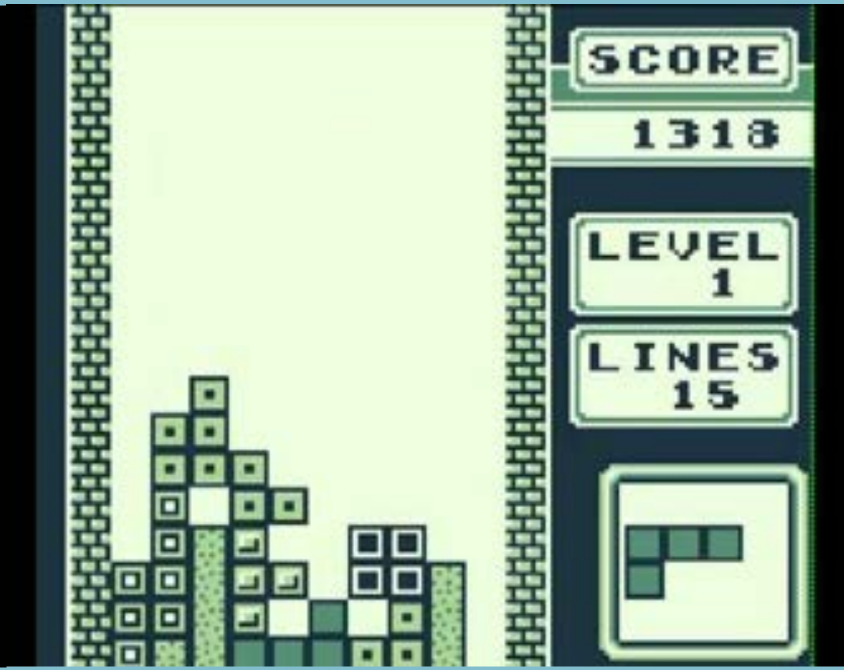
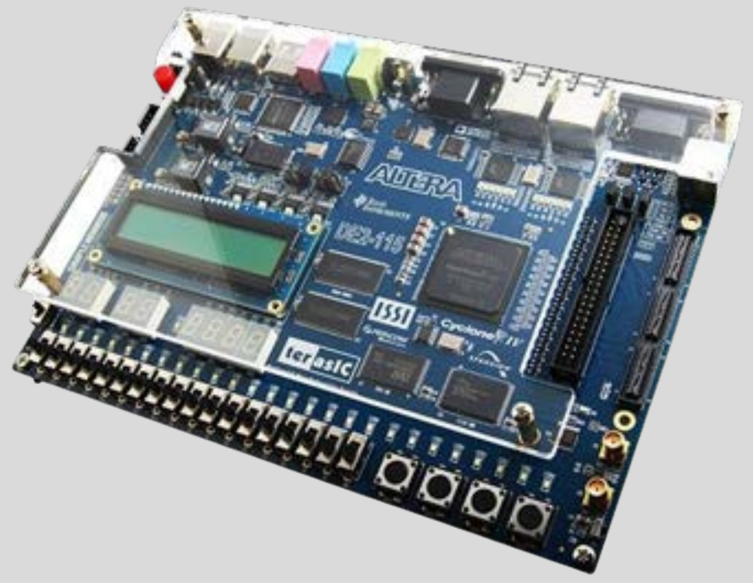
Use Case

RetroCore: FPGA Gameboy Emulator

- Hardware emulator of the original Gameboy (1989)
- <u>Users</u>:
  - Developers to study hardware implementation
  - Perspective gamers with GameBoy ROMs
- <u>MVP</u>:
  - Play Tetris/Dr.Mario ROM on hardware emulator




Altera DE II FPGA

Display Monitor, Speakers, USB Gameboy Controller

Design Requirements

01 **SYSTEM**

- Smooth Gameplay
- Minimal Visual Lag to user
- Modular Design

02 **CPU**
Central Processing Unit Requirements

- ~~4.19~~ 1 Mhz CPU Clock Speed
- Cycle Accurate Emulation

03 **GRAPHICS**
PPU (pixel processing unit) Requirements

- 100% palette mapping accuracy
- Frame Output Delay < 16.7 ms (spec)
- Frame Rate Consistency → Each scanline in **456** cycles, each frame for **70224** cycles (spec)
- Sprite Accuracy → X/Y Coords within **±1** pixel, Priority/Layering with **>90%** Correctness (spec)

04 **MEMORY**
MMU (memory management unit) Requirements

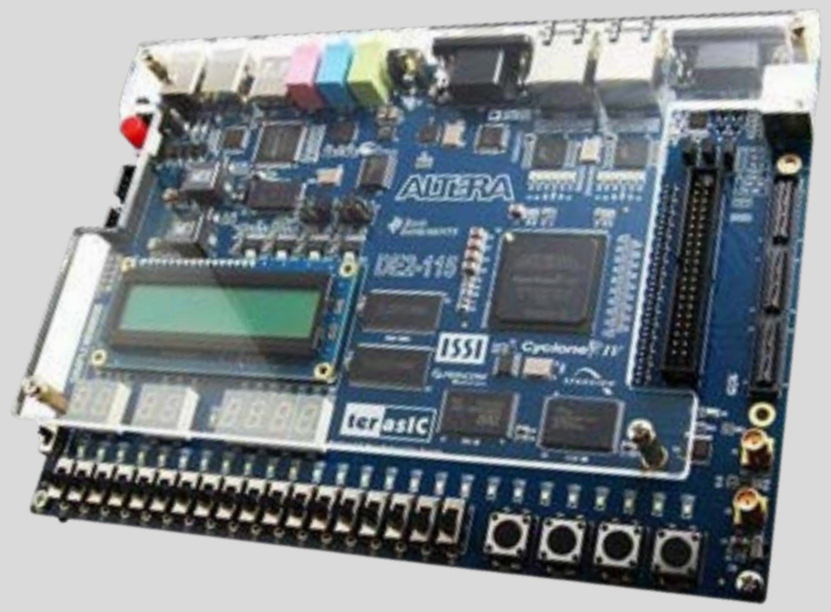
- Accurate Memory Map to Gameboy → ROM, VRAM, WRAM, External RAM, OAM, I/O Registers, HRAM (spec)
- Performs DMA → direct memory access, AKA when sprite data is rapidly dumped to OAM memory region (spec)

05 **DISPLAY/CONTROLLER**
I/O Requirements

- Button press processed within 32-48 ms (spec)
- Display must be 60 fps (modern video game standard)

06 **AUDIO**
APU (audio processing unit) Requirements

- >90% Tone Accuracy
- Bit-perfect Wave RAM (Channel 3)
- Pitch & Volume Check → Volume updates within **1** frame, pitch sweep changes every **7.8** ms (spec)
- Register Write-to-Output Latency < 10ms (spec)

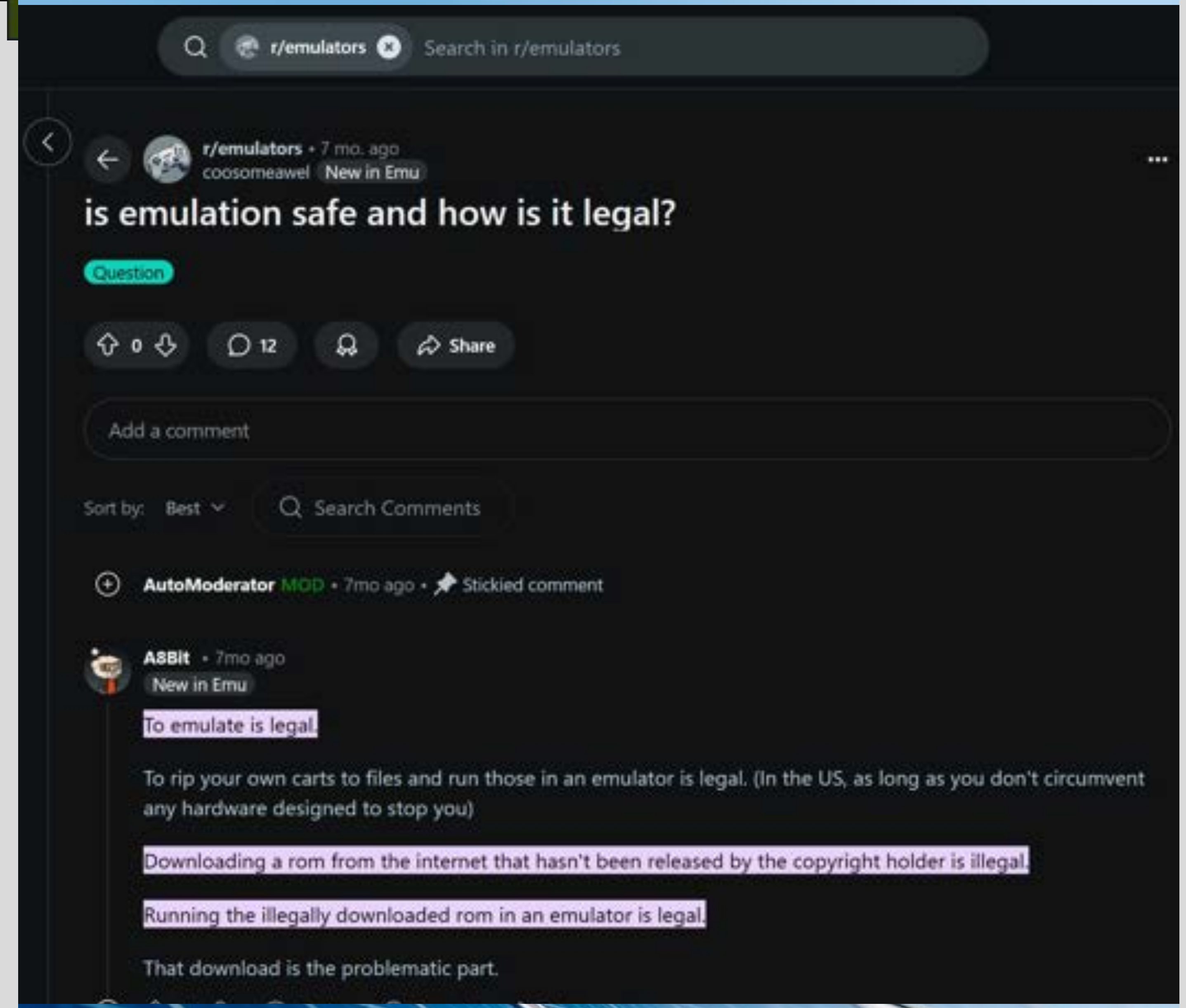


Altera DE2-115 FPGA
We chose this for its I/O capabilities, LUT sizes, and Memory Capacity (155 KB minimum, but more BRAMS => more games)

Category	Spec
On-Chip Memory	• **114k** LUTs • **432 KB** BRAMs
I/O	• VGA port (display) • USB port • **2MB** SRAM/SDRAM • **32** GPIO pins • Wolfson WM8741 Audio CODEC

Ethical Considerations

- Positives:
  - FPGAs are reconfigurable, less silicon waste
  - Design is larger and more user friendly than before (bigger screen and separated + wieldable controller)
- Red-teaming Analysis:
  - FPGAs are more power-hungry
  - ROMs can be illegally acquired, violating copyright law (impacting Nintendo)
    - Emulators are legal, however



r/emulators • 7 mo. ago
coosomeawel New in Emu

is emulation safe and how is it legal?

AutoModerator MOD • 7mo ago • Stickied comment

A8Bit • 7mo ago
New in Emu

To emulate is legal.

To rip your own carts to files and run those in an emulator is legal. (In the US, as long as you don't circumvent any hardware designed to stop you)

Downloading a rom from the internet that hasn't been released by the copyright holder is illegal.

Running the illegally downloaded rom in an emulator is legal.

That download is the problematic part.

Katherine Parry

CPU

Central Processing Unit

Bharathi Sridhar

PPU

Pixel Processing Unit

Ruslana Fogler

MMU

Memory Management Unit

Ruslana Fogler

APU

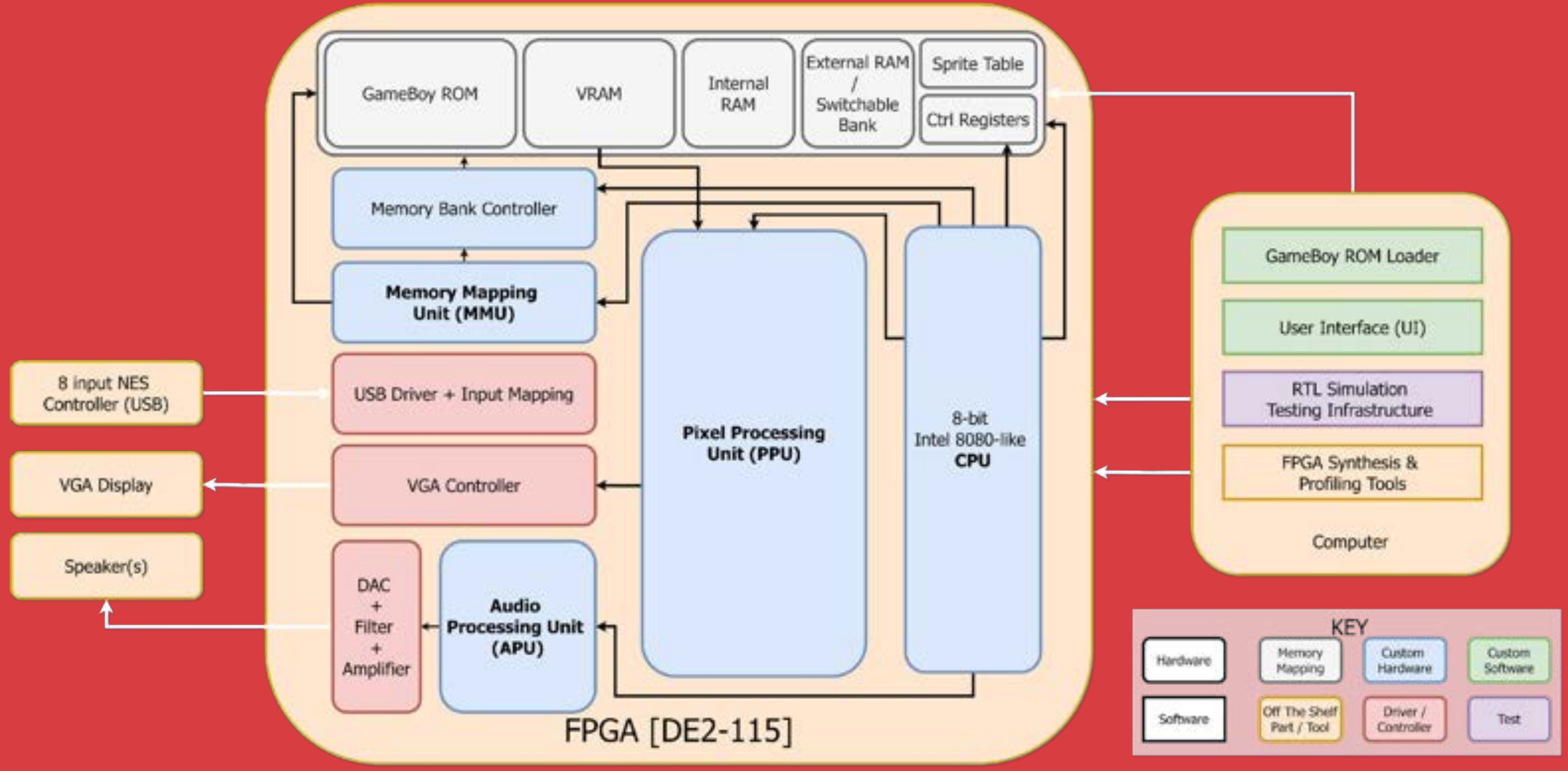
Audio Processing Unit

Everyone

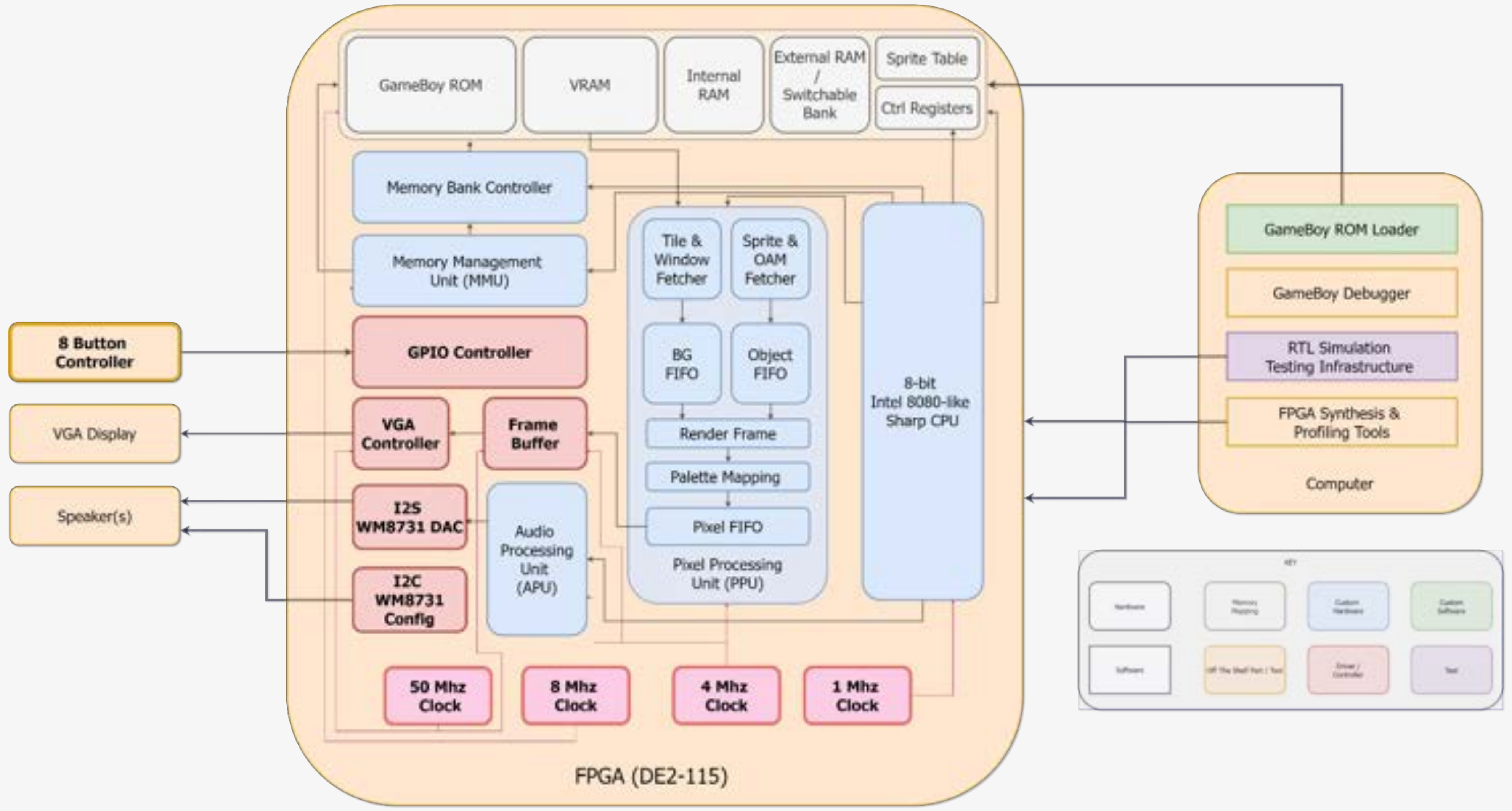
I/O

Controller, Display, Audio

Initial Solution Approach


GameBoy ROM
VRAM
Internal RAM
External RAM / Switchable Bank
Sprite Table
Ctrl Registers
Memory Bank Controller
Memory Mapping Unit (MMU)
USB Driver + Input Mapping
VGA Controller
DAC + Filter + Amplifier
Audio Processing Unit (APU)
Pixel Processing Unit (PPU)
8-bit Intel 8080-like CPU
FPGA [DE2-115]
8 input NES Controller (USB)
VGA Display
Speaker(s)
GameBoy ROM Loader
User Interface (UI)
RTL Simulation Testing Infrastructure
FPGA Synthesis & Profiling Tools
Computer
KEY
Hardware
Memory Mapping
Custom Hardware
Custom Software
Software
Off The Shelf Part / Tool
Driver / Controller
Test

Complete Solution



Design Changes:

- GPIO Controller (USB abandoned)
- **4** Clocks in Design:
  - 1 Mhz CPU
  - 4 Mhz PPU, APU, Frame Buffer
  - 8 Mhz MMU, BRAM modules
  - 50 Mhz VGA, Frame Buffer
- Potentially not integrating APU
  - but it is written + much synthesis work has been done

Our final deliverable for demo will be live play.

Previously…

Test/Verification/Validation

Metric	Test Method	Input	Success Output	Status
Clock Speed	Synthesis	FPGA Clock	CPU operates at 4.19 Mhz	complete
CPU Accuracy	Simulation	Blargg Tests	All tests pass (~25 million lines of assembly)	complete
PPU Accuracy	Simulation	Custom Test ROMs	Random Frames match Software Emulator Dump	complete
MMU Accuracy	Simulation	Personal Testbench queries	Basic memory queries, register interrupts, CPU/PPU servicing timing, and DMA triggers behave correct	complete
Frame Rate	Simulation	Custom Test ROMs	Frame Output Delay < 16.7ms	complete
Palette Mapping	Simulation	Custom Test ROMs	100% palette mapping accuracy	complete
Frame Rate Consistency	Simulation	Custom Test ROMs	Each scanline = 456 cycles, Each frame = 70224 cycles	complete
Sprite Position Accuracy	Simulation	Custom Test ROMs	Sprite X/Y chords within ±1 pixel, Sprite Priority >90%	in progress
Qualitative Checks	Custom Check	Simulation Dump	Matches Software Emulator Dump with 90% Accuracy	in progress
Wave Frequency	Oscilloscope	NR13/14, NR33/NR34	Matches Software Emulator wave tones	not complete
Wave RAM Check	Logic Analyzer	Set to Wave RAM	Matches Software Emulator wave RAM memory dump	not complete
Pitch Check	Oscilloscope	Set NR10, NR13/NR14	Matches Software Emulator pitch display	not complete
Volume Check	Oscilloscope	Set NR12/NR22, Trigger NR14	Matches Software Emulator volume display	not complete
Glitch-free Gameplay	Video Recording	Video Recording of Frame	If video recording of frames matches emulator frames	not complete
Controller Lag	Video Recording	Videos of Button Presses	If button LED responses are <30 ms (spec)	complete
Display Consistency	Video Recording	Rendering Integration Frames	Frame-by-frame differences are incremental x/y updates	complete

Currently…

Test/Verification/Validation

- CPU Verified via:
  - Blargg Tests (on Simulation)
- PPU Verified via
  - DMG Acid Test (on Simulation)
- MMU Verified via:
  - Basic BRAM tests (on FPGA)
  - Personal TB (on Simulation)
- Integration:
  - Using Sameboy to simulate and verify framedumps
- I/O Verified via:
  - Basic VGA test, upscale
  - Basic controller GPIO ⇔ LED test

SAMEBOY

A Friendly and Powerful Game Boy Emulator

Home Features Downloads Changes Bug Tracker GitHub Sponsor

About SameBoy

SameBoy is a user friendly, powerful and open source Game Boy, Game Boy Color and Super Game Boy emulator for macOS, iOS, watchOS, Windows and Unix-like platforms. SameBoy is extremely accurate and includes a wide range of both powerful debugging features and user-facing features, making it ideal for both casual players and developers. Of course, SameBoy also has every feature one would expect from an emulator – from save states to scaling filters.

Download SameBoy 1.0.1 for Windows

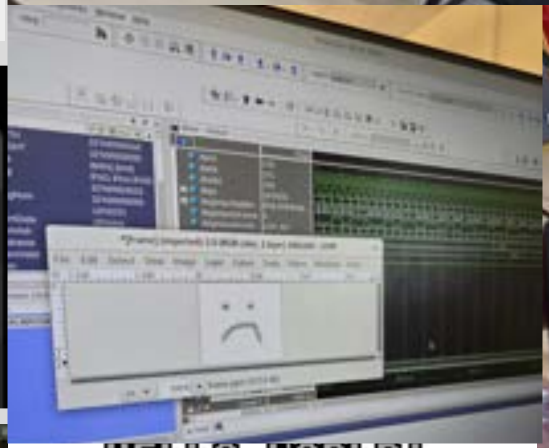
SameBoy for iOS is available on the App Store

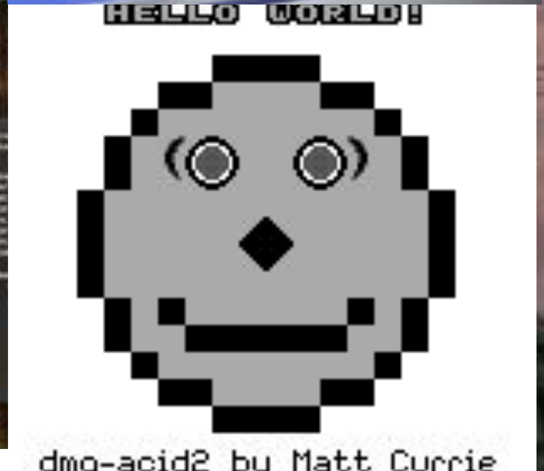
Super Mario Land 2

Game Boy

SameBoy Copyright Lior Halphon © 2015-2025 LIJI32 LIJI32

VRAM Viewer – Tetris (JUE) (V1.1) [!].gb
Tileset Tilemap Objects Palettes

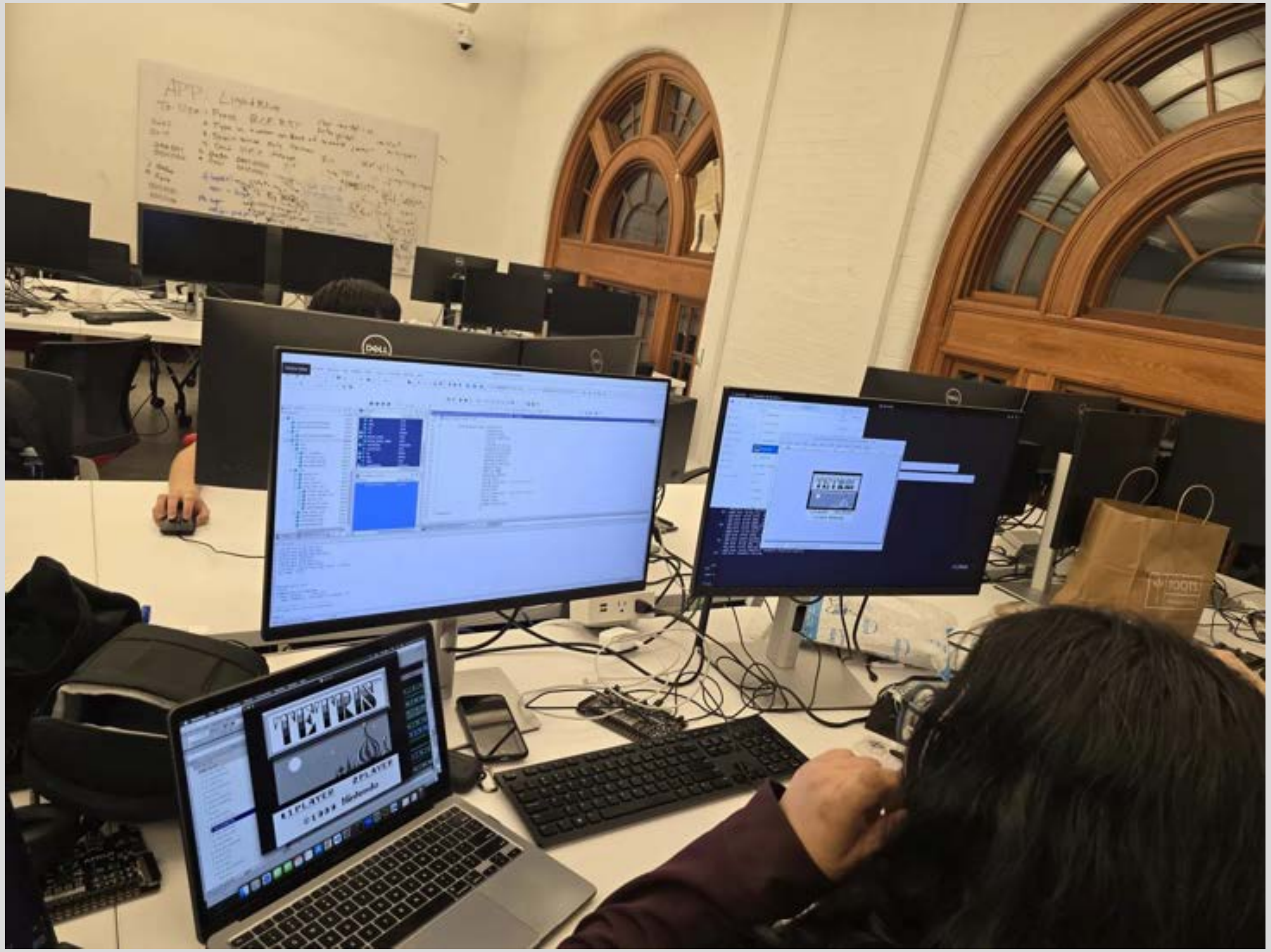




Currently…

Test/Verification/Validation

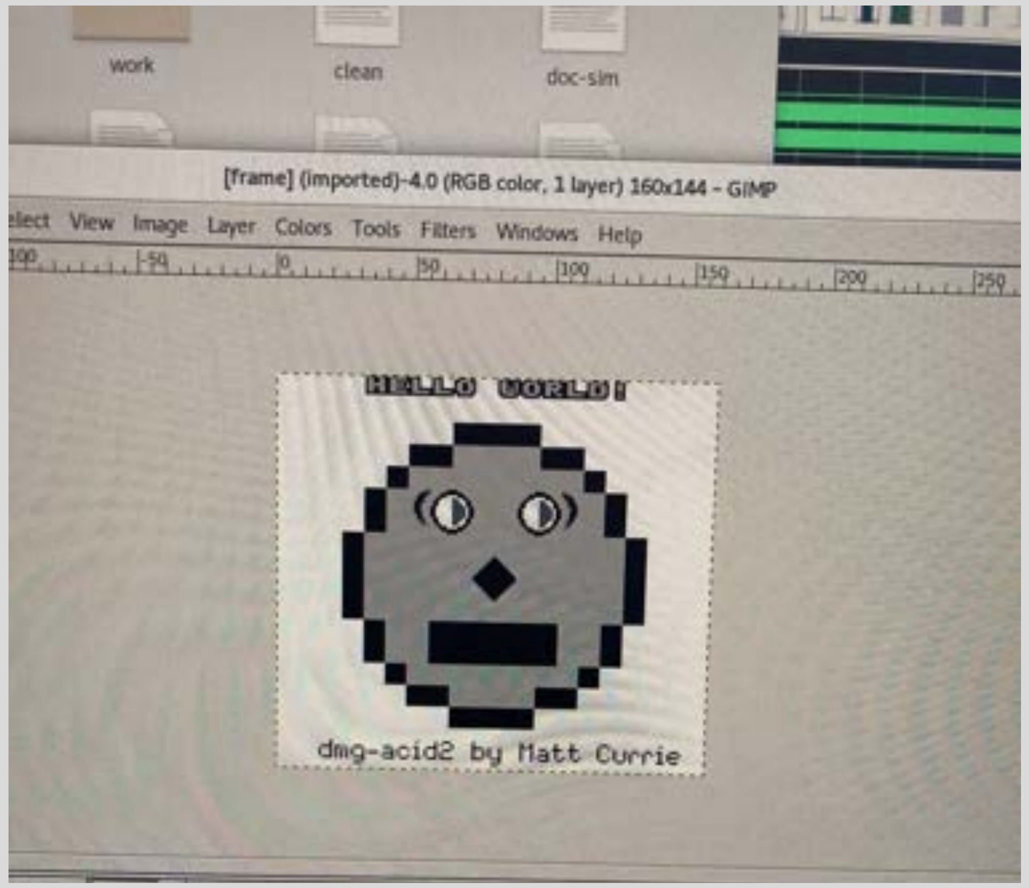
- Debugging Methods on Simulation:
  - Modelsim waveform
  - Sameboy Comparison (ground truth)
- Debugging Methods on Synthesis:
  - Slower clock on FPGA
  - SignalTap real-time logic analyzer on Quartus
- Quartus Results:
  - *8% LUT usage*
  - *45% Memory BRAM Usage*

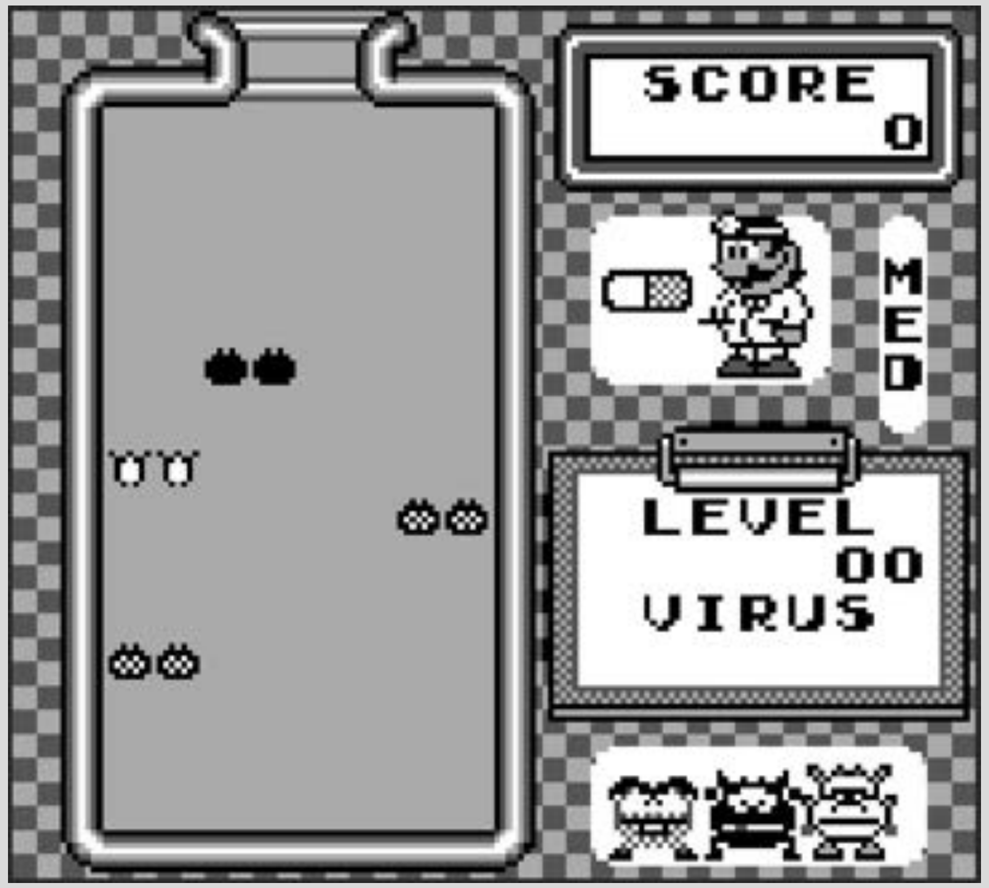


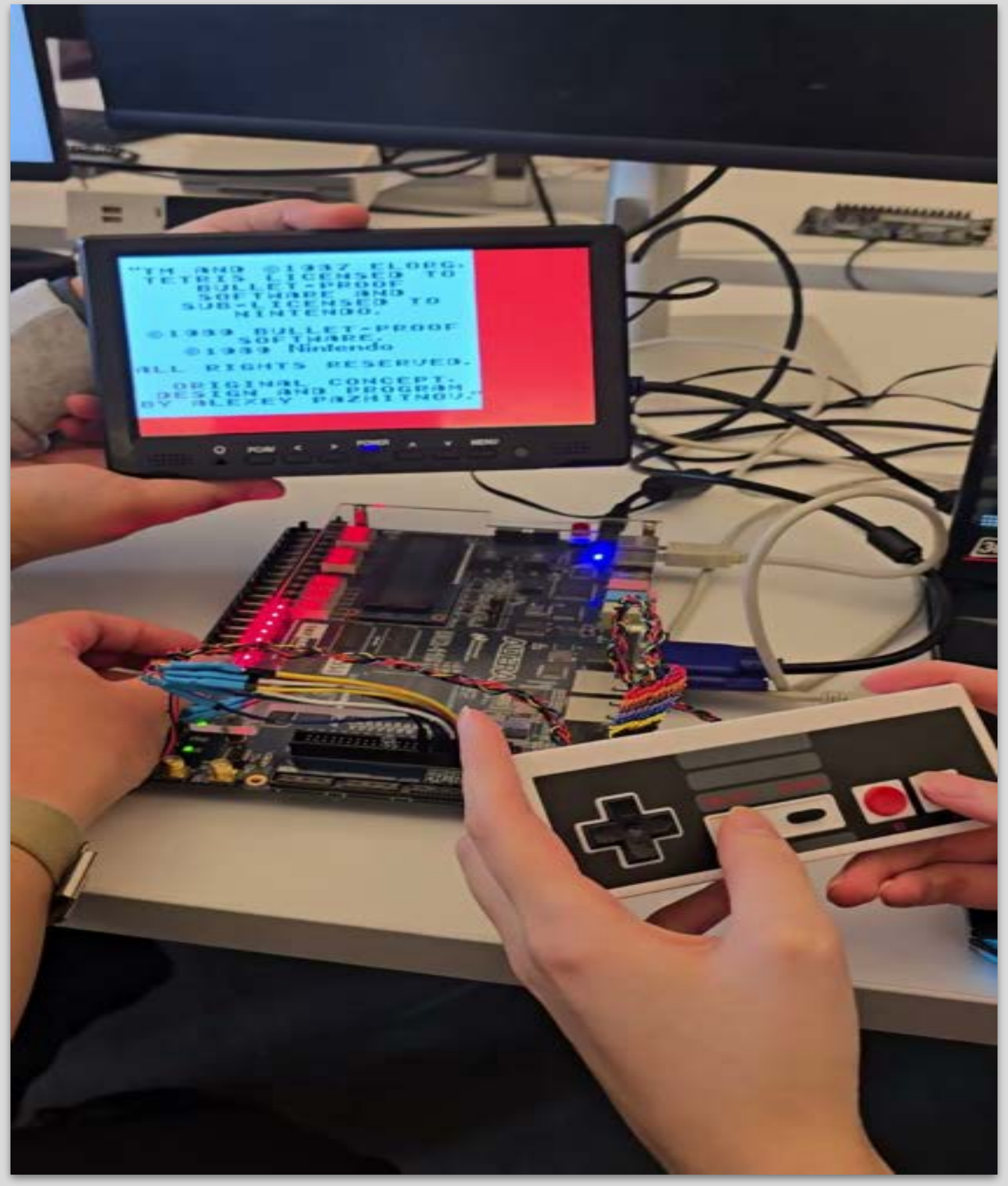
Our Latest

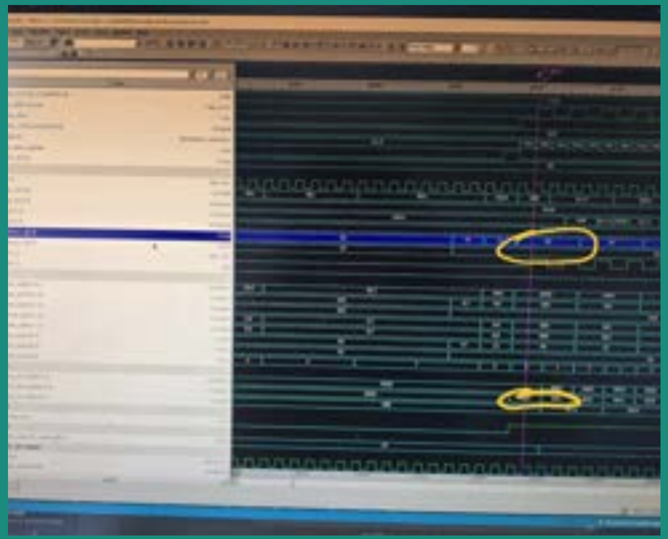
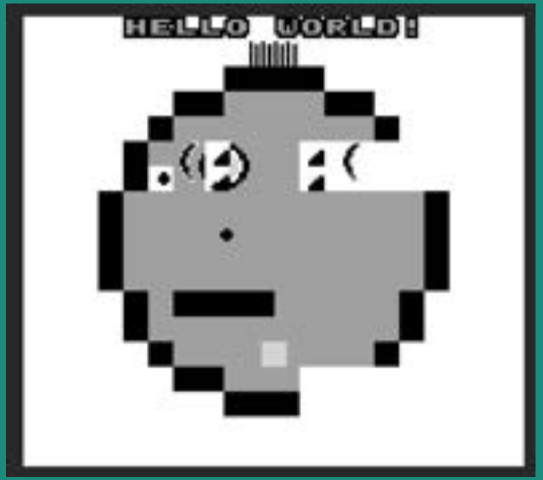
Implementation Performance

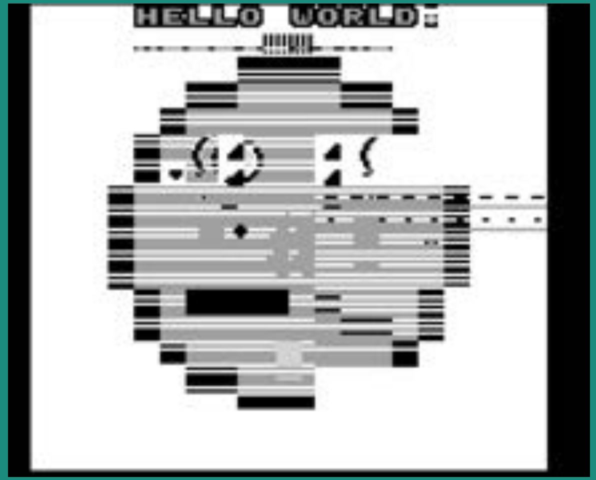
- Tetris is mildly playable
- Dr.Mario appears but freezes (needs debugging)
- Timing of game is 2x faster (needs debugging)
- Video-taping for glitches → needs to be done

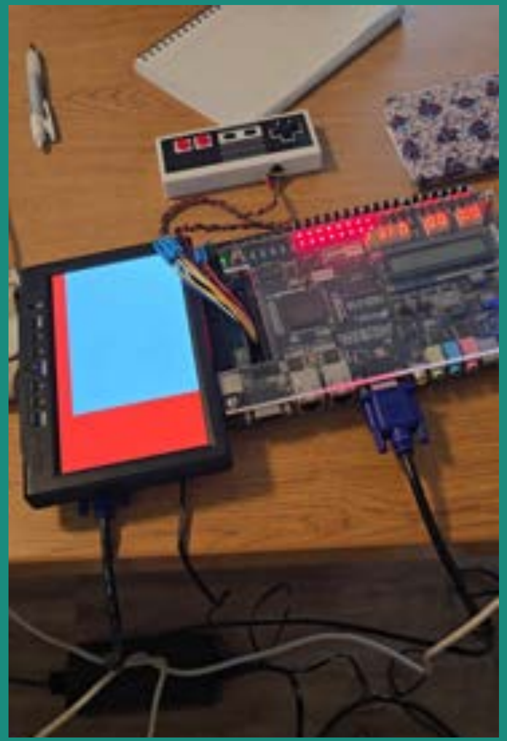
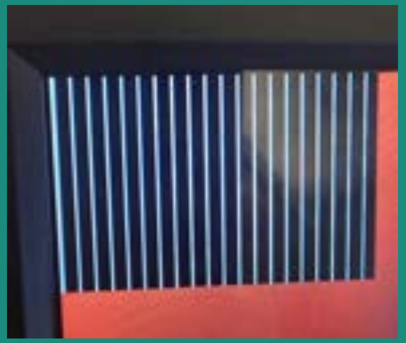
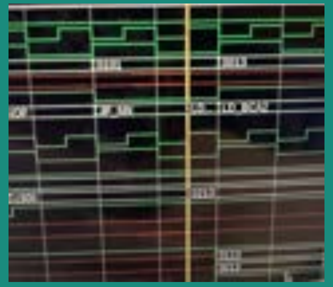

work
clean
doc-sim
[frame] (imported)-4.0 (RGB color, 1 layer) 160x144 – GIMP
Select View Image Layer Colors Tools Filters Windows Help
HELLO WORLD!
dmg-acid2 by Matt Currie

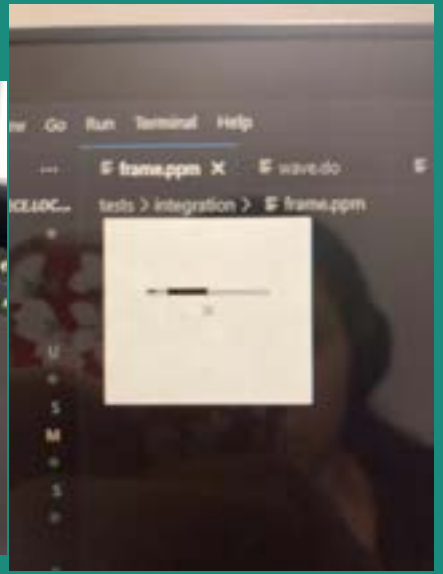


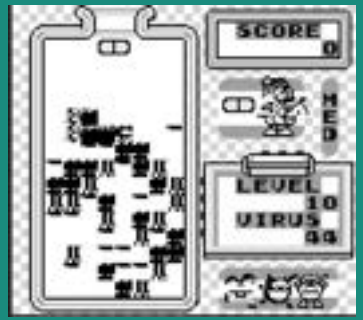


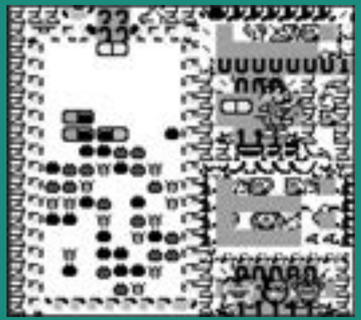
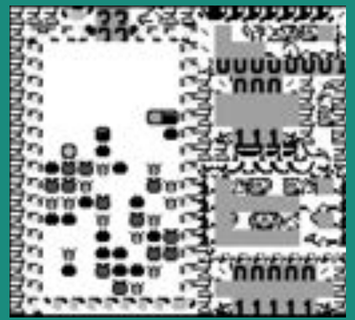



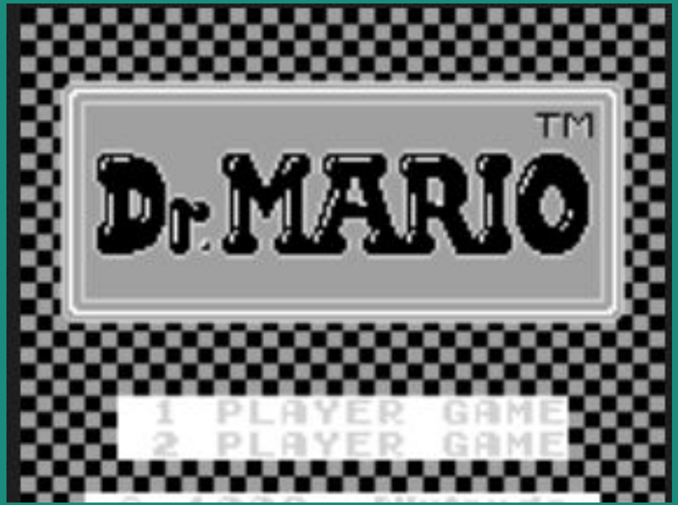


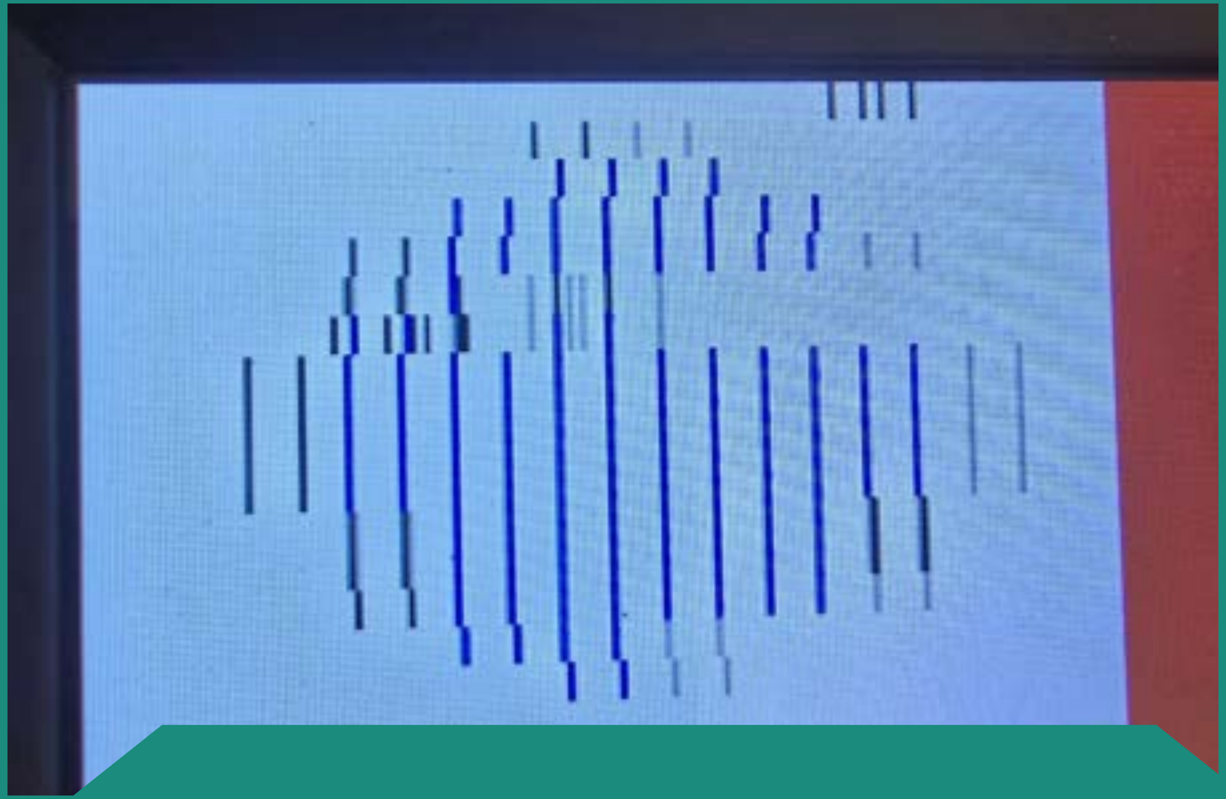
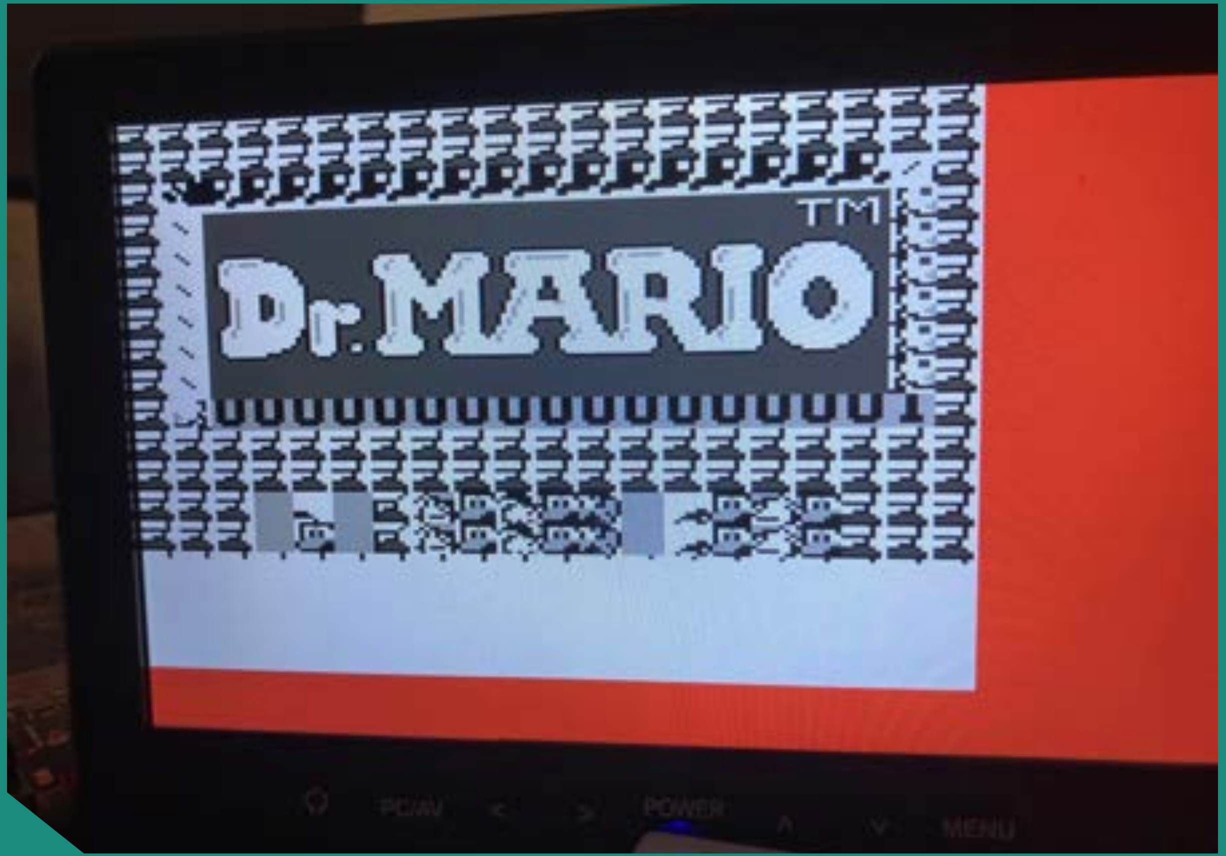










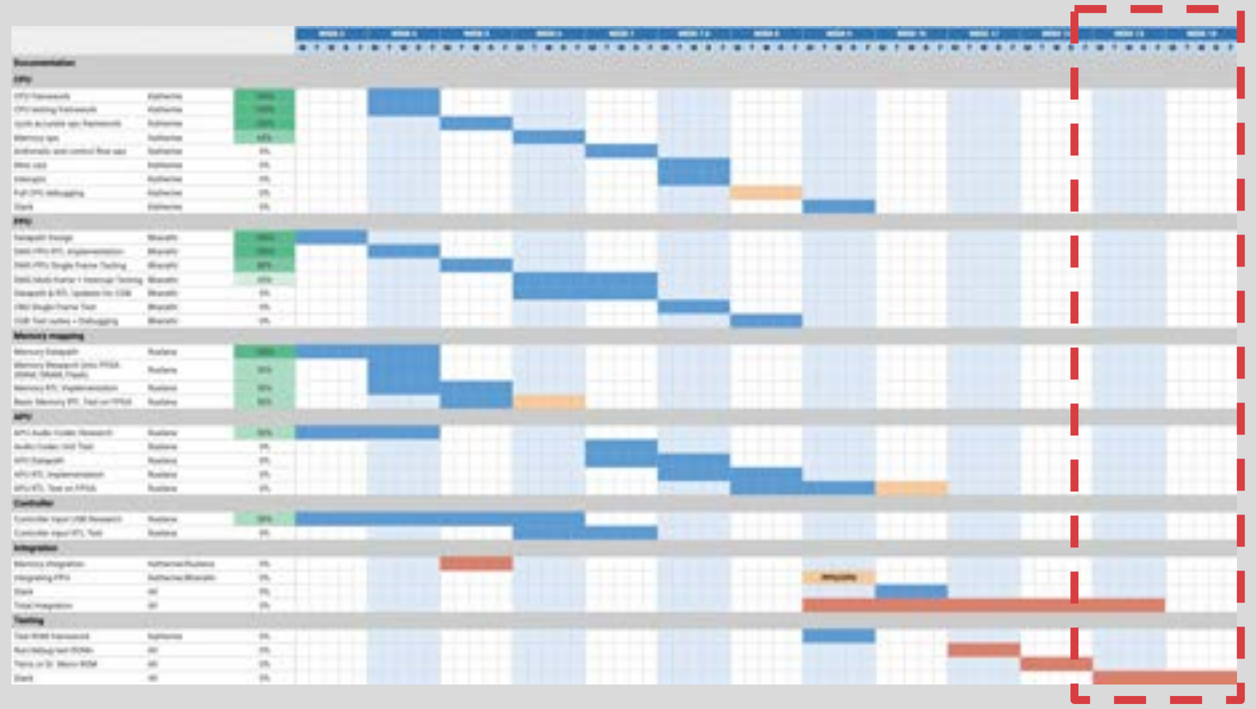



Gallery of Bugs

Our Latest

Design Tradeoffs

- Shift to 16 bit addresses/data
  - 2x bandwidth (original is 8 bit)
  - Less states, but more headache
- Switching from USB to GPIO
- Reducing small features
  - ECHO RAM
  - Potentially Bank Switching
  - Implementing known Gameboy bugs
- Potentially cutting APU
- Remaining Objectives:
  - Tetris/Dr.Mario bug-free
  - Reach goal: Audio?
  - Reach goal: Adding Pokemon/Link’s Awakening



Our Greatest Challenges

- Fighting Gameboy Spec
  - Lots of conflicting information online!
  - Had to go interpret game assembly many times
  - Compare raw register dump with emulators
- Clock Frequency confusion
  - Anyway online claims 4 Mhz clock
  - Subtle misdirection: **1 Mhz CPU clock**
- I/O Challenges
  - Altera USB integration is very involved
  - Audio CODEC is difficult to debug