

## ALU Part 1

The screenshot shows the Quartus II 64-Bit IDE interface. The Project Navigator on the left lists files including sseg32.vhd, bufr.vhd, fulladd.vhd, adder4.vhd, adder16.vhd, adder32.vhd, and alu.vhd. The main editor displays the code for alu.vhd, which defines an entity 'alu' with two 31-bit logic vectors 'a' and 'b', a 2-bit logic vector 'op', and two 31-bit logic vectors 'result' and 'zero'. The architecture is set to 'Behavior of alu is' and includes a component 'adder32'. The code defines ports for 'Cin', 'x', 'y', 's', and 'Cout'. The messages pane at the bottom is empty.

```
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 entity alu is
7 port (
8
9     a :in std_logic_vector(31 downto 0);
10    b :in std_logic_vector(31 downto 0);
11    op :in std_logic_vector(2 downto 0);
12    result :out std_logic_vector(31 downto 0);
13    zero :out std_logic;
14    Cout :out std_logic;
15 );
16 end alu;
17 architecture Behavior of alu is
18     component adder32
19     port (
20
21         Cin :in std_logic;
22         x,y :in std_logic_vector(31 downto 0);
23         s :out std_logic_vector(31 downto 0);
24         Cout :out std_logic;
25     );
26 end component;
27 signal result_s :std_logic_vector(31 downto 0):=(others=>'0');
28 signal result_add :std_logic_vector(31 downto 0):=(others=>'0');
```

## ALU Part 2

The screenshot shows the Quartus II 64-Bit IDE interface with the code for alu.vhd updated to include a process block for the ALU operation. The code defines a process 'process (a, b, op)' that uses a case statement to perform different operations based on the 3-bit 'op' vector. The operations include AND, OR, addition, subtraction, and bit shifting. The messages pane at the bottom is empty.

```
25 );
26 end component;
27 signal result_s :std_logic_vector(31 downto 0):=(others=>'0');
28 signal result_add :std_logic_vector(31 downto 0):=(others=>'0');
29 signal result_sub :std_logic_vector(31 downto 0):=(others=>'0');
30 signal Cout_s :std_logic:='0';
31 signal Cout_add:std_logic:='0';
32 signal Cout_sub:std_logic:='0';
33 signal zero_s :std_logic;
34 begin
35
36     add0 :adder32 port map (op(2), a, b, result_add, Cout_add);
37     sub0 :adder32 port map (op(2), a, not b, result_sub, Cout_sub);
38
39     process (a, b, op)
40     begin
41         case (op) is
42
43             when "000" => -- "000" a and b
44                 result_s <= a and b;
45                 Cout_s <= '0';
46             when "001" => -- "001" a or b
47                 result_s <= a or b;
48                 Cout_s <= '0';
49             when "010" => -- "010" a+b
50                 result_s <= result_add;
51                 Cout_s <= Cout_add;
52             when "011" => -- "011" b
53                 result_s <= b;
```

## ALU Part 3

Applications Places System Mon Feb 13, 14:33

Quartus II 64-Bit - /home/student2/s2aljala/COE608/lab3/Lab3 - Lab3

File Edit View Project Assignments Processing Tools Window Help Search altera.com

Project Navigator

- sseg32.vhd
- bufrr.vhd
- fulladd.vhd
- adder4.vhd
- adder16.vhd
- adder32.vhd
- alu.vhd
- ALUWVFvfwf.vw

Tasks

Flow: F Customize...

```
53 result_s <= b;  
54 Cout_s <= '0';  
55 when "110" => -- "110" a-b  
56 result_s <= result_sub;  
57 Cout_s <= Cout_sub;  
58 when "100" => -- a sll 1  
59 result_s <= a(30 downto 0) & '0';  
60 Cout_s <= a(31);  
61 when "101" => -- a srl 1  
62 result_s <= '0' & a(31 downto 1);  
63 Cout_s <= '0';  
64 when others =>  
65 result_s <= a;  
66 Cout_s <= '0';  
67  
68 end case;  
69 case (result_s) is  
70 when (others => '0') =>  
71 zero_s <= '1';  
72 when others =>  
73 zero_s <= '0';  
74  
75 end case;  
76 end process;  
77 result <= result_s;  
78 Cout <= Cout_s;  
79 zero <= zero_s;  
80 end Behavior;
```

IP Catalog

- Installed IP
- Project Directory
- No Selection Available
- Library
- Basic Functions
- DSP
- Interface Protocols
- Memory Interfaces and Controllers
- Processors and Peripherals
- Search for Partner IP

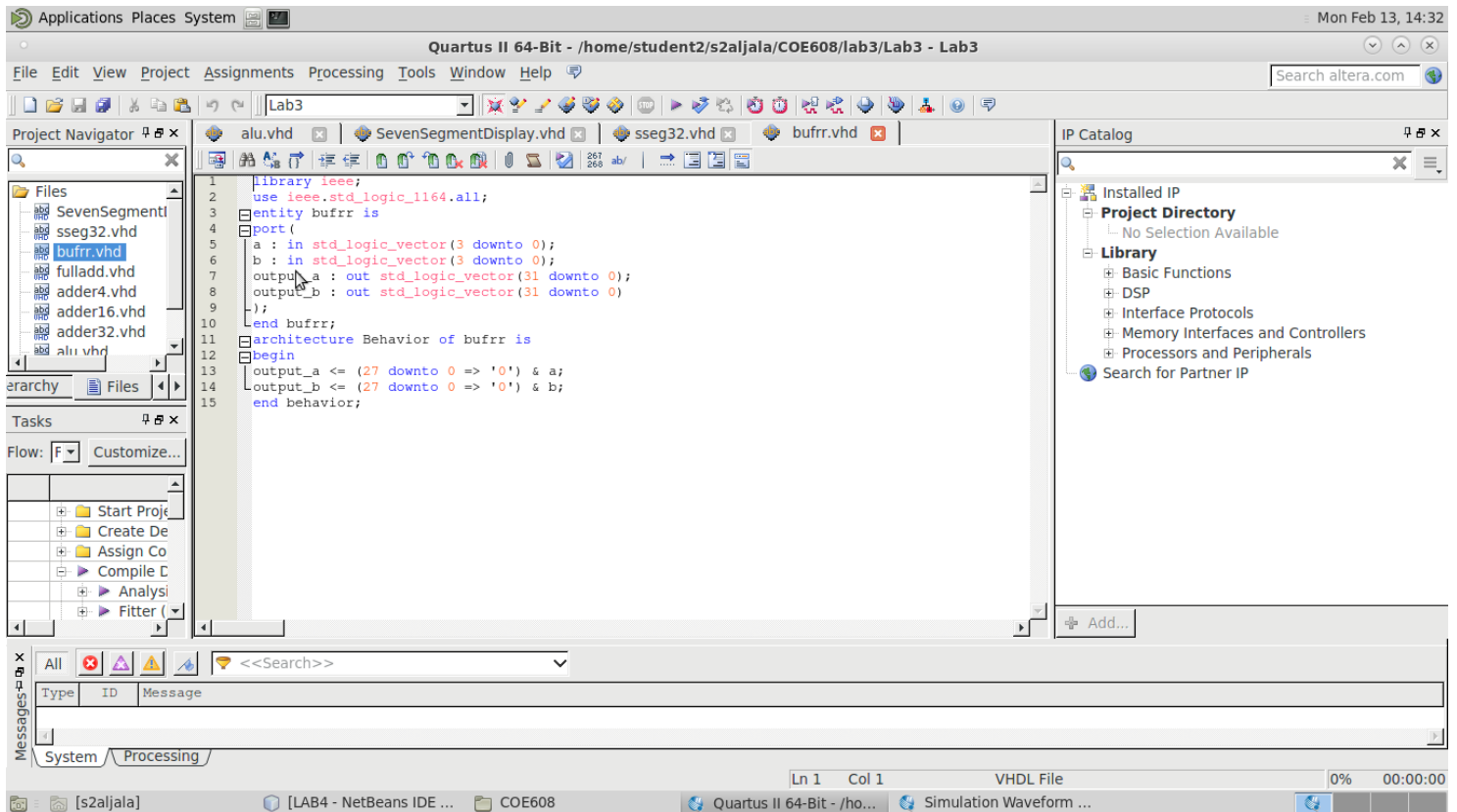
Messages

System / Processing

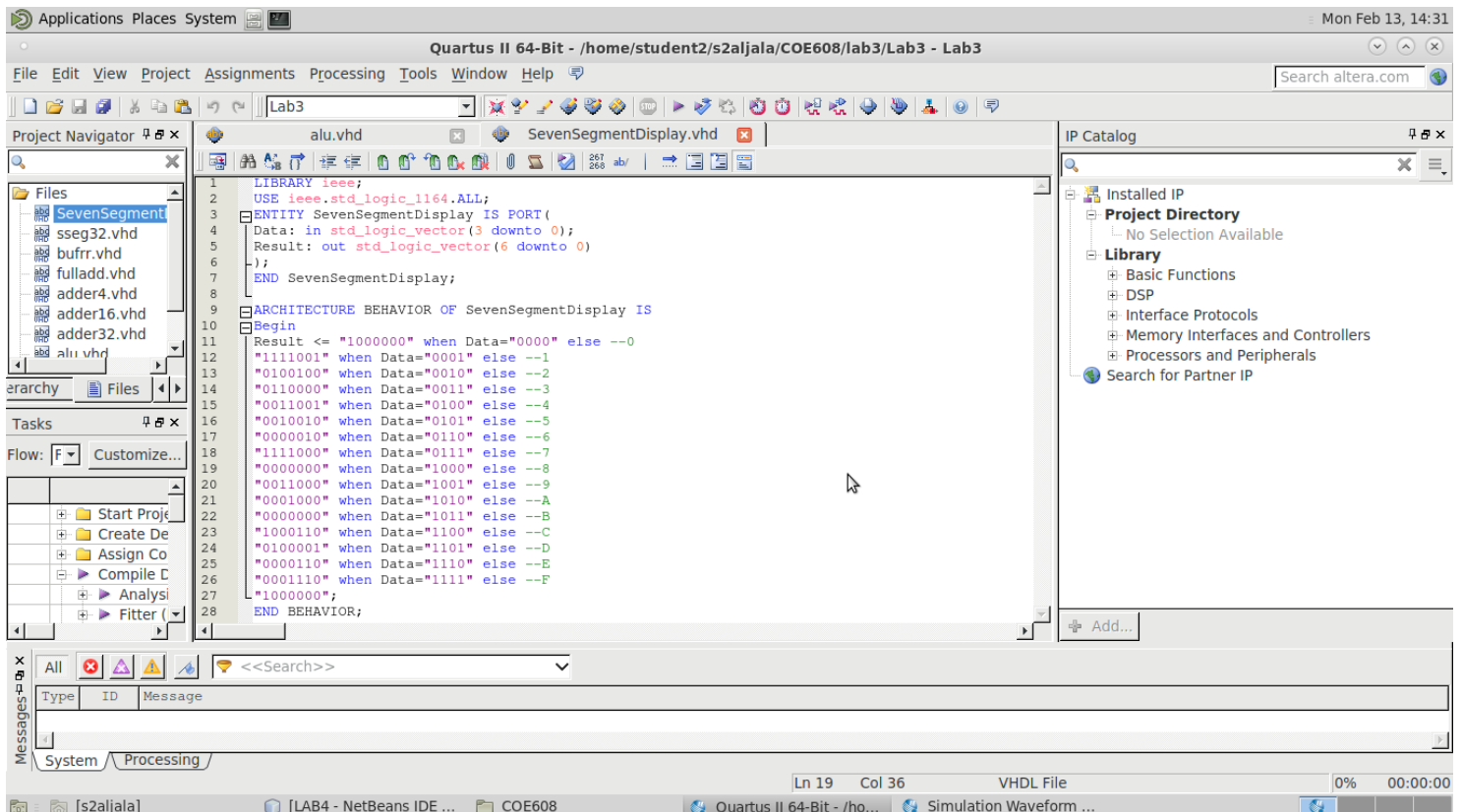
0% 00:00:00

[s2aljala] [LAB4 - NetBeans IDE ...] [COE608] [Quartus II 64-Bit - /ho...] [Simulation Waveform ...]

## Buffer



## Seven Segment



## fulladd

Quartus II 64-Bit - /home/student2/s2aljala/COE608/lab3/Lab3 - Lab3

File Edit View Project Assignments Processing Tools Window Help

Search altera.com

Project Navigator

Files

- SevenSegment1
- sseg32.vhd
- bufr.vhd
- fulladd.vhd
- adder4.vhd
- adder16.vhd
- adder32.vhd
- alu.vhd

Tasks

Flow: F Customize...

Messages

System / Processing

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 entity fulladd is
4 port (
5     Cin, x, y :in std_logic;
6     s, Cout :out std_logic
7 );
8 end fulladd;
9 architecture Behavior of fulladd is
10 begin
11     s<= x xor y xor Cin;
12     Cout<=(x and y) or (Cin and x) or (Cin and y);
13 end Behavior;
```

Ln 1 Col 1 VHDL File 0% 00:00:00

[s2aljala] [LAB4 - NetBeans IDE ...] [COE608] [Quartus II 64-Bit - /ho...] [Simulation Waveform ...]

## adder32

Quartus II 64-Bit - /home/student2/s2aljala/COE608/lab3/Lab3 - Lab3

File Edit View Project Assignments Processing Tools Window Help

Search altera.com

Project Navigator

Files

- SevenSegment1
- sseg32.vhd
- bufr.vhd
- fulladd.vhd
- adder4.vhd
- adder16.vhd
- adder32.vhd
- alu.vhd

Tasks

Flow: F Customize...

Messages

System / Processing

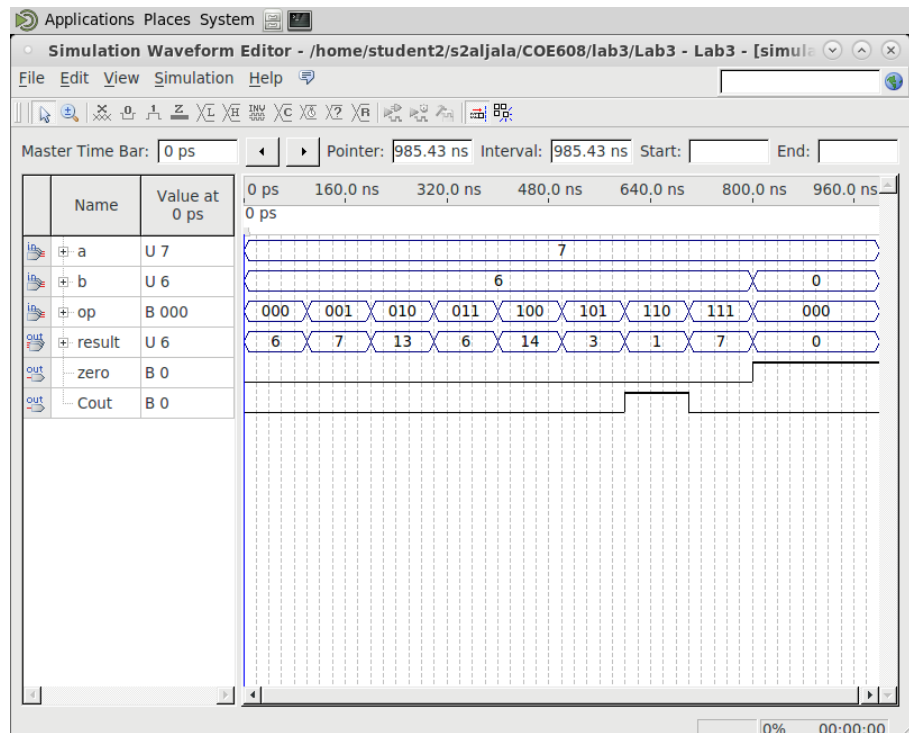
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 entity adder32 is
4 port (
5     Cin :in std_logic;
6     x,y :in std_logic_vector(31 downto 0);
7     s :out std_logic_vector(31 downto 0);
8     Cout :out std_logic
9 );
10 end adder32;
11 architecture Behavior of adder32 is
12     component adder16
13     port (
14         Cin :in std_logic;
15         x,y :in std_logic_vector(15 downto 0);
16         s :out std_logic_vector(15 downto 0);
17         Cout :out std_logic
18     );
19 end component;
20 signal c :std_logic;
21 begin
22     stage0: adder16 port map(Cin, x(15 downto 0), y(15 downto 0), s(15 downto 0), c);
23     stage1: adder16 port map(c, x(31 downto 16), y(31 downto 16), s(31 downto 16), Cout);
24 end Behavior;
```

Ln 1 Col 1 VHDL File 0% 00:00:00

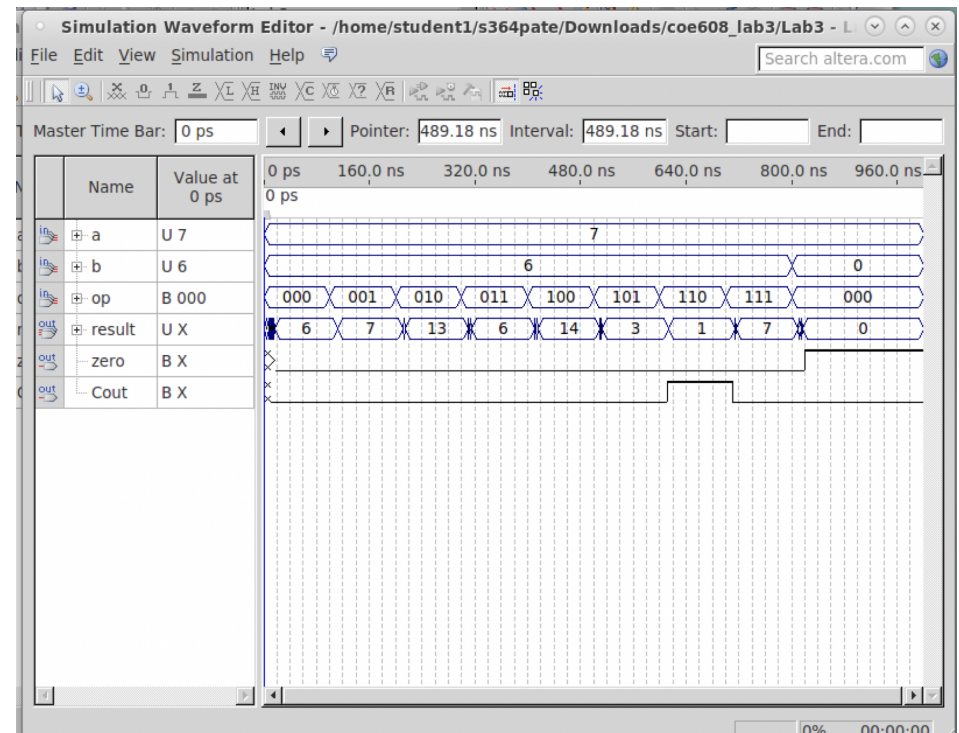
[s2aljala] [LAB4 - NetBeans IDE ...] [COE608] [Quartus II 64-Bit - /ho...] [Simulation Waveform ...]

## 32-bit ALU Waveforms:

### Functional Simulation



### Time Simulation



**\*Please Scroll Down to see the explanation on the two waveforms.**

## **Description of Input and Output Functionalities**

The graphs above are a representation of the 32-bit ALU Design which has inputs a,b, op and results along with outputs zero, Cout. For both waveforms input values a and b are decimal while others are in binary. Input 'a' is set as 7 and input 'b' is set as 6. The op values are default which represent the ALU arithmetics and the "result" is the output of the 32-bit ALU (Arithmetic Logic Unit), that stores the result of the operation performed on the inputs "a" and "b" by the ops. "zero" is a single-bit output signal that represents whether the result of the operation performed by the ALU is equal to zero or not. If the result is equal to zero, the zero output is set to '1', otherwise it is set to '0'. The carry out bit is used to indicate if there was an overflow or carry-out during the operation. If there is an overflow or carry-out, Cout is set to '1', otherwise it is set to '0'.

1. For op = "000", the operation performed is "a and b" and the result is 6.
2. For op = "001", the operation performed is "a or b" and the result is 7.
3. For op = "010", the operation performed is "a + b" and the result is 13.
4. For op = "011", the operation performed is "b" and the result is 6.
5. For op = "100", the operation performed is "a SLL 1" (left shift by 1 bit) and the result is 14.
6. For op = "110", the operation performed is "a - b" and the result is 1. Cout is '1' indicating that there was a borrow during the subtraction operation.
7. For op = "111", the operation performed is "a" and the result is 7.
8. For op = "000", the result is 0 because input b has changed to '0' while input a=7. The zero output is '1' to indicate that the result of ALU equals to 0.

## **Difference Between Functional Simulations Time Simulation**

Both waveforms are very much identical to each other except the only difference is that there is a delay shown in the time waveform. This is because the functional simulation verifies the correct functioning of the circuit, and showcases an ideal output without considering the timing delay of individual gates. On the other hand, time simulation waveform showcases more of a realistic output as it considers the timing delay of individual gates. The delay is calculated based on the type of gate, its size, the load on its output, and the voltage and temperature conditions. This delay is important to consider as it affects the final result and can lead to errors in the circuit.

## **Explanation of the Delays**

The waveform in a time simulation reveals delays at the start and end due to the propagation time of input signals and the setup time of the first flip-flop. There may also be minor delays between specific operations, such as "010" and "011" and "100" and "101," due to the logic gates employed in the design. The subtraction operation (op="110") has the worst-case delay among the various operations since it needs the CPU to invert the integer, making it slower than the addition operation (op="010"). In comparison, the operations "a and b" (op="000") and "a or b" (op="001") have relatively lesser delays, as they require simple logic operations. As a result, these timing delays must be considered since they can affect the accuracy of the circuit's final result.