FPGA side scrolling videogame

Telmo Luis Correa Junior
Gameplay concept

• 2D environment
• Side-view camera angle
• Hardware support to multiple possible genres
  – Platformer
  – Shoot ‘em up
  – Pacman
Implementation concept

• Most computation effectuated by software (Assembly game code)
• Hardware support for slow operations
  – Sprites
  – Collision detection
  – Audio (optional)
Implementation overview

• Microprocessor: “beta”
• Graphics processing unit
  – Sprite management
  – VGA signal generation
  – Collision detection
• Audio processing unit (optional)
  – Background music (wavetable synth)
  – Event-triggered sound effects (ROM)
Microprocessor

• Modified beta: opcode used for communicating with other hardware modules
  – Stall the processor for a cycle (behave like a NOP)
  – Send a control signal to other hardware, which has read-access to part of the regfile
Graphics processing unit

• Capabilities:
  – Display up to 32 sprites on the screen
  – Receives high-level commands from CPU about sprite management (create sprite, change sprite type, destroy sprite, pan screen, etc.)
  – Reports sprite overlap as interrupts
Graphics processing unit

Blob manager

Sprite loader

Graphics ROM

Blob

VGA generator

Collision OR

Interrupt generator

Pixel selection tree
Graphics processing unit

• Core of hardware design process
• Main modules:
  – Blob manager: reads commands from CPU, sends commands down a control bus to the blobs
  – Blob (x32): responsible for displaying and animating one instance of a sprite, or a collision block
  – Sprite loader: accesses the ROM, one sprite at a time
  – Pixel selection tree: selects which pixel output from a blob should be displayed, detects collisions
  – Interrupt generator: sends an interrupt request to the CPU if the collision status of a blob was changed
  – VGA generator: sends coordinates of a pixel to the blob manager, receives it from the pixel selection tree, and produces the VGA signal
Blob manager

- Controls all blobs
- Responsible for interpreting instructions from CPU registers
- Translates screen coordinates received from VGA generator into absolute coordinates for blobs
Blob manager

- **Inputs:** clock, CPU, VGA
  - From CPU
    - Hardware output control signal
    - Registers 0 through 3
  - From VGA generator
    - Pixel coordinates

- **Output:** blob control bus, pixel selector
  - To all blobs
    - Control signals
  - To pixel selector
    - Background color
Blob

• Little FSM
  – S_NONE: blob does not represent an entity
  – S_SPRITE: blob represents an instance of a sprite
  – S_PLATFORM: blob represents an invisible rectangle, that might be collided with
  – S_TILED_SPRITE: blob represents a rectangle filled with copies of a same sprite
Blob

• **Internal state**
  – The sprite itself
  – X1, Y1 coordinates
  – X2, Y2 coordinates (if platform or tiled sprite)
  – Sprite type
  – Animation step
  – Animated sprite
  – Collidable sprite (clip bit)
  – Enemy sprite (enemy bit)
  – Sprite layer
Blob

• Input:
  – blob manager control bus
  – sprite loader

• Output
  – To pixel selection tree
    • Pixel
    • Blob ID
    • Collision control info (layer, clip bit, enemy bit)
  – To sprite loader
    • Sprite request
Sprites loader

• Accesses the ROM
• Receives requests from all blobs (serial scanning)
• Copies requested sprite into target blob’s memory
Pixel selection tree

- Input: all blobs’ output, background pixel color
- Output:
  - To VGA generator:
    - pixel
  - To Interrupt generator:
    - 32x collision info (32x 2 bits)
      - Sprite collides with non-enemy sprite
      - Sprite collides with enemy sprite
Audio processing unit (optional)