Recap and Outline

• Last time: Xilinx
  Distributed RAM (1.2 M bits) and Block RAM (5.3 Mbits)
Outline

• ZBT SRAM
• Video input on ML505
• Video output on ML505
• video frame buffer

XUP Board External SRAM

“ZBT” synchronous SRAM, 9 Mb on 32-bit data bus, with four “parity” bits 256K x 36 bits (located under the removable LCD)
256K x 36 SRAM

Feature Tracking Project
Video Display

- Pixel Array:
  - A digital image is represented by a matrix of values where each value is a function of the information surrounding the corresponding point in the image. A single element in an image matrix is a picture element, or pixel.
  - A pixel includes info for all color components. Common standard is 8 bits per color (Red, Green, Blue)
  - The pixel array size (resolution) varies for different applications, device, & costs, e.g. common value is 1024 X 768 pixels.
  - Frames: The illusion of motion is created by successively flashing still pictures called frames. Frame rates vary depending on application. Usually in range of 25-75 fps. We will use 75 fps (frames per second).
Video Display

- Images are generated on the screen of the display device by “drawing” or scanning each line of the image one after another, usually from top to bottom. Early display devices (CRTs) required time to get from the end of a scan line to the beginning of the next. Therefore each line of video consists of an active video portion and a horizontal blanking interval interval.

- **A vertical blanking** interval corresponds to the time to return from the bottom to the top.
  - In addition to the active (visible) lines of video, each frame includes a number of non-visible lines in the vertical blanking interval.

Video Display

- **Display Devices, CRTs, LCDs, PDP, etc.**
  - Devices come in a variety of native resolutions and frame rates, and also are designed to accommodate a wide range of resolutions and frame rates.
  - Pixels values are sent one at a time through either an analog or digital interface.
  - Formerly, display devices had limited “persistence”, therefore frames were repetitively sent, to create a stable image. Display devices don’t typically store the image in memory.
  - Repetitively sending the image allows motion.

- For a typical resolution and frame rate:
  - **Pixels per frame** = 800x600 = 480,000
  - **Pixel rate** = 75fps X 480,000 = 36,000,000 pixels/sec
  - With blank: 75fps x 1056 x 625 = 49,500,000 pixels/sec

Note: in this example, we use a pixel clock rate of 49.5 MHz to account for blanking intervals.

[Image: http://commons.wikimedia.org/wiki/File:CRT_color.png]
VGA Specs- 800x600@75 Hz

pixel clock = 49.5 MHz

1 line = 21.33 micro sec

Example timing 800x600@75 Hz

<table>
<thead>
<tr>
<th>Screen refresh rate</th>
<th>75 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal refresh rate</td>
<td>46.875 kHz</td>
</tr>
<tr>
<td>Pixel freq.</td>
<td>49.5 MHz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scanline part</th>
<th>Pixels</th>
<th>Time [µs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible area</td>
<td>800</td>
<td>16.161616161616</td>
</tr>
<tr>
<td>Front porch</td>
<td>16</td>
<td>0.32323232323232</td>
</tr>
<tr>
<td>Sync pulse</td>
<td>80</td>
<td>1.6161616161616</td>
</tr>
<tr>
<td>Back porch</td>
<td>160</td>
<td>3.2323232323232</td>
</tr>
<tr>
<td>Whole line</td>
<td>1056</td>
<td>21.333333333333</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frame part</th>
<th>Lines</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible area</td>
<td>600</td>
<td>12.8</td>
</tr>
<tr>
<td>Front porch</td>
<td>1</td>
<td>0.021333333333333</td>
</tr>
<tr>
<td>Sync pulse</td>
<td>3</td>
<td>0.064</td>
</tr>
<tr>
<td>Back porch</td>
<td>21</td>
<td>0.448</td>
</tr>
<tr>
<td>Whole frame</td>
<td>625</td>
<td>13.333333333333</td>
</tr>
</tbody>
</table>

http://www.xess.com/blog/vga-the-rest-of-the-story/
http://www.tinyvga.com/vga-timing/800x600@75Hz
More generally, how does software interface to I/O devices?

Analog Devices AD9980 Display Interface
Analog Devices AD9980 Display Interface

VGA In DVI Out Subsystem

USB
WebCam

Host
PC

VGA Interface

Frame Buffer

SRAM

DVI Interface
Memory Mapped Framebuffer

- A range of memory addresses correspond to the display.
- Write: VGA In, MicroBlaze, Read: DVI Out, MicroBlaze
  
  (# of ports?, # of clocks?, WB)
- CPU writes to memory location pixel values to change display.
- No handshaking required. Independent process reads pixels from memory and sends them to the display interface at the required rate.

**Example MicroBlaze address map**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFFFFFF</td>
<td>800 pixels/line X 600 lines</td>
</tr>
<tr>
<td>0x801D4BFC</td>
<td>Display Origin: Increasing X values to the right. Increasing Y values down.</td>
</tr>
<tr>
<td>0x80000000</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

External SRAM

8Mbits / 480000 = 17.5 bits/pixel max!

If we choose 16 bits/pixel?

{ Red[4:0] ; Green[5:0] ; Blue[4:0] }

---

Framebuffer Details

- Four 8 bit pixel values per 32 bit memory word.

**Example MicroBlaze address map**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFFFFFF</td>
<td>600 lines, 800 pixels/line = 480,000 memory locations</td>
</tr>
<tr>
<td>0x801D4BFC</td>
<td></td>
</tr>
<tr>
<td>0x80000000</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

XUP SRAM memory capacity: ~8 Mbits (in external SRAM).

8Mbits / 480000 = 17.5 bits/pixel max!

We choose 8 bits/pixel- gray scale

- Note, that we assign four 8 bit pixels per memory address.
- Four pixel address map to one address in the SRAM (it is 32bits wide).
Framebuffer Implementation

- Framebuffer like a quad-ported memory. Four independent processes access framebuffer:

  - CPU reads/writes pixel locations. Could be in random order, e.g. drawing an object, or sequentially, e.g. clearing the screen.

  - Video Out Interface continuously reads pixel locations in scan-line order and sends to physical display. (72 Hz?)

  - Video In Interface continuously writes locations in scan-line order. (75 Hz?)

- How big is this memory and how do we implement it?
  For us:
  
  2 Frames x 800 x 600 pixels/frame x 8 bits/pixel

Frame Buffer Implementation

- Which XUP memory resource to use?
- Memory Capacity Summary:
  - LUT RAM 110 kB
  - Block RAM: 0.8 MB
  - External SRAM: 8 MB
  - External DRAM: 256 MB

- DDRAM bandwidth: rated at 400 MHz
More generally, how does software interface to I/O devices?

*ZBT (ZBT stands for zero bus turnaround) — the turnaround is the number of clock cycles it takes to change access to the SRAM from write to read and vice versa. The turnaround for ZBT SRAMs or the latency between read and write cycle is zero.

What frame buffer configuration is possible?
VGA In DVI Out Subsystem

Frame Buffer Physical Interface

More generally, how does software interface to I/O devices?

CPU Video Interface Block: accepts pixel values from FB, streams pixels values and control signals to physical device.

Processor Side: provides a memory mapped programming interface to video display.
Physical Video Interface

- **DVI connector:** accommodates analog and digital formats

  - DVI Transmitter Chip, Chrontel 7301C.
  - Implements standard signaling voltage levels for video monitors.
  - Digital to analog conversion for analog display formats.

2 way Video Interface Details

- **Physical Interface:**
  - Tri-state Buffers
  - CPU
  - SRAM
  - Address Mux
  - Video Interface
  - Xilinx I/O buffer
2 way Video Interface Details

- **Timing:**
  - All CPU frame buffer writes go through FIFO (and crosses clock domain boundary).
  - Store Buffer writes to SRAM 3/4 SRAM cycles.
  - Can Store Buffer fill up? What if CPU runs at lower clock rate?

   - Timing: ~49.5 Mpix/sec
   - Store Buffer writes to SRAM 3/4 SRAM cycles.
   - Can Store Buffer fill up? What if CPU runs at lower clock rate?

2 way Video Interface Details

- **Address Translation:**
  - Reads in pixel number order, 4 per address
  - CPU writes need translation to convert from 20-bit frame buffer address to 19-bit SRAM address:
    \[ PN = X + 800*Y \]
  - How to do this on FPGA? \[ 800 = 0x320 = 512 + 256 + 32 \]
  - How do we write a single pixel? Use SRAM “Byte Write Enables”
example MicroBlaze Clear Routine

// 1024K bytes = 256K 32 bit words
// 600 lines by 800 pixels/line
#include <stdio.h>
long *frame_buffer1, *frame_buffer2;

#define BLACK4 0x00000000

void clear_frame()
{ int row, col;
  frame_buffer1 = (long *) 0x80000000; // check addressing
  frame_buffer2 = (long *) 0x80020000; // check addressing
  for(row =0; row < 600; row++)
    for(col=0; col < 200; col++)
      { *(frame_buffer1 + 800 * row + col) = BLACK4;
       *(frame_buffer2 + 800 * row + col) = BLACK4;
      }
}

Conclusion

• VGA video timing, input/output CODEC
• Frame Buffer overview