FPGA GAME BOY EMULATOR

Overview

We are interested in building an emulator for Game Boy games in Verilog to run and play the games on an FPGA. The game ROM(s) will be stored on a computer, and data will be sent from the ROM file to the FPGA via USB Serial communication. If building a Game Boy processor for arbitrary games is too difficult, we will synthesize hardware that will handle a specific game. The computer monitor in the lab will be used to display the graphics. The Classic USB NES controller will be used for user game input.
We will be working on implementing the original version of the Game Boy that was released in 1989.

First Steps

- The gameboy ROMs come in the gameboy file format (.gb) that we need to research and understand. We will then need to create an interpreter module for an arbitrary bit stream of game data. A simple way to do this is using a Python program using the pyserial library to parse the game data into a format that we know and understand, then forwarding that information to the FPGA via USB interface. One way to understand how to read the .gb file is to read through and understand existing Game Boy software emulators, some of which are open source. An example of that is the gnuboy emulator (http://freecode.com/projects/gnuboy and https://sourceforge.net/p/gnuboy/wiki/WikiHome/). We have also been provided with a copy of the final project report from a 2009 group that synthesized Game Boy hardware on the labkit. This document will also be used for reference as to how the ROM interacts with the hardware.

- We plan on writing or using verilog code to build Game Boy hardware (Zilog Z80, which is an 8-bit based microprocessor) and emulate its functionality on the Nexys FPGA board. There exists open-source resources online that implement an emulator (TV80) version of the processor, running in verilog, written by Guy Hutchinson. (codebase: https://github.com/lipro/tv80). We plan on testing the module and understanding how it works before we start using it. We could write a testbench or attempt reading memory contents and displaying them on the FPGA hex display to see if they seem reasonable and if we are using and understanding the module correctly.

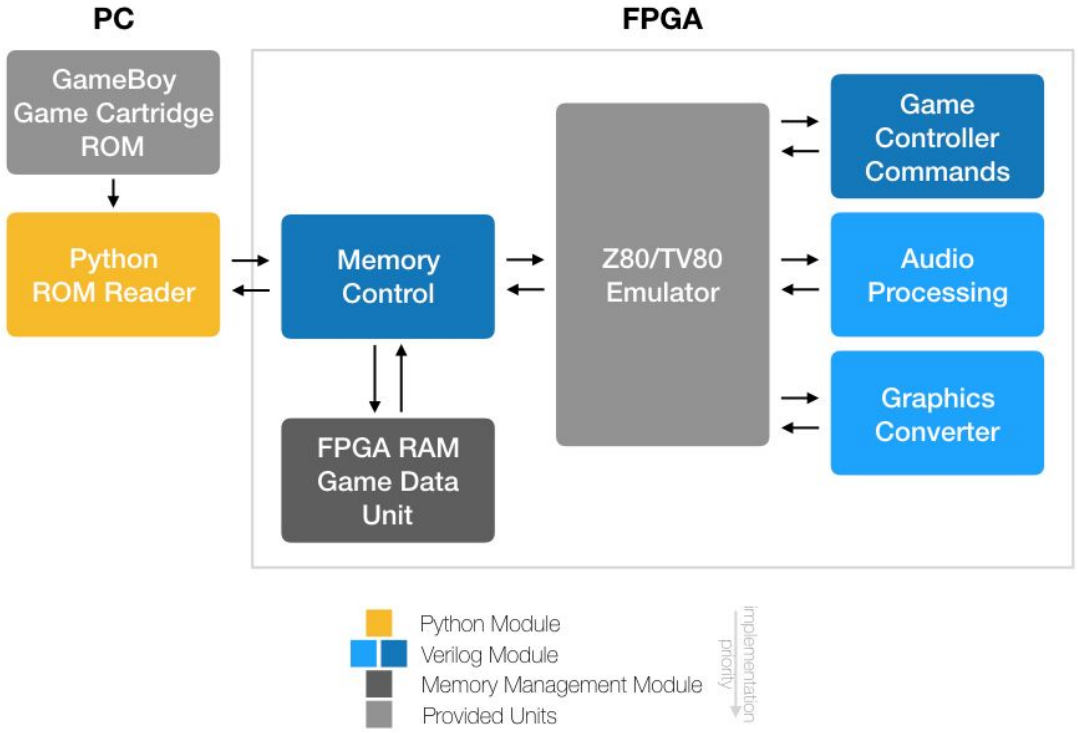
- Finalize our idea of memory data communications and requests handling between the modules (which can request, how to make requests, who handles the requests and how).

- Since the games run on a clock frequency maximum of 8 MHz, we will need to synthesize a module to generate a slower clock for our emulator.

Required Modules

- **Clock Module**
  4 - 8 MHz system clock
- **Python ROM Reader**
  our virtual game cartridge
- **Verilog Memory Control**
  intermediary between Z80 (TV80) and rest of hardware
- **Verilog Game Boy to VGA Graphics**
  convert and scale Game Boy graphics to be displayed on screen
- **Verilog Game Controller Commands**
  handle user input into the game from the NES Controller
- **Verilog Audio Processor**
  process and play the sounds and music of the game
- **Verilog Z80 Emulator Processing Unit**
  microcontroller, will handle all calculations
  TV80 is a open-source Verilog implementation of the Z80
- **Verilog Game Data RAM Unit**
  BRAM for storing Save Game data

Block Diagram



Milestones

Rough weekly goals by the end of each week:

10/29 - 11/4	Collect all background information: - Understand .gb files and how to read from them - Decide on the spec for the python ROM reader unit - Decide on the tasks and spec of the Memory Control Unit
11/5 - 11/11	- Have the python program working - Start testing of TV80 module - Clock Module - Start/make good progress on the Verilog modules Memory Control and Game Controller Commands
11/12 - 11/18	- Implementation of memory control and game controller done - Work on/finish implementation of Graphics Converter
11/19 - 11/25	- Integrated testing and Debugging - Ideally get one game running - Thanksgiving
11/26 - 12/2	- Work on Audio Processing Unit - Debugging
12/3 - 12/9	- Debugging
12/9 - 12/15	- Presentation Week

The Distant Dream

If we are able to successfully run our implementation of the Game Boy and hit all of our Milestones, then we will attempt to implement a Game Boy Color that is backwards compatible with older Game Boy games. This may involve the addition of an extra module that would allow this compatibility.

- Team KWala