13.03.2024 Emir Arda Bayer
22201832-4

Lab 4 - Arithmetic Logic Unit

Purpose

The goal of this lab is to get students knowledgeable about arithmetic logic units which have the capability of calculating mathematical operations by obtaining designs of logical operations such as adders, subtractors and more bitwise and bitshift operations.

Design

A, B are 4 bit binary inputs and K is like a key, a 3 bit input for picking which operations to use. As the value of K increases from 000 to 111 operations of addition, subtraction, and gate, or gate, xor gate, rotation, logical shift and rotation through carry are selected.

Methodology and Results

I created a main VHDL file for designing the complete unit initially, and written the inputs and outputs of the unit, also while creating every operations' components within the architecture section. The operations were addition, subtraction, and gate, or gate, xor gate, rotation, logical shift and rotation through carry, and each of their inputs and outputs were selected. I later assigned each operation to a 3 bit value for later being selected on the FPGA for calculations. Then I created separate design sources for every operation.

Addition (000) and subtraction (100) were quite similar in the basis, they each involved 1 bit full-adders for making 4 bit addition operations, and the only difference was that the subtraction section had a part where the second number's two's complement was taken as the second number to be added. There was an initial carry of '0' on each of these operations.

Then I created sources for and (001), or (010) and xor (100) indicating the input and outputs from the main file and their behaviors. Later I created a source for rotate operation (left, 101) that involved taking the value of MSB to LSB and moving each other bit one bit left. The next one was logical shift (right, 110) which moved every bit one unit to the right, also assigning the MSB '0'. The last one was rotate through carry (111) which took a 4 bit binary and a 1 bit carry to place the carry to the LSB and place every other bit one unit to the left. I assigned the initial carry to be '1'.

Then I created the RTL Schematic of the system, as I also created the constraint file, and observed the system as simulated via the test bench I created. As the results were as desired, I generated a bitstream and observed the same system on the FPGA.

13.03.2024

Emir Arda Bayer
22201832-4

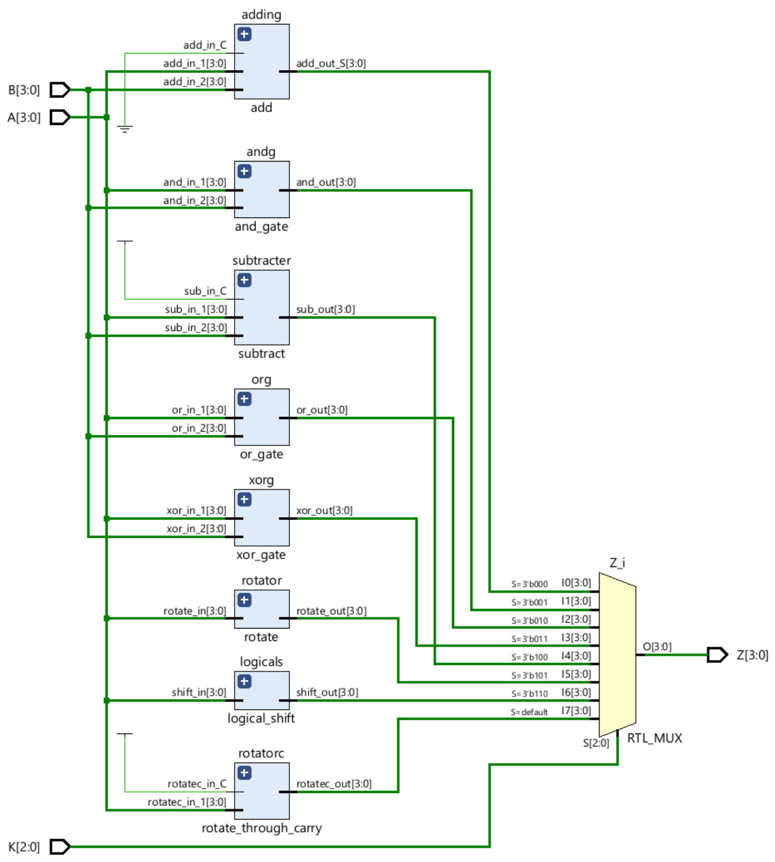

Figure 1: RTL Schematic

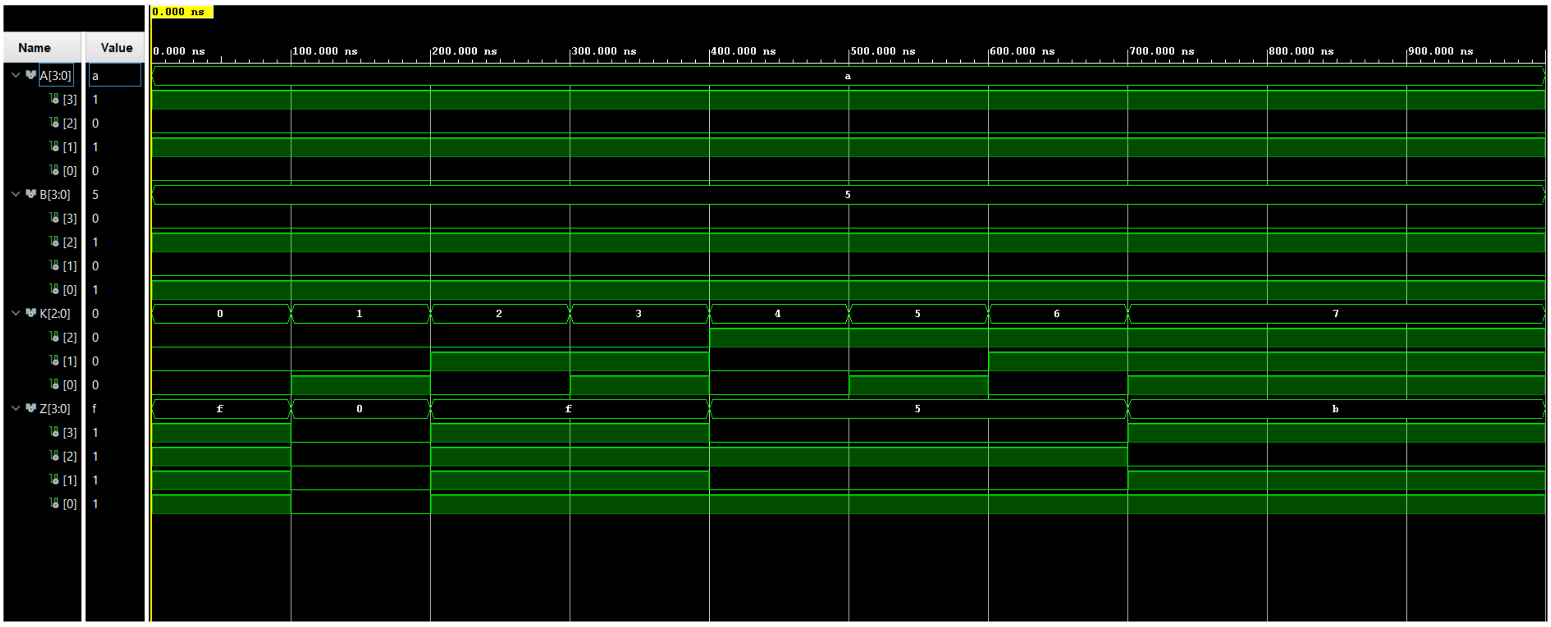

Figure 2: Behavioral Simulation Results

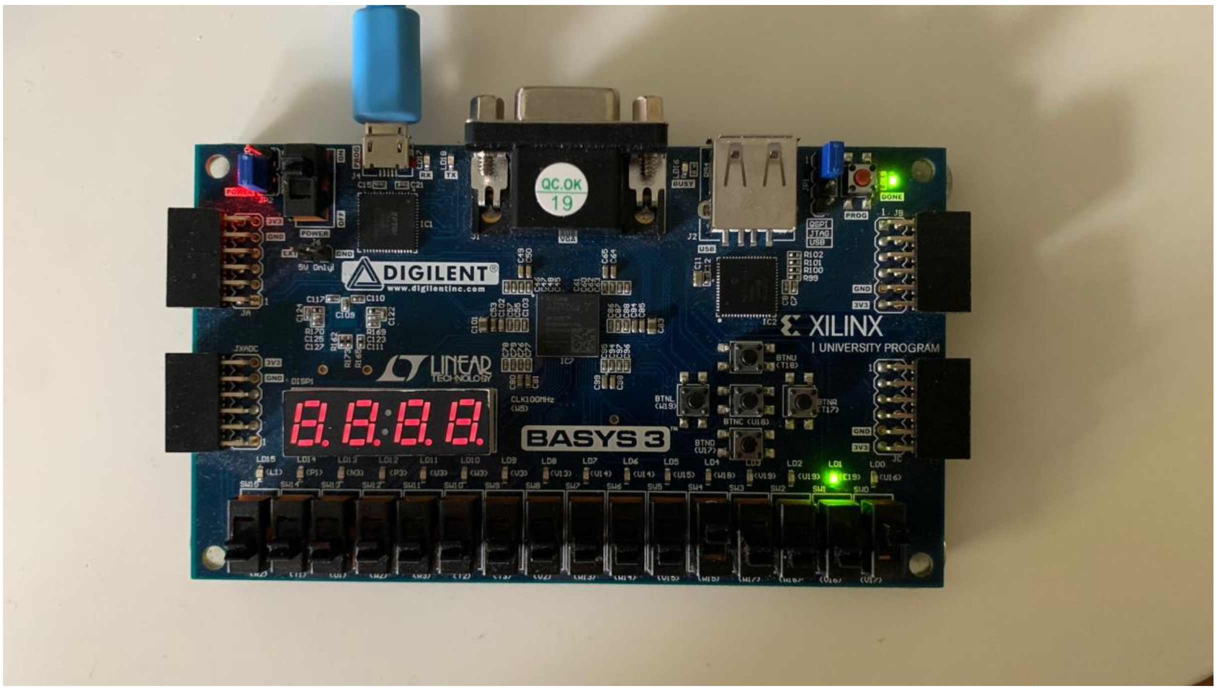

Figure 3: Addition

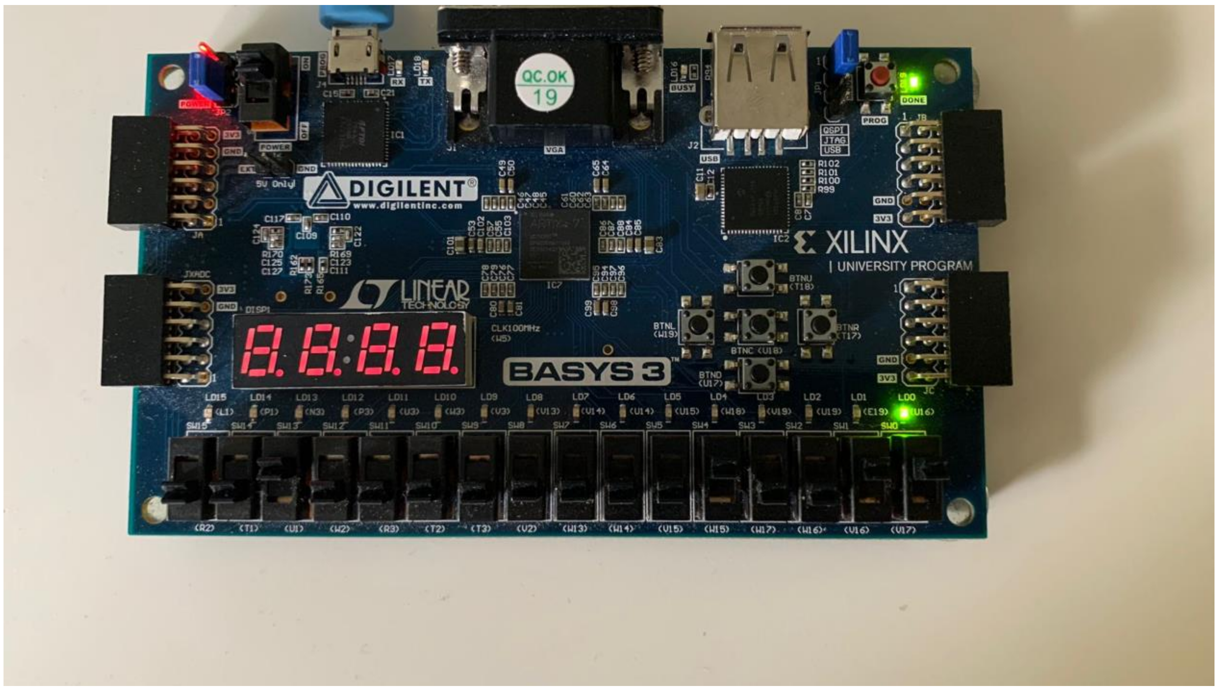

Figure 4: AND Gate

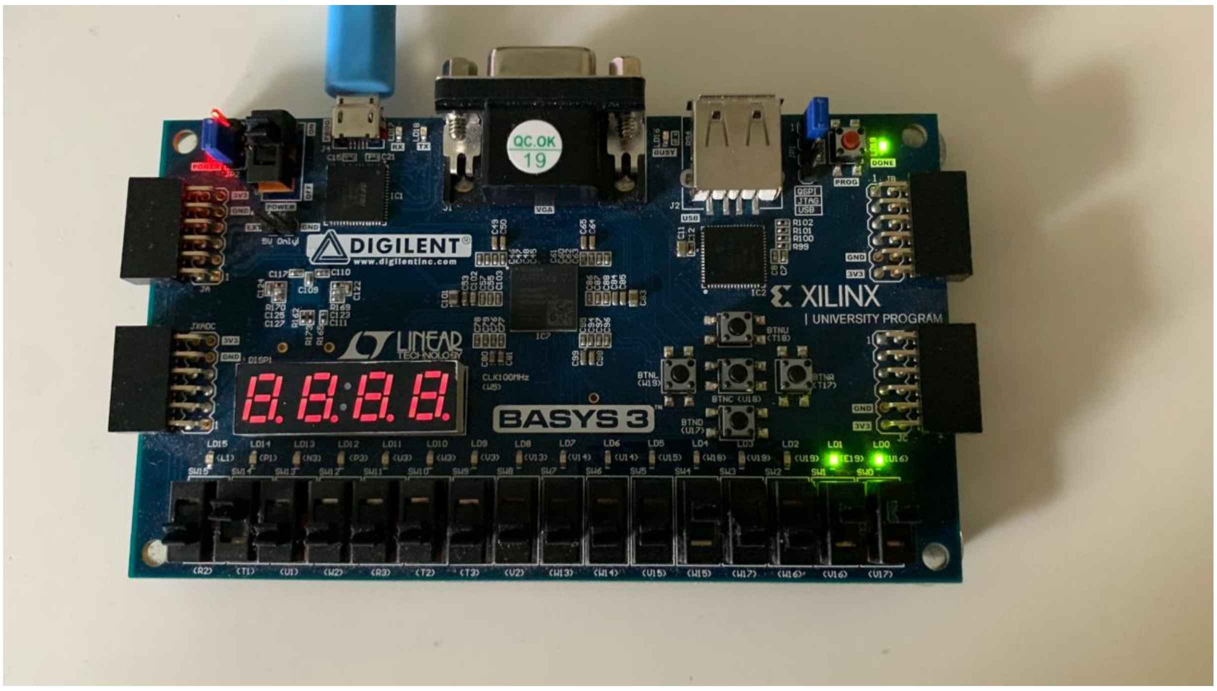

Figure 5: OR Gate

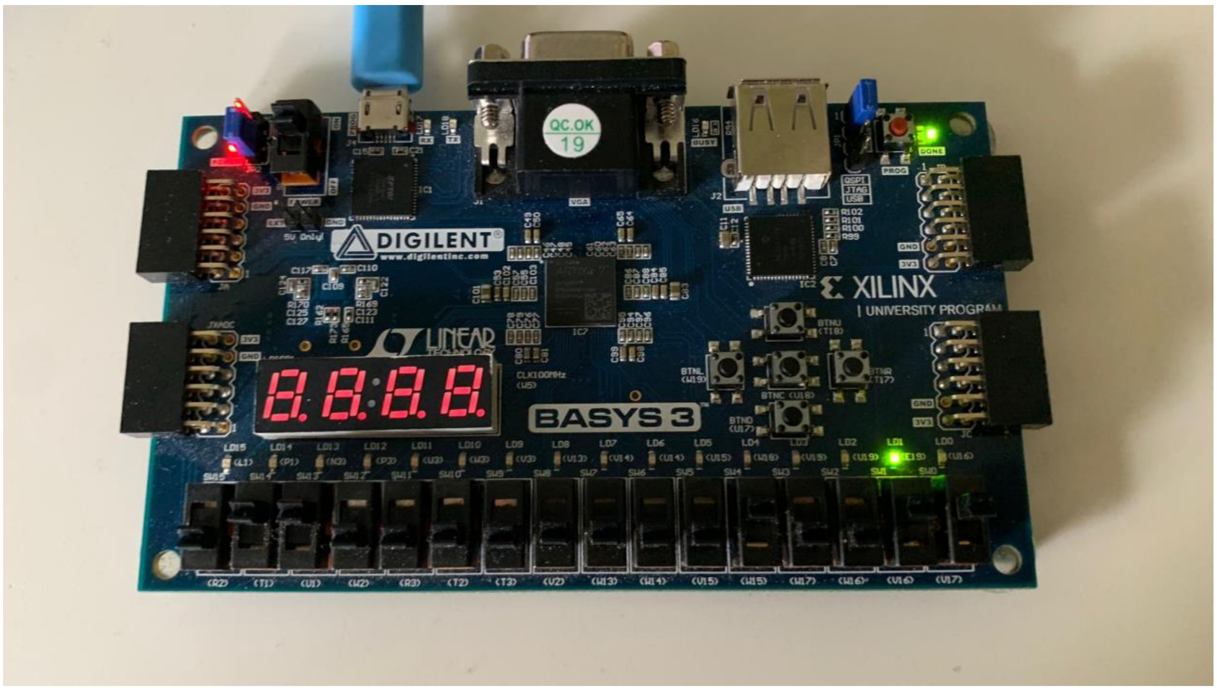

Figure 6: XOR Gate

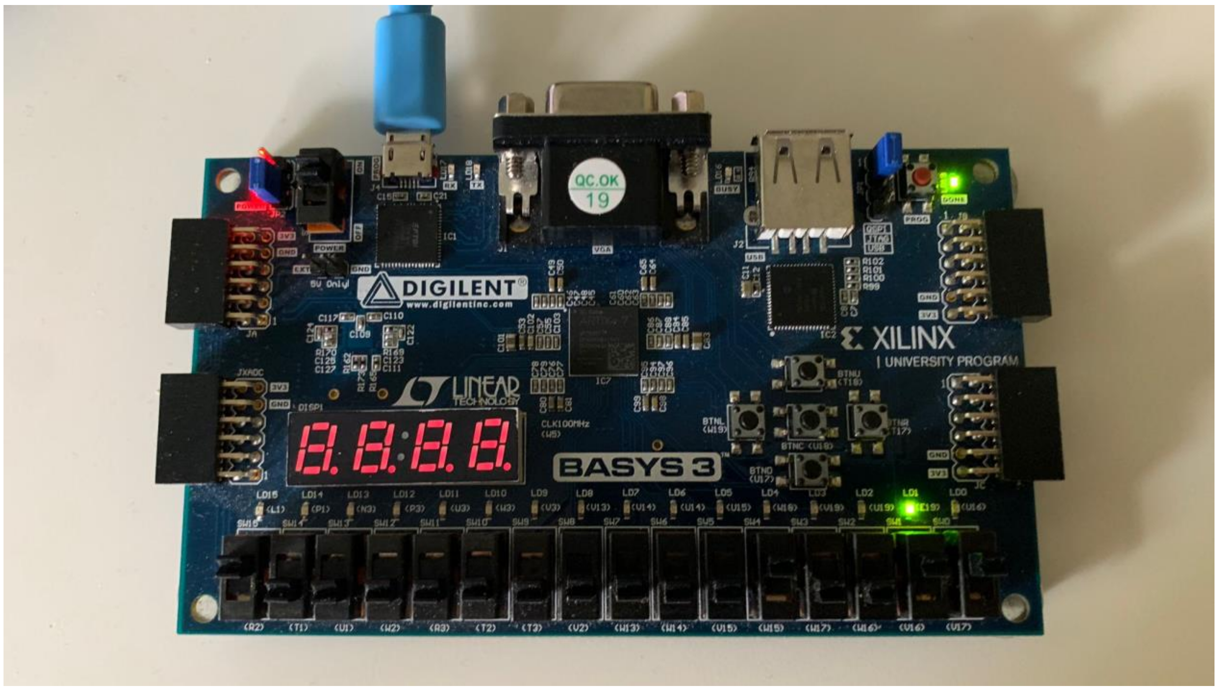

Figure 7: Subtraction

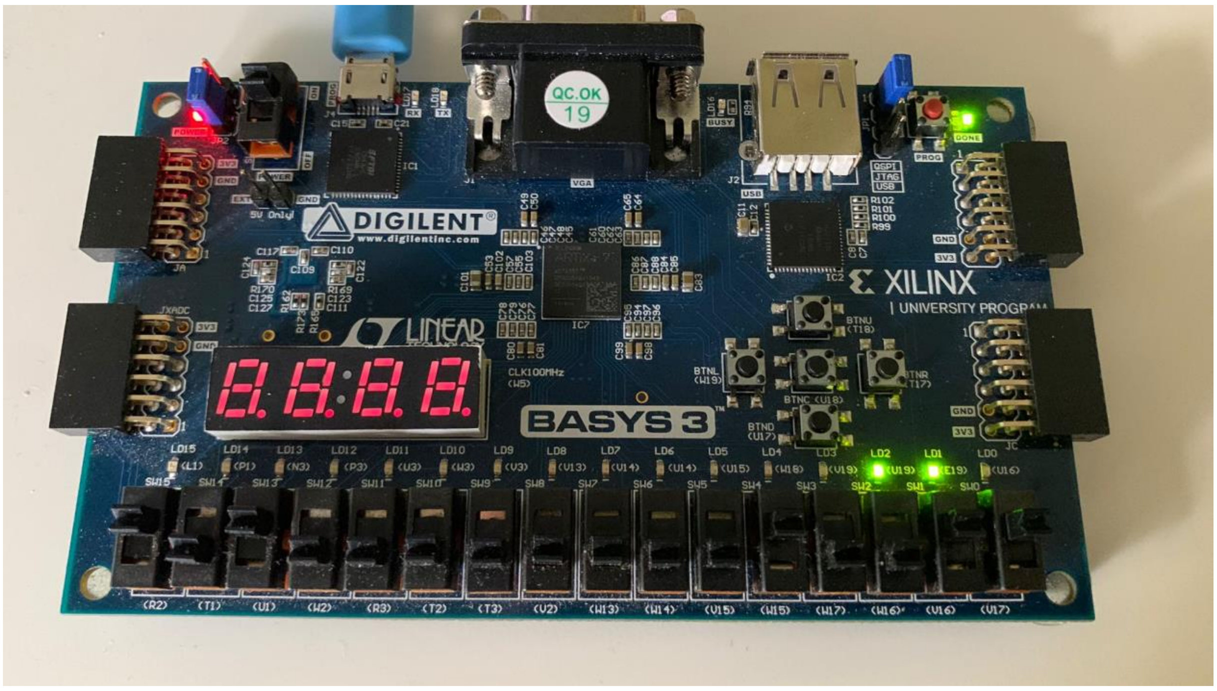

Figure 8: Rotate

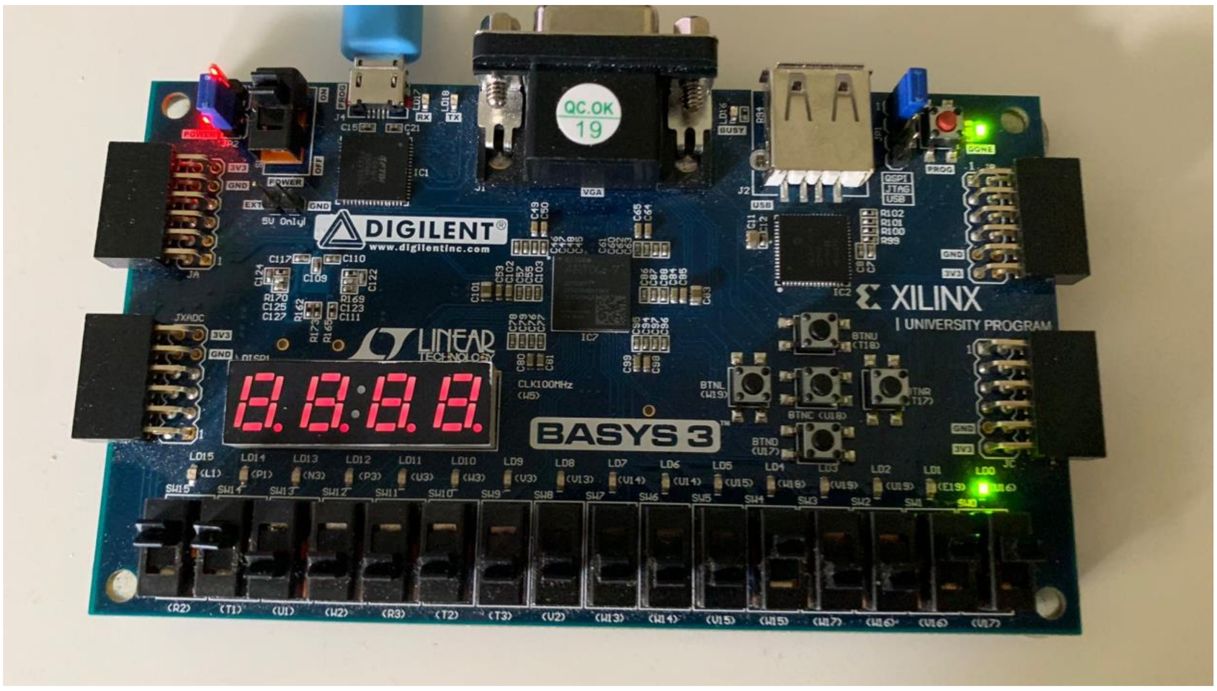

Figure 9: Logical Shift

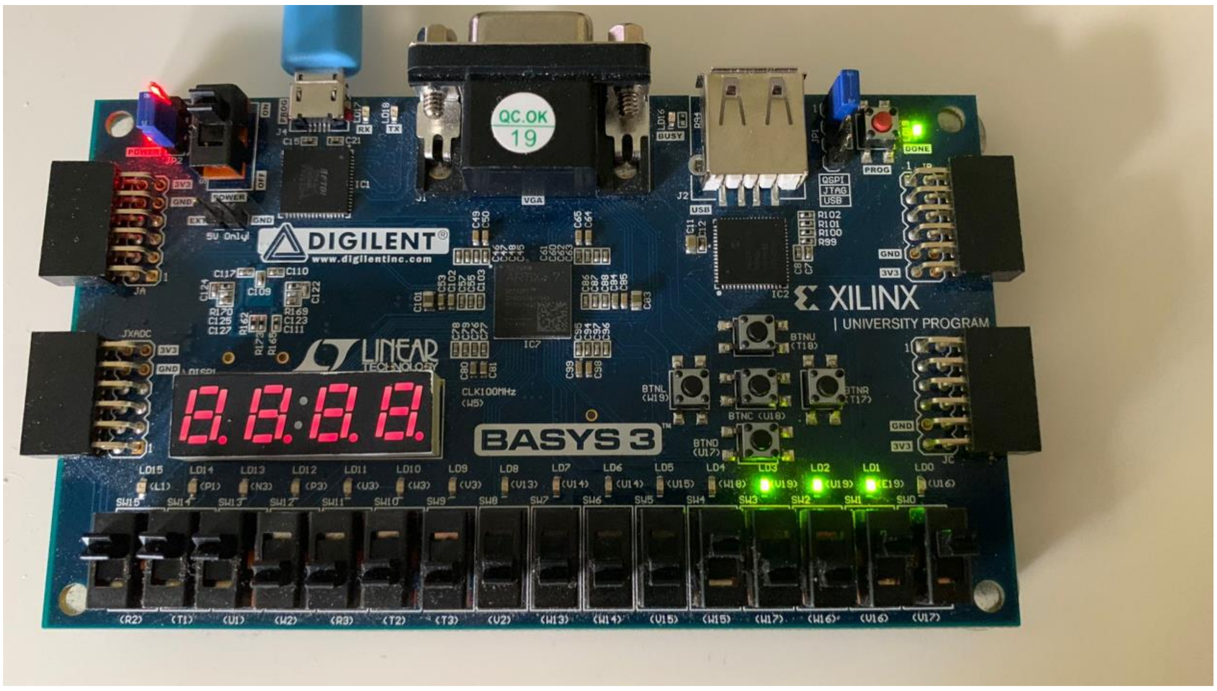

Figure 10: Rotate Through Carry

Conclusion

The purpose of this lab session was to teach students about the logic behind digital designs of mathematical operations such as the ones we used in this lab, and get students have some knowledge about arithmetic logic units and multiplexers to combine every operation on one unit. Students had the opportunity to observe RTL Schematics, write test benches for simulations, and create constraint files, meanwhile for the first time creating separate design sources for different components of a digital design. As this was completed students checked the design on their FPGAs, and they observed their outputs. In this lab, I made the error of not adding 1 to the one's complement of the subtracted number while creating the subtraction operation, I saw my error as the output was the addition of the first number plus the second number's one's complement instead of the two's complement. I also understood what logic vectors are and used them in VHDL. The fourth lab helpful in the sense of being more into VHDL code and creating new designs we were not familiar with, and helpful in visualizing the logic of mathematical operations in the digital systems of computers.

Appendices

alu.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;



entity alu is

    Port (

        A : in std_logic_vector(3 downto 0);

        B : in std_logic_vector(3 downto 0);

        K: in std_logic_vector(2 downto 0);

        Z : out std_logic_vector(3 downto 0));

end alu;



architecture Behavioral of alu is
```

```
component add is
   Port (
      add_in_1 : in std_logic_vector(3 downto 0);
      add_in_2 : in std_logic_vector(3 downto 0);
      add_in_C : in STD_LOGIC;
      add_out_C : out STD_LOGIC;
      add_out_S : out std_logic_vector(3 downto 0));
end component ;

component and_gate is
   Port (
      and_in_1 : in std_logic_vector(3 downto 0);
      and_in_2 : in STD_LOGIC_vector(3 downto 0);
      and_out : out STD_LOGIC_vector(3 downto 0));
end component;

component or_gate is
   Port (
      or_in_1 : in std_logic_vector(3 downto 0);
      or_in_2 : in STD_LOGIC_vector(3 downto 0);
      or_out : out STD_LOGIC_vector(3 downto 0));
end component;

component xor_gate is
   Port (
      xor_in_1 : in std_logic_vector(3 downto 0);
      xor_in_2 : in STD_LOGIC_vector(3 downto 0);
      xor_out : out STD_LOGIC_vector(3 downto 0));
```

```
end component;


component subtract is
   Port (
      sub_in_1 : in STD_LOGIC_vector( 3 downto 0);
      sub_in_2 : in STD_LOGIC_vector(3 downto 0);
      sub_in_C : in STD_LOGIC;
      sub_out : out STD_LOGIC_vector(3 downto 0));
end component;


component rotate is
   Port (
      rotate_in : in std_logic_vector(3 downto 0);
      rotate_out : out std_logic_vector(3 downto 0));
end component ;


component logical_shift is
   Port (
      shift_in : in STD_LOGIC_vector(3 downto 0);
      shift_out : out STD_LOGIC_vector(3 downto 0));
end component ;


component rotate_through_carry is
   Port (
      rotatec_in_1 : in STD_LOGIC_vector(3 downto 0);
      rotatec_in_C : in STD_LOGIC;
      rotatec_out : out STD_LOGIC_vector(3 downto 0);
      rotatec_out_C : 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);


begin



adding: add port map(add_in_1 => A, add_in_2 => B, add_in_C => '0', add_out_S => s1);


andg: and_gate port map (and_in_1 => A, and_in_2 => B, and_out => s2);


org: or_gate port map (or_in_1 => A, or_in_2 => B, or_out => s3);


xorg: xor_gate port map (xor_in_1 => A, xor_in_2 => B, xor_out => s4);


subtracter: subtract port map (sub_in_1 => A, sub_in_2 => B, sub_in_C => '1', sub_out =>
s5);


rotator: rotate port map (rotate_in => A, rotate_out => s6);


logicals: logical_shift port map (shift_in => A, shift_out => s7);


rotatorc: rotate_through_carry port map (rotatec_in_1 => A, rotatec_in_C => '1', rotatec_out
=> s8);
```

```
with K select
    Z <= s1 when "000",
    s2 when "001",
    s3 when "010",
    s4 when "011",
    s5 when "100",
    s6 when "101",
    s7 when "110",
    s8 when others;
end Behavioral;
```

add.vhd

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


entity add is
    Port (
        add_in_1 : in std_logic_vector(3 downto 0);
        add_in_2 : in std_logic_vector(3 downto 0);
        add_in_C : in STD_LOGIC;
        add_out_C : out STD_LOGIC;
        add_out_S : out std_logic_vector(3 downto 0));
end add;
```

```
architecture Behavioral of add is


component full_adder is
   port (
       fa_in_1: in std_logic;
       fa_in_2: in std_logic;
       fa_in_C: in std_logic;
       fa_out_S: out std_logic;
       fa_out_C: out std_logic);
end component;


signal c0, c1, c2, c3: std_logic;
signal ss0, ss1, ss2, ss3: std_logic;


begin


fa1: full_adder port map (fa_in_1 => add_in_1(0), fa_in_2 =>add_in_2(0), fa_in_C => '0',
fa_out_S => ss0 , fa_out_C => c0);


fa2: full_adder port map (fa_in_1 => add_in_1(1), fa_in_2 =>add_in_2(1), fa_in_C => c0,
fa_out_S => ss1, fa_out_C => c1);


fa3: full_adder port map (fa_in_1 => add_in_1(2), fa_in_2 =>add_in_2(2), fa_in_C => c1,
fa_out_S => ss2 , fa_out_C => c2);


fa4: full_adder port map (fa_in_1 => add_in_1(3), fa_in_2 =>add_in_2(3), fa_in_C => c2,
fa_out_S => ss3, fa_out_C => c3);


add_out_C <= c3 ;
add_out_S <= ss3 & ss2 & ss1 & ss0 ;
```

Emir Arda Bayer
22201832-4

```
end Behavioral;
```

and_gate.vhd

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


entity and_gate is
   Port (
       and_in_1 : in std_logic_vector(3 downto 0);
       and_in_2 : in STD_LOGIC_vector(3 downto 0);
       and_out : out STD_LOGIC_vector(3 downto 0));
end and_gate;


architecture Behavioral of and_gate is

begin

and_out(0) <= and_in_1(0) and and_in_2(0);
and_out(1) <= and_in_1(1) and and_in_2(1);
and_out(2) <= and_in_1(2) and and_in_2(2);
and_out(3) <= and_in_1(3) and and_in_2(3);

end Behavioral;
```

or_gate.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;



entity or_gate is

    Port (

        or_in_1 : in std_logic_vector(3 downto 0);

        or_in_2 : in STD_LOGIC_vector(3 downto 0);

        or_out : out STD_LOGIC_vector(3 downto 0));

end or_gate;



architecture Behavioral of or_gate is


begin


or_out(0) <= or_in_1(0) or or_in_2(0);

or_out(1) <= or_in_1(1) or or_in_2(1);

or_out(2) <= or_in_1(2) or or_in_2(2);

or_out(3) <= or_in_1(3) or or_in_2(3);


end Behavioral;
```

xor_gate.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
```

```
entity xor_gate is
   Port (
        xor_in_1 : in std_logic_vector(3 downto 0);
        xor_in_2 : in STD_LOGIC_vector(3 downto 0);
        xor_out : out STD_LOGIC_vector(3 downto 0));
end xor_gate;


architecture Behavioral of xor_gate is

begin

xor_out(0) <= xor_in_1(0) xor xor_in_2(0);
xor_out(1) <= xor_in_1(1) xor xor_in_2(1);
xor_out(2) <= xor_in_1(2) xor xor_in_2(2);
xor_out(3) <= xor_in_1(3) xor xor_in_2(3);

end Behavioral;
```

subtrtact.vhd

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


entity subtract is
   Port (
        sub_in_1 : in STD_LOGIC_vector( 3 downto 0);
        sub_in_2 : in STD_LOGIC_vector(3 downto 0);
        sub_in_C : in STD_LOGIC;
        sub_out : out STD_LOGIC_vector(3 downto 0));
end subtract;



architecture Behavioral of subtract is



component full_adder is
    port (
        fa_in_1: in std_logic;
        fa_in_2: in std_logic;
        fa_in_C: in std_logic;
        fa_out_S: out std_logic;
        fa_out_C: out std_logic);
end component;

signal c0, c1, c2, c3: std_logic;
signal s_0, s_1, s_2, s_3: std_logic;
signal d: std_logic_vector(3 downto 0);

begin

d <= not sub_in_2;

fa1: full_adder port map (fa_in_1 => sub_in_1(0), fa_in_2 =>d(0), fa_in_C => sub_in_C,
fa_out_S => s_0 , fa_out_C => c0);
```

```
fa2: full_adder port map (fa_in_1 => sub_in_1(1), fa_in_2 =>d(1), fa_in_C => c0, fa_out_S
=> s_1 , fa_out_C => c1);

fa3: full_adder port map (fa_in_1 => sub_in_1(2), fa_in_2 =>d(2), fa_in_C => c1, fa_out_S
=> s_2 , fa_out_C => c2);

fa4: full_adder port map (fa_in_1 => sub_in_1(3), fa_in_2 =>d(3), fa_in_C => c2, fa_out_S
=> s_3, fa_out_C => c3);


sub_out <= s_3& s_2& s_1& s_0 ;


end Behavioral;
```

full_adder.vhd

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



entity full_adder is
  port (
    fa_in_1: in std_logic;
    fa_in_2: in std_logic;
    fa_in_C: in std_logic;
    fa_out_S: out std_logic;
    fa_out_C: out std_logic);
end full_adder;



architecture Behavioral of full_adder is
```

```
begin


fa_out_S <= (fa_in_1 xor fa_in_2) xor fa_in_C;
fa_out_C <= (fa_in_1 and fa_in_2) or (fa_in_2 and fa_in_C) or (fa_in_1 and fa_in_C);


end Behavioral;
```

rotate.vhd

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


entity rotate is
   Port (
        rotate_in : in std_logic_vector(3 downto 0);
        rotate_out : out std_logic_vector(3 downto 0));
end rotate ;



architecture Behavioral of rotate is


begin


rotate_out(0) <= rotate_in(3);
rotate_out(1) <= rotate_in(0);
rotate_out(2) <= rotate_in(1);
rotate_out(3) <= rotate_in(2);
```

```
end Behavioral;
```

logical_shift.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;



entity logical_shift is

   Port (

      shift_in : in STD_LOGIC_vector(3 downto 0);

      shift_out : out STD_LOGIC_vector(3 downto 0));

end logical_shift;



architecture Behavioral of logical_shift is


begin


shift_out(0) <= shift_in(1);

shift_out(1) <= shift_in(2);

shift_out(2) <= shift_in(3);

shift_out(3) <= '0';


end Behavioral;
```

13.03.2024 Emir Arda Bayer
22201832-4

rotate_through_carry.vhd

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



entity rotate_through_carry is
    Port (
        rotatec_in_1 : in STD_LOGIC_vector(3 downto 0);
        rotatec_in_C : in STD_LOGIC;
        rotatec_out : out STD_LOGIC_vector(3 downto 0);
        rotatec_out_C : out std_logic);
end rotate_through_carry;



architecture Behavioral of rotate_through_carry is

begin

rotatec_out(1) <= rotatec_in_C;
rotatec_out(0) <= rotatec_in_1(3);
rotatec_out(2) <= rotatec_in_1(0);
rotatec_out(3) <= rotatec_in_1(1);
rotatec_out_C <= rotatec_in_1(2);

end Behavioral;
```

cons.xdc

```
# Switches

set_property PACKAGE_PIN V17 [get_ports {A[0]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]

set_property PACKAGE_PIN V16 [get_ports {A[1]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]

set_property PACKAGE_PIN W16 [get_ports {A[2]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]

set_property PACKAGE_PIN W17 [get_ports {A[3]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]

set_property PACKAGE_PIN W15 [get_ports {B[0]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]

set_property PACKAGE_PIN V15 [get_ports {B[1]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]

set_property PACKAGE_PIN W14 [get_ports {B[2]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]

set_property PACKAGE_PIN W13 [get_ports {B[3]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]

set_property PACKAGE_PIN U1 [get_ports {K[0]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {K[0]}]

set_property PACKAGE_PIN T1 [get_ports {K[1]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {K[1]}]

set_property PACKAGE_PIN R2 [get_ports {K[2]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {K[2]}]



# LEDs

set_property PACKAGE_PIN U16 [get_ports {Z[0]}]

    set_property IOSTANDARD LVCMOS33 [get_ports {Z[0]}]
set_property PACKAGE_PIN E19 [get_ports {Z[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Z[1]}]
set_property PACKAGE_PIN U19 [get_ports {Z[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Z[2]}]
set_property PACKAGE_PIN V19 [get_ports {Z[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Z[3]}]
```

alu_tb.vhd

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;


entity alu_tb is
end;


architecture bench of alu_tb is

component alu
   Port (
      A : in std_logic_vector(3 downto 0);
      B : in std_logic_vector(3 downto 0);
      K: in std_logic_vector(2 downto 0);
      Z : out std_logic_vector(3 downto 0));
end component;
```

```
signal A: std_logic_vector(3 downto 0);

signal B: std_logic_vector(3 downto 0);

signal K: std_logic_vector(2 downto 0);

signal Z: std_logic_vector(3 downto 0);


begin


uut: alu port map (

   A => A,

   B => B,

   K => K,

   Z => Z);



stimulus: process


begin


A<="1010";

B<="0101";

K<="000";

wait for 100 ns;


A<="1010";

B<="0101";

K<="001";

wait for 100 ns;
```

Emir Arda Bayer
22201832-4

```
A<="1010";
B<="0101";
K<="010";
wait for 100 ns;

A<="1010";
B<="0101";
K<="011";
wait for 100 ns;

A<="1010";
B<="0101";
K<="100";
wait for 100 ns;

A<="1010";
B<="0101";
K<="101";
wait for 100 ns;

A<="1010";
B<="0101";
K<="110";
wait for 100 ns;

A<="1010";
B<="0101";
K<="111";
wait for 100 ns;
```

```
wait;


end process;
end;
```