MIPS150 Video Subsystem

- Gives software ability to display information on screen.
- Equivalent to standard graphics cards:
  - Processor can directly write the display bit map
  - 2D Graphics acceleration
**Framebuffer** HW/SW Interface

- A range of memory addresses correspond to the display.
- CPU writes (using sw instruction) pixel values to change display.
- No synchronization required. Independent process reads pixels from memory and sends them to the display interface at the required rate.

**CPU address map**

Ex: 1024 pixels/line X 768 lines

**Framebuffer Implementation**

- Framebuffer like a simple dual-ported memory.
  Two independent processes access framebuffer:
  
  CPU writes pixel locations. Could be in random order, e.g. drawing an object, or sequentially, e.g. clearing the screen. 

  Video Interface continuously reads pixel locations in scan-line order and sends to physical display.

- How big is this memory and how do we implement it? For us:
  
  1024 \times 768 \text{ pixels/frame} \times 24 \text{ bits/pixel}
Memory Mapped Framebuffer

**MIPS address map**

- 0x00000000
- 0x80000000
- 0x8023FFFF
- 0xFFFFFFFF

**Framebuffer**

- 1024 pixels/line X 768 lines
- 0x80000000 = 0
- 0x8023FFFF = (1023, 767)
- Display Origin: Increasing X values to the right. Increasing Y values down.

- 1024 * 768 = 786,432 pixels
- We choose 24 bits/pixel
  \{ Red[7:0] ; Green[7:0] ; Blue[7:0] \}
  \[786,432 * 3 = 2,359,296\] Bytes

- Total memory bandwidth needed to support framebuffer?

Frame Buffer Implementation

- Which XUP memory resource to use?
- Memory Capacity Summary:
  - LUT RAM
  - Block RAM
  - External SRAM
  - External DRAM

- DRAM bandwidth:
Framebuffer Details

XUP DRAM
memory capacity:
256 MBytes (in external DRAM).

With Byte addressed memory, best to use 4 Bytes/pixel

Starting each line on a multiple of 4K leads to independent X and Y address: \{Y[9:0] ; X[11:2]\}
Y == row number, X == pixel in row

Frame Buffer Physical Interface

**Processor Side:** provides a memory mapped programming interface to video display.

You do!

**DRAM “Hub”:** arbitrates among multiple DRAM users.

You do!

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

We do!
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.

Framebuffer Details 2009

- One pixel value per memory location.

**MIPS address map**

<table>
<thead>
<tr>
<th>Address</th>
<th>Framebuffer</th>
<th>Memory Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFFFFFF</td>
<td>768 lines, 1K pixels/line</td>
<td>786,432</td>
</tr>
<tr>
<td>0x803FFFFC</td>
<td>1K</td>
<td></td>
</tr>
<tr>
<td>0x80000000</td>
<td>1K</td>
<td>Virtex-5 LX110T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>memory capacity:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5,328 Kbits (in block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAMs).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(5,328 X 1024 bits) / 786432 =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.9 bits/pixel max!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>We choose 4 bits/pixel</td>
</tr>
</tbody>
</table>

- Note, that with only 4 bits/pixel, we could assign more than one pixel per memory location. Ruled out by us, as it complicated software.
Color Map

4 bits per pixel, allows software to assign each screen location, one of 16 different colors.

However, physical display interface uses 8 bits / pixel-color. Therefore entire pallet is $2^{24}$ colors.

Color Map converts 4 bit pixel values to 24 bit colors.

Color map is memory mapped to CPU address space, so software can set the color table. Addresses: $0x8040_0000 \quad 0x8040_003C$, one 24-bit entry per memory address.

Memory Mapped Framebuffer 2010

- A range of memory addresses correspond to the display.
- CPU writes (using sw instruction) 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.

MIPS address map

800 pixels/line X 600 lines

Display Origin: Increasing X values to the right. Increasing Y values down.

8Mbits / 480000 = 17.5 bits/pixel max!

We choose 16 bits/pixel
{ Red[4:0] ; Green[5:0] ; Blue[4:0] }
Framebuffer Details

XUP Board External SRAM

• Note, that we assign only one 16 bit pixel per memory location.
• Two pixel address map to one address in the SRAM (it is 32bits wide).
• Only part of the mapped memory range occupied with physical memory.

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

600 lines, 800 pixels/line = 480,000 memory locations

• Starting each line on a multiple of 1K leads to independent X and Y address:
  \( \{ Y[9:0] ; X[9:0] \} \)
  Y == row number, X == pixel in row

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.

"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 and 512K x 18
9Mb, PIPELINE 'NO WAIT' STATE BUS
SRAM

What frame buffer configuration is possible?