# Game Boy Emulation on FPGA

Game Boi - Team CO:
Tess (Therese) Chan
Pratyusha Duvvuri
Adolfo Victoria

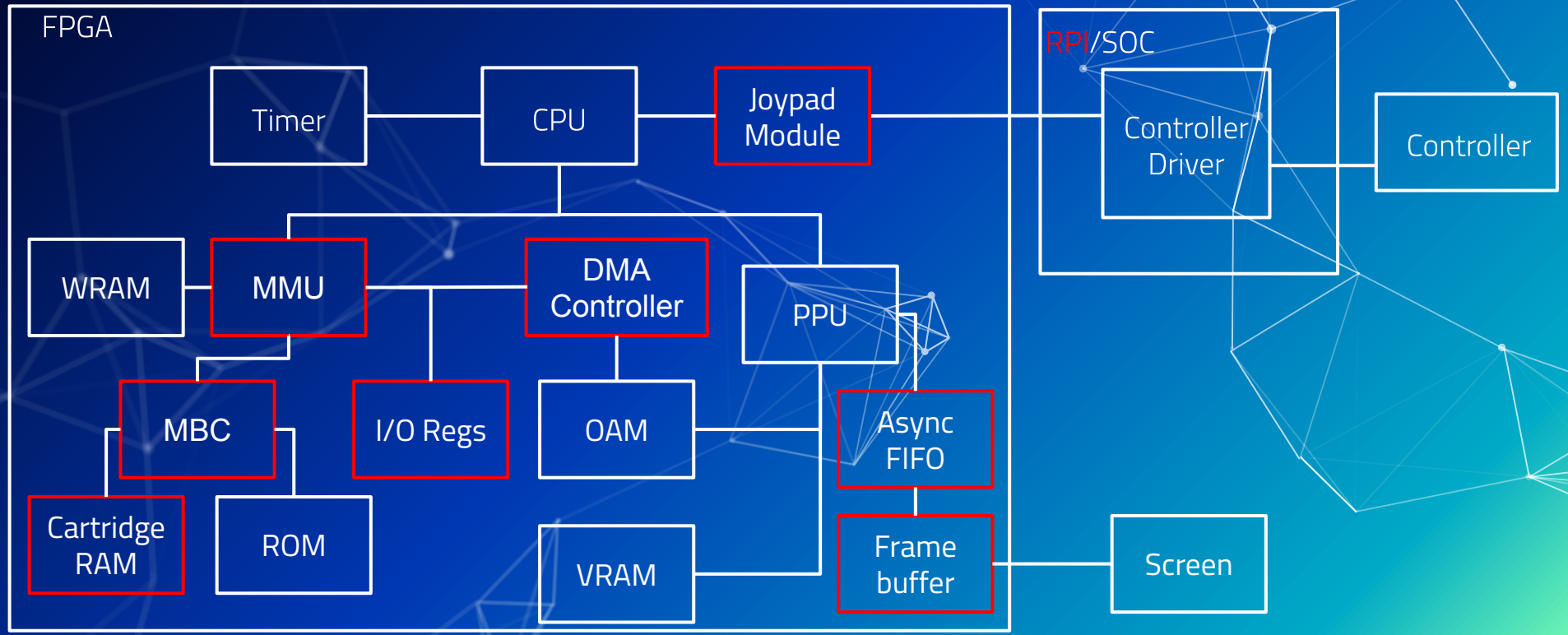
# Application Area

- Creating a cycle accurate Game Boy emulator on an FPGA
- Recreating classic systems on modern hardware to learn how older systems were implemented
- Areas: Hardware, Software

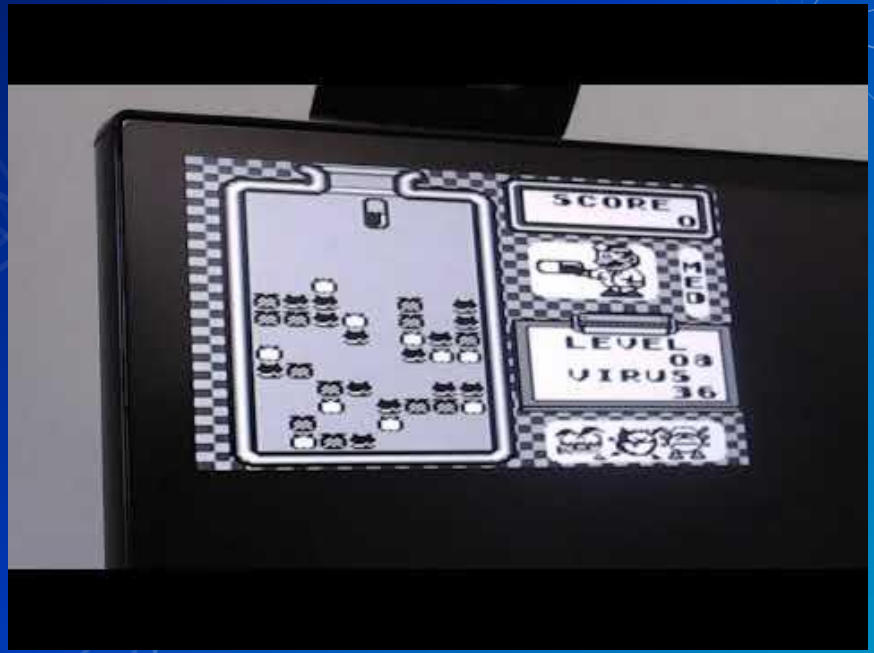
# Solution Approach

- Use DE10-Standard development board with on board FPGA and SoC/external Raspberry Pi (RPi)
  - SOC/RPi connected to FPGA through GPIO pins
  - SoC/RPi will handle: controller inputs
  - FPGA will handle: CPU, PPU, Timer, MMU, and MBC

# Updated System Diagram (Updates in Red)



# Solution



# Requirements

- Pass cycle accuracy tests made by the emulator community
  - Goal is to achieve comparable results to popular Game Boy emulators
- Games should run at 59.7 frames per second, just like the original
- Input latency must be at most 55 ms
- Able to play Tetris and Doctor Mario flawlessly
  - Selected because they do not require memory bank switching and fit nicely on to the FPGA
- Update from design presentation: Audio and SoC memory related requirements were removed for second half

# Testing Approach

- Simulation
  - Created testing script to run all the emulator tests to indicate any regressions
  - We modified a “TV-emulation” script, made by **Ford Seidel**, to show the outputs from the GPU in simulation
- FPGA
  - Due to the scale of the project, we could not use SignalTap to debug, so we mostly debugged in simulation and just uploaded to the FPGA after
- Input
  - Recorded us pressing a button in slow motion and measured the frames between the press and an update showing
- SoC
  - Created unit tests to read controller input and relay to FPGA peripherals-LEDS
  - SDRAM Test

# Requirement Results: Accuracy Tests

- Mooneye tests results
  - 30/30 acceptance tests
  - 2/3 OAM DMA tests
  - 4/12 PPU tests
  - 1/1 interrupt timing test
  - 13/13 MBC1 and MBC5 tests
  - 13/13 timer tests
- Blargg tests results
  - 11/11 instruction accuracy tests
  - 3/3 mem_timing1 and 2
  - 1/1 instr_timing
  - 1/1 interrupt_timing
- Our results surpass most software emulators and all hardware emulators we found, including VerilogBoy, which was our initial goal
  - This excludes the MiSTer FPGA open source project

# Other Requirement Results

- Able to achieve the desired framerate
  - Required having the correct VSYNC and HSYNC timings and using the same clock as the Game Boy
- We have an estimated input lag of: 8.3-12ms
  - Our input lag was 2 frames at 240FPS, giving 8.3 ms of input lag
- The emulator runs Doctor Mario and Tetris with no graphical or instruction glitches
  - Also able to run more complex games as well
- Passed unit tests for controller SoC communication and controller
- Data transmission between SDRAM and Nios II for memory management

# Trade-offs

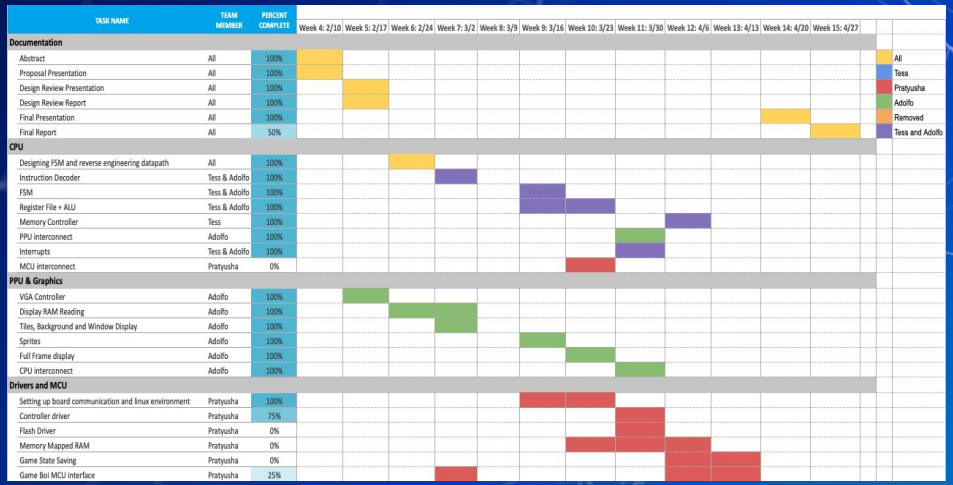
- **Using multiple FSMs** vs. One large FSM
  - Worked
  - Created modular implementation and easier to debug
- **Combinational read** vs. Synchronous read
  - Worked, but not in the assumed way
  - The FPGA did not support combinational read RAM, so we used a work around
- **On board SoC** vs. External microprocessor
  - Achieved SoC communication with FPGA peripherals
  - Proved SoC could talk to FPGA with other sub-program running on FPGA
  - Integration showed issues with running FPGA and SoC programs simultaneously, devised RPI solution
  - However, now we can run programs on both, simultaneously

# Limitations of Solution

- Save game state
- Change game without reflashing FPGA
  - Currently, SoC is being worked on to support this feature and will be worked on before the final demo
  - The FPGA does not have enough memory to hold multiple games
- Unable to fit games bigger than 512kB
  - These are few, but we can't fit them without using SDRAM, for which we would need the SoC to flash
  - Currently, we are trying to:
    - Workaround with a different board
    - Integrate SoC memory management

# Project Management

- Work to be completed before final demo:
  - Get MBC working in synthesis and test more complex games
  - SoC memory management w/ FPGA running
  - Polish code style



| TASK NAME | TEAM MEMBER | PERCENT COMPLETE |
|---|---|---|
| **Documentation** | | |
| Abstract | All | 100% |
| Proposal Presentation | All | 100% |
| Design Review Presentation | All | 100% |
| Design Review Report | All | 100% |
| Final Presentation | All | 100% |
| Final Report | All | 50% |
| **CPU** | | |
| Designing FSM and reverse engineering datapath | All | 100% |
| Instruction Decoder | Tess & Adolfo | 100% |
| FSM | Tess & Adolfo | 100% |
| Register File + ALU | Tess & Adolfo | 100% |
| Memory Controller | Tess | 100% |
| PPU interconnect | Adolfo | 100% |
| Interrupts | Tess & Adolfo | 100% |
| MCU interconnect | Pratyusha | 0% |
| **PPU & Graphics** | | |
| VGA Controller | Adolfo | 100% |
| Display RAM Reading | Adolfo | 100% |
| Tiles, Background and Window Display | Adolfo | 100% |
| Sprites | Adolfo | 100% |
| Full Frame display | Adolfo | 100% |
| CPU interconnect | Adolfo | 100% |
| **Drivers and MCU** | | |
| Setting up board communication and linux environment | Pratyusha | 100% |
| Controller driver | Pratyusha | 75% |
| Flash Driver | Pratyusha | 0% |
| Memory Mapped RAM | Pratyusha | 0% |
| Game State Saving | Pratyusha | 0% |
| Game Boi MCU interface | Pratyusha | 25% |



| | | |
|---|---|---|
| **Memory** | | |
| Block RAM | Tess | 100% |
| ROM loading | Adolfo | 100% |
| DMA Controller | Tess & Adolfo | 100% |
| Memory Bank Controller | Tess & Adolfo | 100% |
| CPU integration | Tess | 100% |
| **Audio** | | |
| Audio research | Tess | 100% |
| APU design | Tess | 100% |
| ~~Waveform output~~ | Tess | 0% |
| ~~Full stereo output~~ | Tess | 0% |
| ~~Game Boi integration~~ | Tess | 0% |
| **Testing (and debugging)** | | |
| Blarggs Tests | Tess & Adolfo | 100% |
| Mooneye Tests | Tess & Adolfo | 100% |
| Video Tests | Tess & Adolfo | 100% |
| ~~Audio Tests~~ | All | 0% |
| Input Lag Tests | Tess & Adolfo | 100% |
| **Joypad** | | |
| Make a Joypad Module | Adolfo | 100% |
| Joypad RPI driver | Adolfo | 100% |
| Connect the RPI to the FPGA through Synthesis | Adolfo | 100% |
| Joypad CPU Interconnect | Adolfo | 100% |