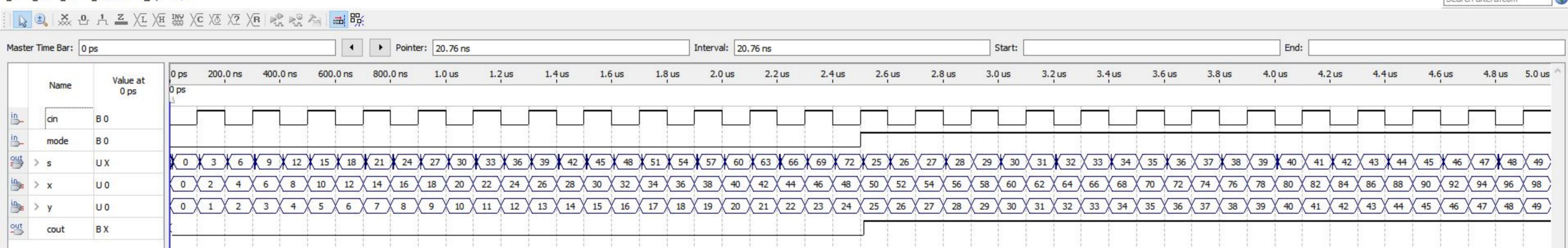
File Edit View Simulation Help

Search altera.com



Master Time Bar: 0 ps Pointer: 5.93 ns Interval: 5.93 ns Start: End:





```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5  use ieee.numeric_std.all;
6
7  entity alu is
8    port(
9      a, b: in std_logic_vector(31 downto 0);
10     op: in std_logic_vector(2 downto 0);
11     result: inout std_logic_vector(31 downto 0);
12     cout, zero: out std_logic);
13 end alu;
14
15 architecture description of alu is
16   component adder32 is
17     port(
18       cin, mode: in std_logic;
19       x, y: in std_logic_vector(31 downto 0);
20       s : out std_logic_vector(31 downto 0);
21       cout: out std_logic);
22   end component adder32;
23   signal adddd: std_logic_vector(31 downto 0);
24   signal coutplus: std_logic;
25   signal shift: std_logic_vector(31 downto 0);
26   begin
27     muxadd: adder32 port map (op(2), op(2), a, b, adddd, coutplus);
28     process (op)
29     begin
30       if op = "000" then
31         result <= a and b;
32       elsif op = "001" then
33         result <= a or b;
34       elsif op = "010" then
35         result <= adddd;
36         cout <= coutplus;
37       elsif op = "110" then
38         result <= adddd;
39         cout <= coutplus;
40       elsif op = "100" then
41         shift(0) <= a(31);
42         shift(31 downto 1) <= a(30 downto 0);
43         result <= shift;
44       elsif op = "101" then
45         shift(31) <= a(0);
46         shift(30 downto 0) <= a(31 downto 1);
47         result <= shift;
48     else
49       result <= (others => '0');
50     end if;
51
52     if result = "00000000000000000000000000000000" then
53       zero <= '1';
54     else
55       zero <= '0';
56     end if;
57
58   end process;
59
60 end description;
```

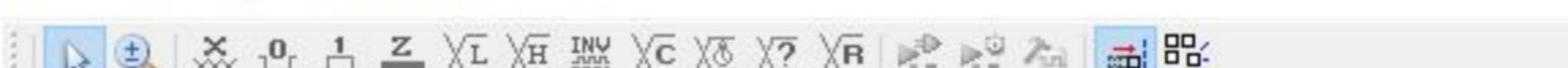
File Edit View Simulation Help Search altova.com



Simulation Waveform Editor - D:/RYERSON_S2020/coe608_2020/lab3a/alu/alu - alu - [simulation/qsim/alu.sim.vwf (Read-Only)]

File Edit View Simulation Help

Search



Master Time Bar: 0 ps ▶◀▶ Pointer: 5.93 ns Interval: 5.93 ns Start: End:

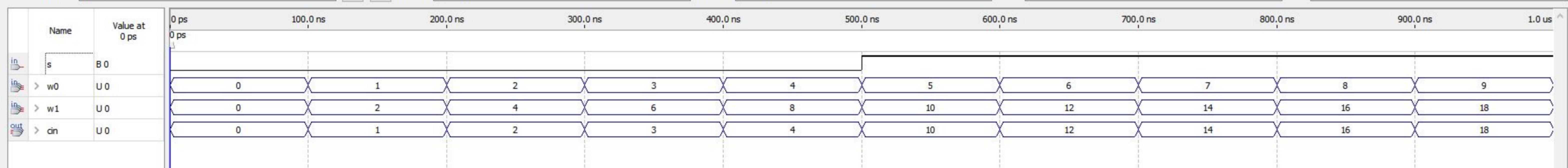
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5 entity mux2too1 is
6 port(
7     st : in std_logic;
8     w0,w1 : in std_logic;
9     cin : out std_logic
10    );
11 end mux2too1;
12
13 architecture description of mux2too1 is begin
14 --if (st = '0') then -- st if start
15 --    --cin <= '0';
16 --elsif(st = '1')then
17 --    --cin <= '1';
18 --end if; --st if end
19 end description;
20
```

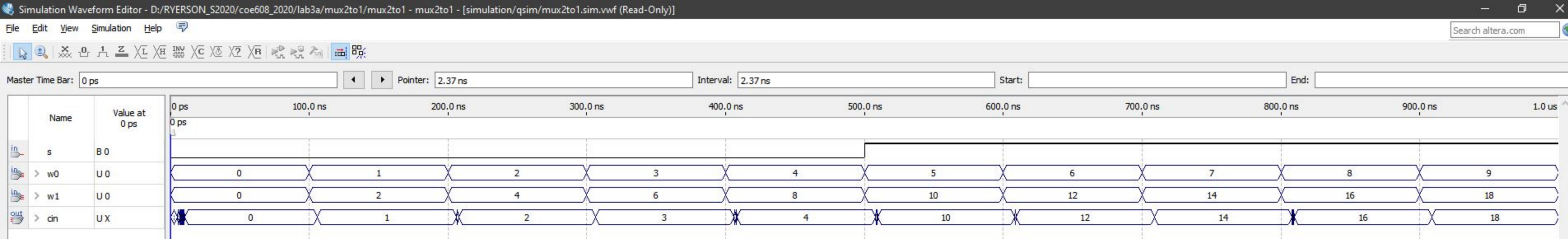
File Edit View Simulation Help

Search altera.com



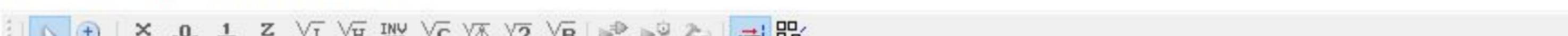
Master Time Bar: 0 ps Pointer: 24.31 ns Interval: 24.31 ns Start: End:





```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity two_bit_asu is
7     port (
8         a, b: in std_logic_vector(1 downto 0);
9         add_sub: in std_logic;
10        result: out std_logic_vector(1 downto 0));
11    end two_bit_asu;
12
13 architecture behavior of two_bit_asu is
14 begin
15     process(a,b,add_sub)
16     begin
17         if add_sub = '0' then --adding
18             result <= a or b;
19         elsif add_sub = '1' then --subtracting
20             result <= (a or (not b))+ 1;
21         end if;
22     end process;
23 end behavior;
```

File Edit View Simulation Help

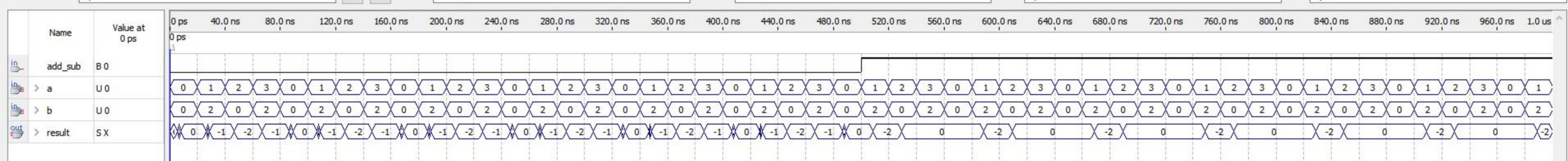


File Edit View Simulation Help

Search altera.com



Master Time Bar: 0 ps Pointer: 13.05 ns Interval: 13.05 ns Start: 0 ps End: 0 ps



Name: Xinyu Hadrian Hu, and Duan Wei Zhang

Student Number: 500194233, and 500824903

Date: June 26, 2020

COE 608: Computer Architecture and Design

COE 608: Lab 3, Part 2 Report

Purpose

The purpose of this lab is to generate the components of an 8-bit ALU, with six different operations. The addition and subtraction units must be done in structural form of VHDL. We also have to include the seven-segment display driver into the build for this ALU. This is a simplified version of the 32-bit ALU for testing before implementing the 32-bit ALU.

Design and Implementation

The following: Figure 1, Figure 2, Figure 3, and Figure 4 are the 8-bit ALU, operational inverter, BCD to seven-segment display driver, and the 8-bit register codes needed for the function of the 8-bit ALU. The truth table for the 8-bit ALU is from the waveforms generated in Quartus II. The values of the 8-bit register are placed as HEX values for the input and output of d and Q, respectively, otherwise the table will be unreasonably large and unhelpful to the reader. There is no need to show the 8-bit adder, since the operations of the 8-bit adder is the same as those of the 32-bit and 1-bit adders from the previous lab 3a.

8-bit ALU				
OP Name	Neg/Tsel	ALU-Select	Operation Performed	
AND (logical)	0	0 0	Result $\leq a \text{ AND } b$	
OR (logical)	0	0 1	Result $\leq a \text{ OR } b$	
ADD	0	1 0	Result $\leq a + b$	
SUB	1	1 0	Result $\leq a - b$	
ROL	1	0 0	Result $\leq a \ll 1$	
ROR	1	0 1	Result $\leq a \gg 1$	

Figure 1: 8-bit ALU Truth Table

Op Inverter	
OP	not OP
0	1
1	0

Figure 2: Op Inverter Truth Table

BCD to 7 segment display		
BCD Code	7seg Code	Symbol
0 0 0 0	0 0 0 0 0 0 1	0
0 0 0 1	1 0 0 1 1 1 1	1
0 0 1 0	0 0 1 0 0 1 0	2
0 0 1 1	0 0 0 0 1 1 0	3
0 1 0 0	1 0 0 1 1 0 0	4
0 1 0 1	0 1 0 0 1 0 0	5
0 1 1 0	0 1 0 0 0 0 0	6
0 1 1 1	0 0 0 1 1 1 1	7
1 0 0 0	0 0 0 0 0 0 0	8
1 0 0 1	0 0 0 0 1 0 0	9
1 0 1 0	0 0 0 1 0 0 0	A
1 0 1 1	1 1 0 0 0 0 0	B
1 1 0 0	0 1 1 0 0 0 1	C
1 1 0 1	1 0 0 0 0 1 0	D
1 1 1 0	0 1 1 0 0 0 0	E
1 1 1 1	0 1 1 1 0 0 0	F

Figure 3: BCD to 7-segment display converter Truth Table

8 bit register					
clk		clr		ld	
0		0		1	0 0
1		0		1	0 1
0		0		1	0 2
1		0		1	0 3
0		0		1	0 4
1		0		1	0 5
0		0		1	0 6
1		0		1	0 7
0		0		1	0 8
1		0		1	0 9
0		0		1	0A
1		0		1	0B
0		0		1	0C
1		0		1	0D
0		0		1	0E
1		0		1	0F
0		0		1	1 0
1		0		1	1 1
0		0		1	1 2
1		0		1	1 3
0		0		1	1 4
1		0		1	1 5
0		0		1	1 6
1		0		1	1 7
0		0		1	1 8
1		1		1	1 9
0		1		1	2A
1		1		1	2B
0		1		1	2C
1		1		1	2D
0		1		1	2E
1		1		1	2F
0		1		1	3 0
1		1		1	3 1

Figure 4: 8-bit register Truth Table

The final implementation of the 8-bit ALU in block diagram, schematic format is shown in the next page:

See Figure 5 below.

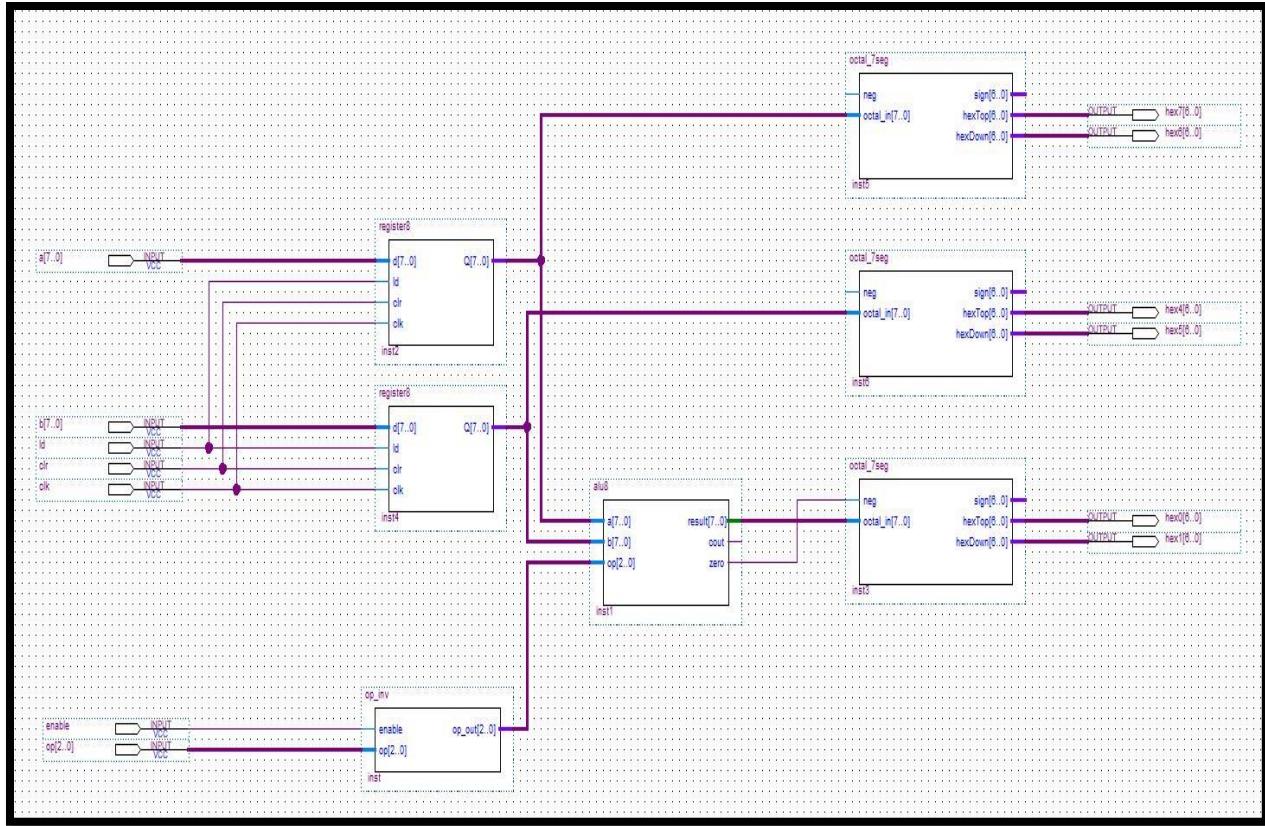


Figure 5: 8-bit ALU Schematic

Observations and Results

Figure 6, Figure 7, Figure 8, Figure 9, Figure 10, Figure 11, Figure 12, and Figure 13 are the waveform results of both functional and timing for the 8-bit ALU.

8-bit Register: Functional and Timing Waveforms

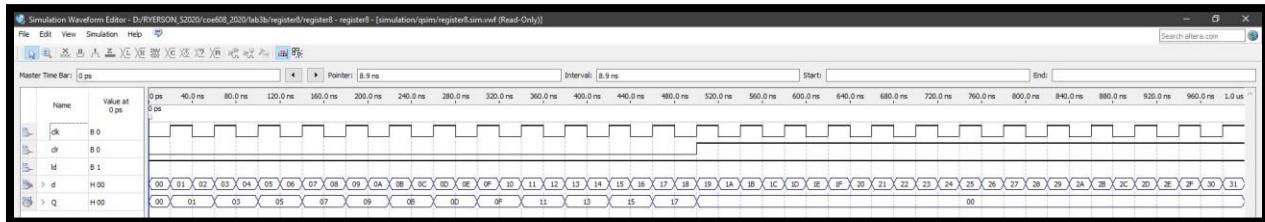


Figure 6: -bit register functional waveform

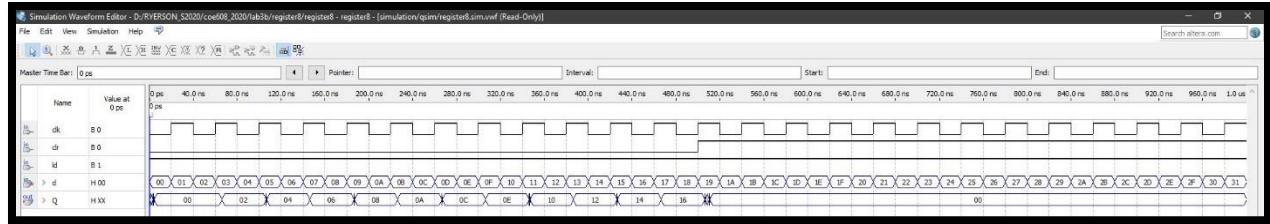


Figure 7: -bit register timing waveform

BCD to 7-segment converter: Functional and Timing Waveforms

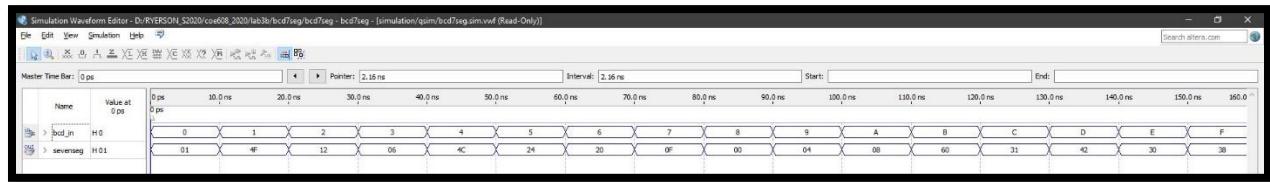


Figure 8:: BCD to 7-segment display functional waveform



Figure 9: BCD to 7-segment display timing waveform

Operational Inverter: Functional and Timing Waveforms

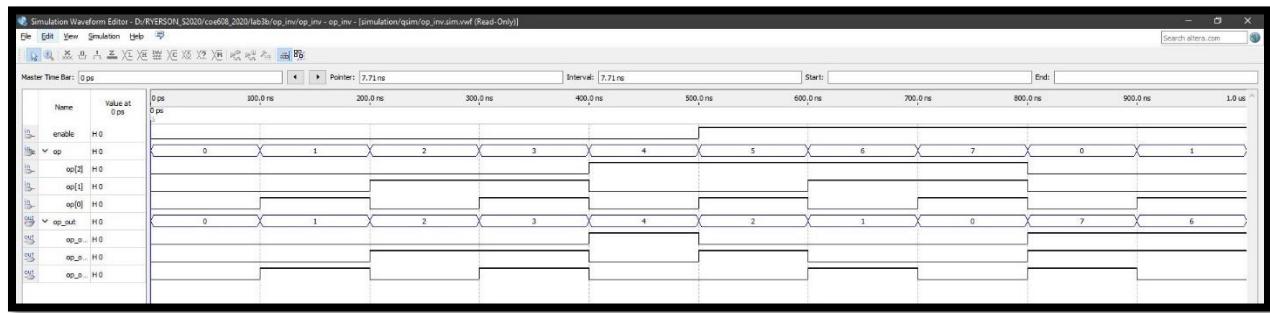


Figure 10: Op-inverter functional waveform

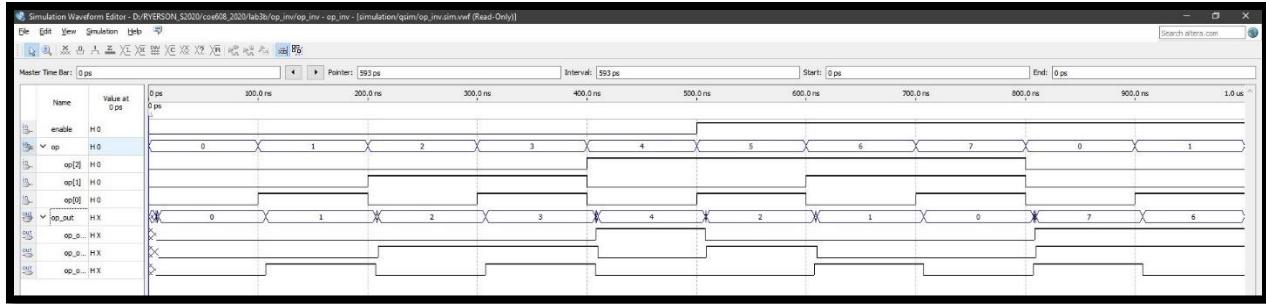


Figure 11: Op-inverter timing waveform

8-bit ALU: Functional and Timing Waveforms

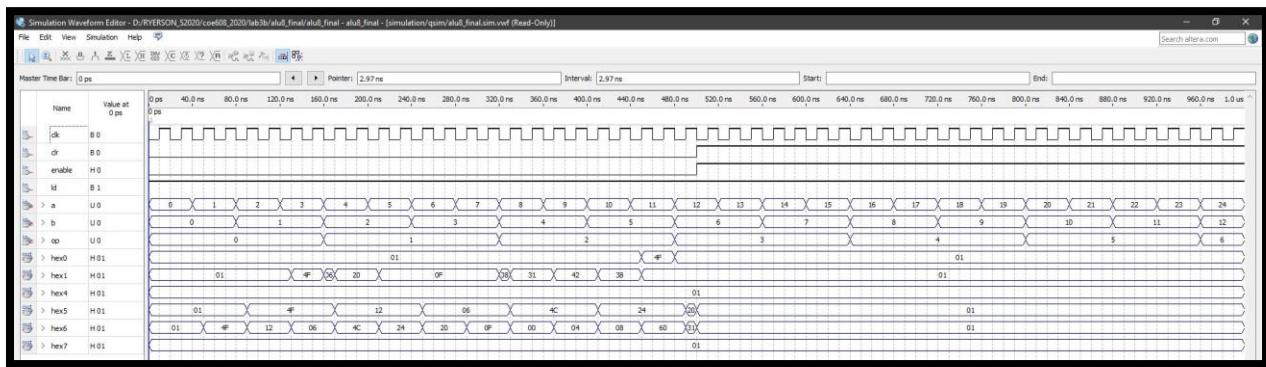


Figure 12: 8-bit ALU functional waveform

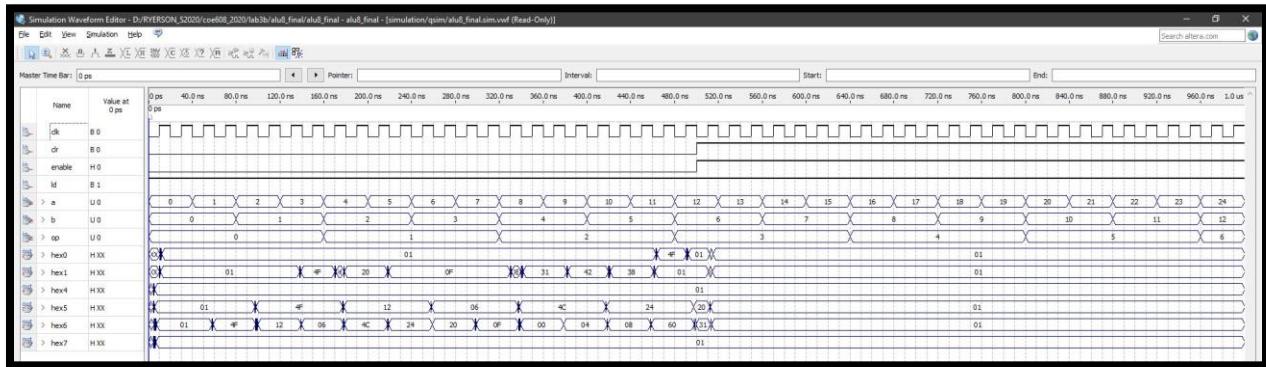


Figure 13: 8-bit ALU timing waveform

Appendices: VHDL Codes for the ALU, 7-segment displays, 8-bit register, and op inverter

The files are attached as PDF to make this document easier to read.

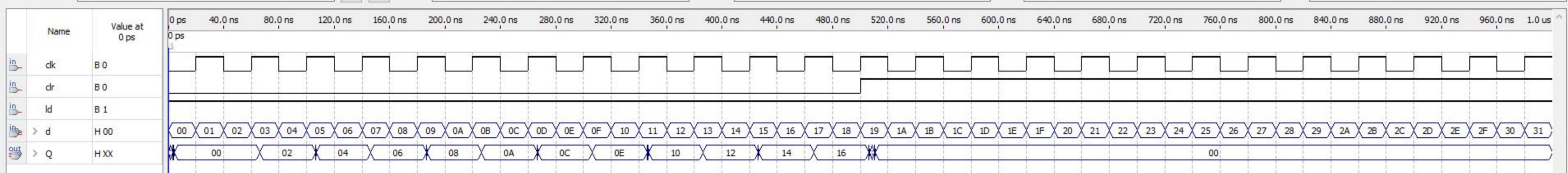
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6
7 entity adder8 is
8 port(
9     cin, mode: in std_logic;
10    x, y : in std_logic_vector(31 downto 0);
11    s: out std_logic_vector(31 downto 0);
12    cout: out std_logic);
13 end adder8;
14
15 architecture description of adder8 is
16 signal c7, c6, c5, c4, c3, c2, c1, cout: std_logic;
17 component adder
18 port(
19     x, y, mode, cin: in std_logic;
20     cout, s : out std_logic);
21 end component adder;
22 begin
23     st0: adder port map (x(0), y(0), mode, mode, c1, s(0));
24     st1: adder port map (x(1), y(1), mode, c1, c2, s(1));
25     st2: adder port map (x(2), y(2), mode, c2, c3, s(2));
26     st3: adder port map (x(3), y(3), mode, c3, c4, s(3));
27     st4: adder port map (x(4), y(4), mode, c4, c5, s(4));
28     st5: adder port map (x(5), y(5), mode, c5, c6, s(5));
29     st6: adder port map (x(6), y(6), mode, c6, c7, s(6));
30     st7: adder port map (x(7), y(7), mode, c7, cout, s(7));
31 end description;
32
33
34
35
36
37
38
39
40
41
42
43
44
```

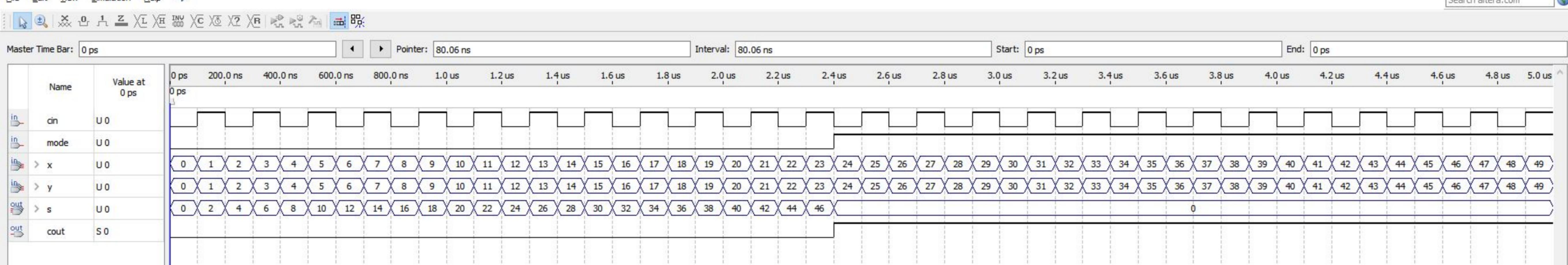
File Edit View Simulation Help

Search altera.com



Master Time Bar: 0 ps Pointer: Interval: Start: End:



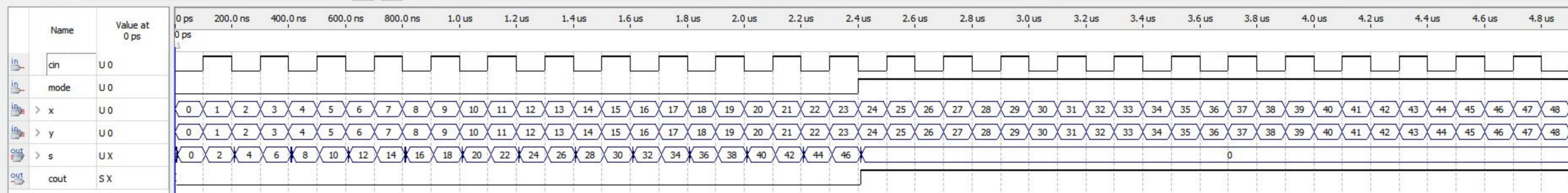


File Edit View Simulation Help

Search altera.com



Master Time Bar: 0 ps Pointer: 53.37 ns Interval: 53.37 ns Start: End:



```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5  use ieee.numeric_std.all;
6
7  entity alu8 is
8      port(
9          a, b: in std_logic_vector(7 downto 0);
10         op: in std_logic_vector(2 downto 0);
11         result: inout std_logic_vector(7 downto 0);
12         cout, zero: out std_logic);
13 end alu8;
14
15 architecture description of alu8 is
16     component adder8 is
17         port(
18             cin, mode: in std_logic;
19             x, y: in std_logic_vector(7 downto 0);
20             s : out std_logic_vector(7 downto 0);
21             cout: out std_logic);
22     end component adder8;
23     signal adddd: std_logic_vector(7 downto 0);
24     signal coutplus: std_logic;
25     signal shift: std_logic_vector(7 downto 0);
26     begin
27         muxadd: adder8 port map (op(2), op(2), a, b, adddd, coutplus);
28         process (op)
29             begin
30                 if op = "000" then
31                     result <= a and b;
32                 elsif op = "001" then
33                     result <= a or b;
34                 elsif op = "010" then
35                     result <= adddd;
36                     cout <= coutplus;
37                 elsif op = "110" then
38                     result <= adddd;
39                     cout <= coutplus;
40                 elsif op = "100" then
41                     shift(0) <= a(7);
42                     shift(7 downto 1) <= a(6 downto 0);
43                     result <= shift;
44                 elsif op = "101" then
45                     shift(7) <= a(0);
46                     shift(6 downto 0) <= a(7 downto 1);
47                     result <= shift;
48                 else
49                     result <= (others => '0');
50                 end if;
51
52                 if result = "00000000" then
53                     zero <= '1';
54                 else
55                     zero <= '0';
56                 end if;
57
58         end process;
59     end description;
```

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  LIBRARY work;
5
6  ENTITY alu8_final IS
7      PORT
8      (
9          enable : IN STD_LOGIC;
10         ld : IN STD_LOGIC;
11         clr : IN STD_LOGIC;
12         clk : IN STD_LOGIC;
13         a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
14         b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
15         op : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
16         hex0 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
17         hex1 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
18         hex4 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
19         hex5 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
20         hex6 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
21         hex7 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
22     );
23 END alu8_final;
24
25 ARCHITECTURE bdf_type OF alu8_final IS
26
27 COMPONENT op_inv
28     PORT(enable : IN STD_LOGIC;
29           op : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
30           op_out : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
31     );
32 END COMPONENT;
33
34 COMPONENT alu8
35     PORT(a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
36           b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
37           op : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
38           result : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
39           cout : OUT STD_LOGIC;
40           zero : OUT STD_LOGIC
41     );
42 END COMPONENT;
43
44 COMPONENT register8
45     PORT(ld : IN STD_LOGIC;
46           clr : IN STD_LOGIC;
47           clk : IN STD_LOGIC;
48           d : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
49           Q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
50     );
51 END COMPONENT;
52
53 COMPONENT octal_7seg
54     PORT(neg : IN STD_LOGIC;
55           octal_in : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
56           hexDown : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
57           hexTop : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
58           sign : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
59     );
60 END COMPONENT;
61
62 SIGNAL SYNTHESIZED_WIRE_7 : STD_LOGIC_VECTOR(7 DOWNTO 0);
```

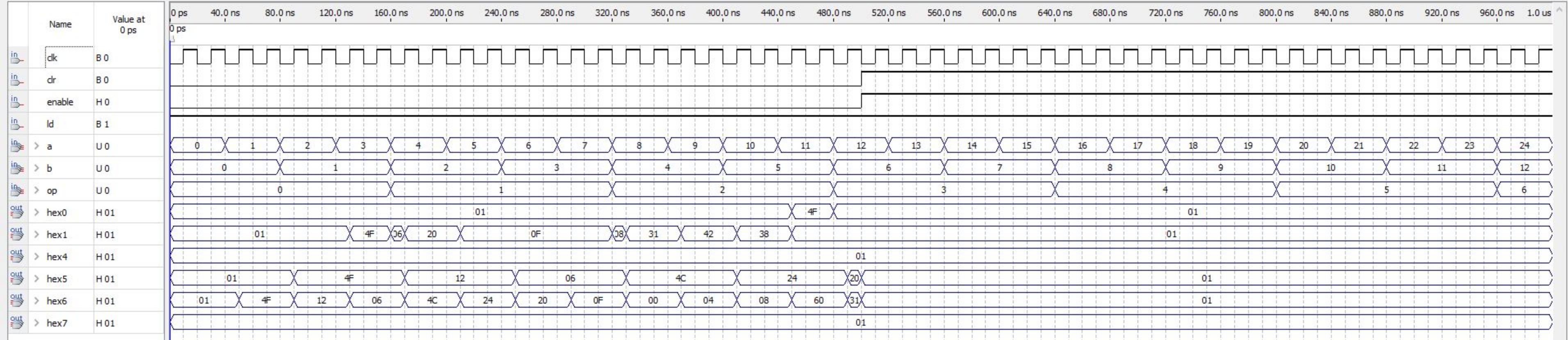
```
63  SIGNAL      SYNTHESIZED_WIRE_8 : STD_LOGIC_VECTOR (7 DOWNTO 0);
64  SIGNAL      op_inv1 : STD_LOGIC_VECTOR (2 DOWNTO 0);
65  SIGNAL      zero1 : STD_LOGIC;
66  SIGNAL      SYNTHESIZED_WIRE_4 : STD_LOGIC_VECTOR (7 DOWNTO 0);
67
68
69 BEGIN
70
71
72
73  b2v_inst : op_inv
74  PORT MAP(enable => enable,
75            op => op,
76            op_out => op_inv1);
77
78
79  b2v_inst1 : alu8
80  PORT MAP(a => A_2_ALU,
81            b => B_2_ALU,
82            op => op_inv1,
83            result => result1,
84            zero => zero1);
85
86
87  b2v_inst2 : register8
88  PORT MAP(ld => ld,
89            clr => clr,
90            clk => clk,
91            d => a,
92            Q => A_2_ALU);
93
94
95  b2v_inst3 : octal_7seg
96  PORT MAP(neg => zero1,
97            octal_in => result1,
98            hexDown => hex1,
99            hexTop => hex0);
100
101
102 b2v_inst4 : register8
103 PORT MAP(ld => ld,
104           clr => clr,
105           clk => clk,
106           d => b,
107           Q => B_2_ALU);
108
109
110 b2v_inst5 : octal_7seg
111 PORT MAP(octal_in => A_2_ALU,
112           hexDown => hex6,
113           hexTop => hex7);
114
115
116 b2v_inst6 : octal_7seg
117 PORT MAP(octal_in => B_2_ALU,
118           hexDown => hex5,
119           hexTop => hex4);
120
121
122 END bdf_type;
```

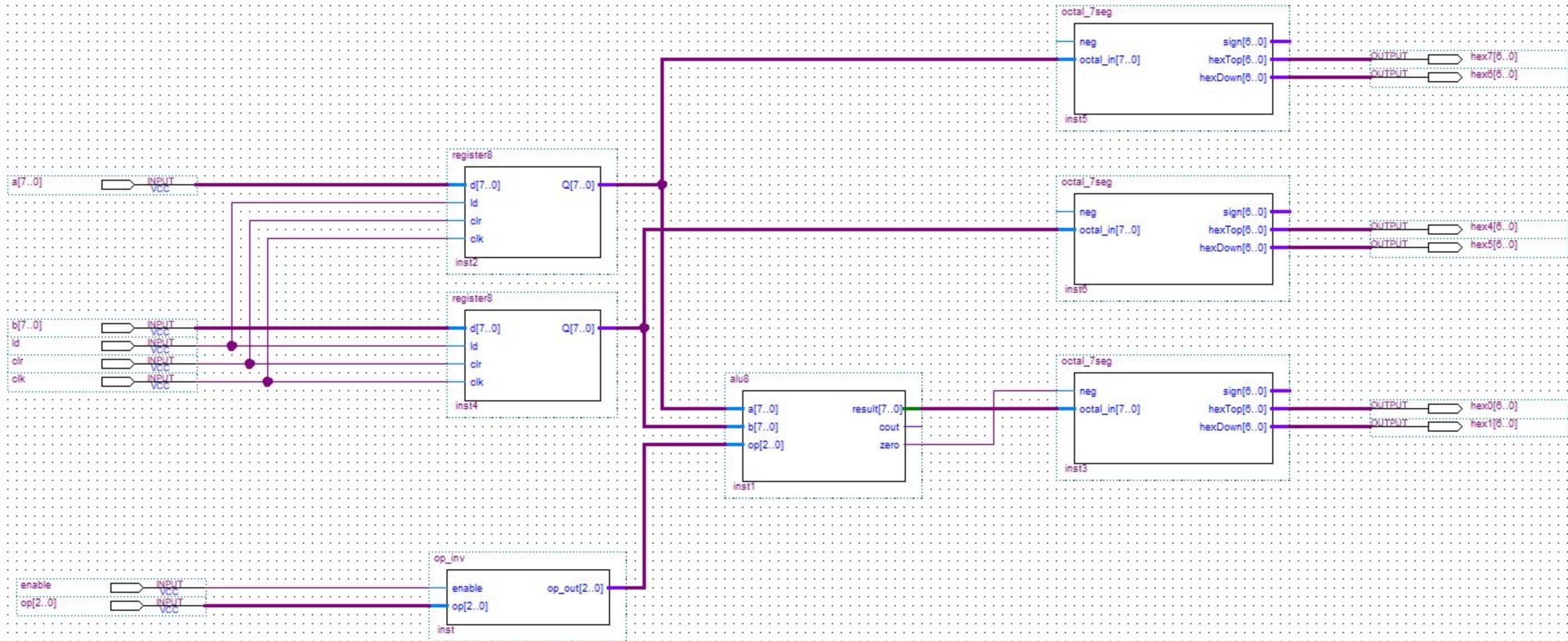
File Edit View Simulation Help

Search altera.com



Master Time Bar: 0 ps Pointer: 2.97 ns Interval: 2.97 ns Start: End:



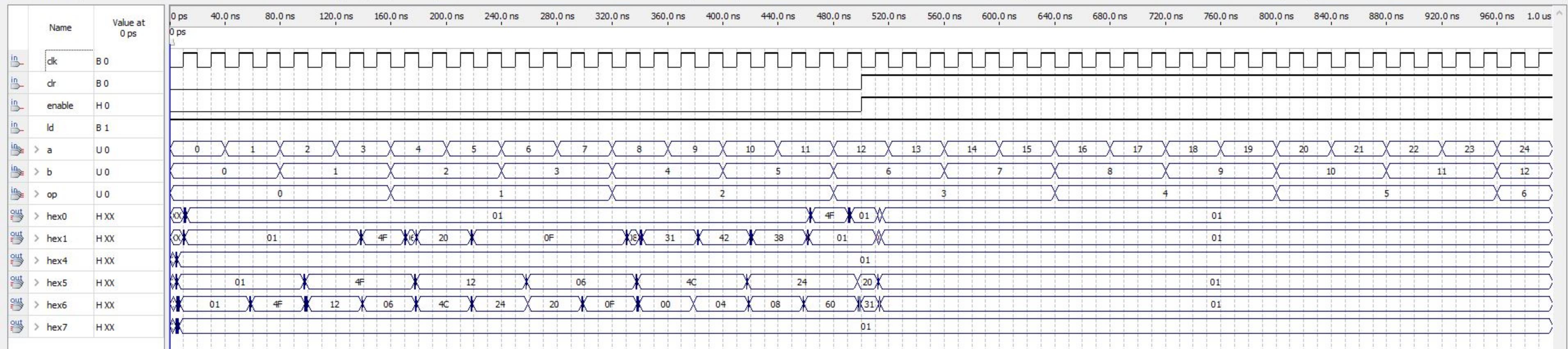


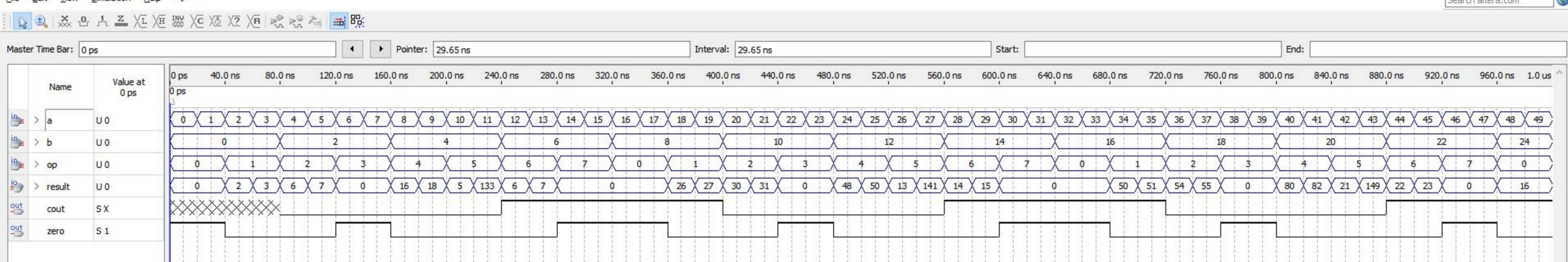
File Edit View Simulation Help

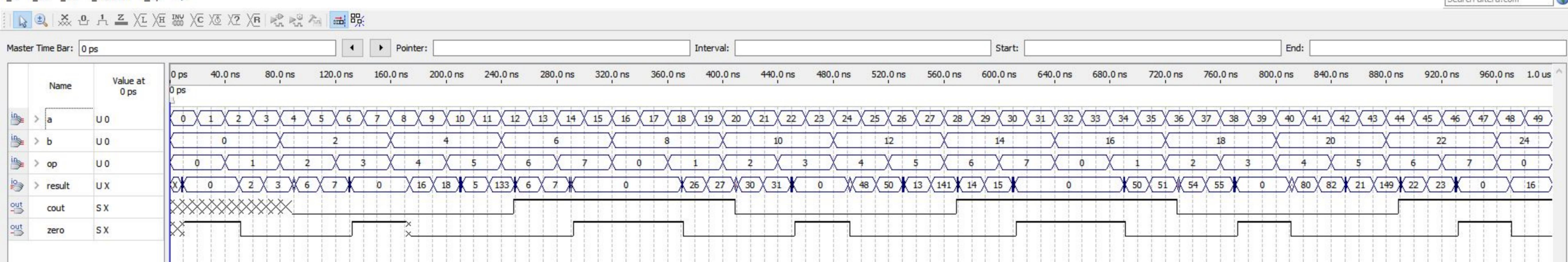
Search altera.com



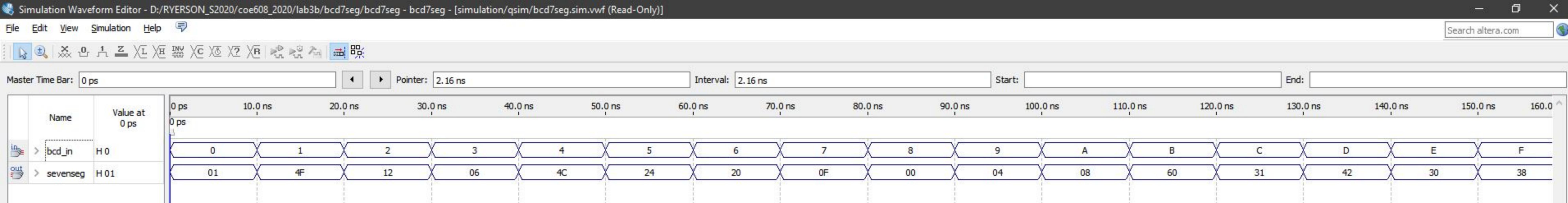
Master Time Bar: 0 ps Pointer: Interval: Start: End:







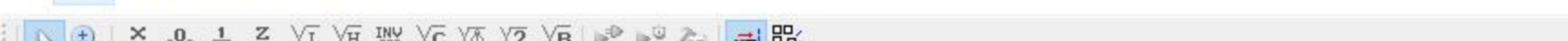
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5  use ieee.numeric_std.all;
6  use ieee.numeric_bit.all;
7
8  entity bcd7seg is
9    port(
10      bcd_in: in std_logic_vector(3 downto 0);
11      sevenseg: out std_logic_vector(6 downto 0));
12 end bcd7seg;
13
14 architecture behavior of bcd7seg is
15 begin
16   process(bcd_in)
17   begin
18     case bcd_in is
19       when "0000" => sevenseg <= "0000001"; --0
20       when "0001" => sevenseg <= "1001111"; --1
21       when "0010" => sevenseg <= "0010010"; --2
22       when "0011" => sevenseg <= "0000110"; --3
23       when "0100" => sevenseg <= "1001100"; --4
24       when "0101" => sevenseg <= "0100100"; --5
25       when "0110" => sevenseg <= "0100000"; --6
26       when "0111" => sevenseg <= "0001111"; --7
27       when "1000" => sevenseg <= "0000000"; --8
28       when "1001" => sevenseg <= "0000100"; --9
29       when "1010" => sevenseg <= "0001000"; --A
30       when "1011" => sevenseg <= "1100000"; --b
31       when "1100" => sevenseg <= "0110001"; --C
32       when "1101" => sevenseg <= "1000010"; --d
33       when "1110" => sevenseg <= "0110000"; --E
34       when "1111" => sevenseg <= "0111000"; --F
35     end case;
36   end process;
37 end behavior;
```



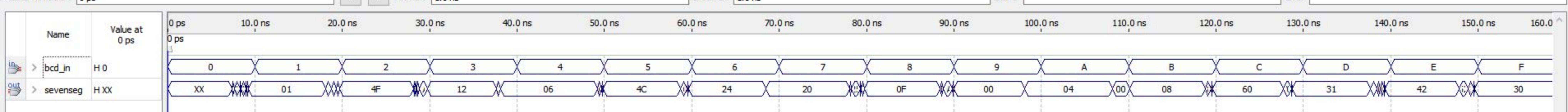
Simulation Waveform Editor - D:/RYERSON_S2020/coe608_2020/lab3b/bcd7seg/bcd7seg - [simulation/qsim/bcd7seg.sim.vwf (Read-Only)]

File Edit View Simulation Help

Search altera.com



Master Time Bar: 0 ps Pointer: 1.6 ns Interval: 1.6 ns Start: End:

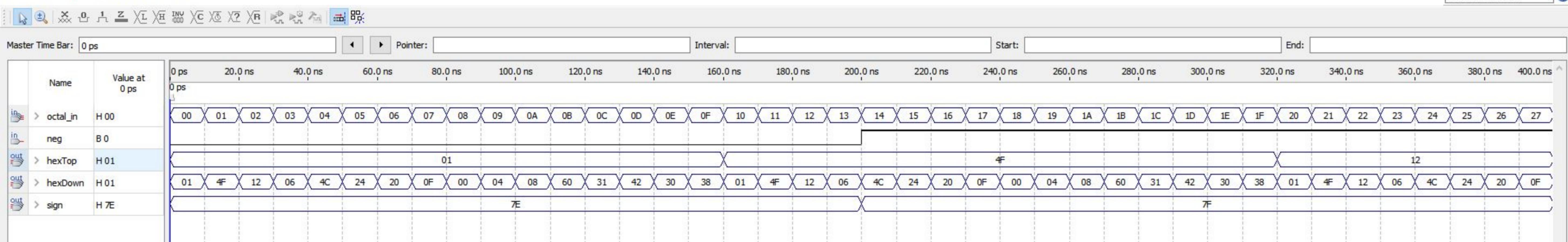


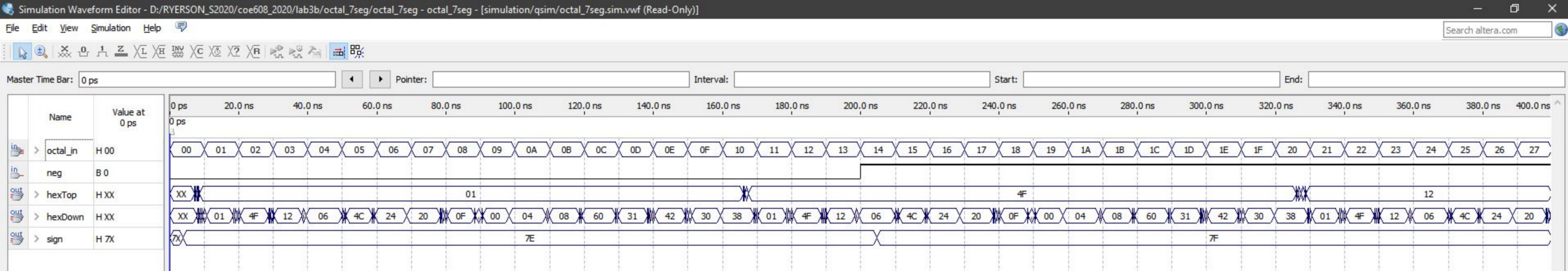
```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5  use ieee.numeric_bit.all;
6  use ieee.numeric_std.all;
7  use ieee.std_logic_misc.all;
8
9  entity octal_7seg is
10   port(
11     neg : in std_logic;
12     octal_in : in std_logic_vector(7 downto 0);
13     sign, hexTop, hexDown : out std_logic_vector(6 downto 0));
14 end octal_7seg;
15
16 architecture behavior of octal_7seg is
17   signal bcdTop, bcdDown : std_logic_vector(3 downto 0);
18 begin
19   begin
20     process(octal_in)
21     begin
22       bcdTop <= octal_in(7 downto 4);
23       bcdDown <= octal_in(3 downto 0);
24       case bcdTop is
25         when "0000" => hexTop <= "0000001"; --0
26         when "0001" => hexTop <= "1001111"; --1
27         when "0010" => hexTop <= "0010010"; --2
28         when "0011" => hexTop <= "0000110"; --3
29         when "0100" => hexTop <= "1001100"; --4
30         when "0101" => hexTop <= "0100100"; --5
31         when "0110" => hexTop <= "0100000"; --6
32         when "0111" => hexTop <= "0001111"; --7
33         when "1000" => hexTop <= "0000000"; --8
34         when "1001" => hexTop <= "0000100"; --9
35         when "1010" => hexTop <= "0001000"; --A
36         when "1011" => hexTop <= "1100000"; --B
37         when "1100" => hexTop <= "0110001"; --C
38         when "1101" => hexTop <= "1000010"; --D
39         when "1110" => hexTop <= "0110000"; --E
40         when "1111" => hexTop <= "0111000"; --F
41         when others => hexTop <= (others => '0');
42       end case;
43       case bcdDown is
44         when "0000" => hexDown <= "0000001"; --0
45         when "0001" => hexDown <= "1001111"; --1
46         when "0010" => hexDown <= "0010010"; --2
47         when "0011" => hexDown <= "0000110"; --3
48         when "0100" => hexDown <= "1001100"; --4
49         when "0101" => hexDown <= "0100100"; --5
50         when "0110" => hexDown <= "0100000"; --6
51         when "0111" => hexDown <= "0001111"; --7
52         when "1000" => hexDown <= "0000000"; --8
53         when "1001" => hexDown <= "0000100"; --9
54         when "1010" => hexDown <= "0001000"; --A
55         when "1011" => hexDown <= "1100000"; --B
56         when "1100" => hexDown <= "0110001"; --C
57         when "1101" => hexDown <= "1000010"; --D
58         when "1110" => hexDown <= "0110000"; --E
59         when "1111" => hexDown <= "0111000"; --F
60         when others => hexDown <= (others => '0');
61       end case;
62       if neg = '0' then
63         sign <= "1111110";

```

```
63      elsif neg = '1' then
64          sign <= "1111111";
65      end if;
66  end process;
67 end behavior;
68
```





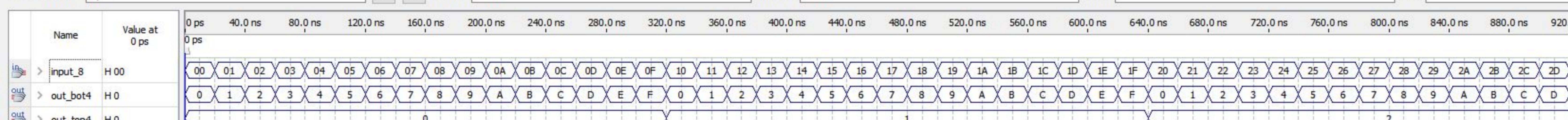
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity octal_bcd_conv is
7 port(
8     input_8 : in std_logic_vector(7 downto 0);
9     out_top4: out std_logic_vector(3 downto 0);
10    out_bot4 : out std_logic_vector(3 downto 0));
11 end octal_bcd_conv;
12
13 architecture behavior of octal_bcd_conv is
14 begin
15     process(input_8)
16     begin
17         out_top4 <= input_8(7 downto 4);
18         out_bot4 <= input_8(3 downto 0);
19     end process;
20 end behavior;
21
22
```

File Edit View Simulation Help

Search altera.com

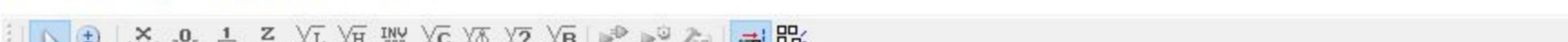


Master Time Bar: 0 ps Pointer: 8.9 ns Interval: 8.9 ns Start: End:

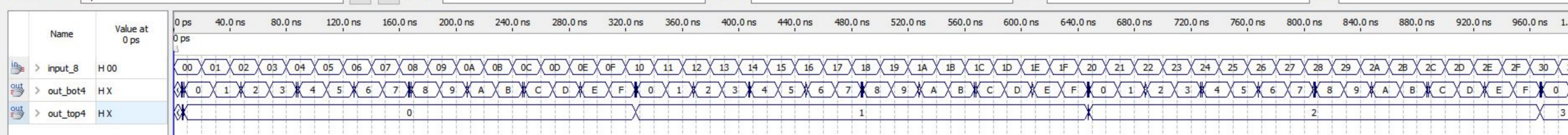


File Edit View Simulation Help

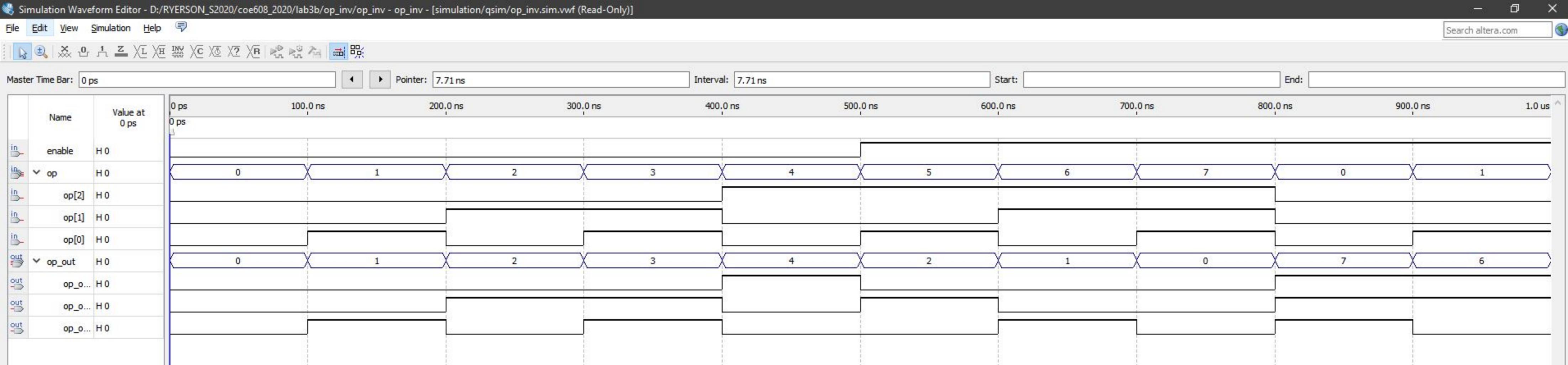
Search altera.com

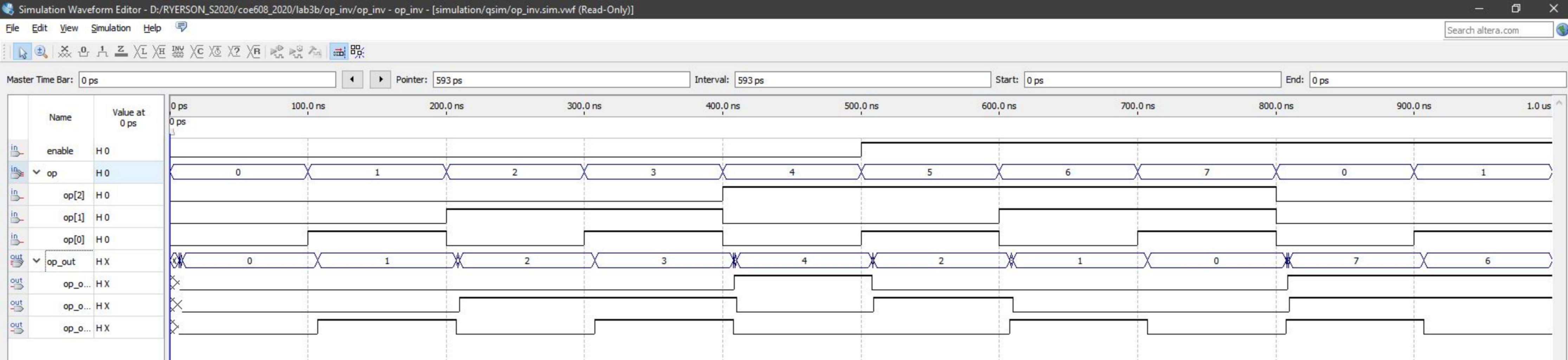


Master Time Bar: 0 ps Pointer: Interval: Start: End:



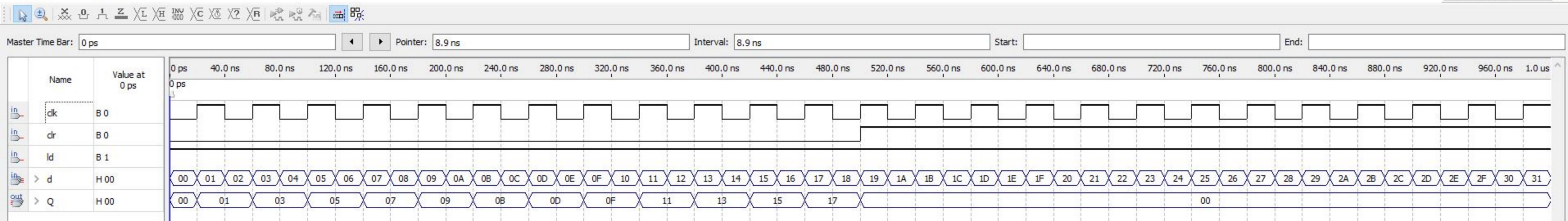
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity op_inv is
7     port(
8         enable: in std_logic;
9         op: in std_logic_vector(2 downto 0);
10        op_out : out std_logic_vector(2 downto 0));
11 end op_inv;
12
13 architecture behavior of op_inv is
14 begin
15     process(enable, op)
16     begin
17         if enable = '0' then
18             op_out <= op;
19         elsif enable = '1' then
20             op_out <= not op;
21         else
22             op_out <= "---";
23         end if;
24     end process;
25 end behavior;
```





```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5 entity register8 is
6     port (
7         d: in std_logic_vector(7 downto 0);
8         ld,clr,clk: in std_logic;
9         Q: out std_logic_vector(7 downto 0));
10    end register8;
11
12 architecture description of register8 is
13 begin
14     process (ld,clr,clk)
15     begin
16         if ld = '0' or clr = '1' then
17             Q <= (others => '0');
18         elsif rising_edge(clk) then
19             Q <= d;
20         end if;
21     end process;
22 end description;
23
24
```

File Edit View Simulation Help



File Edit View Simulation Help

Search altera.com



Master Time Bar: 0 ps Pointer: Interval: Start: End:

