Report: Arithmetic Logic Unit (ALU)

Bilkent University Electrical and Electronics Department

EE102-01 Lab 4

Melih Ege Doğan – 22403329
March 3, 2025

Purpose:

Purpose of this lab was to study Arithmetic Logic Unit (ALU) by creating one of them using Basys3 FPGA board. This ALU can do 8 different operations including 4-bit adder, 4-bit subtractor, bitwise operations and, shifting operations.

Methodology:

First, we had to choose 8 different operations to implement in our design. According to the lab manual these operations should include 4-bit adder, 4-bit subtractor, at least 1 shifting operation and at least 1 bitwise operation. For this lab I used 4-bit adder, 4-bit subtractor, bitwise AND, bitwise NAND, bitwise NOR, bitwise OR, bitwise XOR and, rotational left shift. We also used LEDs on the Basys board to show outputs.

Design Specifications:

For this ALU we used 4-bit inputs. To select between these 4-bit operations we created a select input. Inputs and Outputs can be seen in the Table 1.

```
sel: in std_logic_vector  3-bit long (select inputs)
x: in std_logic_vector 4-bit long (First input)
y: in std_logic_vector 4-bit long (Second input)
cout: out std_logic (Carry Output)
s: out std_logic_vector 4-bit long (Outputs)
```

Table 1: Inputs/Outputs

We created the main.vhd code to create a module that holds all the operations under itself and can switch between these operations. 8 modules under the main function can be seen as following:

- fourbitadder.vhd
- fourbitsub.vhd
- bitwiseand.vhd
- bitwisenand.vhd
- bitwisenor.vhd
- bitwiseor.vhd
- bitwisexor.vhd
- rotateleftshift.vhd

Additionally, we created a full adder module to create 4-bit adder and 4-bit subtractor. And a half adder module to create the full adder. Since there may be an overflow, we created a carry output to observe the overflow. Table 2 shows which operations selected with select inputs.

Select	Operation	Input	Output
000	4-bit adder	X, Y	X + Y
001	4-bit subtractor	X, Y	X - Y
010	bitwiseand	X, Y	X AND Y
011	bitwisenand	X, Y	X NAND Y
100	bitwisenor	X, Y	X NOR Y
101	bitwiseor	X, Y	X OR Y
110	bitwisexor	X, Y	X XOR Y
111	Rotational left shift	X	Left shifted X

Results:

We created a test bench code to simulate our design. When we checked the simulation from test bench, we observed that simulations were expected values. You can see the test bench simulation in Figure 1.

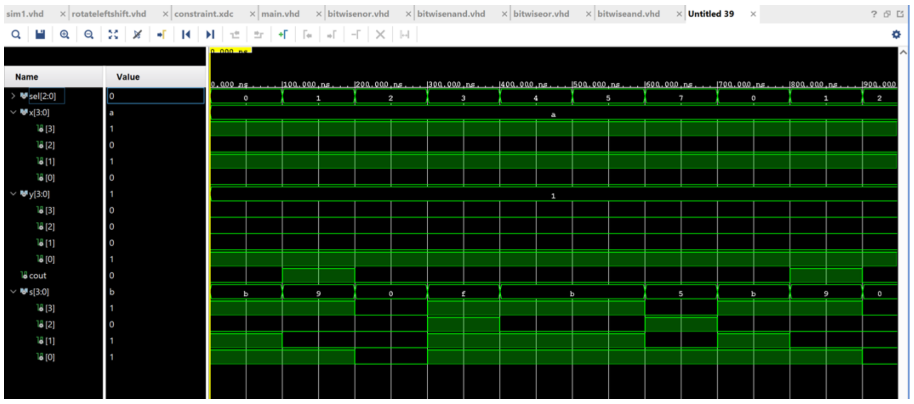


Figure 1: Test bench simulation.

And we created a schematic for the main module (Figure 2). We observed all 8 modules in the schematic as we expected.

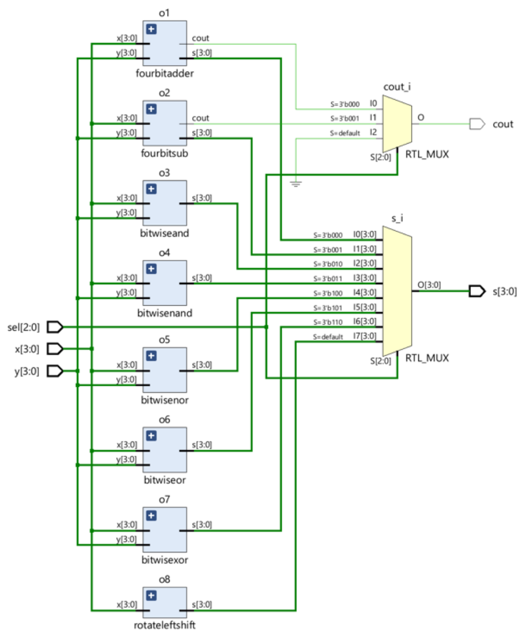
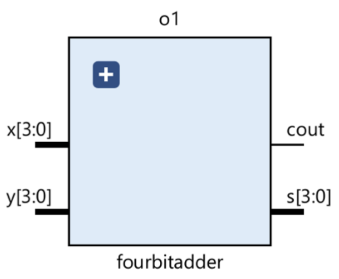
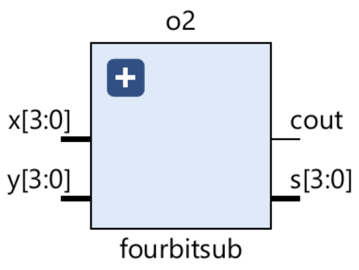


Figure 2: RTL schematic of the main module.

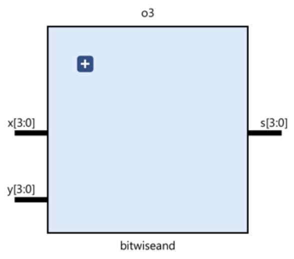
You can see that the main module is working like a multiplexer. We also checked every modules schematic one by one (Figure 3).



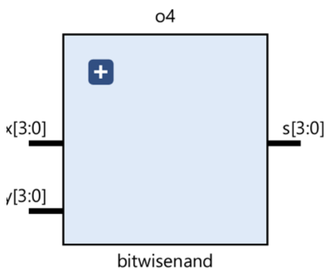
a) 4-Bit Adder



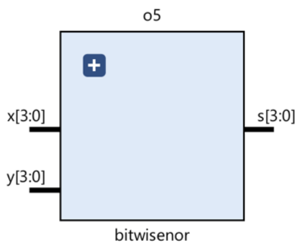
b) 4-Bit Subtractor



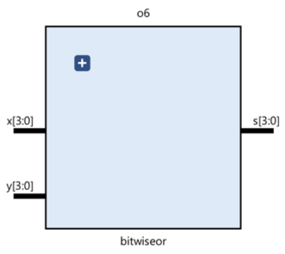
c) Bitwiseand



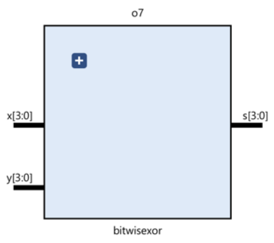
d) Bitwisenand



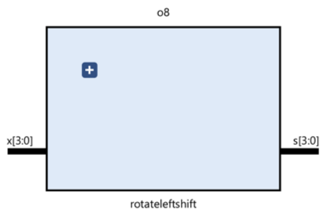
a) Bitwisenor



b) Bitwiseor



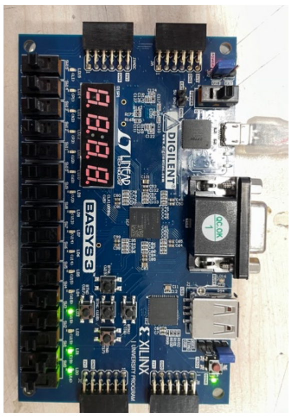
a) Bitwisexor

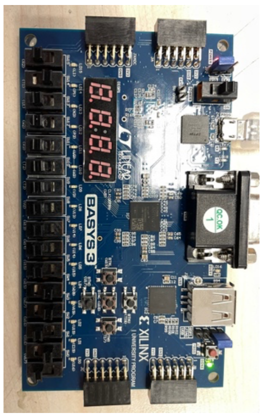


b) Rotate Left Shift

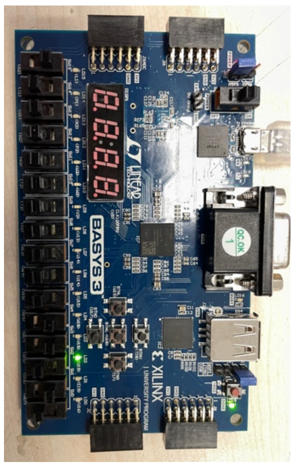
Figure 3: Sub-Modules.

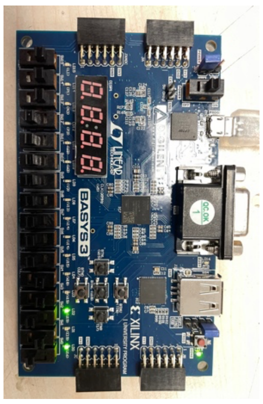
When we observed the outputs in the FPGA board, we found that they are the same with test bench simulations as we expected (Figure 4).


a) 4-Bit adder


b) Bitwiseand

c) Bitwisenand


d) Bitwisenor


e) Rotational Left Shift

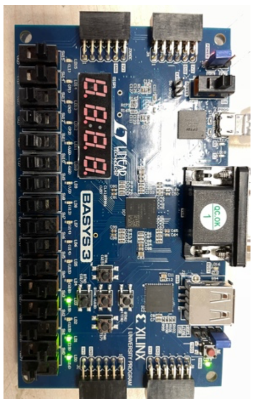

f) Bitwisexor

Figure 4: Some of the outputs.

Conclusion:

In this lab, we successfully designed, simulated, and implemented a circuit that performs eight different mathematical and logical operations, including addition, subtraction, shifting and, bitwise operations. We used a selector input to choose between these operations. Our tests, both in simulation and on the physical hardware, showed the circuit produced the correct results, confirming our design worked as intended. This project gave us practical experience in building a fundamental digital circuit.

References:

- https://github.com/SemihAkkoc/EEE102
- https://en.wikipedia.org/wiki/Arithmetic_logic_unit
- https://www.youtube.com/watch?v=rDVTplzv7P4&ab_channel=ExploreElectronics

Appendices:

Code 1: (main.vhd)

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity main is
    Port (
    sel: in std_logic_vector(2 downto 0);
    x,y: in std_logic_vector(3 downto 0);
    cout: out std_logic;
    s: out std_logic_vector (3 downto 0)
     );
end main;

architecture Behavioral of main is

component bitwisexor
Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end component;

component bitwiseor
Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end component;

component bitwisenor
Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end component;

component bitwisenand
Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end component;
```

```vhdl
component bitwiseand
Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end component;

component rotateleftshift
Port (x: in std_logic_vector(3 downto 0);
        s: out std_logic_vector(3 downto 0)
     );
end component;

component fourbitadder
Port ( x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0);
        cout: out std_logic
  );
end component;

component fourbitsub
Port ( x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0);
        cout: out std_logic
  );
end component;

SIGNAL s1: std_logic_vector(3 downto 0);
SIGNAL s2: std_logic_vector(3 downto 0);
SIGNAL s3: std_logic_vector(3 downto 0);
SIGNAL s4: std_logic_vector(3 downto 0);
SIGNAL s5: std_logic_vector(3 downto 0);
SIGNAL s6: std_logic_vector(3 downto 0);
SIGNAL s7: std_logic_vector(3 downto 0);
SIGNAL s8: std_logic_vector(3 downto 0);

SIGNAL c1: std_logic;
SIGNAL c2: std_logic;


begin
o1 : fourbitadder PORT MAP (x, y, s1,c1);
o2 : fourbitsub PORT MAP (x, y, s2,c2);
o3 : bitwiseand PORT MAP (x, y,s3);
o4 : bitwisenand PORT MAP (x, y,s4);
o5 : bitwisenor PORT MAP (x, y,s5);
o6 : bitwiseor PORT MAP (x, y,s6);
o7 : bitwisexor PORT MAP (x, y,s7);
o8 : rotateleftshift PORT MAP (x,s8);

with sel select
```

```
    s <= s1 when "000",
    s2 when "001",
    s3 when "010",
    s4 when "011",
    s5 when "100",
    s6 when "101",
    s7 when "110",
    s8 when others;

with sel select
        cout <= c1 when "000",  -- Carry-out from adder
                c2 when "001",  -- Carry-out from subtractor
                '0' when others; -- No carry-out for other operations


end Behavioral;
```

Code 2: (sim1.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity sim1 is
--  Port ( );
end sim1;

architecture Behavioral of sim1 is
component main
Port (
sel: in std_logic_vector(2 downto 0);
x,y: in std_logic_vector(3 downto 0);
cout: out std_logic;
s: out std_logic_vector (3 downto 0)
);
end component;
SIGNAL sel: std_logic_vector(2 downto 0);
SIGNAL x,y: std_logic_vector(3 downto 0);
SIGNAL cout: std_logic;
SIGNAL s: std_logic_vector (3 downto 0);

begin
UUT: main PORT MAP(
sel => sel,
x => x,
y => y,
cout => cout,
s => s

);

testbench: process
begin
    sel<="000";
    x<="1010";
    y<="0001";
    wait for 100ns;

    sel<="001";
    x<="1010";
    y<="0001";
    wait for 100ns;

    sel<="010";
    x<="1010";
    y<="0001";
    wait for 100ns;

    sel<="011";
    x<="1010";
    y<="0001";
    wait for 100ns;

    sel<="100";
    x<="1010";
    y<="0001";
    wait for 100ns;

    sel<="101";
    x<="1010";
    y<="0001";
    wait for 100ns;

    sel<="111";
    x<="1010";
    y<="0001";
    wait for 100ns;



end process;

end Behavioral;
```

Code 3: (fourbitadder.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fourbitadder is
  Port ( x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0);
        cout: out std_logic
  );
end fourbitadder;

architecture Behavioral of fourbitadder is
signal sign: std_logic_vector(2 downto 0);

component fulladder
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           cin : in STD_LOGIC;
           s : out STD_LOGIC;
           cout : out STD_LOGIC);

end component;
begin
FA1: fulladder PORT MAP(x(0),y(0),'0',s(0),sign(0));
FA2: fulladder PORT MAP(x(1),y(1),sign(0),s(1),sign(1));
FA3: fulladder PORT MAP(x(2),y(2),sign(1),s(2),sign(2));
FA4: fulladder PORT MAP(x(3),y(3),sign(2),s(3),cout);

end Behavioral;
```

Code 4: (fourbitsub.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
```

```vhdl
--library UNISIM;
--use UNISIM.VComponents.all;

entity fourbitsub is
Port ( x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0);
        cout: out std_logic
  );
end fourbitsub;



architecture Behavioral of fourbitsub is
signal sign: std_logic_vector(2 downto 0);
signal reverse_y: std_logic_vector(3 downto 0);

component fulladder
    Port ( x : in STD_LOGIC;
               y : in STD_LOGIC;
               cin : in STD_LOGIC;
               s : out STD_LOGIC;
               cout : out STD_LOGIC);

end component;
begin
reverse_y <= not y;

FA1: fulladder PORT MAP(x(0),reverse_y(0),'1',s(0),sign(0));
FA2: fulladder PORT MAP(x(1),reverse_y(1),sign(0),s(1),sign(1));
FA3: fulladder PORT MAP(x(2),reverse_y(2),sign(1),s(2),sign(2));
FA4: fulladder PORT MAP(x(3),reverse_y(3),sign(2),s(3),cout);

end Behavioral;
```

Code 5: (bitwiseand.vhd)

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bitwiseand is
  Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
```

```
end bitwiseand;

architecture Behavioral of bitwiseand is

begin
        s <= x and y;
end Behavioral;
```

Code 6: (bitwisenand.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bitwisenand is
    Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end bitwisenand;

architecture Behavioral of bitwisenand is

begin
        s <= x nand y;
end Behavioral;
```

Code 7: (bitwisenor.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bitwisenand is
    Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
```

```
  );
end bitwisenand;

architecture Behavioral of bitwisenand is

begin
        s <= x nand y;
end Behavioral;
```

Code 8: (bitwiseor.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bitwiseor is
    Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end bitwiseor;

architecture Behavioral of bitwiseor is

begin
        s <= x or y;


end Behavioral;
```

Code 9: (bitwisexor.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bitwisexor is
```

```
    Port (x,y: in std_logic_vector (3 downto 0);
        s: out std_logic_vector (3 downto 0)
  );
end bitwisexor;

architecture Behavioral of bitwisexor is

begin
process is
begin
    for a in 0 to 3 loop
        s <= x xor y;
    end loop;
    wait;
end process;

end Behavioral;
```

Code 10: (rotateleftshift.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity rotateleftshift is
    Port (x: in std_logic_vector(3 downto 0);
        s: out std_logic_vector(3 downto 0)
     );
end rotateleftshift;

architecture Behavioral of rotateleftshift is

begin

s(3 downto 1) <= x(2 downto 0);
    s(0)          <= x(3);

end Behavioral;
```

Code 11: (fulladder.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fulladder is
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           cin : in STD_LOGIC;
           s : out STD_LOGIC;
           cout : out STD_LOGIC);
end fulladder;

architecture Behavioral of fulladder is

signal sign : std_logic_vector (2 downto 0);

component halfadder
Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           s : out STD_LOGIC;
           c : out STD_LOGIC);
end component;

begin

adder1: halfadder PORT MAP (x,y,sign(0),sign(1));
adder2: halfadder PORT MAP (cin,sign(0),s,sign(2));

cout <= sign(1) or sign(2);

end Behavioral;
```

Code 12: (halfadder.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity halfadder is
```

```
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           s : out STD_LOGIC;
           c : out STD_LOGIC);
end halfadder;

architecture Behavioral of halfadder is

begin

s <= x xor y;
c <= x and y;


end Behavioral;
```