Synopsis - Week 3: Basic Logic Gates and Finite State Machines (FSMs)

Prof. Jordan C. Hanson

February 25, 2020

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

1. Connect to the PYNQ-Z1 board in the usual fashion, ensuring that the system is logged out before starting a new kernel.
2. If troubleshooting is needed, consult lab partners or the professor.

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. If this looks familiar, create a blank Jupyer notebook and begin the next step. If it doesn't look familiar, take a few minutes to run the steps in boolean generator. Verify the correct behavior of the LEDs.
3. In a new Jupyter notebook, 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)'}
   boolean_generator = logictools_olay.boolean_generator
   boolean_generator.setup(function)
   boolean_generator.run()
   ```

   In a separate cell, you can add the stop function for the boolean generator:

   ```
   boolean_generator.stop()
   ```

   Write the truth table the logic function above. What gate does it represent? Draw the appropriate gate diagram.

4. Reset the kernel, and change the function to

   ```
   function = {'f1': 'LD0 = !(PB0 & PB1)'}
   ```

   Write the truth table the logic function above. What gate does it represent? Draw the appropriate gate diagram.

5. Finally, draw the relevant gate diagram for the function below. Run the code and record the truth table. Can you identify this as a basic logic function? Why or why not?

   ```
   function = {'f1': 'LD0 = !(PB0 & PB1) & !(PB2 & PB3)'}
   ```

3 Finite State Machines (FSMs)

1. Close the previous boolean generator script. Navigate to the finite state machine Jupyter notebook in the logictools directory.
2. Load the finite state machine generator according to the steps, and examine the diagram explaining the relationship between the eight states of the FSM.
3. Notice that each state *transition* goes either from state N to $N + 1$ or from N to $N - 1$. Thus, each state either goes forward or backward
4. Make sure you have two jumper wires to connect the right pins according to the image of the pynq-z1 board.
5. Once you execute the FSM generator according to the plan in the diagram, the *trace analyzer* will give you the timing diagram of the system.
6. **Verify the that the timing diagram is counting in gray code.** Read off the binary values from the timing diagram. If you are not sure how this works, make sure to ask for assistance.