Lab 19: Building an 4-Digit Decimal Down-Counter with 4-Digit Readout which starts at a user selected value.

In this lab you will build upon your previous experience with the Digilab board.

Submission of the lab consists of:

1) (0 pts) a cover sheet with your name and section,
   2) (110 pts total) a print out the VHDL programs (all modules included, except for the four_dig_seven_seg_w_dp_decoder and its modules, which you only need to include in part A),
      15 pts part A
      15 pts part B
      50 pts part C
      30 pts part D
3) (40 pts total) a print of the constraints file (*.xdc) [Delete the unused sections before printing.]
      10 pts part A
      10 pts part B
      10 pts part C
      10 pts part D
4) Demonstrate the working project (part D only) on the Digilab board to your instructor.

Failure to demonstrate your project will cause your grade for the above to be discounted by 50%. A circuit that does not meet the required specifications will result in your lab being discounted by a minimum of 10%.

New in this lab are:

- Counters
- Binary Coded Decimal
- Variable type conversion

Use the LF_clock_source to drive the counter with the 10 Hz output.

Part A) Build the four_dig_seven_seg_w_dp_decoder with selectable decimal point. When you make your counter in part C, you will preload the counter with a decimal number input using the slide switches that represents time in tenths of seconds. You will need the decimal point functionality so the display makes sense. In this part you will create a top level test program to interact with your decoder. Modify the four_dig_seven_seg_decoder you developed in an earlier lab to the display the decimal point, call this new module, four_dig_seven_seg_w_dp_decoder. This version of the decoder will take an additional new 4-bit input array describing the decimal point to be illuminated, ‘1’ for illuminated and ‘0’ for off. Add the modified decoder to your top level test module. For the test module use btnR, btnD, btnL, and btnU to control each bit in the decimal point array. And sw(15-0) as the binary value to display.

Part B) Build the 4-digit BCD readout for sw(15-0). When you make your counter in part B, you will preload the counter with a decimal number input using the slide switches that represents time in tenths of seconds. Computers store numbers in binary format and all logical and arithmetic operations on those numbers occur using those formats. However displaying a number in decimal (base 10) on your display requires a different strategy. This is commonly performed using binary coded decimal (BCD) format. This is simply a value of each decimal digit expressed as a 4-bit binary number. Each nibble (4-bit group of switches) represents a decimal digit. You need to read the switch values for each digit and store the values in a 16-bit STD_LOGIC_VECTOR “bcd_start” which can serve as input to your display. However because each nibble can represent values 0-15, beyond the range of a decimal digit, we will need to logically filter these values so that any value greater than 9 becomes 0.

Use the four_dig_seven_seg_w_dp_decoder as a component to drive your display. The “bcd_start” will be the data sent to the decoder for display. Each group of 4 switches should select a decimal number 0-9 on one digit, sw(15-12)→digit3, sw(11-8)→digit2, etc. Switch values hex A-F should be filtered to 0 by your code.

Part C) Build the 4-digit decimal down counter. You will create a down counter running at approximately 10 Hz from the LF_clock_source. A counter is simple to implement in VHDL using the integer variable type. The arithmetic operations: addition, subtraction, multiplication, and division, are defined for the integer type. Load the counter variable “count” with “count_start” and subtract 1 from it at each tic of the 10 Hz clock:

```
counter : procedure (clock) is
begin
    if rising_edge(clock)
      count <= count - 1;
    end if;
end process counter;
```

For use in the counter, you will need to convert “bcd_start” to an integer “count_start” with the value of the decimal number displayed. Hint: Do this one decimal digit at a time. Take the integer value of each digit multiplied by the value of its place (1, 10, 100, etc.) and sum the result. You will need to use two variable type conversions, as there is no direct path to convert a STD_LOGIC_VECTOR to INTEGER, e.g.

```
s_dig1 <= to_integer( unsigned(bcd_start(7 downto 4) );
counter_start <= s_dig3*1000 + s_dig2*100 + s_dig1*10 + s_dig0;
```

To display the value of counter in decimal, you will need to convert “count” to its four digit BCD representation. Hint: The value of the least significant decimal digit is the reminder after division by 10, e.g. 4 = 24 rem 10. Also dividing an integer by 10 effectively shifts the decimal digits to the right because integer division truncates the result. In this way we can extract the value of each of the four decimal digits and covert them to a 4-bit STD_LOGIC_VECTOR.

```
c_dig1 <= std_logic_vector(  to_unsigned( (count/10 rem 10) , 4 )  );
bcd_counter <= c_dig3 & c_dig2 & c_dig1 & c_dig0;
```

You now have two different four digit decimal numbers to display, the values of the counter (bcd_counter) and of the switches (bcd_start). Your program will display bcd_start when btnC is pressed, otherwise it will display bcd_counter.

You will need to develop your code so that counter as has the behavior described in the specifications below.

Here are the specifications your circuit (part C) must meet.

- The counter is a down counter.
- The counter will stop at 0.
- The counter will display and work for values 999.9 to 000.0 and display the digits in the correct sequence.
- btnR: Loads the value counter_start into the counter. Pressing btnR while the counter is running stops the counter and resets the counter to counter_start. Pressing bntR multiple times after the counter is reset does nothing further.
- btnD: Starts the counter. Pressing bntD multiple times after the counter is reset does nothing further. Pressing btnD after the counter has stopped (at 0) does nothing.
- btnC: Displays the counter start value selected by the switches.
- sw(15-0) provides the BCD input for the counter start value viewed directly by pressing btnC.

Part D) Add a visual alarm. When the counter reaches zero, flash led(15-0) at 10 Hz for 10 seconds.

Here are the specifications your circuit (part D) must meet.

- All the specifications of part C.
- The alarm only activates when the counter reaches zero.
- The alarm stops after approximately ten seconds.
- The alarm stops if the counter is reset (bntR).
- The alarm activates whenever the counter reaches zero.

Debugging hints. The leds are very useful for debugging. You can use them to observe the states of your logic while it is operating. In my work, I set up a simple statement to switch between debug mode debug='1' and norma1 debug='0'. For this purpose I declared 2 additional 16-bit arrays, "dled" for the debug state and "aled" for the normal (alarm) state. The alarm flashing is written to aled, while the dled is assigned the various logic signals I wanted to probe. The signal "debug" selected whether dled or aled would be assigned to led, the signals driving to the physical leds.

```
debug <= '1';
led <= dled when debug = '1' else aled;
```