In-Class Activity - PicoBlaze

In this lab, we will learn how the PicoBlaze microcontroller works and how to program it. This is representative of the usage of embedded processors within FPGAs and ASICs, which is something that every digital engineer needs to know.

1. Download the file "KCPSM3.zip" from the Website and extract it to a new directory "C:\KCPSM3." This file (and, now, the new directory) contains the PicoBlaze manuals (also called "KCPSM3"), the assembler of the PicoBlaze and Verilog code of PicoBlaze.

2. Download the file "ug129.pdf". This is the datasheet of PicoBlaze.

3. Download the file "pracPICO.psm" from the Website and put it in the directory "C:\KCPSM3\Assembler".

4. Download the template ZIP file from the Website. Open the Quartus project, the top-level Verilog file "Picoblaze_Practice.v", the file "picoblaze_template.v", and the file "pacoblaze_instruction_memory.v". We will study these files together in class. PicoBlaze is a Xilinx microprocessor. We will use a PicoBlaze clone called "PacoBlaze". The PacoBlaze files are already in a subdirectory contained in the template ZIP file.

Now you need to compile the software code (pracPICO.psm) and place it in the PicoBlaze Verilog project. This we will do in the following manner.

5. From the "Start" menu in Windows, run a "DOSBOX" terminal. If it is not installed, the installation file is on the website.

6. Mount the "c" drive via the command "mount c c:/", and go to the directory "C:\KCPSM3\Assembler"

7. Compile the PSM file by running:

```
KCPSM3 pracpico > compile.log
```

The compilation may take a few minutes. To check that everything compiled well, look at the file "compile.log" (you can open it in Notepad or Wordpad in Windows). Go to the end of the file to verify that the compilation was successful.

8. Now copy the file "PRACPICO.MEM" (which was generated by the assembler) to the Quartus project directory. If you look at the file "pacoblaze_instruction_memory.v", you can see that "PRACPICO.MEM" is read into the ROM of the PacoBlaze, and this is how the microprocessor is made to execute the program.

9. Compile the Quartus project and load the card.

10. What you should see is that (a) the LEDs are counting in binary every 1 second, and (b) on the LCD is a special message.

11. If one of the switches are set to "1", the message will run left in intervals of 1 second. If all switches are set to 0, the message will appear and disappear every second.

12. Now your goal is to change the "pracPICO.psm" to do the following things. Note that in order to see changes made in "pracPICO.psm", repeat steps 5-9. Make all the following changes incrementally, i.e. one by one (not necessarily in order, but one by one), making sure every change works before continuing. Designing in an incremental way is probably the only way through which you will be able to successfully complete this exercise.

a. Replace the message displayed on the LCD with an original (but decent!) message

b. Change the LCD shift intervals every 0.5 seconds.

c. Have the LCD go left if SW1 or SW2 are 1.

d. If SW[3] == 1, clear the LCD.

e. If SW[0] == 1 in the LCD display will not change

f. Using the PicoBlaze software, make the LEDs do what you did in Lab 1 but now with 0.5 seconds between each position, i.e., change the code "pracPICO.psm" so that the LEDs do the following:

```
0000 000X
0000 00X0
0000 0X00
0000 X000
000X 0000
00X0 0000
0X00 0000
X000 0000
0X00 0000
00X0 0000
000X 0000
0000 X000
0000 0X00
0000 00X0
0000 000X
```

(Look at how many ways we can solve this problem, as we did with Verilog, with PicoCtrl, and now with PicoBlaze ... As they say "there are many ways to skin a cat!").