Pac-man

Rated E for Everyone

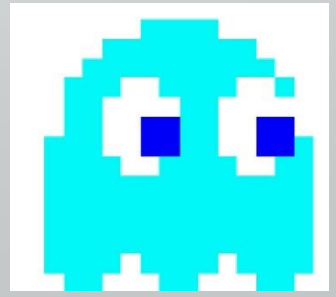
By: Andrew Carle, Peter Isho, Zachary Platte, Rohit Timmagi

Introduction

- This is a two player game that shares many similarities with the original pac-man game
- The goal of player one is to collect the pac-bits and avoid the ghosts
- Player one uses the keyboard and player two uses the FPGA buttons
- Power boosts are scattered throughout the game to assist either player
- Player one gets three lives and the game is over after they have collected all the bits or lost all of their lives
- The game itself is displayed on the PC via a VGA cable

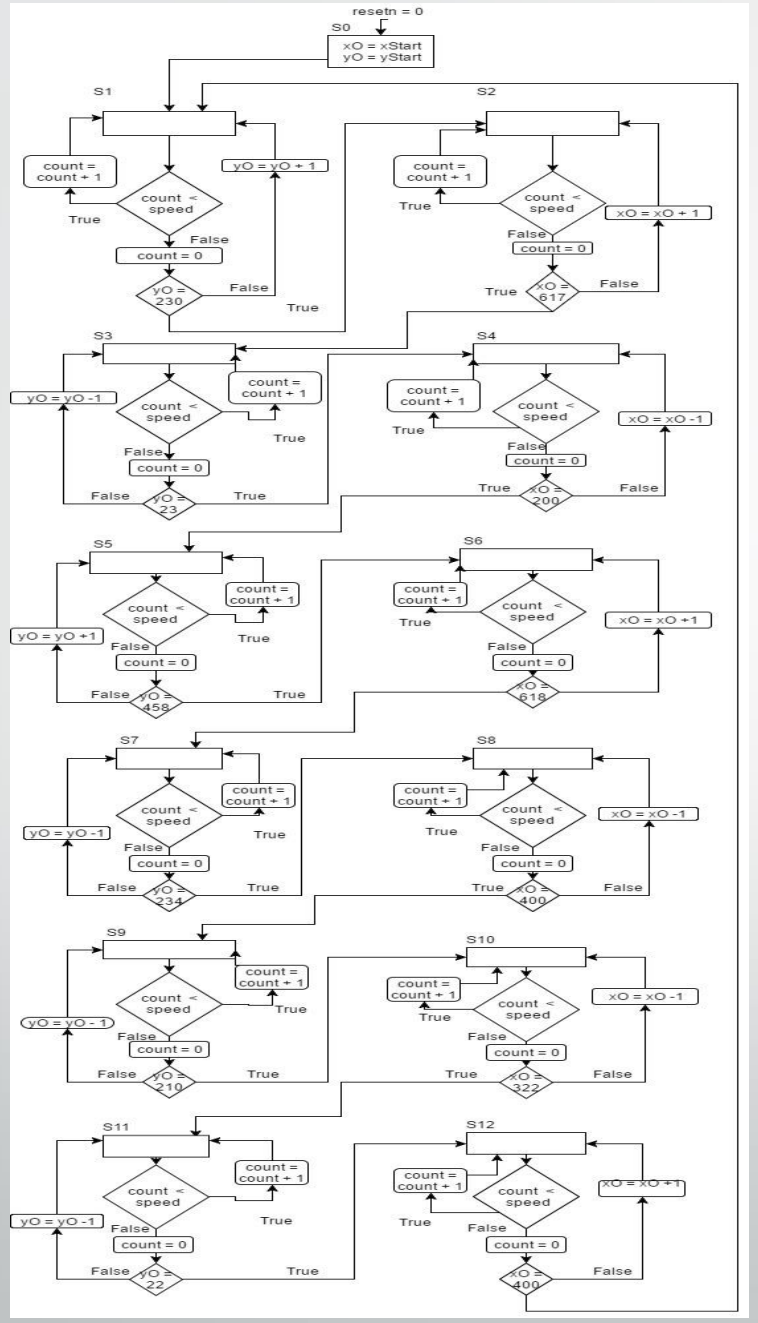
Movement Control

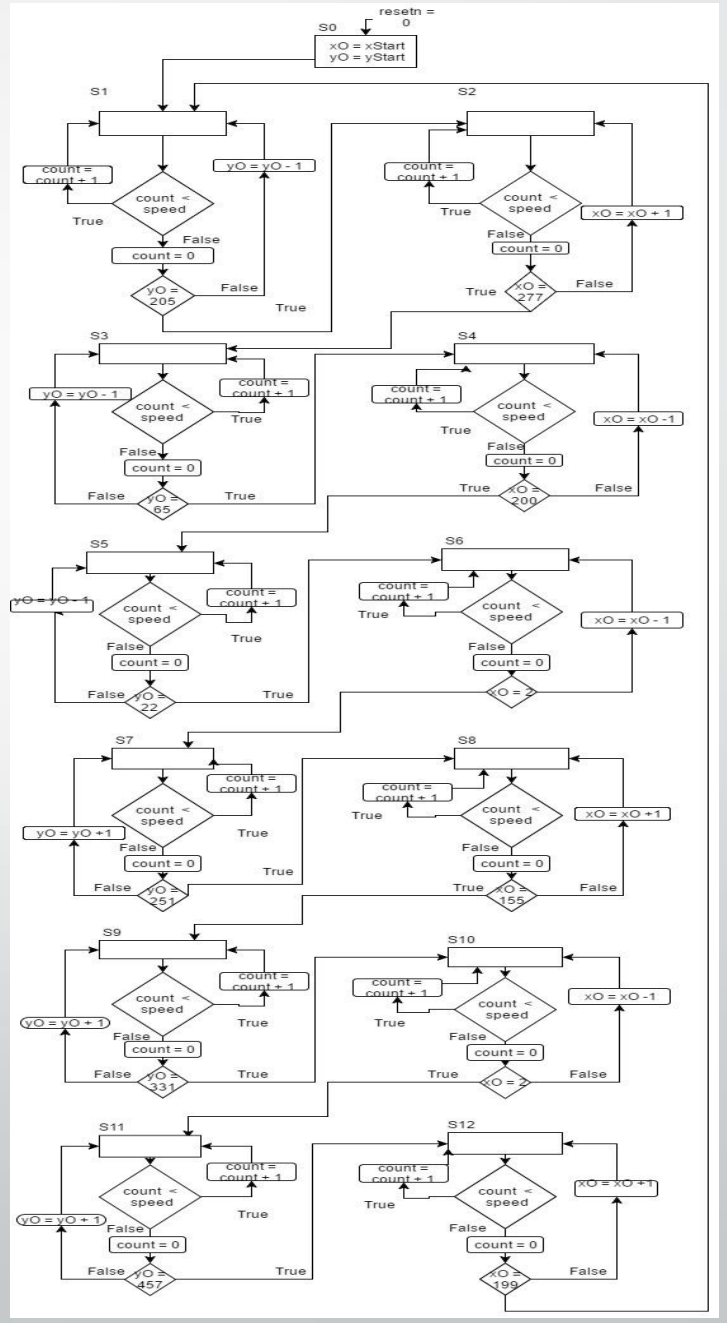
- Controllable characters are moved based on inputs from the FPGA and keyboard
- The speed is based of off a signal called count, without this signal he would move too fast
- Ghosts are controlled using Finite State Machines



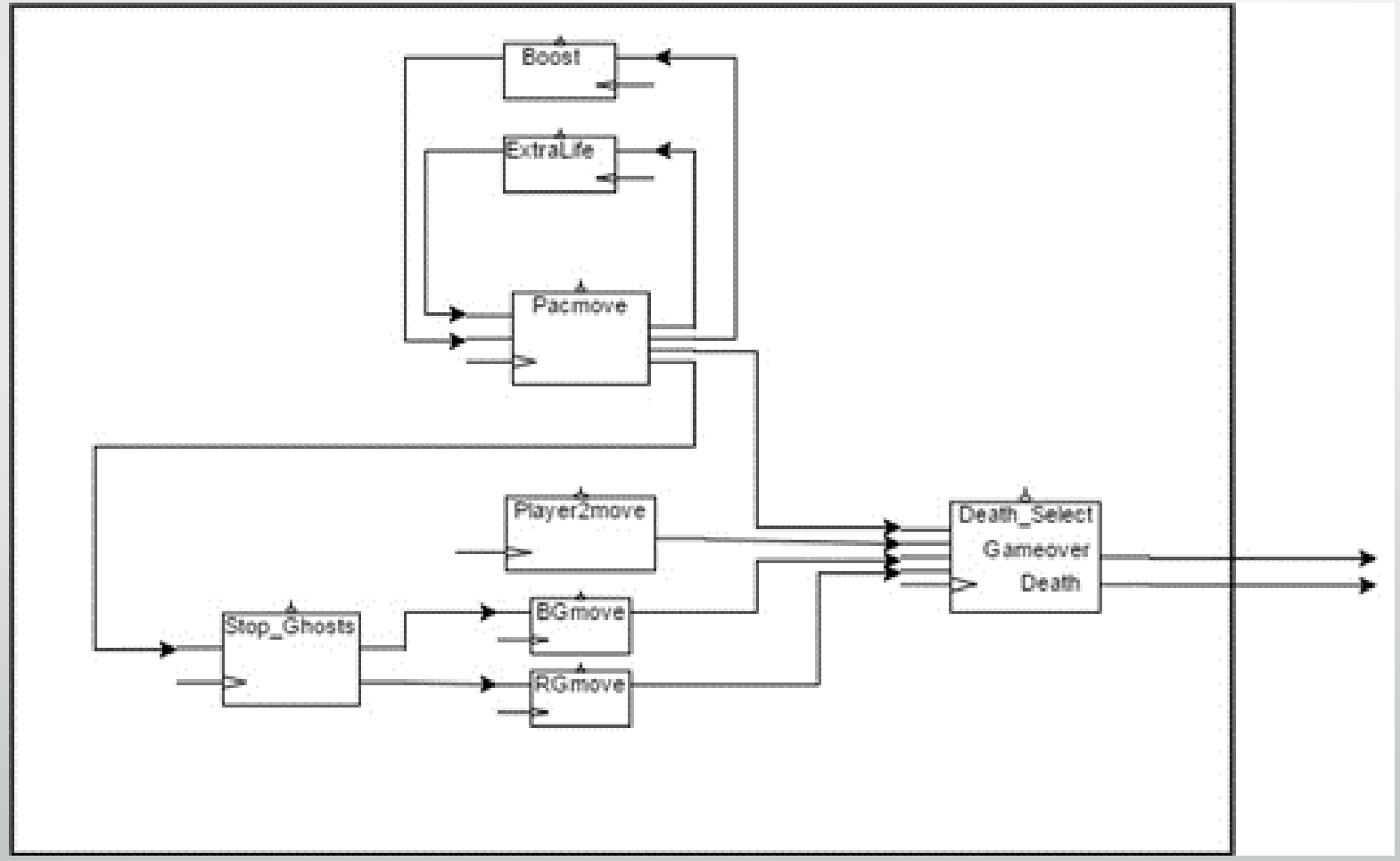
4 Movement Components

- Pacman - Can't go through walls
- Ghost 1 – State Machine with 12 states
- Ghost 2 - State Machine with 12 states
- Player 2 – Can't go through walls
- Each component is internally different but they all output a X and Y coordinate


resetn = 0
S0
xO = xStart
yO = yStart
S1
S2
count = count + 1
count < speed
True
False
count = 0
yO = yO + 1
yO = 230
xO = xO + 1
xO = 617
S3
S4
yO = yO -1
xO = xO -1
yO = 23
xO = 200
S5
S6
yO = yO +1
xO = xO +1
yO = 458
xO = 618
S7
S8
yO = yO -1
xO = xO -1
yO = 234
xO = 400
S9
S10
yO = yO - 1
xO = xO -1
yO = 210
xO = 322
S11
S12
yO = yO -1
xO = xO +1
yO = 22
xO = 400

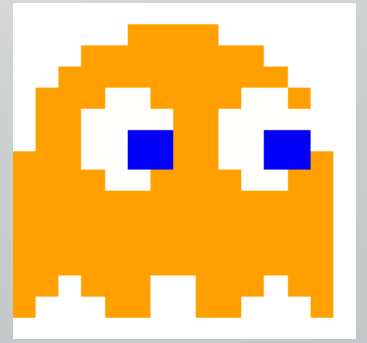

resetn = 0
S0
xO = xStart
yO = yStart
S1
S2
count = count + 1
count < speed
True
False
count = 0
yO = yO - 1
yO = 205
xO = xO + 1
xO = 277
S3
S4
yO = yO - 1
xO = xO - 1
yO = 65
xO = 200
S5
S6
yO = yO - 1
xO = xO - 1
yO = 22
xO = 2
S7
S8
yO = yO +1
xO = xO +1
yO = 251
xO = 155
S9
S10
yO = yO + 1
xO = xO -1
yO = 331
xO = 2
S11
S12
yO = yO + 1
xO = xO +1
yO = 457
xO = 199

Movement Data Path

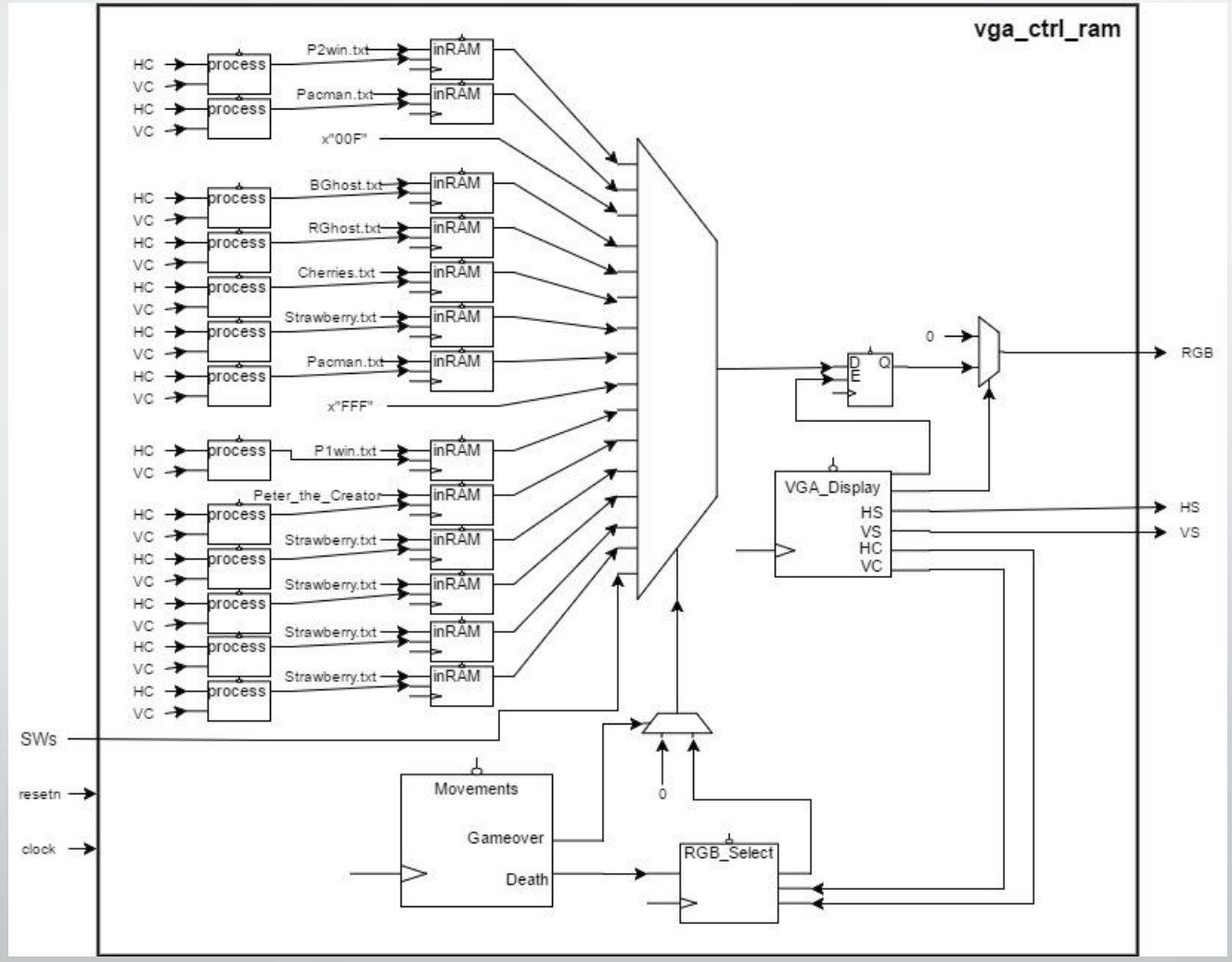


Display Control

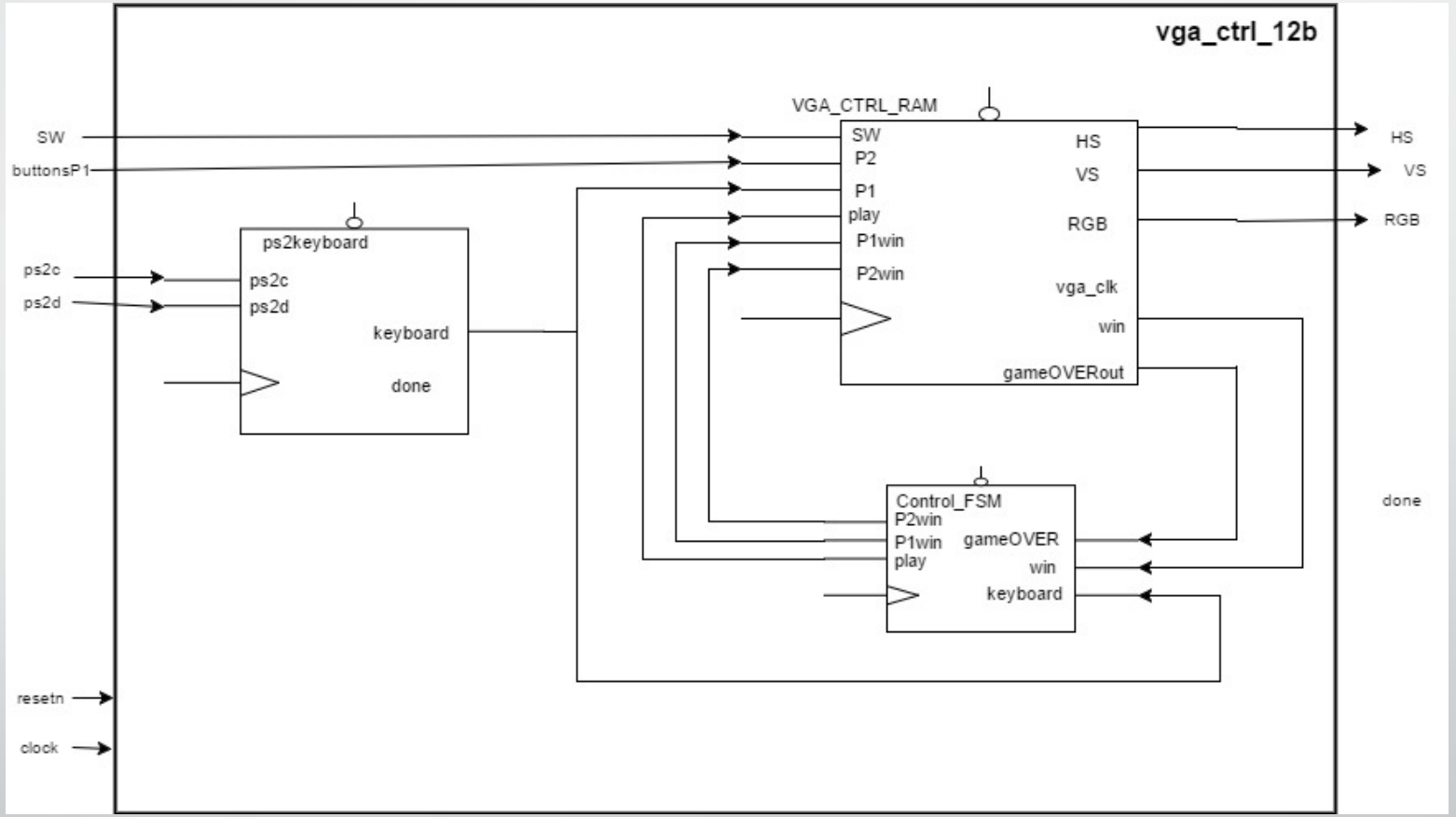
- Memory based implementation for VGA control
- Each object is mapped to a specific pixel location on the map when the game is started



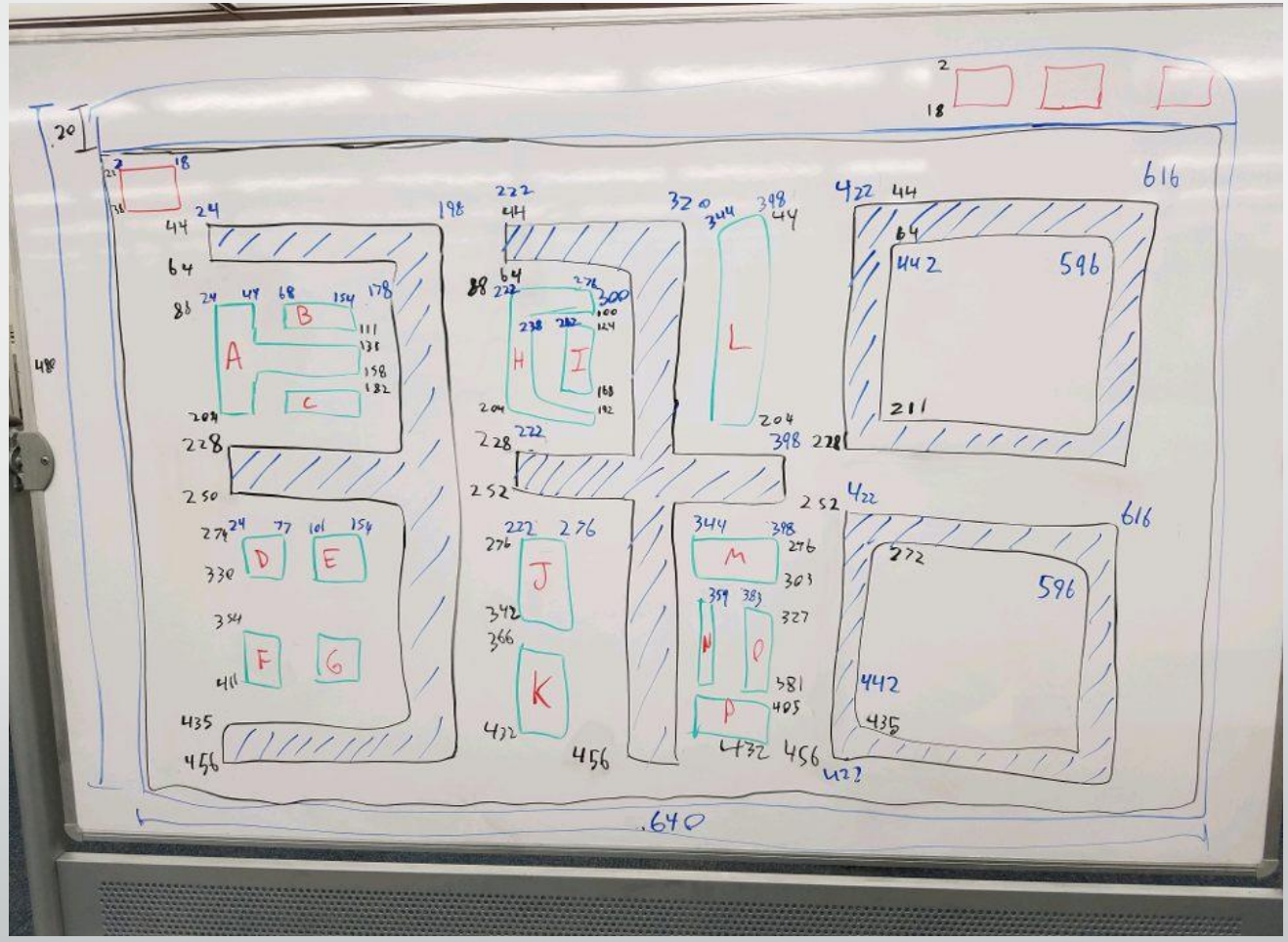
VGA Control Data Path



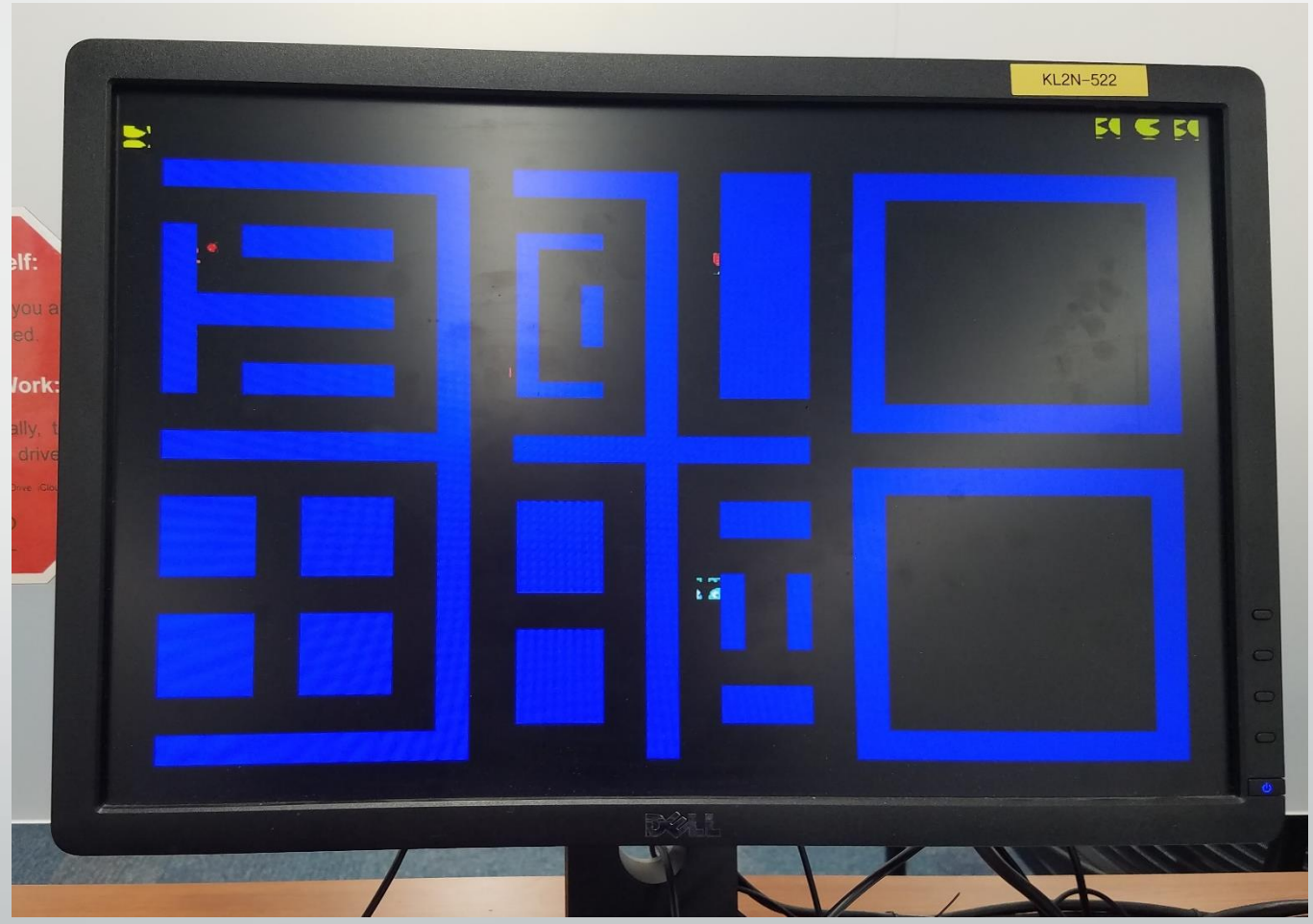
VGA Control Top Level Data Path



Original Game Map Concept



Original Game Map Implementation



Collision Handling

- Collisions with Pac-man are based on his location in comparison to the locations of each object he interacts with
- The Powerup fruits are designed to disappear only when a collision with Pac-man occurs, any collisions with the ghosts are ignored

Display

```
---SELECTS WHICH OBJECT WILL BE PLACED-------------------------------------------------------------------------------
 sel_RGB <= "0000" when gameOVER = 1 and win = '0' and (hcount_buf >= xP2Wins and hcount_buf < (xP2Wins + 128)  --GAMEOVER
                                                    and vcount_buf >= yP2Wins and vcount_buf < (yP2Wins + 128))

           else "1001" when win = '1' and (hcount_buf >= xP1Wins and hcount_buf < (xP1Wins + 128) and
                                           vcount_buf >= yP1Wins and vcount_buf < (yP1Wins + 128))

           else "0001" when (win = '0' and gameOVER = 0 and death = 0 and hcount_buf >= x_PAC and hcount_buf < (x_PAC + 16) and
                                                                          vcount_buf >= y_PAC and vcount_buf < (y_PAC + 16))

           else "1010" when (win = '0' and gameOVER = 0 and hcount_buf >= x_P2 and hcount_buf < (x_P2 + 16) and
                                                            vcount_buf >= y_P2 and vcount_buf < (y_P2 + 16))

           else "1000" when win = '0' and gameOVER = 0 and
```

- Snippet of WHICH object gets placed at a location
- Depends on hcount and vcount

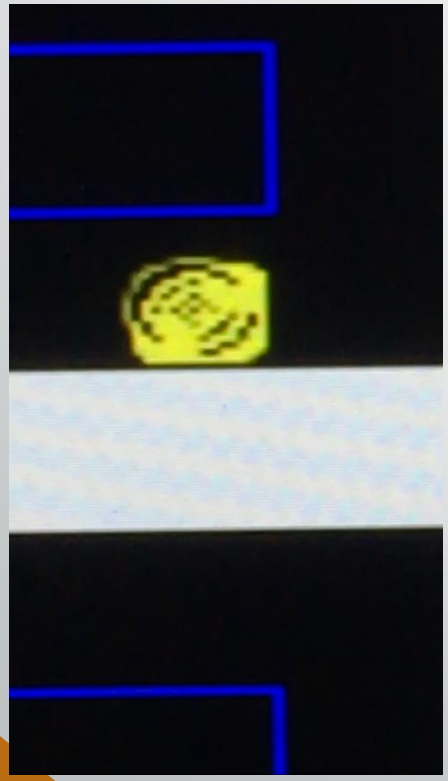
Display

```
with sel_RGB select                                                    -- only the 12 LSBs contain useful data
        in_RGB <= inRAM_odata13(11 downto 0) when "0000",      --gameOVER
                  inRAM_odata(11 downto 0)   when "0001",      --Pacman
                  x"00F"                     when "0010",      --WALLS
                  inRAM_odata3(11 downto 0)  when "0011",      --BlueGHOST1
                  inRAM_odata4(11 downto 0)  when "0100",      --RedGhost2
                  inRAM_odata5(11 downto 0)  when "0101",      --LIFE
                  inRAM_odata6(11 downto 0)  when "0110",      --STOP
                  inRAM_odata11(11 downto 0) when "0111",      --PACMAN LIVES
                  x"FFF"                     when "1000",      --378 WALLS
                  inRAM_odata12(11 downto 0) when "1001",      --WIN SCREEN
                  inRAM_odata2(11 downto 0)  when "1010",      --PLAYER 2
                  inRAM_odata7(11 downto 0)  when "1011",      --BOOST
                  inRAM_odata8(11 downto 0)  when "1100",      --OBJECT1
                  inRAM_odata9(11 downto 0)  when "1101",      --OBJECT2
                  inRAM_odata10(11 downto 0) when "1110",      --OBJECT3

                  sw                         when others;      --Background
```

- Snippet of what DATA gets placed at a location
- Data comes out of memory components

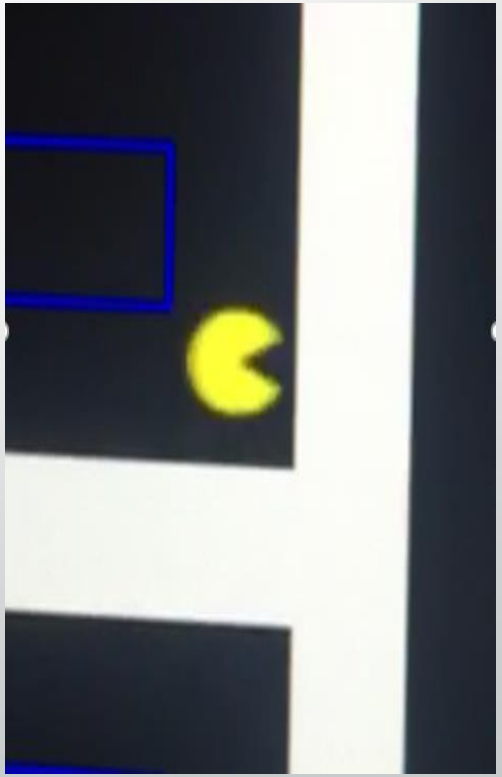
Memory Address



- Left: When your address is incorrect, the corresponding output data is incorrect

- Right: Perfect input, Perfect output
- 2D Array of Pixels 16x16
- Equation to select address for Pacman
  - (16*(vc-y) + (hc-x)) RIGHT
  - (16*(hc-x) + (vc-y)) DOWN
  - (16*(hc-x) + (15 – (vc-y))) UP
  - (16*(vc-y) + (15 – (hc-x))) LEFT

  - "Rotation" of RIGHT image



Problems Faced

- Issue with image rendering using original equation for controlling movement
- Occasional glitches allowing the player to move through the walls
- Trouble getting the first object to move
- Originally keeping Pac-man contained to the screen

Demo Time


LOONEY TUNES
"That's all Folks!"