GATEBOY VIDEO

a.k.a Project Whizzgraphics

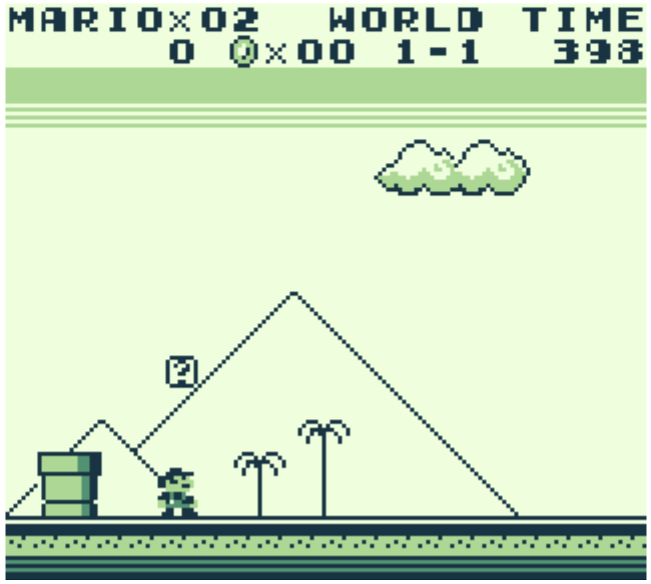
Andy Goetz, Phil Lamb, Kevin Riedl
ECE510
Spring 2013

Introduction

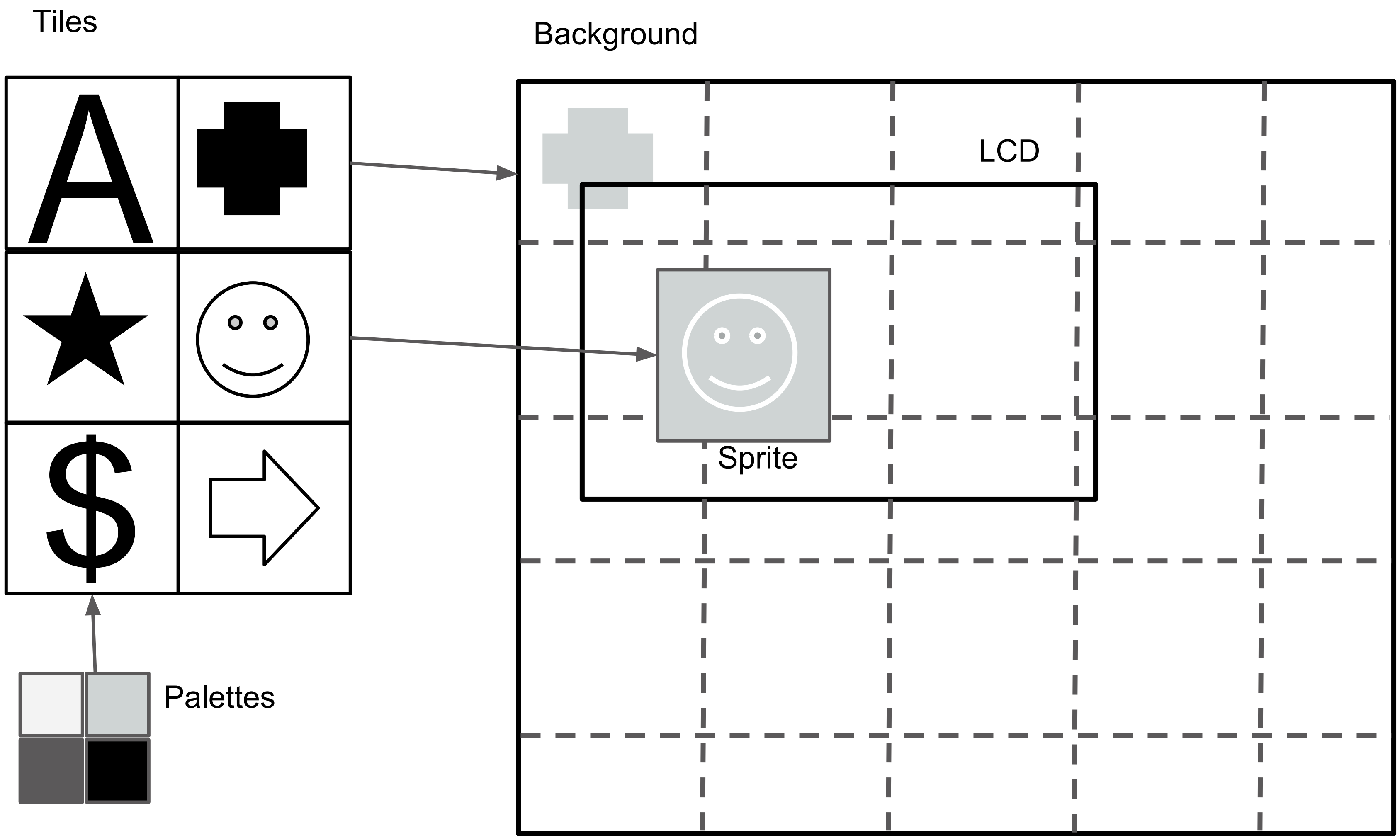
- Video subsystem of original Game Boy
- Generate LCD output
- Support connection to additional subsystems

Introduction (cont)

- 160x144 pixel LCD
- 2-bit grayscale color
- Tile-based background and sprites
- All Tiles are 8x8 pixels

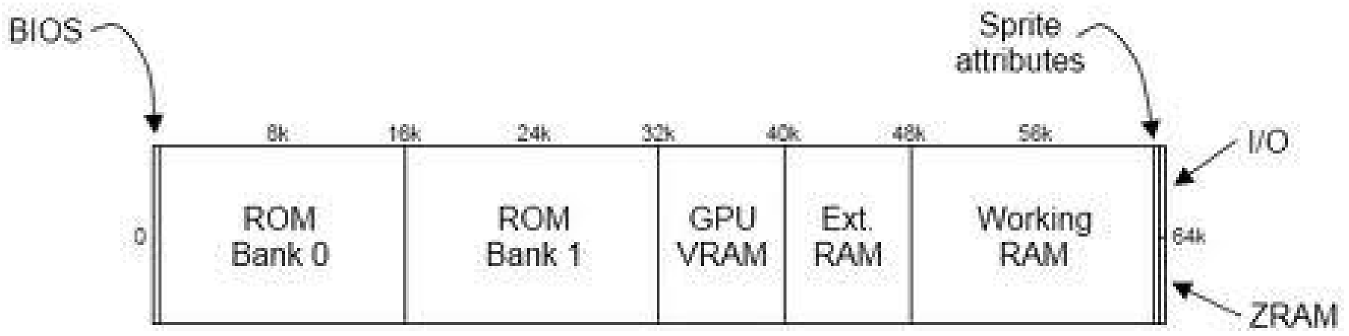


Design Overview

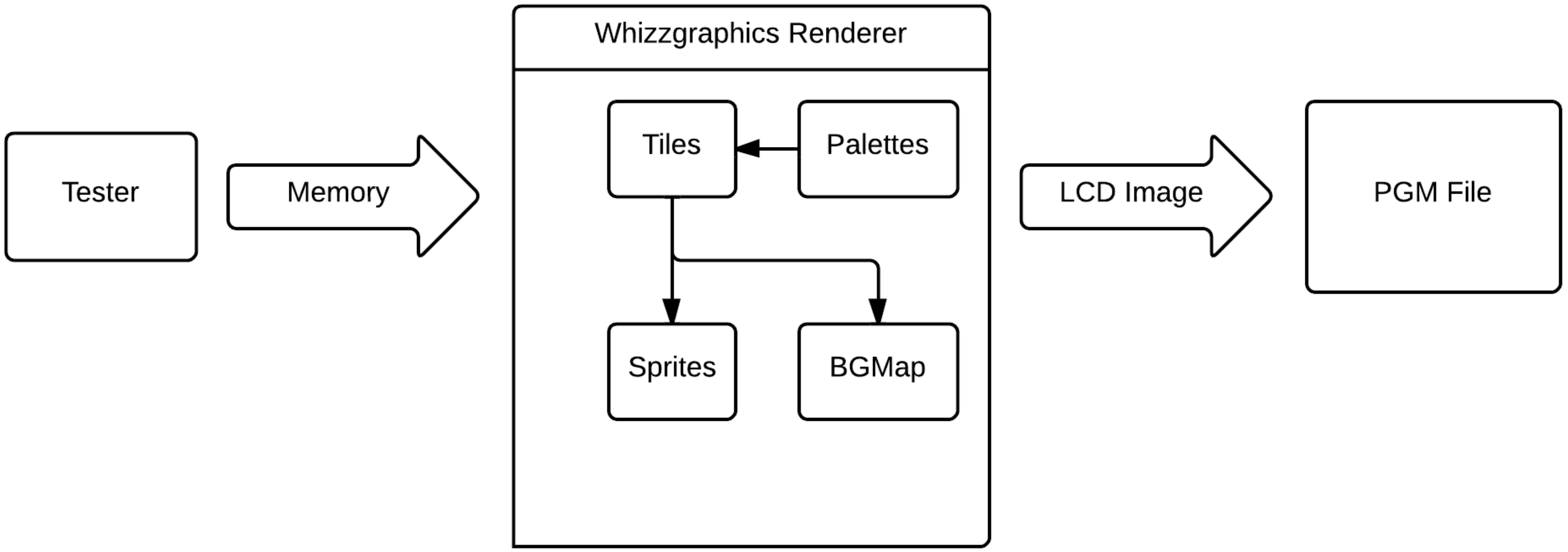


Design Overview

- Sprites, Tiles, Backgrounds, Palettes all stored in video memory (VRAM)
- Memory mapped control and status registers
- Data bus used for interface to the VRAM



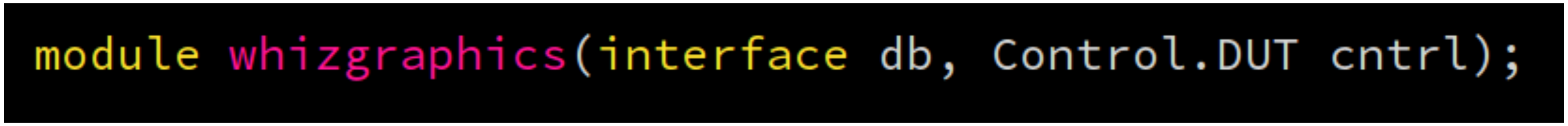
Design Overview



Design Overview

- CPU (simulated as tester) accesses memory via data bus interface

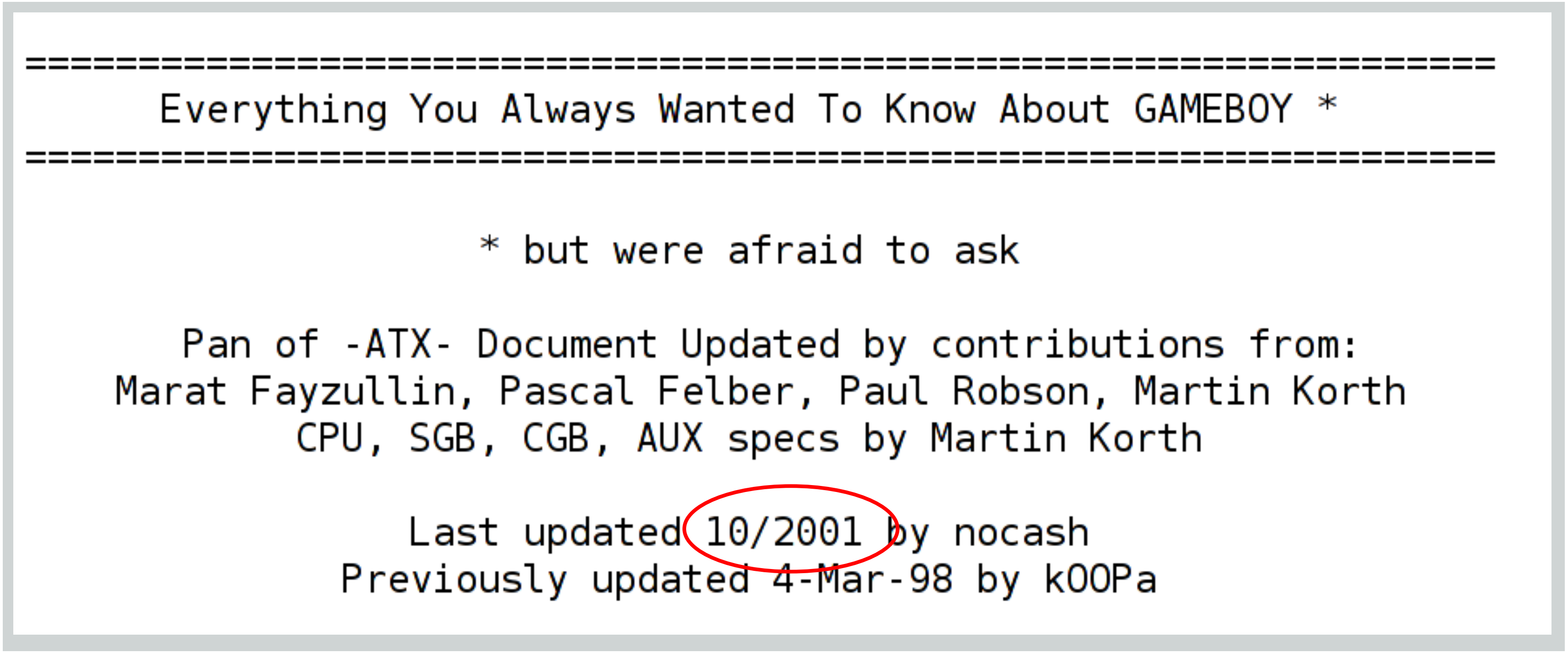
- Implemented behavioral timing model, not necessarily clock accurate



```
module whizgraphics(interface db, Control.DUT cntrl);
```

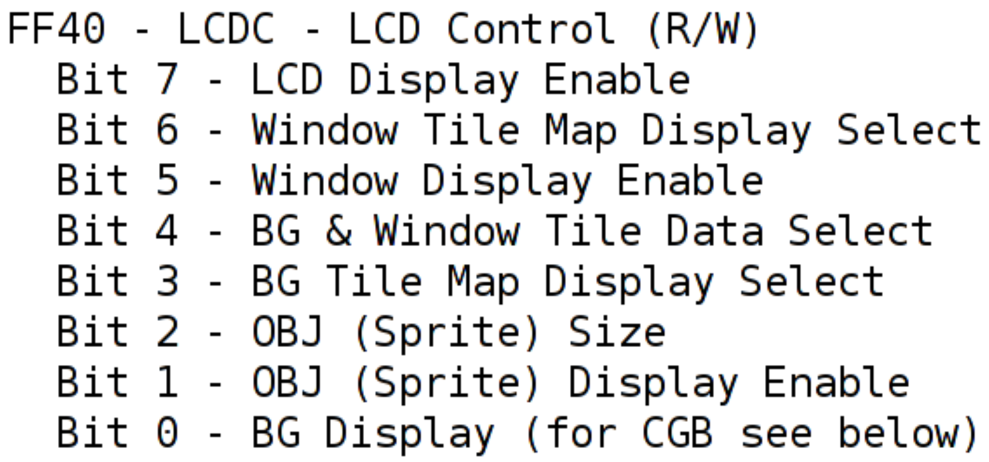
Existing Work / Documentation

- This is pretty much all we had....



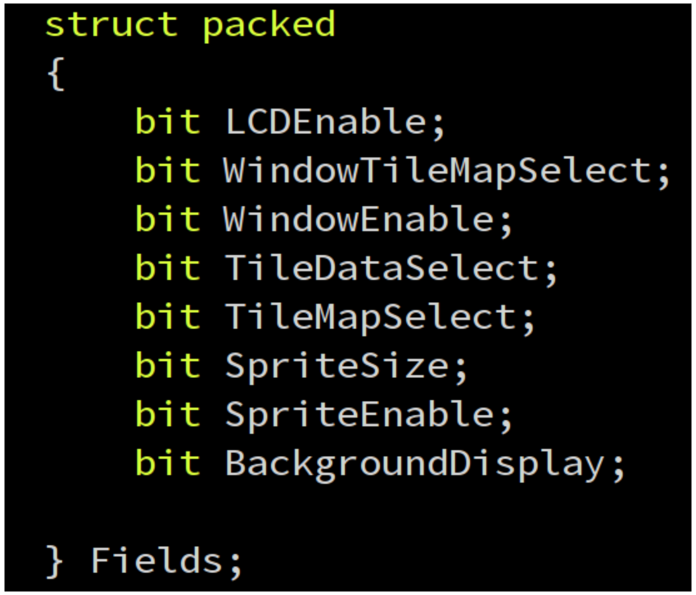
```
============================================================================
        Everything You Always Wanted To Know About GAMEBOY *
============================================================================


                       * but were afraid to ask

        Pan of -ATX- Document Updated by contributions from:
     Marat Fayzullin, Pascal Felber, Paul Robson, Martin Korth
               CPU, SGB, CGB, AUX specs by Martin Korth

                   Last updated 10/2001 by nocash
                 Previously updated 4-Mar-98 by kOOPa
```

Structure Example



```
FF40 - LCDC - LCD Control (R/W)
  Bit 7 - LCD Display Enable
  Bit 6 - Window Tile Map Display Select
  Bit 5 - Window Display Enable
  Bit 4 - BG & Window Tile Data Select
  Bit 3 - BG Tile Map Display Select
  Bit 2 - OBJ (Sprite) Size
  Bit 1 - OBJ (Sprite) Display Enable
  Bit 0 - BG Display (for CGB see below)
```

PanDoc description



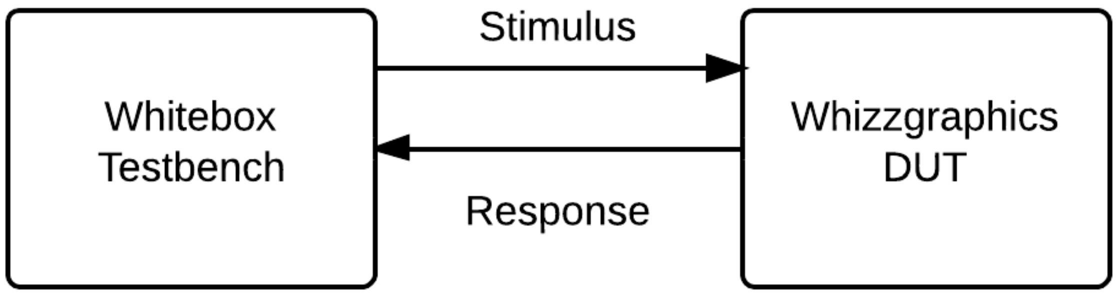
```
struct packed
{
    bit LCDEnable;
    bit WindowTileMapSelect;
    bit WindowEnable;
    bit TileDataSelect;
    bit TileMapSelect;
    bit SpriteSize;
    bit SpriteEnable;
    bit BackgroundDisplay;

} Fields;
```

SystemVerilog® Implementation

Testing

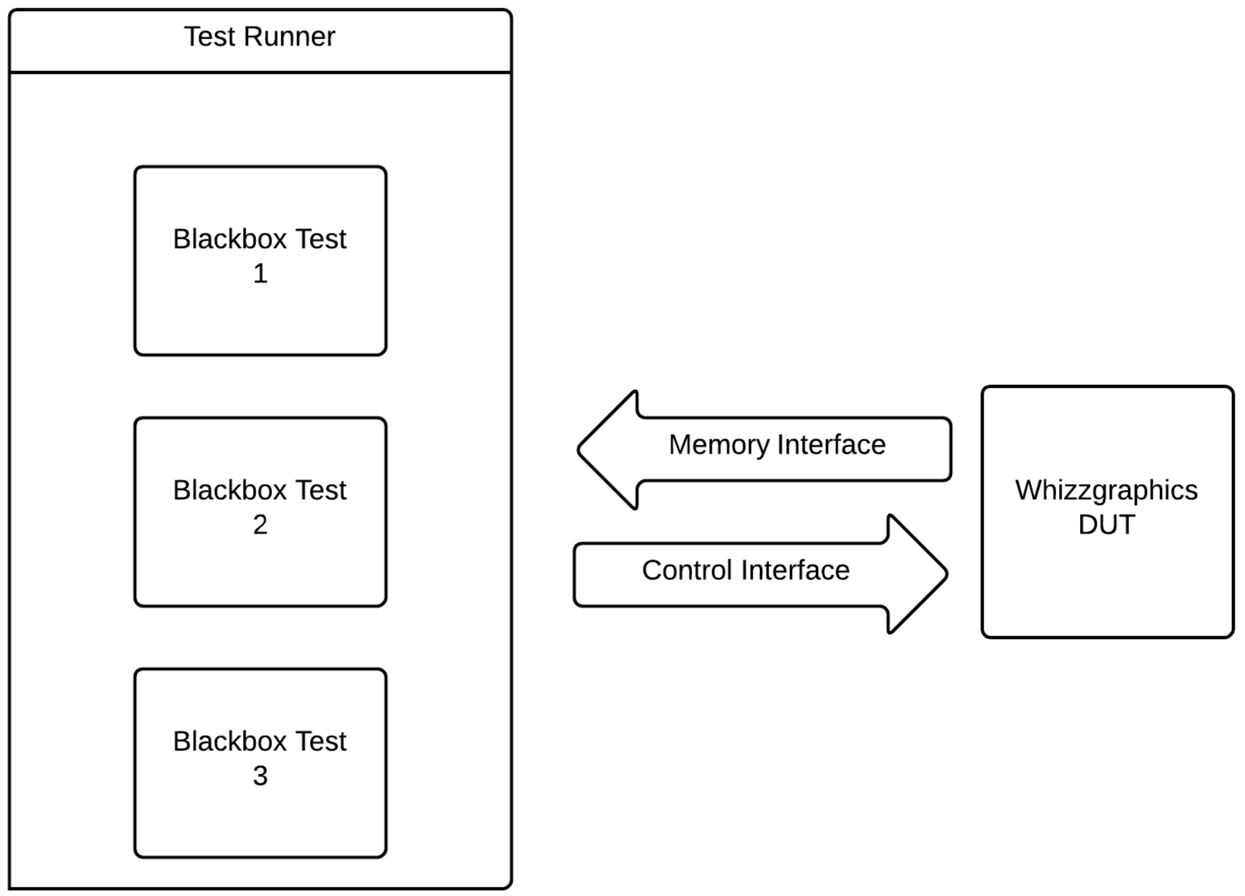
- Two types of testing:
  - Whitebox - Directly accesses internal structures
  - Blackbox - Only uses memory interface to interact with module

Simple, Whitebox Testing



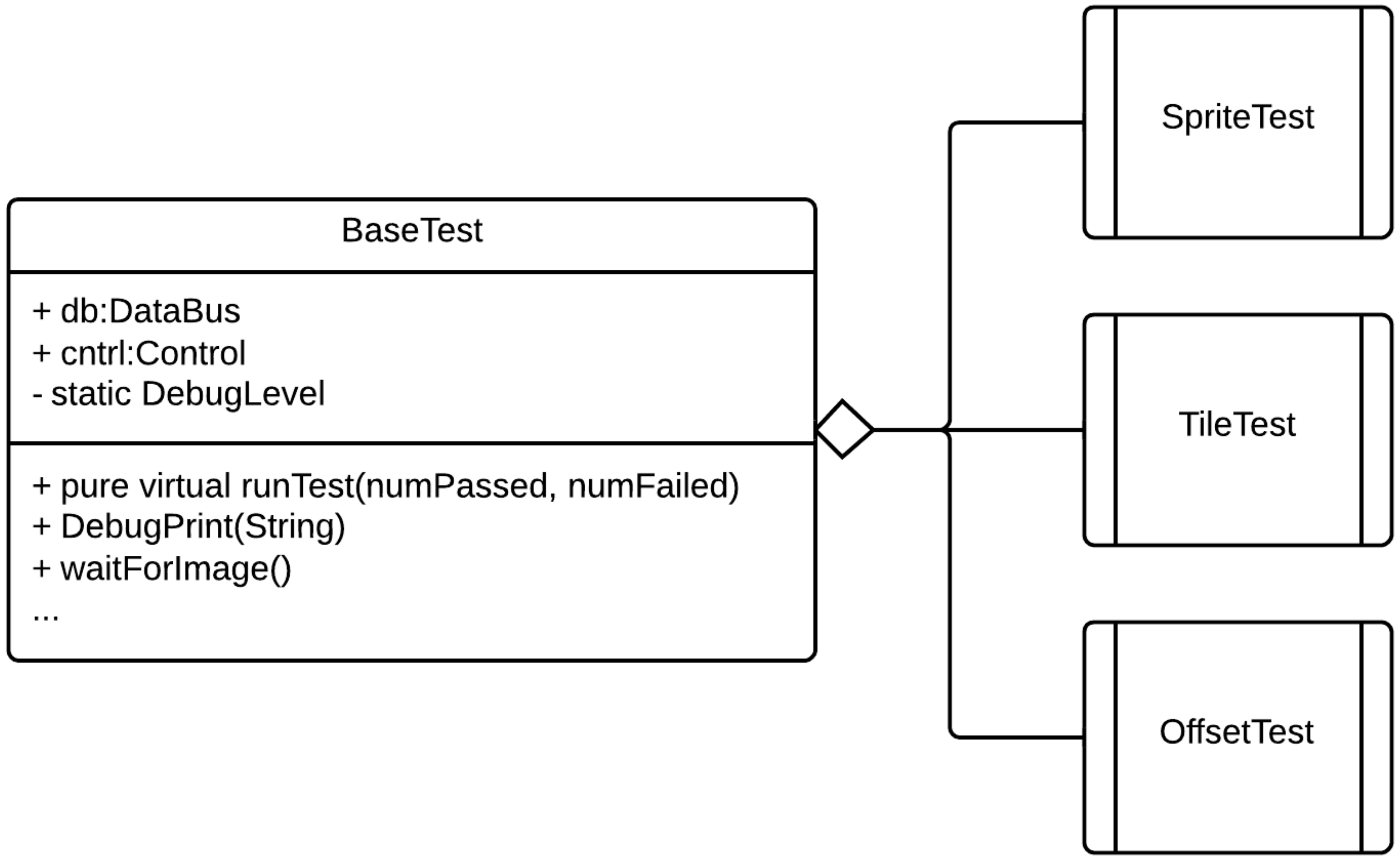
- Tests reach into DUT and directly stimulate data structures.

More Complex, Blackbox Testing

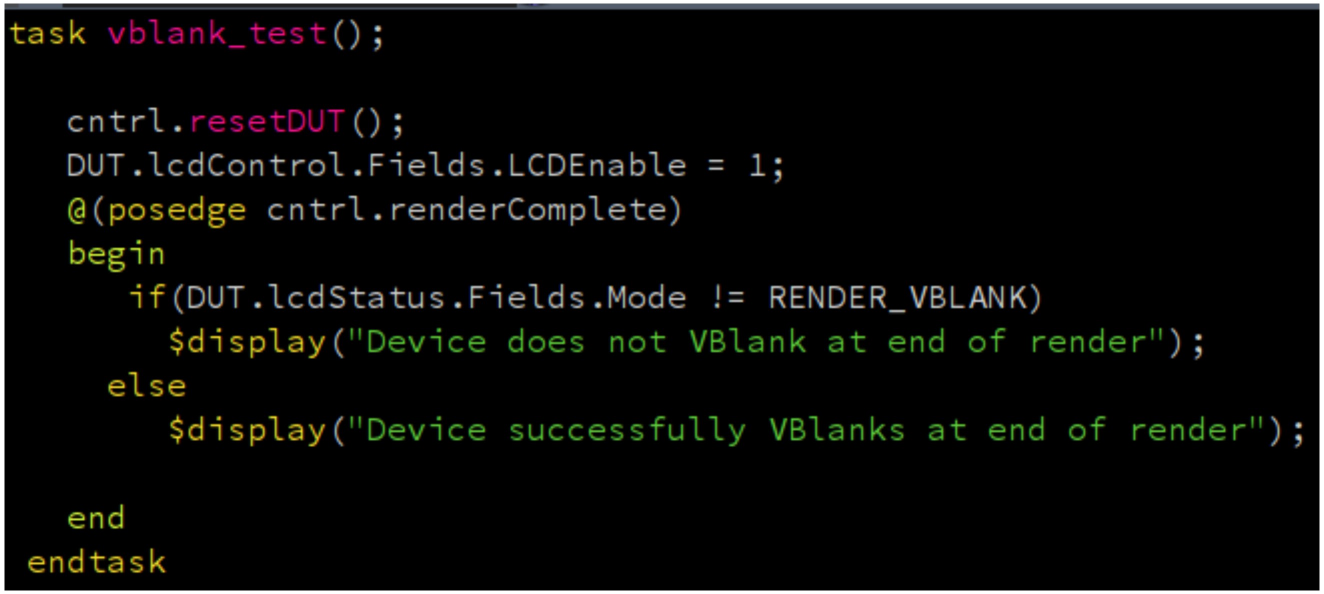


Uses OOP Test Runner

Blackbox UML Diagram

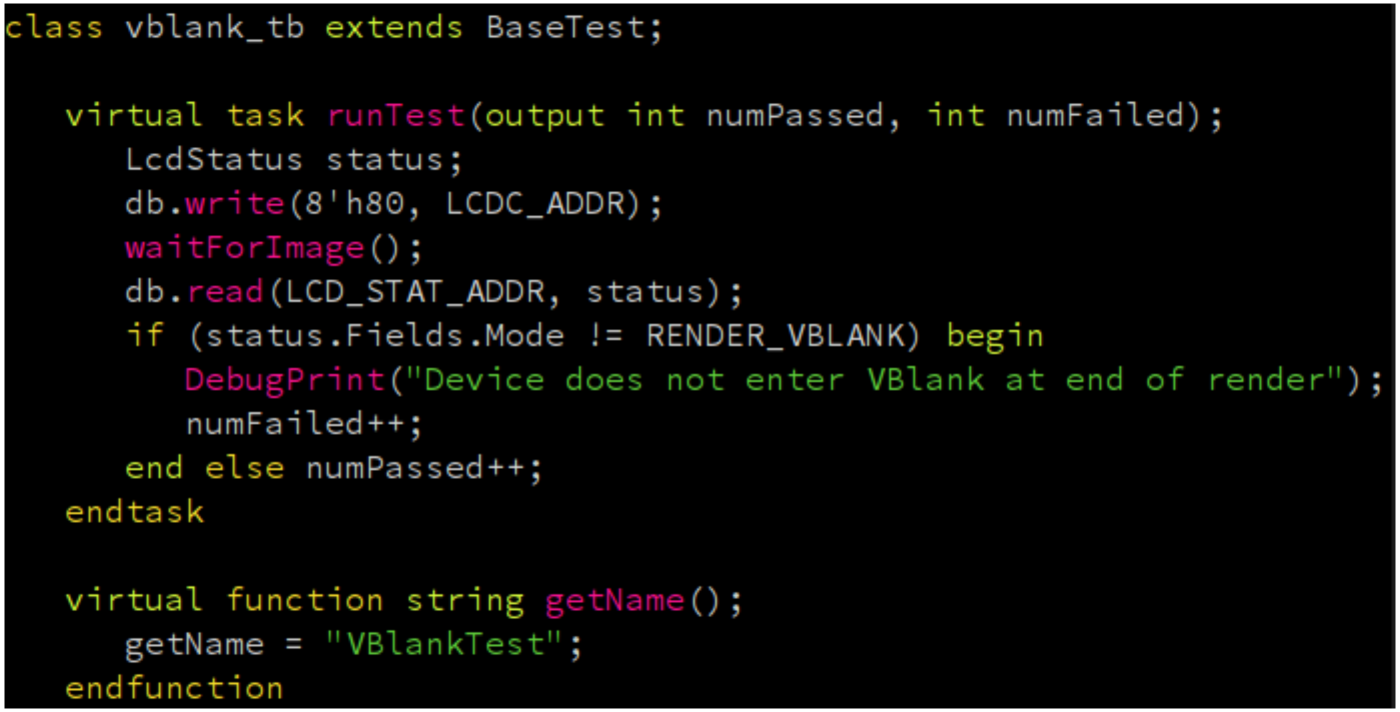


Example Whitebox Test



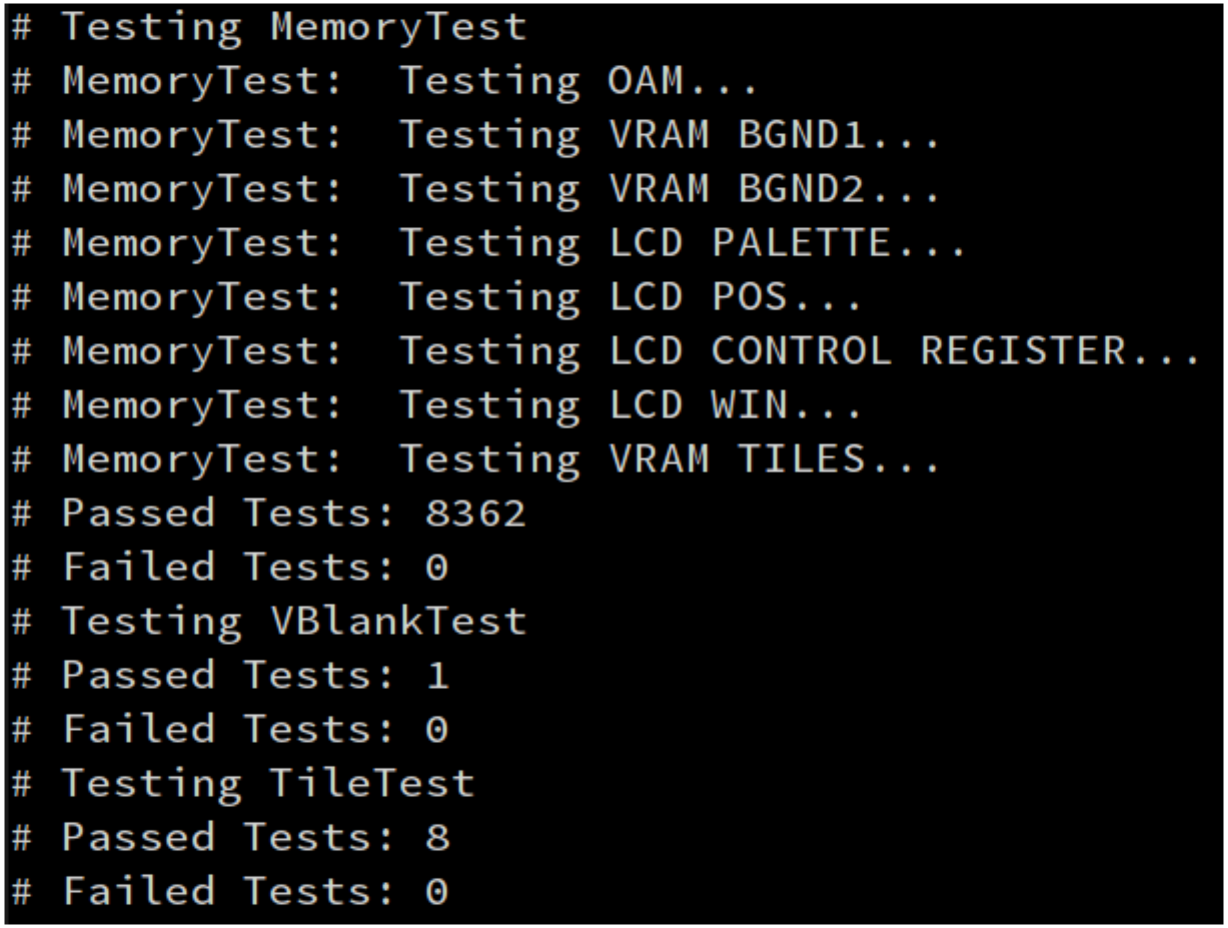
```
task vblank_test();

   cntrl.resetDUT();
   DUT.lcdControl.Fields.LCDEnable = 1;
   @(posedge cntrl.renderComplete)
   begin
      if(DUT.lcdStatus.Fields.Mode != RENDER_VBLANK)
        $display("Device does not VBlank at end of render");
     else
        $display("Device successfully VBlanks at end of render");

   end
 endtask
```

Example Blackbox Test



```
class vblank_tb extends BaseTest;

   virtual task runTest(output int numPassed, int numFailed);
      LcdStatus status;
      db.write(8'h80, LCDC_ADDR);
      waitForImage();
      db.read(LCD_STAT_ADDR, status);
      if (status.Fields.Mode != RENDER_VBLANK) begin
         DebugPrint("Device does not enter VBlank at end of render");
         numFailed++;
      end else numPassed++;
   endtask

   virtual function string getName();
      getName = "VBlankTest";
   endfunction
```

Test Runner Output



```
# Testing MemoryTest
# MemoryTest:  Testing OAM...
# MemoryTest:  Testing VRAM BGND1...
# MemoryTest:  Testing VRAM BGND2...
# MemoryTest:  Testing LCD PALETTE...
# MemoryTest:  Testing LCD POS...
# MemoryTest:  Testing LCD CONTROL REGISTER...
# MemoryTest:  Testing LCD WIN...
# MemoryTest:  Testing VRAM TILES...
# Passed Tests: 8362
# Failed Tests: 0
# Testing VBlankTest
# Passed Tests: 1
# Failed Tests: 0
# Testing TileTest
# Passed Tests: 8
# Failed Tests: 0
```

Whitebox vs Blackbox

Whitebox (modules)

- Can directly twiddle DUT structures
- Repeated code used to set up DUT.
- Tied directly to underlying implementation
- May need Manual Inspection

Blackbox (classes)

- Can use standardized test features
- Relies on Interfaces to access DUT
- Tests programmer's interface
- Programmatically Checks Results

List of Tests

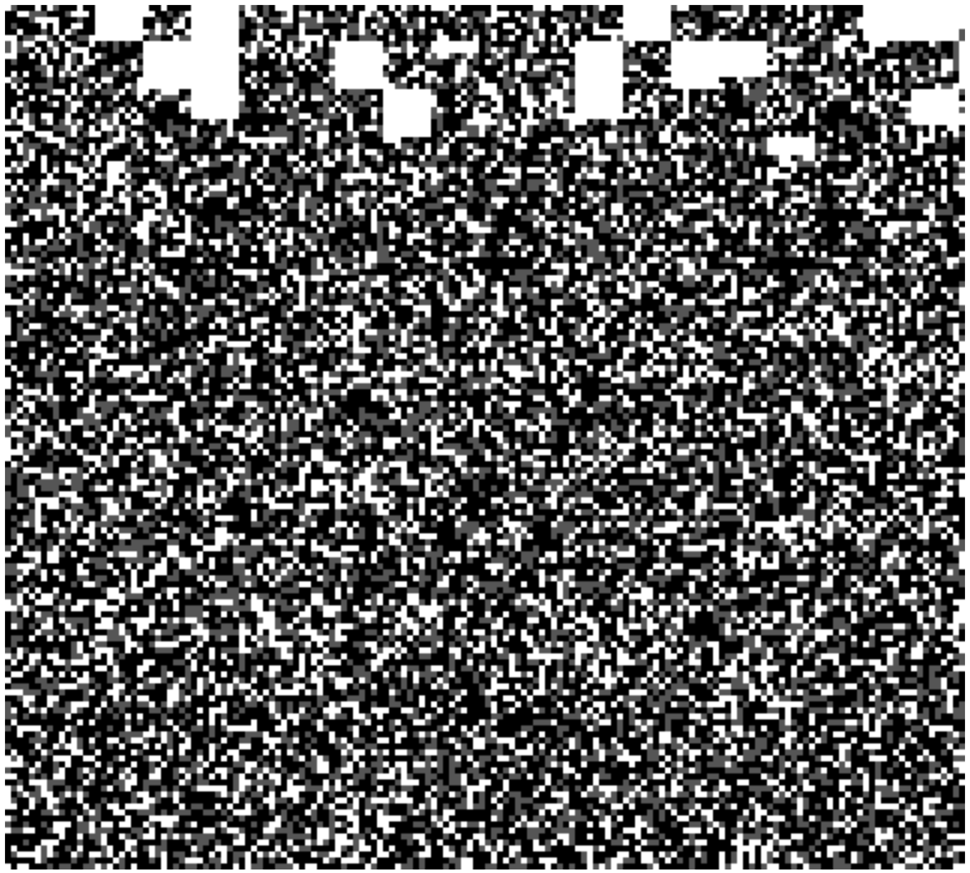
Whitebox

- Data Structure Access
- Memory Interface
- Palette Swapping
- Rendered Images

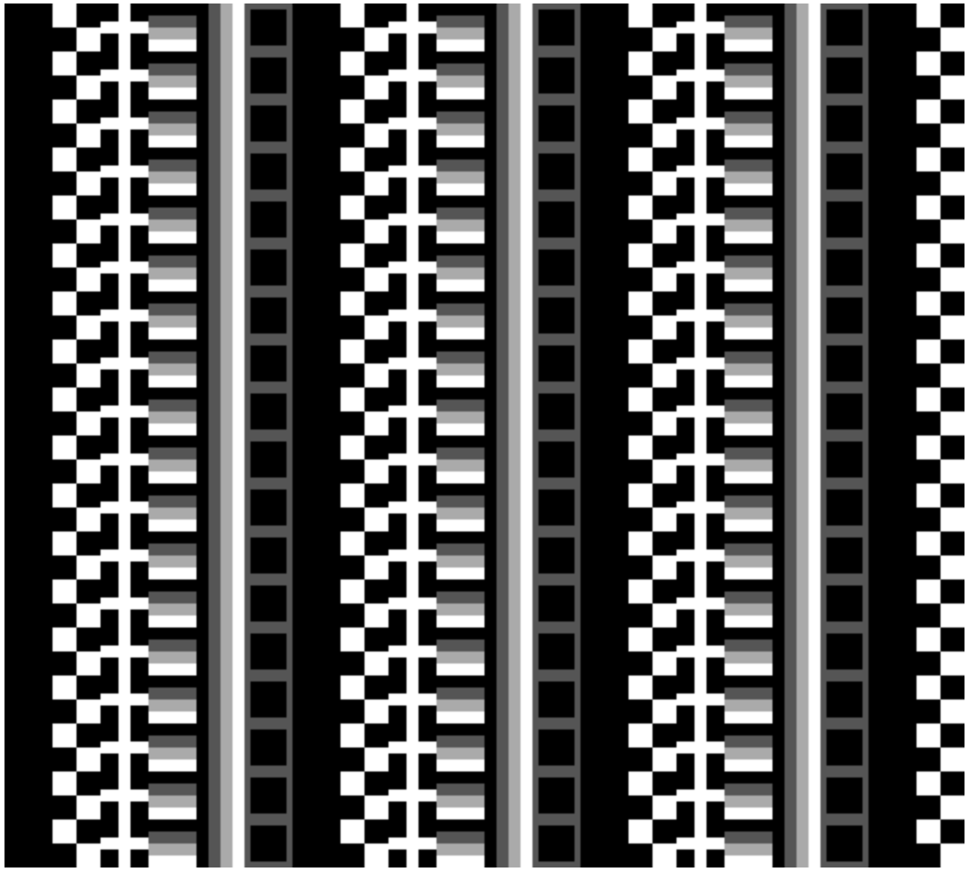
Blackbox

- Data Structure Access
- Vblank mode
- Background Scrolling
- Background Data Source
- Sprite Flipping

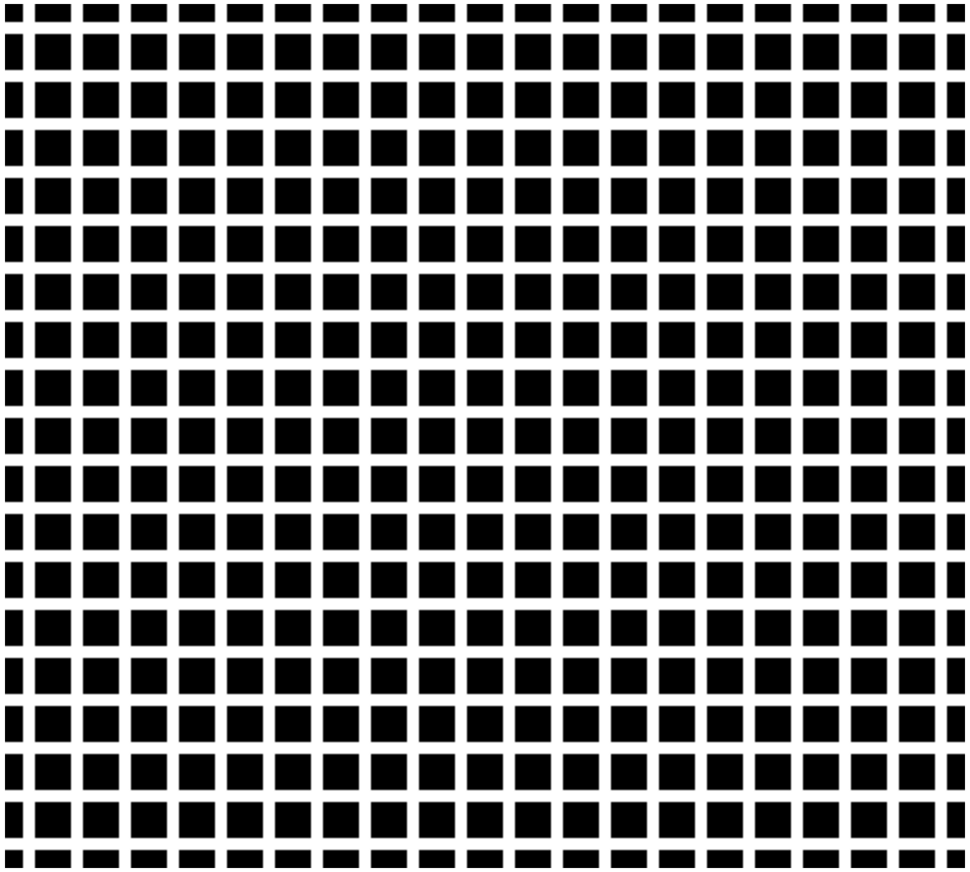
Sample Output (First Image)



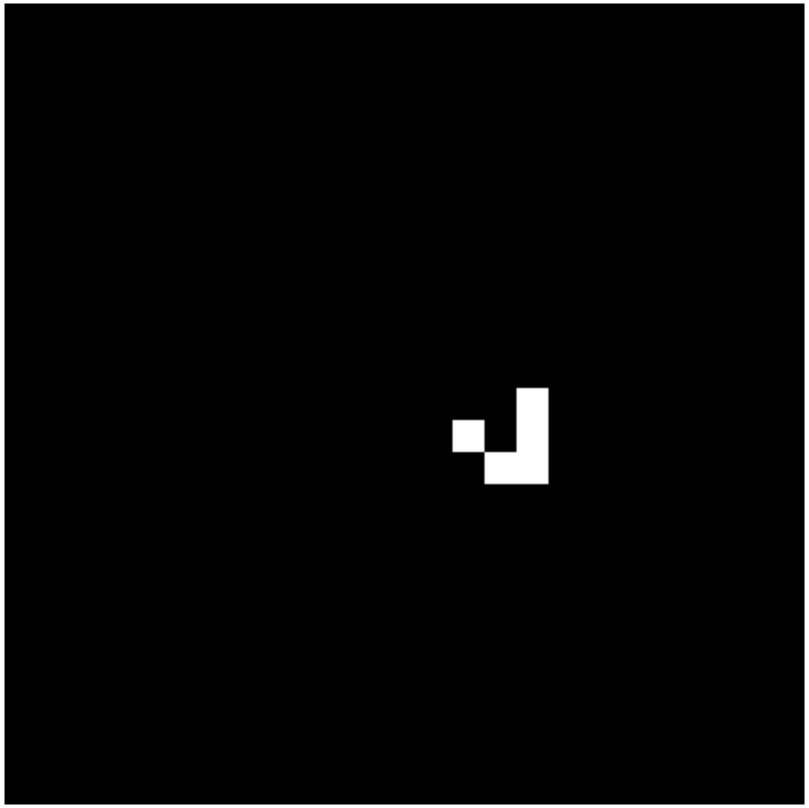
Tile Example



Scrolling Example



Sprite Example



Results - implemented

- Tile based background and sprites
- Advanced sprite rendering (mirroring, flipping, transparency)
- Memory interface to VRAM and Registers
- Palette mapping and switching
- Background scrolling and wrapping
- Sprite movement
- Tile map switching

Results - unimplemented

- Window layer
- Sprite palette selection
- HBlank timing
- Sprite layering

However, all required data structures are implemented.

SystemVerilog© Features

- User defined types
- Structures and unions (Gameboy register interface)
- Interfaces and Clocking Blocks (memory and control)
- Packages (Everywhere)
- Classes (Blackbox testing)
- Dynamic data structures (Whitebox testing)
- Assertions (Error checking)

Lessons Learned

- Bit off more than could be chewed
- Concepts may be simple, but implementation can be non-trivial
- Refactor early, Refactor often
- Spent more time developing tests rather than features
- Investing in test infrastructure can pay off in the end

Questions?