Display Technology

- Images stolen from various locations on the web...

Cathode Ray Tube

- Electron Gun

- Beam Steering Coils

Raster Scanning
Liquid Crystal Displays

DLP Projector

LCoS

- Liquid Crystal on Silicon
- Put a liquid crystal between a reflective layer on a silicon chip
**Grating Light Valve (GLS)**

- Lots (8000 currently) of micro ribbons that can bend slightly
- Make them reflective
- The bends make a diffraction grating that controls how much light where
- Scan it with a laser for high light output
- 4000 pixel wide frame ever 60Hz

**Digistar 3 Dome Projector**

**VGA**

- Stands for Video Graphics Array
- A standard defined by IBM back in 1987
  - 640 x 480 pixels
  - Now superseded by much higher resolution standards...
- Also means a specific analog connector
  - 15-pin D-subminiature VGA connector

**VGA Connector**

1: Red out  
6: Red return (ground)  
11: Monitor ID 0 in
2: Green out  
7: Green return (ground)  
12: Monitor ID 1 in or data from display
3: Blue out  
8: Blue return (ground)  
13: Horizontal Sync
4: Unused  
9: Unused  
14: Vertical Sync
5: Ground  
10: Sync return (ground)  
15: Monitor ID 3 in or data clock

**Raster Scanning**
Raster Scanning

VGA Timing

VGA Timing Summary

Relaxed VGA Timing

This all sounds pretty strict and exact...

It’s not really... The only things a VGA monitor really cares about are:

- Hsync
- Vsync

Actually, all it cares about is the falling edge of those pulses!

The beam will retrace whenever you tell it to

It’s up to you to make sure that the video signal is 0V when you are not painting (i.e. retraceing)
Relaxed VGA Timing

- Horizontal Dots: 128
- Vertical Scan Lines: 255
- Horiz. Sync Polarity: NEG
- A (µs): 30.0 Scanline time
- B (µs): 2.0 Sync pulse length
- C (µs): 10.7 Back porch
- D (µs): 12.8 Active video time
- E (µs): 4.50 Front porch

12.8/128 = 100ns/pixel = 10 MHz pixel clock

VGA Timing

- Horizontal Dots: 128
- Vertical Scan Lines: 255
- Vert. Sync Polarity: NEG
- Vertical Frequency: 60Hz
- O (ms): 16.68 Total frame time
- P (ms): 0.09 Sync pulse length (3x30µs)
- Q (ms): 4.86 Back porch
- R (ms): 7.65 Active video time
- S (ms): 4.08 Front porch

VGA Voltage Levels

- Voltages on R, G, and B determine the color
- Analog range from 0v (off) to +0.7v (on)
- But, our pads produce 0-3.3v outputs!
- For B&W output, just drive RGB together and let 0v=black and 3.3v=white
- Overdrives the input amps, but won’t really hurt anything
- For color you can drive R, G, B separately
  - Of course, this is only 8 colors (including black and white)
  - Requires storing three bits at each pixel location

VGA on Spartan3e Starter

- Series resistors limit output voltage to 0-0.7v
Raster Scanning

VGA on Spartan3e Starter

VGA Assignment

3 Types of bitGen

Character/Glyph-based

3 Types of bitGen

Bitmapped

Table 6-2 96x48 Mode VGA Timing

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time</th>
<th>Clocks</th>
<th>Line</th>
<th>Time</th>
<th>Clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_X$</td>
<td>hSync pulse time</td>
<td>29.7 ns</td>
<td>636.967</td>
<td>551</td>
<td>52.4 ns</td>
</tr>
<tr>
<td>$T_Y$</td>
<td>Display time</td>
<td>17.3 ns</td>
<td>348.327</td>
<td>486</td>
<td>20.1 ns</td>
</tr>
<tr>
<td>$T_{P_{x}}$</td>
<td>Pixel width</td>
<td>10.4 μs</td>
<td>1.686</td>
<td>1</td>
<td>3.562 μs</td>
</tr>
<tr>
<td>$T_{P_{y}}$</td>
<td>Pixel pitch</td>
<td>120 ps</td>
<td>2.399</td>
<td>73</td>
<td>440 ps</td>
</tr>
<tr>
<td>$T_{P_{z}}$</td>
<td>Back pitch</td>
<td>3.3 ps</td>
<td>23.238</td>
<td>29</td>
<td>1.912 μs</td>
</tr>
</tbody>
</table>

Character/Glyph-based

- Break screen into nxm pixl chunks (e.g. 8x8)
- For each chunk, point to one of k nxm glyphs
- Those glyphs are stored in a separate memory
- For 8x8 case (for example)
  - glyph number is hCount and vCount minus the low three bits
  - glyph bits are the low-order 3 bits in each of hCount and vCount
  - Figure out which screen chunk you’re in, then reference the bits from the glyph memory

```
3 Types of bitGen

- Bitmapped
  - Frame buffer holds a separate rgb color for every pixel
  - bitGen just grabs the pixel based on hCount and vCount and splats it to the screen
  - Chews up a LOT of memory

- Character/Glyph-based
  - Break screen into nxm pixl chunks (e.g. 8x8)
  - For each chunk, point to one of k nxm glyphs
  - Those glyphs are stored in a separate memory
  - For 8x8 case (for example)
    - glyph number is hCount and vCount minus the low three bits
    - glyph bits are the low-order 3 bits in each of hCount and vCount
    - Figure out which screen chunk you’re in, then reference the bits from the glyph memory

vgaControl

- Generate timing pulses at the right time
- hSync, vSync, bright, hCount, vCount

bitGen

- Based on bright, hCount, vCount, turn on the bits
```

VGA on Spartan3e Starter

VGA Assignment
3 Types of bitGen

- Direct Graphics
  - Look at hCount and vCount to see where you are on the screen
  - Depending on where you are, force the output to a particular color
  - Tedious for complex things, nice for large, static things

```verilog
parameter BLACK = 3'b 000, WHITE = 3'b111, RED = 3'b100;
// paint a white box on a red background
always@(*)
  if (~bright) rgb = BLACK; // force black if not bright
  else if (((hCount >= 100) && (hCount <= 300)) &&
             ((vCount >= 150) && (vCount <= 350))) rgb = WHITE;
  else rgb = RED; // background color
```

VGA Memory Requirements

- Remember, Spartan3e has 20 18kbit Block RAMs
  - i.e. 20k addresses where each address is a 16-bit (or 18 bit) word
  - But, 16 bits of address = 64k addresses
  - So, you can’t use all the address space with just Block RAMs

- 640x480 VGA (bitmapped)
  - 307,200 pixels
  - 3 bits per pixel
  - 6 pixels per 18-bit word
  - 50k locations for 640x480
  - Oops – we only have 20k, and you need some space for code and other data…

- 320x240 VGA (bitmapped)
  - 76,800 pixels
  - Each stored pixel is 2x2 screen pixels
  - 3 bits per pixel
  - 6 pixels per 18-bit word
  - 12.5k 18-bit words needed
  - Much more realistic… 7.5k left over for code/data

- 80 char by 60 line display (8x8 glyphs)
  - 4800 locations
  - Each location has one of 256 char/glyphs
  - 8-bits per location – 2 locations per word
  - 2400 addresses for frame buffer
  - Each char/glyph is (say) 8x8 pixels
    - results in 640x480 display…
  - 8x8x256 bits for char/glyph table
    - 16bits (1k words) for char/glyph table

Character Example…

- 64 characters each 8x8 pixels
Character Example...

The Character ROM works on the 64-member ASCII upper-case character set. The characters are addressed with a 6-bit binary address A16=0 and a 16-bit upper character address, R065=OE-0D. The Character ROM outputs a single row of the selected character at a time on the signals T[15].

A16=0 decodes one of the four rows of 16 characters in the ROM.
A16=0 = 0 = first row
A16=0 = 1 = second row
A16=0 = 2 = third row
A16=0 = 3 = fourth row

The sixteen signals nR065, nOE5, nOE6, nOE7, nOE8, nOE9, nOE10, nOE11, nOE12, nOE13, nOE14, nOE15, nOE16, nA065, nA066 are selected one of the sixteen columns of four characters. These signals are active low and only one is asserted at any time. For instance, nOE6=0 selects the first column with the four characters "A", "D", "G" and "T". If A16=0 selects "A" then A065=0 will produce the following binary output on T[7:0].

<table>
<thead>
<tr>
<th>AD16</th>
<th>Variable Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000000</td>
</tr>
<tr>
<td>1</td>
<td>00000000</td>
</tr>
<tr>
<td>2</td>
<td>00000000</td>
</tr>
<tr>
<td>3</td>
<td>00000000</td>
</tr>
<tr>
<td>4</td>
<td>00000000</td>
</tr>
<tr>
<td>5</td>
<td>00000000</td>
</tr>
<tr>
<td>6</td>
<td>00000000</td>
</tr>
<tr>
<td>7</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Remember the skier/chicken/tron example?
That used character/glyph graphics similar to this...

Tbird VGA Assignment

- Get VGA working
- Start with full-screen flood
- Then play around with direct VGA graphics
- Take the Tbird state machine
- Outputs are six lights
- Define six regions of the screen
- Make those regions change color when the state machine says the lights should be on

Other I/O (more details later)

- LCD display
  - 2-line 16-char display
  - Reasonably easy to use, once you can do it under program control!
  - Reading and writing memory-mapped 8-bit registers
- PS/2 mouse/keyboard port
- RS323 connector and level converter
- DAC
  - 12 bit unsigned resolution – four outputs
- ADC
  - Dual-channel – 14 bit resolution
- Seven-segment LCDs
  - Already in your kits…

See the Starter Board user’s guide for more details!