In this lab you will learn about:

1. *Hardware Description Language*
2. *Verilog HDL*
3. *Field Programmable Gate Arrays*
4. *Development suits for FPGAs – Altera Quartus II*

In this lab you will work on:

1. *Getting your hands on EDA Playground*
2. *Gate Level Modeling*
3. *Translating your Lab 5 Circuit to Verilog*

Resources for the Lab:

1. *Getting your hands on EDA Playground*
2. *Tutorial: "http://www.asic-world.com/verilog/veritut.html"*

## Hardware Description Language

In computer engineering, a hardware description language (HDL) is a specialized computer language used to describe the structure and behavior of electronic circuits, and most commonly, digital logic circuits. A hardware description language enables a precise, formal description of an electronic circuit that allows for the automated analysis and simulation of an electronic circuit. It also allows for the synthesis of a HDL description into a netlist (a specification of physical electronic components and how they are connected), which can then be placed and routed to produce the set of masks used to create an integrated circuit.

A hardware description language looks much like a programming language such as C; it is a textual description consisting of expressions, statements and control structures. One important difference between most programming languages and HDLs is that HDLs explicitly include the notion of time.

Throughout the history, there have been multiple HDLs. Few honorable mentions are:

1. ISPS
2. VHDL
3. Verilog HDL
4. System Verilog HDL

As part of our course work, we will be focusing on the Verilog HDL. A brief introduction to the Verilog language is given in the video lecture.

## Verilog HDL:

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits, as well as in the design of genetic circuits.

A Verilog design consists of a hierarchy of modules. Modules encapsulate design hierarchy and communicate with other modules through a set of declared input, output, and bidirectional ports.

Internally, a module can contain any combination of the following: net/variable declarations (wire, reg, integer, etc.), concurrent and sequential statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed inside a begin/end block and executed in sequential order within the block. However, the blocks themselves are executed concurrently, making Verilog a dataflow language.

## Activities

### Activity 1: Getting Hands on EDA Playground

Follow the step by step procedure in the EDA Playground set up video, which you can find in this week's module on canvas.

### Activity 2: Create your Lab 5 Circuit

Now that you know how to add a project in EDA Playground and test it, your assignment is to create the circuit you found in Lab 5 using gate level modeling. We have fully written the testbench so you do not need to make any changes to that, but you may add extra test cases if you'd like.

Here is the link to the EDA Playground project for this week's lab:
https://www.edaplayground.com/x/QcSP

Please copy this to your account and rename it "Lab_6_<psu username>"

To submit, make sure your project is saved and it is set to public view (default) then copy and paste your link into the submission text box for Lab 6.

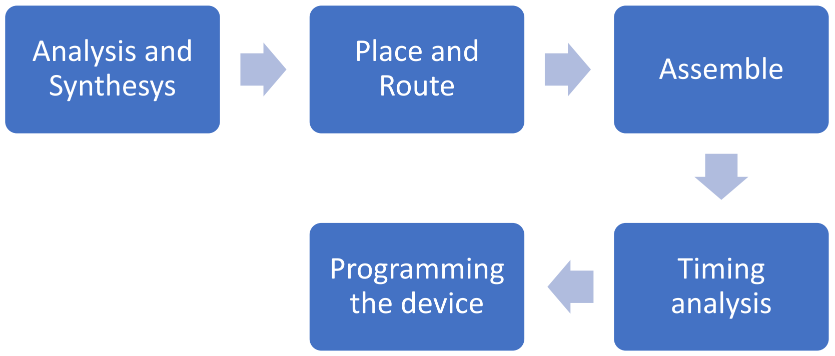
***We won't be using the FPGA boards since we are not in the physical lab but you can learn more about them below***

## Field Programmable Gate Array

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), like that used for an application-specific integrated circuit (ASIC). Circuit diagrams were previously used to specify the configuration, but this is increasingly rare due to the advent of electronic design automation tools. FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements. Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software.

## Development Suits for FPGAs

The FPGA development follows multiple steps (depicted in the figure below). Instead of doing these step individually, we tend to use development suits to for the full flow. The TAs will tell you a bit about each of these parts (ask your TAs). As we will be using the Altera FPGAs, we need the development suit from Altera which is called Quartus II. Altera Quartus II is programmable logic device design software produced by Altera, before Altera was acquired by Intel and the tool was renamed to Intel Quartus Prime. Quartus II enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Quartus includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation.


Analysis and Synthesys
Place and Route
Assemble
Timing analysis
Programming the device