Christopher Spadavecchia
Eli Shtindler
CPE 487

Lab 1

Project 1 - leddec:

```
set_property PROBES.FILE {} [get_hw_devices xc7a100t_0]
set_property FULL_PROBES.FILE {} [get_hw_devices xc7a100t_0]
set_property PROGRAM.FILE {C:/Users/cspad/leddec/leddec.runs/impl_1/leddec.bit} [get_hw_devices xc7a100t_0]
program_hw_devices [get_hw_devices xc7a100t_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
```

After uploading the code to the board was successful (the aftermath of the 'Program Device' step), the following lines were shown in the console.

Gray Code

In the video, flipping one of the first four switches (switches 0, 1, 2,3) adds its corresponding hexadecimal value to the counter. For example, flipping switch 3 adds 8 to the total. The counter continuously sums these values and displays the result on the board's segmented LEDs. In the second part of the video, the three leftmost switches (switches 13, 14, 15) control the counter's movement in binary. Flipping switch 15 shifts the counter 4 spaces to the left. When multiple switches are flipped, the counter moves by their combined value, with the maximum shift being 7 spaces to the left.

Project 2 - hexcount:

hexcount.vhd:

Original:

```
L1 : leddec
PORT MAP(dig => "000", data => S, anode => anode, seg => seg);
```

Modified:

```
L1 : leddec
PORT MAP(dig => S(2 DOWNTO 0), data => S, anode => anode, seg => seg);
```

The only thing changed in hexcount was the port map of the leddec component. Instead of assigning "000" to dig, digits 2-0 of S were used.

counter.vhd:

Original:

```
ARCHITECTURE Behavioral OF counter IS
	SIGNAL cnt : STD_LOGIC_VECTOR (28 DOWNTO 0); -- 29 bit counter
BEGIN
	PROCESS (clk)
	BEGIN
		IF clk'EVENT AND clk = '1' THEN -- on rising edge of clock
			cnt <= cnt + 1; -- increment counter
		END IF;
	END PROCESS;
	count <= cnt (28 DOWNTO 25);
END Behavioral;
```

Modified:

```
ARCHITECTURE Behavioral OF counter IS
	constant counter_bits : integer := 25;
	SIGNAL cnt : STD_LOGIC_VECTOR (counter_bits-1 DOWNTO 0); -- counter
BEGIN
	PROCESS (clk)
	BEGIN
		IF clk'EVENT AND clk = '1' THEN -- on rising edge of clock
			cnt <= cnt + 1; -- increment counter
		END IF;
	END PROCESS;
	count <= cnt (counter_bits-1 DOWNTO counter_bits-4);
END Behavioral;
```

A new variable called counter_bits was made. The signal cnt was changed to be of length counter_bits instead of 29 bits. The indices used to assign the value to count were also changed to be the last 4 bits of cnt.

Hex Counter Moving at a Regular Rate

Here the effects of the change to hexcount.vhd can be seen. When the value of S changes, both the value displayed and the position of the number change. As S counts up the digit's position moves from right to left.

Hex Counter Moving at a Sped Up Rate

By changing the counter_bits variable, the rate at which the number changes. Only the last 4 bits of the counter are used in hexcount. Making the counter larger causes the last 4 bits to change at a slower rate and making the counter smaller causes the last 4 bits to change faster. In the video counter_bits was set to 25.