# Lab 1 Pre-lab Notes

*Created by Andrew Douglass*

**Purpose**

- The focus of this week's lab is to become familiar with Viviado (which is a software tool used for synthesis and analysis of HDL designs by Xilinx) and Verilog.
- Verilog is one of the two most popular HDLs used (VHDL being the other). An HDL is a programming language used to describe hardware.
- Pitfalls and Fallacies to avoid when programming in an Verilog (or any HDL):
    - reg vs. wire
        - If you plan on assigning your output signal in sequential logic then you should use a reg. In other words, do not make an assignment to a wire in an always block.
        - https://inst.eecs.berkeley.edu/~cs150/Documents/Nets.pdf
    - Do not mix posedge with non-posedge triggered signals in your always block parameter list.

        -
          ```
          always@(posedge CLK or RESET)  // DO NOT DO THIS
          ```
    - Do not make assignments to a signal in multiple always blocks.

        -
          ```
          always@(posedge CLK)
               dividedClk <= ‘0’;
          always@(INPUT or TEST)       // DO NOT DO THIS
               dividedClk <= ‘1’;
          ```
    - Make sure you remember that always blocks are independent and run in parallel.
    - Do not have any time delays in synthesizable code.
    - Remember to use non-blocking assignments in always blocks (<=). Especially do not use non-blocking and blocking statements in the same always block.
    - Do not use initial blocks in your synthesizable Verilog code.
    - All outputs in an always block must always be assigned, no matter what conditions occur! If a signal does not get assigned during an always block execution a latch is inferred. This is usually (pretty much always) not what you want to happen.

**Lab Steps and Tips**

- Three important points before we get started:
    - Do not copy and paste the code from the pdf. Instead use the files from “lab1_files.zip”
    - Create a new project for the counter and another new project for the jackpot.
    - Make sure the JTAG jumper is properly set on the ZYBO board or you will not be able to program the FPGA.
- The lab this week consists of three main parts.
    - The first is a basic switches Verilog design which will utilizes the switches on the ZYBO board to turn on and off the LEDs.

    - This part of the lab should be fairly straight-forward and following the instructions should be all you need to do.
  - The second is a 4-bit counter using the LEDs that will count upwards when you press Button 0 on the ZYBO board and will count downwards when you press button 1 on the ZYBO board. You will show the count on the LEDs on the boards. Here are a few useful tips:
    - Keep a counter that will increment to a specified value in order to divide the clock. Only need to strobe the divided clock.
    - Create a separate module for the clock divider circuit, it can be used for the jackpot if you do things correctly.

      ```
      module divide_clock(
           input Clk,
           input Reset,
           output reg OutClk );
      ```
  - Finally, you will create a jackpot game which will involve flipping the switches on the board to as the lights turn on/off.
    - The jackpot game can use the divided clock circuit from above.
    - All the lights should turn on when the switch is turned on that corresponds to that switches LED.
    - Reset can be implemented how you want, recommend using one of the push buttons.