# Chip 8
# A Hardware Implementation
# ECE260

Ben Nollan

February 16, 2017

**Abstract**
This paper is about defining, in hardware, a Chip 8. The Chip 8 was designed to be a simulated processor that accelerated the development time of making games on different platforms. My goal is to implement a hardware design on an FPGA.

# 1 Introduction

The Chip 8 was first created as a virtual machine for the Telmac 1800 in the mid-1970's. Since then it has been ported to hundreds of platforms due to it's simple nature. People have had luck making this CPU using an FPGA, one such example is an implementation using a Xilinx FPGA programmed with the Lava hardware description language[1]. He used a different FPGA and a different hardware description language than I am going to use. The people who would want to use this project would be developers that would like to program games in assembly. This platform is good for beginners because it is relatively easy to get thing to draw to the screen and there are a lot of example games that are in the public domain.

# 2 Specification

The Chip 8 has certain specifications that need to be adhered to while designing it. For input, the Chip 8 takes 16 inputs, usually given from a hexadecimal keypad, my plan is to use the switches and buttons included on the development board. For output, the Chip 8 writes to a 64x32 screen space with support for 2 colors per pixel (one bit). This graphics array will be scaled up and outputted with a VGA signal operating at 1920X1080. The Chip 8 uses a timer operating at 60Hz to keep track of time while running a game. This timer will be run off the vertical sync because that is also 60Hz. Internally, the Chip 8 has 35 op codes stored in 2 byte big-endian format[2]. The CPU has standard op codes that do ALU operations and branching, it also has more specialized ones for reading input from a controller and for writing graphics to the screen. These will need to be assembled as the memory is only 8 bits wide. There is 4K of memory for the processor, for other forms of memory it has 16 internal registers and a 12 level stack.

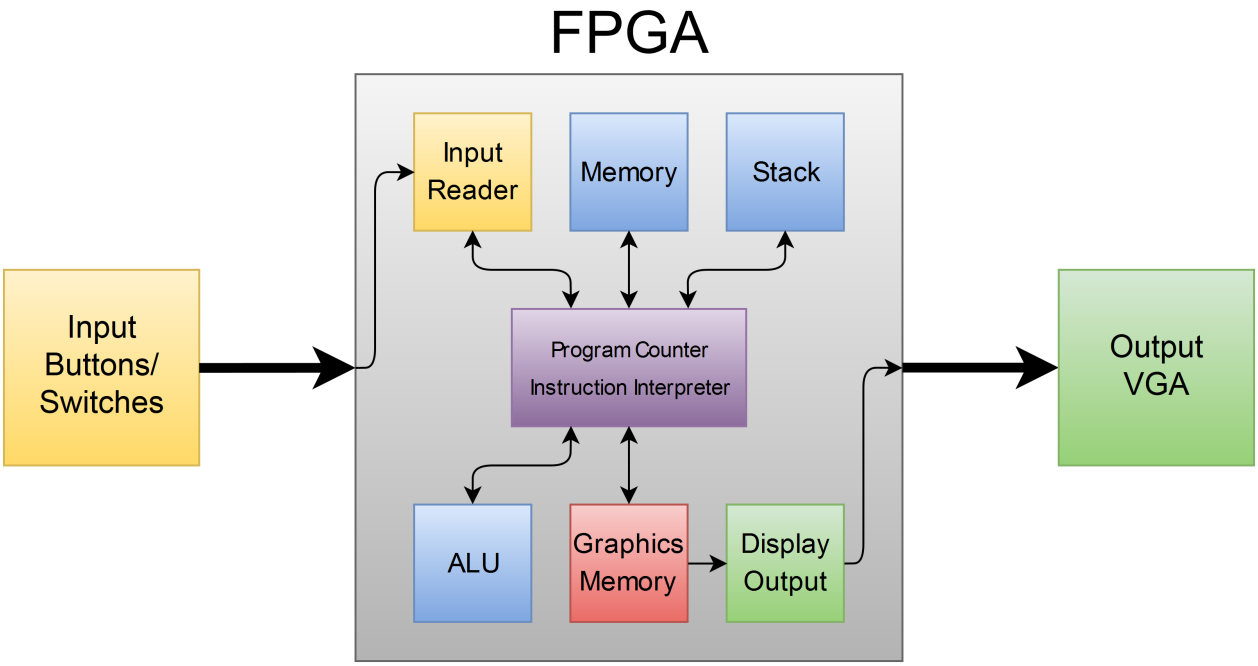
# 3 Design

I am going to use the Digilent Nexys 4 FPGA Development board to implement the Chip 8. This platform utilizes the Xilinx Artix-7 FPGA that has plenty of room for this project. This FPGA will be programmed in Verilog Hardware description language.

# 4 Design Verification

To test my design I plan on implementing it in steps, verifying each piece of the design to reduce the time spent on finding which module is the one at fault. I plan on using the simulation tool built into the Vivado suite to handle initial testing and use the logic analyzer IP block for testing or troubleshooting any faults when the design is built on the hardware.

Figure 1: Block Diagram of the project.




# 5 Schedule

Figure 2: Planned schedule for the project.

| | Week 1 | Week 2 | Week 3 | Week 4 | Week 5 |
|---|---|---|---|---|---|
| VGA output | ----------> | | | | |
| Timer | ----------> | | | | |
| ALU | ---------- | ----------> | | | |
| Memory | | ---------- | ----------> | | |
| Stack | | ---------- | ----------> | | |
| Graphics Output | | | ---------- | ----------> | |
| Button Input | | | ---------- | ----------> | |
| Testing and Verification | ---------- | ---------- | ---------- | ---------- | ----------> |

# 6 References


1. Gergo Erdi, ”My First Computer”, March 2014. https://gergo.erdi.hu/blog/2014-03-29-my_first_computer/

2. Joseph Weisbecker, ”An Easy Programming System,” *BYTE magazine*, December 1978, pp. 108122. https://archive.org/details/byte-magazine-1978-12