Jack Burton
TAs: Cory and Karen

Summary: Combining a sequential circuit with a combinational logic trick to display a 2-digit number in base 10 on a seven segment display.

Bench: 15

Prelab

88MHz oscillator = 88,000,000 cycles/sec
Top bit = 2^25 = 33,554,432

Therefore, time for one LED to go from on->off->back on = 48000000/33554432 = 1.43sec
Measured to be 1.46sec, so very close.

Decided to use DIP switch.

Lab Journal

L1

Already built and flashed FPGA in pre-lab
For counter(25): 1.46 sec measured
For counter(24): 0.5 sec measured
For counter(23): 0.2 sec measured
For counter(22): Too quick to measure but still visibly blinking
Counter(20): Still strobing
Counter(19): Barely strobing but still enough to hurt my eyes
Counter(18): No longer visibly blinking at all!

L2

Red: A
Green: B
Blue: C
Grey: D
Yellow: E
Brown: F
Orange: G

Wired according to color guide above and seven-seg reference manual

L3

Using DIP switch for number generation.
Wired DIP switch as follows:
Count[0]: Red, pin 28
Count[1]: Green, pin 38
Count[2]: Blue, pin 42
Count[3]: Grey, pin 36
Count[4]: Yellow, pin 43
Count[5]: Brown, pin 34

Checked LSB for various values and each LSB made sense, so went ahead to L4 (adding mux for second digit).

L4

Used process block to implement a mux that enabled digit 0 when LED_0 was high, or digit 1 when LED_1 was high.

Tested random assortment of digits:

Binary	Expected	Actual
000000	00	00
000001	01	01
000011	03	03
000111	07	07
001111	15	15
011111	31	31
111111	63	63
010101	21	21
010100	20	20
110101	53	53
001000	08	08

This was enough test cases to cover many different number values in digits 0 and 1, which convinced me that they both work properly.

Design Description

For pin mappings see L3 and circuit photos.
"LED0/LED1" are std_logic values to enable digits 0 or 1, respectively (oscillating signals).
"7-seg digit 0 or 1" is a 7-bit std_logic_vector to hold value of digit 0 or 1 depending on LED0/LED1.
"Binary input" is the 6-bit binary number from the DIP switch to be displayed in decimal on 7-seg.
7-seg pin mappings:

Complete Circuit Photos

Testing

Intermediate tests are recorded in lab journal, with final test results recorded in L4. I sequentially tested after each step, so I did not encounter any major issues other than misplacing a single wire connection to the seven-segment, which I quickly debugged by looking at the reference for the seven-seg and swapping the two wires accordingly.

Questions

What was the most valuable thing you learned, and why? I learned that using clever tricks, like super quick oscillations for displays, can significantly reduce necessary hardware for certain implementations. This is very useful when building much bigger displays than this, as it cuts down the amount of wires needed greatly, and therefore reduces the cost.

What skills or concepts are you still struggling with? What will you do to practice these? VHDL still confuses me since I came from a software background, although I was more comfortable and had less issues with it on this lab versus the previous one, so I think that just doing more VHDL coding and flashing onto FPGAs will improve my skills in VHDL.

How long did it take you to complete the lab? 2.5hrs

VHDL Code

Top.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity top is
port (
input_num : in unsigned(5 downto 0);
current_display : out std_logic_vector(6 downto 0);
LED_0 : out std_logic;
LED_1 : out std_logic
);
end top;

architecture synth of top is
component HSOSC is
generic (
CLKHF_DIV : String := "0b00");
port (
CLKHFPU : in std_logic := 'X';
CLKHFEN : in std_logic := 'X';
CLKHF : out std_logic := 'X');
end component;

component counter is
port(
clk : in std_logic;
result : out std_logic_vector(25 downto 0)
);
end component;

component dddd is
port (
count : in unsigned(5 downto 0);
digit0 : out std_logic_vector(6 downto 0);
digit1 : out std_logic_vector(6 downto 0)
);
end component;
signal clk : std_logic;
signal int : std_logic_vector(25 downto 0);
signal display_pins0 : std_logic_vector(6 downto 0);
signal display_pins1 : std_logic_vector(6 downto 0);
begin
osc : HSOSC generic map ( CLKHF_DIV => "0b00") -- mapping oscillator
to clock
port map (CLKHFPU => '1',
CLKHFEN => '1',
CLKHF => clk);
dut : counter port map (clk, int); -- mapping counter to LEDs
-- Logic to blink LEDs rapidly
LED_0 <= not int(18);
LED_1 <= int(18);
dual_display : dddd port map (input_num, digit0 => display_pins0,
digit1 => display_pins1); -- mapping dddd to top
process (LED_0, LED_1, display_pins0, display_pins1) is
begin
if LED_0 = '1' then
        current_display <= display_pins0; -- Show digit0
    elsif LED_1 = '1' then
        current_display <= display_pins1; -- Show digit1
    else
        current_display <= display_pins0; -- other case
    end if;
end process;
end;
```

Dddd.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


entity dddd is
port(
count : in unsigned(5 downto 0);
digit0 : out std_logic_vector(6 downto 0);
digit1 : out std_logic_vector(6 downto 0)
);
end dddd;

architecture synth of dddd is
component seveseg is
port(
D : in unsigned(3 downto 0);
segments : out std_logic_vector(6 downto 0)
);
end component;
signal lowBCD : unsigned(3 downto 0);
signal highBCD : unsigned(3 downto 0);
signal tensplace : unsigned(12 downto 0);
begin
-- Do the math to split up the digits. Input `count` is 6 bit unsigned
lowBCD <= count mod 4d"10"; -- Low digit result is 4 bit unsigned
-- Multiply by 52. Intermediate term is 13 bit unsigned
tensplace <= count * 5d"52";
-- Divide by 512 (2^9). High digit result is 4 bit unsigned
highBCD <= tensplace(12 downto 9);
sevenseg_0 : seveseg port map (highBCD, digit0);
sevenseg_1 : seveseg port map (lowBCD, digit1);
end;
```

Counter.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


entity counter is
port(
clk : in std_logic;
result : out std_logic_vector(25 downto 0)
);
end counter;

architecture synth of counter is
signal count : unsigned(25 downto 0);
begin
process (clk) begin
if rising_edge(clk) then
count <= count + 1;
end if;
end process;
result <= std_logic_vector(count);
end;
```

Sevenseg.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


entity seveseg is
  port(
  D : in unsigned(3 downto 0);
  segments : out std_logic_vector(6 downto 0)
  );
end seveseg;


architecture synth of seveseg is
begin
  with D select
    segments <= "0000001" when "0000",
                "1001111" when "0001",
                "0010010" when "0010",
                "0000110" when "0011",
                "1001100" when "0100",
                "0100100" when "0101",
                "0100000" when "0110",
                "0001111" when "0111",
                "0000000" when "1000",
                "0001100" when "1001",
                "0001000" when "1010",
                "1100000" when "1011",
                "0110001" when "1100",
                "1000010" when "1101",
                "0110000" when "1110",
                "0111000" when "1111",
                "1111111" when others;

end;
```