Saadet Büşra ÇAM-22201857
11.03.2024(reloaded on 15.03.2024)
EE102-1 Lab 4 Report

LAB 4: ARITHMETIC LOGIC UNIT

Purpose

The purpose of this lab is to design an arithmetic logic unit with VHDL language and Vivado, also implement it into Basys3. The switches will be inputs and LEDs will be outputs. ALU has at least 8 functions, including summation and substraction.

Methodology

Initially, eight different functions which will be implemented are determined. “arithmeticlogicunit.vhd” module, which is the top module is written. Two initial functions are determined as addition, substraction on lab manual. Other six functions are chosen as comparison,circular shift, left shift, XNOR, NAND, and NOR. For ALU, 4 bit unsigned binary numbers chosen. The implementation on basys3 investigates results.

Design Specifications

While implementing this ALU, 4-bit unsigned numbers are used and with select inputs, the chosen functions performed. Additionally, in order to make addition and subtraction operations, full adder and half adder are implemented. The following table illustrates chosen pins, functions and outputs.

Operation	Select	Input	Output
Addition	000	A,B	A+B
Substraction	001	A,B	A-B
Circular Shift	010	A	Circular Shift A(a3,a2,a1)
Comparison	011	A,B	A’, B’
Left Shift	100	A	Left Shift A(a1,a2,a3)
Bitwise Xnor	101	A,B	A xnor B
Bitwise nand	110	A,B	A nand B
Bitwise Nor	111	A,B	Not(A or B)

Table-1: Functionality Table

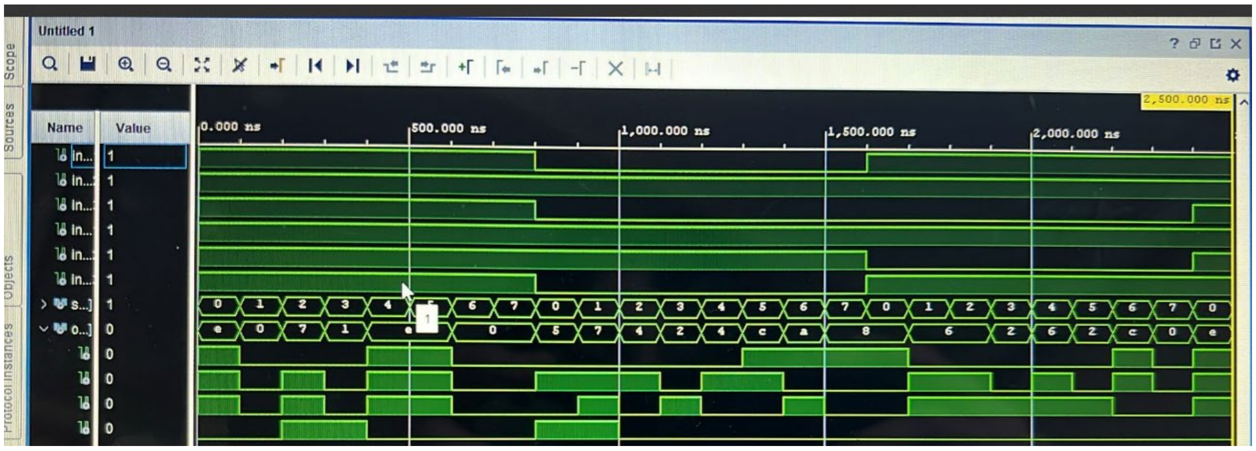

Figure-1: Simulation Result of ALU

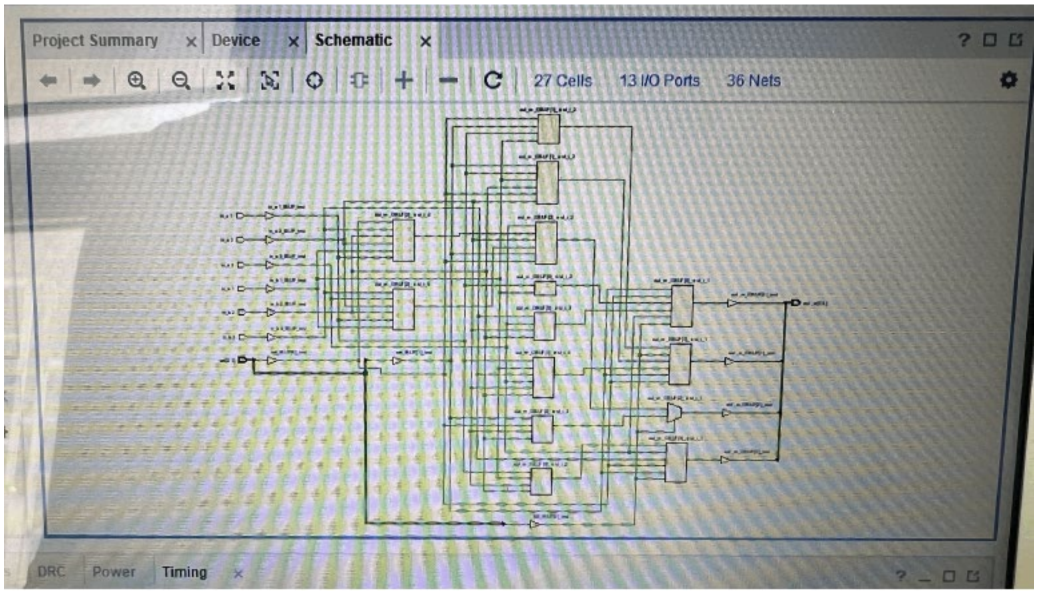

Figure-2: RTL Schematic of ALU

Pin set of circuit design is:

V17:inputpin_A1
V16: inputpin_A2
W16: inputpin_A3
W15: inputpin_B1
V15: inputpin_B2
W14: inputpin_B3
U16: out_m(0)
E19: out_m(1)
U19: out_m(2)
V19: out_m(3)

Results

RTL design for some of operations are provided in figure 11, figure 12 and figure 13. Additionally, 8 different input cases for operations and their results in basys3 board also added into figure-3 – figure-10. My results matched with expected from table-1 and the design of ALU claims the experiment was consistent.

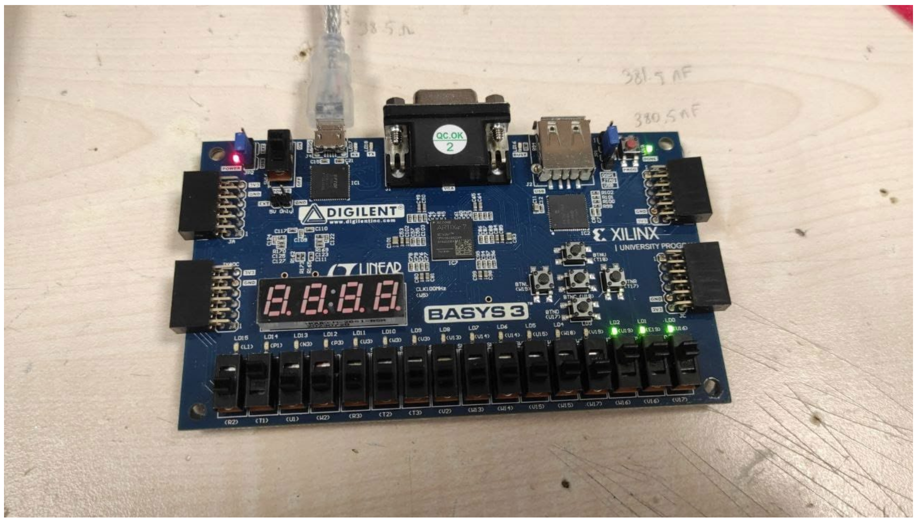

Figure 3: Addition

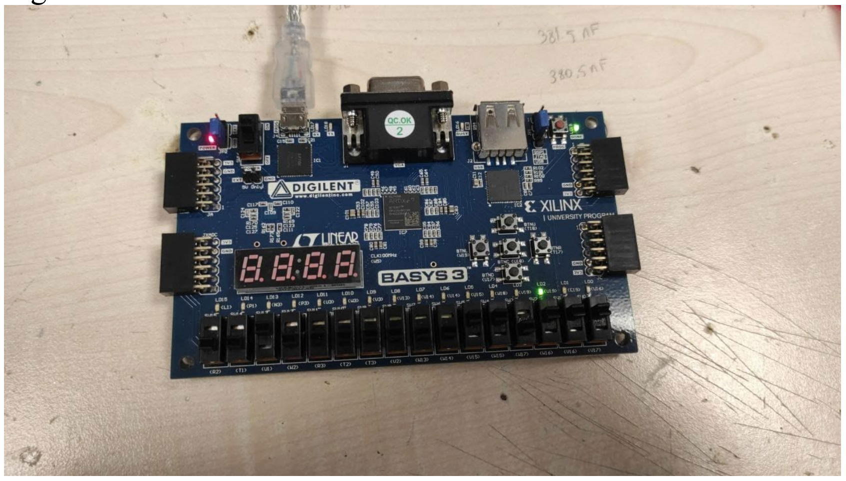

Figure-4: Substraction

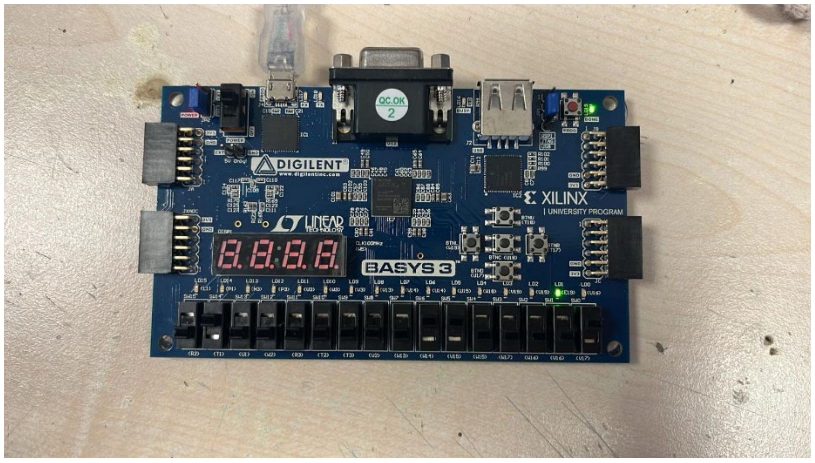
Figure-5: Circular Shift

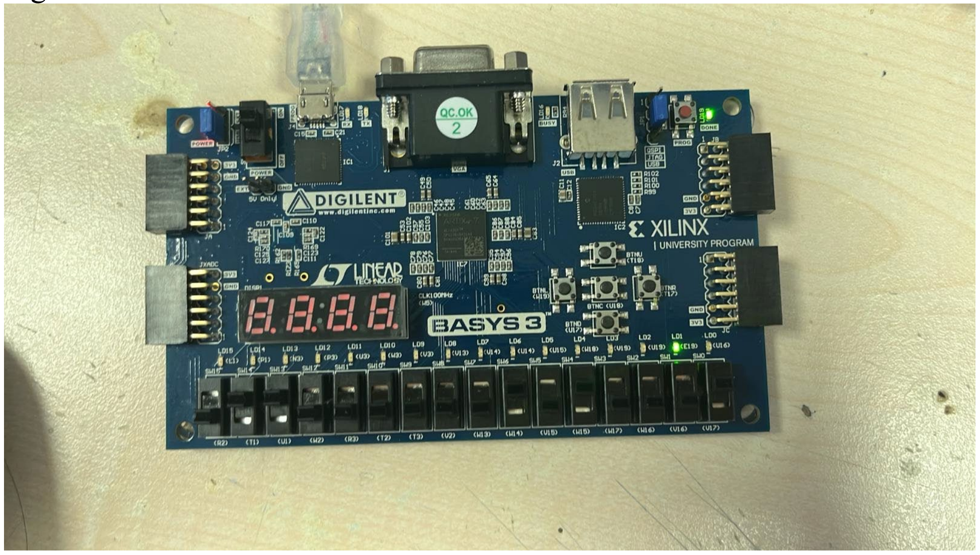
Figure-6: Comparison

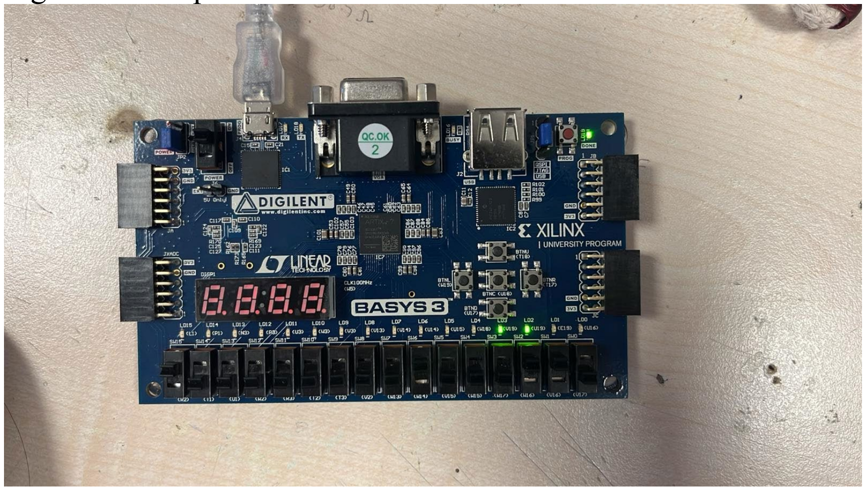
Figure-7: Left Shift

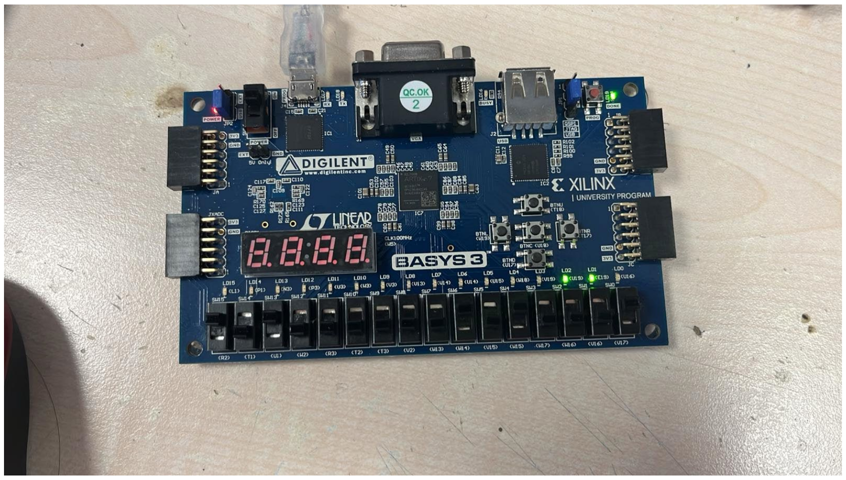
Figure-8: Bitwise XNOR

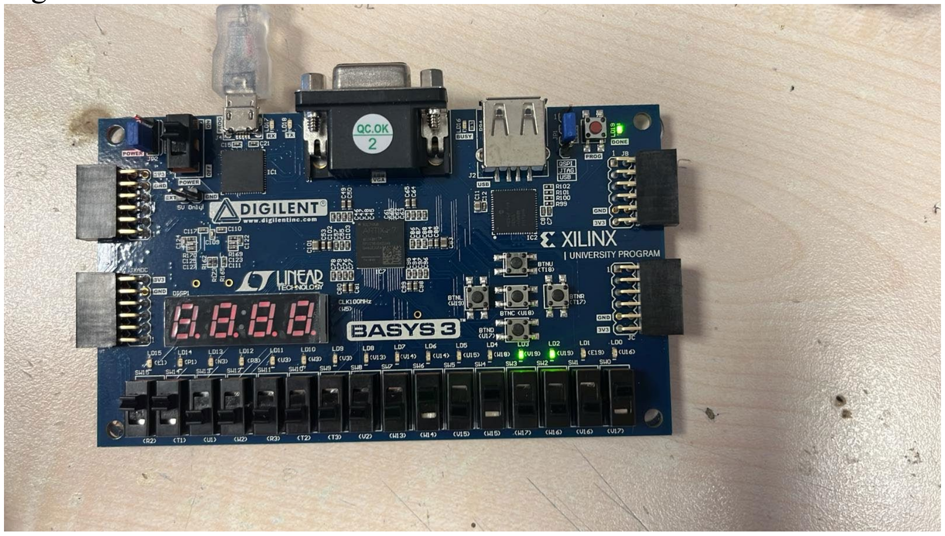
Figure-9: Bitwise Nand

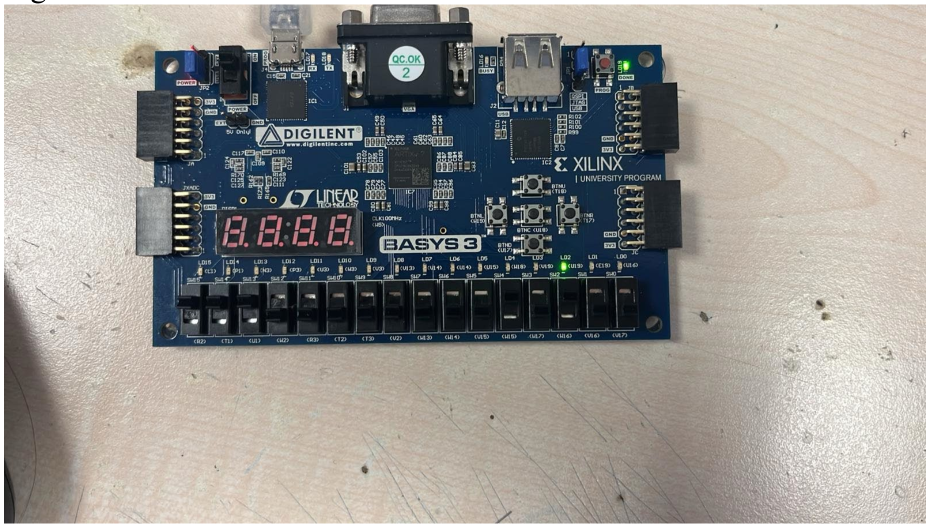
Figure-10: Bitwise Nor

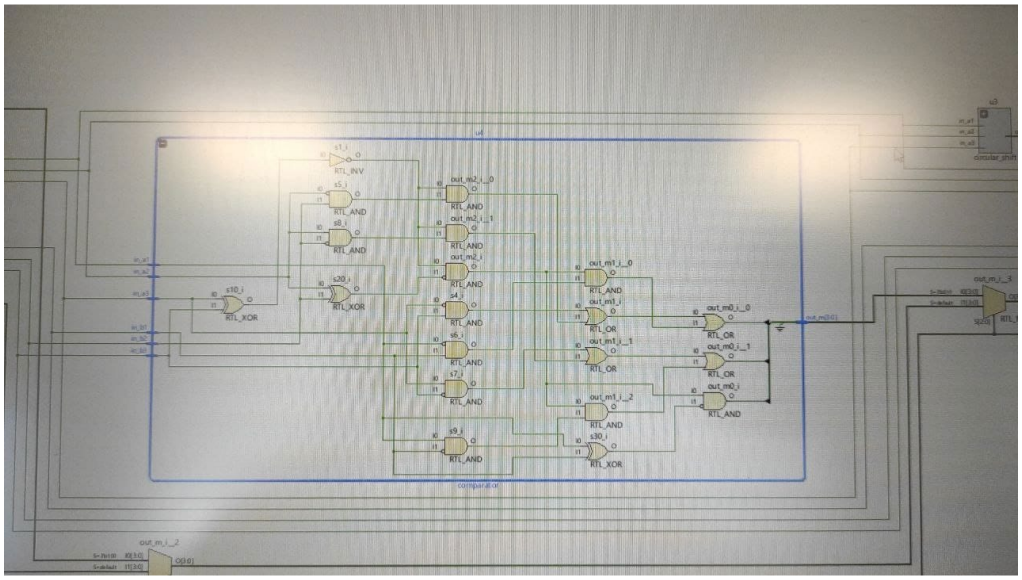

Figure-11: Comparison RTL Design

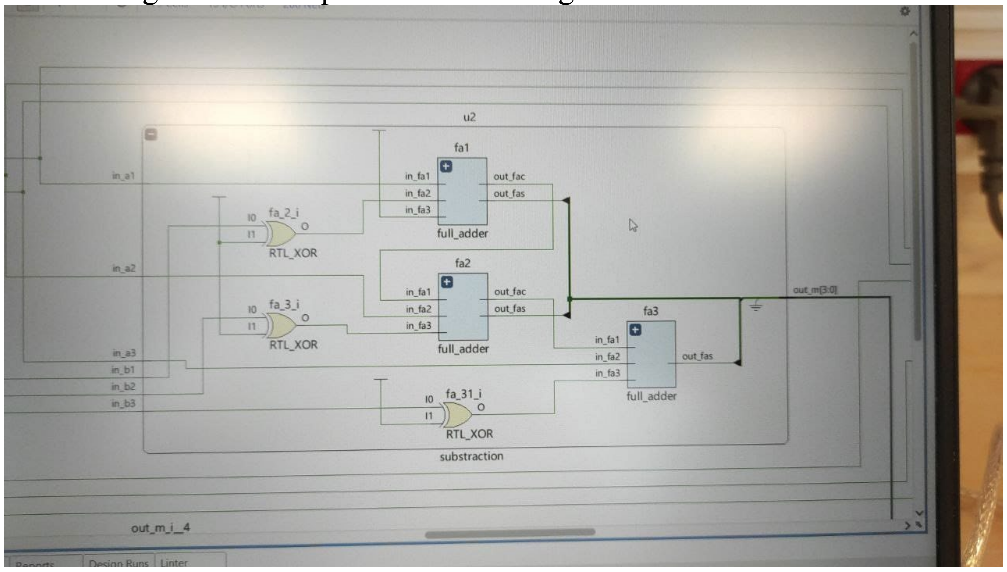

Figure-12: Substraction RTL Design

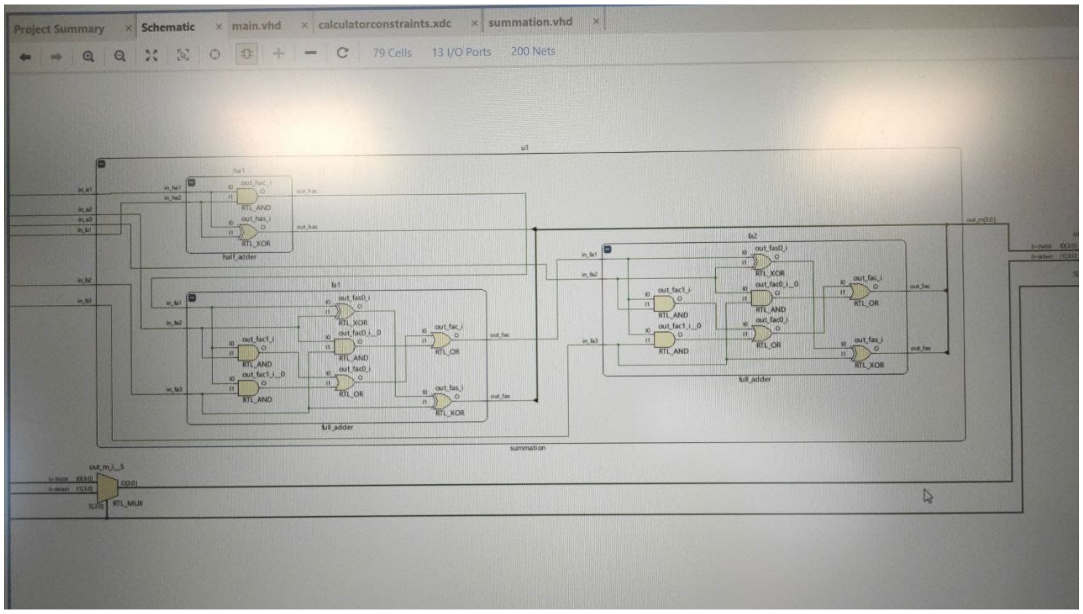

Figure-13: Summation RTL design

Conclusion

In conclusion, the ALU design is implemented via VHDL into basys3 on this lab. I get experienced on creating arithmetic logic unit using basys3, VHDL, and also get more knowledged about Vivado interface. Eight different input combinations are used and I get familiar with different operations such as substraction, shifting, addition. To do that we choose eight different functions, and 6 inputs are inputted into the design and the outputs are 3-bit. To call these operations select function is used. I also have some difficulties with connection of my basys3 board with computer, however, I fixed it. The experiment was succesfull as my results matched with truth table.

References

Hanna, K. T. (2021, August 9). *What is an arithmetic-logic unit (ALU) and how does it work?*. WhatIs. https://www.techtarget.com/whatis/definition/arithmetic-logic-unit-ALU

Wikimedia Foundation. (2024, March 6). *Arithmetic logic unit*. Wikipedia. https://en.wikipedia.org/wiki/Arithmetic_logic_unit

https://github.com/SemihAkkoc/EEE102

Appendix

Arithmeticlogicunit.vhd

```
entity arithmeticlogicunit is
```

```
   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: in
STD_LOGIC;
         sel: in std_logic_vector(2 downto 0);
         out_m: out std_logic_vector(3 downto 0));
end arithmeticlogicunit;

architecture alu_rtl of arithmeticlogicunit is

component summation
   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3 : in std_logic;
         out_m: out std_logic_vector(3 downto 0));
end component;

component substraction
   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3 : in std_logic;
         out_m: out STD_LOGIC_vector(3 downto 0));
end component;

component circularshift
   Port (inputpin_A1, inputpin_A2, inputpin_A3: in std_logic;
         out_m: out STD_LOGIC_vector(3 downto 0));
end component;

component leftshift
   Port (inputpin_A1, inputpin_A2, inputpin_A3: in std_logic;
         out_m: out STD_LOGIC_vector(3 downto 0));
end component;

component comparisonoperator
   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3 : in std_logic;
         out_m: out STD_LOGIC_vector(3 downto 0));
end component;

component xnor_gate_operator
   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3 : in std_logic;
         out_m: out STD_LOGIC_vector(3 downto 0));
end component;

component nand_gate_operator
   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3 : in std_logic;
         out_m: out STD_LOGIC_vector(3 downto 0));
end component;

component nor_gate_operator
   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3 : in std_logic;
         out_m: out STD_LOGIC_vector(3 downto 0));
end component;

signal inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: std_logic;
signal outputpin1, outputpin2, outputpin3, outputpin4, outputpin5, outputpin6, outputpin7, outputpin8:
std_logic_vector(3 downto 0);
```

```
begin

inputpinA1 <= inputpin_A1;
inputpinA2 <= inputpin_A2;
inputpinA3 <= inputpin_A3;
inputpinB1 <= inputpin_B1;
inputpinB2 <= inputpin_B2;
inputpinB3 <= inputpin_B3;

u1: summation
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        inputpin_B1 => inputpinB1,inputpin_B2 => inputpinB2, inputpin_B3 => inputpinB3,
        out_m => outputpin1);

u2:  substraction
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        inputpin_B1 => inputpinB1,inputpin_B2 => inputpinB2, inputpin_B3 => inputpinB3,
        out_m => outputpin2);

u3: circularshift
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        out_m => outputpin3);

u4: comparisonoperator
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        inputpin_B1 => inputpinB1,inputpin_B2 => inputpinB2, inputpin_B3 => inputpinB3,
        out_m => outputpin4);

u5: leftshift
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        out_m => outputpin5);

u6: xnor_gate_operator
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        inputpin_B1 => inputpinB1,inputpin_B2 => inputpinB2, inputpin_B3 => inputpinB3,
        out_m => outputpin6);

u7: nand_gate_operator
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        inputpin_B1 => inputpinB1,inputpin_B2 => inputpinB2, inputpin_B3 => inputpinB3,
        out_m => outputpin7);

u8: nor_gate_operator
port map(inputpin_A1 => inputpinA1,inputpin_A2 => inputpinA2, inputpin_A3 => inputpinA3,
        inputpin_B1 => inputpinB1,inputpin_B2 => inputpinB2, inputpin_B3 => inputpinB3,
        out_m => outputpin8);

process(sel, outputpin1, outputpin2, outputpin3, outputpin4, outputpin5, outputpin6, outputpin7,
outputpin8)
begin
   if sel = "000" then
```

```
        out_m <= outputpin1;
    elsif sel = "001" then
        out_m <= outputpin2;
    elsif sel = "010" then
        out_m <= outputpin3;
    elsif sel = "011" then
        out_m <= outputpin4;
    elsif sel = "100" then
        out_m <= outputpin5;
    elsif sel = "101" then
        out_m <= outputpin6;
    elsif sel = "110" then
        out_m <= outputpin7;
    elsif sel = "111" then
        out_m <= outputpin8;
    else
        out_m <= "0000";
    end if;
end process;
end alu_rtl;
```

Addition.vhd

```
        library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity summation is
    Port ( inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: in
STD_LOGIC;
           out_m: out STD_LOGIC_vector(3 downto 0));
end summation;
architecture Behavioral of summation is
component halfadderoperator
    Port ( in_ha1 : in STD_LOGIC;
           in_ha2 : in STD_LOGIC;
           out_has : out STD_LOGIC;
           out_hac : out STD_LOGIC);
end component;
component fulladderoperator
    Port ( in_fa1 : in STD_LOGIC;
           in_fa2 : in STD_LOGIC;
           in_fa3 : in STD_LOGIC;
           out_fas : out STD_LOGIC;
           out_fac : out STD_LOGIC);
end component;
signal hac,fac1: std_logic;
begin
      ha1:halfadderoperator
    port map(in_ha1 => inputpin_A1, in_ha2 => inputpin_B1,
             out_has => out_m(0), out_hac => hac);
    fa1:fulladderoperator
    port map(in_fa1 => hac, in_fa2 => inputpin_A2, in_fa3 => inputpin_B2,
             out_fas => out_m(1), out_fac => fac1);
    fa2:fulladderoperator
    port map(in_fa1 => fac1, in_fa2 => inputpin_A3, in_fa3 => inputpin_B3,
             out_fas => out_m(2), out_fac => out_m(3));

end Behavioral;
```

halfadderoperator.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity halfadderoperator is

    Port ( ih1 : in STD_LOGIC;

           ih2 : in STD_LOGIC;

           outs : out STD_LOGIC;

           outc: out STD_LOGIC);

end halfadderoperator;

architecture Behavioral of halfadderoperator is

begin

outs <= ih1 xor ih2;

outc <= ih1 and ih2;

end Behavioral

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
```

Fulladderoperator.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL

entity fulladderoperator is

    Port ( if1 : in STD_LOGIC;
          if2 : in STD_LOGIC;

          if3 : in STD_LOGIC;

          outfc : out STD_LOGIC;

          outfs : out STD_LOGIC);

end fulladderoperator;

archıtecture rtladder of fulladderoperator is

begin

outfs <= if1 xor if2 xor if3;

outfc <= (if1 and if2) or (if1 and if3) or (if2 and if3);

end rtladder;
```

substractor.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity substraction is
   Port ( inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: in
STD_LOGIC;
          out_m: out STD_LOGIC_vector(3 downto 0));
end substraction;
architecture Behavioral of substraction is
component full_adder
   Port ( if1 : in STD_LOGIC;
          if2 : in STD_LOGIC;
          if3 : in STD_LOGIC;
          outs : out STD_LOGIC;
          outc : out STD_LOGIC);
end component;
signal dummy,fac1,fac2,fa_2,fa_3,fa_31: std_logic;
begin
   fa_2 <= inputpin_B1 xor '1';
   fa_3 <= inputpin_B2 xor '1';
   fa_31 <= inputpin_B3 xor '1';
   fa1:full_adder
   port map(in_fa1 => inputpin_A1, in_fa2 => fa_2, in_fa3 => '1',
           out_fas => out_m(0), out_fac => fac1);
   fa2:full_adder
   port map(in_fa1 => fac1, in_fa2 => inputpin_A2, in_fa3 => fa_3,
           out_fas => out_m(1), out_fac => fac2);
   fa3:full_adder
   port map(in_fa1 => fac2, in_fa2 => inputpin_A3, in_fa3 => fa_31,
           out_fas => out_m(2), out_fac => dummy);
   out_m(3) <= '0';
end Behavioral;
```
comparatoroperator.vhd
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity comparator is
    Port ( inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: in
STD_LOGIC;
          out_m: out STD_LOGIC_vector(3 downto 0));
end comparator;
architecture Behavioral of comparatoroperator is
signal signal1, signal2, signal3,signal4,signal5,signal6,signal7,signal8,signal9: std_logic;
begin
signal1 <= not(inputpin_A3 xor inputpin_B3);
signal2 <= not(inputpin_A2 xor inputpin_B2);
signal3 <= not(inputpin_A1 xor inputpin_B1);
signal4 <= (not inputpin_A3) and inputpin_B3;
signal5 <= (not inputpin_A2) and inputpin_B2;
signal6 <= (not inputpin_A1) and inputpin_B1;
signal7 <= inputpin_A3 and (not inputpin_B3);
signal8 <= inputpin_A2 and (not inputpin_B2);
signal9 <= inputpin_A1 and (not inputpin_B1);
out_m(0) <= signal1 and signal2 and signal3;
out_m(1) <= signal4 or (signal1 and signal5) or (signal1 and signal2 and signal6);
out_m(2) <= signal7 or (signal1 and signal8) or (signal1 and signal2 and signal9);
out_m(3) <= '0';
end Behavioral;
```

leftshift.vhd
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity leftshift is
    Port ( inputpin_A1, inputpin_A2, inputpin_A3: in STD_LOGIC;
          out_m: out STD_LOGIC_vector(3 downto 0));
end leftshift;
architecture Behavıoral of leftshift is
signal A1, A2, A3: std_logic;
begin
A1<=  inputpin_A1;
A2<=  inputpin_A2;
A3<= inputpin_A1;
out_m <= A3 & A2 & A1 & '0' ;
end Behavioral;
```
xnor_gate_operator.vhd
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity xnor_gate_operator is
    Port ( inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: in
STD_LOGIC;
          out_m out STD_LOGIC_vector(3 downto 0):= "0000");
```

```
end xnor_gate_operator;
archıtecture Behavioral of xnor_gate_operator is
begin
out_m(1) <= not inputpin_A1 xor inputpin_B1);
out_m(2) <= not(inputpin_A2 xor inputpin_B2);
out_m(3) <= not(inputpin_A3 xor inputpin_B3);
end Behavioral;
```

nand_gate_operator.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity nandgate is
   Port ( inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: in
STD_LOGIC;
         out_m: out STD_LOGIC_vector(3 downto 0):= "0000");
end nand_gate_oparator;
architecture Behavioral of nand_gate_operator is
begin
out_m(1) <= not(inputpin_A1 and inputpin_B1);
out_m(2) <= not(inputpin_A2 and inputpin_B2);
out_m(3) <= not(inputpin_A3 and inputpin_B3);
end Behavioral;
```

nor_gate_operator.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity norgate is
   Port ( ( inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3 : in
STD_LOGIC;
         out_m: out STD_LOGIC_vector(3 downto 0):= "0000");
end nor_gate_operator;
architecture Behavioral of nor_gate_operator is
begin
out_m(1) <= not(inputpin_A1 or inputpin_B1);
out_m(2) <= not(inputpin_A2 or inputpin_B2);
out_m(3) <= not(inputpin_A3 or inputpin_B3);
end Behavioral;
```

constraint file:

```
set_property PACKAGE_PIN V17 [get_ports {inputpin_A1}]
set_property IOSTANDARD LVCMOS33 [get_ports {inputpin_A1}]
set_property PACKAGE_PIN V16 [get_ports {inputpin_A2}]
set_property IOSTANDARD LVCMOS33 [get_ports {inputpin_A2}]
set_property PACKAGE_PIN W16 [get_ports {inputpin_A3}]
set_property IOSTANDARD LVCMOS33 [get_ports {inputpin_A3}]
set_property PACKAGE_PIN W15 [get_ports {inputpin_B1}]
set_property IOSTANDARD LVCMOS33 [get_ports {inputpin_B1}]
set_property PACKAGE_PIN V15 [get_ports {inputpin_B2}]
set_property IOSTANDARD LVCMOS33 [get_ports {inputpin_B2}]
set_property PACKAGE_PIN W14 [get_ports {inputpin_B3}]
set_property IOSTANDARD LVCMOS33 [get_ports {inputpin_B3}]
set_property PACKAGE_PIN U1 [get_ports {sel[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sel[0]}]
set_property PACKAGE_PIN T1 [get_ports {sel[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sel[1]}]
set_property PACKAGE_PIN R2 [get_ports {sel[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sel[2]}]
set_property PACKAGE_PIN U16 [get_ports {out_m[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out_m[0]}]
set_property PACKAGE_PIN E19 [get_ports {out_m[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out_m[1]}]
set_property PACKAGE_PIN U19 [get_ports {out_m[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out_m[2]}]
set_property PACKAGE_PIN V19 [get_ports {out_m[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out_m[3]}]
```

Alutestbench:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity testBench_main is

end testBench_main;

architecture Behavioral of testBench_main is

component alu_main

   Port (inputpin_A1, inputpin_A2, inputpin_A3, inputpin_B1, inputpin_B2, inputpin_B3: in
STD_LOGIC;

          sel: in std_logic_vector(2 downto 0);

          out_m: out STD_LOGIC_vector(3 downto 0));

end component;

signal sel: std_logic_vector(2 downto 0);

signal out_m: std_logic_vector(3 downto 0);

begin

u1: alu_main

Port map(inputpin_A1 => in_a1, inputpin_A2 => in_a2, inputpin_A3 => in_a3, inputpin_B1 =>
in_b1, inputpin_B2 => in_b2, inputpin_B3 => in_b3, sel => sel, out_m =>out_m);
```

```
testBench_main: process
begin
      sel <= "000";
        inputpin_A1 <= '1';
        inputpin_A2 <= '1';
        inputpin_A3 <= '1';
        inputpin_B1 <= '1';
       inputpin_B2 <= '1';
        inputpin_B3 <= '1';
        wait for 100ns;

        sel <= "001";
        inputpin_A1 <= '1';
        inputpin_A2 <= '1';
        inputpin_A3 <= '1';
        inputpin_B1 <= '1';
       inputpin_B2 <= '1';
        inputpin_B3 <= '1';
        wait for 100ns;

        sel <= "010";
        inputpin_A1 <= '1';
```

```
    inputpin_A2 <= '1';

    inputpin_A3 <= '1';

    inputpin_B1 <= '1';

   inputpin_B2 <= '1';

    inputpin_B3 <= '1';


wait for 100ns;

    sel <= "011";

    inputpin_A1 <= '1';

    inputpin_A2 <= '1';

    inputpin_A3 <= '1';

    inputpin_B1 <= '1';

   inputpin_B2 <= '1';

    inputpin_B3 <= '1';


wait for 100ns;

    sel <= "100";

    inputpin_A1 <= '1';

    inputpin_A2 <= '1';

    inputpin_A3 <= '1';

    inputpin_B1 <= '1';

   inputpin_B2 <= '1';

    inputpin_B3 <= '1';
```

```
wait for 100ns;
    sel <= "101";
    inputpin_A1 <= '1';
    inputpin_A2 <= '1';
    inputpin_A3 <= '1';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '1';

wait for 100ns;
    sel <= "110";
    inputpin_A1 <= '1';
    inputpin_A2 <= '1';
    inputpin_A3 <= '1';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '1';

wait for 100ns;
    sel <= "111";
    inputpin_A1 <= '1';
    inputpin_A2 <= '1';
    inputpin_A3 <= '1';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '1';

wait for 100ns;
    sel <= "000";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';

wait for 100ns;
    sel <= "001";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';
```

```
 wait for 100ns;
    sel <= "010";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';

wait for 100ns;
    sel <= "011";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';

wait for 100ns;
    sel <= "100";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';

wait for 100ns;
    sel <= "101";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';

wait for 100ns;
    sel <= "110";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';

wait for 100ns;
```

```
    sel <= "111";
    inputpin_A1 <= '0';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '1';
    inputpin_B3 <= '0';

wait for 100ns;
    sel <= "000";
    inputpin_A1 <= '1';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '0’;
    inputpin_B3 <= '1';

wait for 100ns;
    sel <= "001";
    inputpin_A1 <= '1';
    inputpin_A2 <= '1';
    inputpin_A3 <= '0';
    inputpin_B1 <= '1';
   inputpin_B2 <= '0’;

    inputpin_B3 <= '1';


 wait for 100ns;

    sel <= "010";

    inputpin_A1 <= '1';

    inputpin_A2 <= '1';

    inputpin_A3 <= '0';

    inputpin_B1 <= '1';

   inputpin_B2 <= '0’;

    inputpin_B3 <= '1';


wait for 100ns;

    sel <= "011";

    inputpin_A1 <= '1';

    inputpin_A2 <= '1';

    inputpin_A3 <= '0';

    inputpin_B1 <= '1';

   inputpin_B2 <= '0’;

    inputpin_B3 <= '1';


wait for 100ns;

    sel <= "100";
    inputpin_A1 <= '1';

    inputpin_A2 <= '1';

    inputpin_A3 <= '0';

    inputpin_B1 <= '1';

   inputpin_B2 <= '0’;

    inputpin_B3 <= '1';


wait for 100ns;

    sel <= "101";

    inputpin_A1 <= '1';

    inputpin_A2 <= '1';

    inputpin_A3 <= '0';

    inputpin_B1 <= '1';

   inputpin_B2 <= '0’;

    inputpin_B3 <= '1';


wait for 100ns;

    sel <= "110";

    inputpin_A1 <= '1';

    inputpin_A2 <= '1';

    inputpin_A3 <= '0';

    inputpin_B1 <= '1';

   inputpin_B2 <= '0’;
```

```
        inputpin_B3 <= '1';



    wait for 100ns;

        sel <= "111";

        inputpin_A1 <= '1';

        inputpin_A2 <= '1';

        inputpin_A3 <= '0';

        inputpin_B1 <= '1';

       inputpin_B2 <= '0’;

        inputpin_B3 <= '1';


   wait for 100ns;

end process;

end Behavioral;
```