**Computer Engineering -CSC 7011**

**Prof. Stephen Taylor**

**Arnika Vishwakarma (@01367603)**

**LAB REPORT- 08**

**Title:** Building a 32-bit Arithmetic Logic Unit(ALU).

**Purpose:**

* The purpose of thus lab is to design and simulate the 32-bit ALU.

**Requirements:**

* Altera Quartus II 9.1p2 software

**Description:**

**Altera Quartus II 9.1p2 –** It’s a wed edition software which allow the user to analyze and synthesize the HDL designs, which enables the user to design the schematic model and simulate the design. Quartus implement the VHDL and Verilog for hardware description, vector waveform simulation and visual editing of logic circuits.

**Truth table of ALU Operations:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **F[2:0]** | **A** | **B** | **Y** | **Zero** |
| ADD 0+0 | 2 | 00000000 | 00000000 | 00000000 | 1 |
| ADD 0+(-1) | 2 | 00000000 | FFFFFFFF | FFFFFFFF | 0 |
| ADD 1+(-1) | 2 | 00000001 | FFFFFFFF | 00000000 | 1 |
| ADD FF+1 | 2 | 000000FF | 00000001 | 0000100 | 0 |
| SUB 0-0 | 6 | 00000000 | 00000000 | 00000000 | 1 |
| SUB 0-(-1) | 6 | 00000000 | FFFFFFFF | 00000001 | 0 |
| SUB 1-1 | 6 | 00000001 | FFFFFFFF | 00000000 | 1 |
| SUB 100-1 | 6 | 00000100 | 00000001 | 000000FF | 0 |
| SLT 0,0 | 7 | 00000000 | 00000000 | 00000000 | 1 |
| SLT 0,1 | 7 | 00000000 | 00000001 | 00000001 | 0 |
| SLT 0,-1 | 7 | 00000000 | FFFFFFFF | 00000000 | 1 |
| SLT 1,0 | 7 | 00000001 | 00000000 | 00000000 | 1 |
| SLT -1,0 | 7 | FFFFFFFF | 00000000 | 00000001 | 0 |
| AND FFFFFFFF, FFFFFFFF | 0 | FFFFFFFF | FFFFFFFF | FFFFFFFF | 0 |
| AND FFFFFFFF, 12345678 | 0 | FFFFFFFF | 12345678 | 12345678 | 0 |
| AND 12345678, 87654321 | 0 | 12345678 | 87654321 | 02244220 | 0 |
| AND 00000000, FFFFFFFF | 0 | 00000000 | FFFFFFFF | 00000000 | 1 |
| OR FFFFFFFF, FFFFFFFF | 1 | FFFFFFFF | FFFFFFFF | FFFFFFFF | 0 |
| OR 12345678, 87654321 | 1 | 12345678 | 87654321 | 97755779 | 0 |
| OR 00000000, FFFFFFFF | 1 | 00000000 | FFFFFFFF | FFFFFFFF | 0 |
| OR 00000000, 00000000 | 1 | 00000000 | 00000000 | 00000000 | 1 |

**ALU Operations:**

|  |  |
| --- | --- |
| **F2:0** | **Function** |
| **000** | **A AND B** |
| **001** | **A OR B** |
| **010** | **A +B** |
| **011** | **NOT USED** |
| **100** | **A AND B´** |
| **101** | **A OR B´** |
| **110** | **A - B** |
| **111** | **SLT** |

**For SLT function if**

* A <B then, Y=1
* A > B then, Y=0

**For ALU Operation truth table:**

* If Y= 0 then, zero=True i.e. =1
* If Y=1 then, zero=False i.e.= 0

**SCHEMATIC DESIGN CODE:**

After working for more than 2 hours in lab we developed the schematic design using Altera Quartus II, simulated the design in modelsim.

**Step 1:** Creating a new schematic, click **File/New/VHDL,** after the windows opens click on the Templates icon and choose **VHDL.**

**Step 2:** Below is the code for ALU Design:

**alu.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_TEXTIO.ALL;

use IEEE.NUMERIC\_STD.ALL;

use STD.TEXTIO.all;

entity alu is

port(a,b: in std\_logic\_vector(31 downto 0);

f: in std\_logic\_vector(2 downto 0);

y: out std\_logic\_vector(31 downto 0);

zero: out std\_logic

);

end entity;

architecture sim of alu is

signal y1: std\_logic\_vector(31 downto 0);

begin

y<= y1;

process(f, a, b) begin

case f is

when "000" =>

y1 <= a and b;

when "001" =>

y1 <= a or b;

when "010" =>

y1 <= std\_logic\_vector(signed(a) + signed(b));

when "100" =>

y1 <= a and not b;

when "101" =>

y1 <= a or not b;

when "110" =>

y1 <= std\_logic\_vector(signed(a) - signed(b));

when "111"=>

if (signed(a) < signed(b)) then

y1 <= "00000000000000000000000000000001";

else

y1 <= "00000000000000000000000000000000";

end if;

when others =>

y1 <= a;

end case;

if(y1 = "00000000000000000000000000000000")then

zero<= '1';

else

zero<= '0';

end if;

end process;

end ;

**alu.tv**

2\_00000000\_00000000\_00000000\_1

2\_00000000\_FFFFFFFF\_FFFFFFFF\_0

2\_00000001\_FFFFFFFF\_00000000\_1

2\_000000FF\_00000001\_00000100\_0

6\_00000000\_00000000\_00000000\_1

6\_00000000\_FFFFFFFF\_00000001\_0

6\_00000001\_00000001\_00000000\_1

6\_00000100\_00000001\_000000FF\_0

7\_00000000\_00000000\_00000000\_1

7\_00000000\_00000001\_00000001\_0

7\_00000000\_FFFFFFFF\_00000000\_1

7\_00000001\_00000000\_00000001\_0

7\_FFFFFFFF\_00000000\_00000000\_1

0\_FFFFFFFF\_FFFFFFFF\_FFFFFFFF\_0

0\_FFFFFFFF\_12345678\_12345678\_0

0\_12345678\_87654321\_02244220\_0

0\_00000000\_FFFFFFFF\_00000000\_1

1\_FFFFFFFF\_FFFFFFFF\_FFFFFFFF\_0

1\_12345678\_87654321\_97755779\_0

1\_00000000\_FFFFFFFF\_FFFFFFFF\_0

1\_00000000\_00000000\_00000000\_1

**TESTBENCH.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_TEXTIO.ALL;

use STD.TEXTIO.all;

entity testbench3 is

end;

architecture sim of testbench3 is

component alu

port (a,b: in std\_logic\_vector(31 downto 0);

f: in std\_logic\_vector(2 downto 0);

y: out std\_logic\_vector(31 downto 0);

zero: out std\_logic

);

end component;

signal a,b,y: std\_logic\_vector(31 downto 0);

signal f: std\_logic\_vector(2 downto 0);

signal zero: std\_logic;

signal y\_expected: std\_logic\_vector(31 downto 0);

signal zero\_expected: std\_logic;

signal clk, reset: std\_logic;

begin

-- instantiate device under test

dut: alu port map(a,b,f,y,zero);

-- generate clock

process begin

clk <= '1' ; wait for 5 ns;

clk <= '0' ; wait for 5 ns;

end process;

-- at start of test, pulse reset

process begin

reset <= '1'; wait for 27 ns; reset <= '0';

wait;

end process;

-- run tests

process is

file tv: text;

variable L: line;

variable vector\_a: std\_logic\_vector(31 downto 0);

variable dummy1: character;

variable vector\_b: std\_logic\_vector(31 downto 0);

variable dummy2: character;

variable vector\_f: std\_logic\_vector(3 downto 0);

variable dummy3: character;

variable vector\_y\_ex: std\_logic\_vector(31 downto 0);

variable dummy4: character;

variable vector\_z\_ex: std\_logic\_vector(3 downto 0);

variable vectornum: integer := 0;

variable errors: integer := 0;

begin

FILE\_OPEN(tv, "J:\Lab8\alu.tv", READ\_MODE);

while not endfile(tv) loop

-- change vectors on rising edge

wait until rising\_edge(clk);

-- read the next line of testvectors split it up

readline(tv, L);

hread(L,vector\_f);

read(L, dummy3);

hread(L, vector\_a);

read(L, dummy1);

hread(L,vector\_b);

read(L, dummy2);

hread(L,vector\_y\_ex);

read(L, dummy4);

hread(L,vector\_z\_ex);

a <= vector\_a after 1 ns;

b <= vector\_b;

f <= vector\_f(2 downto 0);

y\_expected <= vector\_y\_ex;

zero\_expected <= vector\_z\_ex(0);

wait until falling\_edge(clk);

if y /= y\_expected then

report("Error with line: " & integer'image(vectornum));

-- report string'image(L);

report("zero = " & std\_logic'image(zero));

-- report("Error: y=" & std\_logic\_vector'image(y));

-- report("Error: a=" & std\_logic\_vector'image(a));

-- report("Error: b=" & std\_logic'image(b));

--report("Error: f=" & std\_logic'image(f));

errors := errors + 1;

end if;

vectornum := vectornum + 1;

end loop;

if errors=0 then

report "NO ERRORS" & integer'image(vectornum) & "tests completed" severity failure;

else

report integer'image(errors) & " ERRORS in " &

integer'image(vectornum) & "tests" severity failure;

end if;

end process;

end;

**SIMULATION**

After designing the schematic model of seven segment we simulate the design in HDL such as VHDL using ModelSim simulator. Following are the steps followed to simulate the design:

**Step 1:** We simulate the design using ModelSim. ModelSim expects a description of a circuit in a hardware description language (HDL).

**Step 2:** Now fire up ModelSim SE 10.1b from the Tools, RTL Run Simulation Tool pulldown or directly from the start menu. Make sure all the two files alu.vhd and testbench.vhd should be there to compile.

**Step 3:** Chose **Compile / Compile All** to compile the VHDL code into a form that ModelSim can simulate. Then chose **Simulate / Start Simulation**.

**Step 4:** When the simulator starts, ModelSim will open more panes including sim and Objects that help us select signals for the waveform viewer. In the objects window, we’ll see all the inputs, outputs, and internal wires. Shift-click to select them all. Then right-click and choose Add to Wave, Selected Signals. A Wave pane will pop up with the signals.

**Step 5:** To add the wave position, right-click on the names in the object pane and choose Add to wave.

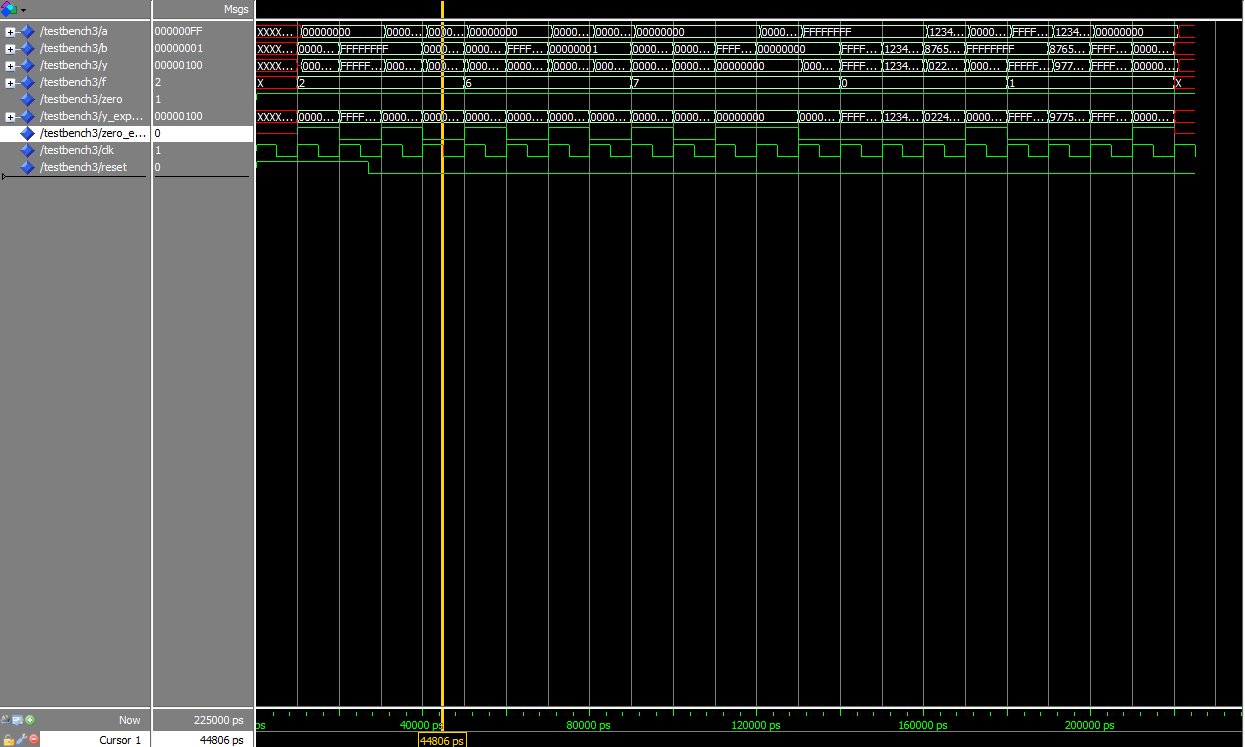
****

Fig. 1 Simulation waveform of ALU operations