Synopsis - Week 2: Binary Numbers

Prof. Jordan C. Hanson

January 30, 2024

1 Introductions to Connect and Disconnect to PYNQ-Z1 System-on-a-Chip (SoC)

1. **Connect** the PYNQ-Z1 board to the laptop via the USB to Ethernet converter. Next, connect the micro-USB cable between the PYNQ-Z1 and the laptop. Open a browser and navigate to http://192.168.2.99:9090.
2. You should be prompted to enter a password. Type `xilinx`. You are now inside the chip at the center of the board, running a version of linux on the dual-core ARM. Navigate to the Getting Started folder by clicking, and run the tutorial entitled `1_jupyter_notebook.ipynb`. Python notebooks can run code and contain writing in markup.
3. **Disconnect:** Click the "Running" tab when you are done with the notebook, and close the jupyter notebook. Click "New" in the upper right hand corner to open a terminal. In the terminal, type `shutdown now`. Close the browser tabs and power down the PYNQ-Z1 board.
4. **Kernel reset:** Whenever you make a mistake or want to change something, you should click the kernel reset button at the top of the screen. This will reset all variables and make the SoC a blank slate again. Kernel errors usually occur when the programmable logic is not programmed but the software asks for it.

2 Boolean Generator on PYNQ-Z1

1. Navigate to the `logictools` directory.
2. Read the instructions for how to create a *boolean generator*. The boolean generator links digital inputs and outputs based on operations like NOT, OR, AND, and XOR. Based on how the LEDs blink when you press the push buttons, what is the truth table of XOR?

3. Create a python dictionary entitled *function* and use the boolean generator to perform multiple logic operations:

```
from pynq.overlays.logictools import LogicToolsOverlay
logictools_olay = LogicToolsOverlay('logictools.bit')
function = {'f1': 'LD0 = PB0 ^ PB1', 'f2': 'LD1 = PB2 ^ PB3'}
boolean_generator = logictools_olay.boolean_generator
boolean_generator.setup(function)
boolean_generator.run()
```

   Write the truth table for each separate logic function above.

4. Create a dictionary of logic functions f_i that (a) add the input of up to three pressed buttons as if each is worth 1 point. (b) Displays the 2's compliment of the pressed buttons as if the buttons represent a bitstream. Write example code for the functions here:

5. Can you construct and verify an example that involves the associative property? That is, a binary expression of the following sort:

   ```
   f1 = '(PB0 ^ PB1) & PB2'
   ```

   What *should* the output be? Does the LED output reflect this?