Lab 6
Using PicoBlaze. Speed Punching Game

In this lab, you will program a PicoBlaze microcontroller to interact with various VHDL components in order to implement a game. In this game, the FPGA will repeatedly display a sequence of numbers on the seven segment displays. Players must then enter the indicated sequence on the buttons as fast as possible. Players receive a score based on how fast they input the sequence. After having played a few games, users can then browse a history of all the games as well as view high scores for each player. The high scores are then optionally transmitted back to a PC in a readable format.
Task 1:

Implement the block diagram shown above, consisting of the PicoBlaze microcontroller, and the following VHDL components that provide inputs to and output data from the PicoBlaze.

1. An interrupt interface consisting of a 5-bit set-reset register. This register holds the status of the four push-buttons. If one of the buttons are pressed, the corresponding bit in the register should be set high. These values should stay high until PicoBlaze acknowledges the interrupt, at which point the register should be reset. The interrupt line is simply an OR function of all five bits in the interrupt register.

- A pseudorandom number generator (PRNG), should be based on Linear Congruential Generator (LCG) with an 8-bit output to the Picoblaze, as shown in the diagram below:

- An LCG generates a sequence of pseudo-random numbers according to the following recurrence congruence. The LCG will have an 8-bit state \( m = 2^8 \).

\[
R_{n+1} = a \times R_n + c \pmod{m}
\]

where \( R_n \) is the sequence of pseudorandom values, \( a \) is the multiplier, \( c \) is the increment and \( m \) is the modulus. \( R_0 \) is the initial seed value. Please assume the following default values of parameters \( a, c, \) and \( R_0 \): \( a = 0x11, c = 0x9D, R_0 = 0xD7 \). Additionally, assume that \( \times \) represents an unsigned multiplication.
The LCG generates one output per 1 clock cycle.

- A single-port 128x8 Data RAM that is used to store player scores.
- Seven segment display controller. This component should contain registers that store the seven segment values as they are written by the PicoBlaze.
- A hardware counter that keeps track of the amount of time that the player takes to finish the game. The counter should take values in the range of 0-99.99 seconds. If the player takes longer than 99.99 seconds, the timer value should not wrap, but should instead stay constant at 99.99 seconds.
- An address decoder that enables the correct read and write operations to the other components, based on the address output from PicoBlaze. As part of the design of the address decoder, you will need to create a memory map that defines the addresses for the I/O registers and RAM.

**Task 2 : Game Implementation in PicoBlaze**

Write assembly code for the PicoBlaze that enables users to play the game and then stores the scores in the Data RAM. The flow for the game is as follows:

- The player uses four switches to indicate a player ID ranging from 0 to 9 and presses button 0 to begin the game. At this point the timer is reset to zero.
- PicoBlaze samples the PRNG value and uses it to output 4 values ranging from 0-3 on each of the four seven segment displays. The player then enters the sequence
using buttons 0-3. Each time the player successfully enters the given sequence, a new sequence is generated and displayed. This process is repeated a total of 5 times.

- At the completion of the game, the time it took the player to finish the game is displayed on the seven segment display and is also stored in the next available location in the Data RAM. Note that each entry will require multiply bytes of storage. One possible way of storing the data is shown below. Note that P is the player ID.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>0x0P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 1</td>
<td>Time Value (Seconds) in BCD</td>
</tr>
<tr>
<td>Byte 2</td>
<td>Fractional Time in BCD</td>
</tr>
</tbody>
</table>

- At the end of each game, the best score for that player ID should be compared to the current score; if necessary, the best score should be updated with the new score.

- All ten players should be able to play an aggregate of at least 20 games. As an example, one player could play the game 20 times, two players could play 10 times each, or any other combination of players that adds up to 20.

**Task 3 : Browsing Game Data**

While waiting to start a new game, the user can hit button 3 to transition into browsing mode. In this mode, the user can press buttons 0 and 1 to scroll up and down through the scores in the Data RAM. The memory entries should be displayed on the seven segment display in the format described in Task 2. As an example, if player 2 had a time of 12.34 seconds, the displayed values should be:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12.34</td>
</tr>
</tbody>
</table>

The value “2” should be displayed first. Pressing the down button will display the time “12.34”. Pressing down again will display the player ID for the next score, and so on. The scrolling should wrap; that is, if only three games are played, pressing down after viewing
the score for the 3rd game should cause the player ID for the 1st game to appear again. Similarly, pressing up from the player ID for the 1st game should result in the score from the 3rd game being displayed.

Task 4: Sorting and Browsing Best Scores

When the user requests entry to browsing mode, PicoBlaze should sort the best scores for each of the ten players. These values were stored in the Data RAM in Task 2. Each score is assigned a rank, with the fastest time receiving first place, and the slowest time receiving 10th place. When in regular browsing mode, the user can change to best score browsing mode by pressing button 3. When in the best score browsing mode, the user should be able to return to regular browsing mode by hitting button 3. The display format for this mode is similar to that of Task 3, except each player’s rank is also displayed. So if Player 1’s best score was 23.45 Player 2’s best score was 34.56, and player 3’s best score was 12.34, the values would be displayed as

```
1 3
12.34

2 1
23.45

3 2
34.56
```

Players that did not play a game should not have a score displayed.

Task 5: Outputting Best Scores to PC (Bonus)

While the game is in either browsing mode, the PC can send a read request over EPP using adept. Upon receiving this request, PicoBlaze should transmit a report of the best scores to the PC in the same format outlined in Task 4. The data should be in ASCII format, so PicoBlaze will need to translate, for example, 1 to 49, 2 to 50, and so on before transmitting to the PC.
For each of the above Tasks:

Perform the following tasks to verify the correctness of your designs:

1. Define details of your timer, address decoder, and input and output interfaces. This includes the memory map for the address decoder.
2. Describe the system in VHDL.
3. Write and debug your assembly language program using the programming environment learnt in the class.
5. Synthesize your code and perform post-synthesis simulation.
6. Prepare the UCF (User Constraints File) specifying pin allocations.
7. Implement your code using an appropriate UCF file and chip specification.
8. Check thoroughly implementation reports. Pay attention to pin allocations.
10. Download the bitstream to the FPGA board.
11. Verify experimentally the correct operation of your circuit.

Include in the lab submission:

1. A detailed block diagram of the Datapath of your circuit.
2. Assembly language source code of the program run on PicoBlaze.
3. VHDL code for your circuit and all testbenches used to verify this circuit.
4. Your UCF file for FPGA.
5. Simulation waveforms from the functional, post-synthesis, and timing simulations, proving the correct operation of your circuit, and demonstrating the delay of its critical path.
6. A report listing all tasks completed and problems encountered.

Determine the following parameters of the entire circuit

- maximum clock frequency
- critical path
- resource utilization.
## Important Dates

<table>
<thead>
<tr>
<th></th>
<th>Monday Section</th>
<th>Wednesday Section</th>
<th>Thursday Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hands-on Sessions and Introduction to the Experiment</td>
<td>04/20/2014</td>
<td>04/15/2014</td>
<td>04/16/2014</td>
</tr>
<tr>
<td>Demonstration and Deliverables Due for Schedule A</td>
<td>05/04/2014</td>
<td>04/29/2014</td>
<td>04/30/2014</td>
</tr>
<tr>
<td>Demonstration and Deliverables Due for Schedule B</td>
<td>05/11/2014</td>
<td>05/06/2014</td>
<td>05/07/2014</td>
</tr>
</tbody>
</table>