Phoenix: The Reboot

A System Design Project Created By:
Vaishnavi Murthy, vm2591
Ignacio Ramirez, ir2331
Brianna Williams, bjw2135
Phoenix

- Space-themed “slide and shoot” arcade game
- Game developed by Taito and Amstar Electronics in early 1980s
- Our original goal was to be able to implement one level of the game
Hardware
Hardware

- Hardware
  - Sprite and Tile generation based off of the TMS9918 video processor
  - Tile and Sprite modules in System Verilog code with generator table, name table, and color table
  - Priority encoder in the top level hardware module to decide whether sprites or tiles display on that
Background

- Created with Tile generator module
- Various tiles and color patterns referenced for different types of stars, letters, numbers
Sprites

Just like the TMS9918 document, we use this flow to build our sprites for display
The Sprite Module

Separate module that works to generate the correct sprite colors if they are assigned to a certain position on the screen

- Table to hold the sprite positions
- Table to keep sprite patterns
- Table to hold the possible colors needed for design
The VGA Display

From the output of the tile and sprite module, we use a priority encoder to decide which of the returned colors will take precedence on the screen.
Controlling through Software

- Use methods from software to control the sprites and score on the screen. 5 inputs:
  - Sprite_change, sprite_num, new_x, new_y, new_name, new_tag
- Use a flip flop to both read the data coming from software and write the resulting data back to the sprite_att_table
- Data is being passed back and forth with the help of the Avalon bus

```verilog
always_ff @(posedge clk)
  if (reset) begin
    sprite_change <= 8'b0;
    sprite_name <= 8'b0;
    new_x <= 8'b0;
    new_y <= 8'b0;
    new_name <= 8'b0;
    new_tag <= 8'b0;
  end else if (chipselect && write)
    case (address)
      3'h0 : sprite_change <= writedata;
      3'h1 : sprite_name <= writedata;
      3'h2 : new_x <= writedata;
      3'h3 : new_y <= writedata;
      3'h4 : new_name <= writedata;
      3'h5 : new_tag <= writedata;
    endcase
```
Software
Game Screen
Game Logic

- Different threads to handle sprite movements
- Birds move randomly - shoot when ship is directly below
- One thread to handle joystick input and move ship left/right
- Function to calculate collisions
- Sprites explode when hit - keep track of score & lives
- TIME_CONSTANT determines speed of the game
Handling collisions

- Checks if bullet sprite coordinates are within ship and/or bird sprite coordinates
- Return the name of the sprite that is hit
- One thread per bullet continuously calls this function
Score Manipulation

- Increment score whenever a collision occurs
- Changed from main thread
Live Demo