ECEN 429: Introduction to Digital Systems Design Laboratory

North Carolina Agricultural and Technical State University

Department of Electrical and Computer Engineering

Ian Parker (Reporter)

Tayanna Lee (Lab Partner)

February 21, 2019

Lab #4

Introduction:

In this lab we are learning to work with multiplexers and use them to implement functions. Multiplexers can be great tools used in complex logic circuits and we see how they're used in everyday life (such as train systems for example). This lab also focuses on the use of components in VHDL code. This lab should give us a good understanding of how to efficiently and effectively re-use code using components.

Before we begin solving the parts of the lab we must first have an understanding of a 4:1 MUX

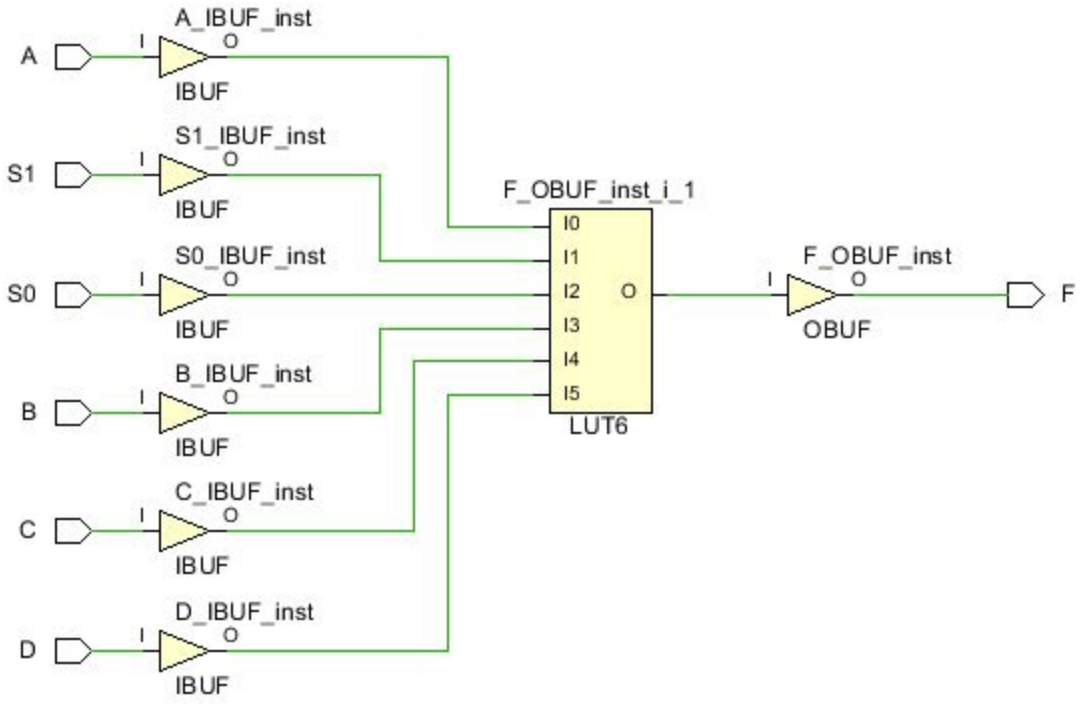
Possible Inputs for 4:1 MUX

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	**1**	**0**	**1**
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	**0**	**1**	**0**
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Truth Table

S1	S0	**F**
0	0	**A**
0	1	**B**
1	0	**C**
1	1	**D**

Schematic for a 4:1 MUX



Pin Assignments for a 4:1 MUX

NAME	I/O	PIN
A	IN	V15
B	IN	W15
C	IN	W17
D	IN	W16
S0	IN	V17
S1	IN	V16
F	OUT	U16

Results for a 4:1 MUX

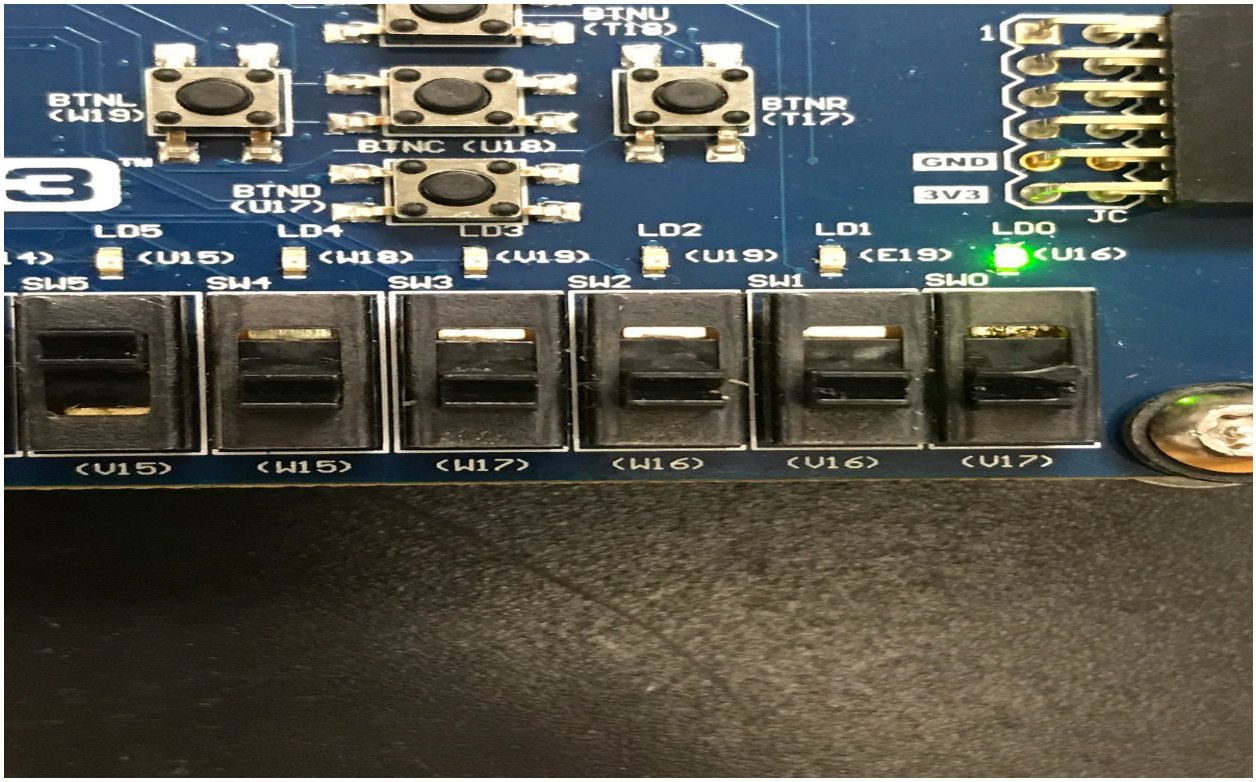


Figure 1.A) S1 =0; S0 = 0; Thus F = A

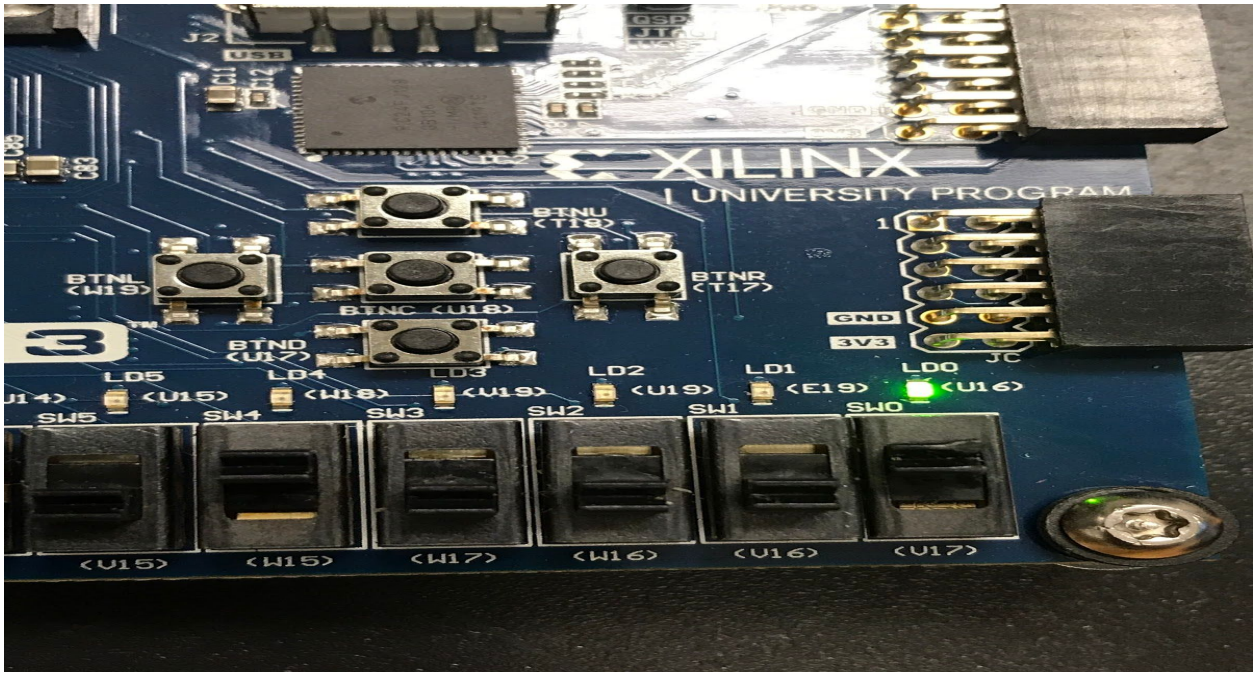


Figure 1.B) S1 =0; S0 = 1; Thus F = B

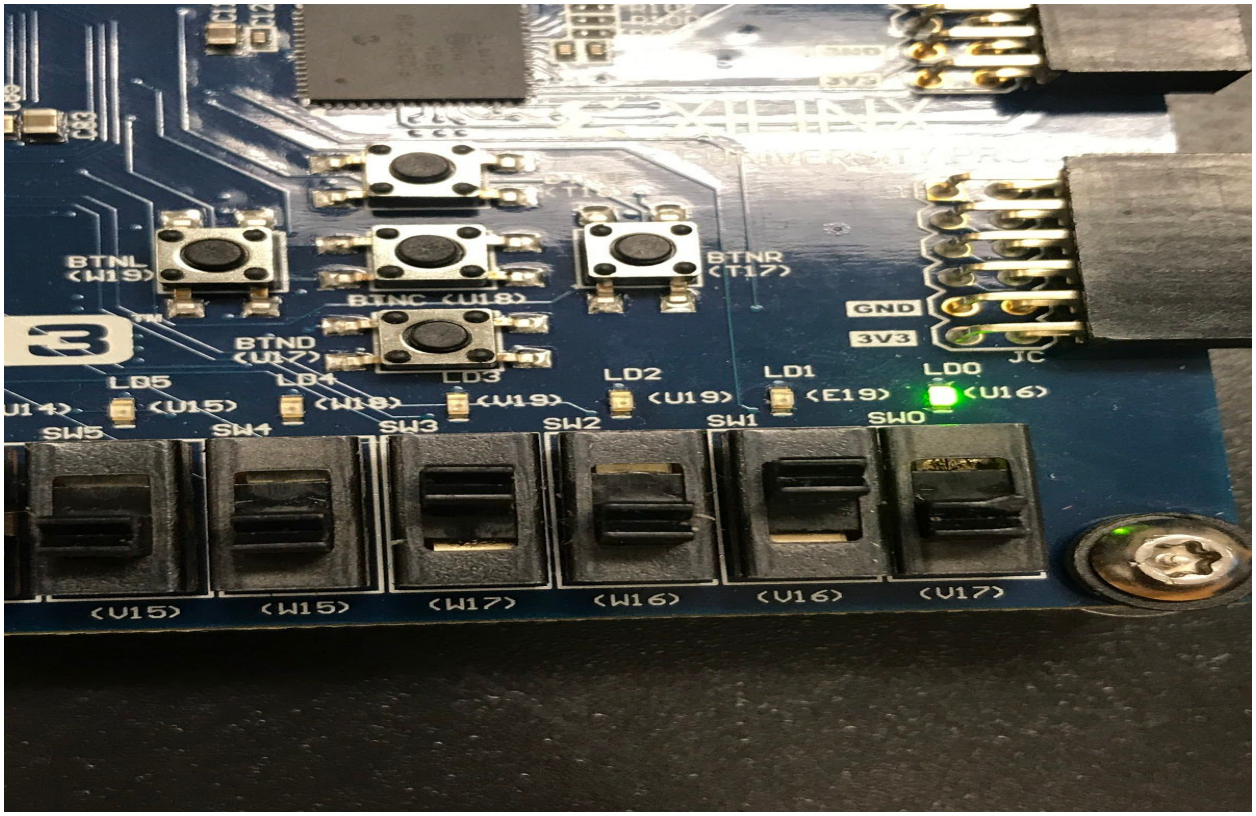


Figure 1.C) S1 =1; S0 = 0; Thus F = C

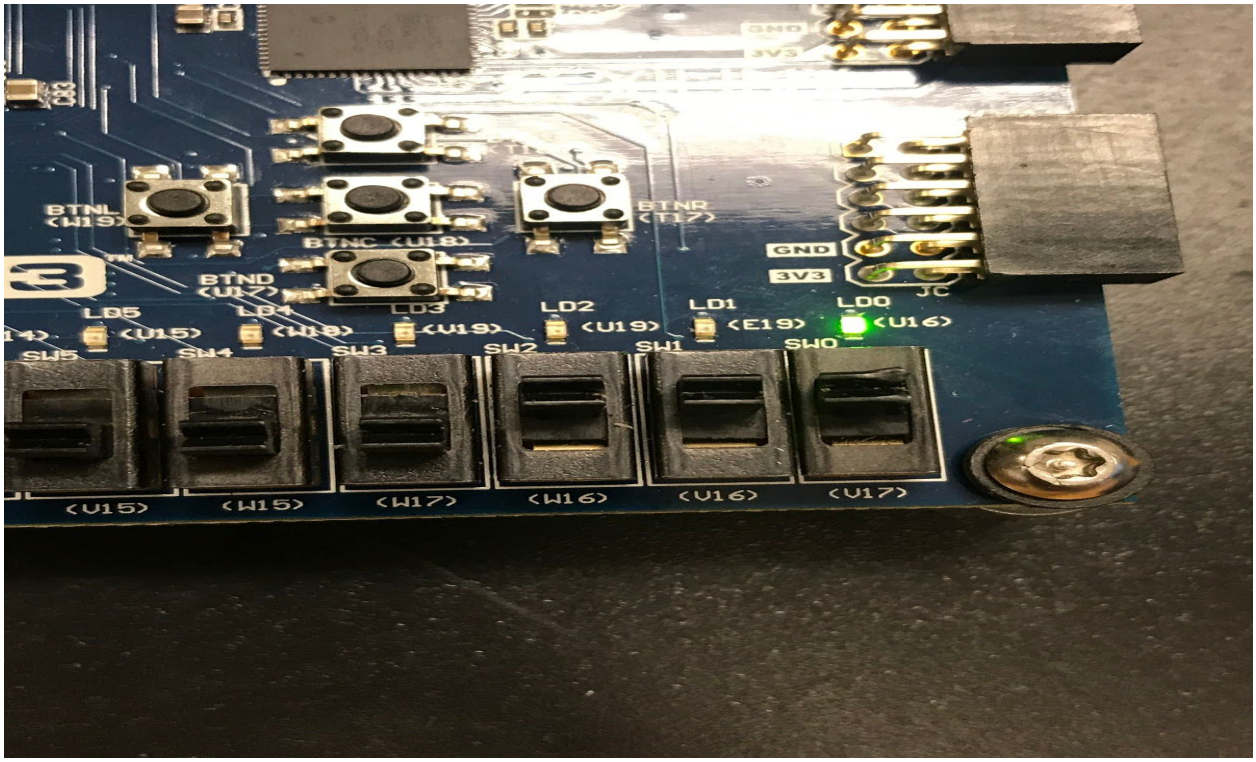


Figure 1.D) S1 =1; S0 = 1; Thus F = D

Part 1 and 2)
Now that we have an understanding of a 4:1 MUX we can use it to implement the following scenarios...

1) *A circuit that has 2 bits input and the output is their sum.*
2) *A circuit that has 2 bits input and the output is their subtraction.*

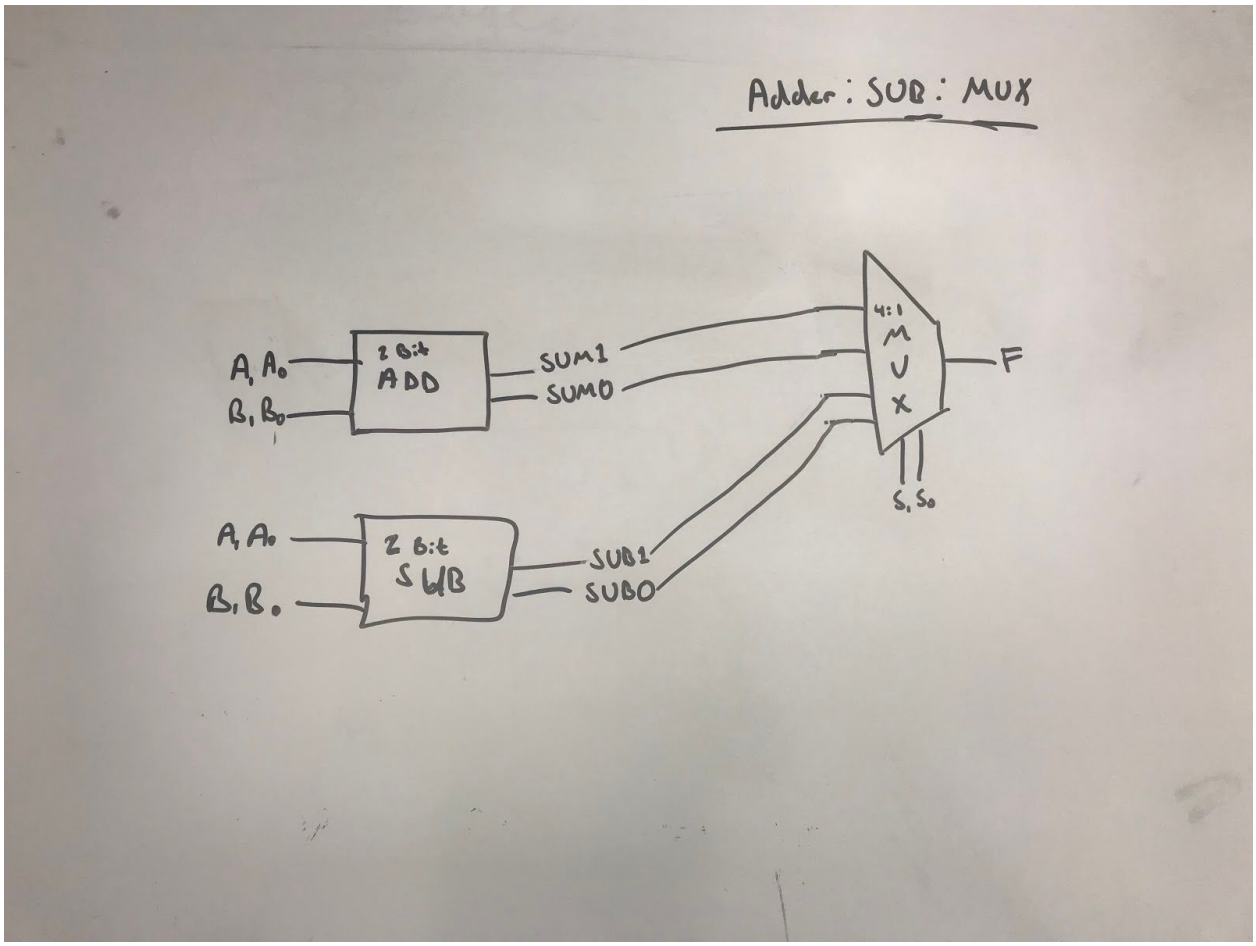


Figure: *A diagram showing how a 2 Bit ADDER and a 2 bit SUBTRACTOR can be implemented with a 4:1 Multiplexer.*

Truth Table for 2-Bit ADDER

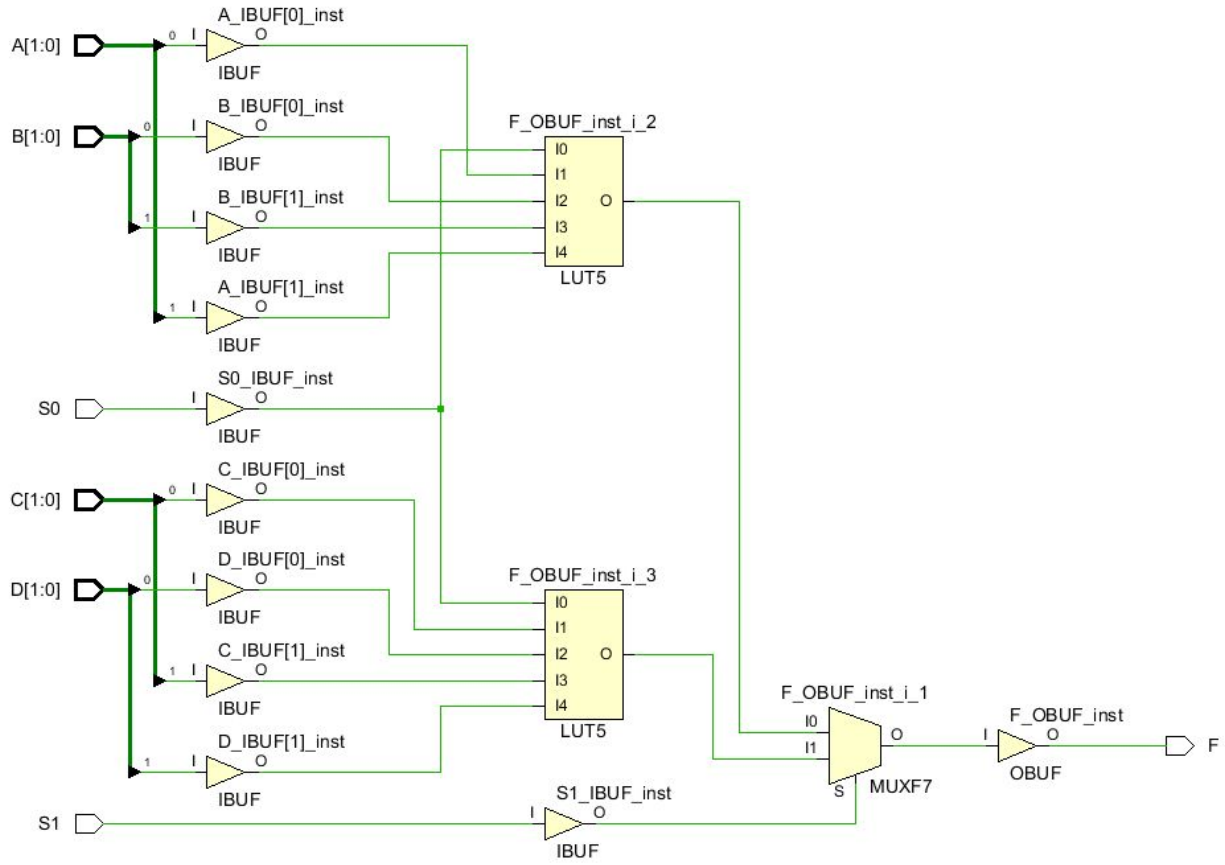
Inputs				Outputs		
a1	a0	b1	b0	c	s1	s0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0

Inputs				Outputs		
a1	a0	b1	b0	c	s1	s0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Truth Table for 2-Bit SUBTRACTOR

A1	A0	B1	B0	**SUB1**	**SUB0**
0	0	0	0	**0**	**0**
0	0	0	1	**0**	**0**
0	0	1	0	**0**	**0**
0	0	1	1	**0**	**0**
0	1	0	0	**0**	**1**
0	1	0	1	**0**	**0**
0	1	1	0	**0**	**0**
0	1	1	1	**0**	**0**
1	0	0	0	**1**	**0**
1	0	0	1	**0**	**1**
1	0	1	0	**0**	**0**
1	0	1	1	**0**	**0**
1	1	0	0	**1**	**1**
1	1	0	1	**1**	**0**
1	1	1	0	**0**	**1**
1	1	1	1	**0**	**0**

Schematic for a 2-Bit Adder and Subtractor feeding into a 4:1 MUX



\

Results

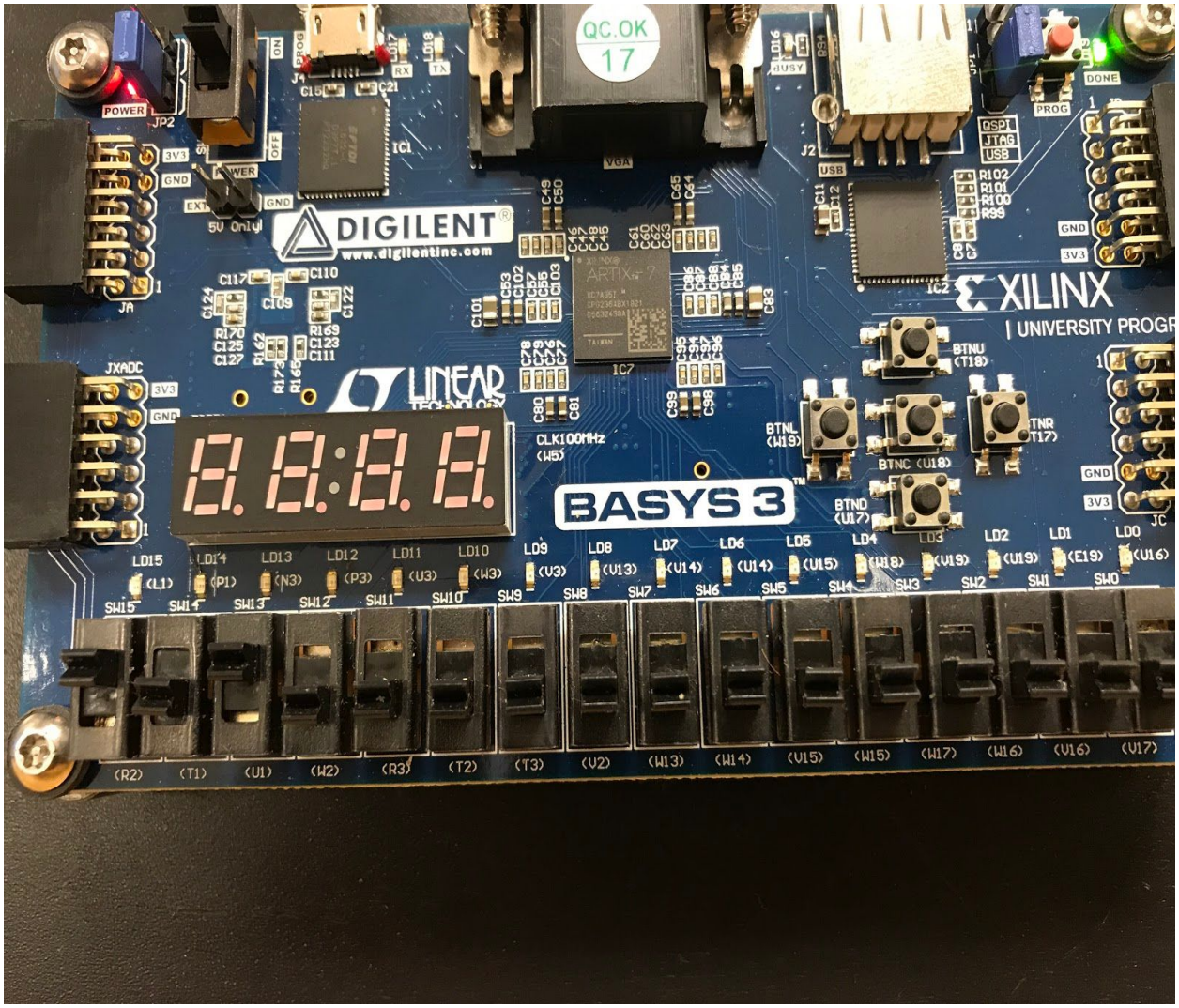


Figure 1) Adder = 1010 Select = 00 Output is 0

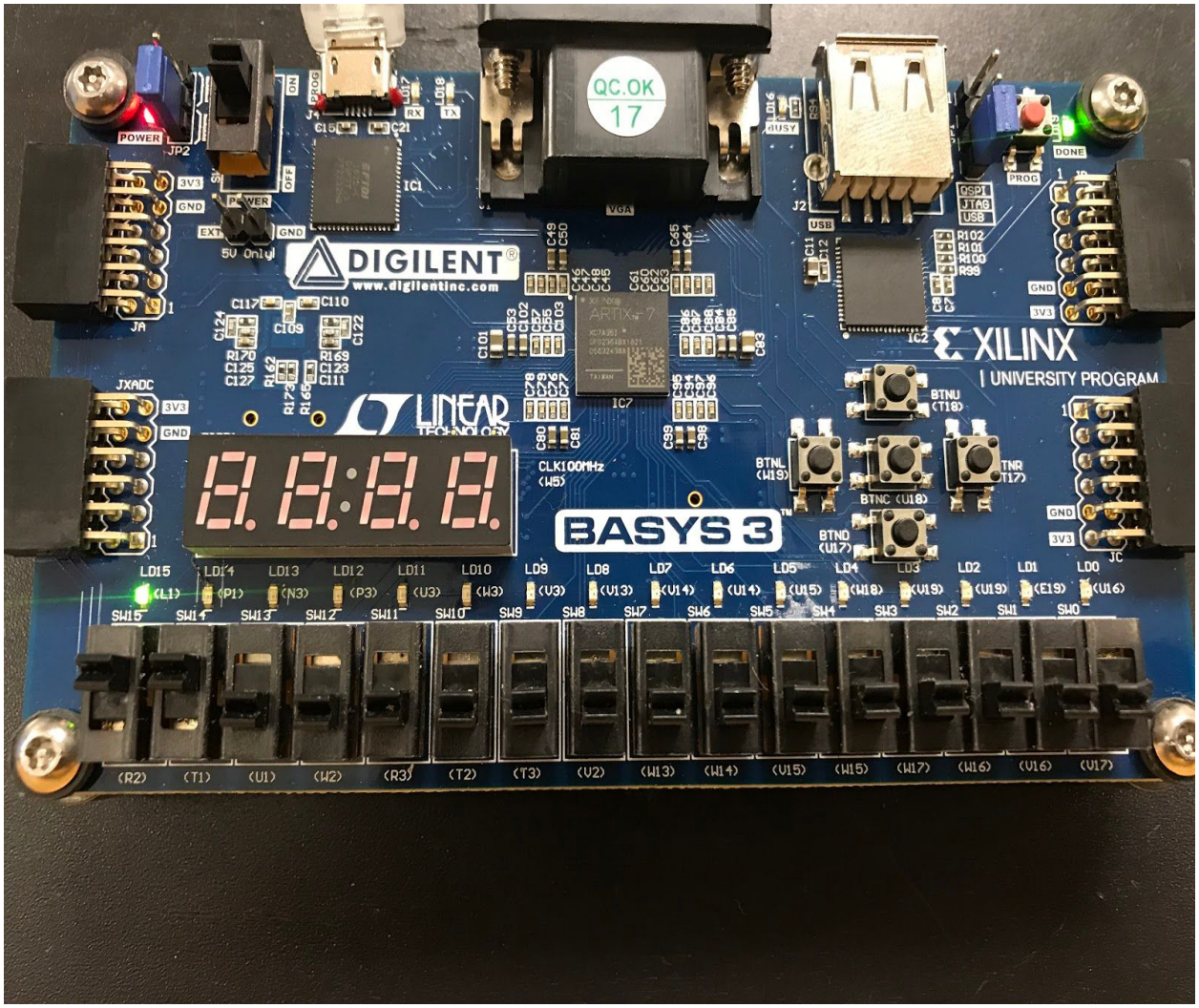


Figure 2) Adder = 1100 Select = 00 Output = 1

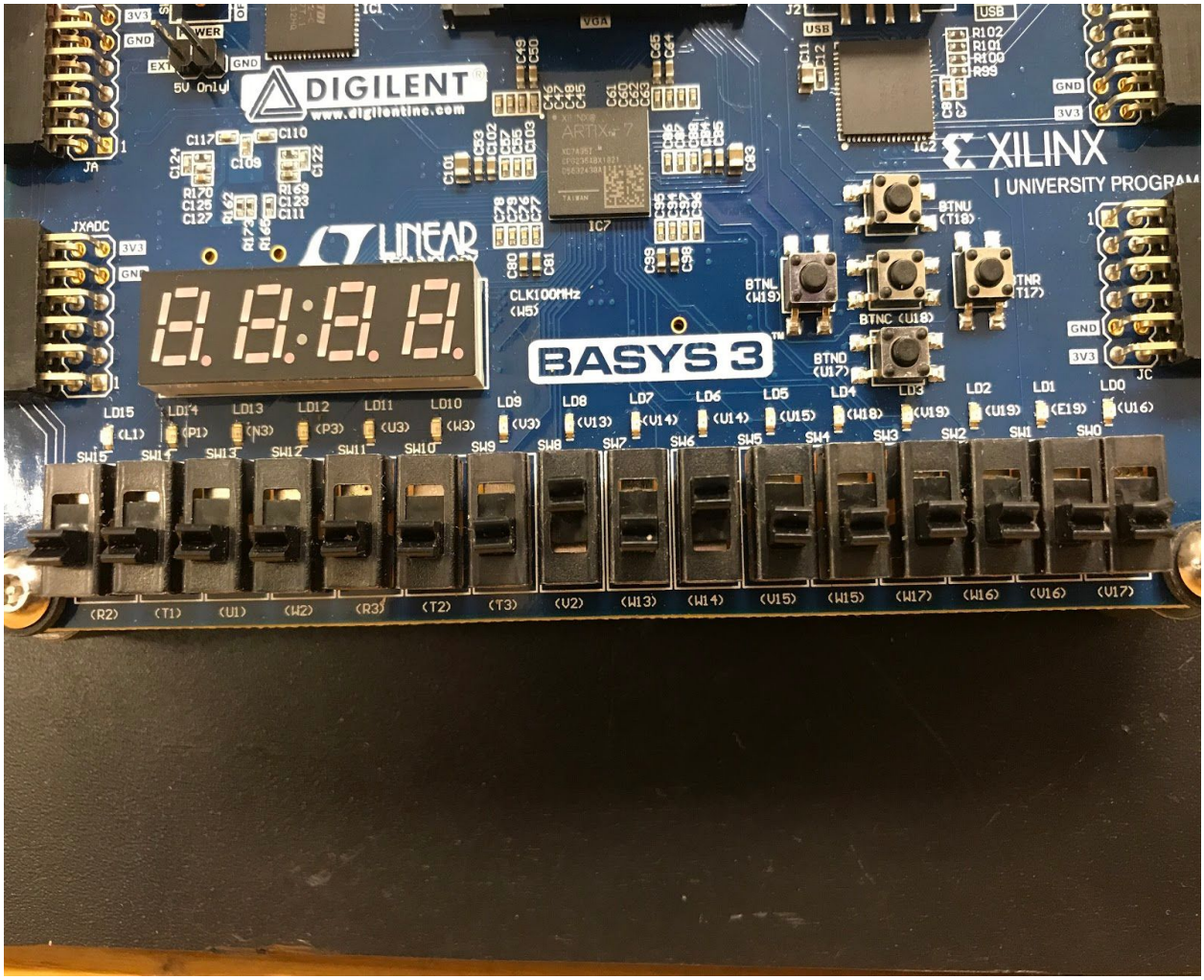


Figure 3) Subtractor = 0001 Select = 01 Output = 0

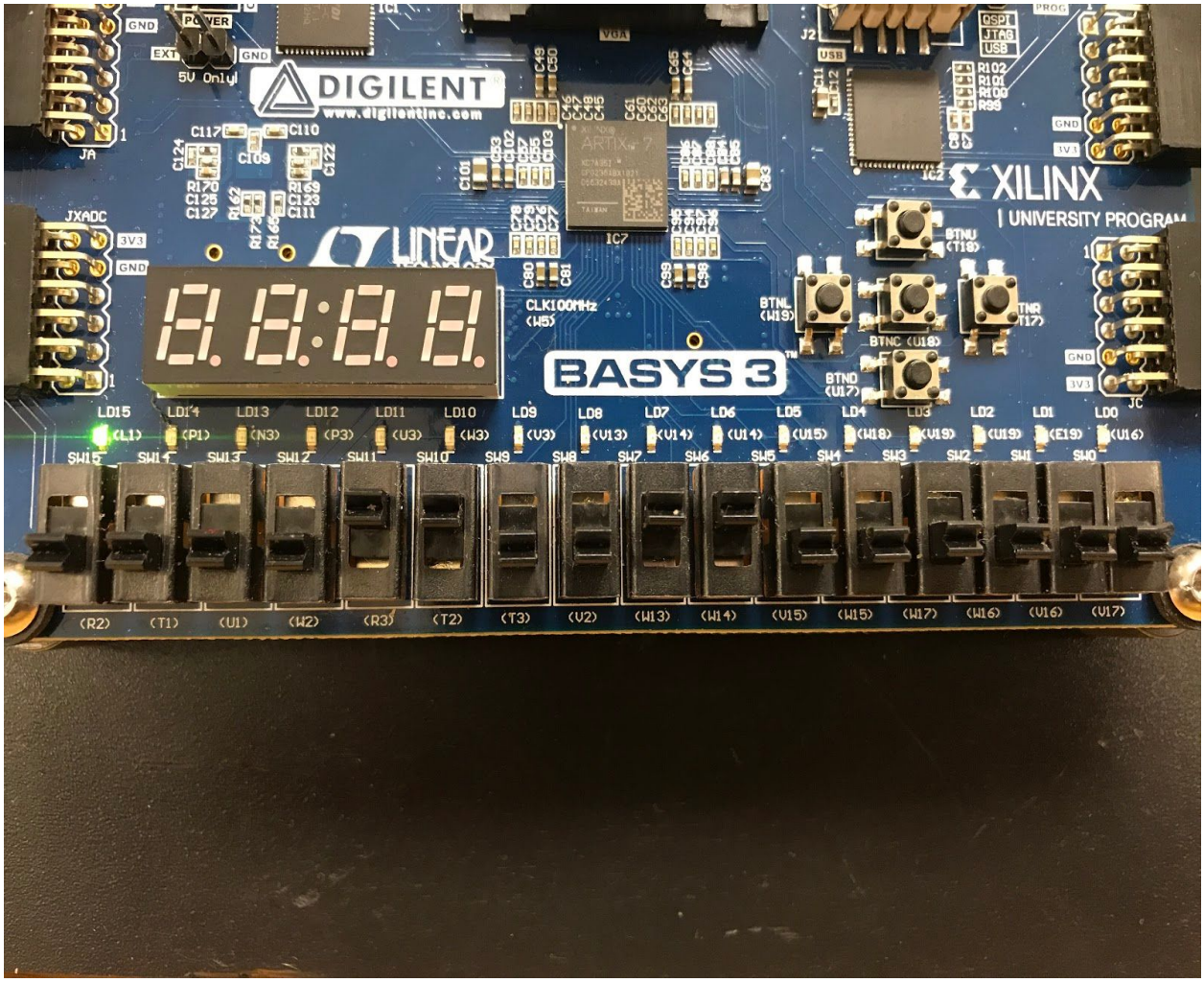


Figure 4) Subtractor = 1100 Select = 11 Output = 1

Appendices

Part 1 FourToOneMux VHDL Code:

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

entity FourToOneMux is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : in STD_LOGIC;
           D : in STD_LOGIC;
           S0 : in STD_LOGIC;
           S1 : in STD_LOGIC;
           F : out STD_LOGIC);
end FourToOneMux;

architecture Behavioral of FourToOneMux is
    begin
        process(A,B,C,D,S0,S1) is
        begin
            if(S0 = '0' AND S1 = '0')then
                F <= A;
            elsif(S0= '1' AND S1 = '0')then
                F <= B;
            elsif(S0 = '0' AND S1 = '1')then
                F <= C;
            else
                F <= D;
            end if;
         end process;
end Behavioral;
```

Part 1.1 TwoBitAdder VHDL Code

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

entity TwoBitAdder is
    Port (A,B      : in std_logic_vector(1 downto 0);
          SUM1, SUM0: out std_logic );
end TwoBitAdder;

architecture Behavioral of TwoBitAdder is
    signal ANOT1,ANOT0, BNOT1,BNOT0 : std_logic;
        begin
            ANOT1 <= not A(1);
            ANOT0 <= not A(0);
            BNOT1 <= not B(1);
            BNOT0 <= not B(0);

            sum1 <= (ANOT1 and b(1) and BNOT0)
               or (a(1) and ANOT0 and BNOT1 and b(0))
               or (ANOT1 and ANOT0 and b(1))
               or (ANOT1 and a(0) and BNOT1 and b(0))
               or (a(1) and a(0) and BNOT1 and BNOT0)
               or ( a(1) and a(0) and b(1) and b(0));

            sum0 <= (ANOT1 and ANOT0 and b(0))
               or (a(0) and b(1) and BNOT0)
               or (a(0) and BNOT1 and BNOT0)
               or (a(1) and ANOT0 and BNOT1)
               or (a(1) and ANOT0 and b(0));



end Behavioral;
```

Part 1.3 TwoBitSubtractor VHDL Code

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

entity TwoBitSubtractor is
    Port ( A,B: in STD_LOGIC_VECTOR(1 downto 0);
           SUB1, SUB0 : out STD_LOGIC);
end TwoBitSubtractor;

architecture Behavioral of TwoBitSubtractor is
    signal ANOT1, ANOT0, BNOT1, BNOT0 : STD_LOGIC;
        begin
            ANOT1 <= NOT A(1);
            ANOT0 <= NOT A(0);
            BNOT1 <= NOT B(1);
            BNOT0 <= NOT B(0);

            SUB1 <= (A(1) AND A(0) AND BNOT1) OR (A(1) AND ANOT0 AND BNOT1 AND
BNOT0);

            SUB0 <= (A(0) AND BNOT1 AND BNOT1) OR (A(1) AND ANOT0 AND BNOT1
AND B0)) OR (A(1) AND A(0) AND B(1) AND BNOT0);

end Behavioral;
```

Part 2 AddSubtractMux VHDL Code

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

entity AddSubtractMUX is
    Port ( A,B,C,D : in STD_LOGIC_VECTOR(1 downto 0);
           S1,S0 : in STD_LOGIC;
              F : out STD_LOGIC);
end AddSubtractMUX;

architecture Behavioral of AddSubtractMUX is
        --MUX
        component FourToOneMux is
            port (A,B,C,D,S0,S1 : STD_LOGIC;
                F : out STD_LOGIC);
        end component;
        --ADDER
        component TwoBitAdder is
            port (A,B : in STD_LOGIC_VECTOR(1 downto 0);
                SUM1,SUM0: out STD_LOGIC);
        end component;
        --SUBTRACTOR
        component TwoBitSubtractor is
            port (A,B : in STD_LOGIC_VECTOR(1 downto 0);
                SUB1,SUB0: out STD_LOGIC);
        end component;

        signal SUM0Temp, SUM1Temp, SUB1Temp, SUB0Temp : STD_LOGIC;

    begin

    ADDER: TwoBitAdder port map(A, B, SUM1Temp, SUM0Temp);
    SUBTRACTOR: TwoBitSubtractor port map(C, D, SUB1Temp, SUB0Temp);
    MUX: FourToOneMux port map(SUM1Temp, SUM0Temp, SUB1Temp, SUB0Temp, S0,S1,
F);

end Behavioral;
```

Constraint File

```
set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {C[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {C[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {D[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports F]
set_property IOSTANDARD LVCMOS33 [get_ports S0]
set_property IOSTANDARD LVCMOS33 [get_ports S1]
set_property PACKAGE_PIN R2 [get_ports {A[1]}]
set_property PACKAGE_PIN T1 [get_ports {A[0]}]
set_property PACKAGE_PIN U1 [get_ports {B[1]}]
set_property PACKAGE_PIN W2 [get_ports {B[0]}]
set_property PACKAGE_PIN R3 [get_ports {C[1]}]
set_property PACKAGE_PIN T2 [get_ports {C[0]}]
set_property PACKAGE_PIN T3 [get_ports {D[1]}]
set_property PACKAGE_PIN V2 [get_ports {D[0]}]
set_property PACKAGE_PIN L1 [get_ports F]
set_property PACKAGE_PIN W13 [get_ports S0]
set_property PACKAGE_PIN W14 [get_ports S1]
```