Mustafa Cankan BALCI
22101761
EE-102-01
03/10/2022

# LAB 2: Introduction to VHDL

**Purpose**

The investigation will aim to design and implement a combinational circuit on BASYS 3 by using VHDL. The experiment also focuses on understanding the usage of the Vivado Design Suite program. Vivado Design Suite program is used to write VHDL code for the design and implement a combinational circuit.

**Design Specifications**

A multiplexer was designed in the investigation. It is a combinational circuit that selects and routes several inputs to one output signal. There are different kinds of multiplexers. A 4-to-1 multiplexer was designed in the investigation. The 4-to-1 multiplexer consisted of 2 selective inputs (s1, s2), 4 data inputs (x1, x2, x3, x4) and one output (f). The selective inputs (s1 and s2) were V17 and V16 switches on BASYS3. Moreover, the data inputs (x1, x2, x3, x4) were W16, W17, W15, and V15 switches on the FGPA board. The output of the circuit was U16 led in BASYS3 board. 4-to-1 multiplexer had to pass the data input, which was selected by the values of the selective inputs. The expected result was a truth table which was table 1 in the appendix.

**Methodology**

The methodology of the investigation had four parts, which were coding, designing the combinational circuit, creating the simulation for the design code, creating a constraint file, and uploading on BASYS3. In the design of the 4-to-1 multiplexer, the IF-ELSE statement was used in the design code of the circuit. The design code is in the appendix. An entity was the first part of the design. The interface of the component was declared in entity. The inputs and outputs

were asserted inside of PORT in entity. Another section of the code is architecture. The behavior of the circuit was coded in architecture. The statements were assigned under architecture in the investigation. In the first statement, s1 equals to 0 was defined. Under the statement, sub statement was declared. It was the output equaled to x1 if s2 equaled to 0. However, x2 was the output if s2 was 1. The else section of the first statement was the condition when s1 equaled to 1. In the else part of the first statement, the same process was applied. If s2 was 0, the output was x3. However, the output was x4 if s2 was 1.

Moreover, a testbench code was written to control the behavior of the multiplexer was same as expected. In In the first part, components of the design code were declared for Unit Under Test in architecture part of the code. The design file name was written before in VHDL code for using the design file in the testbench.

After, the selective and data input's signals were declared. All inputs were assigned to 0. Also, the output signal was declared. The time constants for each input were assigned differently for each. Unit Under Test was instantiated. The design code file name was written after UUT, and PORT MAP was declared at the end of the code. PORT MAP was used to create a connection between ports on the design files entity.

Afterward, the process for each input was created. Firstly, 0 was assigned to input. After then, the signal has been 0 for half of the time constant of the corresponding input. When the time passed, one was assigned to input until half of the time constant of corresponding input. These steps applied for each input to create process. Subsequently, stimulus process was used to start simulation. The testbench file ran gave the simulation graph.

Furthermore, constraint file was created for assign the interface of the components on BASYS3. Constraint file used for assigning inputs and outputs of corresponding switches and leds on the Basys3. Finally, BASYS3 is connected to Vivado project. The project was uploaded on BASYS3 board.

**Results**

The RTL schematics of the 4-to-1 multiplexer is figure 1. In RTL schematics, there are 3 2-to-1 multiplexers. The first two 2-to-1 multiplexers' selective input is s2. One of 2-to-1

multiplexer's data inputs are x1 and x2. Other 2-to-1 multiplexer's data inputs are x3 and x4. The results of the first 2-to-1 multiplexers are the data inputs of another 2-to-1 multiplexers. The selective input of it is s1 in the schematics.

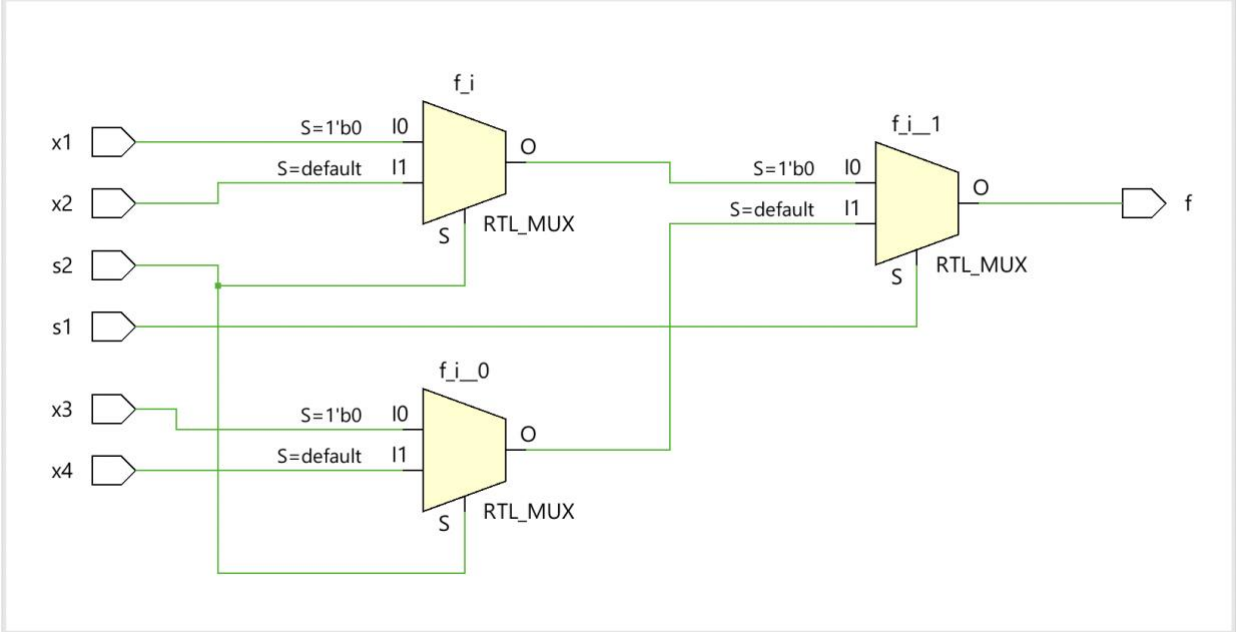


Figure 1: RTL Design

The waveform diagram is the result of the testbench. The waveform graph shows the timing diagram of each input in an orderly. The result of the combinational circuit is f. The timing result was the same as expected according to the truth table. For instance, s1 and s2 are 1, and x4 is one between 150 and 200 ns. The result f is one between the timelines.

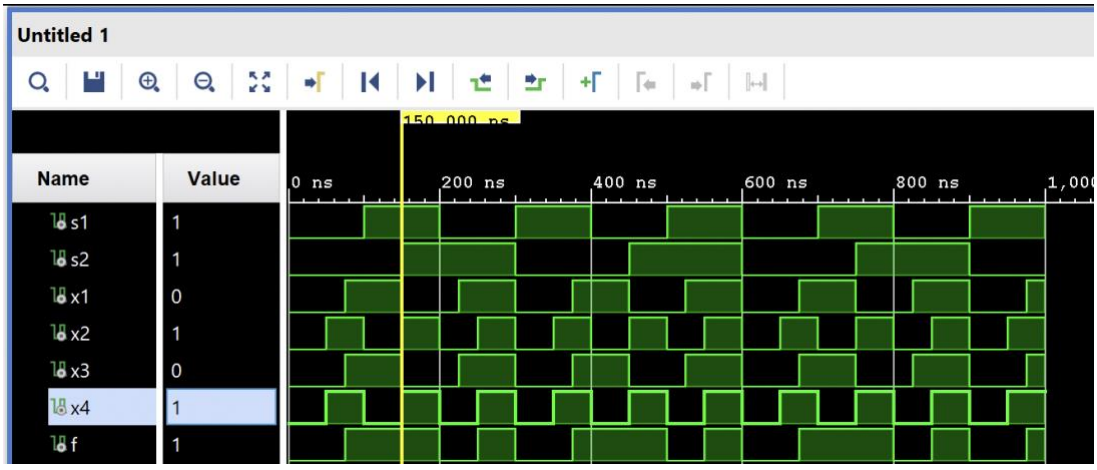


Figure 2: Waveform Graph

The application of the 4-to-1 multiplexer on BASYS 3 is the same as expected. For instance, when V17, V16, and V15 pulled up, the led at U16 turned on and gave a green light. There are also example photos of different inputs for a combinational circuit's BASYS3 implementation.

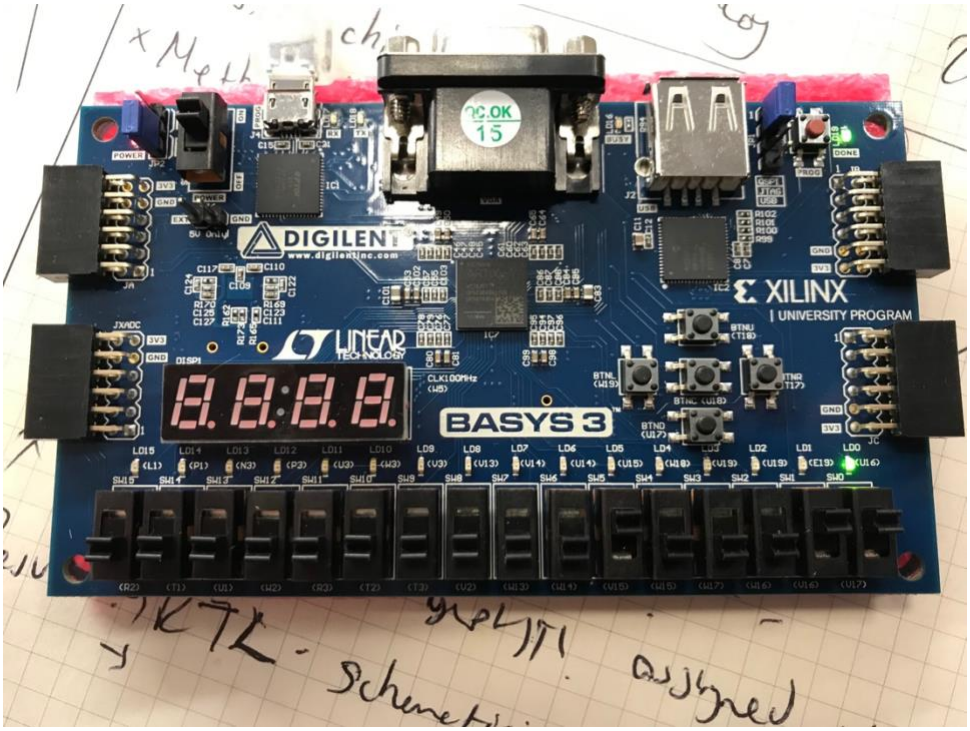


Figure 3: V15, V16 and V17 pulled U16 on

**Conclusion**

In this experiment, a combinational circuit was designed and implemented on BASYS3 by using VHDL. Vivado Design Suite was used the write a code. A 4-to-1 multiplexer was designed for the required combinational circuit. The experiment result came as expected. The waveform graph and the BASYS3 implementation gave the same result as the truth table. In the testbench part of the investigation, different processes were applied. The process was assigned to each input in the testbench part. As a result, the values of inputs were not assigned manually. In this way, the testbench code was more readable.

**Appendix**

| s1 | s2 | x1 | x2 | x3 | x4 | f |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 1: Truth Table

```
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 test1 is
   Port ( s1 : in STD_LOGIC;
          s2 : in STD_LOGIC;
          x1 : in STD_LOGIC;
          x2 : in STD_LOGIC;
          x3 : in STD_LOGIC;
          x4 : in STD_LOGIC;
          f : out STD_LOGIC);
end test1;

architecture Behavioral of test1 is
begin
```

```
PROCESS(x1, x2, x3, x4, s1, s2)
   BEGIN
   IF s1 = '0' THEN
      IF s2 = '0' THEN
         f <= x1;
      ELSE
         f <= x2;
      END IF;
    ELSE
      IF s2 = '0' THEN
         f <= x3;
      ELSE
         f <= x4;
      END IF;
    END IF;
    END PROCESS;
end Behavioral;
```

Code 1: Design Code of 4-to-1 Multiplexer

```
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 testBench is
--  Port ( );
end testBench;

architecture Behavioral of testBench is
-- Component Declaration for Unit Under Test
COMPONENT test1 PORT(
   x1 : IN STD_LOGIC;
   x2 : IN STD_LOGIC;
   x3 : IN STD_LOGIC;
   x4 : IN STD_LOGIC;
   s1 : IN STD_LOGIC;
```

```
    s2 : IN STD_LOGIC;
    f : OUT STD_LOGIC);
  END COMPONENT;
-- Inputs
SIGNAL s1: STD_LOGIC := '0';
SIGNAL s2: STD_LOGIC := '0';
SIGNAL x1: STD_LOGIC := '0';
SIGNAL x2: STD_LOGIC := '0';
SIGNAL x3: STD_LOGIC := '0';
SIGNAL x4: STD_LOGIC := '0';


-- Outputs
SIGNAL f: STD_LOGIC;

-- Constant Times of Inputs
constant x1_period : time := 150ns;
constant x2_period : time := 100ns;
constant x3_period : time := 150ns;
constant x4_period : time := 100ns;
constant s1_period : time := 200ns;
constant s2_period : time := 300ns;
begin
-- Instantiate the Unit Under Test
uut: test1 PORT MAP(
    x1 => x1,
    x2 => x2,
    x3 => x3,
    x4 => x4,
    s1 => s1,
    s2 => s2,
    f => f );

s1_process: process
begin
    s1 <= '0';
    wait for s1_period/2;
    s1 <= '1';
    wait for s1_period/2;
   end process;

s2_process : process
begin
    s2 <= '0';
```

```
   wait for s2_period/2;
   s2 <= '1';
   wait for s2_period/2;
 end process;

x1_process : process
begin
   x1 <= '0';
   wait for x1_period/2;
   x1 <= '1';
   wait for x1_period/2;
 end process;

x2_process : process
begin
   x2 <= '0';
   wait for x2_period/2;
   x2 <= '1';
   wait for x2_period/2;
 end process;

x3_process : process
begin
   x3 <= '0';
   wait for x3_period/2;
   x3 <= '1';
   wait for x3_period/2;
 end process;

 x4_process: process
 begin
   x4 <= '0';
   wait for x4_period/2;
   x4 <= '1';
   wait for x4_period/2;
 end process;

 -- Stimulus process
 stim_proc: process
 begin
   -- hold reset state for 100 ms
   wait for 100 ms;
   wait for s1_period*10;
   -- insert stimulus here
```

```
  wait;
  end process;
END;
```

Code 2: Testbench Code

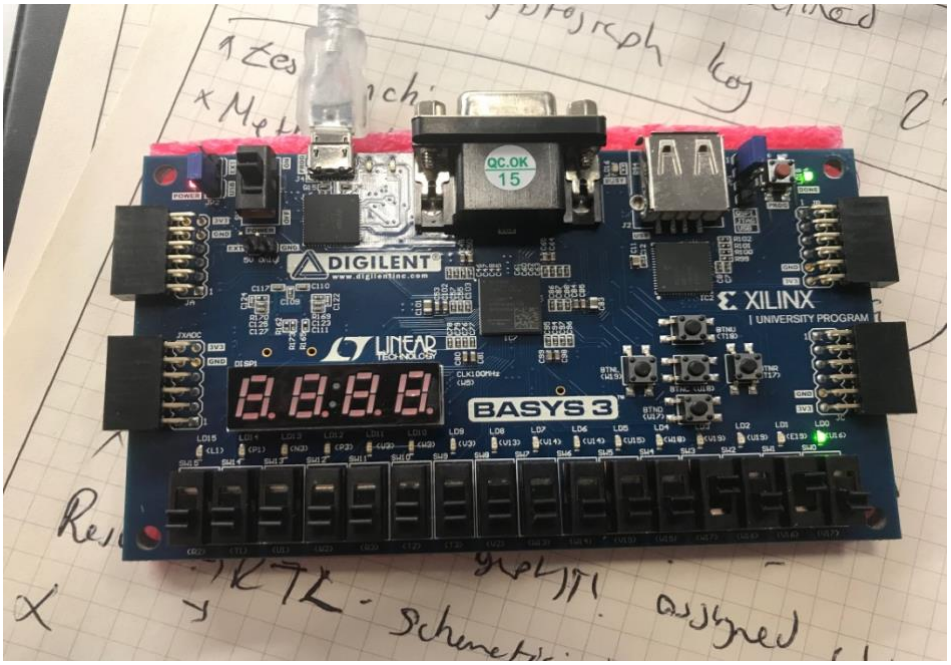


Figure 4: V16 and W17 pulled U16 on

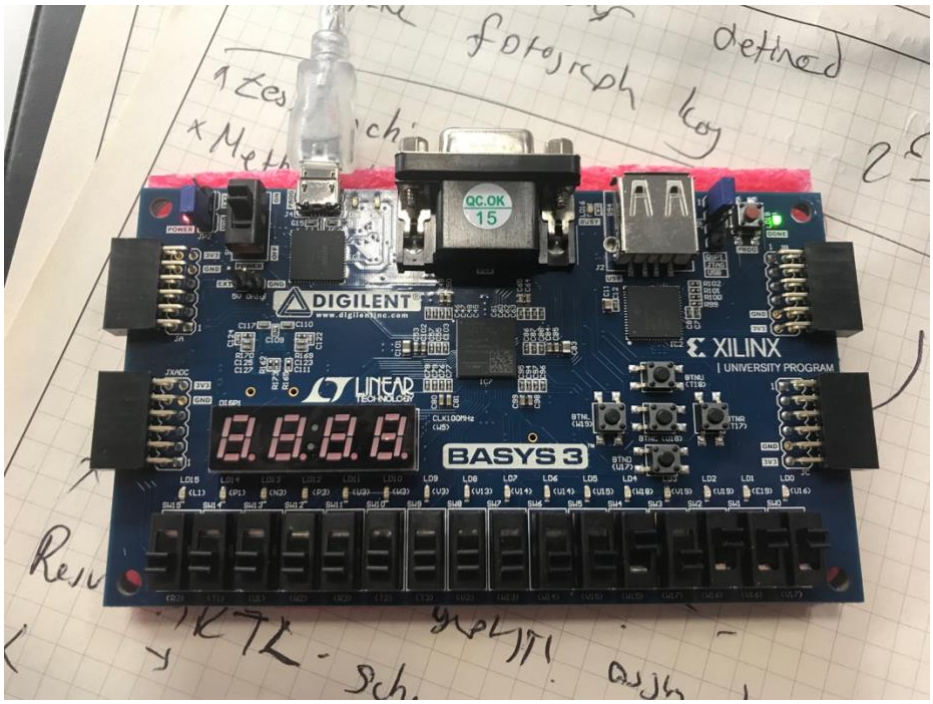


Figure 5: W15, W16, V16, V17 pulled U16 off

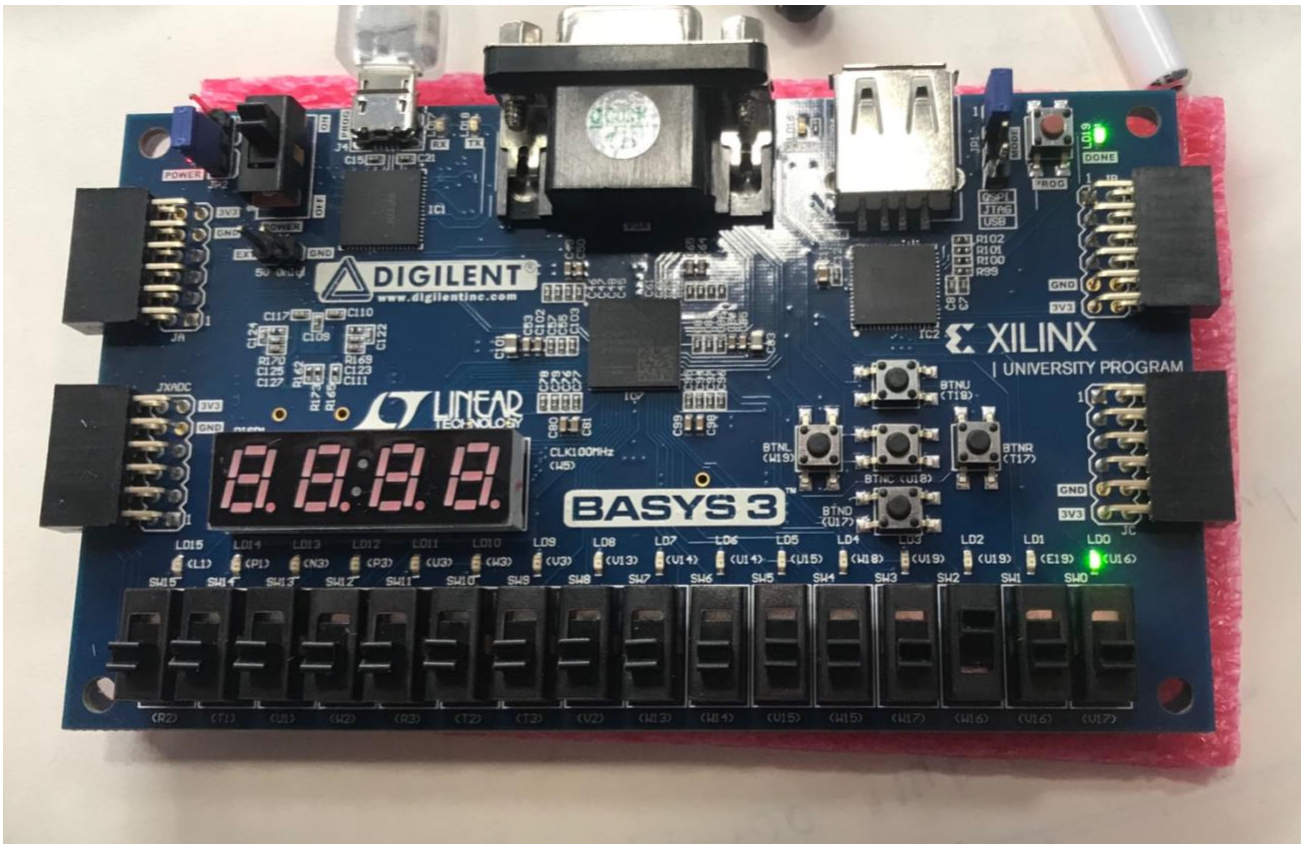


Figure 6:  W16 pulled U 16 on

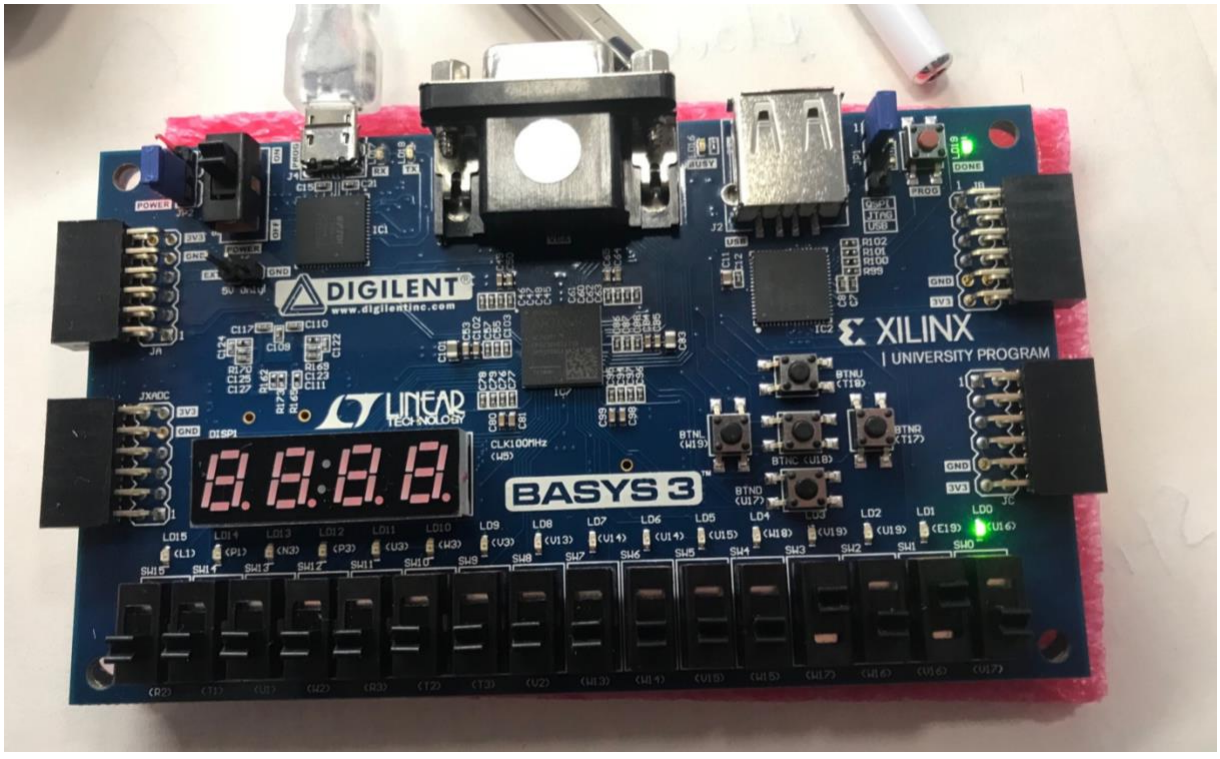


Figure 7: W17 and V16 pulled U16 on

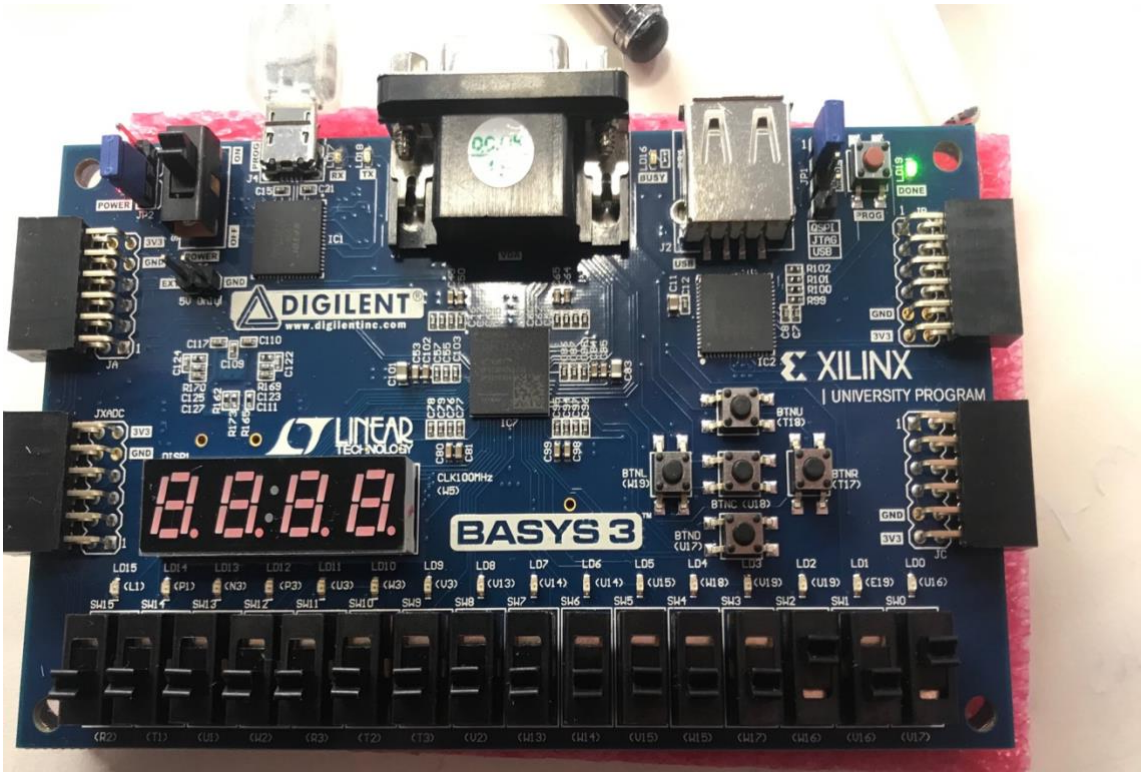


Figure 8: V17 and W16 pulled up U16 off