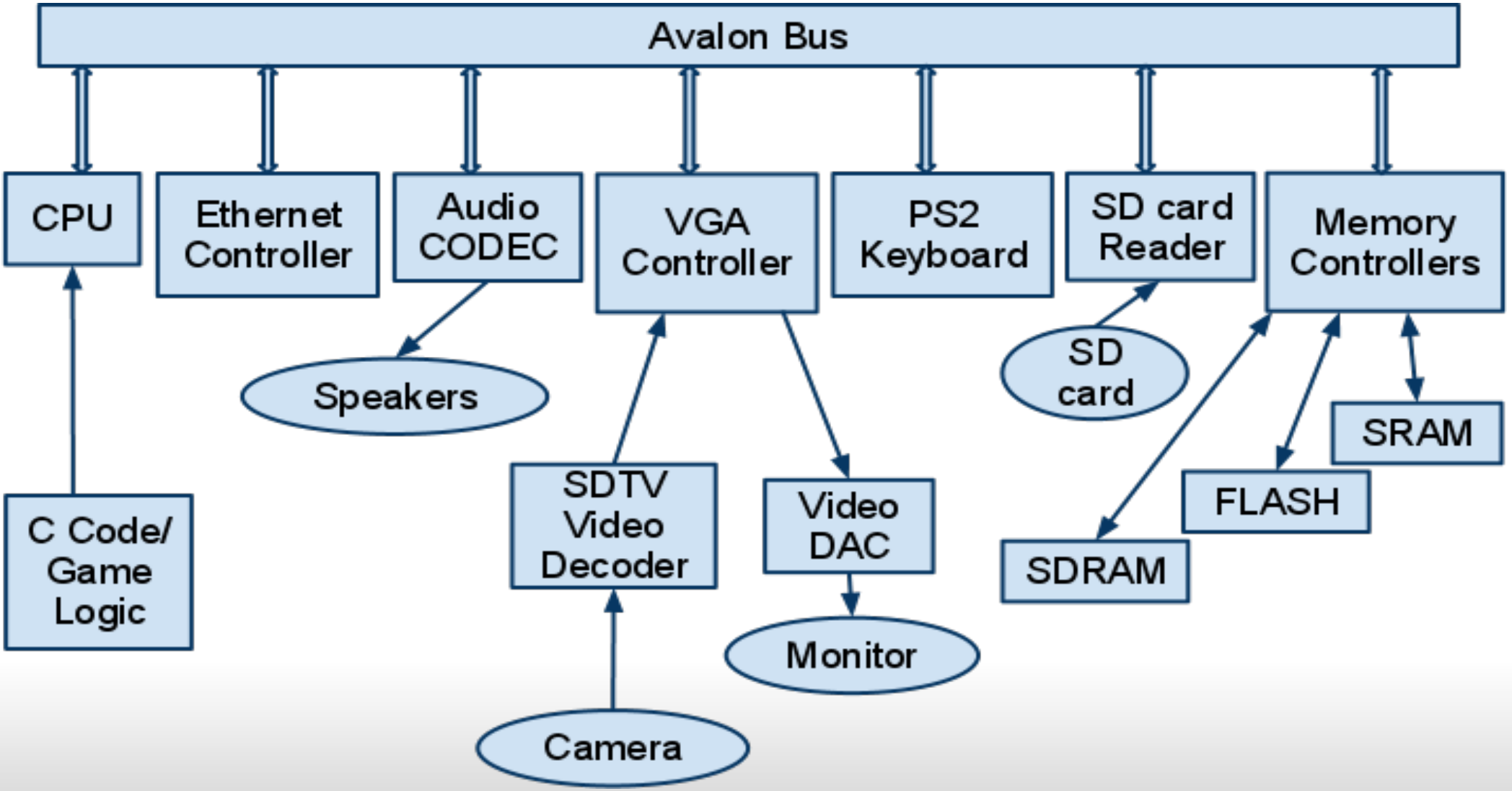
SAGa

Sprite Animated Game

Block Diagram



VGA controller

At each new line on the screen:

- Cycle 0: check which sprite/s are in that line.
- Cycle 1-128: copy the corresponding row from each sprite to local buffers.
- Beyond the cycle 144:

1. Compare all the pixel from each sprites (overlapping).
2. Shift the registers.

Game Logic

- Elements
  - Generate Random Terrain
    - Random curvature with bounds
  - Place worms randomly
    - Over entire game board length
  - Animations
    - Organized by frame
  - Game States
    - switch statement to control game mode
  - Game Parameters
    - Health, active weapon, choosing next worm, etc.
  - Multiplayer
    - Send terrain / game states / keys

Game Logic Animations

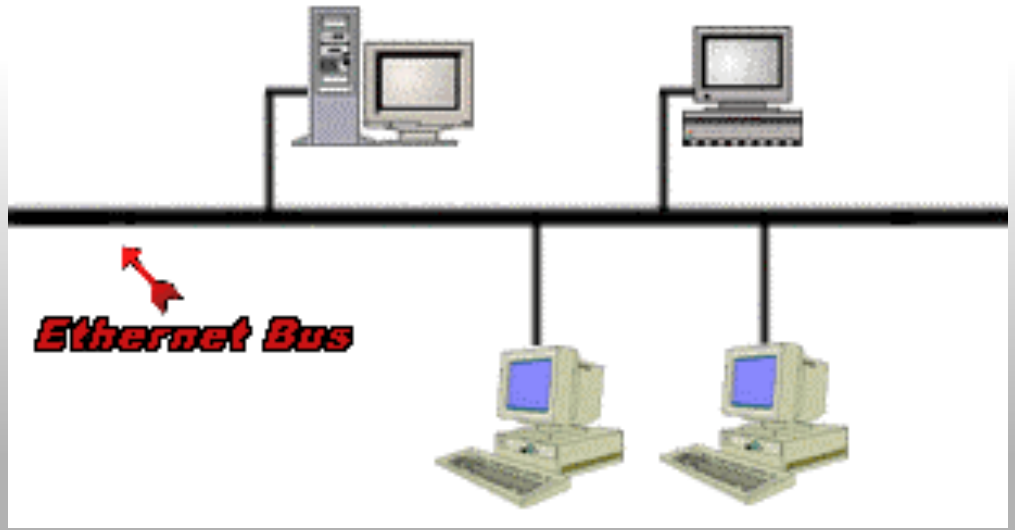
- Problem:
  - The game requires multiple animations to run simultaneously
  - Different animations have different speeds
- Solution
  - All game changes are organized by a master frame rate
  - Each animation divides the master frame rate

Networking

DM9000A Controller was temperamental!

Required using Verilog for top level entity and including epcs controller.

Packets sent on keypress for movement, weapon changes, firing and turn changing as well as for synchronization.



Networking - Synchronization

Delete key triggers synchronization packets.

First synchronization packet sends half of the terrain data.

Second synchronization packet sends the remaining terrain data as well as worm locations.

Due to packet loss in-game synchronization is occasionally required.

The machine that registers the synchronization keypress becomes the master.

Lessons Learned

1. Hardware and software integration can be quite difficult, especially with multiple peripherals.
2. Displaying "realistic" animations efficiently requires optimizations to the data structures storing the data as well as methods for reading and writing said data.
3. Memory limits are very real especially when the number of sprites and peripherals grows.

Questions?