EECS 140 Scrolling Display1

Scrolling 7-Segment Display Output

Contents

Objectives

The objective of this laboratory exercise is for you 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.

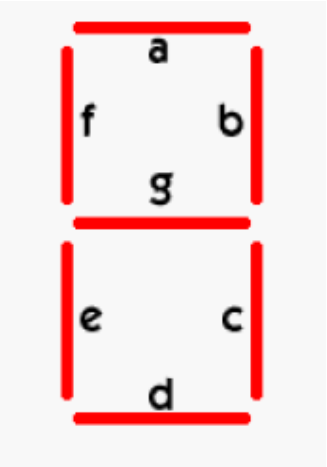

Figure 1: Seven Segment Display

Quiz

Please answer the following questions and submit to your TA at the start of the lab:

1. (Current Lab) What components will be used in completing this lab?
2. How many connections (signals) will connect the counter to the display driver?
3. How will we test the result of this lab?

Block Diagram

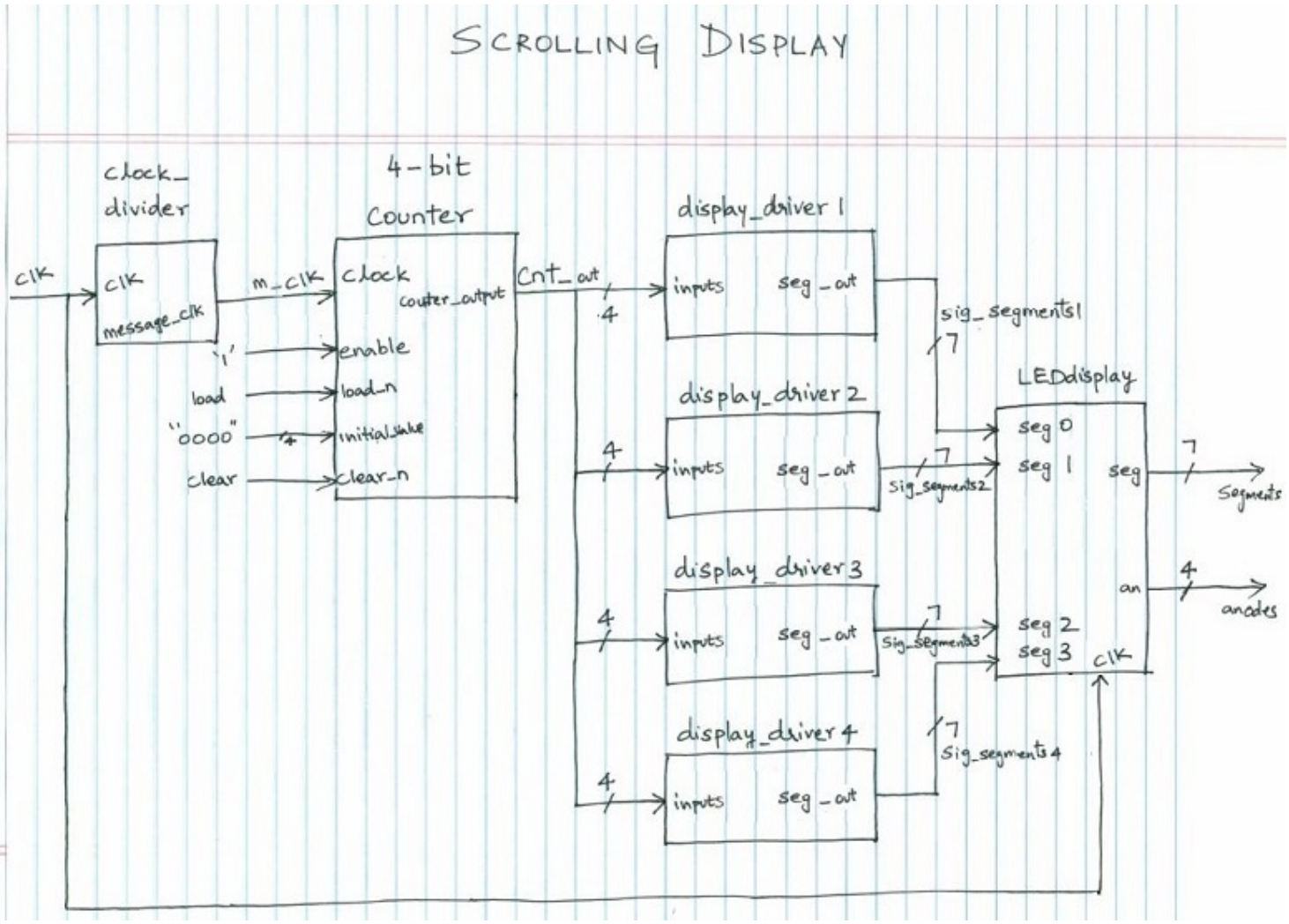

Figure 2: Flow Chart of Scrolling Display

Click here to access the block diagram (Higher Quality)

Components in design

Source1: clock_divider

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 sec. We will call this new clock as *message_clk*. This will control how fast or slow your message will scroll on the 4 7-segment displays. You can test this component by hooking it up to an LED (say LD0) and make sure it blinks every 1 second or so).

- Input : clk (std_logic)
- Output : message_clk (std_logic)

```
architecture Behavioral of clock_divider is
```

```
--Create a signal called "count" (26 bit vector). That is signal declaration.
--Look at the code snippet below which is a counter that depends on clock signal.
```

```
begin
process(clk)
begin
   if (clk'event and clk='1') then
   count <= count + 1;
   end if;
end process;
```

```
  --Assign the 24th bit of the signal count to the output message_clk.
```

```
end Behavioral;
```

- Remember to add std_logic_unsigned to your IEEE Library in top of your VHDL files as follow:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
```

Source2: Counter

- Add std_logic_unsigned to your IEEE Library in top of your VHDL files as follow:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
```

You will then design a 4-bit counter that runs at the rate of message_clk (the output of the clock divider is now the clock input of the counter). The output will be a 4-bit vector called counter_output. The counter also has extra inputs: enable, initial_value, clear_n, and load_n.

- If enable is on (positive logic, On = '1'), the counter increments its value with time, else it remains at the same value.
- When clear_n is '0' (_n = negative logic, On = '0'), the counter resets to zero "0000"
- initial_value is a 4-bit signal that contains an initial value to be loaded into the counter (std_logic_vector)
- load_n (negative logic) loads the initial_value input signal into the counter
- counter_output is the 4-bit counter output (std_logic_vector)

```
architecture Behavioral of counter is
signal counter_signal:std_logic_vector(3 downto 0) :="0000";
begin
process (clock, clear_n)
begin
   if clear_n='0' then
     counter_signal <= (others=>'0');
   elsif (clock'event and clock='1') then
     if load_n = '0' then
     counter_signal <=initial_value;
     else
     if enable ='1' then
     counter_signal <= counter_signal +1;
     else
     counter_signal <= counter_signal;
     end if;
   end if;
   end if;
end process;
```

```
counter_output <= counter_signal;
end Behavioral;
```

Source3-6: Display_Drivers

You will now create 4 display_drivers. Use 'when' statements as we did in lab 7 (click here to look at the display_driver.vhd from lab 7). Note that this was done to display 0 thru F on the 7-segments. You should modify it for your message. Think about how to write the 4 display drivers so that your message "scrolls" on the 4 displays from left to right.

- NOTE: The 4 copies of display driver source files should have different entity names! (Ex: display_driver1, display_driver2 etc), but the port names can remain same.
- Display your name or any creative phrase similar to the example below,has to be unique to you to score points

```
- - - E
- - E E
- E E C
E E C S
E C S -
C S - -
S - - E
- - E E
- E E C
E E C S
E C S -
C S - -
S - - E
- - E E
- E E C
E E C S
```

- Each column above corresponds to each display driver

Source7: LEDdisplay.vhd

This component is used to switch between the outputs of display_driver1, display_driver2, display_driver3, and display_driver4.

- LEDdisplay.vhd

Source8: Toplevel.vhd

You now will write a toplevel structural VHDL module for the block diagram provided. You will need to declare the above components and instantiate (port map) them to reflect the interconnections shown on the handout.

1. Create Toplevel entity with required input and output ports. (clk, segments (7bits), anodes(4bits), enable, load, clear, initialvalue(4bits))
2. Declare components clock divider, counter, Display drivers 1 to 4, LEDdisplay
3. Declare Signals m_clk, cnt_out, sig_segments1,sig_segments2,sig_segments3,sig_segments4 (which are neither inputs nor outputs) required for

toplevel.

```
Syntax: signal signal_name: std_logic; --for signals which store 1 bit values
        signal signal_name: std_logic_vector(N downto 0); --for signals which store a vector of bits of length N+1;
```

1. Component instantiation/ port mapping for above declared components.

2. All the ports defined in Toplevel.vhd should be mapped using constraints file to implement the design.

```
NOTE: Port mapping symbol is "=>". Instance name should be different from component name
Syntax: Instance_name: Component_name PORT MAP
                          (portname1=> connection1,
                             .           .
                             .           .
                           portnameN=> connectionN);
       Instance_name -> user defined name ex: display_driver1
       Component_name -> name of the component that is being port mapped, i.e, the name used after key word entity ex: display_driver
       portname1,portname2.. etc -> are input and output ports of the component being port mapped ex: inputs,seg_out
       connection1, connection2.. -> are the respective signals/ports of toplevel being connected to the ports of THE component
```

toplevel.XDC

- Before you create constraints file right click on your toplevel.vhd file under sources and set it as top file.

```
NOTE: All PORTS defined in Toplevel entity should be declared in the contraints file as well.
```

Basys3_Constraints

Lab report

Write your lab report as per the instructions by TA.

Retrieved from "https://wiki.ittc.ku.edu/ittc_wiki/index.php?title=EECS_140_Scrolling_Display1&oldid=22981"

- This page was last edited on 5 February 2021, at 14:56.