FP-GAme

FPGA-Based Retro Game Development Console

Team C1: Andrew Spaulding, Joseph Yankel

Presenter: Joseph Yankel

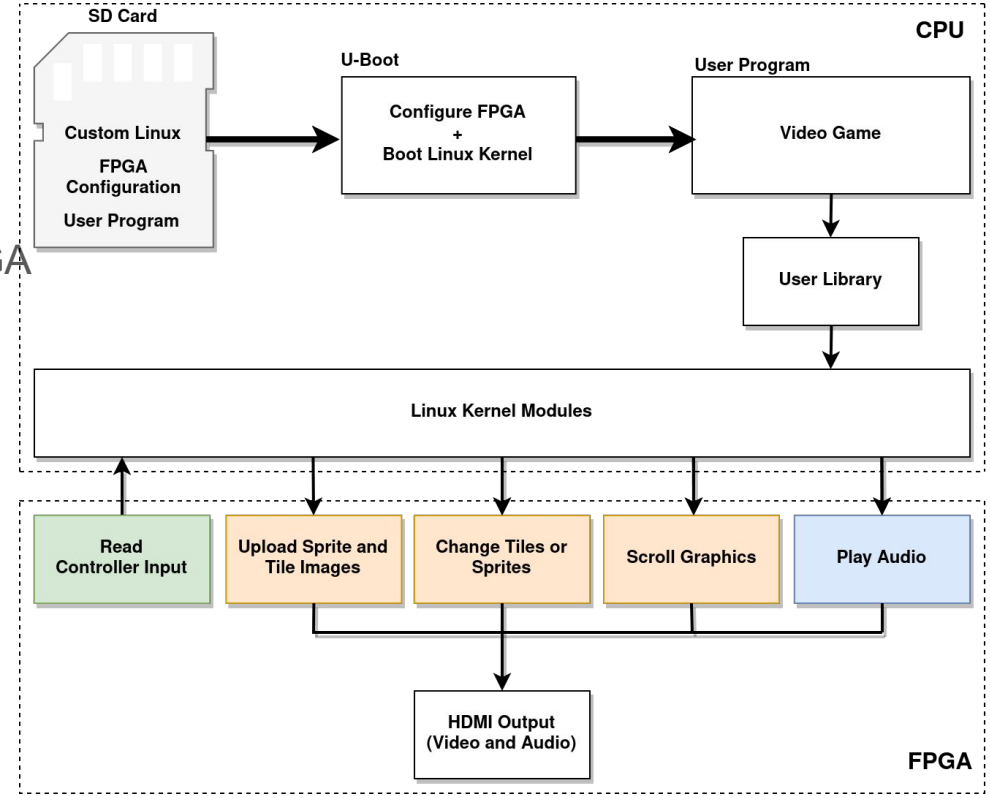
Agenda: Final Presentation + Q&A

Application Area

- Many people are interested in retro console development.
  - Often, the console matters less to hobbyists than the experience
- Software development for retro consoles is inaccessible.
  - Requires knowing very specific hardware details
  - Kits only sold to developers for high prices (~$2000), now collectors items.
- FP-GAme will provide a similar development experience
  - Cost will be kept in the range of $200.
  - Kernel modules and user library will be used to hide unnecessary hardware details.
  - Similar experience, more accessible.

Solution Approach

- User downloads FP-GAme image. Uploads their game to SD Card
- Automatically configures FPGA and runs Linux on boot
- User accesses PPU, APU, Input through Convenient Library
- DE10-Nano FPGA has HDMI for audio/video and GPIO for input.



Complete Solution

- Overall design is as outlined in the previous slide, no major changes to system architecture.

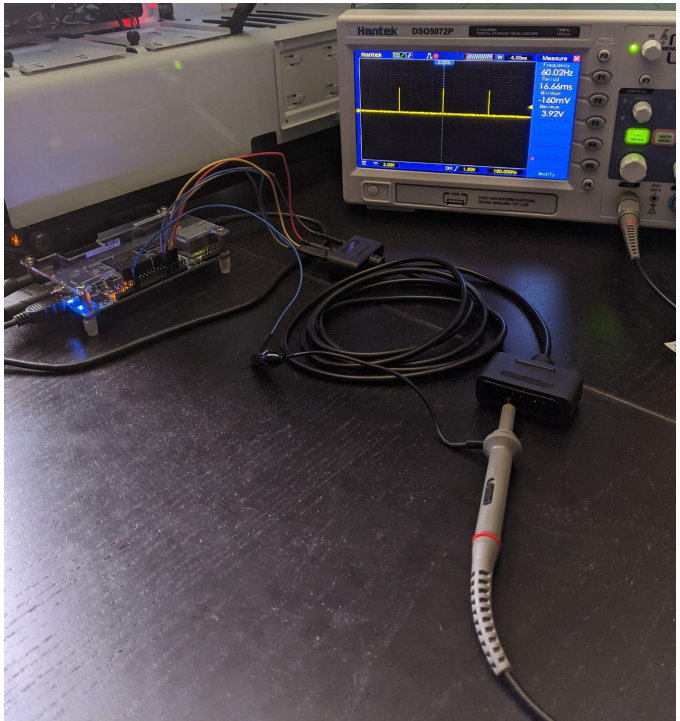
- Public demonstration will include two technical demos
  - Simple tech demo demonstrating scrolling, sprites, layering, and basic audio.
  - NES Console emulation demonstrating more advanced usage.
    - Consistent APU usage.
    - Stressful CPU and memory usage.
    - Graphics accelerated high-level emulation.

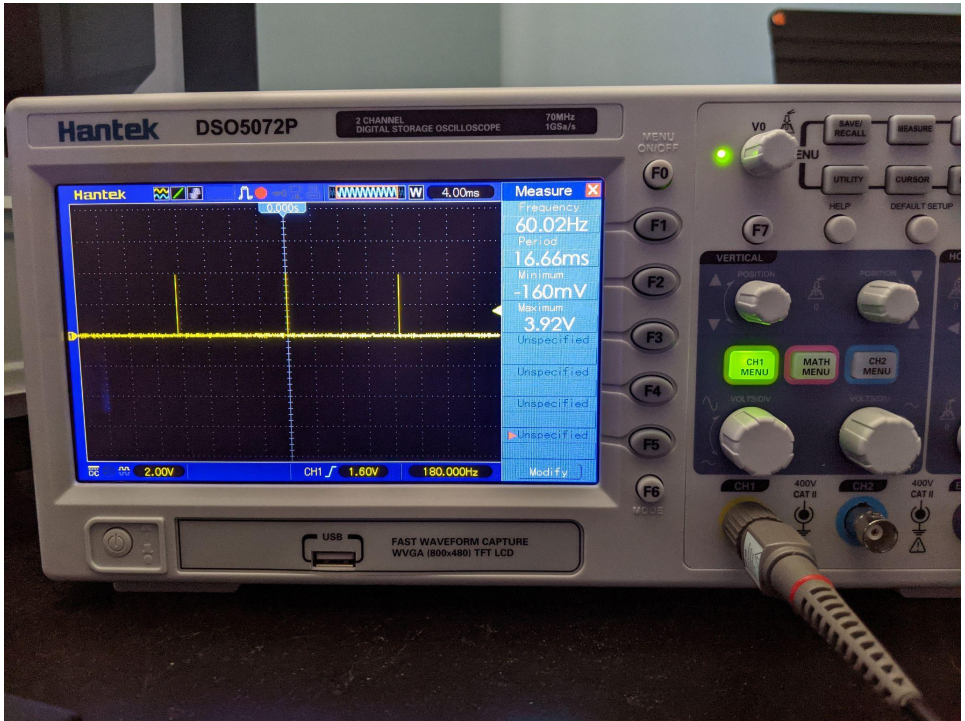
Metrics and Validation

Component	Test	Pass/Fail Condition
APU	Validate audio with 2 tests: 1. HDMI -> AUX cord -> Computer recording. 2. Sampling rate limit alias test	Audio signal undamaged: Pass Output audio does not alias: Pass
PPU	~~Graphics Stress Test~~ (Changed)	N/A
Input	Use oscilloscope to verify 60Hz controller sample rate.	If sample rate ≥ 60Hz: Pass.
System	System Stress Test Playable Tech Demo which must implement: ● Scrolling foreground and background layer ● Sprites ● Input ● Audio	If system is able to run user program without audio/visual glitches: Pass

Results - Input

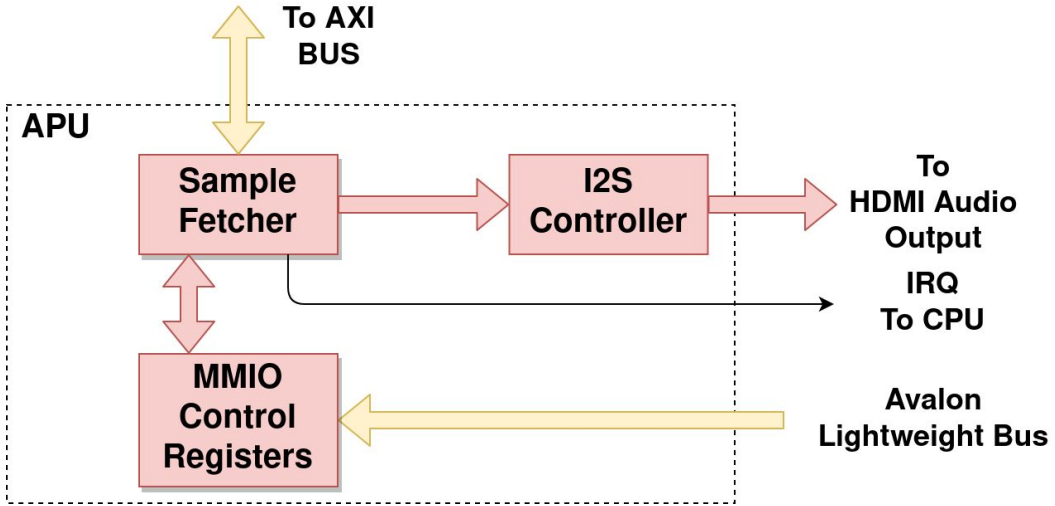
Input confirmed to sample at 60Hz





Design Tradeoffs - APU

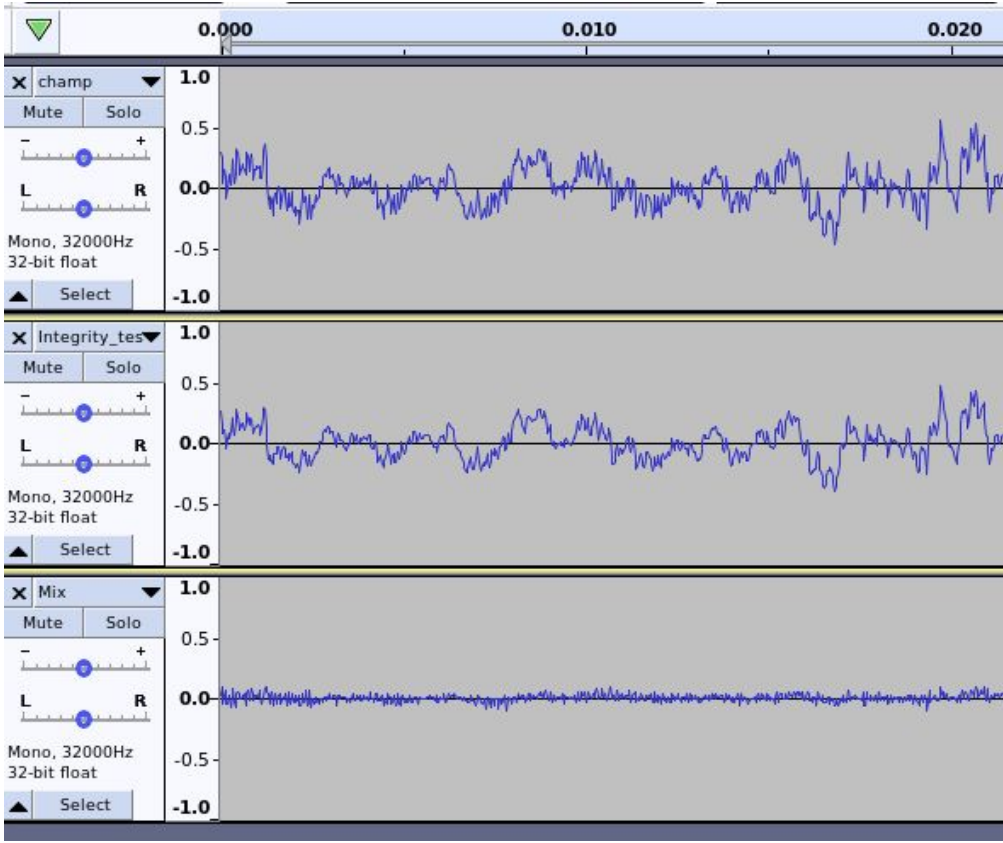
- More flexible latency demands, so no need for internal caching via M10K.
- Original library design called for user-mode callbacks from the interrupt
  - Linux does not support upcalls
  - Signals were used, with libraries wrapping the callback usage, instead.
- Sample buffer size was decreased to 512 bytes
  - More demanding for the CPU, but our CPU can handle it
  - Lower latency from audio generation to playback



Results - APU

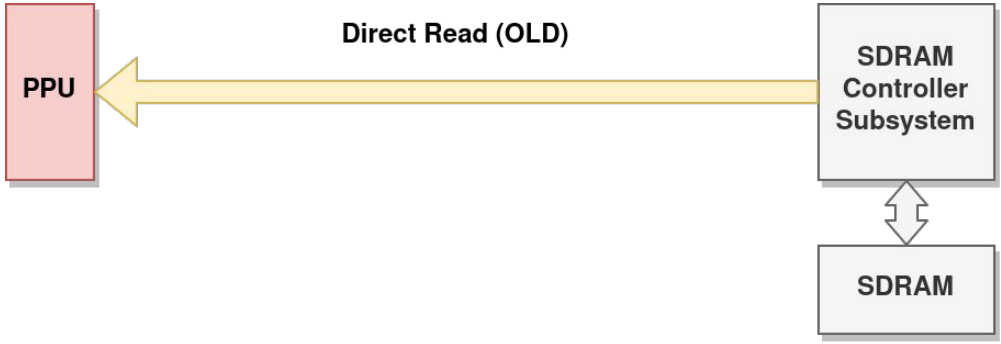
- Signal Integrity Test
  - Original audio compared with output from the APU (by signal subtraction).
  - Signal integrity intact, minus some noise (Test Pass).

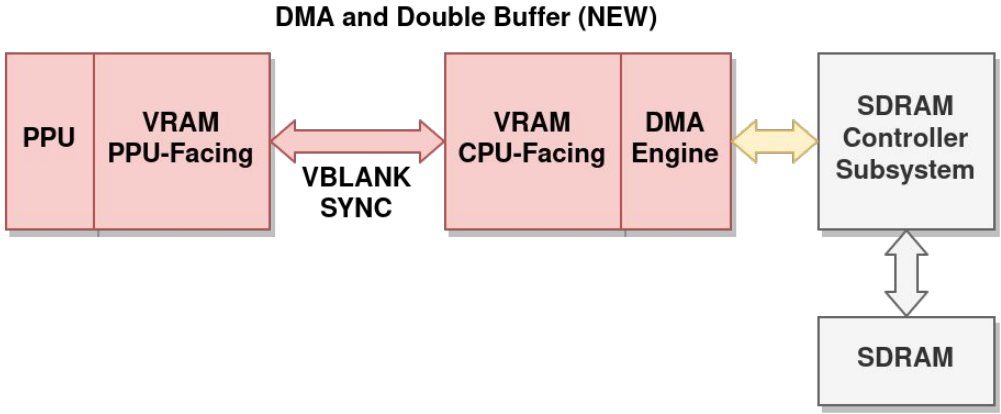
- Sampling Rate Limit Test
  - 16KHz sine wave played
  - Check for aliasing
  - Aliasing did not occur (Test Pass).



Design Tradeoffs - PPU

- Placing upper bound on SDRAM read latency not feasible.
  - No failsafe for long read delays
  - Direct read method complicates logic
- New Design:
  - Fixed-latency M10K VRAM simplifies logic
  - Double Buffered VRAM implements failsafe
- One more tradeoff:
  - 64-bit AXI Bus (low latency)
  - OR 128-bit SDRAM bus (higher throughput)
    - We chose this option.
    - New PPU design no longer requires low read latency.
    - Instead requires high throughput





Results - Tech Demo

Results - System Stress Test

- Places the system under stress comparable to running a full game.

- Tech demo run with an NES game emulation (no video) executing in the background.
  - Observed greater than 50% CPU utilization.
  - Emulator is known to make a large number of memory accesses.
  - Emulation requires computationally intense floating point math to generate sound output.

- Results: No issues observed.
  - No audio glitches.
  - No video issues or missed frames observed.
  - No apparent emulation slowdown.

Updated Gantt Chart

- Many initial tasks took longer than expected.
  - Tile engine was coupled with other tasks.
  - APU, Sprite Engine, and Drivers took longer than expected.
- Initial project tasks finished :)
- New task for extra project documentation
- New task NES Emulator Port
  - Used for public demo

FP-GAme Schedule

Task / Week	2/22	3/1	3/8	3/15	3/22	3/29	4/5	4/12	4/19	4/26	5/3
CPU											
Define System Call Interface		A/J									
Implement PPU Driver									J		
Implement Audio Driver						A					
Implement Controller Driver					A						
Slack											
PPU											
Design PPU Interface		J									
HDMI - Video/Audio Bringup	J										
Timing and PPU FSM Design			J								
Row Buffers, VRAMs, CPU->VRAM Interface								J			
PPU SDRAM DMA								J			
Sprite Engine									A		
Tile Engines						J					
Video Demo							J				
Slack				J	J		A/J				
Audio											
Design APU Interface		A									
Research I2S	A										
PCM Sample Buffer Transmit over I2S						A					
APU -> CPU Interrupt on Buffer Empty						A					
APU -> SDRAM Bug								A			
Input											
Design Controller Interface	A										
Decide on Controller	A										
Implement Input Protocol		A									
Expose to CPU (Bring up FPGA -> CPU Interface)			A								
Slack				A							
Misc											
Implement Tech Demo										J	
Port NES Emulator											A
Project Documentation and Public Repositories											J
Class											
Design Presentation Slides		A/J									
Design Presentation Report			A/J								
Interim Demo							A/J				
Final Presentation										A/J	A/J
Final Video										A/J	A/J
Final Report											A/J