

EECS 140: Lab 10

Scrolling Display

Morgan Bergen

KUID: 3073682

Date submitted: 12/3/2022

1. Introduction and Background

The objective of this laboratory exercise is to learn how to use modular design in VHDL to display a scrolling phrase up to 16 characters long on the 4 7-segment displays on the Basys 3 board. The components that were used in completing this lab are the following,

clock_divider.vhd

counter.vhd

display_driver_1.vhd

display_driver_2.vhd

display_driver_3.vhd

display_driver_4.vhd

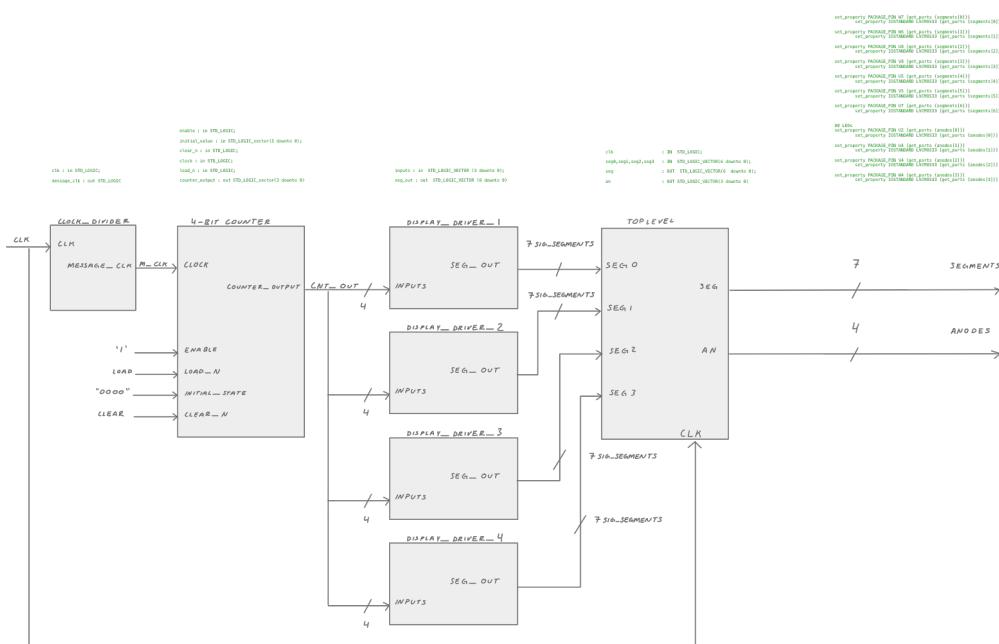
LEDdisplay.vhd

toplevel.vhd

toplevel.xcd

All of the pictures comprising the code to these modules can be found on page 3, 4, & 5.

The total amount of connections (signals) that connected to the counter will have 4 signals connecting to each of the display drivers, the variable for the signals is a vector of length 4. The following shows the block diagram of how this lab was structured.



Project Manager: lab10

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD.TEXT_UNSIGNED.all;

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

entity clock_divider is
  Port ( clk_in : in STD_LOGIC;
        message_clk : out STD_LOGIC);
end clock_divider;

architecture Behavioral of clock_divider is
begin
  process
    variable count: STD_LOGIC_VECTOR(15 downto 0);
  begin
    loop
      if (count >= "1111111111111111") then
        count := count + 1;
      end if;
      report "Count is: " & integer'image(count);
      message_clk <= count(14);
    end loop;
  end process;
end Behavioral;
  
```

To Console | Messages | Log | Reports | Design Runs

clock_divider.vhd

Project Manager: lab10

```

library IEEE;
use IEEE.STD.TEXT_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED_UNSIGNED.all;

entity counter is
  Port ( enable : in STD_LOGIC;
         initial_value : in STD_LOGIC_VECTOR(0 to 31);
         clock : in STD_LOGIC;
         count : out STD_LOGIC_VECTOR(0 to 31));
end counter;

architecture Behavioral of Counter is
begin
  counter: process (enable, clock)
  begin
    if rising_edge(clock) then
      if enable = '1' then
        counter_signal <= initial_value;
      else
        counter_signal <= counter_signal + 1;
      end if;
    end if;
    report "Count is: " & integer'image(counter);
    counter_output <= counter_signal;
  end process;
end Behavioral;
  
```

To Console | Messages | Log | Reports | Design Runs

counter.vhd

Project Manager: lab10

```

library IEEE;
use IEEE.STD.TEXT_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED_UNSIGNED.all;

entity display_driver_1 is
  Port ( i : in STD_LOGIC_VECTOR(15 downto 0);
        seg_out : out STD_LOGIC_VECTOR(0 downto 0));
end display_driver_1;

architecture Behavioral of display_driver_1 is
begin
  process
    begin
      report "No inputs selected";
      seg_out <= "0000000000000000";
    end process;
  
```

To Console | Messages | Log | Reports | Design Runs

display_driver_1.vhd

Project Manager: lab10

```

library IEEE;
use IEEE.STD.TEXT_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED_UNSIGNED.all;

entity display_driver_2 is
  Port ( i : in STD_LOGIC_VECTOR(15 downto 0);
        seg_out : out STD_LOGIC_VECTOR(0 downto 0));
end display_driver_2;

architecture Behavioral of display_driver_2 is
begin
  process
    begin
      with inputs select
        seg_out <= "0000000000000000";
        "1111111111111111" when "0000000000000000";
        "1111111111111111" when "1111111111111111";
        "0000000000000000" when "0000000000000000";
        "1111111111111111" when "0000000000000000";
        "0000000000000000" when "1111111111111111";
        "1111111111111111" when "0000000000000000";
        "0000000000000000" when "1111111111111111";
        "0000000000000000" when "0000000000000000";
        "1111111111111111" when "1111111111111111";
        "0000000000000000" when "0000000000000000";
      end if;
    end process;
  
```

To Console | Messages | Log | Reports | Design Runs

display_driver_2.vhd

Project Manager: lab10

```

library IEEE;
use IEEE.STD.TEXT_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED_UNSIGNED.all;

entity display_driver_3 is
  Port ( i : in STD_LOGIC_VECTOR(15 downto 0);
        seg_out : out STD_LOGIC_VECTOR(0 downto 0));
end display_driver_3;

architecture Behavioral of display_driver_3 is
begin
  process
    begin
      for i in 0 to 15 loop
        if (i = 10) then
          report "No inputs selected";
        end if;
        seg_out <= "0000000000000000";
      end loop;
    end process;
  
```

To Console | Messages | Log | Reports | Design Runs

display_driver_3.vhd

Project Manager: lab10

```

library IEEE;
use IEEE.STD.TEXT_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED.all;
use IEEE.STD.TEXT_UNSIGNED_UNSIGNED_UNSIGNED.all;

entity display_driver_4 is
  Port ( i : in STD_LOGIC_VECTOR(15 downto 0);
        seg_out : out STD_LOGIC_VECTOR(0 downto 0));
end display_driver_4;

architecture Behavioral of display_driver_4 is
begin
  process
    begin
      for i in 0 to 15 loop
        if (i = 10) then
          report "No inputs selected";
        end if;
        seg_out <= "0000000000000000";
      end loop;
    end process;
  
```

To Console | Messages | Log | Reports | Design Runs

LEDdisplay.vhd

File Edit View Tools Report Browse Layout View Help Quick Access

wire_bluetooth Complete DefaultLayout

PROJECT MANAGER - lab10

Header files: counter.h, display_driver_1.h, display_driver_2.h, display_driver_3.h, display_driver_4.h, LIDDDisplay.h, topLevel.adc

Source files: main.cpp, test.h, test.cpp

Binary files: None

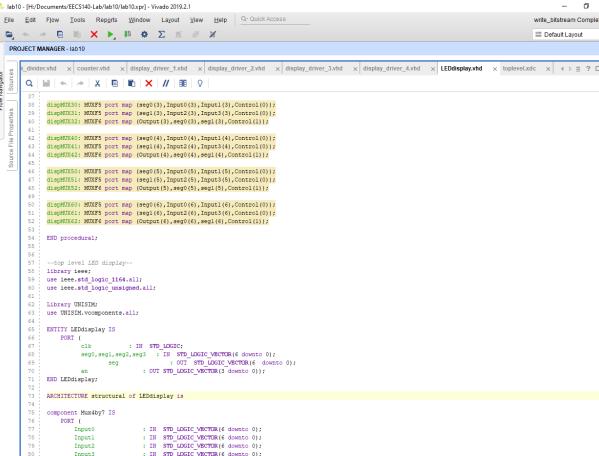
```
1 // This file contains the architecture of the LIDDDisplay
2
3 #ifndef LIDDDisplay_h
4 #define LIDDDisplay_h
5
6 #include "counter.h"
7 #include "display_driver_1.h"
8 #include "display_driver_2.h"
9 #include "display_driver_3.h"
10 #include "display_driver_4.h"
11
12 #include <iostream>
13 #include <vector>
14 #include <string>
15
16 #include <sys/types.h>
17 #include <sys/stat.h>
18 #include <fcntl.h>
19 #include <sys/time.h>
20 #include <sys/types.h>
21 #include <sys/types.h>
22 #include <sys/types.h>
23 #include <sys/types.h>
24 #include <sys/types.h>
25 #include <sys/types.h>
26 #include <sys/types.h>
27 #include <sys/types.h>
28 #include <sys/types.h>
29 #include <sys/types.h>
30 #include <sys/types.h>
31 #include <sys/types.h>
32 #include <sys/types.h>
33 #include <sys/types.h>
34 #include <sys/types.h>
35 #include <sys/types.h>
36 #include <sys/types.h>
37 #include <sys/types.h>
38 #include <sys/types.h>
39 #include <sys/types.h>
40 #include <sys/types.h>
41 #include <sys/types.h>
42 #include <sys/types.h>
43 #include <sys/types.h>
44 #include <sys/types.h>
45 #include <sys/types.h>
46 #include <sys/types.h>
47 #include <sys/types.h>
48 #include <sys/types.h>
49 #include <sys/types.h>
50 #include <sys/types.h>
51 #include <sys/types.h>
52 #include <sys/types.h>
53 #include <sys/types.h>
54 #include <sys/types.h>
55 #include <sys/types.h>
56 #include <sys/types.h>
57 #include <sys/types.h>
58 #include <sys/types.h>
59 #include <sys/types.h>
60 #include <sys/types.h>
61 #include <sys/types.h>
62 #include <sys/types.h>
63 #include <sys/types.h>
64 #include <sys/types.h>
65 #include <sys/types.h>
66 #include <sys/types.h>
67 #include <sys/types.h>
68 #include <sys/types.h>
69 #include <sys/types.h>
70 #include <sys/types.h>
71 #include <sys/types.h>
72 #include <sys/types.h>
73 #include <sys/types.h>
74 #include <sys/types.h>
75 #include <sys/types.h>
76 #include <sys/types.h>
77 #include <sys/types.h>
78 #include <sys/types.h>
79 #include <sys/types.h>
80 #include <sys/types.h>
81 #include <sys/types.h>
82 #include <sys/types.h>
83 #include <sys/types.h>
84 #include <sys/types.h>
85 #include <sys/types.h>
86 #include <sys/types.h>
87 #include <sys/types.h>
88 #include <sys/types.h>
89 #include <sys/types.h>
90 #include <sys/types.h>
91 #include <sys/types.h>
92 #include <sys/types.h>
93 #include <sys/types.h>
94 #include <sys/types.h>
95 #include <sys/types.h>
96 #include <sys/types.h>
97 #include <sys/types.h>
98 #include <sys/types.h>
99 #include <sys/types.h>
100 #include <sys/types.h>
101 #include <sys/types.h>
102 #include <sys/types.h>
103 #include <sys/types.h>
104 #include <sys/types.h>
105 #include <sys/types.h>
106 #include <sys/types.h>
107 #include <sys/types.h>
108 #include <sys/types.h>
109 #include <sys/types.h>
110 #include <sys/types.h>
111 #include <sys/types.h>
112 #include <sys/types.h>
113 #include <sys/types.h>
114 #include <sys/types.h>
115 #include <sys/types.h>
116 #include <sys/types.h>
117 #include <sys/types.h>
```

LEDdisplay.vhd

The screenshot shows the Vivado IDE interface with the following details:

- Title Bar:** The title bar displays "E:\Documents\EEG101\Lab\lab1\toplevel - Vivado 2019.2".
- File Menu:** Includes options like New, Open, Save, Tools, Report, Window, Help, and Exit.
- Quick Access:** A dropdown menu containing "wts_3d_hsiawm_Complex" and "DefaultLayout".
- Project Manager:** Shows "toplevel" as the selected project.
- Source List:** Lists files including "toplevel.vhd", "clock_driver.vhd", "counter.vhd", "display_driver_1.vhd", "display_driver_2.vhd", "display_driver_3.vhd", and "display_driver_4.vhd".
- Code Editor:** The main window displays the VHDL code for "toplevel.vhd". The code includes entities for "toplevel", "clock_driver", "counter", and four "display_driver" components. The "toplevel" entity has a port "clk" and a bus port "display". The "clock_driver" component generates a clock signal "clock_p" based on "display". The "counter" component counts up to 3 and then loops back to 0. The "display_driver" components are responsible for displaying values on a 7-segment display.

toplevel.vhd



The screenshot shows the Vivado 2019.2 IDE interface. The top menu bar includes File, Flow, Tools, Report, Window, Logout, Help, and Quick Access. A toolbar with icons for New, Open, Save, Run, Stop, and Close is located above the main workspace. The left sidebar features a Project Manager titled 'LEDDisplay' with tabs for Sources, Block Diagram, and IP Catalog. Below it is a 'File Navigator' pane. The main workspace displays the Verilog source code for 'LEDDisplay.rhd'. The code defines a module 'LEDDisplay' with various ports and internal logic, including a counter and a 4-to-16 decoder. The bottom status bar shows the current file as 'LEDDisplay.rhd' and the build progress at 0%.

```
project LEDDisplay
  module LEDDisplay;
    input [3:0] Counter;
    output [15:0] Output;
    input [3:0] Input0;
    input [3:0] Input1;
    input [3:0] Input2;
    input [3:0] Input3;
    input [3:0] Input4;
    input [3:0] Input5;
    input [3:0] Input6;
    input [3:0] Input7;
    input [3:0] Input8;
    input [3:0] Input9;
    input [3:0] Input10;
    input [3:0] Input11;
    input [3:0] Input12;
    input [3:0] Input13;
    input [3:0] Input14;
    input [3:0] Input15;
    output [3:0] Control;
  endmodule

  module counter_1_bit;
    input [3:0] Counter;
    output [15:0] Output;
    input [3:0] Input0;
    input [3:0] Input1;
    input [3:0] Input2;
    input [3:0] Input3;
    input [3:0] Input4;
    input [3:0] Input5;
    input [3:0] Input6;
    input [3:0] Input7;
    input [3:0] Input8;
    input [3:0] Input9;
    input [3:0] Input10;
    input [3:0] Input11;
    input [3:0] Input12;
    input [3:0] Input13;
    input [3:0] Input14;
    input [3:0] Input15;
    output [3:0] Control;
  endmodule

  module display_driver_1_bit;
    input [3:0] Counter;
    output [15:0] Output;
    input [3:0] Input0;
    input [3:0] Input1;
    input [3:0] Input2;
    input [3:0] Input3;
    input [3:0] Input4;
    input [3:0] Input5;
    input [3:0] Input6;
    input [3:0] Input7;
    input [3:0] Input8;
    input [3:0] Input9;
    input [3:0] Input10;
    input [3:0] Input11;
    input [3:0] Input12;
    input [3:0] Input13;
    input [3:0] Input14;
    input [3:0] Input15;
    output [3:0] Control;
  endmodule

  module display_driver_2_bit;
    input [3:0] Counter;
    output [15:0] Output;
    input [3:0] Input0;
    input [3:0] Input1;
    input [3:0] Input2;
    input [3:0] Input3;
    input [3:0] Input4;
    input [3:0] Input5;
    input [3:0] Input6;
    input [3:0] Input7;
    input [3:0] Input8;
    input [3:0] Input9;
    input [3:0] Input10;
    input [3:0] Input11;
    input [3:0] Input12;
    input [3:0] Input13;
    input [3:0] Input14;
    input [3:0] Input15;
    output [3:0] Control;
  endmodule

  module display_driver_3_bit;
    input [3:0] Counter;
    output [15:0] Output;
    input [3:0] Input0;
    input [3:0] Input1;
    input [3:0] Input2;
    input [3:0] Input3;
    input [3:0] Input4;
    input [3:0] Input5;
    input [3:0] Input6;
    input [3:0] Input7;
    input [3:0] Input8;
    input [3:0] Input9;
    input [3:0] Input10;
    input [3:0] Input11;
    input [3:0] Input12;
    input [3:0] Input13;
    input [3:0] Input14;
    input [3:0] Input15;
    output [3:0] Control;
  endmodule

  module display_driver_4_bit;
    input [3:0] Counter;
    output [15:0] Output;
    input [3:0] Input0;
    input [3:0] Input1;
    input [3:0] Input2;
    input [3:0] Input3;
    input [3:0] Input4;
    input [3:0] Input5;
    input [3:0] Input6;
    input [3:0] Input7;
    input [3:0] Input8;
    input [3:0] Input9;
    input [3:0] Input10;
    input [3:0] Input11;
    input [3:0] Input12;
    input [3:0] Input13;
    input [3:0] Input14;
    input [3:0] Input15;
    output [3:0] Control;
  endmodule

  module LEDDisplay;
    input [3:0] Counter;
    output [15:0] Output;
    input [3:0] Input0;
    input [3:0] Input1;
    input [3:0] Input2;
    input [3:0] Input3;
    input [3:0] Input4;
    input [3:0] Input5;
    input [3:0] Input6;
    input [3:0] Input7;
    input [3:0] Input8;
    input [3:0] Input9;
    input [3:0] Input10;
    input [3:0] Input11;
    input [3:0] Input12;
    input [3:0] Input13;
    input [3:0] Input14;
    input [3:0] Input15;
    output [3:0] Control;
  endmodule
```

LEDdisplay.vhd

LEDdisplay.vhd

Page 4 of 8

```

lab10 - D:\Documents\EECS140-Lab\Lab10\lab10.vpr] - Viva 2019.2.1
PROJECT MANAGER - lab10
File Edit File Edit Projects Window Layout View Help Q: Quick Access
File Edit File Edit Projects Window Layout View Help Q: Quick Access
PROJECT MANAGER - lab10
File Edit File Edit Projects Window Layout View Help Q: Quick Access

```

The screenshot shows two side-by-side Viva IDE windows. Both windows have tabs at the top: 'File', 'Edit', 'File', 'Edit', 'Projects', 'Window', 'Layout', 'View', 'Help', 'Q: Quick Access'. The left window's title bar says 'lab10 - D:\Documents\EECS140-Lab\Lab10\lab10.vpr] - Viva 2019.2.1'. Its project manager tab is selected. Below it, there are tabs for 'PROJECT SUMMARY', 'clock_divider.vhd', 'counter.vhd', 'display_driver_1.vhd', 'display_driver_2.vhd', 'display_driver_3.vhd', 'display_driver_4.vhd', and 'wira_7segment.vhd'. The code editor shows the 'toplevel.xcd' file, which includes declarations for STD_LOGIC_VECTOR types, components for clock_divider, counter, and four display_drivers, and a process for LD0. The right window has a similar structure but its code editor shows the 'toplevel.xcd' file with a different set of declarations and component instantiations.

toplevel.xcd

toplevel.xcd

2. Implementation Process

Our first task was to define the `clock_divider.vhd`. This component is responsible to take the on-board 450MHz clock input and divide it so that the period of the resulting clock is about 1 second. The name of this clock was `message_clk` and it controlled the speed at which my message will scroll on the 4 7-segment displays. And I was able to test component by hooking it up to an LED (say LD0) and make sure it blinks every 1 second or so). Next was to define, source 2 counter.vhd. I then designed a 4-bit counter that ran at the rate of `message_clk` which is the output of the clock divider, that was now the clock input of the counter). The output was a 4-bit vector called `counter_output`. The counter also has extra inputs: `enable`, `initial_value`, `clear_n`, `load_n`. Then there after was the 4 display drivers (analogous to the `display_driver.vhd` `display_driver_1.vhd` `display_driver_2.vhd` `display_driver_3.vhd` `display_driver_4.vhd`). All of which need to be defined so that the output could look as follows,

```
L 0 V E  
1 2 3 4  display_drivers each column is 1 driver
```

```
-- L  
-- L 0  
- L 0 V  
L 0 V E  
L V E -  
V E --  
E -- L  
-- L 0  
- L 0 V  
L 0 V E  
L V E -  
V E --  
E -- L  
-- L 0  
- L 0 V  
L 0 V E
```

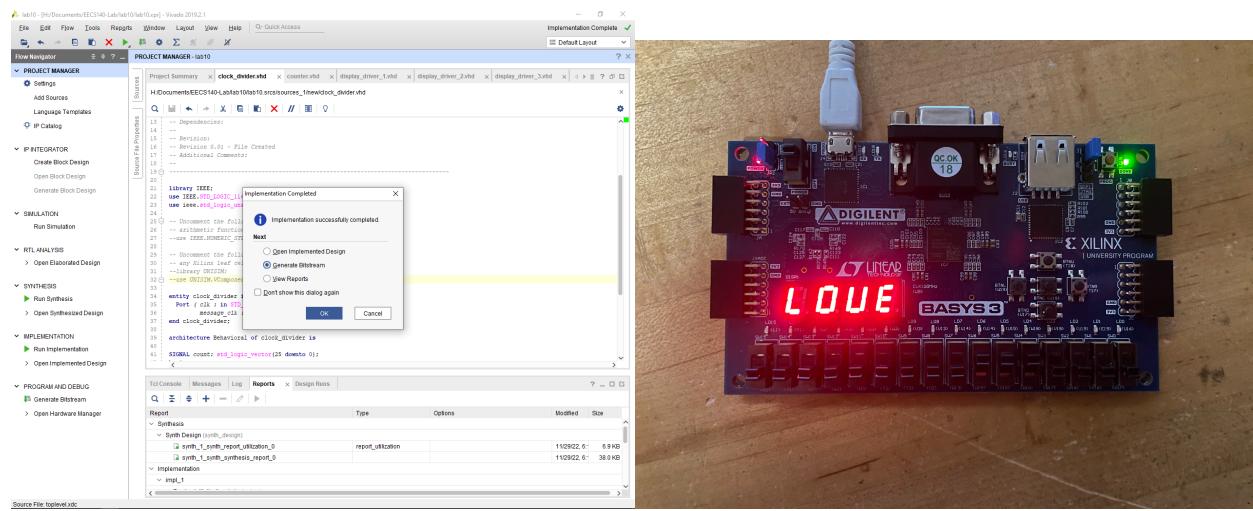
```
L = "1110001" = "abcdefg"  
0 = "0000001" = "abcdefg"  
V = "1000001" = "abcdefg"  
E = "0110000" = "abcdefg"
```

Next was defining the LEDdisplay, then the The toplevel structural VHDL module for the block diagram is provided below, I declared the components and instantiated (port map =>) them to reflect the interconnections from the diagram above. toplevel entity with required input and output ports: clk, segments(7 bits), anodes(4 bits), enable, load, clear, initial_value, declare components clock divider, counter, display_driver_1, display_driver_2, display_driver_3, display_driver_4, LEDdisplay I declared signals m_clk, cn_out, sig_segments1, sig_segments2, sig_segments3, sig_segments4. The component instantiation / port mapping had all ports defined in toplevel.vhd should be mapped using constraints file to implement the design.

Finally was the constants, toplevel.xdc. However before creating constraints file, I needed to ensure that hierarchical design is established over all other sources so to pull from the following constraints and modify variable so the defined Toplevel entity matches with the declared constraints file as well.

3. Evaluation Process

Lastly, in order to evaluate and test the lab we need to check the accuracy of the result of this lab will be tested by using the Basys 3 board and the Xilinx Vivado software. The Basys 3 board will be used to display the scrolling phrase on the 4 7-segment displays. The Xilinx Vivado software will be used to simulate the scrolling phrase on the 4 7-segment displays. I displayed ‘love’ and to looked as follows.

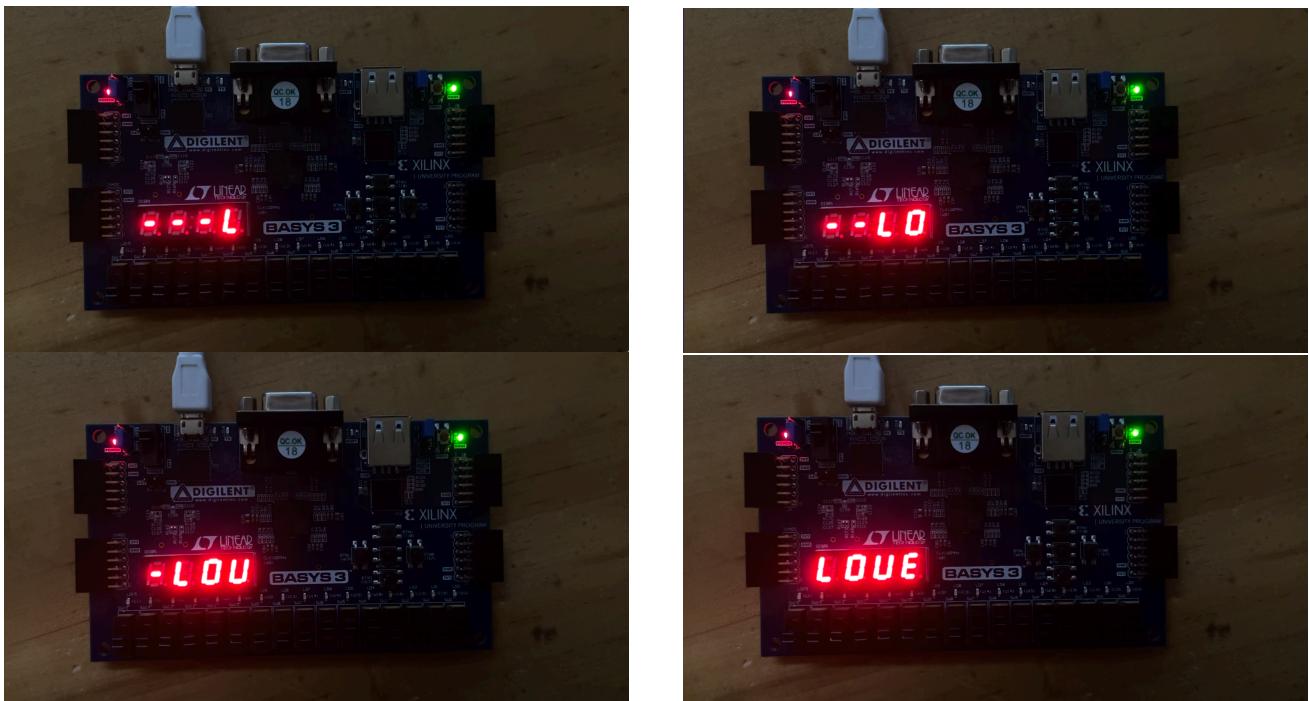


successful implementation

scrolling display

4. Results and Discussion

The result was a scrolling segment display, with the word love. The display driver definitions were the most important component to the output.



5. Conclusion and Recommendations

I properly learned how to use modular design in VHDL in order to display a scrolling phrase that was all the way p to 16 characters long on the board. I really enjoyed this lab and it was the most exciting output I have generated.