FP-GAme

FPGA-Based Retro Game Development Console

Team C1: Andrew Spaulding, Joseph Yankel

Presenter: Andrew Spaulding

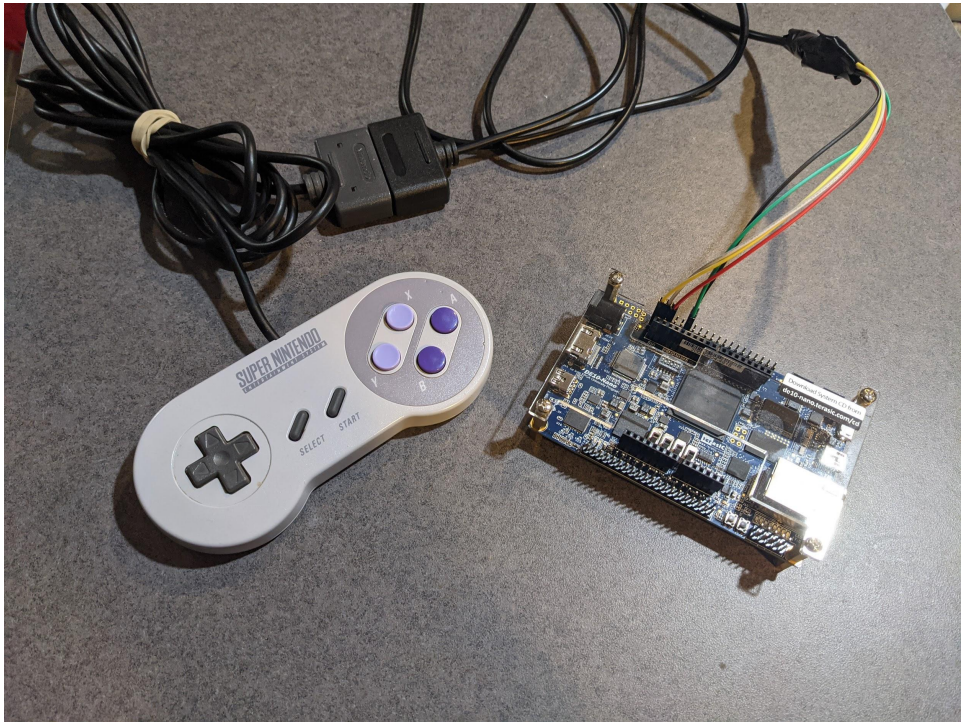
Agenda: Design Review + Q&A

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 will be used to hide unnecessary hardware details.
  - Similar experience, more accessible.

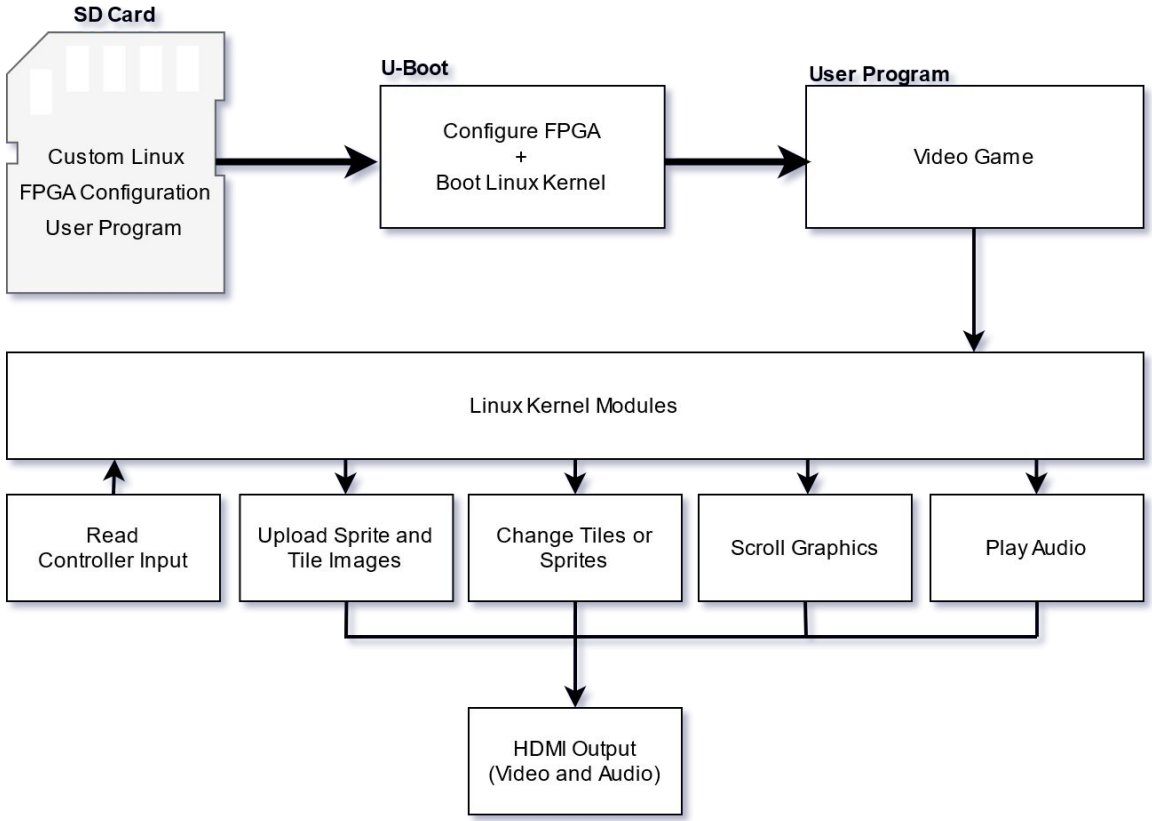
Solution Approach: The DE10-Nano

- Cyclone V with an ARM Core
  - High speed communication
- Input/Output on board
  - GPIO
  - HDMI
- Linux support
  - No need for custom kernel and boot
- Lots of memory
  - 1GB DDR3
  - 553 M10K blocks



System Overview

- User uploads FP-GAme source and game to SD Card
- U-Boot flashes bitstream to FPGA, boots Linux Kernel
- Linux Kernel runs user's game
- User accesses PPU, APU, Input through Kernel Module



System Call Interface

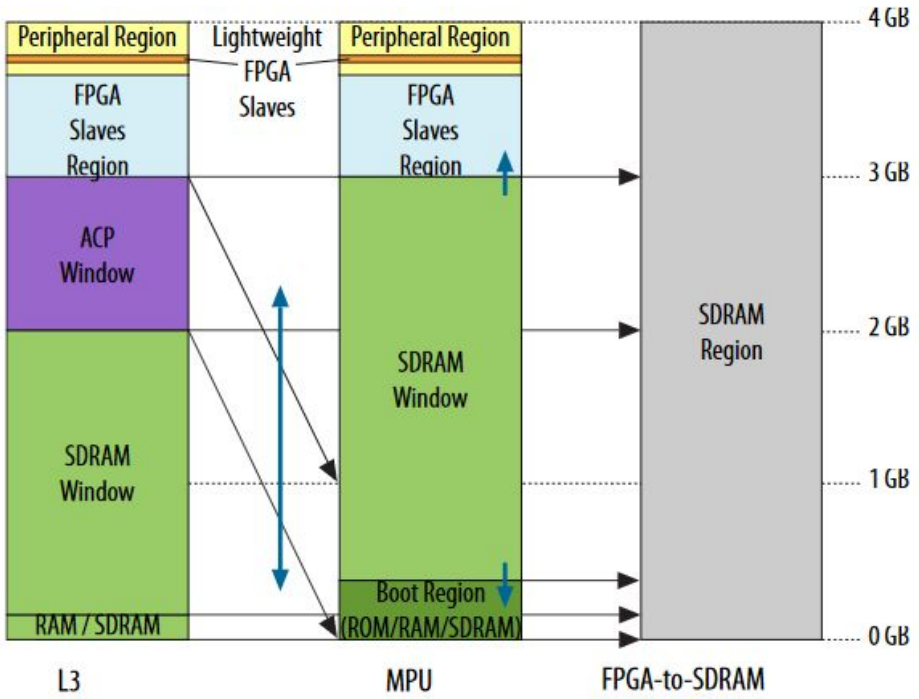
- Controller Calls
  - Getting controller state and masks for parsing the state.

- Audio (APU) Calls
  - Registering a callback for when the buffer empties.
  - Sending a new buffer of samples.

- Graphics (PPU)
  - Rewriting Object Attribute Memory (OAM)
  - Rewriting tile data
  - Changing the scroll value
  - Rewriting the FG/BG Name Table
  - Rewriting palette data

MMIO Interface

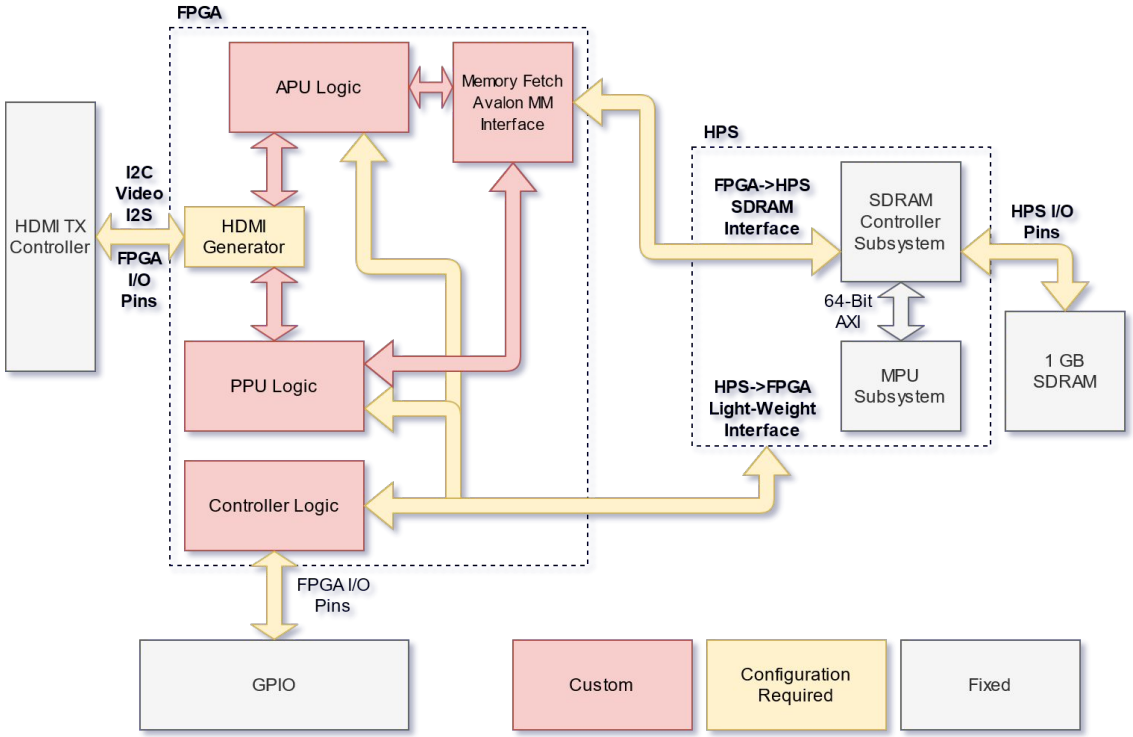
- Controller I/O
  - (RO) Current state
- APU I/O
  - (WO) Next buffer
- PPU I/O
  - (WO) Control
  - (WO) Background color
  - (WO) Base addresses for data
  - (WO) Scroll settings



Source: Cyclone V HPS Technical Reference Manual (Fig. 2-3)

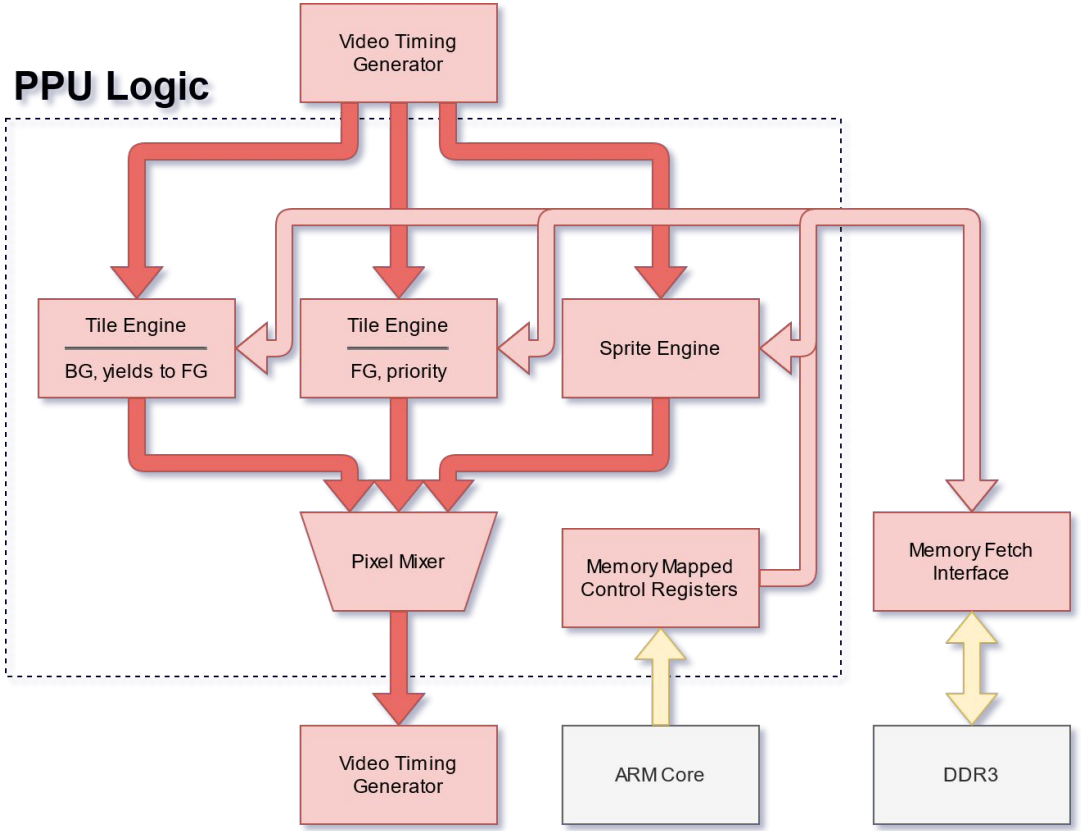
System Interconnect

- Utilizes Intel Avalon® Interface
  - Platform Designer (Qsys) auto-generates buses.
- Hard Processor System (HPS)
  - Accesses APU, PPU, and Input control registers via MMIO.
- FPGA
  - APU and PPU access SDRAM via dedicated SDRAM Interface.
- Memory Fetch Interface
  - Handles and routes DDR3 read requests for APU and PPU.



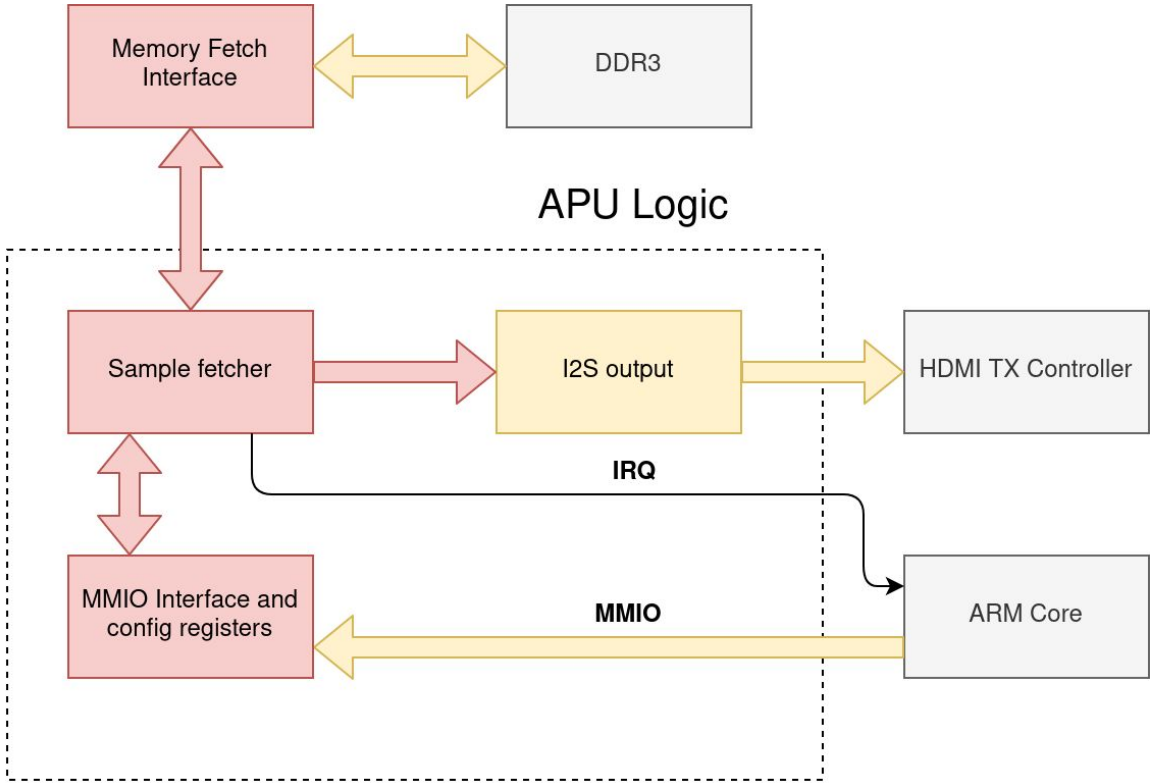
PPU Logic

- Video Timing Generator
  - Sends control/timing signals to Tile Engines and Sprite Engines
- Pixel Mixer
  - Prioritizes one of Sprite, Foreground, or Background pixel outputs.
- Tile Engine
  - Displays tile graphics, fetched from DDR3.
- Sprite Engine
  - Tracks and displays up to 64 sprites in various sizes.



APU Logic

- I2S Output
  - Partially implemented in DE-10 Nano Demo files.
- MMIO Control Register
  - Tells Sample Fetcher address of audio buffer in DDR3.
- Sample Fetcher
  - Times memory fetches to meet continuous I2S requirement.
  - Shifts 8-bit audio samples to 16-bit audio to meet I2S standard.



Metrics and Validation

Component	Test	Pass/Fail Condition
APU	Validate audio with 2 tests: 1. HDMI -> AUX cord to -> Computer recording. 2. Sampling rate limit alias test	Audio signal undamaged: Pass Output audio does not alias: Pass
PPU	Graphics stress test: ● Scroll both directions at 1 tile per frame ● Modify all of OAM every frame.	If test runs without visual glitches: Pass.
Input	Use oscilloscope to verify 60Hz controller sample rate.	If sample rate ≥ 60Hz: Pass.
System	Playable game which must implement: ● Scrolling foreground and background layer ● Sprites ● Input ● Audio	If system is able to run user program without audio/visual glitches: Pass If not fun: Fail

Risk and Mitigation Strategy

- Consistent DDR3 access timing may not be achievable with high traffic.
  - Mitigation 1: Use M10K memory blocks as a cache. Download pixel/audio data long before it is needed. M10K accesses are faster and timing is deterministic.
  - Mitigation 2: Use a fast dual-port M10K memory to hold all of PPU and APU data instead of DDR3.
- User callback for the audio buffer may not be safe for a kernel module.
  - Mitigation: The C standard reserves 2 signals for user programs. Use one of these signals to alert when the audio driver needs a new chunk of samples to be sent.

Updated Gantt Chart for Team C1

FP-GAme Schedule

Task	Week 1 (2/22)	Week 2 (3/1)	Week 3 (3/8)	Week 4 (3/15)	Week 5 (3/22)	Week 6 (3/29)	Week 7 (4/5)	Week 8 (4/12)	Week 9 (4/19)	Week 10 (4/26)	Week 11 (5/3)
CPU											
Define system call interface		A/J									
Implement PPU driver							J				
Implement audio driver					A						
Implement controller driver			A								
Implement full test game							A		A/J		
Slack										A	A
PPU											
Design PPU interface		J									
Construct simple HDMI output	J										
DRAM Tile Fetch			J								
Single-palette, non-scrolling tile engine				J							
Palette-indirect tile engine					J						
Sprites added to tile engine						A					
Scrolling added to tile engine						J					
Slack							A/J			J	J
Audio											
Design APU interface		A									
Research I2S	A										
Add audio buffer for samples to be sent over I2S				A							
Add interrupt for when buffer empties					A						
Slack					A						
Input											
Design controller interface	A										
Decide on controller	A										
Implement input protocol		A									
Expose to CPU (bring up FPGA to CPU interface)			A								
Slack				A							
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	A/J