FP-GAme

A Retro-Game Console DevKit, Accessible to Everyone

ECE Areas:

- Hardware
- Software

Team C1: Andrew Spaulding, Joseph Yankel
Presenter: Joseph Yankel

Programming on Retro Consoles is Inaccessible

- Standard DevKits are not widely available
  - Expensive
  - Out of support
  - Proprietary
- Software emulation doesn’t help
  - Inaccuracy hides hardware details
- Documentation is scarce
  - Confidential

What is the FP-GAme?

- A more accessible DevKit
  - Open documentation
  - Lower cost
  - Simplified development
- Offers similar specs to “16-bit” era consoles
  - 8-bit Pulse Code Modulation (PCM) for audio
  - Pixel Processing Unit (PPU) for graphics

Requirements: Software

- Kernel and software library provided
  - Standard C library
  - Graphics, audio, and timing system calls
- Safe development environment
  - Kernel and drivers protected from user code
  - Internal crash handler provided

Requirements: Hardware

- CPU
  - Modern, C-compiler friendly, CPU with a minimum clock rate of 16MHz
  - Timing, memory protection, kernel/user modes, and external interrupt requests
- PPU
  - 480p video at 60Hz scaled up from 240p
  - Independently scrollable foreground and background tile layers
  - At least 1024 8x8 tiles, with 16-colors per tile
  - Sprite layer of 64 sprites with up to 16 sprites per scanline
- PCM Audio
  - 32KHz audio
  - Delay of less than 45ms

Requirements: Storage

- Comfortably sized program and working memory
  - 512KB of RAM
  - 4MB of program data
- PCM sample buffer
  - 1KB 8-bit sample buffer
- Tile, layer, and metadata VRAM
  - 32KB of tile data
  - 16KB of layering data
  - 2KB of sprite and metadata

Requirements: Communications

- Transfer of samples from CPU to PCM unit.
- Frame-by-frame updates of critical video data
  - Rewrite sprite state
  - Scroll the foreground and background
- Controller support
  - Updates from the controller at least once per frame

Technical Challenges

- High speed communication between the PPU and CPU
  - ~2KB of data transferred for each frame.
- Meeting memory requirements for PPU and CPU
- Obtaining enough storage capacity for programs
  - Must be easily programmable
  - Should be expandable and non-volatile

Solution Approach - DE10-Nano FPGA

- Low cost
  - $135 for board
- Cortex A9 Arm core
  - 1GB of DDR4 memory
  - AXI communication with Cyclone V FPGA
  - 800 MHz clock rate
- Cyclone V FPGA
  - 110K Logic elements
  - Almost 700 KB of RAM
- HDMI output
  - Support for 480p at 60Hz
  - I2S audio support

Testing / Verification / Metrics

Component	Test	Metrics
PPU	Independent scrolling demo using both graphics layers and moving sprites	Visual check for accuracy
PCM Sound	Playing arbitrary sound over the PCM	Listen for audio quality
Controller	Input demo program which displays the currently pressed buttons to the screen	Accurate visual feedback for each button pressed
Software Protection	Simple user program which performs incorrect memory accesses	The program should be unable to perform these actions and must trigger the crash handler

Tasks and Division of Labor

- Andy
  - PCM Audio and drivers
  - Controller hardware and drivers
  - Kernel memory protection and system calls
- Joseph
  - PPU
  - PPU driver
- Both
  - System call interface design
  - Interim demo
  - Final presentation and report

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										
Construct simple kernel which can switch to user mode			A								
Add memory protection to kernel				A							
Implement PPU driver				J							
Implement audio driver					A						
Implement controller driver								A			
Implement system calls									A		
Slack										A	A
PPU											
Design PPU interface	J										
Construct simple HDMI output		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					
Add scrolling foreground above sprites							J				
Implement layering between foreground layer and sprites									J		
Slack										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								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