EECS150 - Digital Design

Lecture 14 - Project Description, Part 3

March 4, 2010

John Wawrzynek

Verilog Memory Synthesis Notes

- Block RAMS and LUT RAMS all exist as primitive library elements (similar to FDRSE) and can be instantiated. However, it is much more convenient to **use inference**.
- Depending on how you write your verilog, you will get either a collection of block RAMs, a collection of LUT RAMs, or a collection of flip-flops.
- The synthesizer uses size, and read style (synch versus asynch) to determine the best primitive type to use.
- It is possible to force mapping to a particular primitive by using synthesis directives. However, if you write your verilog correctly, you will not need to use directives.
- The synthesizer has limited capabilities (eg., it can combine primitives for more depth and width, but is limited on porting options). Be careful, as you might not get what you want.
- See **Synplify User Guide**, and **XST User Guide** for examples.

Inferring RAMs in Verilog

```
// 64X1 RAM implementation using distributed RAM

 module ram64X1 (clk, we, d, addr, q);
  input clk, we, d;
  input [5:0] addr;
  output q;

 reg [63:0] temp;
 always @ (posedge clk)
     if(we)
        temp[addr] <= d;
 assign q = temp[addr];

 endmodule
```

Verilog reg array used with "always @ (posedge ... infers memory array.

Asynchronous read infers LUT RAM

Dual-read-port LUT RAM

```
 //
 // Multiple-Port RAM Descriptions
 //
 module v_rams_17 (clk, we, wa, ra1, ra2, di, do1, do2);
     input  clk;
     input  we;
     input  [5:0] wa;
     input  [5:0] ra1;
     input  [5:0] ra2;
     input  [15:0] di;
     output [15:0] do1;
     output [15:0] do2;
     reg    [15:0] ram [63:0];
     always @(posedge clk)
     begin
         if (we)
             ram[wa] <= di;
     end
     assign do1 = ram[ra1];
     assign do2 = ram[ra2];
 endmodule
```

Multiple reference to same array.

Block RAM Inference

```
//
// Single-Port RAM with Synchronous Read
//
module v_rams_07 (clk, we, a, di, do);
    input  clk;
    input  we;
    input  [5:0] a;
    input  [15:0] di;
    output [15:0] do;
    reg    [15:0] ram [63:0];
    reg    [5:0] read_a;
    always @(posedge clk) begin
        if (we)
            ram[a] <= di;
        read_a <= a;
    end
    assign do = ram[read_a];
endmodule
```

Synchronous read (registered read address) infers Block RAM

Block RAM initialization

```
module RAMB4_S4 (data_out, ADDR, data_in, CLK, WE);
   output[3:0] data_out;
   input [2:0] ADDR;
   input [3:0] data_in;
   input CLK, WE;
   reg [3:0] mem [7:0];
   reg [3:0] read_addr;

   initial
     begin
       $readmemb("data.dat", mem);
     end

   always@(posedge CLK)
     read_addr <= ADDR;

   assign data_out = mem[read_addr];

   always @(posedge CLK)
     if (WE) mem[ADDR] = data_in;

   endmodule
```

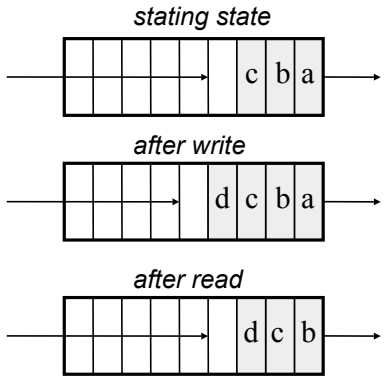
"data.dat" contains initial RAM contents, it gets put into the bitfile and loaded at configuration time. (Remake bits to change contents)

Dual-Port Block RAM

```
module test (data0,data1,waddr0,waddr1,we0,we1,clk0, clk1, q0, q1);

   parameter d_width = 8;  parameter addr_width = 8; parameter mem_depth = 256;

   input [d_width-1:0] data0, data1;
   input [addr_width-1:0] waddr0, waddr1;
   input we0, we1, clk0, clk1;

   reg [d_width-1:0] mem [mem_depth-1:0]
   reg [addr_width-1:0] reg_waddr0, reg_waddr1;
   output [d_width-1:0] q0, q1;

   assign q0 = mem[reg_waddr0];
   assign q1 = mem[reg_waddr1];

   always @(posedge clk0)
     begin
       if (we0)
         mem[waddr0] <= data0;
         reg_waddr0 <= waddr0;
     end

   always @(posedge clk1)
     begin
       if (we1)
         mem[waddr1] <= data1;
         reg_waddr1 <= waddr1;
     end

   endmodule
```

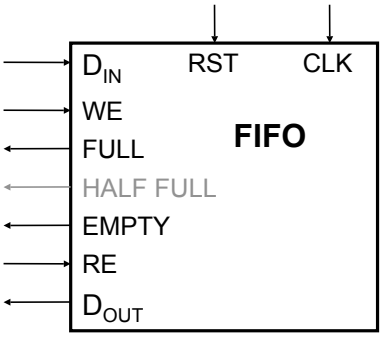

First-in-first-out (FIFO) Memory

- Used to implement *queues*.
- These find common use in computers and communication circuits.
- Generally, used to "decouple" actions of producer and consumer:



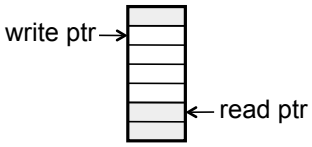
- Producer can perform many writes without consumer performing any reads (or vis versa). However, because of finite buffer size, on average, need equal number of reads and writes.
- Typical uses:
  - interfacing I/O devices. Example network interface. Data bursts from network, then processor bursts to memory buffer (or reads one word at a time from interface). Operations not synchronized.
  - Example: Audio output. Processor produces output samples in bursts (during process swap-in time). Audio DAC clocks it out at constant sample rate.

FIFO Interfaces

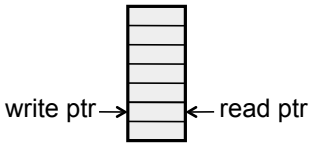


- After write or read operation, FULL and EMPTY indicate status of buffer.
- Used by external logic to control own reading from or writing to the buffer.
- FIFO resets to EMPTY state.
- HALF FULL (or other indicator of partial fullness) is optional.

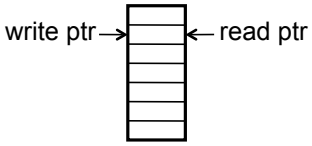
- Address pointers are used internally to keep next write position and next read position into a dual-port memory.



- If pointers equal after write ⇒ FULL:



- If pointers equal after read ⇒ EMPTY:



FIFO Implementation Details

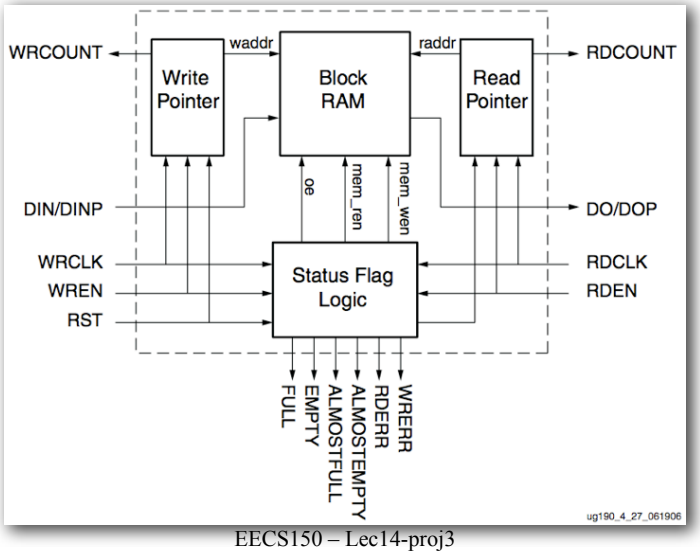
- Assume, dual-port memory with asynchronous read, synchronous write.
- Binary counter for each of read and write address. CEs (count enable) controlled by WE and RE.
- Equal comparator to see when pointers match.
- Flip-flop each for FULL and EMPTY flags:

WE	RE	equal	$EMPTY_i$	$FULL_i$
0	0	0	0	0
0	0	1	$EMPTY_{i-1}$	$FULL_{i-1}$
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	$EMPTY_{i-1}$	$FULL_{i-1}$

- Control logic (FSM) with truth-table shown to left.

Xilinx Virtex5 FIFOs

- Virtex5 BlockRAMS include dedicated circuits for FIFOs.
- Details in User Guide (ug190).
- Takes advantage of separate dual ports and independent ports clocks.

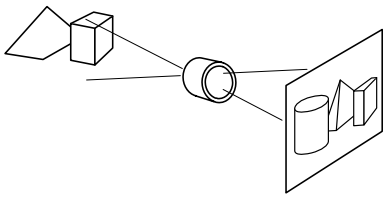


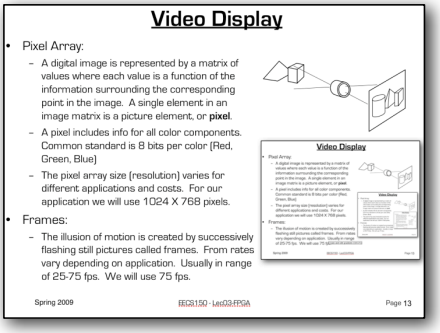
Processor Design Considerations (1/2)

- **Register File: Consider distributed RAM (LUT RAM)**
  - Size is close to what is needed: distributed RAM primitive configurations are 32 or 64 bits deep. Extra width is easily achieved by parallel arrangements.
  - LUT-RAM configurations offer multi-porting options - useful for register files.
  - Asynchronous read, might be useful by providing flexibility on where to put register read in the pipeline.
- **Instruction / Data Memories : Consider Block RAM**
  - Higher density, lower cost for large number of bits
  - A single 36kbit Block RAM implements 1K 32-bit words.
  - Configuration stream based initialization, permits a simple "boot strap" procedure.
- **Other Memories in Project? Ethernet? Video?**

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.



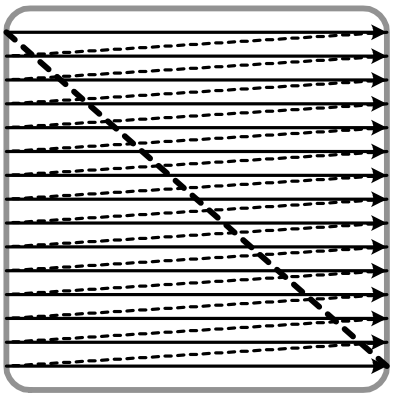


- Frames:

– 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.
  - Display devices have limited "persistence", therefore frames must be repetitively sent, to create a stable image. Display devices don't typically store the image in memory.
  - Repetitively sending the image also allows motion.
  - For a typical resolution and frame rate:

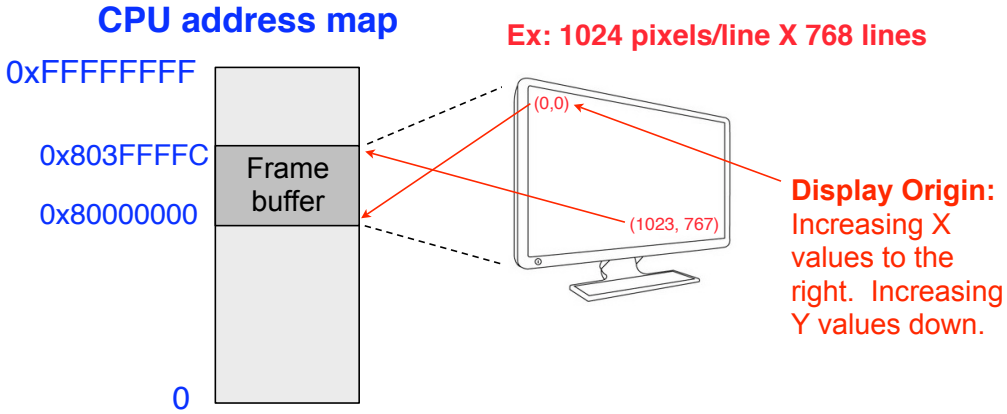
Samsung LCD with analog interface.

Pixels per frame = 1024 X 768 = 786,432
Pixel rate = 75fps X 786432 = 58,982,400 pixels/sec

Note: in this example, we use a pixel clock rate of 78.75 MHz to account for blanking intervals

"Framebuffer" HW/SW Interface

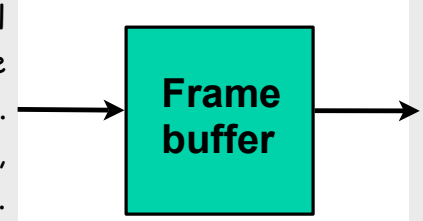
- A range of memory addresses correspond to the display.
- CPU writes (using sw instruction) pixel values to change display.
- No synchronization required. Independent process reads pixels from memory and sends them to the display interface at the required rate.



Framebuffer Implementation

- Framebuffer is 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.



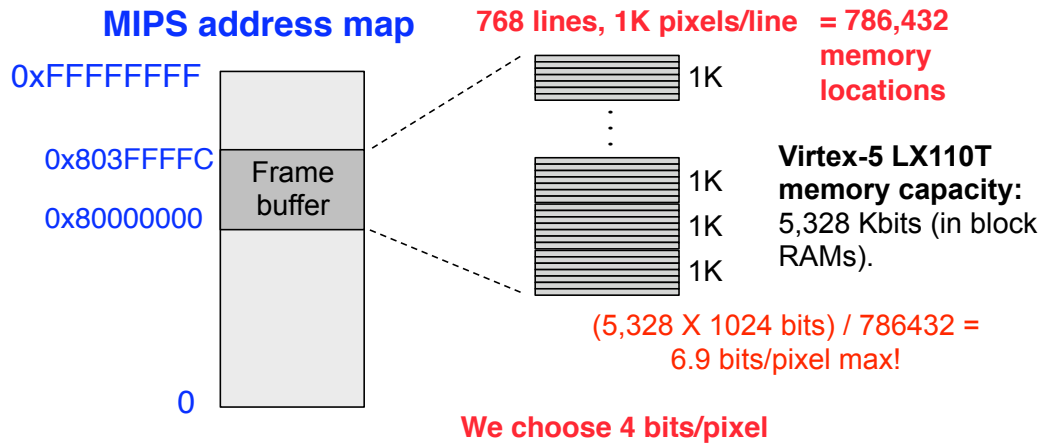
Video Interface continuously reads pixel locations in scan-line order and sends to physical display.

- How big is this memory and how do we implement it?

1024 x 768 pixels/frame x 24 bits/pixel

Framebuffer Details last year

- One pixel value per memory location.



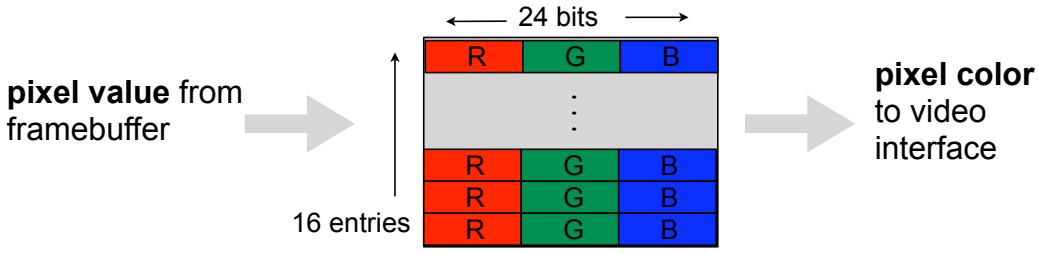
- Note, that with only 4 bits/pixel, we could assign more than one pixel per memory location. Ruled out by us, as it complicated software.

Color Map

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

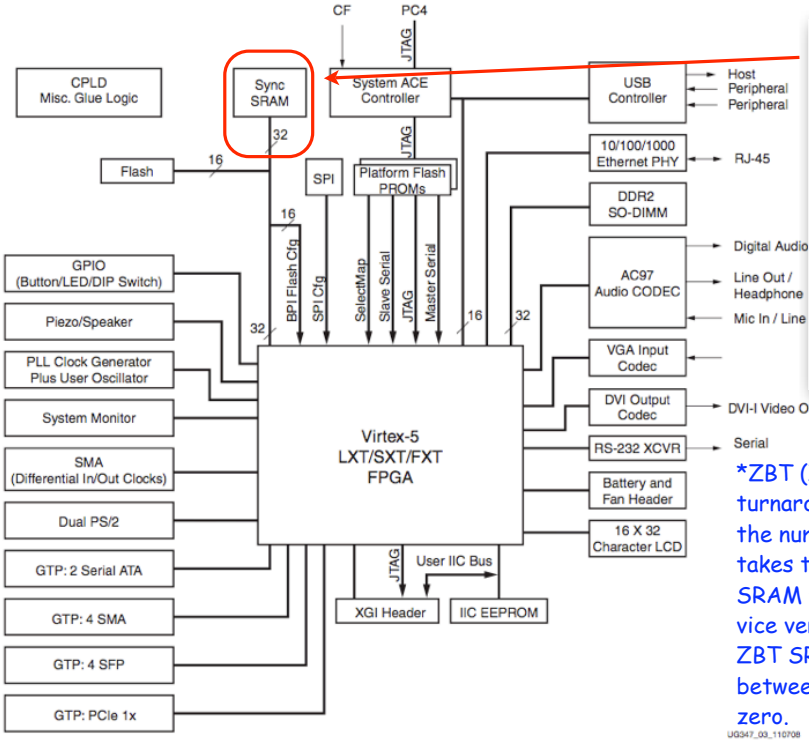
However, physical display interface uses 8 bits / pixel-color. Therefore entire pallet is 2^{24} colors.

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 0x8040_003C**, one 24-bit entry per memory address.

XUP Board External SRAM



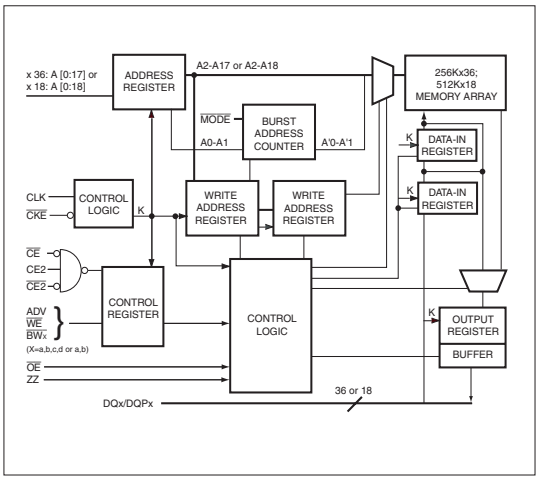
"ZBT" synchronous SRAM, 9 Mb on 32-bit data bus, with four "parity" bits
256K x 36 bits
(located under the removable LCD)

*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.

256K x 36 and 512K x 18
9Mb, PIPELINE 'NO WAIT' STATE BUS SRAM

MARCH 2008

BLOCK DIAGRAM



PIN DESCRIPTIONS

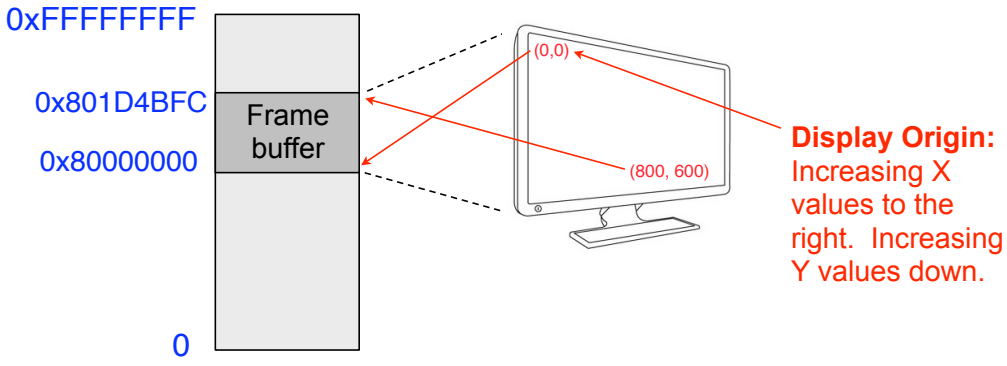
A0, A1	Synchronous Address Inputs. These pins must tied to the two LSBs of the address bus.
A	Synchronous Address Inputs
CLK	Synchronous Clock
ADV	Synchronous Burst Address Advance
$\overline{BW}$a-$\overline{BW}$d	Synchronous Byte Write Enable
$\overline{WE}$	Write Enable
$\overline{CKE}$	Clock Enable
Vss	Ground for Core
NC	Not Connected
$\overline{CE}$, CE2, $\overline{CE2}$	Synchronous Chip Enable
$\overline{OE}$	Output Enable
DQa-DQd	Synchronous Data Input/Output
DQPa-DQPd	Parity Data I/O
MODE	Burst Sequence Selection
V_{DD}	+3.3V/2.5V Power Supply
V_{SS}	Ground for output Buffer
V_{DDQ}	Isolated Output Buffer Supply: +3.3V/2.5V
ZZ	Snooze Enable

What frame buffer configuration is possible?

Memory Mapped Framebuffer

- 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

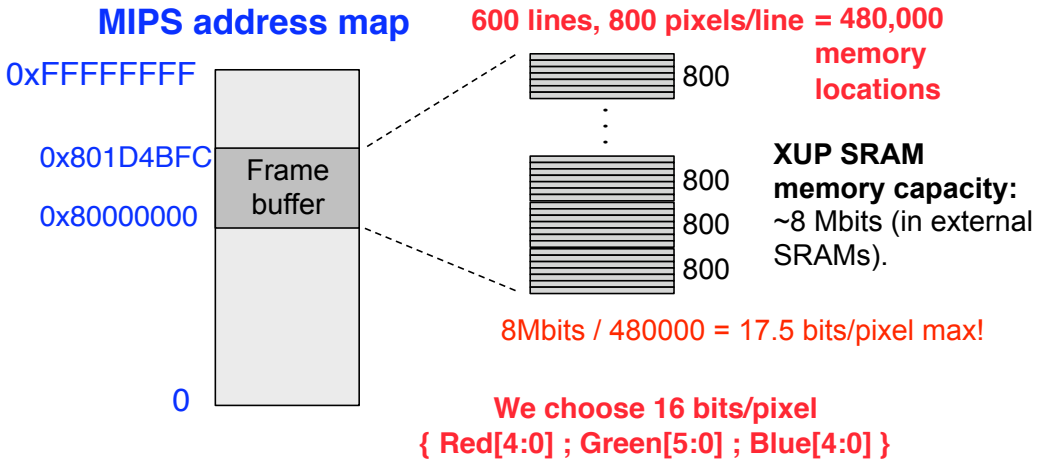


800 pixels/line X 600 lines

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

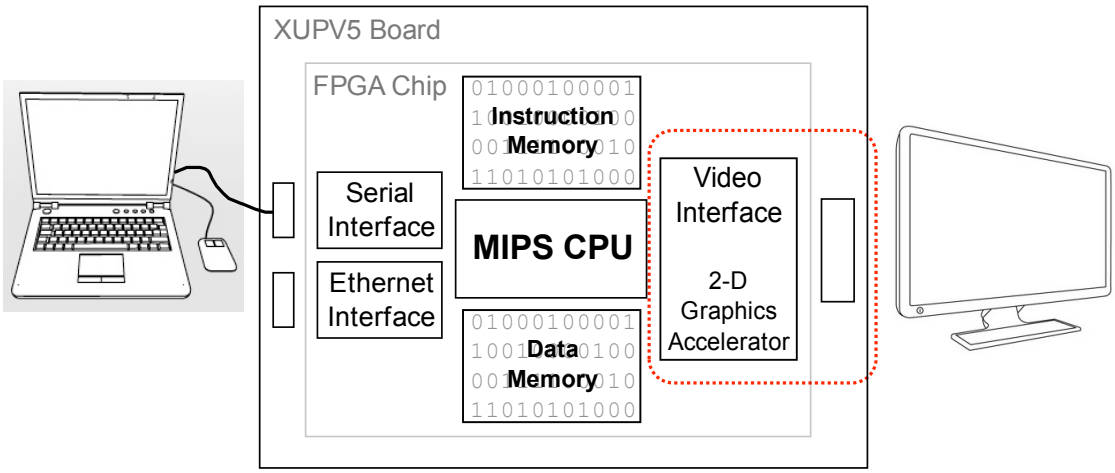
Framebuffer Details

- One pixel value per memory location.



- 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).

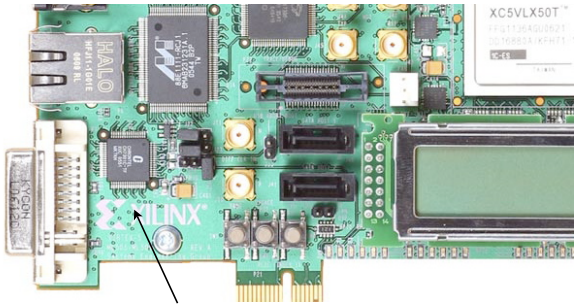
MIPS150 Video Subsystem



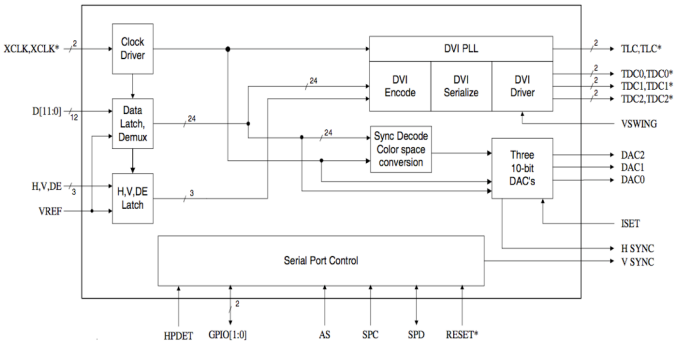
- Gives software ability to display information on screen.
- Equivalent to standard graphics cards:
  - Processor can directly write the display bit map
  - 2D Graphics acceleration

Physical Video Interface

DVI connector: accommodates analog and digital formats

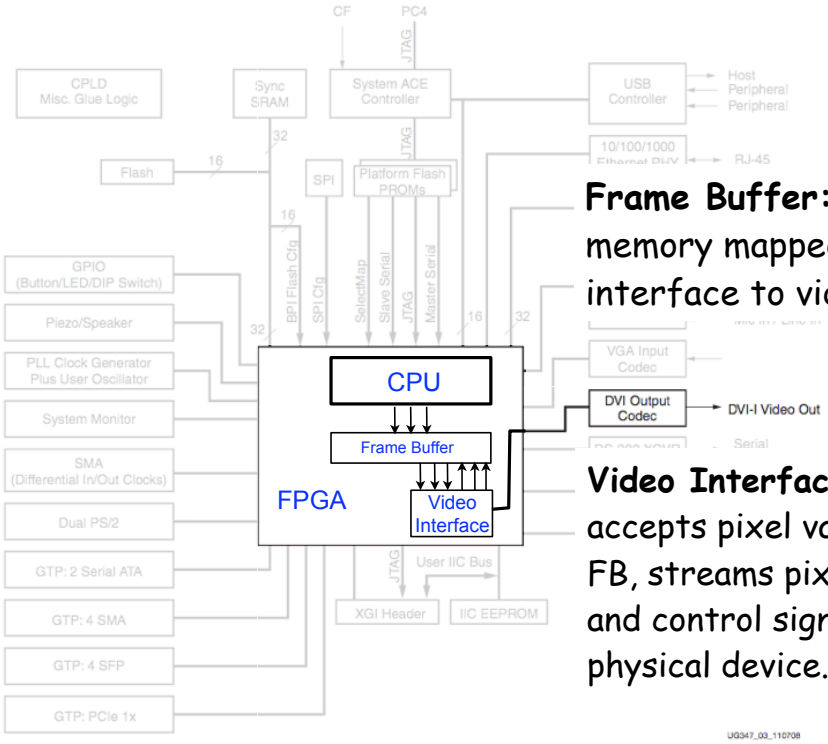


DVI Transmitter Chip, Chrontel 7301C.



Implements standard signaling voltage levels for video monitors. Digital to analog conversion for analog display formats.

Video Interface



Frame Buffer: provides a memory mapped programming interface to video display.

You do!

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

We do!