LabWork 5 ALU using VHDL
Batu Arda Düzgün
21802633
11.7.2019

Abstract / Objective:

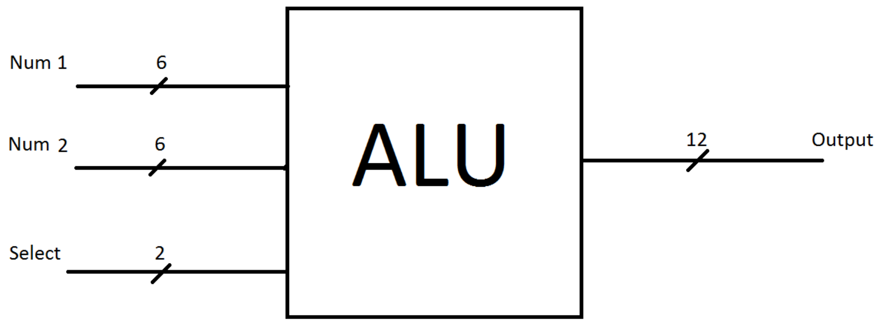
In this lab I will design and implement an arithmetic ALU. ALUs are basic operation units of computers most of them have tons of them in their CPUs. So making an ALU is one of our first steps towards making a computer. ALU does operations on two input signal based on another selection signal. My ALU will be an arithmetic's only ALU so it will do addition, absolute value subtraction and multiplication on 2 6 bit input busses.

Design Specification Plan:

To do the arithmetic operation I will use IEEE.STD_LOGIC_1164, IEEE.NUMERIC_STD, IEEE.STD_LOGIC_UNSIGNED, use IEEE.STD_LOGIC_ARITH. This already have logic circuits for addition, subtraction and multiplication for any length (bit) of signal. I can write my own logic equations for all of these but writing the logic equation for a 6 bit 6 bit multiplier would take close to 1 hundred logic gates (AND, OR, XOR). Then I will connect those operations out puts to a 4 to 1 multiplexer then chose which operation output will be given with the selection signal. Finally I will use the leds to display the output because it is the most convenient option.

Proposed Design Methodology:

My ALU will do addition, subtraction and multiplication. It will take 2 6 bit inputs from the switches 0 to 11 and it will take a 2 bit selection input from the switches 15 to 16. Then it will display its 12 bit output on the Led 0 to 11. I will have a top module without any processes in it so will have 3 components for the addition subtraction and multiplication. And another component for the multiplexer. The subtraction operation can only give a positive number because our inputs are not singed. Also it output is 6 bits so I will add "000000"to left of its signal to increase it to 12 bits. In the addition operation I am adding "0" to the left of both signals and then adding them to not lose the 7th bit. Then I will add "00000" to the left of the signal to increase it to 12 bits. The multiplication operation already gives a 12 bit output so it doesn't need any correction from us.



Block diagram of the ALU

LabWork 5 ALU using VHDL
Batu Arda Düzgün
21802633
11.7.2019

VHDL Model:

I know I did not have to write the code but I already did write it so I am putting it here.

Top modele:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity top_module is
   Port ( Sw : in STD_LOGIC_VECTOR (11 downto 0);
         slc : in STD_LOGIC_VECTOR (1 downto 0);
         LED : out STD_LOGIC_VECTOR (11 downto 0));
end top_module;

architecture Behavioral of top_module is

signal addition : STD_LOGIC_VECTOR (6 downto 0);
signal subtraction : STD_LOGIC_VECTOR (5 downto 0);
signal multiplication : STD_LOGIC_VECTOR (11 downto 0);

component adder
port(
   Anum1 : in STD_LOGIC_VECTOR (5 downto 0);
   Anum2 : in STD_LOGIC_VECTOR (5 downto 0);
   Aoutput : out STD_LOGIC_VECTOR (6 downto 0));
```

```
end component;


component subtracter
port(
   Snum1 : in STD_LOGIC_VECTOR (5 downto 0);
   Snum2 : in STD_LOGIC_VECTOR (5 downto 0);
   Soutput : out STD_LOGIC_VECTOR (5 downto 0));
end component;


component multiplier
port(
   Mnum1 : in STD_LOGIC_VECTOR (5 downto 0);
   Mnum2 : in STD_LOGIC_VECTOR (5 downto 0);
   Moutput : out STD_LOGIC_VECTOR (11 downto 0));
end component;


component multiplexer
port(
   Msum : in STD_LOGIC_VECTOR (6 downto 0);
   Msub : in STD_LOGIC_VECTOR (5 downto 0);
   Mmul : in STD_LOGIC_VECTOR (11 downto 0);
   Mslc : in STD_LOGIC_VECTOR (1 downto 0);
   MLED : out STD_LOGIC_VECTOR (11 downto 0));
end component;


begin
uut1 : adder port map(Anum1 => Sw(11 downto 6),Anum2 => Sw(5 downto 0),Aoutput =>
addition);
```

```
uut2 : subtracter port map(Snum1 => Sw(11 downto 6),Snum2 => Sw(5 downto 0),Soutput
=> subtraction);

uut3 : multiplier port map(Mnum1 => Sw(11 downto 6),Mnum2 => Sw(5 downto 0),Moutput
=> multiplication);

uut4 : multiplexer port map(Msum => addition , Msub => subtraction , Mmul =>
multiplication , Mslc => slc , MLED => LED);


end Behavioral;
```

Adder:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;


entity adder is

port(

   Anum1 : in STD_LOGIC_VECTOR (5 downto 0);

   Anum2 : in STD_LOGIC_VECTOR (5 downto 0);

   Aoutput : out STD_LOGIC_VECTOR (6 downto 0));

end adder;


architecture Behavioral of adder is


begin


   Aoutput <= ("0" & Anum1) + ("0" & Anum2);
```

```
end Behavioral;
```

Subtraction:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity subtracter is
port(
   Snum1 : in STD_LOGIC_VECTOR (5 downto 0);
   Snum2 : in STD_LOGIC_VECTOR (5 downto 0);
   Soutput : out STD_LOGIC_VECTOR (5 downto 0));
end subtracter;

architecture Behavioral of subtracter is

begin
   process (Snum1,Snum2)
   begin
      if Snum1 > Snum2 then
         Soutput <= Snum1 - Snum2;
      elsif Snum2 > Snum1 then
         Soutput <= Snum2 - Snum1;
      end if;
   end process;

end Behavioral;
```

<u>Multiplier:</u>

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;



entity multiplier is
port(
   Mnum1 : in STD_LOGIC_VECTOR (5 downto 0);
   Mnum2 : in STD_LOGIC_VECTOR (5 downto 0);
   Moutput : out STD_LOGIC_VECTOR (11 downto 0));
end multiplier;

architecture Behavioral of multiplier is

begin

Moutput <= Mnum1 * Mnum2;

end Behavioral;
```

<u>Multiplexer:</u>

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity multiplexer is
port(
   Msum : in STD_LOGIC_VECTOR (6 downto 0);
   Msub : in STD_LOGIC_VECTOR (5 downto 0);
   Mmul : in STD_LOGIC_VECTOR (11 downto 0);
   Mslc : in STD_LOGIC_VECTOR (1 downto 0);
   MLED : out STD_LOGIC_VECTOR (11 downto 0));
end multiplexer;

architecture Behavioral of multiplexer is

begin

process (Msum, Msub, Mmul, Mslc)
   begin
   if Mslc = "00" then
      MLED(5 downto 0) <= "000001";
      MLED(11 downto 6) <= "000000";
   elsif Mslc = "01" then
      MLED(6 downto 0) <= Msum;
```

```
        MLED(11 downto 7) <= "00000";
    elsif Mslc = "11" then
        MLED <= Mmul;
    elsif Mslc = "10" then
        MLED(5 downto 0) <= Msub;
        MLED(11 downto 6) <= "000000";
    end if;
end process;

end Behavioral;
```

<u>Constrants:</u>

```
# Switches
set_property PACKAGE_PIN V17 [get_ports {Sw[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw[0]}]
set_property PACKAGE_PIN V16 [get_ports {Sw[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw[1]}]
set_property PACKAGE_PIN W16 [get_ports {Sw[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw[2]}]
set_property PACKAGE_PIN W17 [get_ports {Sw[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw[3]}]
set_property PACKAGE_PIN W15 [get_ports {Sw[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw[4]}]
set_property PACKAGE_PIN V15 [get_ports {Sw[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw[5]}]
set_property PACKAGE_PIN W14 [get_ports {Sw[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw[6]}]
set_property PACKAGE_PIN W13 [get_ports {Sw[7]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Sw[7]}]
set_property PACKAGE_PIN V2 [get_ports {Sw[8]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Sw[8]}]
set_property PACKAGE_PIN T3 [get_ports {Sw[9]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Sw[9]}]
set_property PACKAGE_PIN T2 [get_ports {Sw[10]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Sw[10]}]
set_property PACKAGE_PIN R3 [get_ports {Sw[11]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Sw[11]}]


set_property PACKAGE_PIN T1 [get_ports {slc[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {slc[0]}]
set_property PACKAGE_PIN R2 [get_ports {slc[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {slc[1]}]


# LEDs
set_property PACKAGE_PIN U16 [get_ports {LED[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
set_property PACKAGE_PIN E19 [get_ports {LED[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
set_property PACKAGE_PIN U19 [get_ports {LED[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
set_property PACKAGE_PIN V19 [get_ports {LED[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]
set_property PACKAGE_PIN W18 [get_ports {LED[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED[4]}]
set_property PACKAGE_PIN U15 [get_ports {LED[5]}]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[5]}]
set_property PACKAGE_PIN U14 [get_ports {LED[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[6]}]
set_property PACKAGE_PIN V14 [get_ports {LED[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[7]}]
set_property PACKAGE_PIN V13 [get_ports {LED[8]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[8]}]
set_property PACKAGE_PIN V3 [get_ports {LED[9]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[9]}]
set_property PACKAGE_PIN W3 [get_ports {LED[10]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[10]}]
set_property PACKAGE_PIN U3 [get_ports {LED[11]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[11]}]
```