Pac-Man

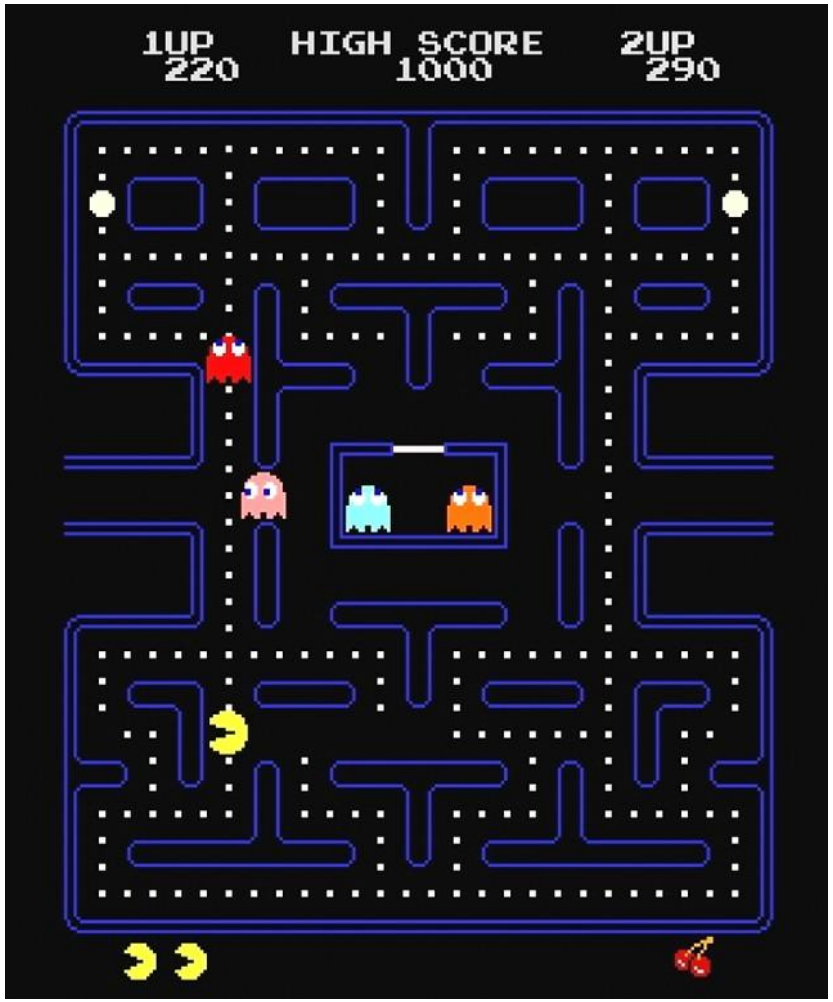
Caiwu Chen, Emma Li, Haoming Ma, Tz-Jie Yu

Prof. Stephen A. Edwards

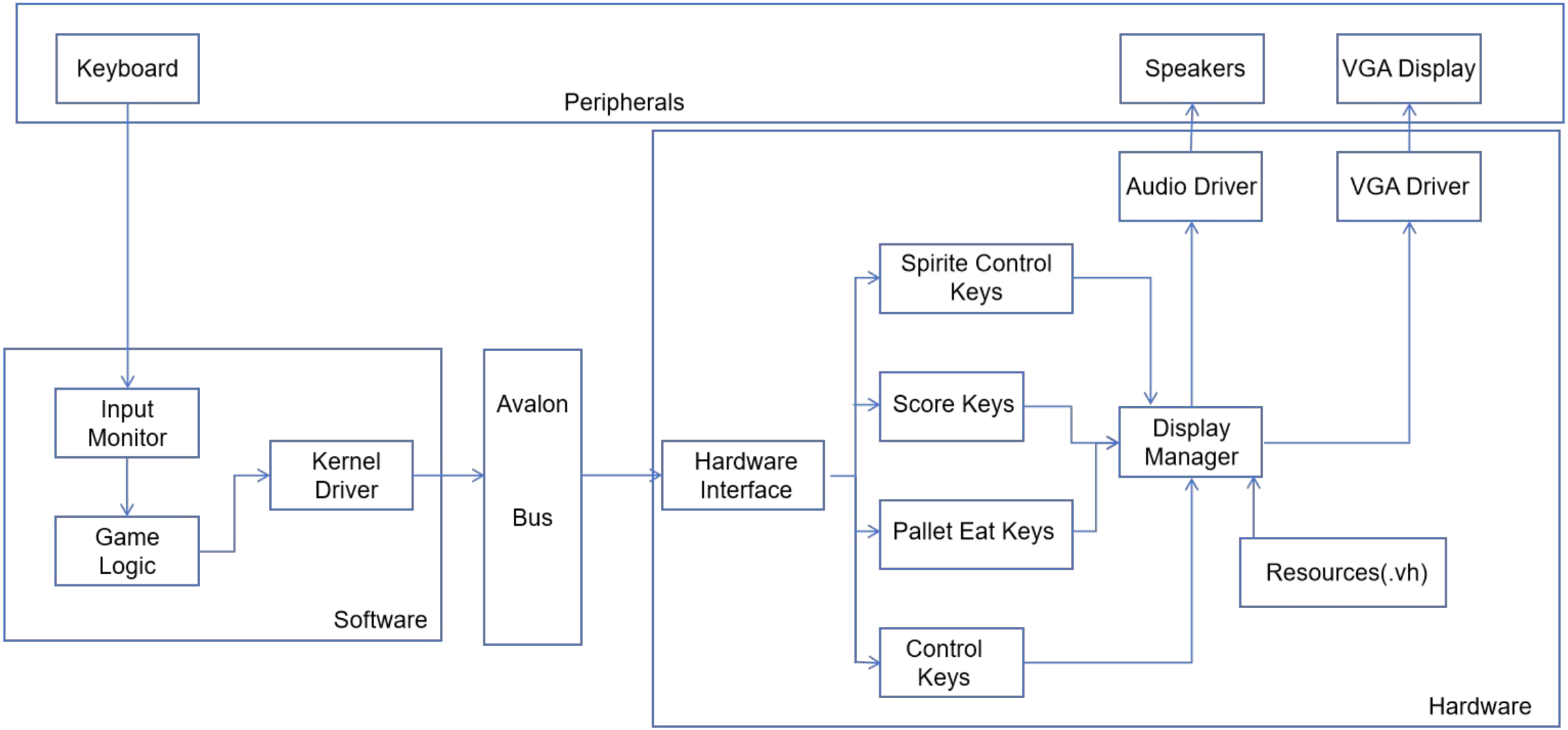
COLUMBIA ENGINEERING
The Fu Foundation School
of Engineering and Applied Science

Background

- Pac-Man is an arcade game developed by the Japanese company Namco in 1980.
- Pac-Man became one of the most iconic games of the 1980s, pioneering character-based gaming.
- Players control a round character navigating a maze, eating pellets while avoiding ghosts.



System Architecture



Hardware System

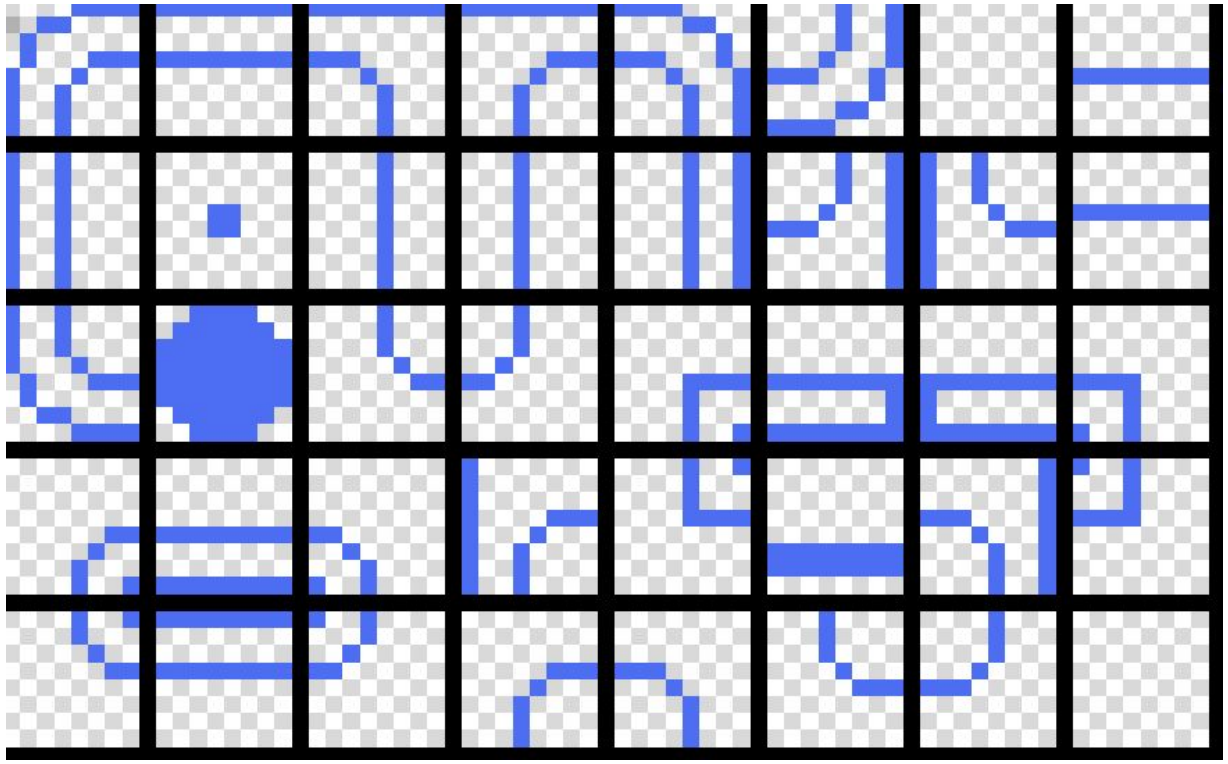
- vga_ball.sv is the top-level hardware module responsible for rendering graphics, handling VGA timing, managing sprite state, and playing audio.
- Communicates with software via memory-mapped registers on the Avalon Bus.
- VGA output uses a 80×60 tile grid (each tile is 8×8 pixels).
- Sprite data (Pac-Man, ghosts, pellets) is sent via 16-bit writes.
- Data updates a reg [7:0] tiles[0:4799] array for tile layout.
- This grid allows fast, direct mapping of visuals to screen locations.
- Enables smooth, real-time updates during gameplay.

Hardware: Sprites

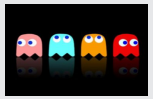
- Sprite Palette: Stores pixel data for individual visual elements.
  - Have 21 different sprites
- Sprite Attribute Interface: Determines each sprite's screen location and state.
  - 1 byte for vertical position
  - 1 byte for horizontal position
  - 1 byte for direction of the sprite

Hardware: Tiles

- Pattern Generator Table: Holds graphical data for reusable static elements, including walls, pellets, and UI text
  - Each pattern occupies lines for 74 distinct tile types.
- Pattern Name Table: Maps each on-screen tile to a pattern's base address for rendering.
  - Contains 4800 entries for a 8×8 tile grid, forming the basis of the entire maze layout.



Resource Budget

Category	Name	Graphics	Size (bits) Width x Height x Channel x Bit-depth	# of images	Total size (bits)
Sprite	Pac-Man		16 x 16 x 3 x 8	16	98304
Sprite	Ghost		16 x 16 x 3 x 8	5	30720
Tile	Borders		8 x 8 x 3 x 8	36	53760
Tile	Pellets		8 x 8 x 3 x 8	2	3072
Tile	Letters		8 x 16 x 3 x 8	26	79872
Tile	Numbers		8 x 16 x 3 x 8	10	30720
Sound	Music				545408

Audio

- Selected Pac-Man theme and death sounds
- Converted .mp3 → .wav (16-bit, mono, 8kHz) → .hex
- Prepared audio data for on-chip memory
- Working on implementing playback on the board using the WM8731 audio CODEC

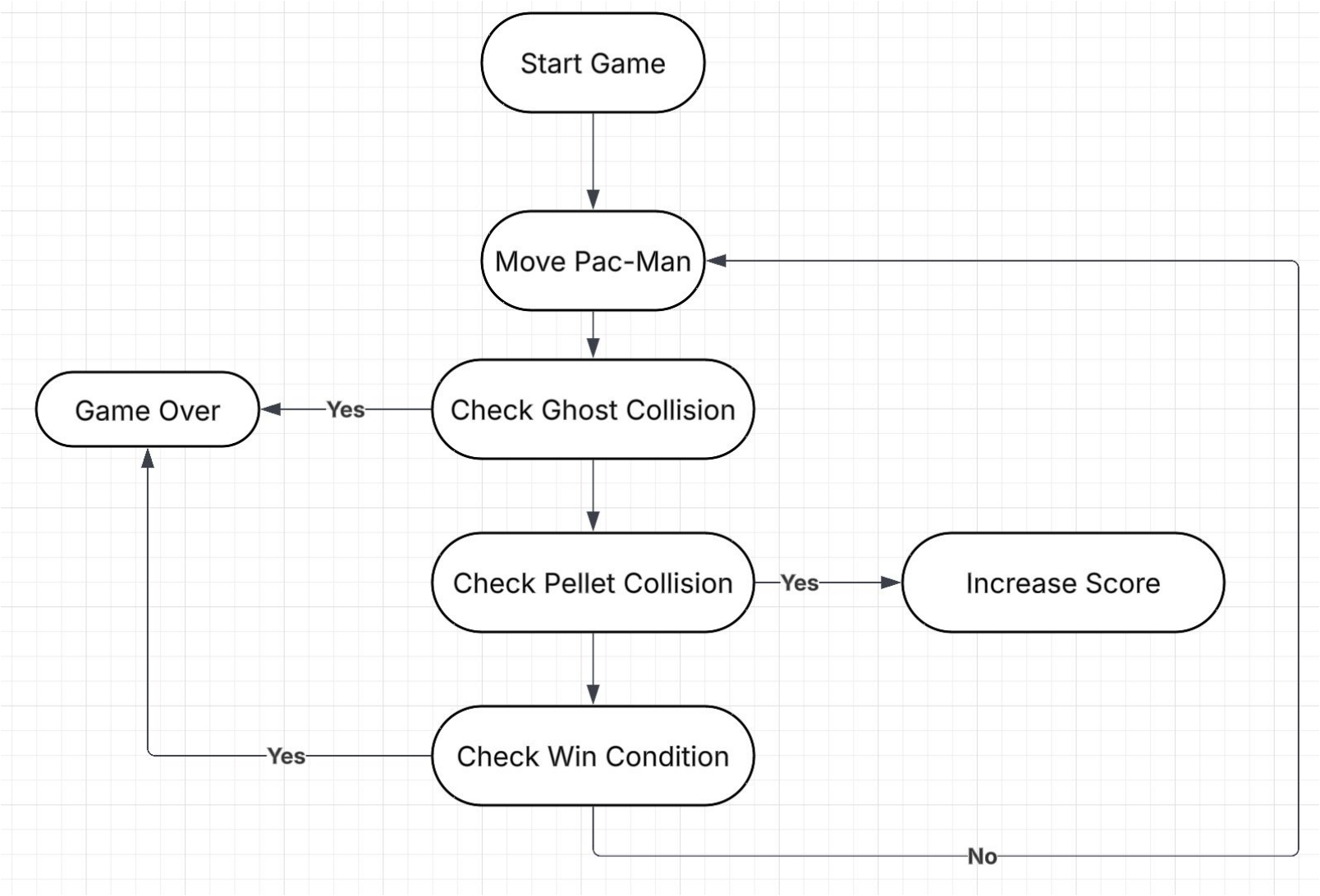
Software System

- Manages game logic: player movement, ghost AI, pellet detection, score calculation, and win/lose conditions.
- Reads controller input to control Pac-Man in real-time.
  - KIWITATA SNES USB Controller
- Updates game state (score, sprite positions, pellet status) every frame.
- Sends updated values to hardware via memory-mapped registers.

Software: User interface

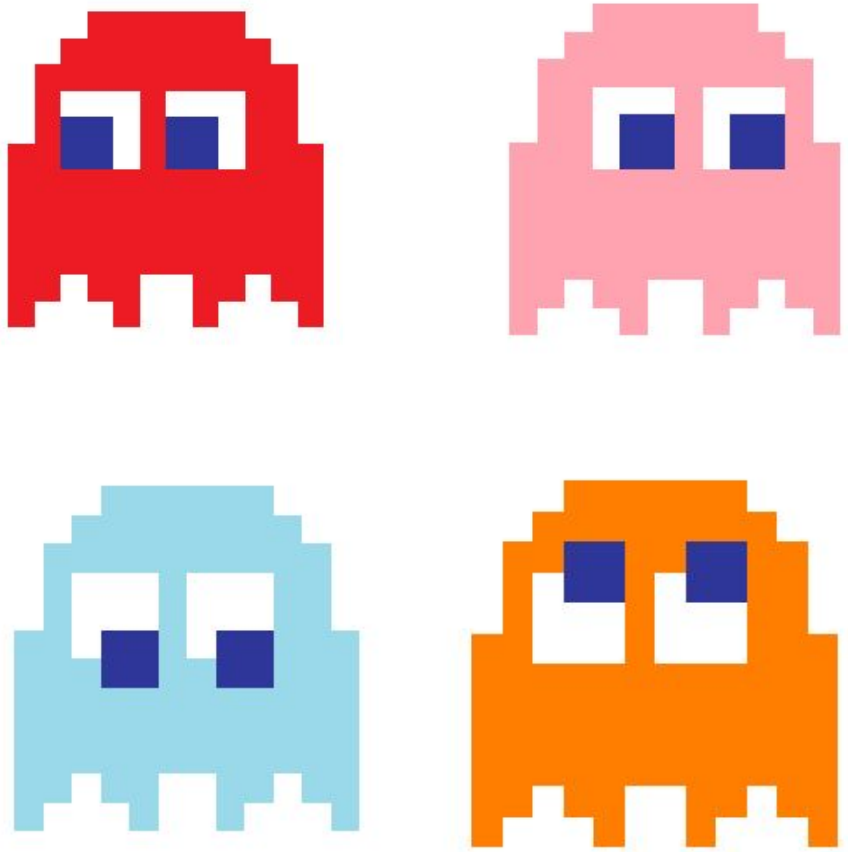
- Uses KIWITATA SNES USB Controller
- Input captured via libusb in C
- Reads directional input (Left, Right, Up, Down)
- Button A start/restart, X pause/resume

Game Logic


Start Game
Move Pac-Man
Check Ghost Collision
Yes
Game Over
Check Pellet Collision
Yes
Increase Score
Check Win Condition
Yes
No

Software: Ghost AI

- Blinky (Red): Directly targets Pac-Man's current position.
- Pinky (Pink): Attempts to ambush Pac-Man by aiming 4 tiles ahead.
- Inky (Blue): Uses both Blinky's and Pac-Man's positions; unpredictable.
- Clyde (Orange): Approaches Pac-Man when far, retreats when close—acts erratically.



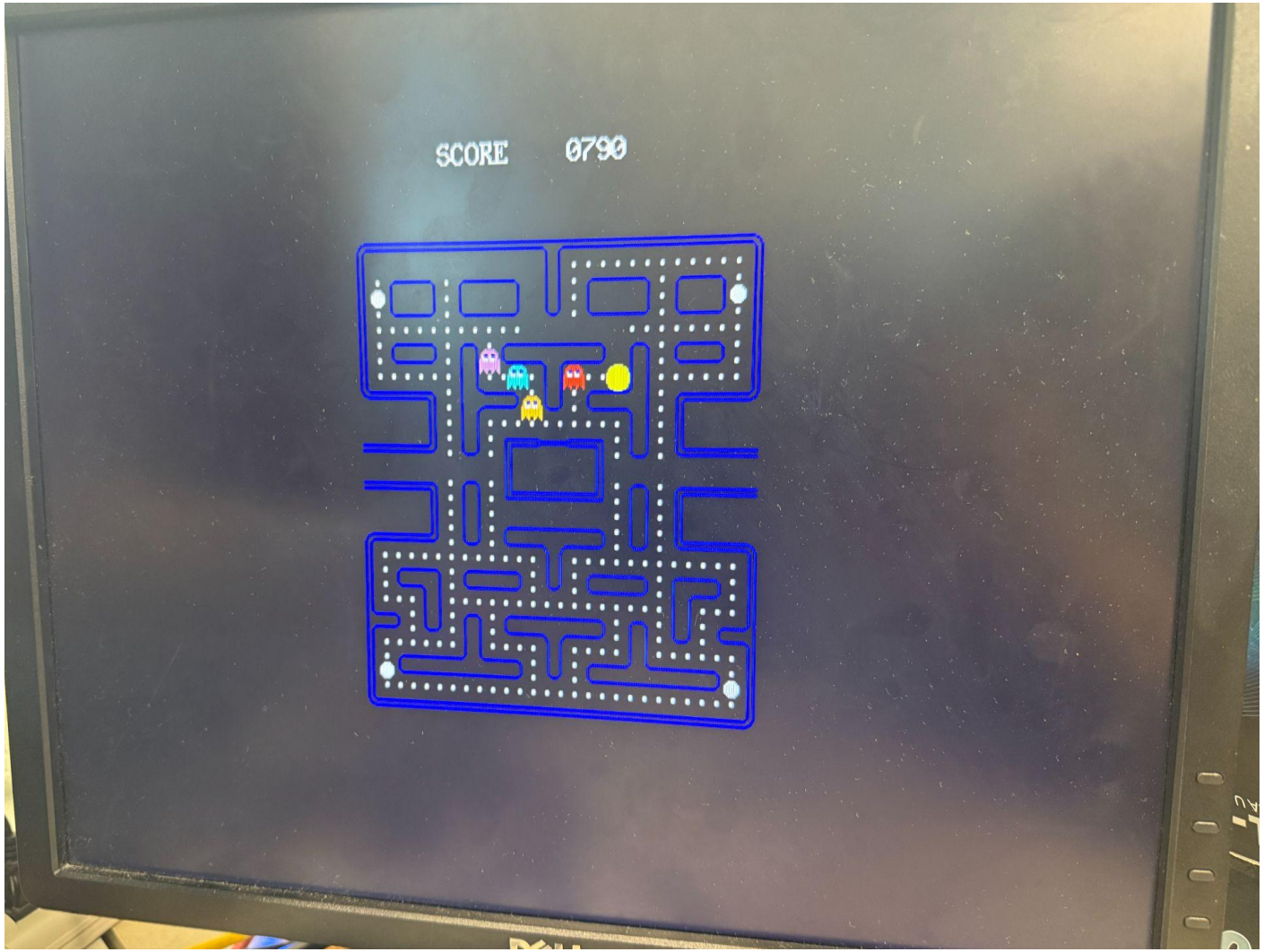
Software and Hardware Interface

- Game logic, video rendering, input detection, and audio control are structured as distinct modules, interfaced explicitly with underlying hardware.
- Player inputs are handled via hardware interrupt.
- The CPU communicates with video processors, sound generators, and input via memory-mapped registers, enabling low-latency hardware access.
- The software configures sprite display attributes per frame, transmitting data to hardware through dedicated attribute and pattern tables for rendering.

Address Map

Base Address	Name	Description
0x0000 - 0x0027	Sprite Descriptors	Descriptors for sprites, including position, orientation. Max 5 sprites (1Pac-man, 4 ghosts), 8B for each.
0x0028 - 0x0029	Score Register	Current score (16 bits)
0x002A	Control Register	Signals for start/reset/pause/display sync etc.
0x002C-0x003F	Reserved Register	Reserved for future use

Screenshot of finished game



Demo