**FPGA Introduction Lab**

**A. La Rosa, B. Comnes**

**Physics 315**

**Portland State University**

# Abstract

In this lab we will be briefly exploring Field Programmable Gate Arrays (FPGA). This will be achieved by a brief background discussion of FPGAs, followed up by a tutorial illustrating a very basic workflow, including FPGA design using the schematic layout as well as an example of using VHDL to implement very basic digital circuits. The student is run through how to set up the design, how to write a test, and how to compile and upload the final design to the FPGA. The lab was designed around the Digilent Nexys 3 prototyping board, which provides relatively access to a Xilinx Spartan6 XC6LX16-CS324 FPGA chip along with a programmer, program memory, and array of switches, buttons, LEDs and I/O ports.

# FPGA Introduction

## What is an FPGA?

FPGAs are used as tools in modern digital circuit design. FPGAs provide a large array of generic logic gates inside of a single integrated circuit (IC) that can be configured arbitrarily, so long as it does not exceed the capabilities of your FPGA chip.

The FPGA is configured using a Hardware Description Language (HDL), making it possible for FPGA designs to be portable between hardware, so long as the correct constraints are used to map the design to the specific details of the FPGA that is being used. There are also other methods to configure the FPGA such as schematic layout interpreters, as well as proprietary solutions, such as LabVIEW.

Not only can FPGAs be configured into any arbitrary digital circuit, they can also be reconfigured at a later time, if for example, you need to fix a problem in the circuit after you have integrated it into a project. It can even reconfigure itself to act as a different circuit depending on external conditions.

## How is it different than other embedded systems

There is a large array of programmable logic (PL) technologies available, and at this point it might even seem like FPGAs are a glorified microcontroller. This is not so. When you configure an FPGA, the logic gates inside the IC are configured correctly such that you are actually, physically creating the circuit that you have specified, all inside the FPGA.

Also, unlike other PL, the FPGA can be rewritten as many times as needed. The flash memory, which stores the program to configure the FPGA on power up, will be the limiting factor, with a re-write limit of about 100,000.

Many FPGAs have an array of IP (Intellectual Property) modules which are premade FPGA configurations that can perform a complicated task. For example there are IP modules to implement a soft CPU in your FPGA so that it can be used as a more general purpose computer. These modules typically cost money and include some form of DRM or obfuscation to keep you from seeing how they work. The whole purpose of not just simply sharing the HDL code is to prevent others from seeing how the developer implemented a particular design for whatever reason.

## Advantages

When implementing complex digital circuits, FPGAs provide numerous advantages. The design can be written, tested and simulated on the computer. The designs can be done in a way so that they are portable to other FPGA devices, for repeatable and rapid deployment in many devices. Multiple people can work on the same HDL files and increase the speed of circuit development.

Once the design is ready for implementation, the entire production process is uploading the design to the board, as the FPGA will configure the circuit for you, rather than having to build it with a machine or by hand. If there is ever an issue with the circuit that is discovered after deployment, it is only a matter up updating the HDL code and re-uploading it to the FPGA. The FPGA could even check online for schematic updates potentially.

Running the circuit in a single chip also allows for high speed circuitry that would simply be unattainable on a breadboard or other circuit design methods. When used correctly, FPGAs can be extremely fast, which is typically why they are used over other options.

## Drawbacks

FPGAs have been historically quite expensive. Additionally, they can be fairly difficult to work with, requiring a wide range of knowledge of HDLs and digital circuit design to use effectively.

The software used to work with many FPGAs can be quite expensive, and as well as bulky in size. The IDE we are using is more than 8Gb in download size, and as you will learn, has a user interface that will take some time getting used too.

# Working with the FPGA

## Workflow overview

For working with our Nexys 3 FPGA, we will be using the **ISE Project Navigator**. This integrated development environment (IDE) will allow us to set up a project and create different design, test and constraint files, as well as act as a debugger and handle the compiling of our designs.

Once we have design, tested, and compiled our FPGA configuration, we will need to upload it to the board. For this, we must use a tool called **Adept** which is made by Digilent. This program is used to upload .bit files to many different Digilent FPGA boards.

## Development options

There are a number of different options for configuring an FPGA, such as HDLs, schematics or proprietary solutions. We will focus on schematics and an HDL called **VHDL**.

## Hardware Description Languages

FPGA designs are usually made with an HDL such as **VHDL** or **Verilog**. It is easier to maintain code rather than a 2-D schematic, which is why it HDLs are used more often.

These languages are similar to a programming language, but are rather geared towards digital circuit design, so they have the same limitations has digital circuits. For this lab, these differences will not be of focus. Understanding these differences would be important if you were actually to use an FPGA in a more complex project.

# Configuring the FPGA using schematics

## In this section we will implement a simple AND gate, add constraints to hook op two onboard switches and an LED to the AND gate, test and deploy our design to the actual FPGA.

**Find a computer.**

You must use one of the 201 lab computers for this lab, as they are the only ones with the licensed software already installed.

**Open ISE Project Explorer**

There should be a desktop icon, otherwise search through the Start menu to find it, or ask for help if you cant find it.

**Go to File-> New Project**

Give your project new name. Don’t use any spaces or special characters. Use an underscore (\_) if you need to put a space in your file name. Save it to your thumb drive or user directory. The ISE Project explorer will make a royal mess inside this file, so do save it into a folder with files that already exist.

**Also, Under Top-Level Source, Choose HDL**

**Click Next.**

**Under Evaluation Development Board, choose Spartan 6 SP601 Evaluation Platform.**

This will set the category, family, speed and package options once you select our board.

**Choose VHDL under the Preferred Language setting**

**Click next. A project summary window will show. Check for any oddities and click finish to close the window.**

This closes the new project wizard, and drops you into your ISE project navigator IDE. It is a very non-intuitive interface, that you will only become accustom to with experience.

**Next to the File menu, Select Project -> New Source from the drop down menu.**

This will open the new source wizard.

**Within the New Source Wizard, Select Schematic, and give it a name.**

Again, do not include special characters or spaces in the name. The ISE is a whole hodgepodge of tools with known file name bugs if you go to far out of the naming scope.

**Continue through the wizard and check for oddities in the summary window.**

Once the new source wizard closes, you your schematic will automatically open. Notice that the vertical toolbar next to the schematic window has now changed. This vertical toolbar is where you find your context specific tool buttons. To the right of this vertical toolbar is where the file you are working on is displayed. To the left of the toolbar, is your project navigator, plus your toolbox for some of the tools on the toolbar. The UI is really random, so just learn to deal with it. There is a whole slew of tabs, on this left hand side, but the **Design** tab is the most useful. It acts as your project navigator and task launcher for compiling and testing your code and schematics. The **Design** tab has two modes, selected by the radio buttons at the top: a Simulation mode and Implementation mode. We will be switching between the two so get used to it.

# Configuring the FPGA using VHDL