Chip-8 Emulation on a SoCKit FPGA

Team: Ashley Kling, Levi Oliver, Gabrielle Taylor, David Watkins
Supervisor: Prof. Stephen Edwards

1

Chip-8 Emulation Overview

Not your garden variety
interpreted programming
language

Opcodes and Instructions

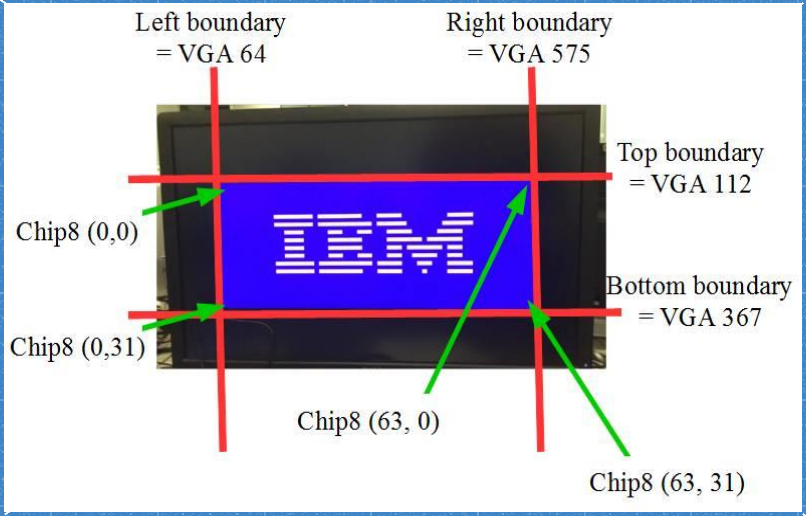
- Chip-8 has a total of 35 instructions

00E0 - CLS	00EE - RET	0nnn - SYS addr	1nnn - JP addr	2nnn - CALL addr	3xkk - SE Vx, byte	4xkk - SNE Vx, byte
5xy0 - SE Vx, Vy	6xkk - LD Vx, byte	7xkk - ADD Vx, byte	8xy0 - LD Vx, Vy	8xy1 - OR Vx, Vy	8xy2 - AND Vx, Vy	8xy3 - XOR Vx, Vy
8xy4 - ADD Vx, Vy	8xy5 - SUB Vx, Vy	8xy6 - SHR Vx {, Vy}	8xy7 - SUBN Vx, Vy	8xyE - SHL Vx {, Vy}	9xy0 - SNE Vx, Vy	Annn - LD I, addr
Bnnn - JP V0, addr	Cxkk - RND Vx, byte	Dxyn - DRW Vx, Vy, nibble	Ex9E - SKP Vx	ExA1 - SKNP Vx	Fx07 - LD Vx, DT	Fx0A - LD Vx, K
Fx15 - LD DT, Vx	Fx18 - LD ST, Vx	Fx1E - ADD I, Vx	Fx29 - LD F, Vx	Fx33 - LD B, Vx	Fx55 - LD [I], Vx	Fx65 - LD Vx, [I]

- Unsupported
- Supported
- Cycle Intensive

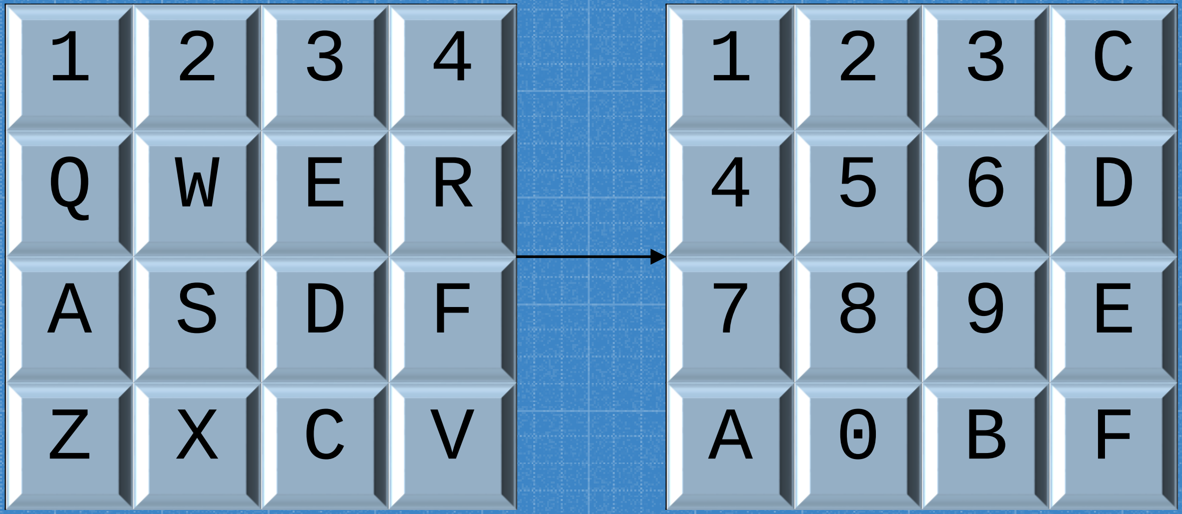
Chip-8 Hardware Specifications

- 64x32 bit display
- 64B stack
- 4KB memory
- 16-key input
- 16B register file



Keyboard Layout

- O – Reset
- P – Pause
- Enter – Start
- Keyboard Mapping



1	2	3	4
Q	W	E	R
A	S	D	F
Z	X	C	V

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

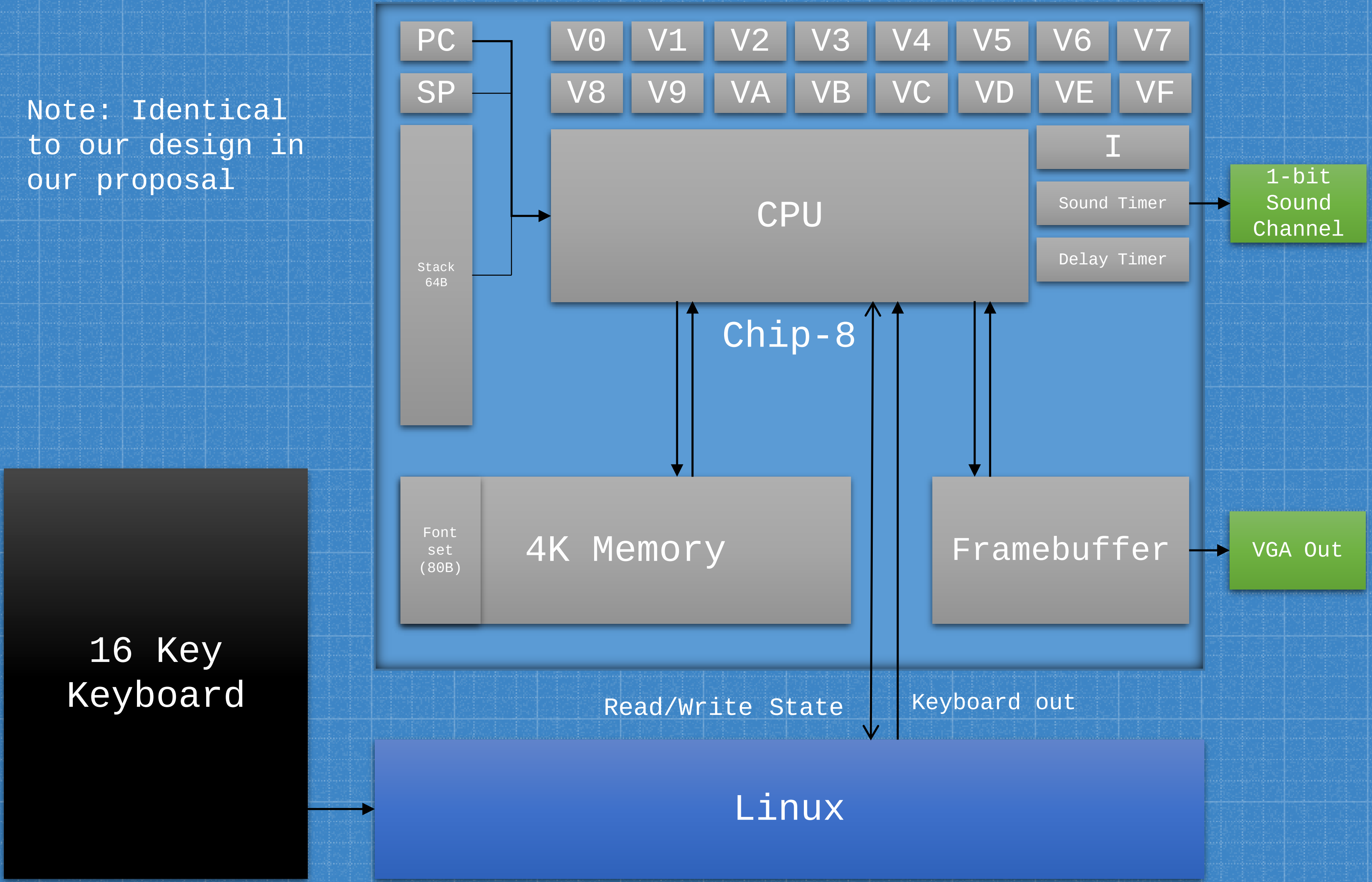
2

Emulator Layout

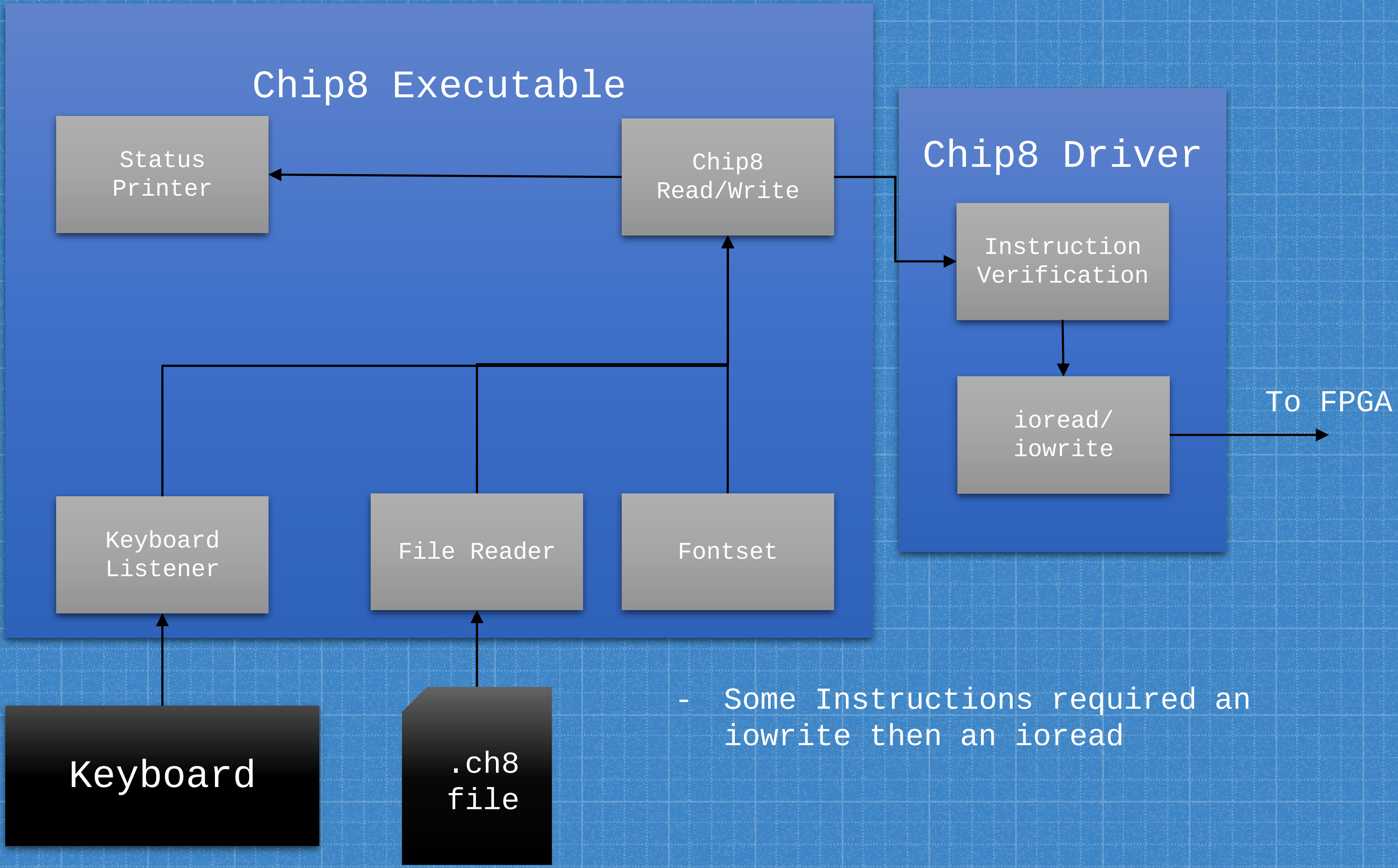
About as nice looking as this powerpoint

Linux to SoCKit Bridge

Note: Identical to our design in our proposal

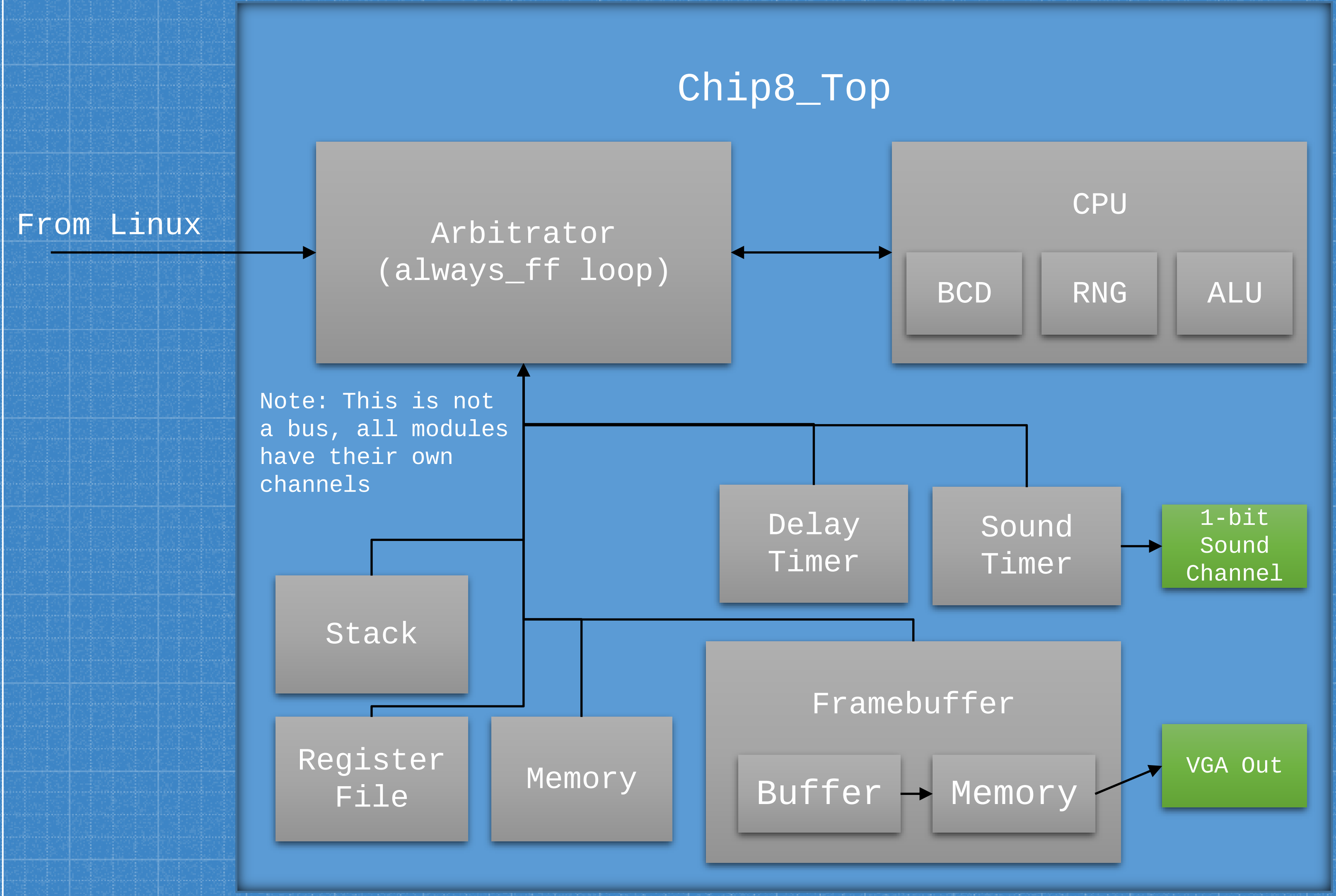


Linux Layout



- Some Instructions required an iowrite then an ioread

Hardware Layout

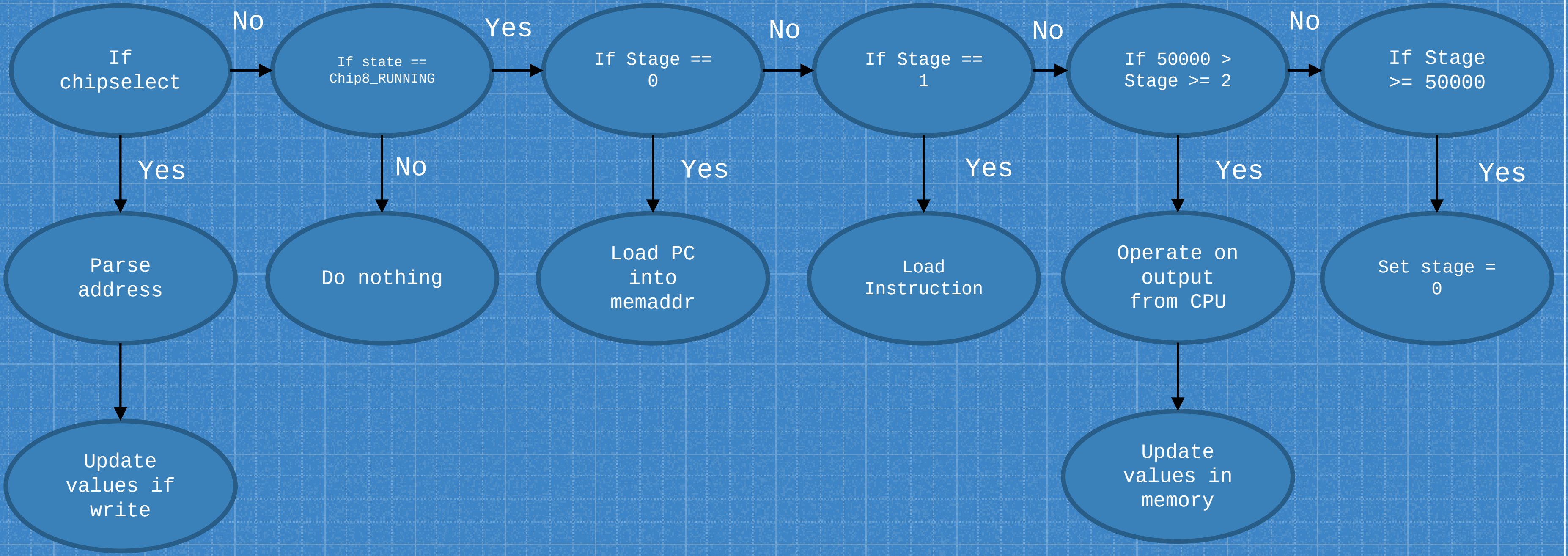


3

Module Design

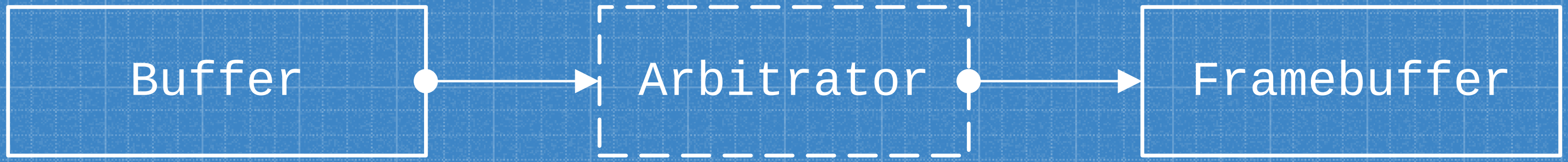
Insert something snarky here

Chip8-Top as Master Control Unit



Note: Stage is incremented on each clock cycle while state == Chip8_Running and the device is not waiting for keyboard input

Framebuffer Double Buffer



- The Framebuffer manages two 64x32 bit memories in an effort to reduce flicker
- The arbitrator will copy the buffer over to the framebuffer only when it has been 4 CPU cycles since the last draw instruction or if it has been 10 CPU cycles since the last copy
- Chip8 erases sprites by drawing them over existing pixels which can cause extreme flickering

Draw Instruction Over Multiple Cycles

```
reg_addr1 = instruction[11:8];
reg_addr2 = instruction[ 7:4];
num_rows_written = {7'b0,stageminus16[31:7]};
mem_addr1 = num_rows_written + reg_I_readdata;
mem_request = 1'b1;
fb_addr_x = reg_readdata1 + ({5'b0, stageminus16[6:4]});
fb_addr_y = reg_readdata2 + ({4'b0, num_rows_written[3:0]});
fb_writedata = mem_readdata1[3'h7 - stageminus16[6:4]] ^ fb_readdata;
fb_WE = (num_rows_written < {28'h0, instruction[3:0]}) & (&(stage[3:0]));
bit_overwritten = (mem_readdata1[3'h7 - stageminus16[6:4]]) & (fb_readdata) & fb_WE;
isDrawing = 1'b1;
```

- The locations being drawn are a function of the stage
- We need to make sure the memory has enough time to propagate, which means that we are looking at the [6:4] bits of stage for x, and [10:7] for y
- 16 <= stage <= 272

Draw Instruction Over Multiple Cycles

Stage:

Number of rows written						X offset			WE			
…	11	10	9	8	7	6	5	4	3	2	1	0

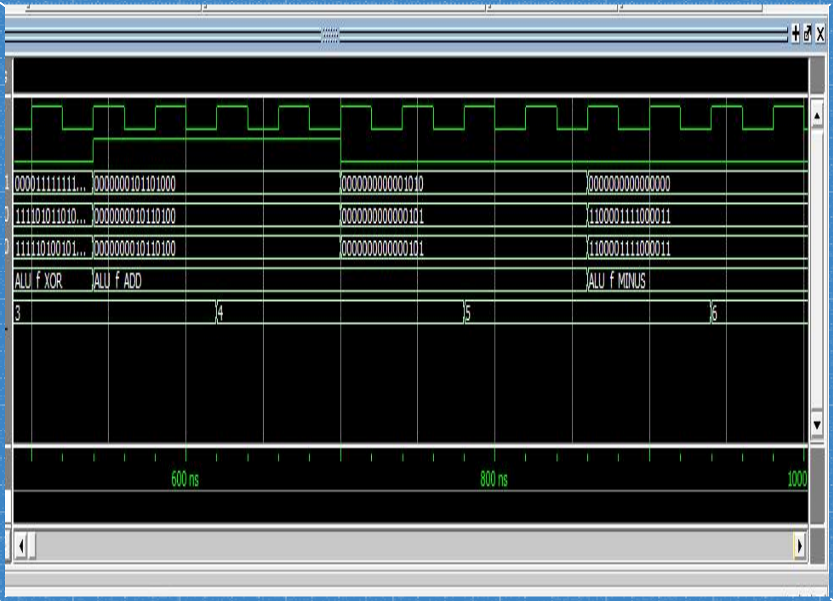
4

Testbenches Galore

Aggressively tested

Testbenched Modules

- CPU
- Stack
- ALU
- Memory
- Framebuffer
- Top level
- Random number generator
- BCD
- Register file



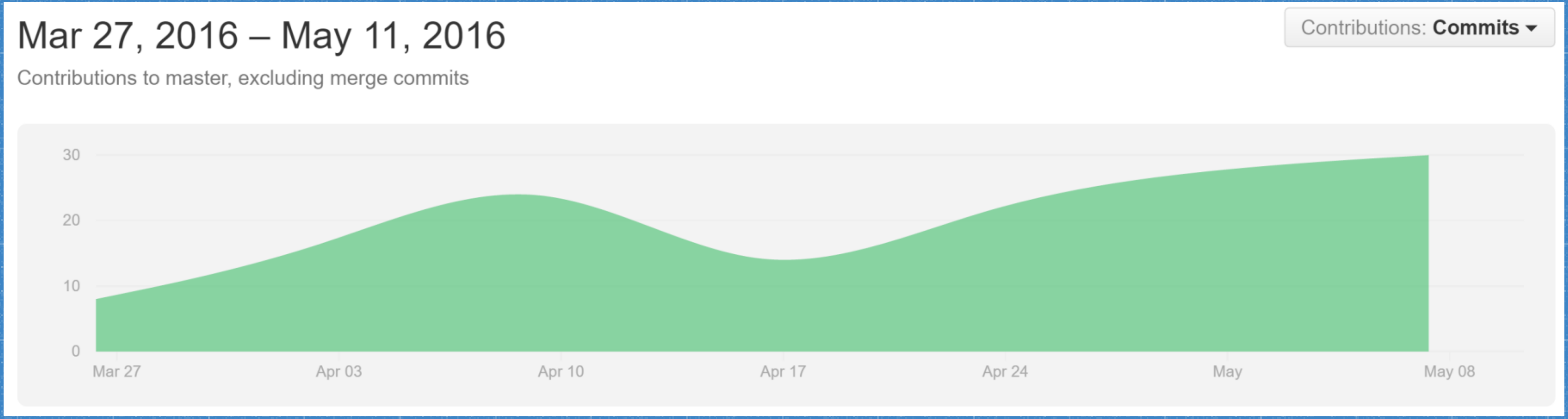
5

Project Workflow

Our tips to surviving all nighters in 1235 Mudd

Timeline

- We lagged behind goals during the semester, but we completed the final goal.


Mar 27, 2016 – May 11, 2016

Contributions to master, excluding merge commits

Challenges

- Memory was not always as ready as we were
- Installing Linux on an FPGA is more difficult than bathing cats
- Bugs are very common and FPGAs do not have proper pesticides yet

Lessons Learned

- Write testbenches early
- Test givens (including megafunctions, especially megafunctions)
- Start early!
- During testing, bugs are your best friend

Demo Time!

Hope you like Paddles and Tapeworms