Use Case / Application

Motivation

- Original GameBoy hardware is not in production
- Game developers needing an accurate hardware environment for testing and optimization

Use Case

- Gamers looking to play GameBoy ROMs on modern hardware
- Developers interested in FPGA-based emulation of Gaming Systems

MVP

Play ***Tetris*** and ***Dr. Mario***

Design Requirements [1]

USE CASE REQUIREMENTS	DESIGN REQUIREMENTS	
	Subsystem	Constraint
4.19MHz CPU Clock Speed →	CPU	Synthesize CPU with Clock Speed 4.19MHz
CPU Accuracy →	CPU	Cycle Accurate
60 Frames per second →	Graphics (PPU)	Frame Output Delay < 16.7 ms
Smooth gameplay, minimal visual lag →	Graphics (PPU)	100% palette mapping accuracy
	Graphics (PPU)	Frame Rate Consistency: • Each scanline drawn in 456 cycles • Each frame lasts 70224 cycles
	Graphics (PPU)	• Sprite X/Y coords within ±1 pixel 90% of the time • Sprite priority and layering with > 90% correctness

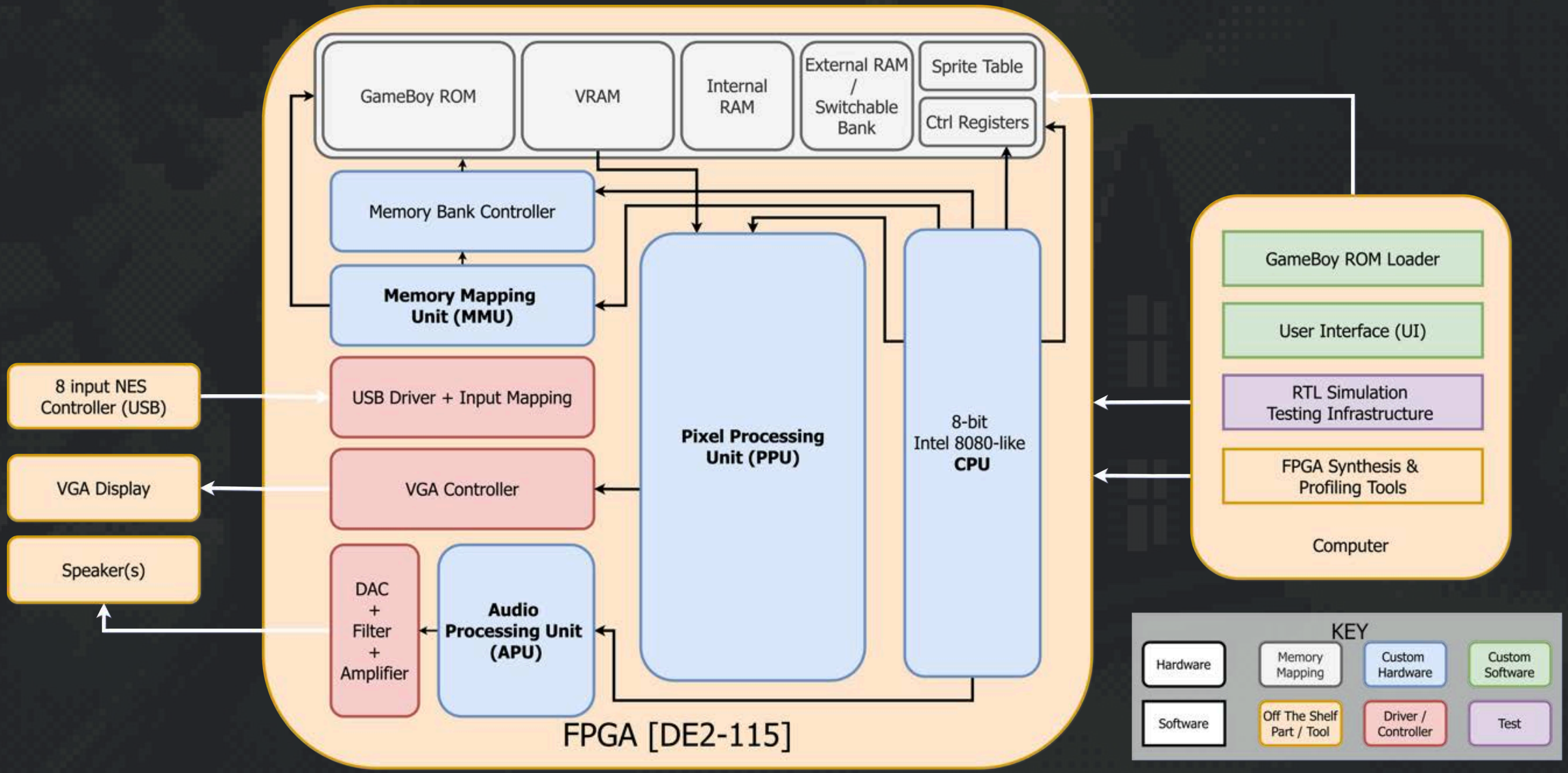
Design Requirements [2]

USE CASE REQUIREMENTS

DESIGN REQUIREMENTS

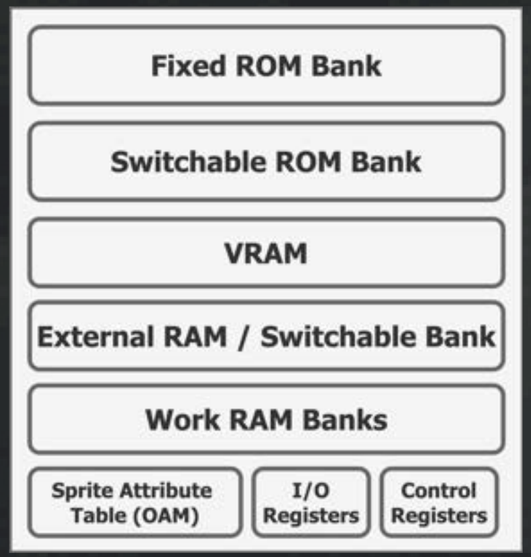
USE CASE REQUIREMENTS	Subsystem	Constraint
Minimal audio lags / glitches	**Audio**	> 90% Wave Frequency Accuracy
	Audio	Bit-perfect Wave RAM (Channel 3)
	Audio	Pitch & Volume check: • Volume envelope updates within 1 frame • Pitch sweeps change every 7.8ms step
Minimal audio stalling	**Audio**	Register Write-to-Output Latency < 10ms
32ms to 48ms Input Lag	**Controller**	Button press processed within 30 ms
Customizable, Upgradable	**Full System**	Highly Modular Design

Complete System Architecture



Solution Approach - FPGA, Memory, Controller

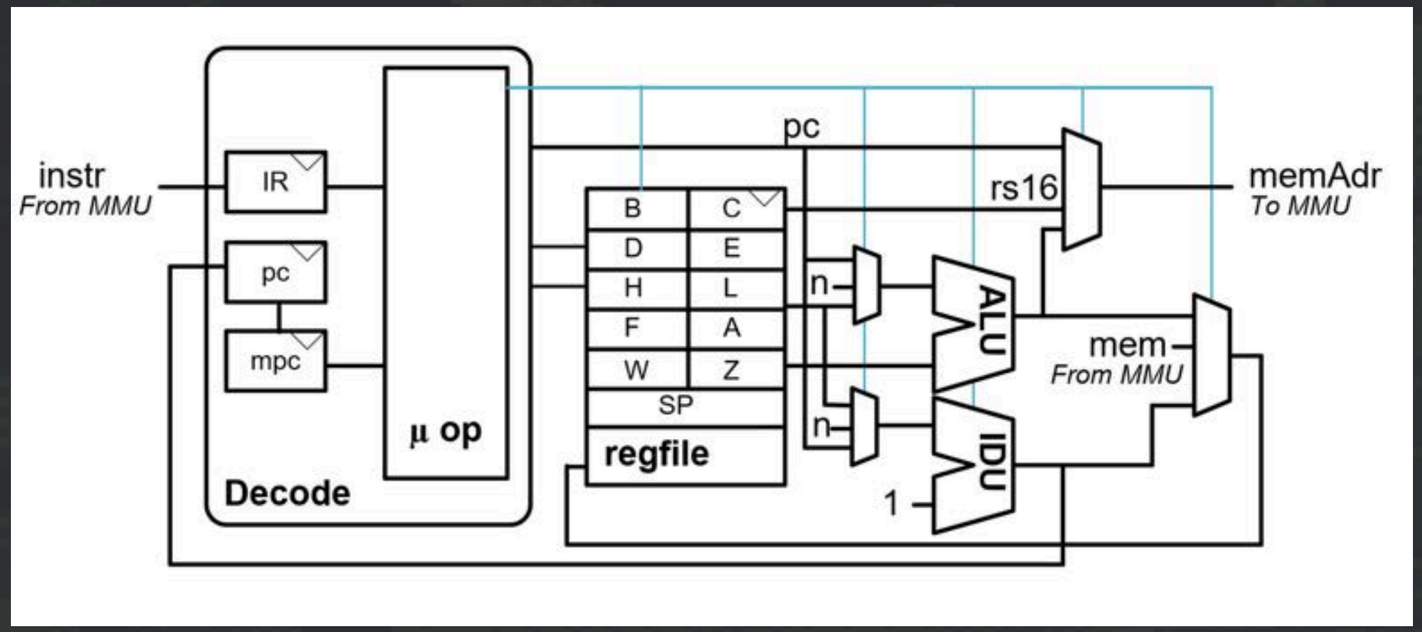
- 114k LEs
- 432 KB Block RAM + 2MB SRAM + 128MB SDRAM
- Support for USB 2.0, VGA, Audio CODEC
- Familiar Altera/Intel toolchain



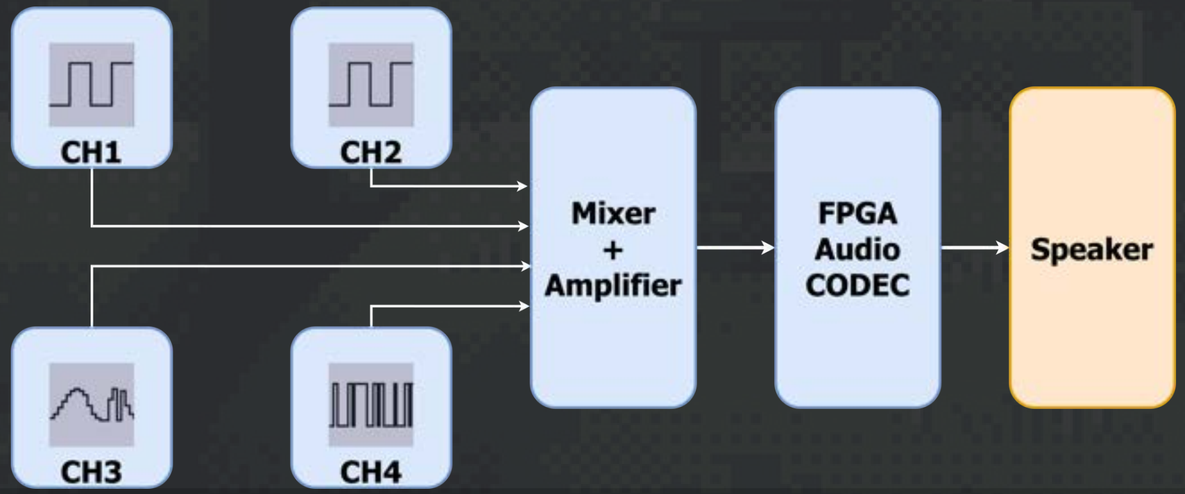
- Memory Mapping
- Memory Controller
- Concurrent Reads & Writes
- Block RAM
- SRAM / SDRAM

- USB 2.0
- Supported by other emulators

Solution Approach - CPU, APU

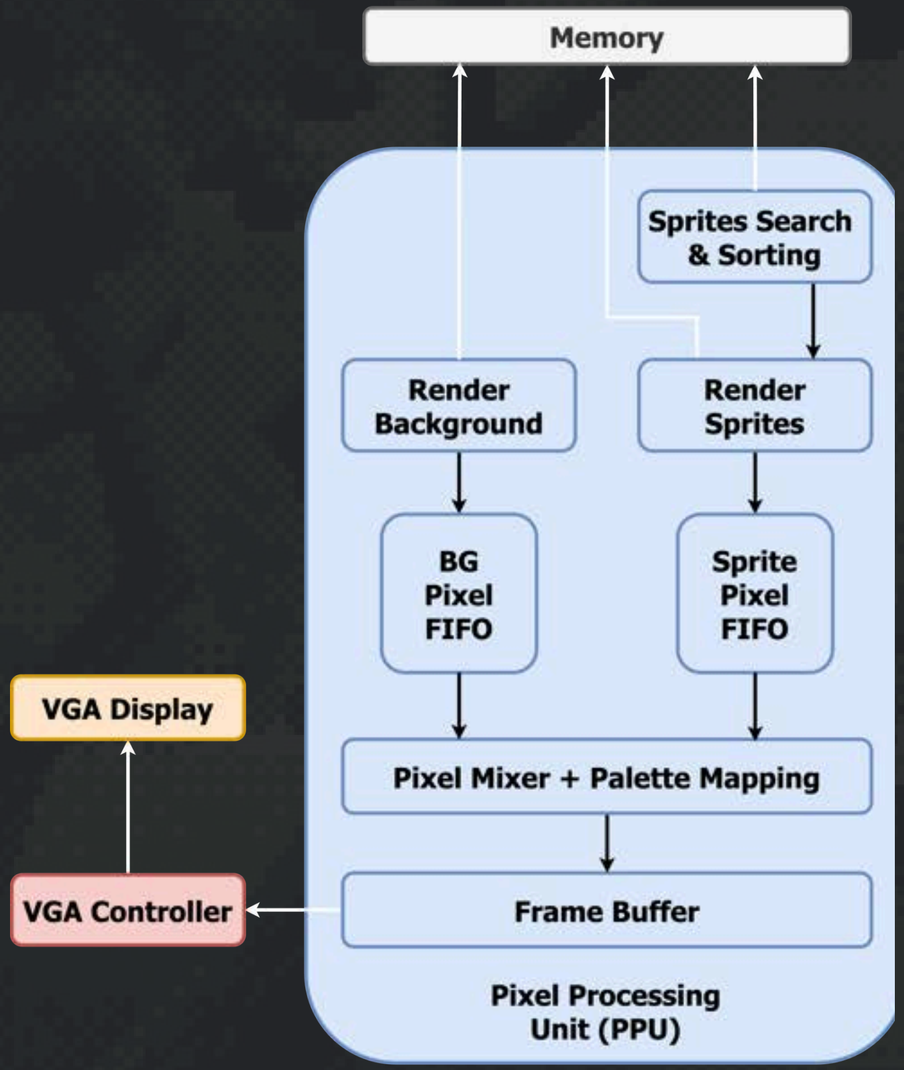


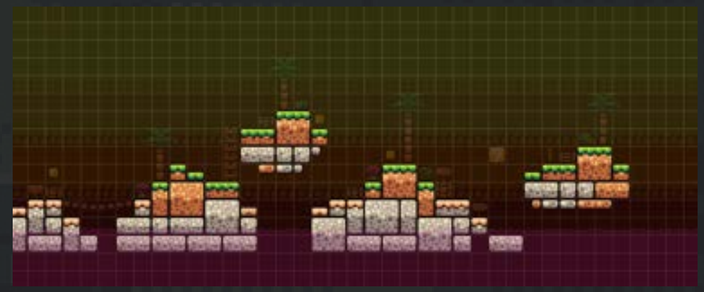
- Intel 8080-like Sharp CPU
- Cycle accurate
- Complex Memory instructions
- Interrupt Handling



- 4 Channels + mixer
- Volume Control
- FPGA CODEC to output sound

Solution Approach - PPU





- Handles backgrounds, windows, and sprites
- Video RAM stores tiles & maps, OAM Table stores Sprite data
- Renders the screen one line at a time
- 4 Modes: Sprite Searching, Drawing, H-blanking, V-blanking
- VGA to display screen

Tileset referenced: https://www.spritefusion.com/tilesets/four-seasons-platformer

Implementation Flow

Testing, Verification, Metrics [1]

Metric / Value	Test Method	Input	Success Output
Clock Speed	Synthesis	FPGA Clock	CPU operates at 4.19MHz
CPU Accuracy	Simulation	Intel-8080 CPU tests	Cycle accurate
Frame Rate	Simulation	Custom Test ROMs	Frame Output Delay < 16.7 ms
Palette Mapping	Simulation	Custom Test ROMs	100% palette mapping accuracy
Frame Rate Consistency	Simulation	Custom Test ROMs	• Each scanline drawn in 456 cycles • Each frame lasts 70224 cycles
Sprite Position Accuracy	Simulation	Custom Test ROMs	• Sprite X/Y coords within ±1 pixel 90% of the time • Sprite priority and layering with > 90% correctness
Qualitative Checks	Custom Script	Simulation Dump	Matches Software Emulator Output with > 80% accuracy

Testing, Verification, Metrics [2]

Metric / Value	Test Method	Input	Success Output
Wave Frequency	Oscilloscope	Set NR13/NR14 or NR33/NR34	Wave frequency matches control register
Wave RAM Accuracy	Oscilloscope / Logic Analyzer	Custom Input to Wave RAM	Bit-perfect Wave RAM (Channel 3)
Pitch Check	Oscilloscope	Set NR10, NR13/NR14	Pitch sweeps change every 7.8ms step
Volume Check	Oscilloscope	Set NR12/NR22, trigger NR14	Volume envelope updates within 1 frame
Audio Output Latency	Simulation	Write NR14, measure output time	Register Write-to-Output Latency < 10ms
Controller Input Lag	Video Recording	Recorded videos of Button presses	Control register reflects correct button press within 30ms

Risk Mitigation

CPU	Continuous implementation & testing; incremental instruction set validation.
Pixel Processing Unit (PPU)	Multi-stage rendering tests; simulation unit testing with assertions & custom testbenches.
Audio Processing Unit (APU)	Verify each channel using oscilloscope & logic analyzer. *Backup: Output audio directly to GPIO pins if the codec fails.*
Memory	Use BRAM for Game Boy memory, SRAM/SDRAM for ROM storage. *Backup: Store everything in BRAM if needed.*
Controller / Input	USB 2.0 NES controllers. *Backup: Custom controller mapped to GPIO pins.*
Integration	Multi-step integration, rigorous system-wide testing, debugging logs for failure analysis.
Misc	Workarounds for missing documentation; reference open-source software emulators.

Schedule

			WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 7.5	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14
			M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F
Documentation															
CPU															
CPU framework	Katherine	100%													
CPU testing framework	Katherine	100%													
cycle accurate cpu framework	Katherine	100%													
Memory ops	Katherine	65%													
Arithmatic and control flow ops	Katherine	0%													
Misc ops	Katherine	0%													
Interupts	Katherine	0%													
Full CPU debugging	Katherine	0%													
Slack	Katherine	0%													
PPU															
Datapath Design	Bharathi	100%													
DMG PPU RTL Implementation	Bharathi	100%													
DMG PPU Single Frame Testing	Bharathi	80%													
DMG Multi-frame + Interrupt Testing	Bharathi	25%													
Datapath & RTL Updates for CGB	Bharathi	0%													
CBG Single Frame Test	Bharathi	0%													
CGB Test suites + Debugging	Bharathi	0%													
Memory mapping															
Memory Datapath	Ruslana	100%													
Memory Research (into FPGA SRAM, DRAM, Flash)	Ruslana	50%													
Memory RTL Implementation	Ruslana	50%													
Basic Memory RTL Test on FPGA	Ruslana	50%													
APU															
APU Audio Codec Research	Ruslana	50%													
Audio Codec Unit Test	Ruslana	0%													
APU Datapath	Ruslana	0%													
APU RTL Implementation	Ruslana	0%													
APU RTL Test on FPGA	Ruslana	0%													
Controller															
Controller Input USB Research	Ruslana	50%													
Controller Input RTL Test	Ruslana	0%													
Integration															
Memory integration	Katherine/Ruslana	0%													
Integrating PPU	Katherine/Bharathi	0%								PPU/CPU					
Slack	All	0%													
Total Integration	All	0%													
Testing															
Test ROM framework	Katherine	0%													
Run/debug test ROMs	All	0%													
Tetris or Dr. Mario ROM	All	0%													
Slack	All	0%													