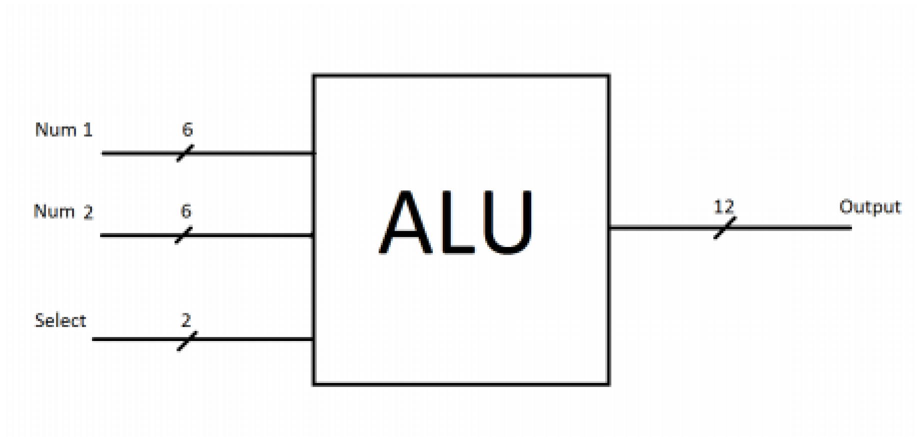
LabWork 5 ALU using VHDL
Batu Arda Düzgün
21802633
11.11.2019

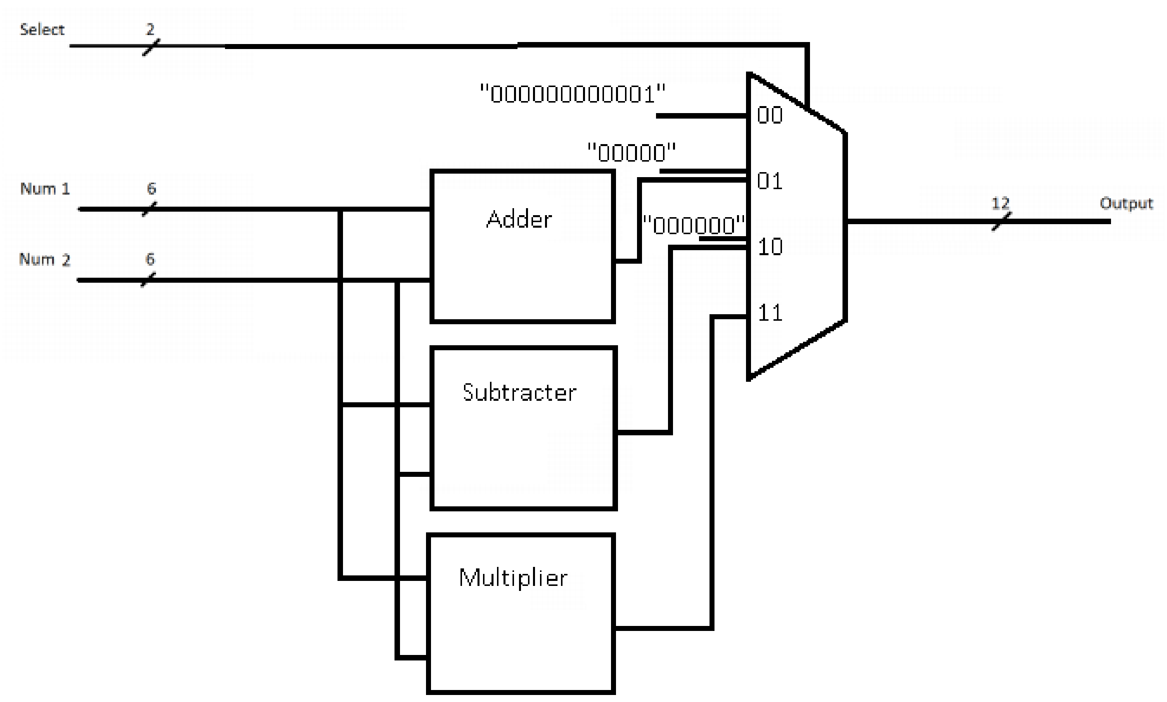
The Design Methodology:

ALU is the basic operation unit of computers. Most computers have tons of them in their CPUs. ALU does operations on two input signal based on another selection signal and gives an output. My ALU will do addition, subtraction and multiplication. I did everything as I planed in the preliminary work. To do the arithmetic operation I used IEEE.NUMERIC_STD, IEEE.STD_LOGIC_UNSIGNED, IEEE.STD_LOGIC_1164, IEEE.STD_LOGIC_ARITH. These already have logic circuits for addition, subtraction and multiplication for any length (bit) of signal. My ALU takes two 6 bit inputs from the switches 0 to 11 and it takes a 2 bit selection input from the switches 15 to 16. Then it displays its 12 bit output on the Led 0 to 11. I has a top module without any processes in it so has 3 components for the addition subtraction and multiplication. Also it has 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 added "000000" to left of its signal to increase it to 12 bits. In the addition operation I added "0" to the left of both signals and then added them to not lose the 7th bit. Then I added "00000" to the left of the signal to increase it to 12 bits. The multiplication operation already gives a 12 bit output so it didn't need any correction.



Block diagram of the ALU

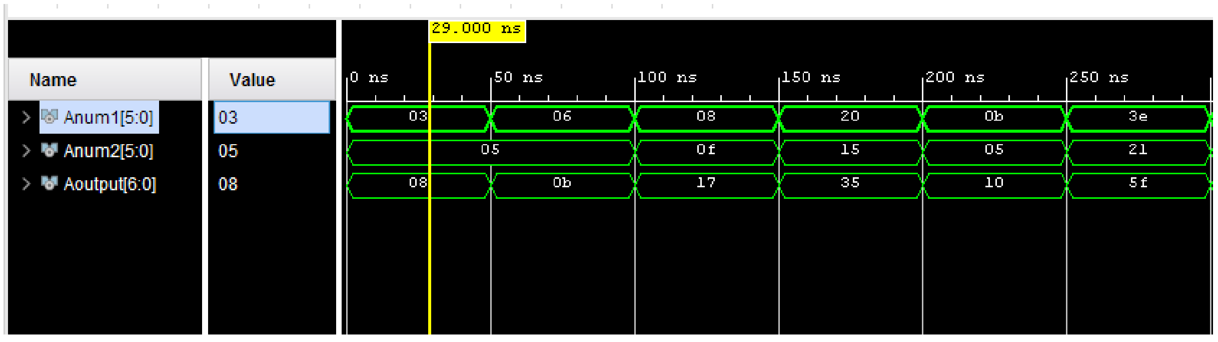
LabWork 5 ALU using VHDL
Batu Arda Düzgün
21802633
11.11.2019



Block diagram of the inside of ALU

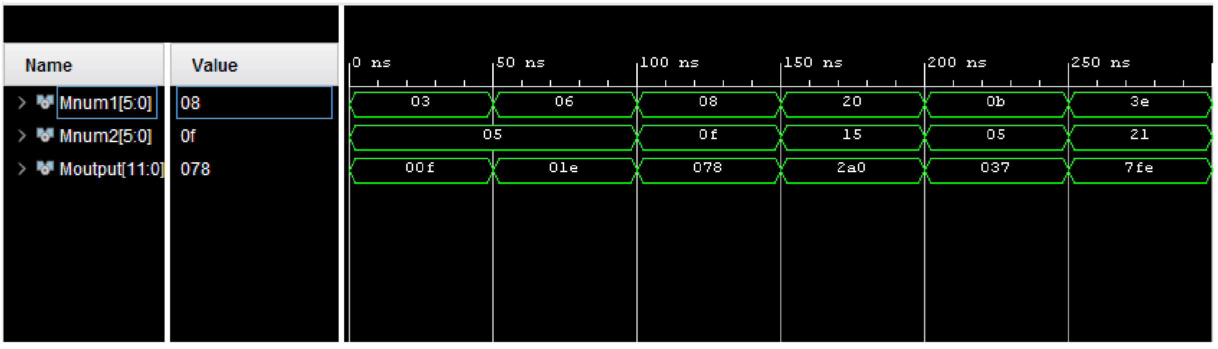
Results:

The results are what I expected. Everything works properly.



```
Name            Value   29.000 ns
Anum1[5:0]      03      03 | 06 | 08 | 20 | 0b | 3e
Anum2[5:0]      05      05      | 0f | 15 | 05 | 21
Aoutput[6:0]    08      08 | 0b | 17 | 35 | 10 | 5f
                        0 ns, 50 ns, 100 ns, 150 ns, 200 ns, 250 ns
```

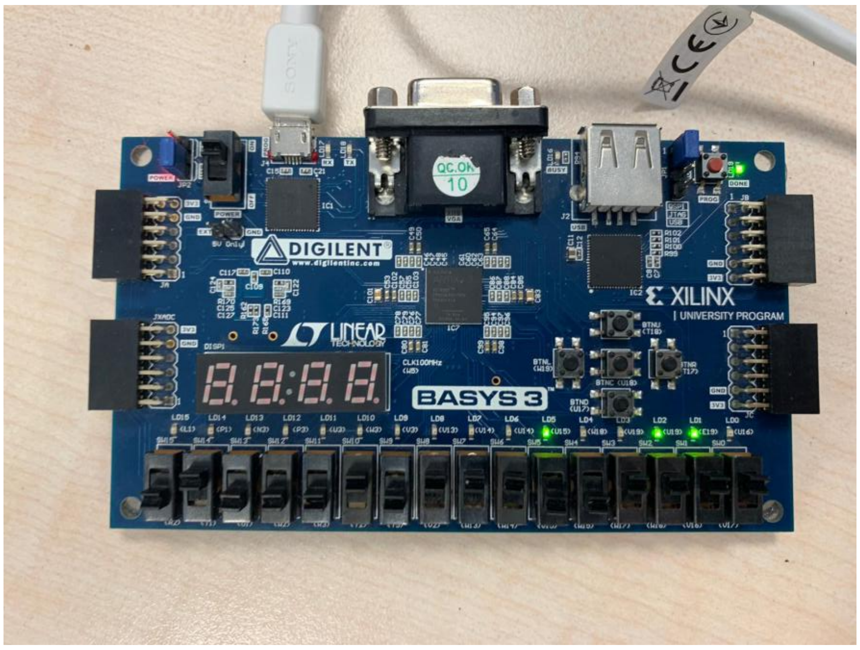
TestBench for the Adder component.



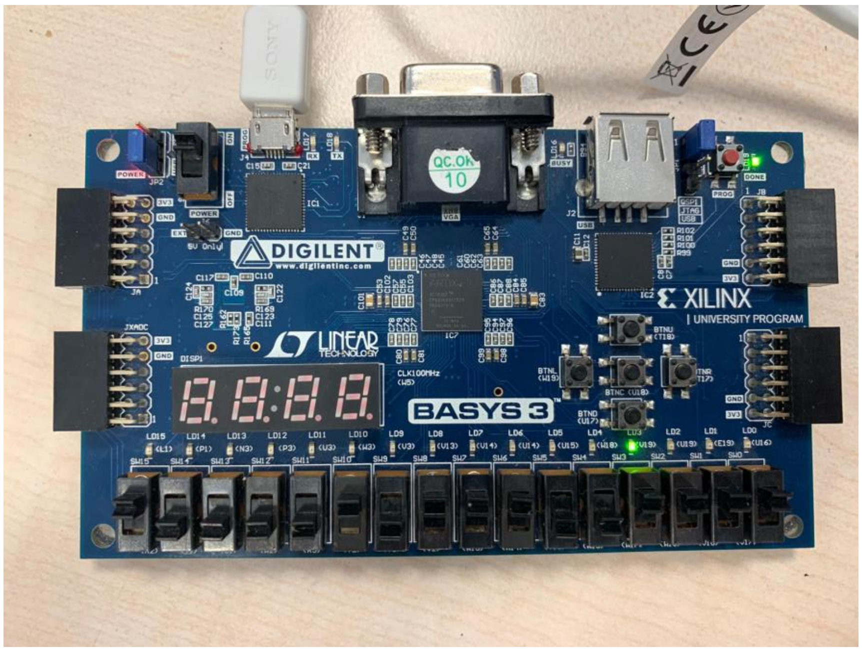
```
Name            Value
Mnum1[5:0]      08      03  | 06  | 08  | 20  | 0b  | 3e
Mnum2[5:0]      0f      05        | 0f  | 15  | 05  | 21
Moutput[11:0]   078     00f | 01e | 078 | 2a0 | 037 | 7fe
                        0 ns, 50 ns, 100 ns, 150 ns, 200 ns, 250 ns
```

TestBench for the Multiplier component.

LabWork 5 ALU using VHDL
Batu Arda Düzgün
21802633
11.11.2019

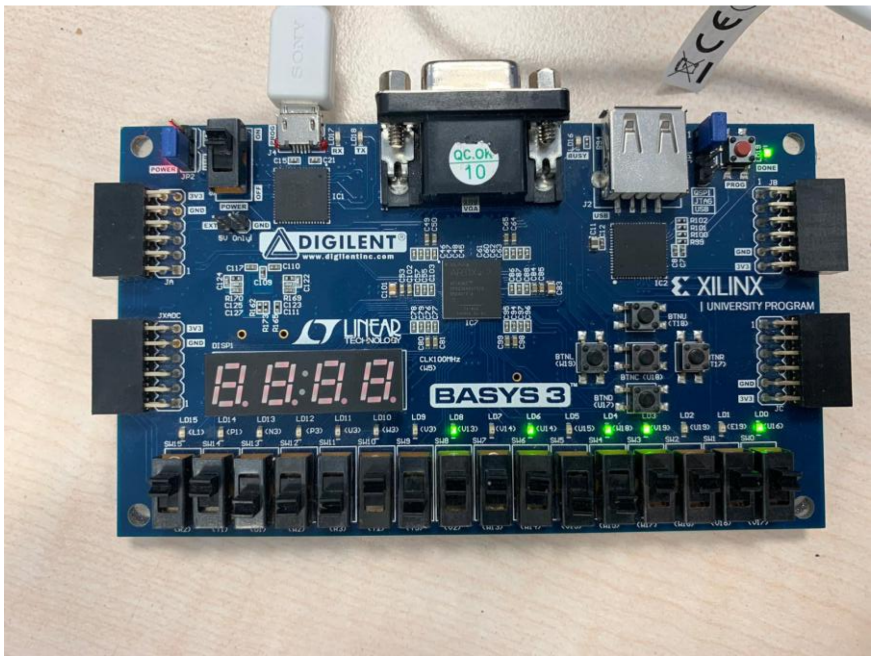


This is 15 + 23 = 38



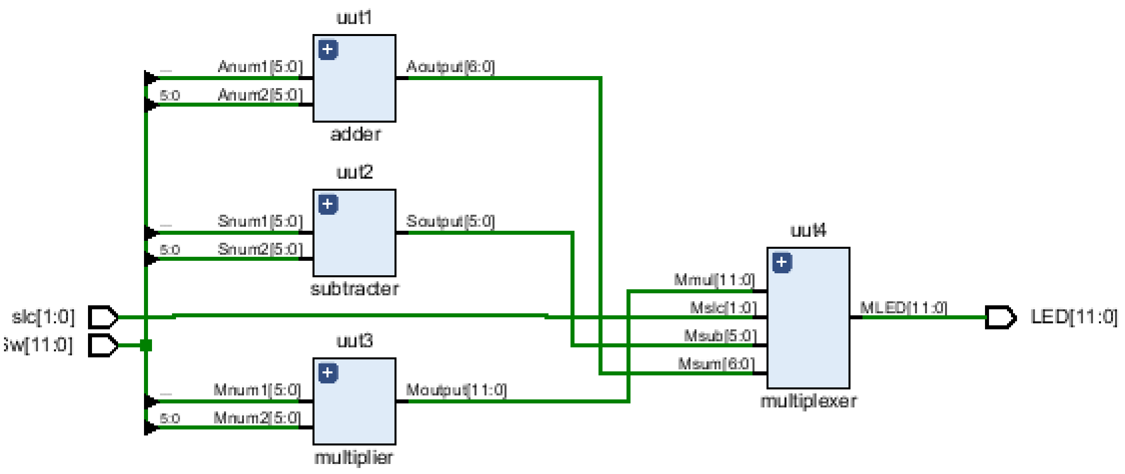
This is 23 – 15 = 8

LabWork 5 ALU using VHDL
Batu Arda Düzgün
21802633
11.11.2019



this is 15 * 23 = 345

Discussion and Conclusion:



RTL schematic looks like the Block diagram I drew. In RTL schematic the program turns my components in to closed boxes which I can press on to open and look inside of. Using IEEE.STD_LOGIC_1164, IEEE.NUMERIC_STD, IEEE.STD_LOGIC_UNSIGNED and IEEE.STD_LOGIC_ARITH was a realy good idea. I could have writen 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). I understand The FPGA board I have been using has limits but I also think I am far away from them. This was an overall really good lab for me I created one of the basic parts of a computer which felt really good. Now I feel one step closer to making a computer from nothing. I also learned how to create, join or only use one part of busses. I learned using packages which makes our lives a lot simpler.

VHDL Code:

<u>Top module:</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 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;
```

<u>adder:</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 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;
```

multiplier:

```
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;


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;
```

subtracter:

```
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;

 else

 Soutput <= "000000";

 end if;

 end process;

 end Behavioral;
```

multiplexer:

```
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]}]
```

Testbench for adder:

```
library ieee;

use ieee.std_logic_1164.all;


entity tb_adder is
```

```
end tb_adder;


architecture tb of tb_adder is


    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;


    signal Anum1   : std_logic_vector (5 downto 0);

    signal Anum2   : std_logic_vector (5 downto 0);

    signal Aoutput : std_logic_vector (6 downto 0);


begin


    dut : adder

    port map (Anum1   => Anum1,

              Anum2   => Anum2,

              Aoutput => Aoutput);


    stimuli : process

    begin

        -- EDIT Adapt initialization as needed

        Anum1 <= "000011";
        Anum2 <= "000101";

wait for 50ns;

        Anum1 <= "000110";

        Anum2 <= "000101";

wait for 50ns;

        Anum1 <= "001000";

        Anum2 <= "001111";

wait for 50ns;

        Anum1 <= "100000";

        Anum2 <= "010101";

wait for 50ns;

        Anum1 <= "001011";

        Anum2 <= "000101";

wait for 50ns;

        Anum1 <= "111110";

        Anum2 <= "100001";

wait for 50ns;

    end process;


end tb;


-- Configuration block below is required by some simulators. Usually no need to edit.


configuration cfg_tb_adder of tb_adder is

    for tb
```

```
   end for;
end cfg_tb_adder;
```

Testbench for multiplier:

```
library ieee;
use ieee.std_logic_1164.all;

entity tb_multiplier is
end tb_multiplier;

architecture tb of tb_multiplier is

   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;

   signal Mnum1   : std_logic_vector (5 downto 0);
   signal Mnum2   : std_logic_vector (5 downto 0);
   signal Moutput : std_logic_vector (11 downto 0);

begin
```

```
   dut : multiplier

   port map (Mnum1   => Mnum1,

           Mnum2   => Mnum2,

           Moutput => Moutput);


   stimuli : process

   begin

      Mnum1 <= "000011";

      Mnum2 <= "000101";

wait for 50ns;

      Mnum1 <= "000110";

      Mnum2 <= "000101";

wait for 50ns;

      Mnum1 <= "001000";

      Mnum2 <= "001111";

wait for 50ns;

      Mnum1 <= "100000";

      Mnum2 <= "010101";

wait for 50ns;

      Mnum1 <= "001011";

      Mnum2 <= "000101";

wait for 50ns;

      Mnum1 <= "111110";

      Mnum2 <= "100001";

wait for 50ns;
```

```
    end process;


end tb;


-- Configuration block below is required by some simulators. Usually no need to edit.


configuration cfg_tb_multiplier of tb_multiplier is

    for tb

    end for;

end cfg_tb_multiplier;
```