Project Proposal: Chip-8 Emulator

Jennifer Bi (jb3495)
Nelson Gomez (ng2573)
Kundan Guha (kg2632)
Justin Wong (jw3554)

Background

Chip-8 is an interpreted programming language designed for applications to be run on 8-bit computers. Chip-8 can also refer to an interpreter or virtual machine which interprets Chip-8 code and emulates instructions. There are a number of classic video games ported to CHIP-8, such as Pong, Space Invaders, Tetris, and Pac-Man.

Project Description

We plan to implement a Chip-8 emulator, on which Chip-8 games can be run. Our goal is to successfully implement the emulator to be able to run games on it, and to learn about SystemVerilog and FPGA implementation in the process. If time allows, we could extend our implementation with SCHIP support. (This would require implementing additional opcodes but also allow high-resolution graphics and a wider variety of applications).

Design/Implementation

We will implement registers, stack, timers, sound, graphics (e.g. framebuffer) in hardware. We use Cowgod's technical reference: `http://devernay.free.fr/hacks/chip8/C8TECH10.HTM`.

Memory: Chip-8 was designed to be implemented on systems with 4K RAM. On these systems, Chip-8 is stored in the first 512 bytes of memory, and can address 4K locations.

Registers: Chip-8 has 16 8-bit data registers (named `V0-VF`) and a 16-bit address register (named `I`). The `VF` register is also set as a flag for certain instructions.

Keyboard, Graphics, and Display: The computers on which Chip-8 programs were run had a 16-key hexadecimal keypad. They used a 64x32-pixel display. Sprites in Chip-8 graphics could be up to 8x15 pixels, but 8x5-pixel sprites could be stored in the Chip-8 interpreter memory area and referenced.

Timers: Chip-8 has two 60Hz timers, the delay timer (using delay timer register `DT`) and sound timer (sound timer register `ST`). A single-tone sound (configured by the interpreter) is emitted whenever the sound timer register (ST) is non-zero.

Instructions: Chip-8 has 35 opcodes, all of which are two bytes in big-endian format. The instructions implement control flow, arithmetic operations, key/graphics control.

Timeline

Mar 15-22: Read through technical specification, collect any necessary resources and tools
Mar 29 (Design): Set up/decide how to organize .sv files and any C files
Apr 5 (Milestone 1): Have working ALU instructions, control flow, and graphics/framebuffer instructions
Apr 19 (Milestone 2): Have working timers and sound
May 3 (Milestone 3): Tetris?
May 14: Final presentation