EE 343, Lab #9: Reaction Game

Generic VHDL and standard counters; frequency dividers; push-button keys, registers, and 7-segment-displays

Assigned 11/22/2022
(this is a two-week lab)
Report due 12/06/2022 by 2:00 pm

INTRODUCTION

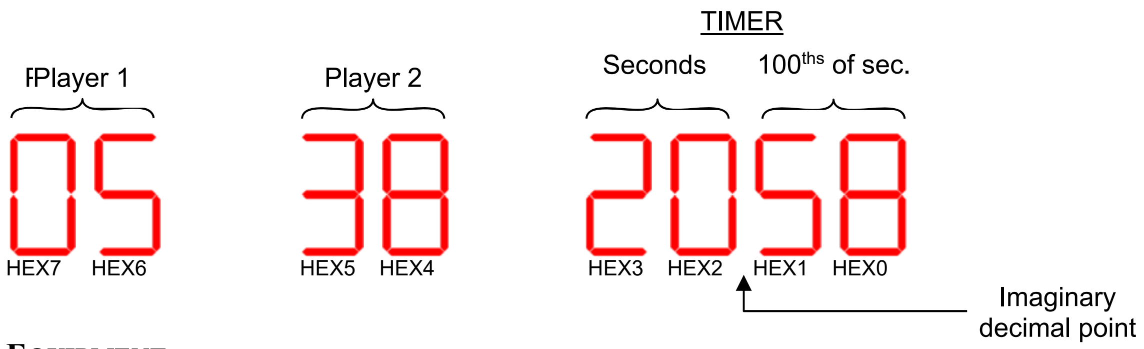
The goal of today's lab exercise is to design a reaction game for two players. Here's how the game is played:

- A timer with a 10 ms resolution (1/100 of a second) is started and displayed on HEX digits;
- The players agree on a specific time (for example, 12.00);
- Each player tries to press a push-button key (e.g., KEY3 and KEY0) as close as possible to the agreed time, but without going over (i.e., if you hit it when it's 12.01, you lose).

The timer should be implemented as a four-digit **decimal** counter (i.e. each digit should count from 0-9) and should be displayed on digits HEX3 – HEX0 (see figure). HEX3 and HEX2 should display seconds (i.e., HEX2 digit should change once per second), HEX1 and HEX0 should display hundredths of a second.

When a player presses one of the keys (e.g., KEY3 for Player 1, KEY0 for Player 2) the current fractions count should be captured and displayed on digits HEX7, HEX6 for Player 1 and HEX5, HEX4 for Player 2 (see figure).

Switch SW17 should be used to start/stop the counter.



EQUIPMENT:

1. Intel Quartus Primes
2. Terasic DE2-115 Education Kit
3. Oscilloscope

PROCEDURE:

1. Create a paper draft of your design. Draw a schematic diagram, paying special attention to connections between components. Try to create (or change) components in such a way to make it easier to connect them (for example, decide whether the outputs will be created as individual single-bit outputs or as vectors).
2. Design a **generic** VHDL counter that generates a short pulse every time group HEX1 HEX0 (1/100th seconds counter) should be incremented. <u>The template is given on the next page</u>. The counter will use the 50 MHz clock from the DE2-115 board. The parameters of this generic VHDL component should be the width (the number of bits) and the limit (when it counts up to the limit it should generate a pulse and restart the counter).

   Calculate the required clock frequency and enter it here: ____________________

   This means that the limit should be: ____________________

   The width (# of bits) of your counter should be: ____________________
3. Locate your 4-bit-binary-number-to-seven-segment-LCD components from one of the previous lab assignments. Create a symbol out of this component, or simply copy the corresponding *.bsf and *.vhd files into the folder used for this lab assignment.
4. Implement your design for the game as a **schematic diagram**. Use a standard 4-bit **74163** counter (available in the "*others->maxplus2*" library) for each of the digits. Use any 8-bit register (for example, **74377**) to capture the fractions counter value. Make this a modular design. Work in steps:
   a. Add your newly designed counter and test it. *Route the output (the short pulse) to one of the pins of the expansion port on the board and capture it. Measure the width of this signal, save the oscilloscope image and attach it to your report.* Because the pulse will be very short, use a flip-flop to turn it into a 50% duty clock and measure the frequency of that clock. *Is the frequency of that clock the one you wanted to produce? If not, what should be the new* ***limit*** *in order to keep the fractions counter counting correctly? Capture the image from the oscilloscope showing the newly produced clock and its frequency. Attach this to your report as well.*

      The new limit should be: ____________________

      The new width of the counter (frequency divider): ____________________

   b. Add the first two digits (*fractions counter*) and make sure they work
      i. Route the *seconds counter* to one of the output pins and measure it using the oscilloscope. Is it really a one-second clock?
   c. Add the next two digits
   d. Add the necessary logic for the first player and make sure it works as it should.
   e. Finally, add the second player (it will probably be a simple copy and paste, with some signal name changes).

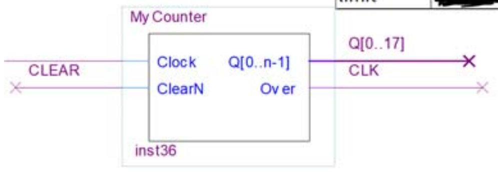
Don't forget to import the .QSF file with the assignments and to use the appropriate pin names.

Demonstrate the functionality of the design to your TA, including the signals captured by the oscilloscope. For the report, submit only answers to questions, VHDL code, captured oscilloscope images, and your schematic diagrams. You do not need to include descriptions and procedures.

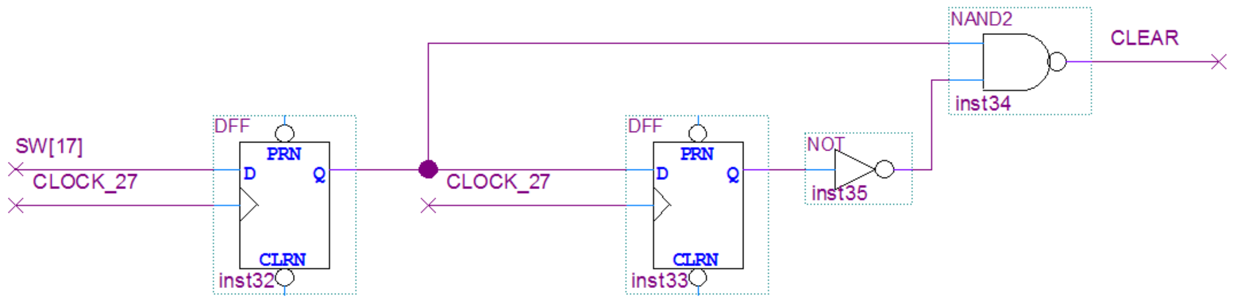
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY MyCounter IS
        GENERIC ( N : INTEGER := __________; limit : INTEGER := _________ );
        PORT  ( Clock, ClearN :       IN STD_LOGIC;
                          Q :         BUFFER STD_LOGIC_VECTOR(0 TO N-1);
                          Over :      BUFFER STD_LOGIC);
END MYCounter;

ARCHITECTURE Behave OF MyCounter IS
BEGIN
        PROCESS

        BEGIN
                Wait Until Clock'EVENT AND Clock='1';
                If ClearN = '0' THEN
                        Q <= (OTHERS => '0');
                Else
                        Q <= Q + 1;
                End If;
                If Q >= limit THEN
                        Over <= Over XOR '1';
                        Q <= (OTHERS => '0');
                End If;

        END PROCESS;
END Behav;
```

Creates this:

Parameter	Value	Type
N	[redacted]	Unsigned Integer
limit	[redacted]	Unsigned Integer



This schematic diagram could be useful if you need to create a short negative pulse when something is turned on (*for example, to reset a counter*):



This will generate a short negative pulse
when ENABLE (SW[17]) is flipped ON