# CS254 (Spring 2017): Lab Assignment 2

February 25-March 1, 2017

Please refer to the FPGALink (2012 version) user manual, a link to which has been provided on the course web page. The figure on Page 1 of the document provides a very clear picture of what you need to remember when trying to program a host application logic (a C program) to communicate with an FPGA application logic (a VHDL program).

You must have both FPGALink and ISE installed on your laptop (preferably under Linux) in order for this assignment to be completed. You will also have to add the path to the `xst` executable generated when you install ISE before you run the FPGALink commands to compile your VHDL design. To do this on Linux, execute the following from a terminal (bash shell):

```
export PATH=''Your_ISE_Install_Dir/14.7/ISE_DS/ISE/bin/lin64:$PATH''
```

Before proceeding with the instructions in this sheet, you must have successfully run `flcli` on your laptop, and **read as well as written** bytes from and to the Digilent Atlys board through the USB port. If you have not been able to do this so far, contact the instructor/TAs immediately. The remainder of this sheet won't make sense otherwise.

1. ***Programming the host application logic:***
   Use a text editor to open the file `main.c` in the directory
   `Your_FPGALink_Install_Dir/20140524/makestuff/apps/flcli`

   This is the source file for the `flcli` application. You are required to modify and recompile `main.c` to implement the following:

   - Add a line in `main.c` after line 620 that prints out the string
     `******* Modified for CS254 Lab projects *******`
   - Create a new command-line option "-y" that makes `flcli` behave as specified below:
     The application loops 128 times, and for iteration $i$ (starting from $i = 0$ to $i = 127$) of the loop, it reads a byte from channel $i$, prints it out on the screen, adds $i$ to it, and writes the resulting byte (modulo overflows) to channel $(i + 1) \pmod{1}28$, and waits for 1 second before starting the next iteration.
   - Rebuild `flcli` with the above changes. For this, you need to run `make deps` in the directory `makestuff/apps/flcli` after saving the above changes.
   - Now run the `flcli` executable just created from `Your_FPGALink_Distribution/rel/flcli`, where `Distribution` is `lin.x64`, `msvc.x64` or `msvc.x86`, as appropriate.

2. ***Programming the FPGA application logic:***
   Use a text editor to open the file `cksum_rtl.vhdl` in the directory
   `Your_FPGALink_Install_Dir/20140524/makestuff/hdlmake/apps/makestuff/swled/cksum/vhdl`

   This is the source file for the `checksum` application that has been programmed into your FPGA. You are required to modify and recompile `cksum_rtl.vhdl` to implement the following:

- Add a comment in `cksum_rtl.vhdl` after line 22 that says
  `Modified for CS254 Lab projects`
- Change the logic implemented in architecture `rtl` of `swled` so that the FPGA application logic behaves as specified below:

  The module has a register bank of 128 registers, each 8 bits wide. On start-up, all the registers in the bank must have the value 0. **You must not use a "for" or "while" loop to do this initialization**. Subsequently (say after 500 clock cycles from start – think of using these cycles to do the initialization), if the module receives a request to read from channel $i$ (where $i$ varies from 0 to 127), it reads the data from the $i^{th}$ register and sends it to the host. If, on the other hand, the module receives a request to write to channel $i$ , it simply writes the data received from the host to the $i^{th}$ register, overwriting whatever was already written there.

  The entity declaration for the `swled` module is available in the file
  `Your_FPGALink_Install_Dir/20140524/makestuff/hdlmake/apps/makestuff/swled/templates/harness.vhdl`
  Please go through the comments in this file to understand the purpose of the various ports of the `swled` module. You should use these ports judiciously for implementing the functionality you want.

  You may also want to inspect the top level VHDL module that is actually mapped down to your FPGA board in the file
  `Your_FPGALink_Install_Dir/20140524/makestuff/hdlmake/apps/makestuff/swled/templates/fx2all/vhdl/top_level.vhdl`
  Note that this instantiates the `swled` module and also another module called `comm_fpga_fx2`. The module `comm_fpga_fx2` is responsible for implementing the actual interface between the Digiltent Atlys board and the host computer using the on-board Cypress FX2LP USB interface micro-controller.

  While it is not necessary for you to peek into the details of `comm_fpga_fx2` for purposes of this lab, the "curious cats" may look at the VHDL source for this at
  `Your_FPGALink_Install_Dir/20140524/makestuff/hdlmake/libs/makestuff/comm-fpga/fx2/vhdl/comm_fpga_fx2.vhdl`
  Similarly, the constraints file (that defines which signal is mapped to which switch, LED, pin etc. on the actual Atlys board) for your design is available at
  `Your_FPGALink_Install_Dir/20140524/makestuff/hdlmake/apps/makestuff/swled/templates/fx2all/boards/atlys/board.ucf`

  **Please do not make changes to `comm_fpga_fx2.vhdl` and `board.ucf` for purposes of this assignment. We will need to do so for the next assignment.**
- Make sure your VHDL code compiles (you may use ISE for this purpose).
- Compile your modified VHDL file by running

  ```
  Your_FPGALink_Install_Dir/20140524/makestuff/hdlmake/bin/hdlmake.py
  -t Your_FPGALink_Install_Dir/20140524/makestuff/hdlmake/apps/makestuff/swled/templates/fx2all/vhdl
  -b atlys -p fpga
  ```
- Reprogram your FPGA with the newly generated FPGA programming file by running
  `Your_FPGALink_Install_Dir/20140524/makestuff/apps/flcli/lin.x64/rel/flcli -v 1d50:602b:0002 -p J:D0D2D3D4:fpga.xsvf`

3. ***Running the modified host application logic and FPGA application logic:*** Run the new host application logic and the new FPGA application logic together and show your results.