Projecto e Controlo em Lógica Digital

www.lip.pt/~pedjor/PCLD

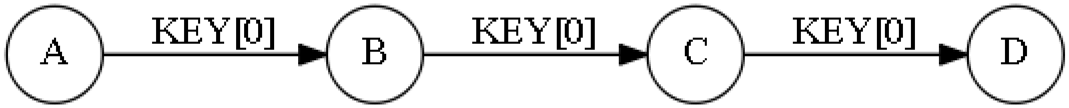
- VGA display

Refs:
Cyclone II device Handbook, Altera corp.
Quartus II Handbook , Altera corp.
DE2 documentation
Verilog HDL, S. Palnitkar, Prentice Hall

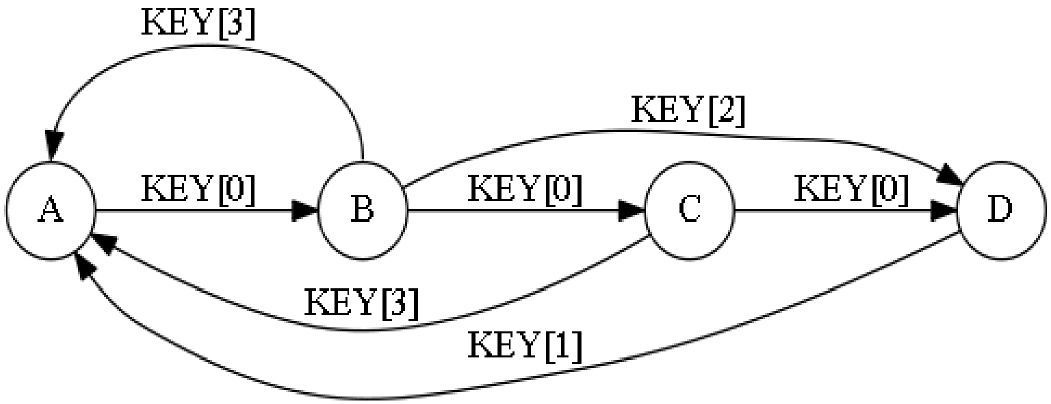
Inspired on course 6.111 from MIT * Some support material taken from 6.111 (thank you)

State Machines

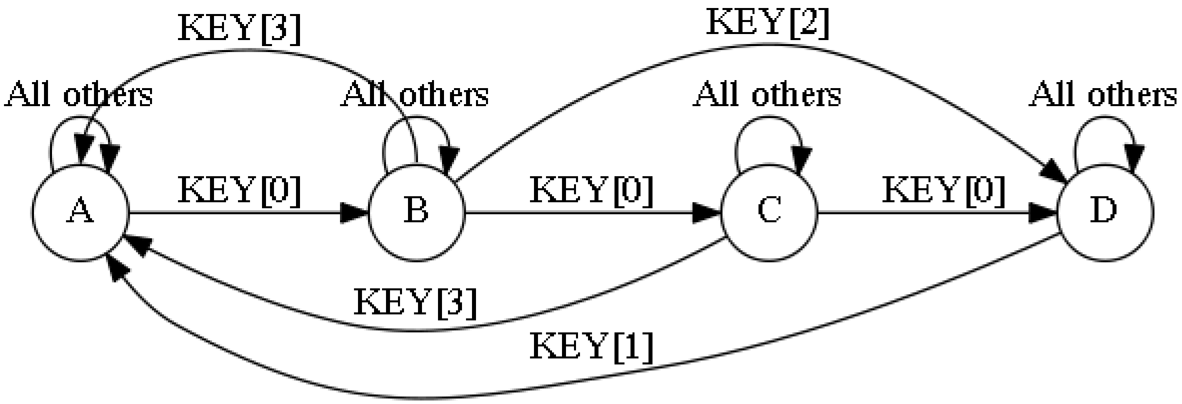
Which clock

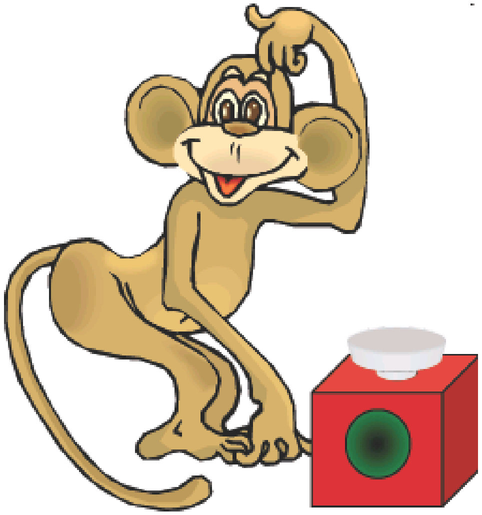


Which clock



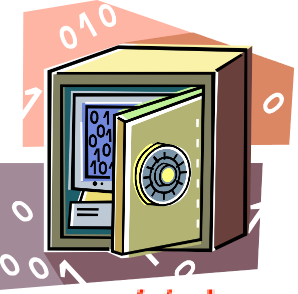
For completeness!





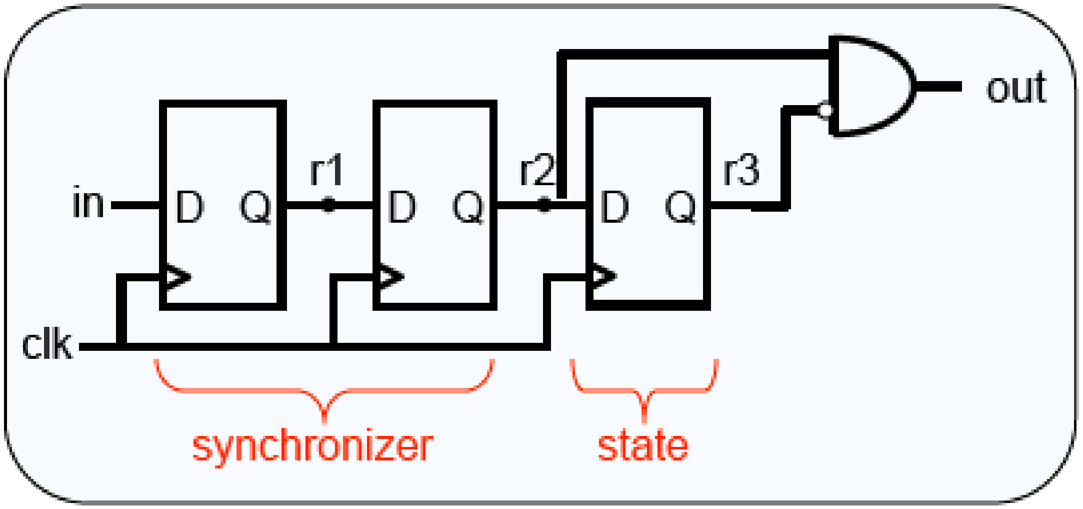
On/off button: final answer

```
module onoff_sync(input clk, reset, button_in,
                  output reg light);
  // synchronizer
  reg button,btemp;
  always @(posedge clk)
    {button,btemp} <= {btemp,button_in};

  // debounce push button
  wire bpressed;
  debounce db1(.clock(clk),.reset(reset),
               .noisy(button),.clean(bpressed));

  reg old_bpressed;  // state last clk cycle
  always @ (posedge clk) begin
    if (reset)
      begin light <= 0; old_bpressed <= 0; end
    else if (old_bpressed==0 && bpressed==1)
      // button changed from 0 to 1
      light <= ~light;
    old_bpressed <= bpressed;
  end
endmodule
```

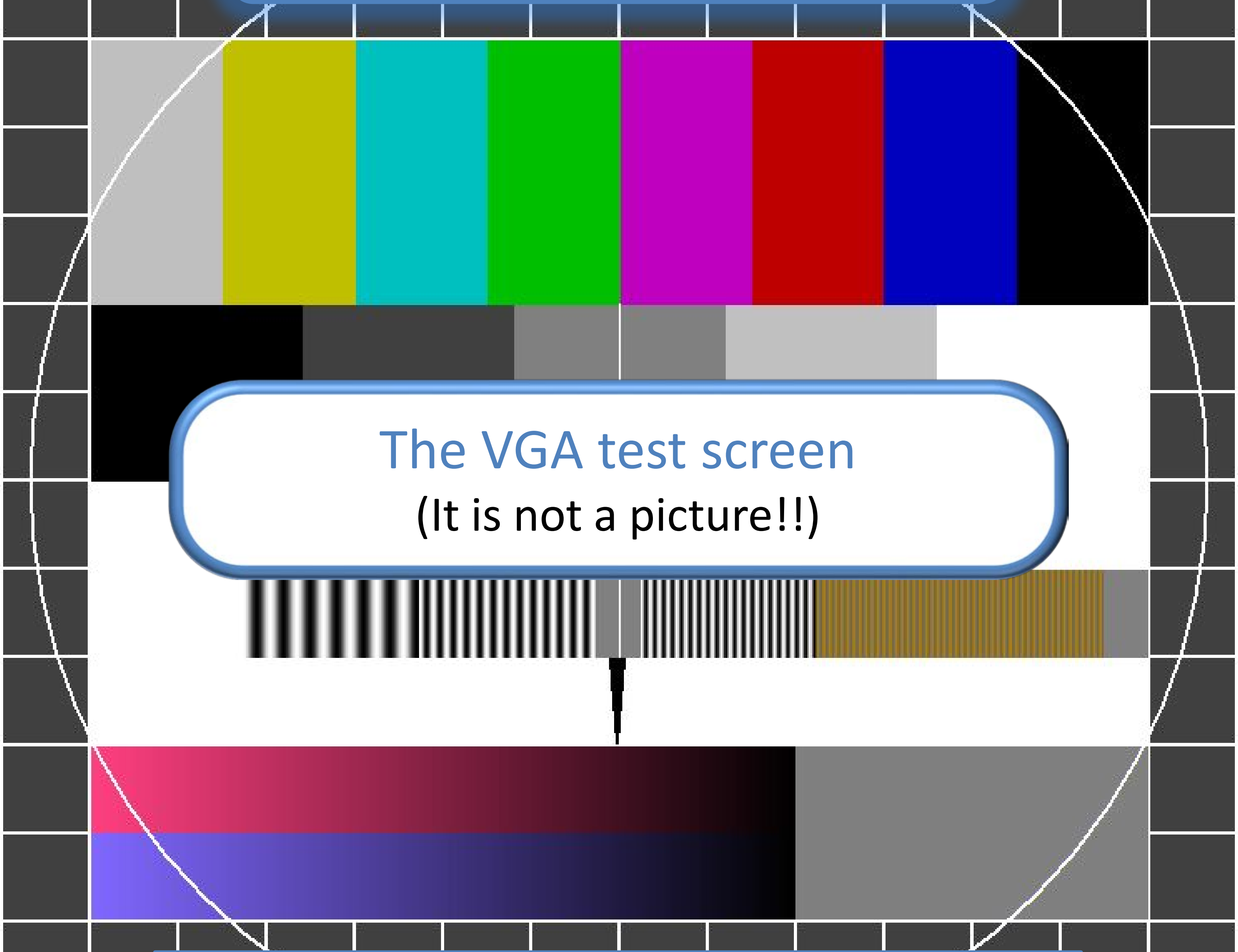


Step 2A: Synchronize buttons

```
// button
// push button synchronizer and level-to-pulse converter
// OUT goes high for one cycle of CLK whenever IN makes a
// low-to-high transition.

module button(
  input clk,in,
  output out
);
  reg r1,r2,r3;
  always @(posedge clk)
  begin
    r1 <= in;   // first reg in synchronizer
    r2 <= r1;   // second reg in synchronizer, output is in sync!
    r3 <= r2;   // remembers previous state of button
  end

  // rising edge = old value is 0, new value is 1
  assign out = ~r3 & r2;
endmodule
```



The 4th lab



The VGA test screen

(It is not a picture!!)

Remember to split the program in blocks!!
(Will be usefull for your next project)

Applications of Flat-Panel Displays

SMALL FORMAT

Medical Defibrillator

MP3 Player

Personal Digital Assistant

Car Navigation & Entertainment

LARGE FORMAT

Courtesy of PixTech

Desktop Monitor (color)

Electronic Book

Large Screen Television (color)

Some Display Terminologies

Term	Definition
Pixel	Picture element—The smallest unit that can be addressed to give color and intensity
Pixel Matrix	Number of Rows by the Number of Columns of pixels that make up the deisplay
Aspect Ratio	Ratio of display width to display height; for example 4:3, 16:9
Resolution (ppi)	Number of pixels per unit length (ppi=pixels per inch)
Frame Rate (Hz)	Number of Frames displayed per second
Viewing Angle (°)	Angular range over which images from the display could be viewed without distortion
Diagonal Size	Length of display diagonal
Contrast Ratio	Ratio of the highest luminance (brightest) to the lowest luminance (darkest)

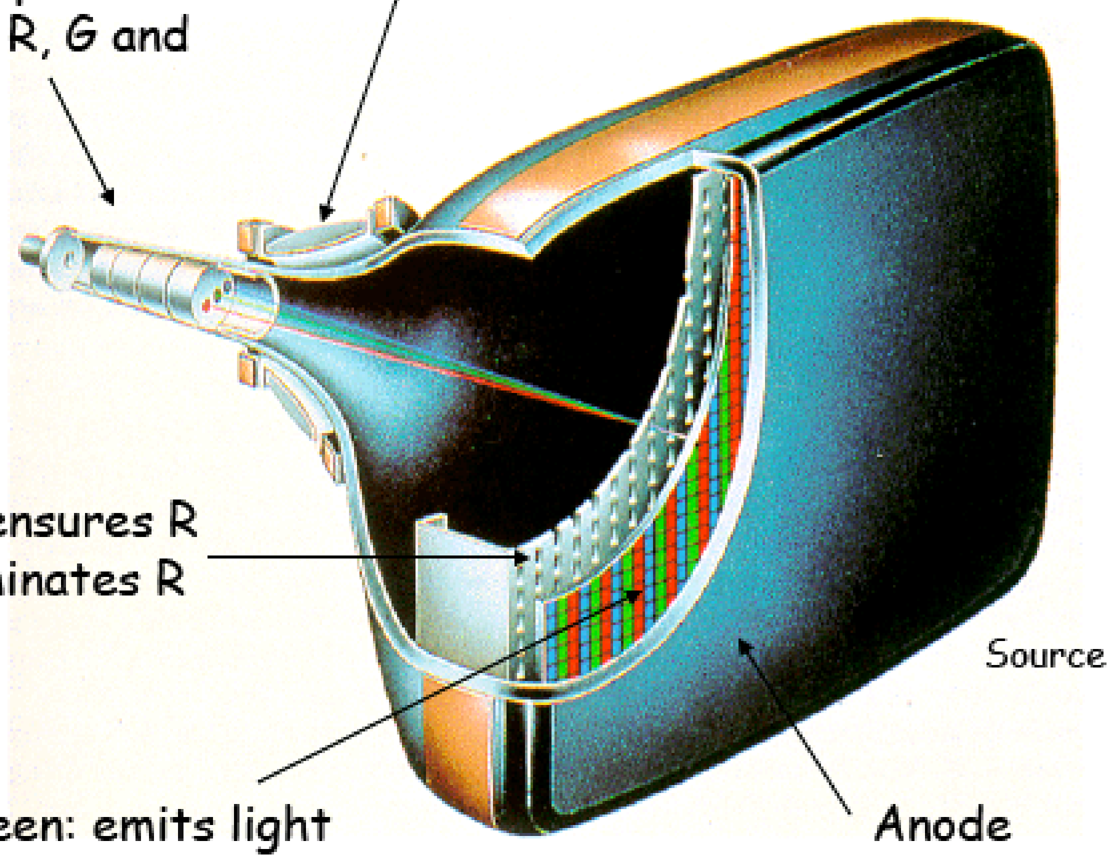
Classifications of Displays by Technology

- Displays could be classified into two broad categories
  - Light Generation (**Emissive Displays**)
  - Light Modulation (**Light Valve Displays**)

- **Emissive Displays** generate photons from electrical excitation of the picture element (pixels)
  - Cathode Ray Tubes (CRTs), Organic Light Emitting Displays (OLEDs), Plasma Displays (PDs)

- **Light Valve Displays** spatially and temporally modulate the intensity pattern of the picture elements (pixels)
  - Liquid Crystal Displays (LCDs), Digital Light Processors (DLPs), Electrophoretic Displays (EPDs)

Background: Cathode Ray Tubes



Deflection coil (aka yoke): magnetically steers beam in a left-to-right top-to-bottom pattern. There are separate H and V coils.

Cathode: separate beams for R, G and B

Shadow mask: ensures R beam only illuminates R pixels, etc.

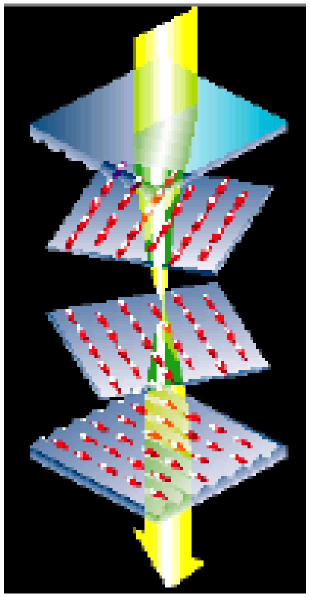
Phosphor Screen: emits light when excited by electron beam, intensity of beam determines brightness

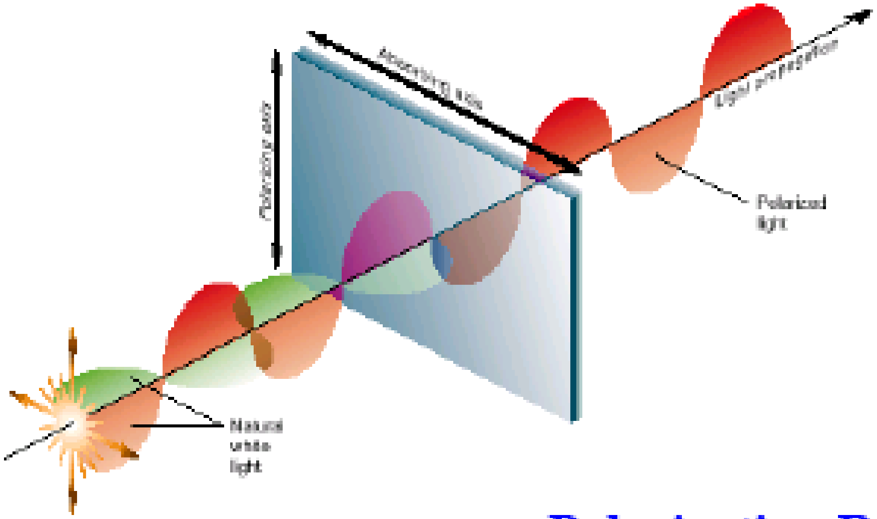
Anode

Source: PixTech

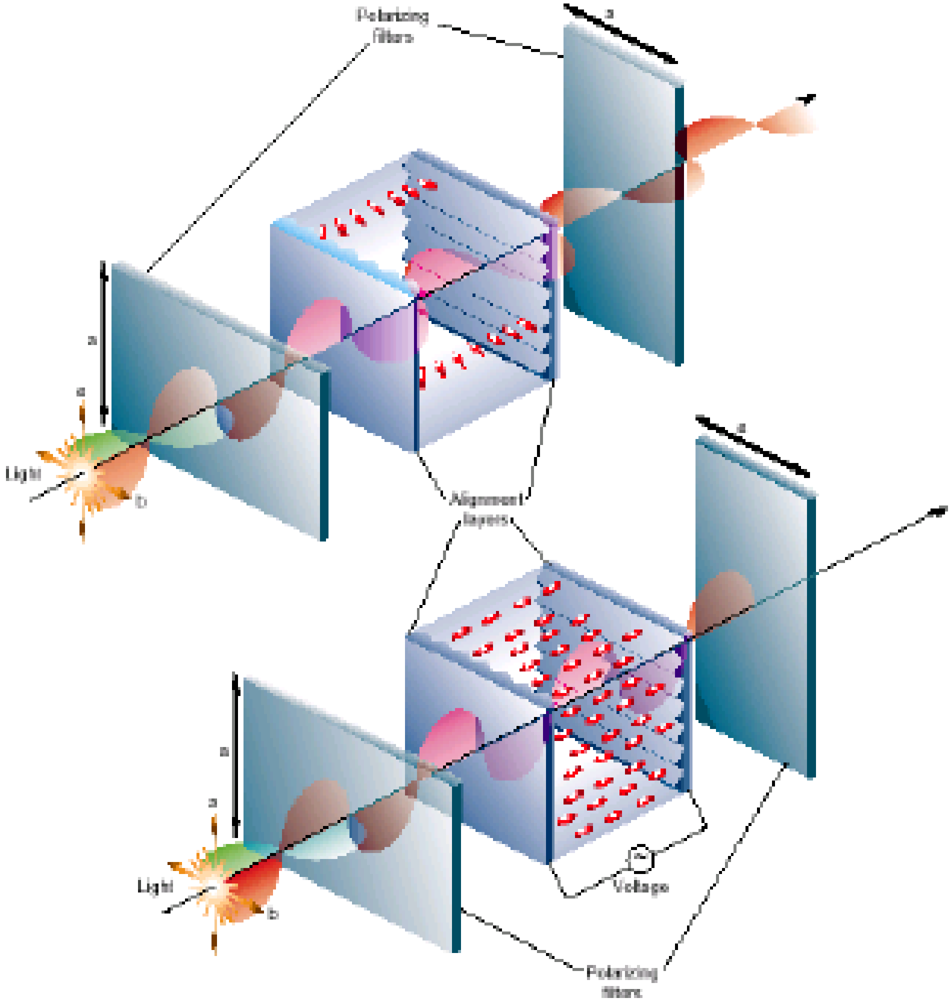
Liquid Crystal Displays

Liquid Crystals rotate the plane of polarization of light when a voltage is applied across the cell





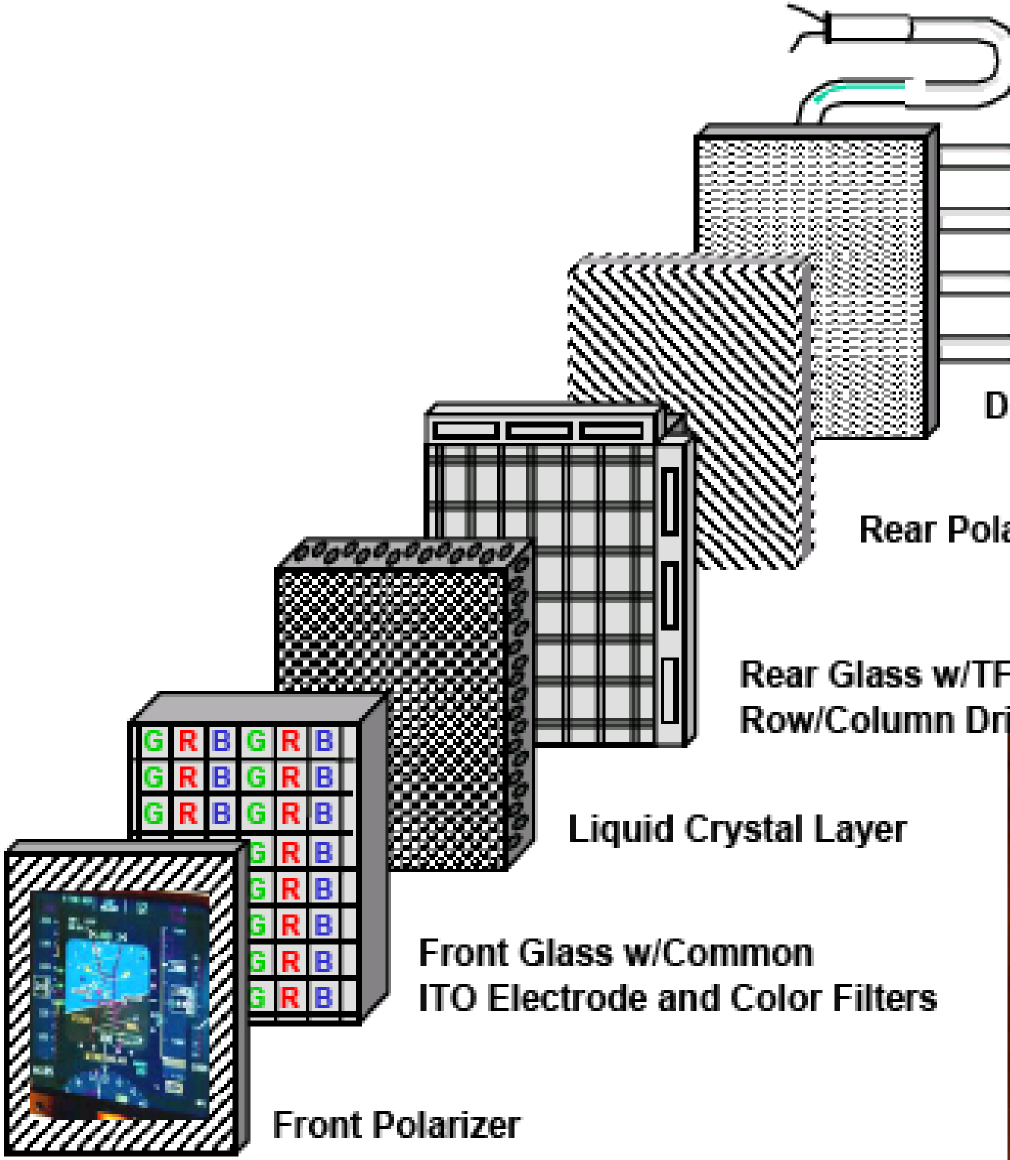
Polarization Rotator



Courtesy of Silicon Graphics

TFT AMLCD

Active Matrix LCD



Fluorescent Lamp
(Backlight)

Diffuser

Rear Polarizer

Rear Glass w/TFT Array and
Row/Column Drivers

Liquid Crystal Layer

Front Glass w/Common
ITO Electrode and Color Filters

Front Polarizer

82” TFT AMLCD

세계 최대 82인치(S-PVA) TFT-LCD 개
세계 최대 82" (S-PVA) TFT-LCD
SAMSUNG

K. Sarma

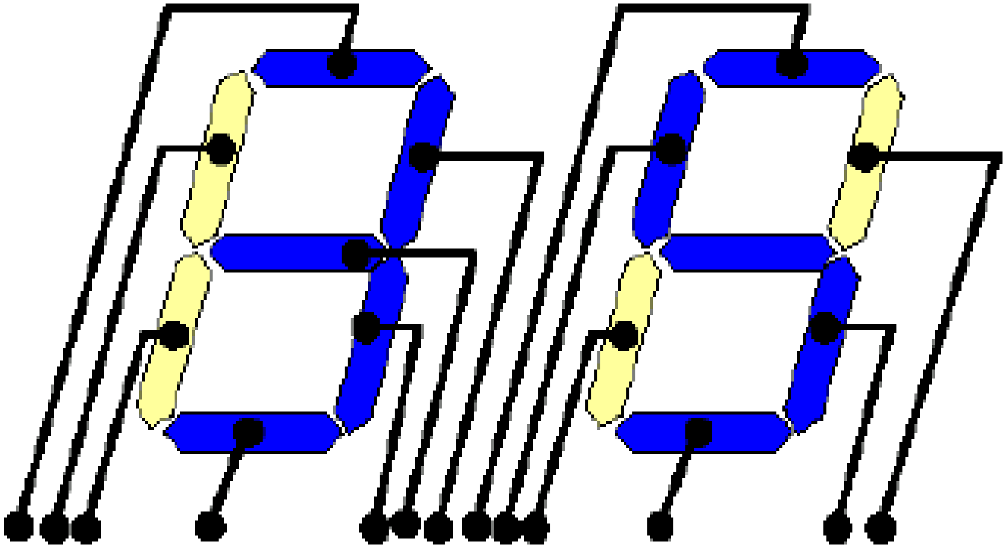
SID 05

Standard Display Addressing Modes

- Sequential Addressing (pixel at a time)
  - CRT, Laser Projection Display
- Matrix Addressing (line at a time)
  - Row scanning, PM LCD, AMLCD, FED, PDPs, OLEDs
- Direct Addressing
  - 7-segment LCD
- Random Addressing
  - Stroke-mode CRT

Direct vs. Matrix Addressing

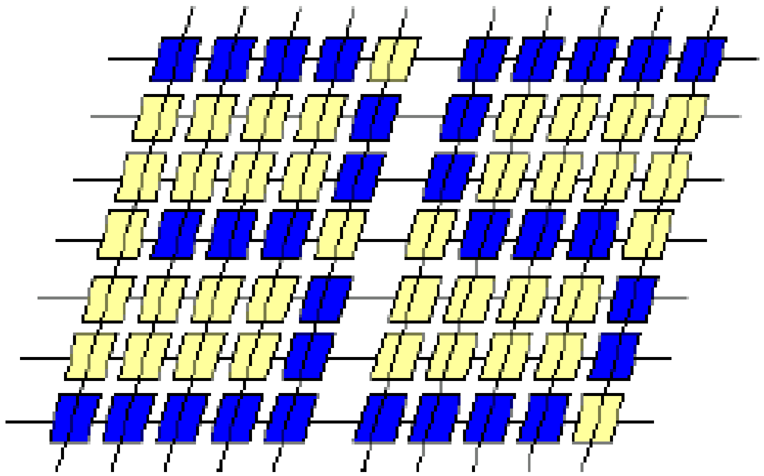
Direct Driving



Segment Display

(7-segment)

Multiplex Driving



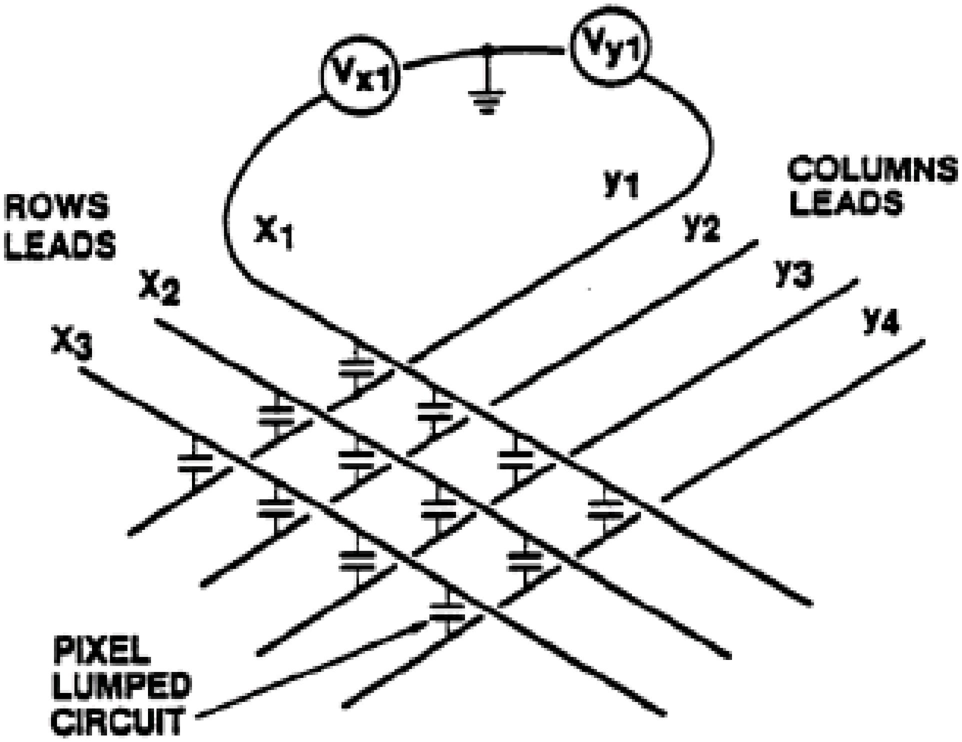
Matrix Display

(dot-matrix)

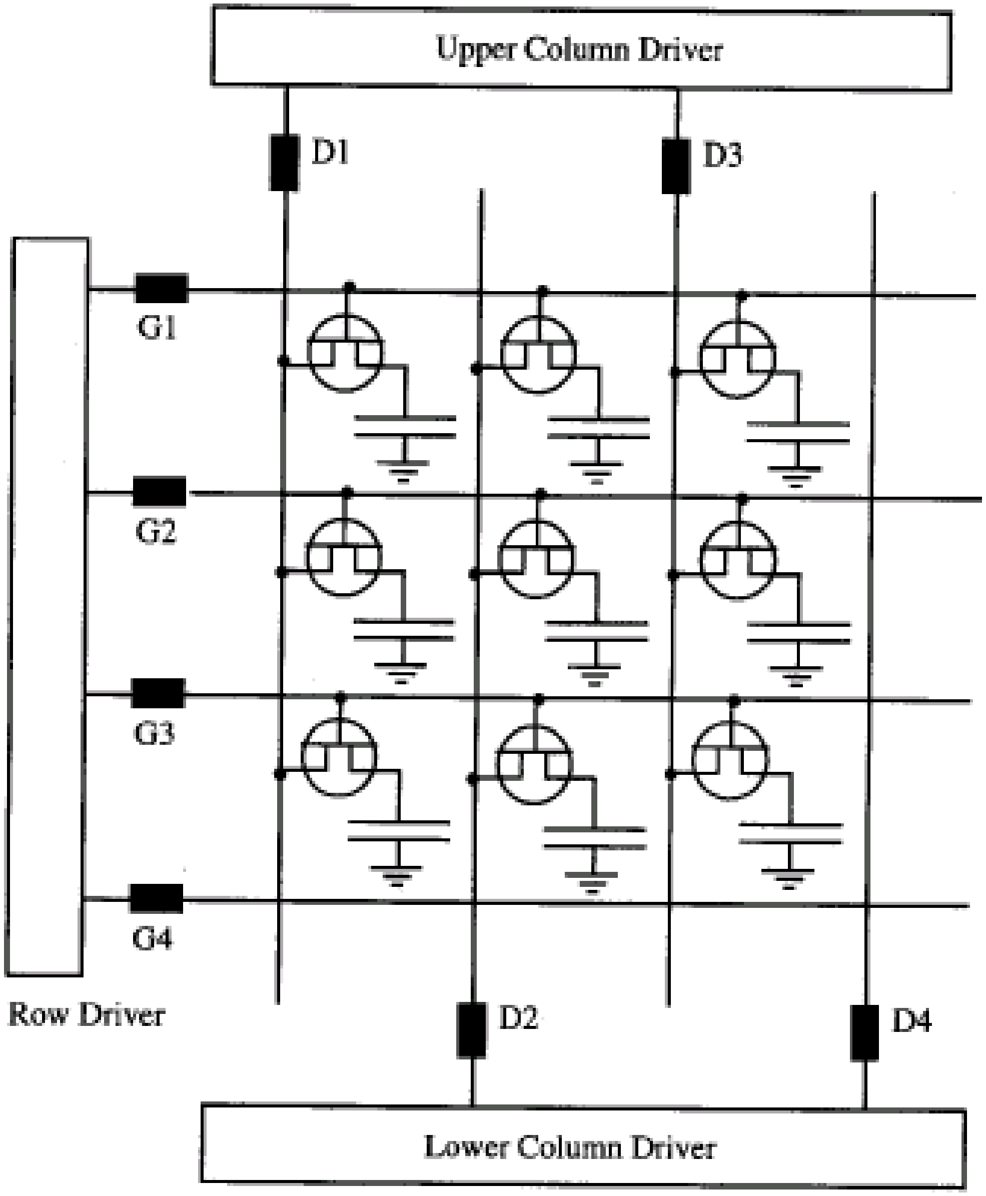
Kim, SID 2001

Matrix Addressing

- Time multiplexed
- Row at a time scanning
  - A column displayed during the time assigned to a row
- For a N rows by M columns display
  - M + N electrodes are required
- Row scanning rate scales with number of rows
- Data rate scales with number of pixels
- Duty cycle scales with number of rows



Active Matrix Addressing



•Introduce non linear device that improves the selection.

•Storage of data values on capacitor so that pixel duty cycle is 100%

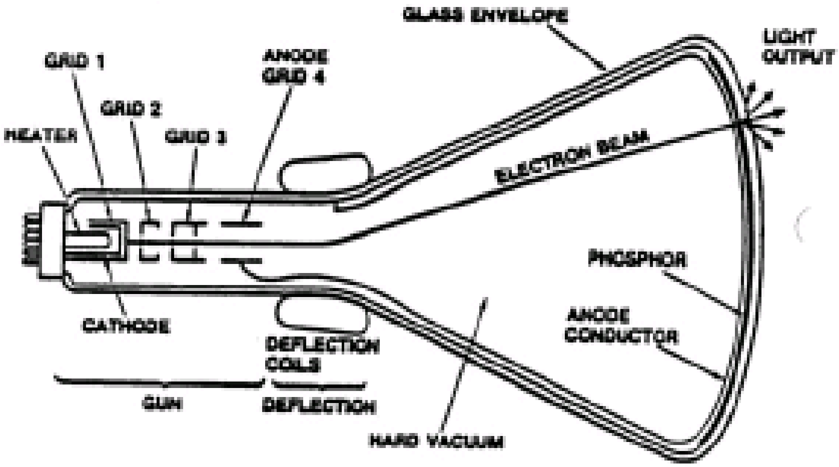
•Improve brightness of display by a factor of N (# of rows) over passive matrix drive

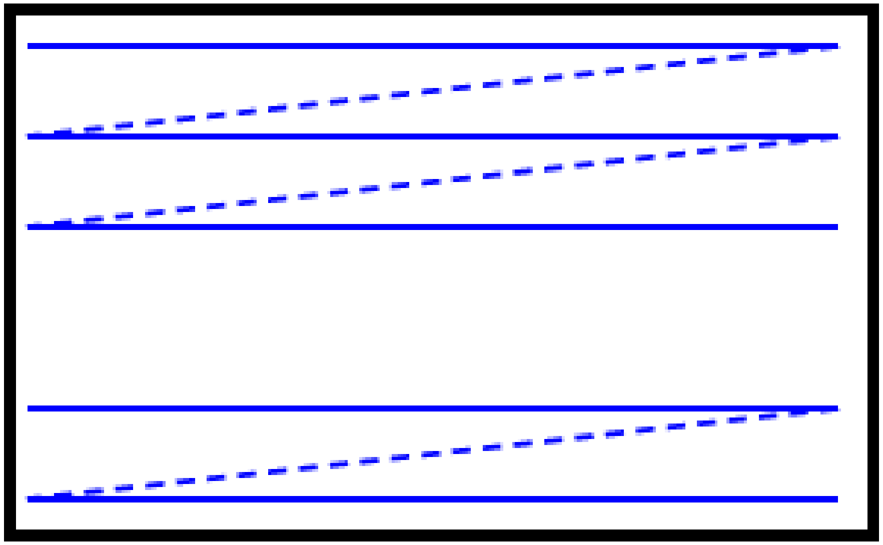
•Display element could be LC, EL, OLED, FED etc

Yeh & Gu

Sequential Addressing (Raster Scan)

- Time is multiplexed
  - Signal exists in a time cell
- A pixel is displayed at a time
  - Single data line
- Rigid time sequence and relative spatial location of signal
  - Raster scan
- Data rate scales with number of pixels
- Duty cycle scales with number of pixels
- Horizontal sync coordinates lines
- Vertical sync coordinates frames
- Blanking signals (vertical & horizontal) so that retraces are invisible





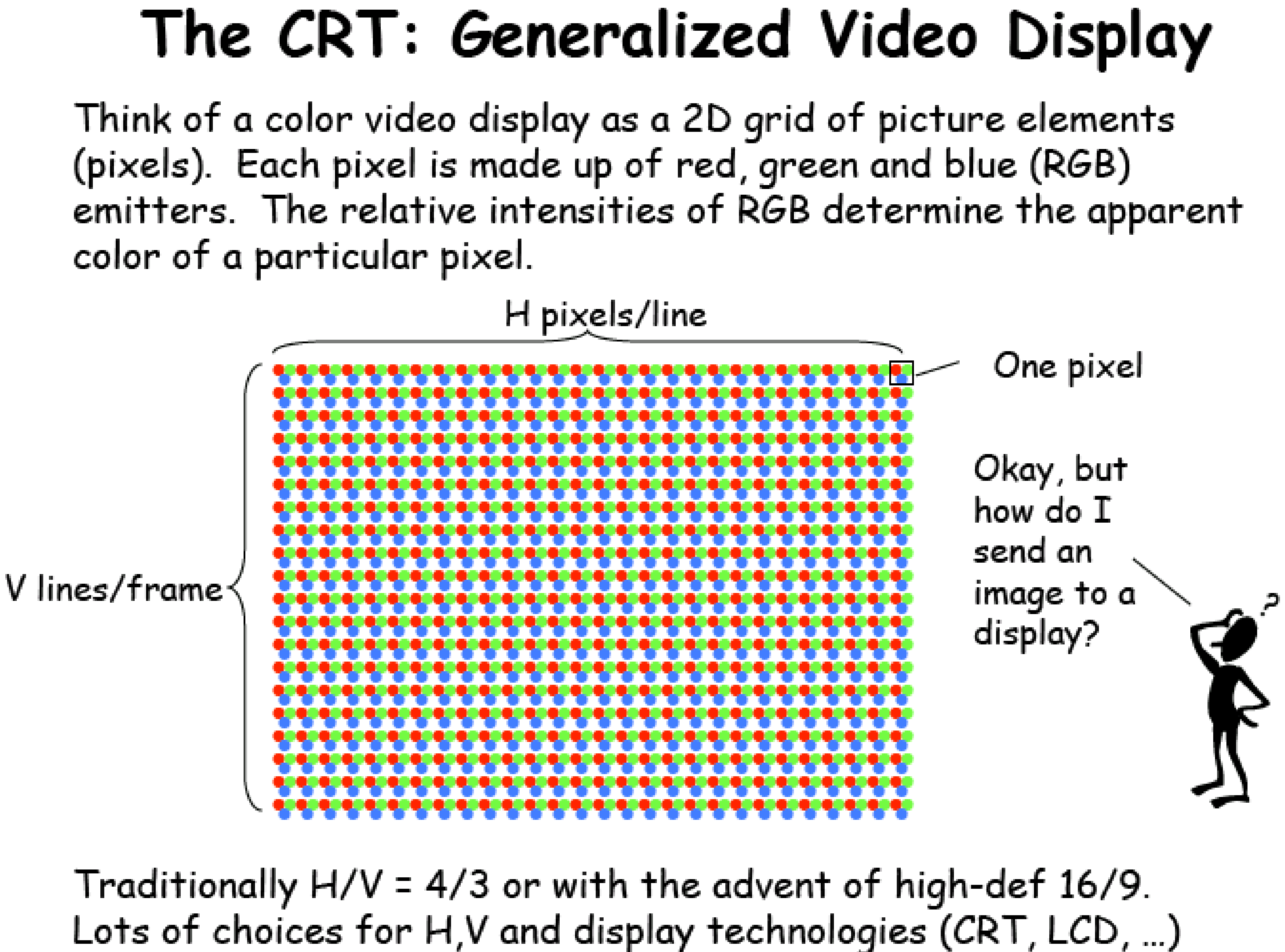
—— Scan Lines

- - - - Retrace Lines

Tannas, SID 00 Applications Seminar

Video(analog signals)

Means DAC



The CRT: Generalized Video Display

Think of a color video display as a 2D grid of picture elements (pixels). Each pixel is made up of red, green and blue (RGB) emitters. The relative intensities of RGB determine the apparent color of a particular pixel.

Traditionally H/V = 4/3 or with the advent of high-def 16/9.
Lots of choices for H,V and display technologies (CRT, LCD, ...)

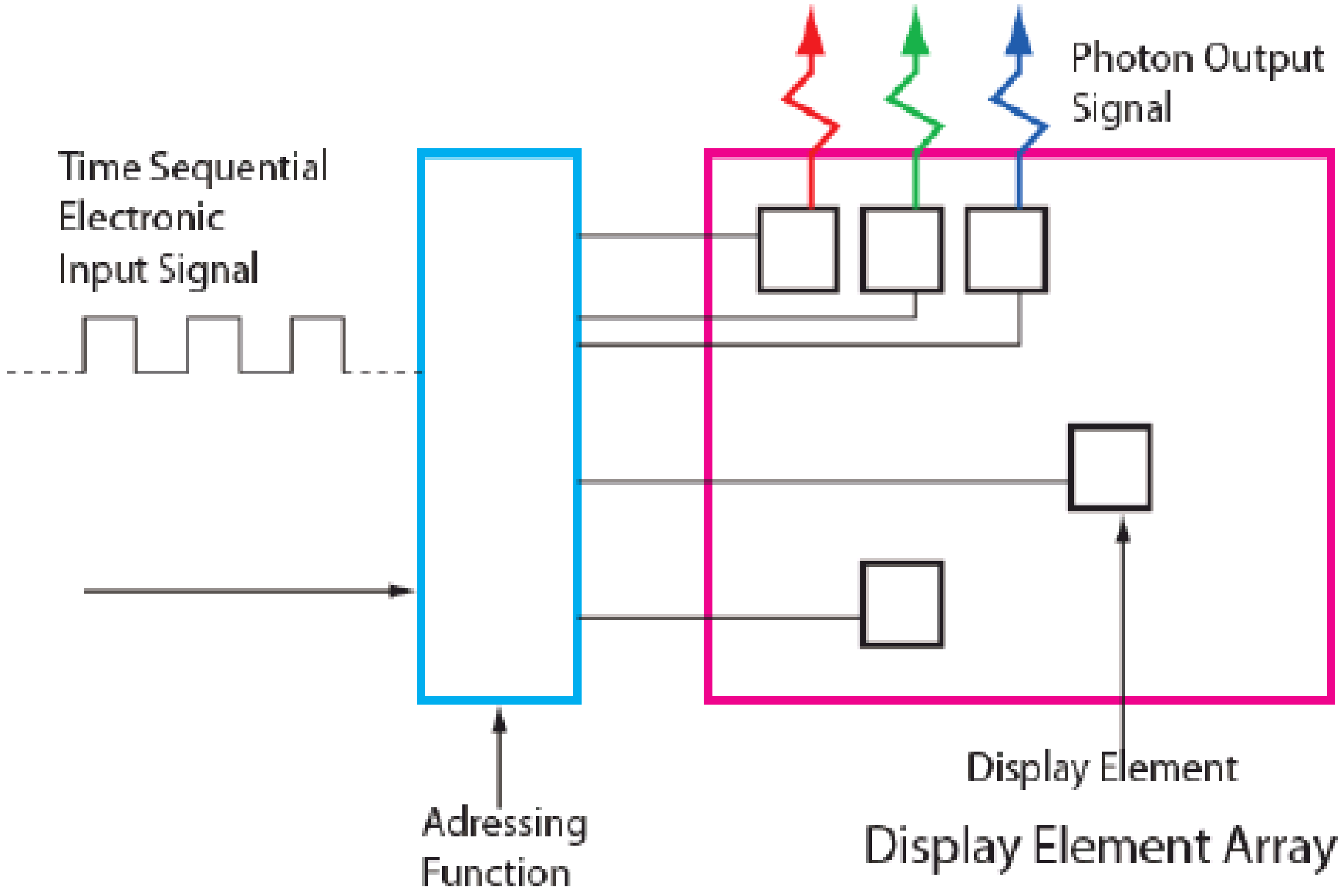
Information Capacity of Displays

(Pixel Count)

Resolution	**Pixel**	**Ratio**
Video Graphic Array (**VGA**)	640 x 480 x RGB	4:3
Super Vedio Graphic Array (**SVGA**)	800 x 600 x RGB	4:3
eXtended Graphic Array (**XGA**)	1,024 x 768 x RGB	4:3
Super eXtended Graphic Array (**SXGA**)	1,280 x 1,024 RGB	5:4
Super eXtended Graphic Array plus (**SXGA+**)	1,400 x 1,080 x RGB	4:3
Ultra eXtended Graphic Array (**UXGA**)	1,600 x 1,200 x RGB	4:3
Quad eXtended Graphics Array (**QXGA**)	2048 x 1536 x RGB	4:3
Quad Super eXtended Graphics Array (**QSXGA**)	2560 x 2048 x RGB	4:3

Display Devices, No. 21, Spring 2000, p. 41

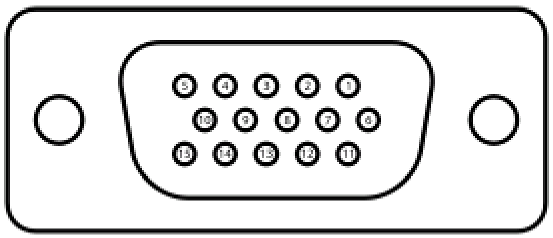
How Do Displays Work?



Pankove

- “**Time Sequential Electrical Signals**” converted into **images**.
  - Signals routed to the display elements (**similar to memory addressing**)
  - Pixels convert the electrical signal into light of color and intensity (**inverse of image capture**)

Pin out



The image and table detail the 15-pin VESA DDC2/E-DDC connector; the diagram's pin numbering is that of a female connector functioning as the graphics adapter output. In the male connector, this pin numbering corresponds with the mirror image of the cable's wire-and-solder side.

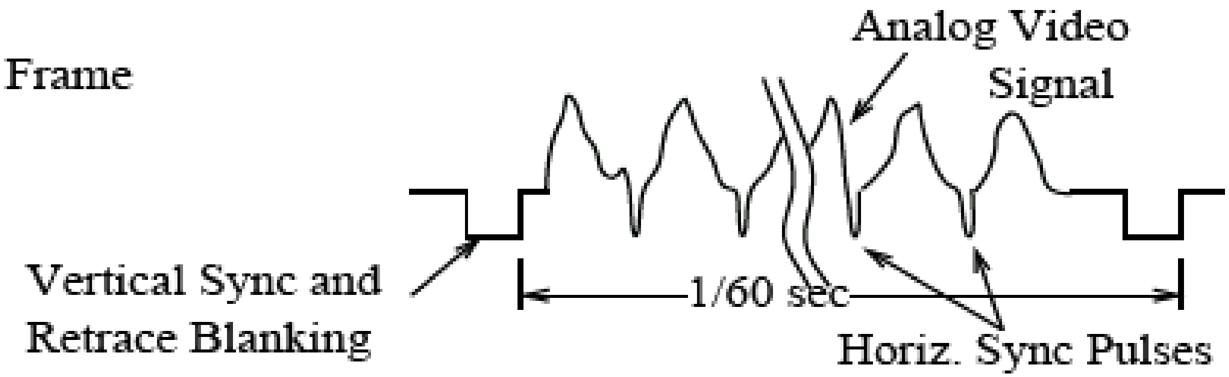
A female DE15 socket (videocard side).

Pin 1	RED	Red video
Pin 2	GREEN	Green video
Pin 3	BLUE	Blue video
Pin 4	ID2/RES	formerly Monitor ID bit 2, reserved since E-DDC
Pin 5	GND	Ground (HSync)
Pin 6	RED_RTN	Red return
Pin 7	GREEN_RTN	Green return
Pin 8	BLUE_RTN	Blue return
Pin 9	KEY/PWR	formerly key, now +5V DC
Pin 10	GND	Ground (VSync, DDC)
Pin 11	ID0/RES	formerly Monitor ID bit 0, reserved since E-DDC
Pin 12	ID1/SDA	formerly Monitor ID bit 1, I²C data since DDC2
Pin 13	HSync	Horizontal sync
Pin 14	VSync	Vertical sync
Pin 15	ID3/SCL	formerly Monitor ID bit 3, I²C clock since DDC2

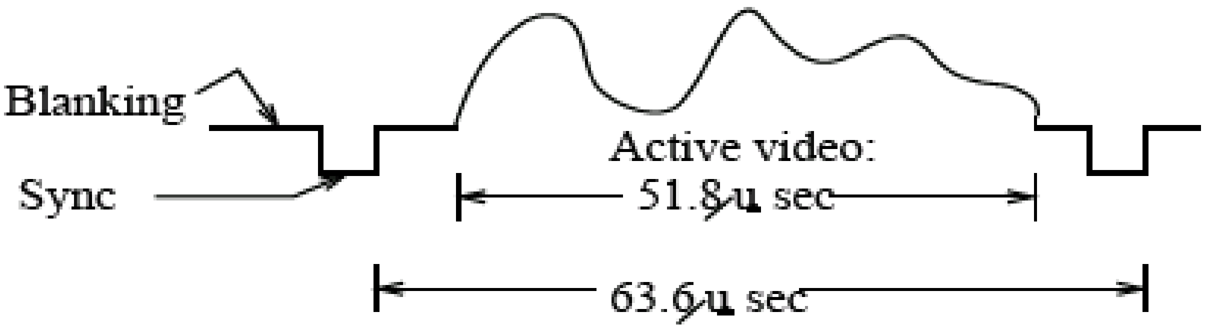
Composite Frames

- The ‘frame’ is a single picture (snapshot).
  - It is made up of many lines.
  - Each frame has a synchronizing pulse (vertical sync).
  - Each line has a synchronizing pulse (horizontal sync).
  - Brightness is represented by a positive voltage.
  - Horizontal and Vertical intervals both have blanking so that retraces are not seen (invisible).

Composite Frame



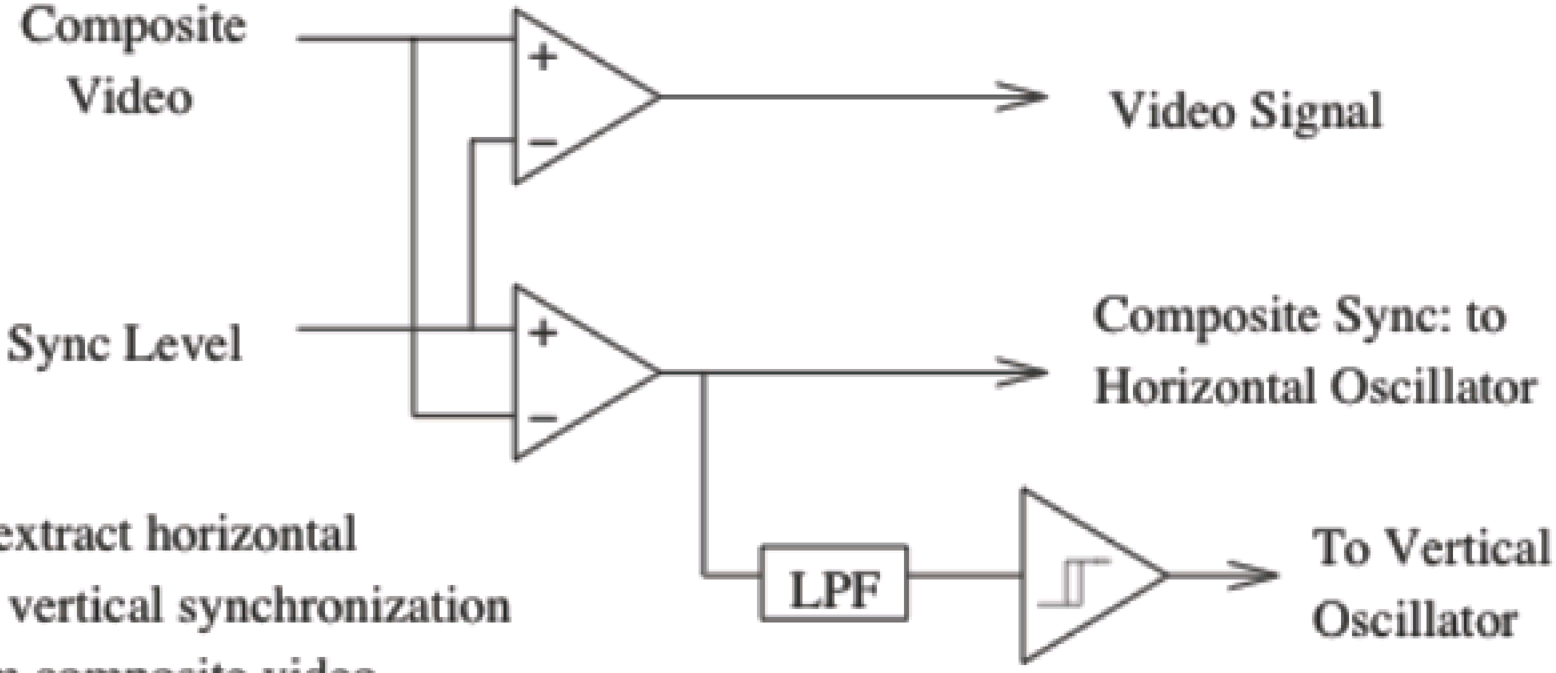
Horizontal Line



Slide by Professor Don Troxel

Video Capture: Signal Recovery

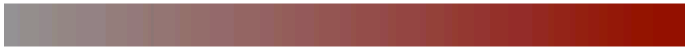
- Composite video has picture data and both syncs.
  - Picture data (video) is above the sync level.
  - Simple comparators extract video and composite sync.
- Composite sync is fed directly to the horizontal oscillator.
- A low-pass filter is used to separate the vertical sync.
  - The edges of the low-passed vertical sync are squared up by a Schmidt trigger.



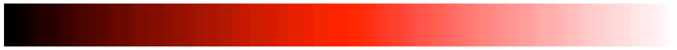
To extract horizontal and vertical synchronization from composite video

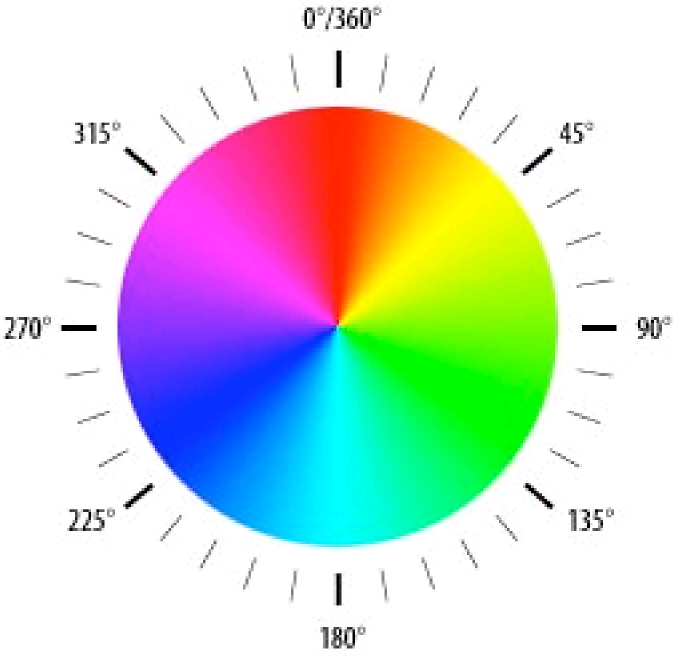
Video Feature Extraction

- A common technique for finding features in a real-time video stream is to locate the center-of-mass for pixels of a given color
  - Using RGB can be a pain since a color (eg, red) will be represented by a wide range of RGB values depending on the type and intensity of light used to illuminate the scene. Tedious and finicky calibration process required.
- Consider using a HSL/HSV color space
  - H = hue (see diagram)
  - S = saturation, the degree by which color differs from neutral gray (0% to 100%)



  - L = lightness, illumination of the color (0% to 100%)

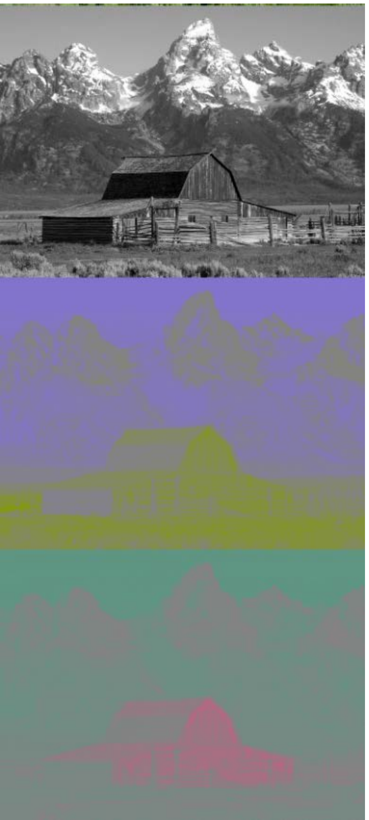


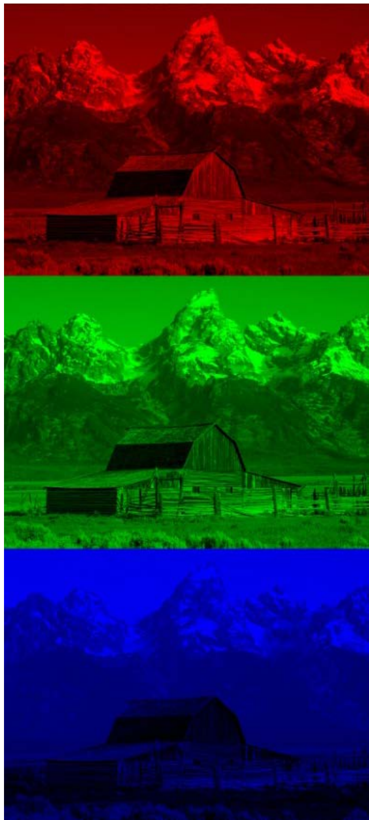


- Filter pixels by hue!

YCrCb to RGB (for display)

- 8-bit data
  - R = 1.164(Y - 16) + 1.596(Cr - 128)
  - G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.392(Cb - 128)
  - B = 1.164(Y - 16) + 2.017(Cb - 128)
- 10-bit data
  - R = 1.164(Y - 64) + 1.596(Cr - 512)
  - G = 1.164(Y - 64) - 0.813(Cr - 512) - 0.392(Cb - 512)
  - B = 1.164(Y - 64) + 2.017(Cb - 512)
- Implement using
  - Integer arithmetic operators (scale constants/answer by 2^{11})
  - 5 BRAMs (1024x16) as lookup tables for multiplications





luminance (luminosity)
two chrominance (color) components

How does DE2 FPGA controls VGA?

VGA

DE2 User manual: page 37: 4.7 Using VGA

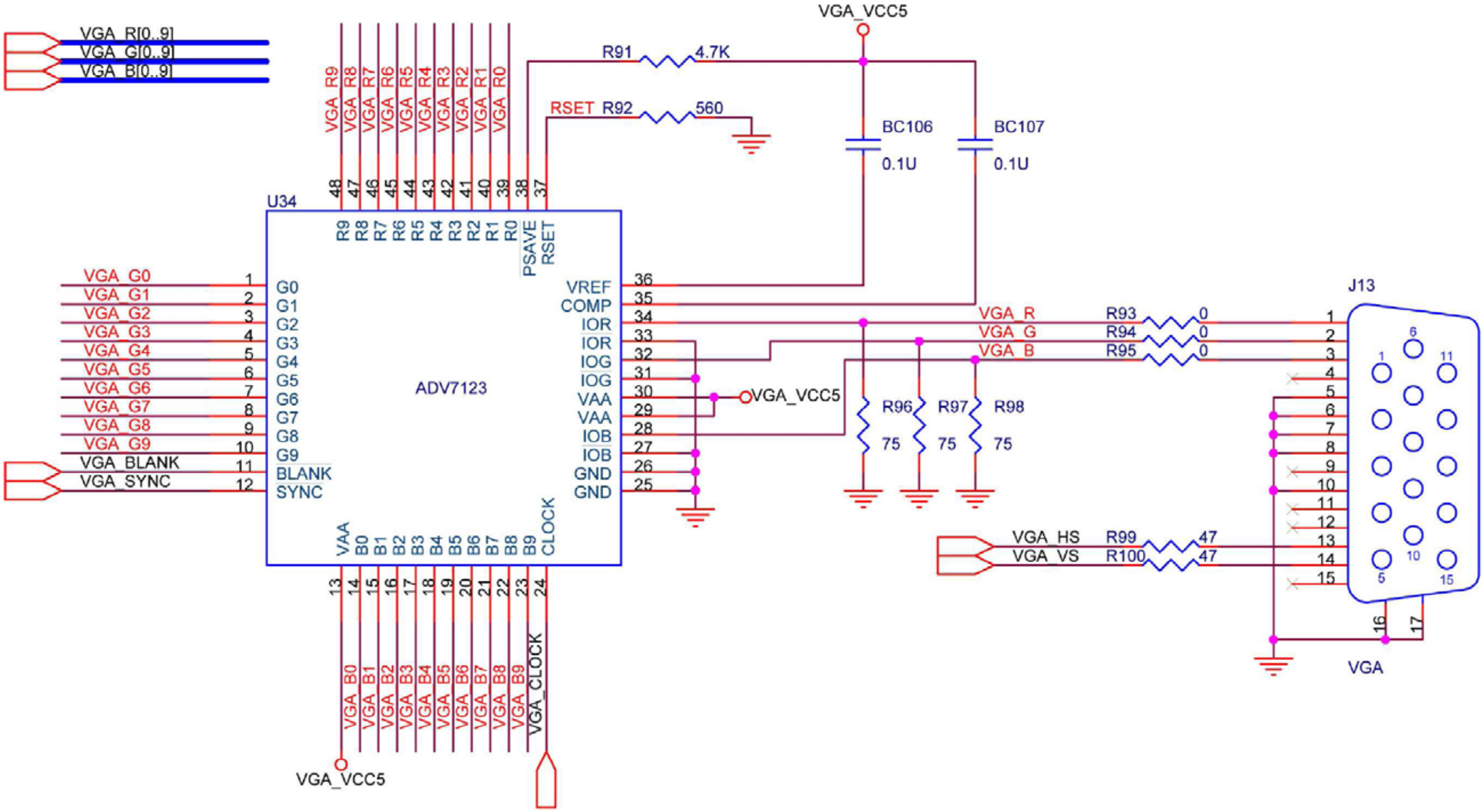
VGA output

· Uses the ADV7123 240-MHz triple 10-bit high-speed video DAC

· With 15-pin high-density D-sub connector

· Supports up to 1600 x 1200 at 100-Hz refresh rate

· Can be used with the Cyclone II FPGA to implement a high-performance TV Encoder

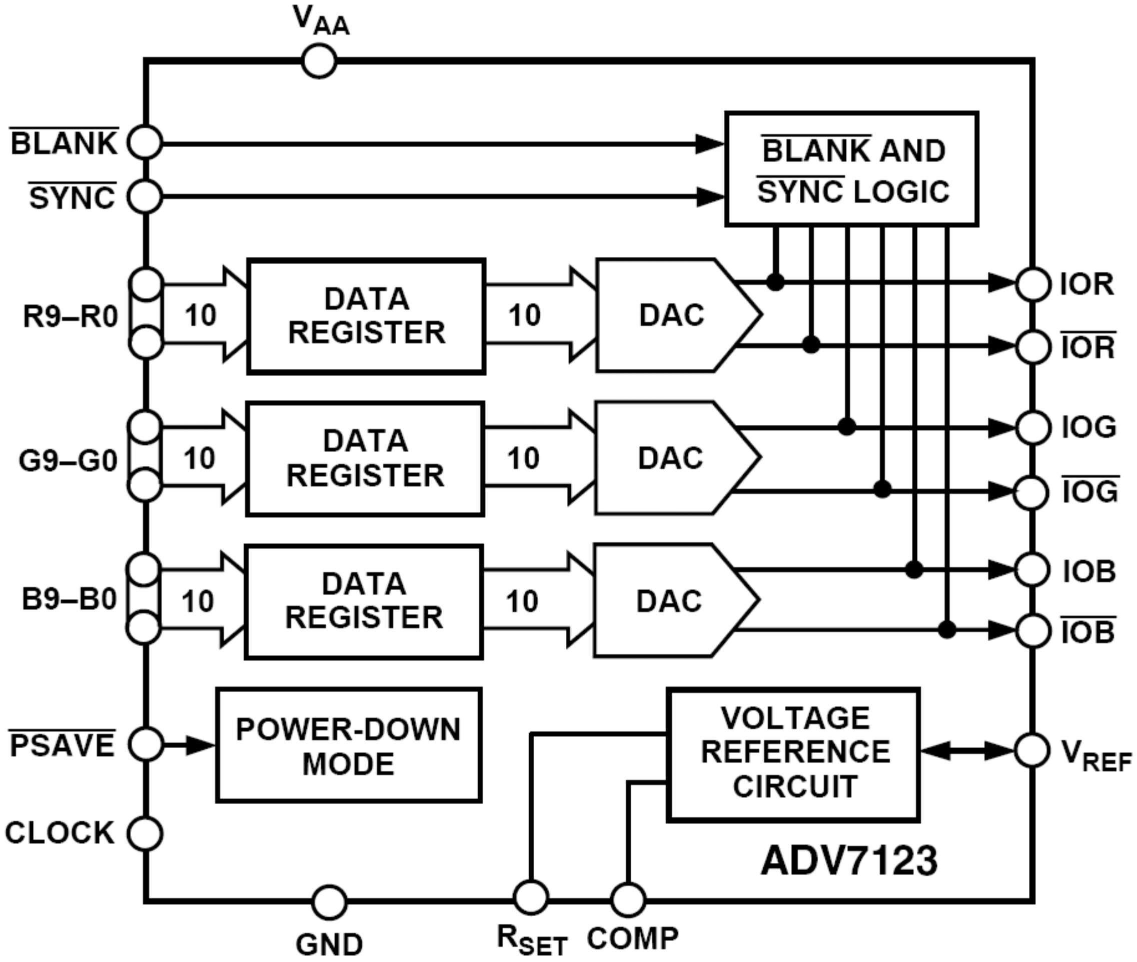


ANALOG DEVICES

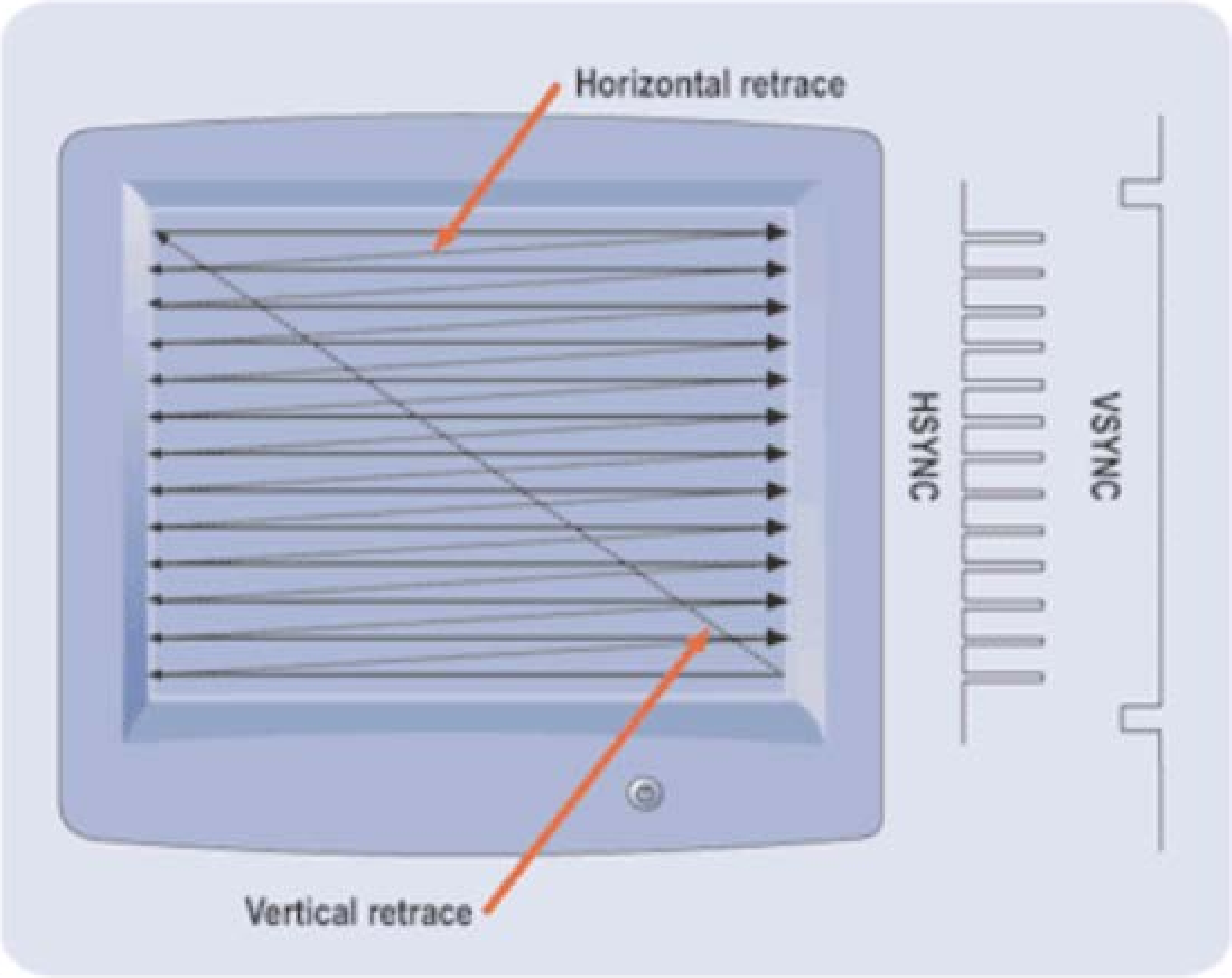
CMOS, 240 MHz Triple 10-Bit High Speed Video DAC

ADV7123

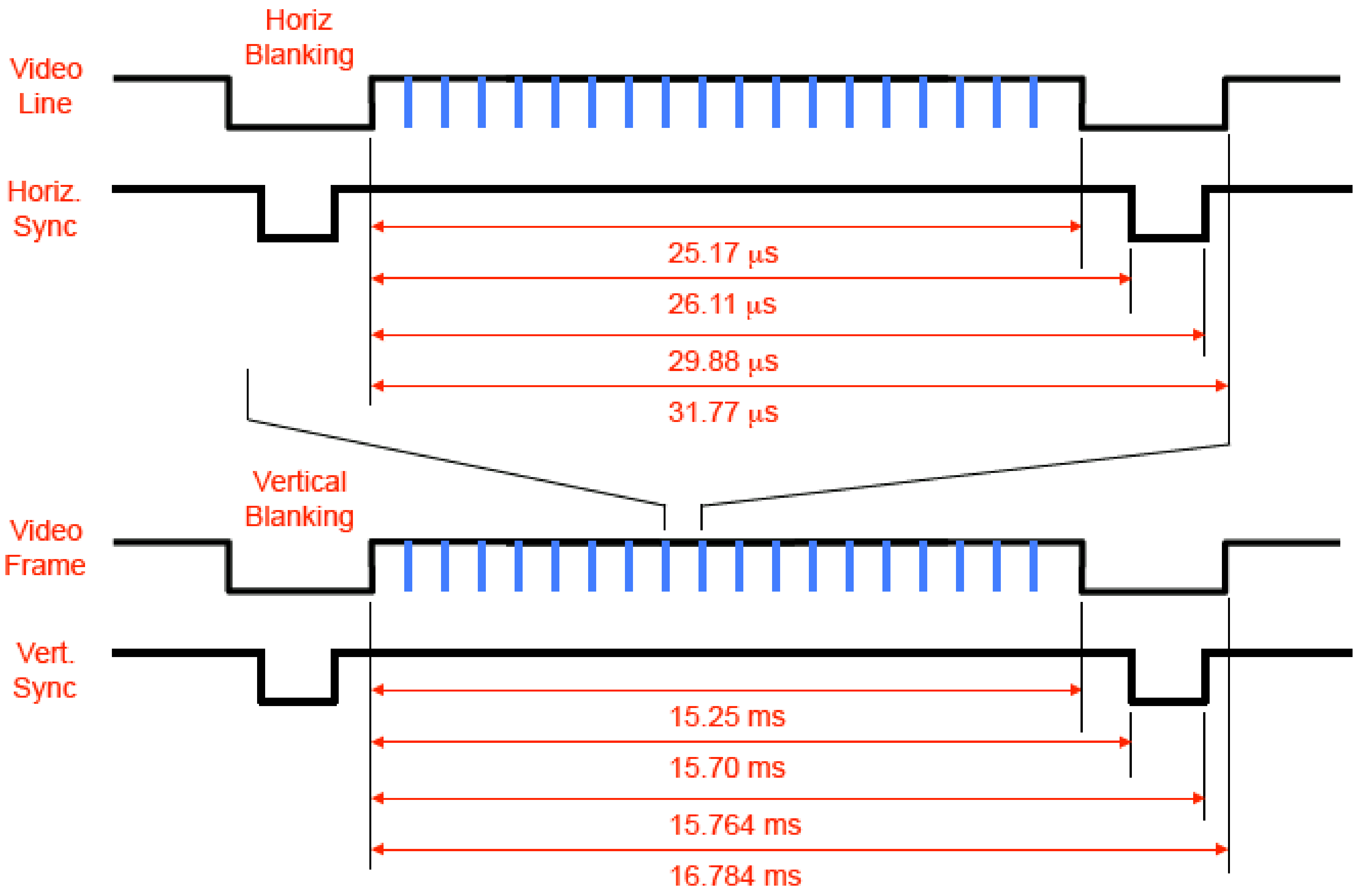
FUNCTIONAL BLOCK DIAGRAM



Sync Signals (HS and VS)



VGA (640x480) Video



What kind of clock do you need?

Information Capacity of Displays

(Pixel Count)

Resolution	**Pixel**	**Ratio**
Video Graphic Array (**VGA**)	640 x 480 x RGB	4:3
Super Vedio Graphic Array (**SVGA**)	800 x 600 x RGB	4:3
eXtended Graphic Array (**XGA**)	1,024 x 768 x RGB	4:3
Super eXtended Graphic Array (**SXGA**)	1,280 x 1,024 RGB	5:4
Super eXtended Graphic Array plus (**SXGA+**)	1,400 x 1,080 x RGB	4:3
Ultra eXtended Graphic Array (**UXGA**)	1,600 x 1,200 x RGB	4:3
Quad eXtended Graphics Array (**QXGA**)	2048 x 1536 x RGB	4:3
Quad Super eXtended Graphics Array (**QSXGA**)	2560 x 2048 x RGB	4:3

Display Devices, No. 21, Spring 2000, p. 41

e.g. 1024x768 @ 60Hz

In each period of refresh rate we need to scan ALL pixels

How many pixels?

Scanline part	Pixels
Visible area	1024
Front porch	24
Sync pulse	136
Back porch	160
Whole line	**1344**

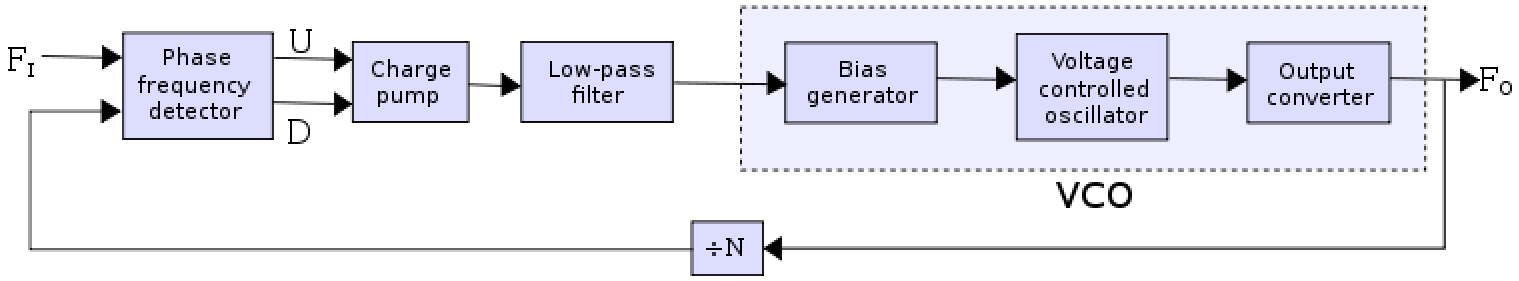
Frame part	Lines
Visible area	768
Front porch	3
Sync pulse	6
Back porch	29
Whole frame	**806**

Total Number of pixels: 1344 x 806 = 1083264

To Have a full scan in 60 MHZ we need a clock w/
f = 60 x 1083264 = 64995840 = **65 MHz**

http://tinyvga.com/vga-timing

The Phase Locked Loop (PLL)

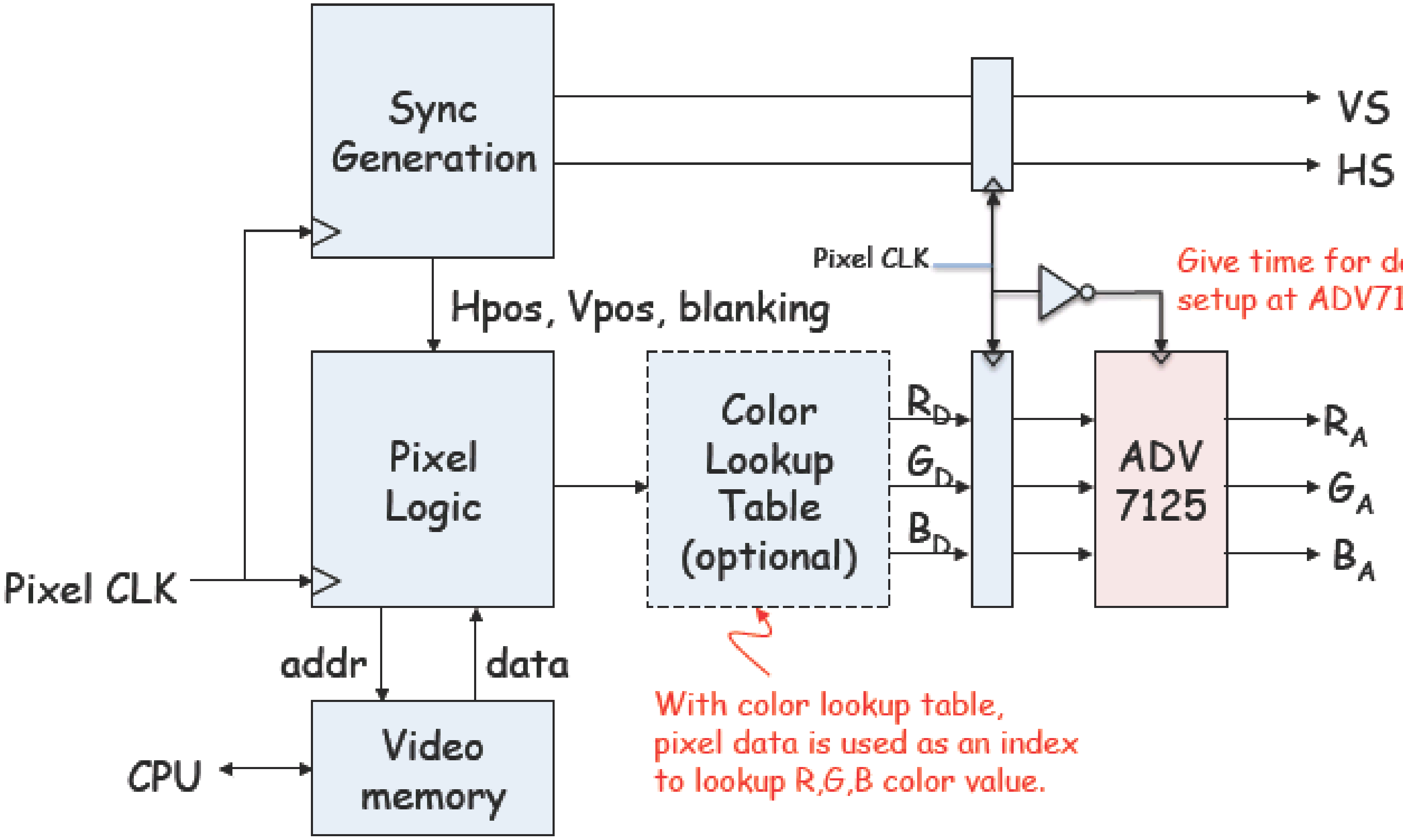


A feature available in Cyclone II devices that employs a phase-locked loop (PLL). The Cyclone II PLL provides advanced multiplication, programmable duty cycle, phase shifting, programmable bandwidth, manual clock switchover, clock outputs driving all networks, and a source synchronous mode.
You can take advantage of the Cyclone II PLL with the altpll megafunction.

Phase-Locked Loops (ALTPLL)

Megafunction User Guide

Generating VGA-style Video

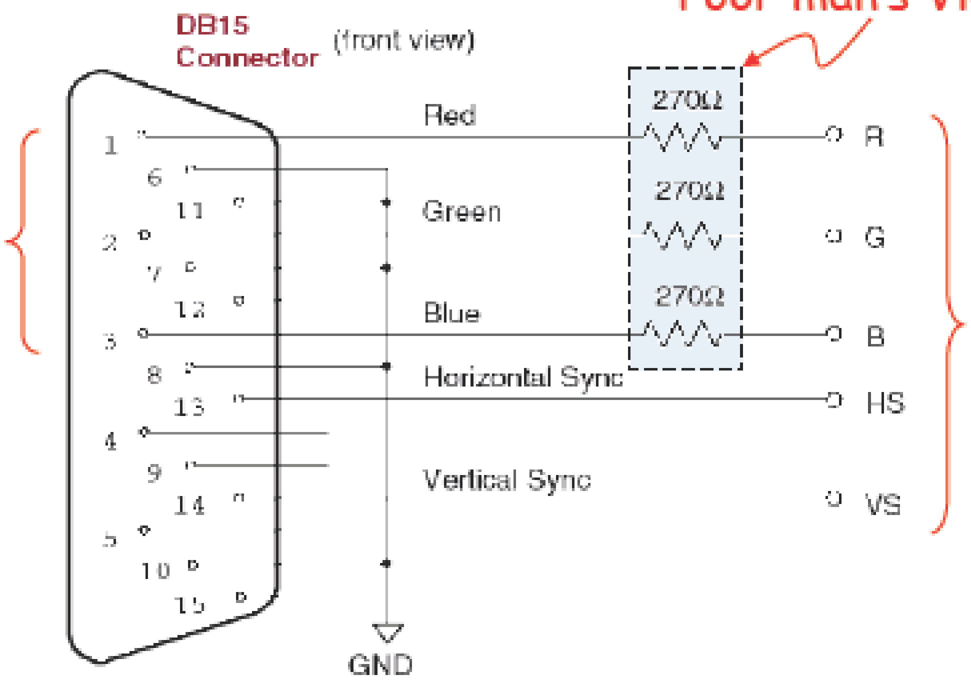


Give time for data to setup at ADV7125

With color lookup table, pixel data is used as an index to lookup R,G,B color value.

Without color lookup table, pixel data is used directly as R,G,B value (aka "true color")

Simple VGA Interface for FPGA



Poor man's Video DAC

Your circuitry should produce TTL-level signals (3.3V high level)

HS, VS are active-low signals.

R, G, B are active-high.

Shown: a simple "8-color" scheme

The R, G and B signals are terminated with 75 Ohms to ground inside of the VGA monitor. So when you drive your 3.3V signal through the 270 Ohm series resistor, it shows up at the monitor as 0.7V – exactly what the VGA spec calls for.

$$0.7V = (\frac{75}{75+270})(3.3V)$$

Verilog: XVGA Display (1024x768)

```
module xvga(clk,hcount,vcount,hsync,vsync);
  input clk;                // 64.8 Mhz
  output [10:0] hcount;
  output [9:0] vcount;
  output hsync, vsync;
  output [2:0] rgb;

  reg hsync,vsync,hblank,vblank,blank;
  reg [10:0] hcount;        // pixel number on current line
  reg [9:0] vcount;         // line number

  wire hsyncon,hsyncoff,hreset,hblankon; // next slide for generation
  wire vsyncon,vsyncoff,vreset,vblankon; // of timing signals

  wire next_hb = hreset ? 0 : hblankon ? 1 : hblank; // sync & blank
  wire next_vb = vreset ? 0 : vblankon ? 1 : vblank;

  always @(posedge clk) begin
    hcount <= hreset ? 0 : hcount + 1;
    hblank <= next_hb;
    hsync <= hsyncon ? 0 : hsyncoff ? 1 : hsync;   // active low

    vcount <= hreset ? (vreset ? 0 : vcount + 1) : vcount;
    vblank <= next_vb;
    vsync <= vsyncon ? 0 : vsyncoff ? 1 : vsync;   // active low
  end
```

XVGA (1024x768) Sync Timing

```
// assume 65 Mhz pixel clock

// horizontal: 1344 pixels total
// display 1024 pixels per line
assign hblankon = (hcount == 1023);  // turn on blanking
assign hsyncon = (hcount == 1047);   // turn on sync pulse
assign hsyncoff = (hcount == 1183);  // turn off sync pulse
assign hreset = (hcount == 1343);    // end of line (reset counter)

// vertical: 806 lines total
// display 768 lines
assign vblankon = hreset & (vcount == 767);   // turn on blanking
assign vsyncon = hreset & (vcount == 776);    // turn on sync pulse
assign vsyncoff = hreset & (vcount == 782);   // turn off sync pulse
assign vreset = hreset & (vcount == 805);     // end of frame
```

Video Test Patterns

- Big white rectangle (good for “auto adjust” on monitor)

```
always @(posedge clk) begin
   if (vblank | (hblank & ~hreset)) rgb <= 0;
   else
    rgb <= 7;
end
```

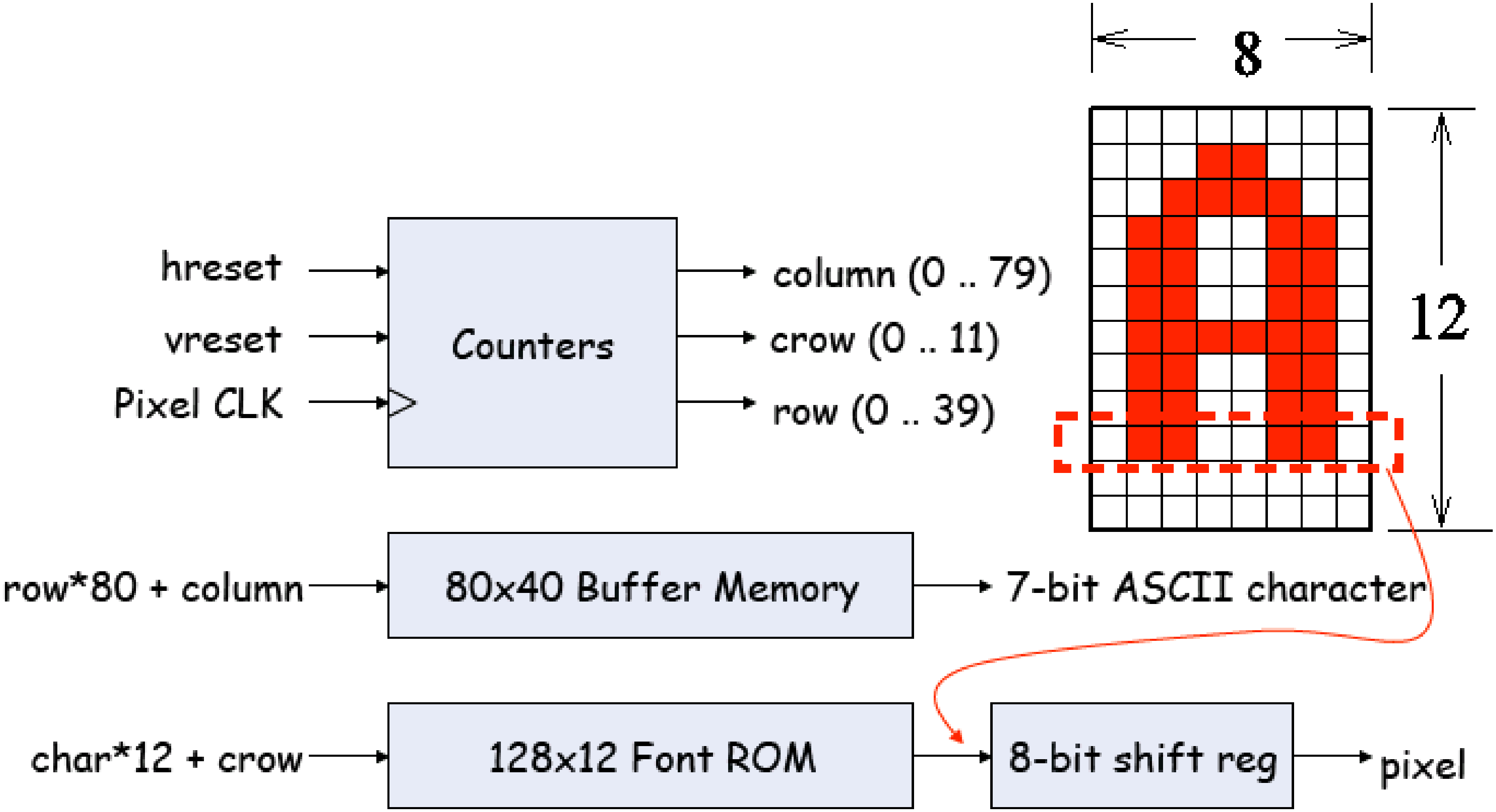
- Color bars

```
always @(posedge clk) begin
   if (vblank | (hblank & ~hreset)) rgb <= 0;
   else
    rgb <= hcount[8:6];
end
```

RGB	*Color*
000	black
001	blue
010	green
011	cyan
100	red
101	magenta
110	yellow
111	white

Character Display

(80 columns x 40 rows, 8x12 glyph)



hreset

vreset

Pixel CLK

Counters

column (0 .. 79)

crow (0 .. 11)

row (0 .. 39)

8

12

row*80 + column

80x40 Buffer Memory

7-bit ASCII character

char*12 + crow

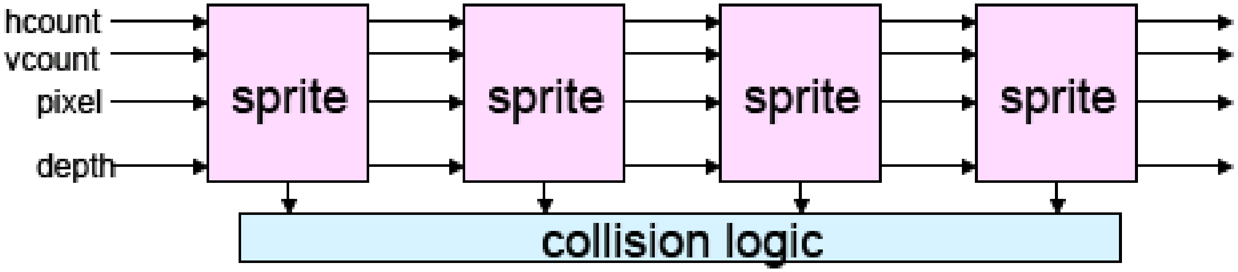
128x12 Font ROM

8-bit shift reg

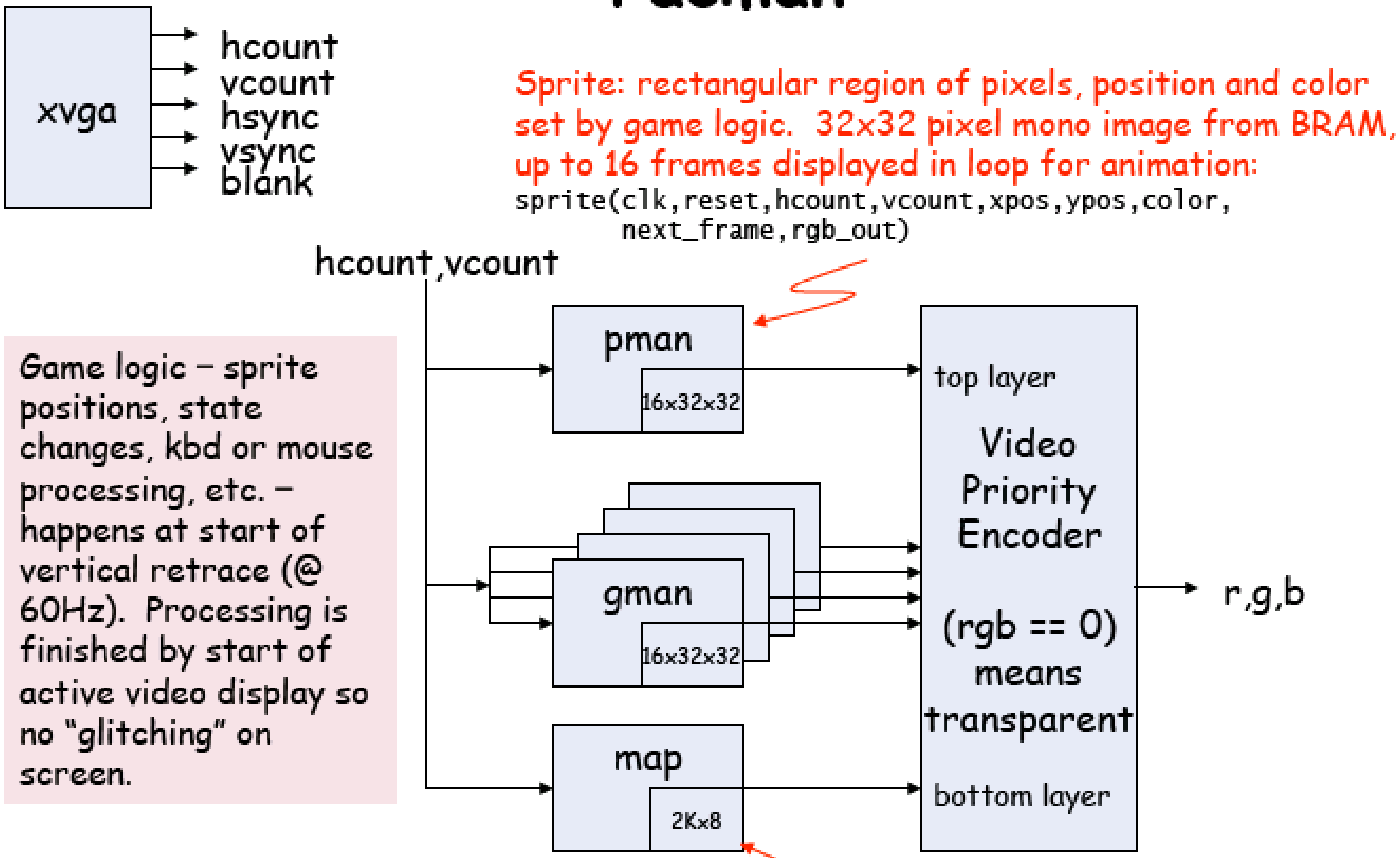
pixel

Game Graphics using Sprites

- Sprite = game object occupying a rectangular region of the screen (it's bounding box).
  - Usually it contains both opaque and transparent pixels.
  - Given (H,V), sprite returns pixel (0=transparent) and depth
  - Pseudo 3D: look at current pixel from all sprites, display the opaque one that's in front (min depth): see sprite pipeline below
  - Collision detection: look for opaque pixels from other sprites
  - Motion: smoothly change coords of upper left-hand corner
- Pixels can be generated by logic or fetched from a bitmap (memory holding array of pixels).
  - Bitmap may have multiple images that can be displayed in rapid succession to achieve animation.
  - Mirroring and 90° rotation by fooling with bitmap address, crude scaling by pixel replication, or resizing filter.



Pacman



Sprite: rectangular region of pixels, position and color set by game logic. 32x32 pixel mono image from BRAM, up to 16 frames displayed in loop for animation:

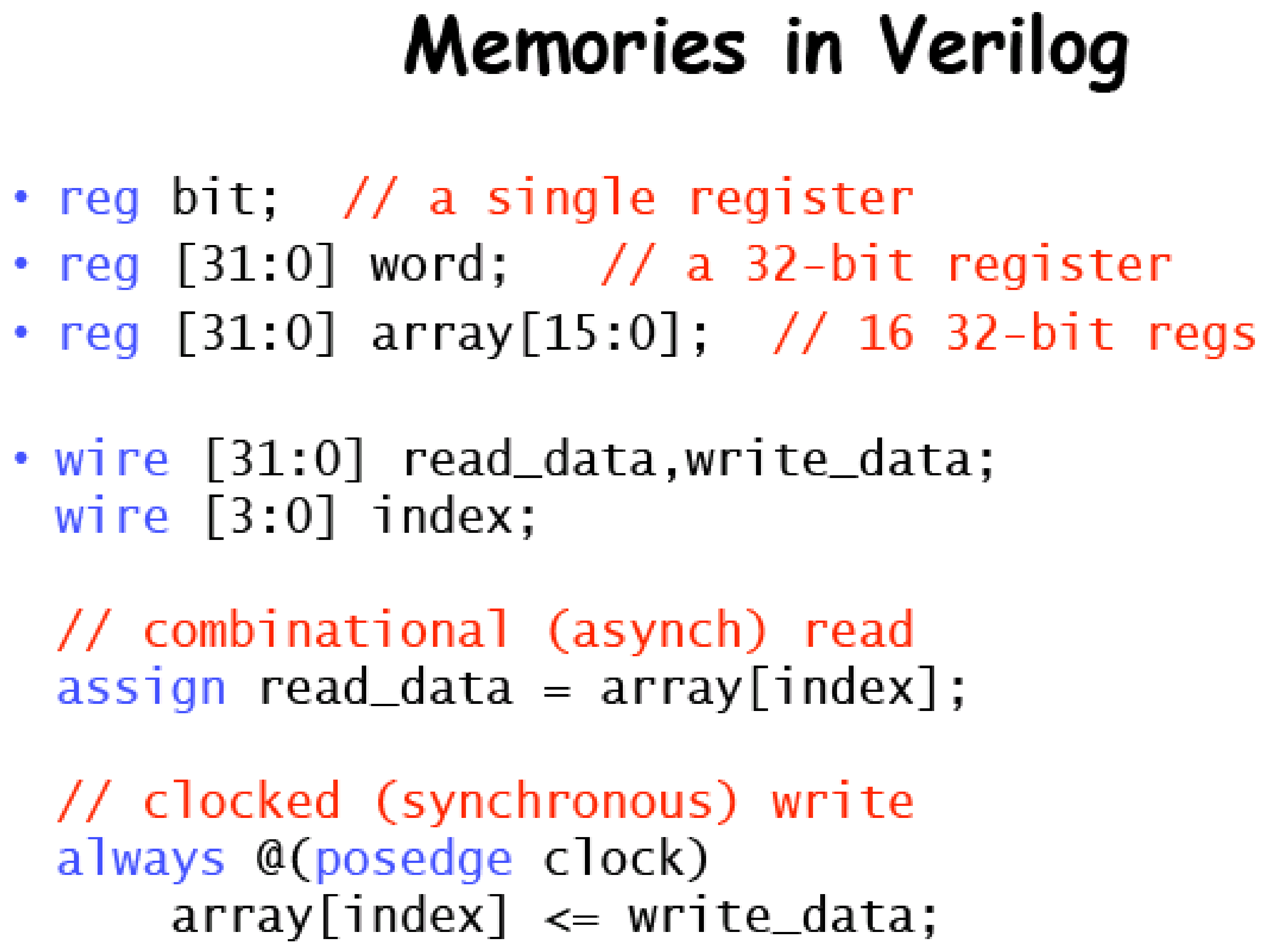
```
sprite(clk,reset,hcount,vcount,xpos,ypos,color,
       next_frame,rgb_out)
```

Game logic – sprite positions, state changes, kbd or mouse processing, etc. – happens at start of vertical retrace (@ 60Hz). Processing is finished by start of active video display so no "glitching" on screen.

4 board maps, each 512x8
each map is 16x24 tiles (376 tiles)
Each tile has 8 bits: 4 for move direction (==0 for a wall), pills

Memories

Memories



Memories in Verilog

```
• reg bit;  // a single register
• reg [31:0] word;   // a 32-bit register
• reg [31:0] array[15:0];  // 16 32-bit regs

• wire [31:0] read_data,write_data;
  wire [3:0] index;

  // combinational (asynch) read
  assign read_data = array[index];

  // clocked (synchronous) write
  always @(posedge clock)
      array[index] <= write_data;
```

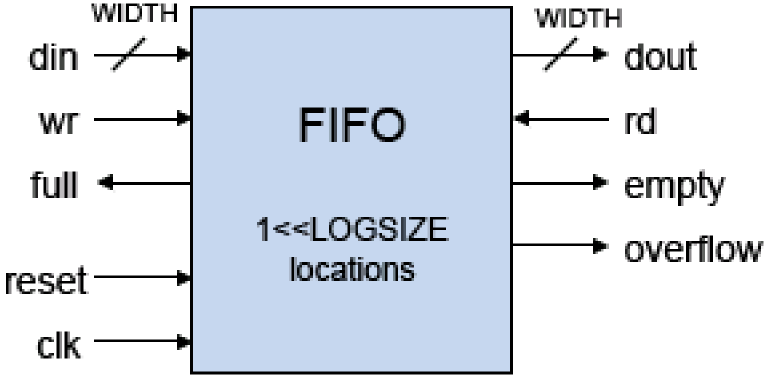
Multi-port Memories (aka regfiles)

```
reg [31:0] regfile[30:0]; // 31 32-bit words

// Beta register file: 2 read ports, 1 write
wire [4:0] ra1,ra2,wa;
wire [31:0] rd1,rd2,wd;

assign ra1 = inst[20:16];
assign ra2 = ra2sel ? inst[25:21] : inst[15:11];
assign wa = wasel ? 5'd30 : inst[25:21];

// read ports
assign rd1 = (ra1 == 5’d31) ? 32’d0 : regfile[ra1];
assign rd2 = (ra2 == 5’d31) ? 32’d0 : regfile[ra2];
// write port
always @(posedge clk)
  if (werf) regfile[wa] <= wd;

assign z = ~| rd1;   // used in BEQ/BNE instructions
```

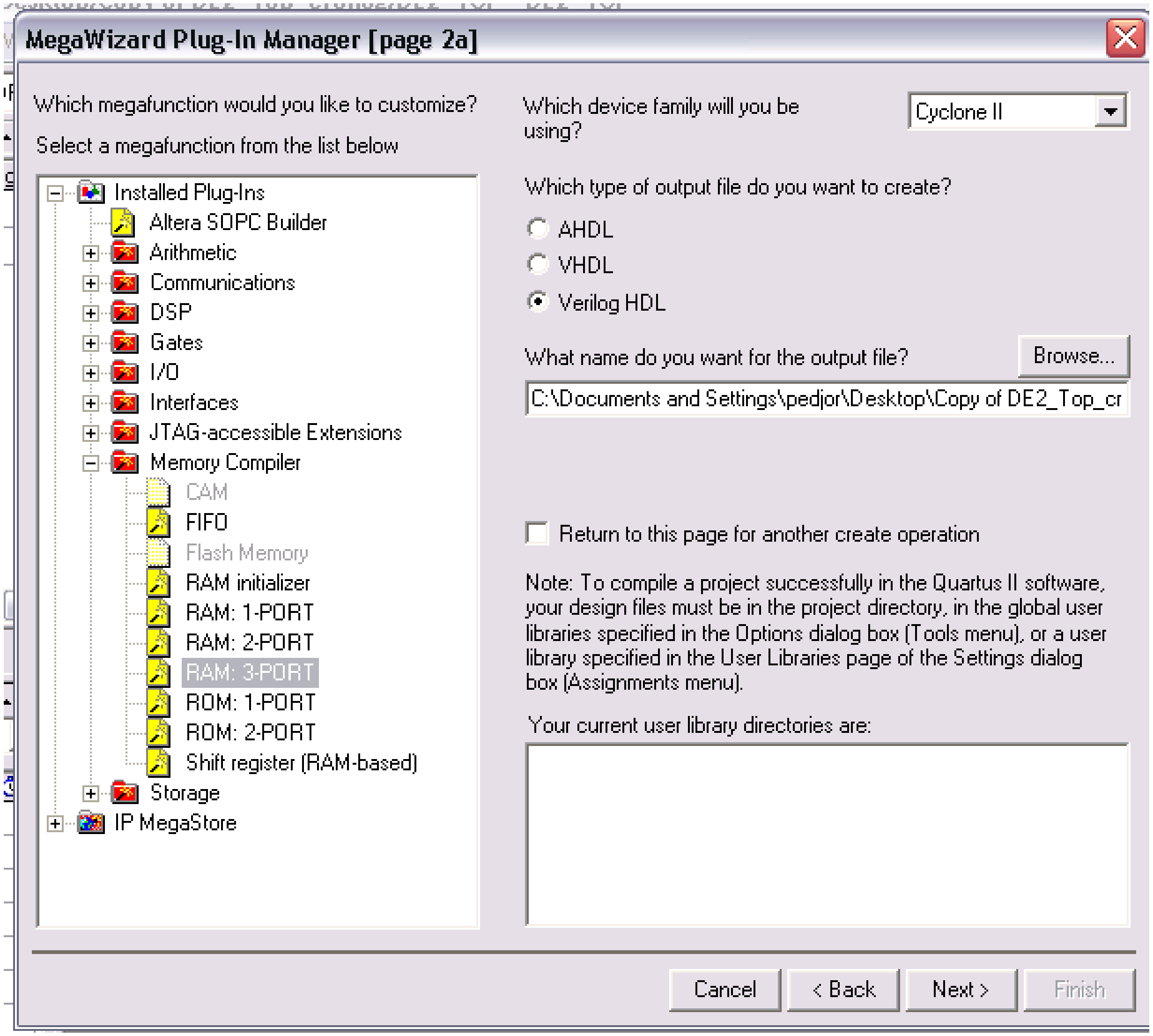
Address

Data

FIFOs



```
// a simple synchronous FIFO (first-in first-out) buffer
// Parameters:
//    LOGSIZE  (parameter) FIFO has 1<<LOGSIZE elements
//    WIDTH    (parameter) each element has WIDTH bits
// Ports:
//    clk      (input) all actions triggered on rising edge
//    reset    (input) synchronously empties fifo
//    din      (input, WIDTH bits) data to be stored
//    wr       (input) when asserted, store new data
//    full     (output) asserted when FIFO is full
//    dout     (output, WIDTH bits) data read from FIFO
//    rd       (input) when asserted, removes first element
//    empty    (output) asserted when fifo is empty
//    overflow (output) asserted when WR but no room, cleared on next RD
module fifo #(parameter LOGSIZE = 2,    // default size is 4 elements
                        WIDTH = 4)      // default width is 4 bits
            (input clk,reset,wr,rd, input [WIDTH-1:0] din,
             output full,empty,overflow, output [WIDTH-1:0] dout);
…
```



MegaWizard Plug-In Manager [page 2a]

Which megafunction would you like to customize?

Select a megafunction from the list below

Which device family will you be using? Cyclone II

Which type of output file do you want to create?

○ AHDL

○ VHDL

● Verilog HDL

What name do you want for the output file?

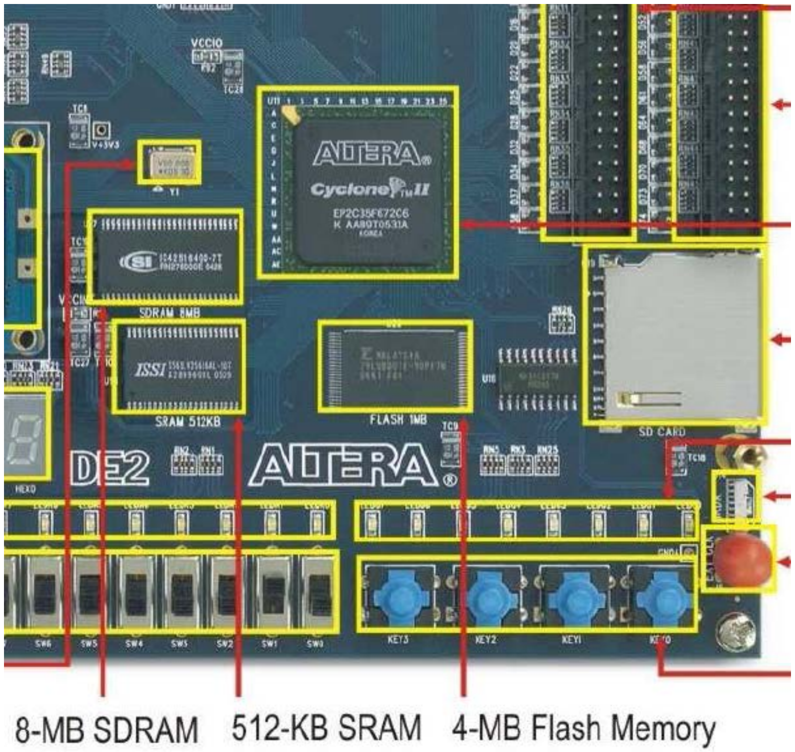
C:\Documents and Settings\pedjor\Desktop\Copy of DE2_Top_cr

□ Return to this page for another create operation

Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:

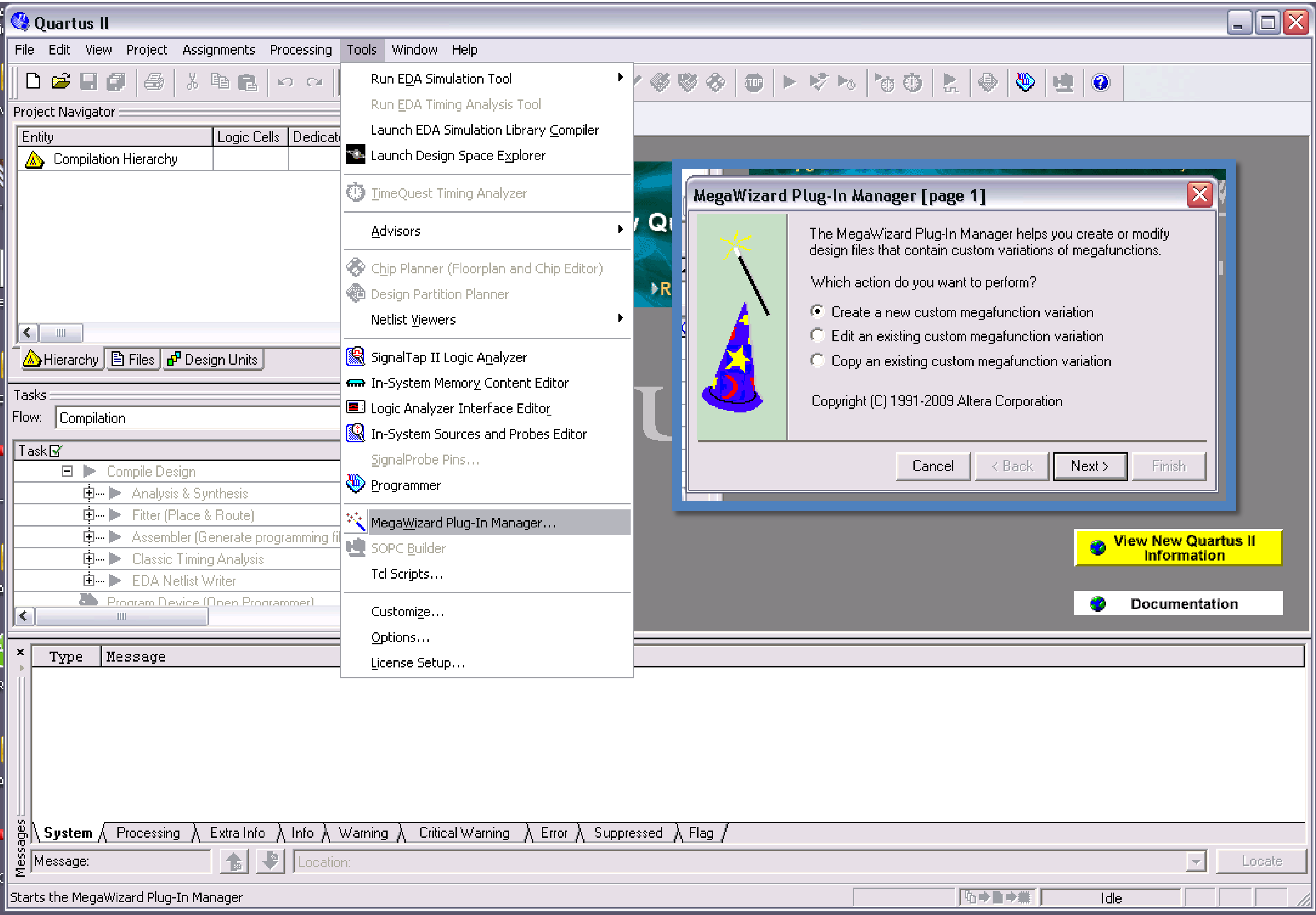
Memories external to the FPGA



When interfacing external memories you should look at the datasheet!
Need to understand the exact protocol, adressing, timing, etc.
Some tools may build you the interface to external memories

TOOLS

Tools – Mega Wizard



MegaWizard Plug-In Manager [page 1]

The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions.

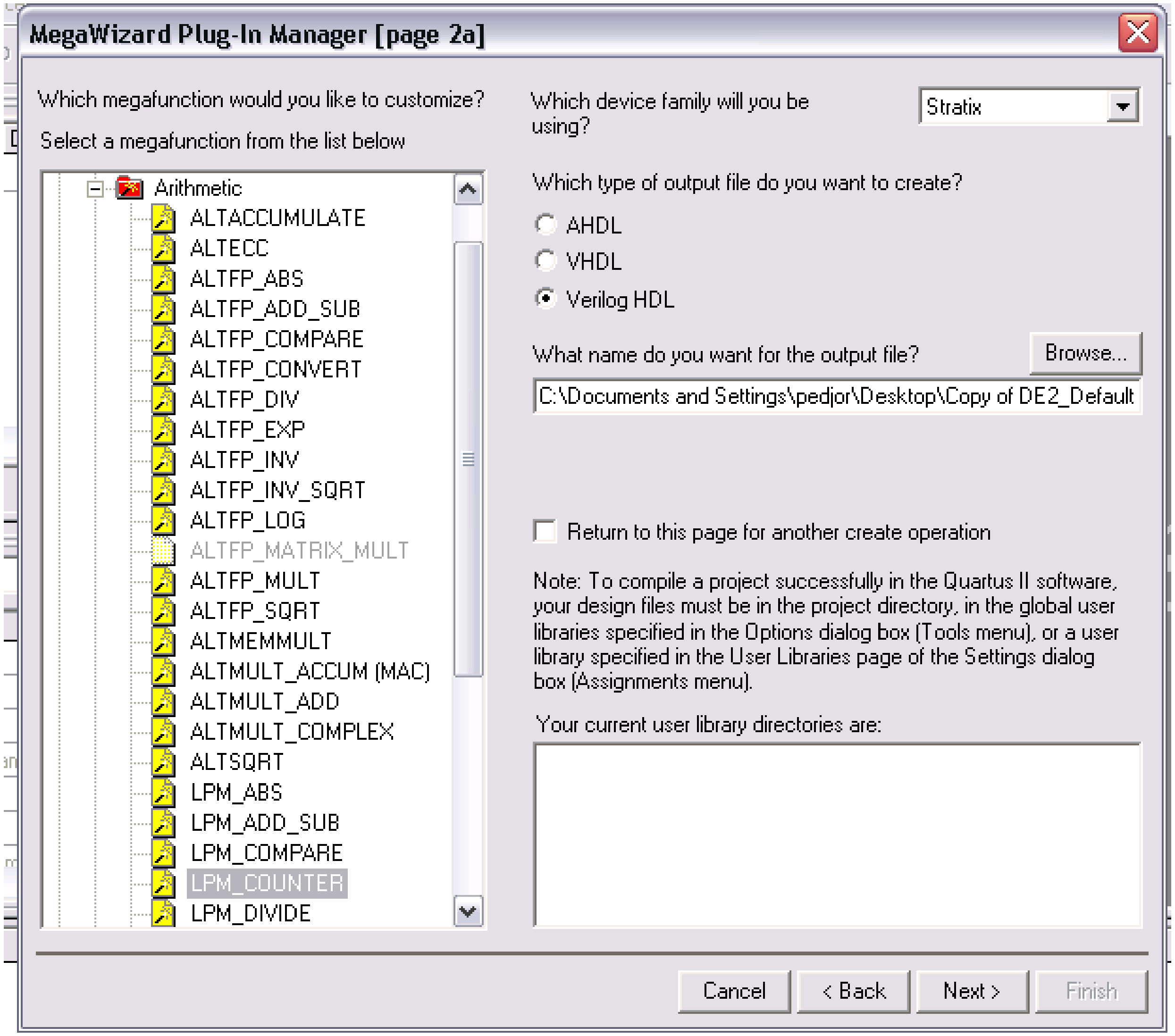
Which action do you want to perform?

- ● Create a new custom megafunction variation
- ○ Edit an existing custom megafunction variation
- ○ Copy an existing custom megafunction variation

Copyright (C) 1991-2009 Altera Corporation

Cancel | < Back | Next > | Finish

Starts the MegaWizard Plug-In Manager



Which megafunction would you like to customize?

Select a megafunction from the list below

- Arithmetic
  - ALTACCUMULATE
  - ALTECC
  - ALTFP_ABS
  - ALTFP_ADD_SUB
  - ALTFP_COMPARE
  - ALTFP_CONVERT
  - ALTFP_DIV
  - ALTFP_EXP
  - ALTFP_INV
  - ALTFP_INV_SQRT
  - ALTFP_LOG
  - ALTFP_MATRIX_MULT
  - ALTFP_MULT
  - ALTFP_SQRT
  - ALTMEMMULT
  - ALTMULT_ACCUM (MAC)
  - ALTMULT_ADD
  - ALTMULT_COMPLEX
  - ALTSQRT
  - LPM_ABS
  - LPM_ADD_SUB
  - LPM_COMPARE
  - LPM_COUNTER
  - LPM_DIVIDE

Which device family will you be using? Stratix

Which type of output file do you want to create?

○ AHDL
○ VHDL
● Verilog HDL

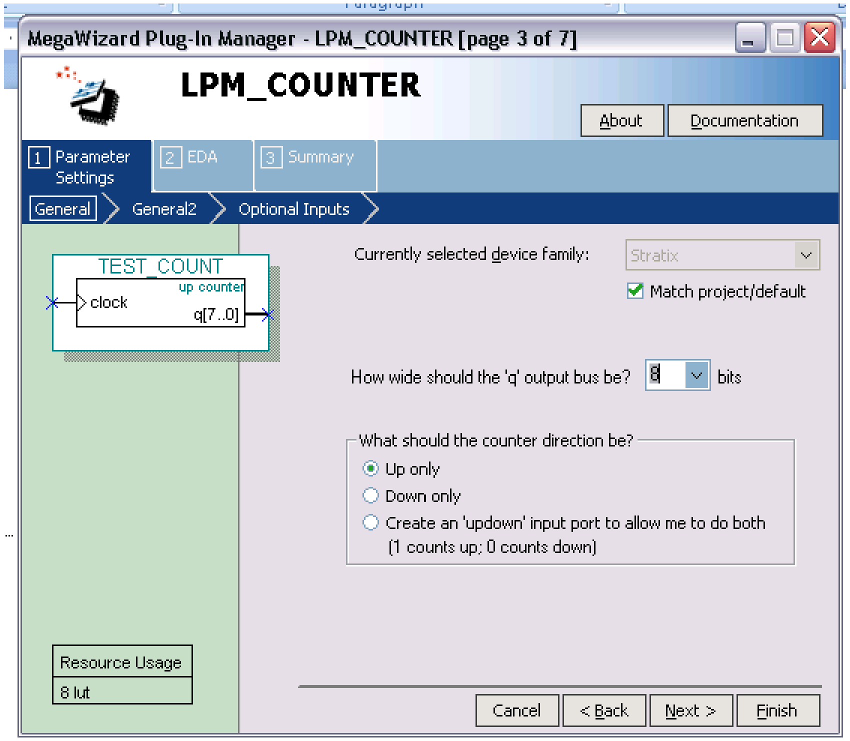
What name do you want for the output file? Browse...

C:\Documents and Settings\pedjor\Desktop\Copy of DE2_Default

□ Return to this page for another create operation

Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:



MegaWizard Plug-In Manager - LPM_COUNTER [page 3 of 7]

LPM_COUNTER

Currently selected device family: Stratix

☑ Match project/default

How wide should the 'q' output bus be? 8 bits

What should the counter direction be?

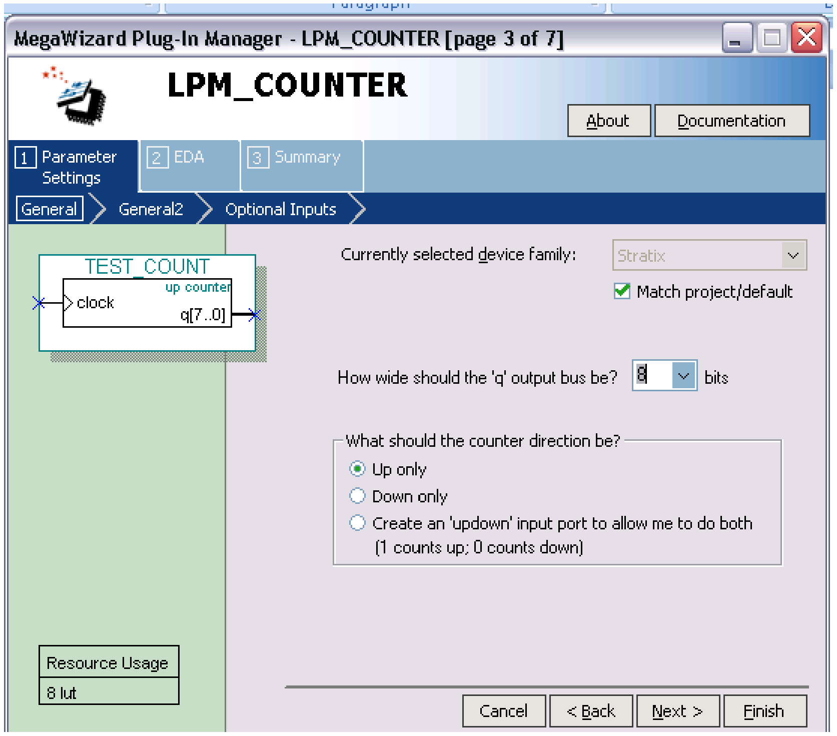
● Up only

○ Down only

○ Create an 'updown' input port to allow me to do both (1 counts up; 0 counts down)

Resource Usage

8 lut



MegaWizard Plug-In Manager - LPM_COUNTER [page 3 of 7]

LPM_COUNTER

Currently selected device family: Stratix

☑ Match project/default

How wide should the 'q' output bus be? 8 bits

What should the counter direction be?

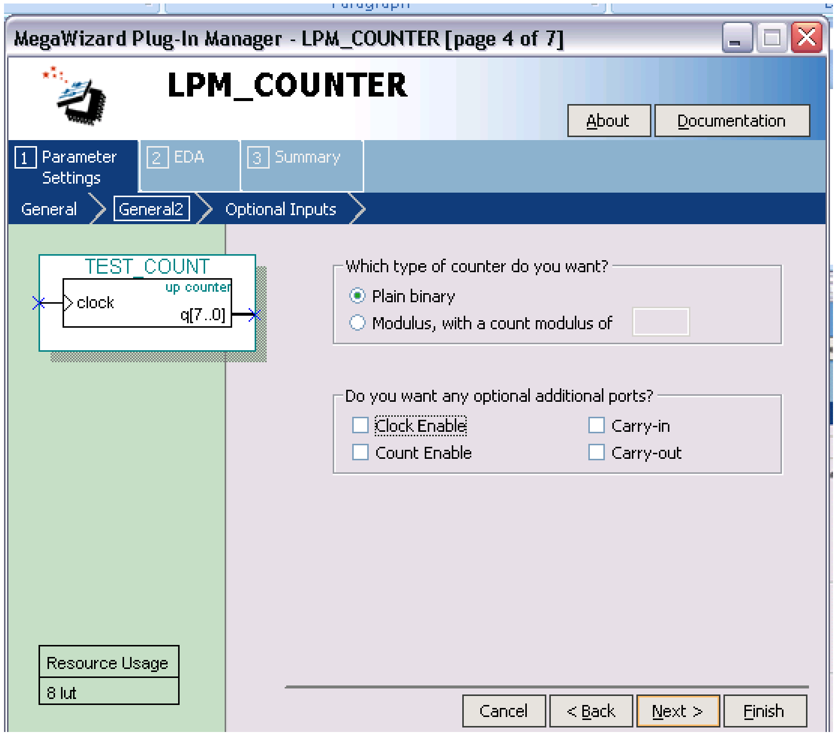
● Up only

○ Down only

○ Create an 'updown' input port to allow me to do both (1 counts up; 0 counts down)

Resource Usage

8 lut



MegaWizard Plug-In Manager - LPM_COUNTER [page 4 of 7]

LPM_COUNTER

Which type of counter do you want?

● Plain binary

○ Modulus, with a count modulus of

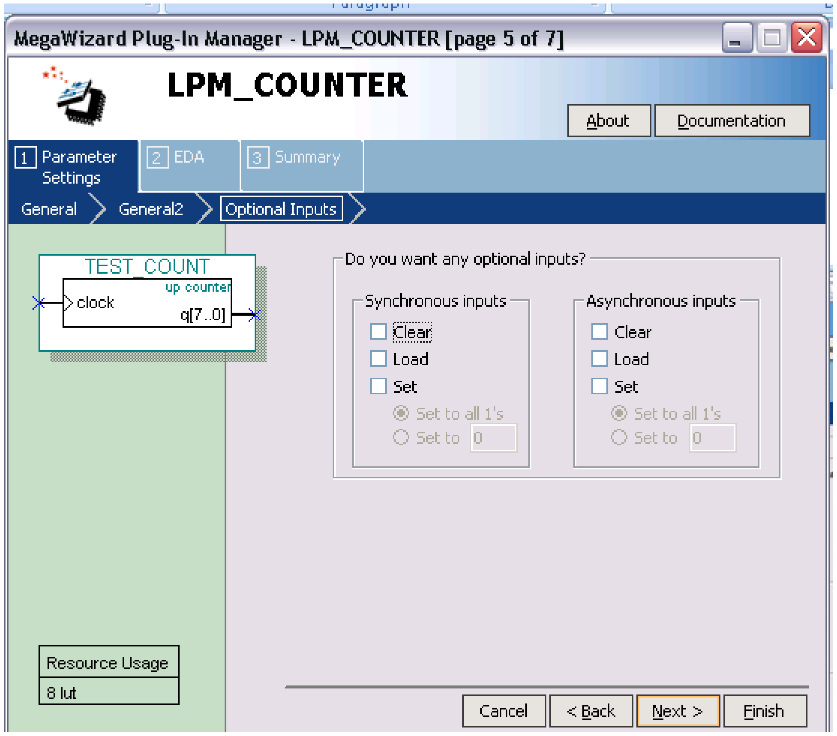
Do you want any optional additional ports?

☐ Clock Enable ☐ Carry-in

☐ Count Enable ☐ Carry-out

Resource Usage

8 lut



MegaWizard Plug-In Manager - LPM_COUNTER [page 5 of 7]

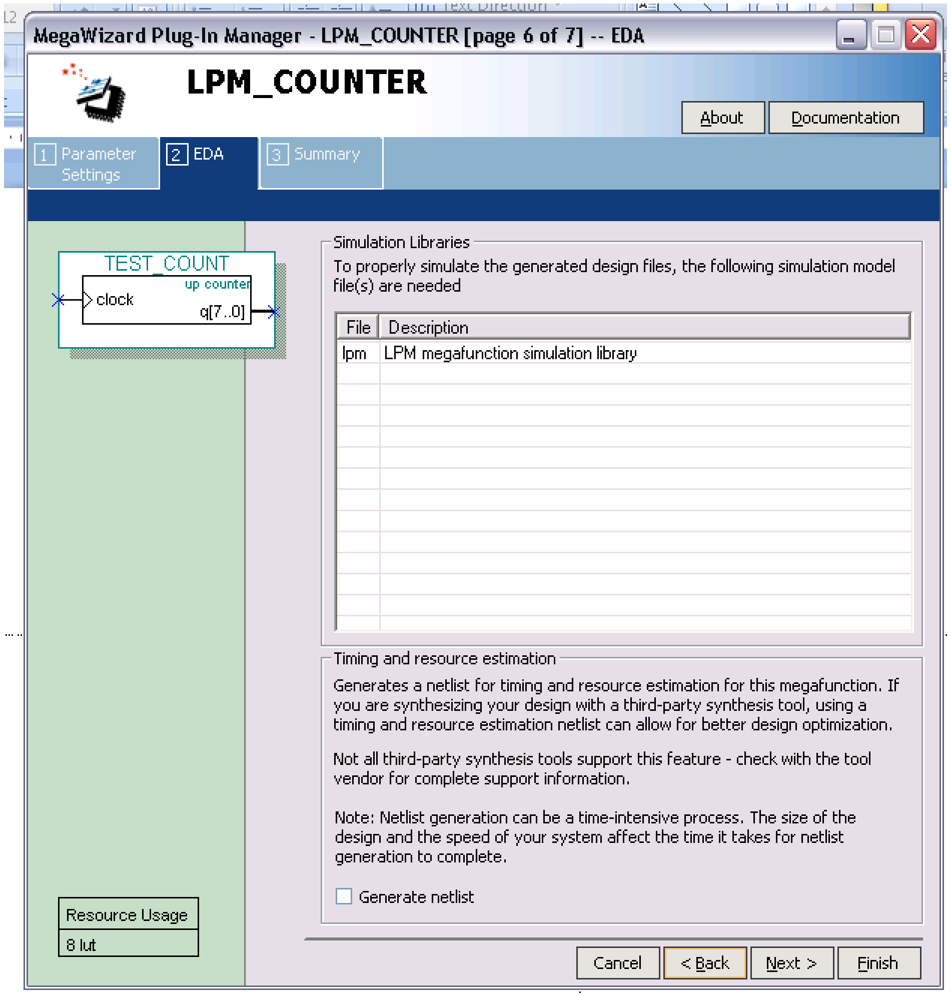
LPM_COUNTER

Do you want any optional inputs?

Synchronous inputs	Asynchronous inputs
☐ Clear	☐ Clear
☐ Load	☐ Load
☐ Set	☐ Set
● Set to all 1's	● Set to all 1's
○ Set to 0	○ Set to 0

Resource Usage

8 lut



Simulation Libraries

To properly simulate the generated design files, the following simulation model file(s) are needed

File	Description
lpm	LPM megafunction simulation library

Timing and resource estimation

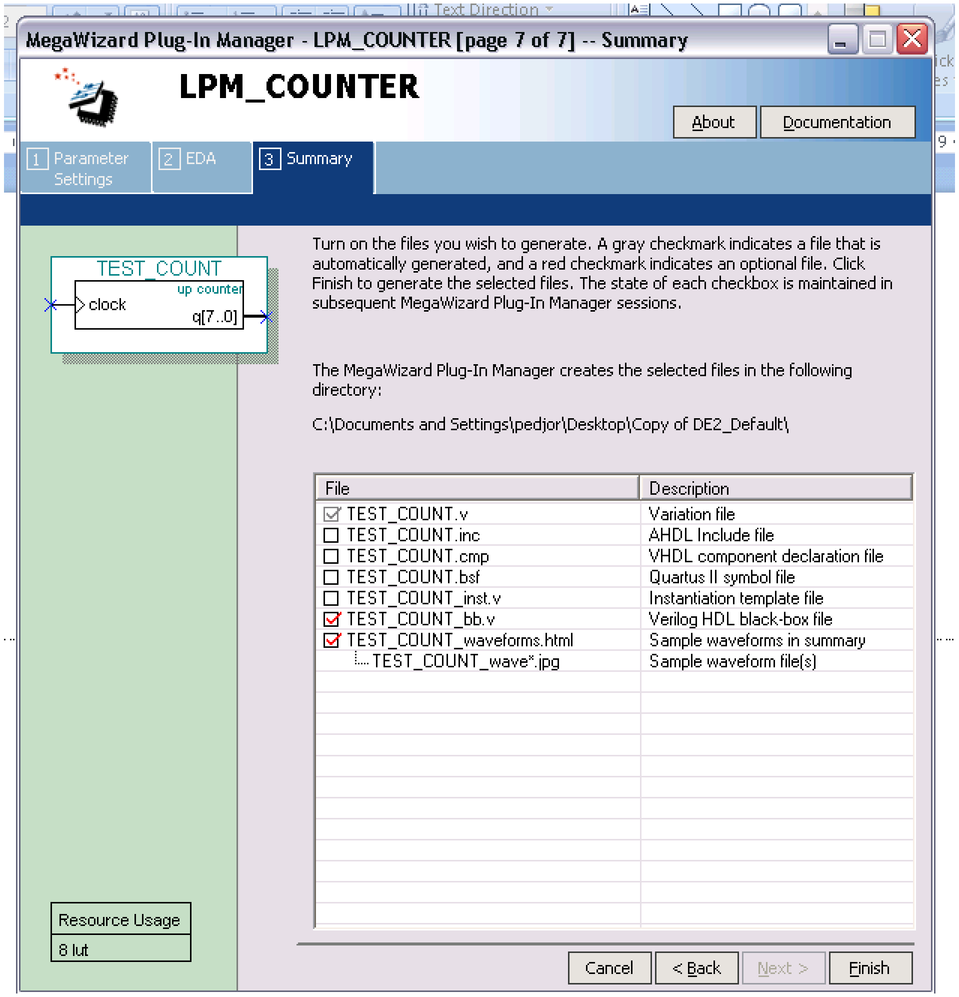
Generates a netlist for timing and resource estimation for this megafunction. If you are synthesizing your design with a third-party synthesis tool, using a timing and resource estimation netlist can allow for better design optimization.

Not all third-party synthesis tools support this feature - check with the tool vendor for complete support information.

Note: Netlist generation can be a time-intensive process. The size of the design and the speed of your system affect the time it takes for netlist generation to complete.

☐ Generate netlist

Resource Usage
8 lut



Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

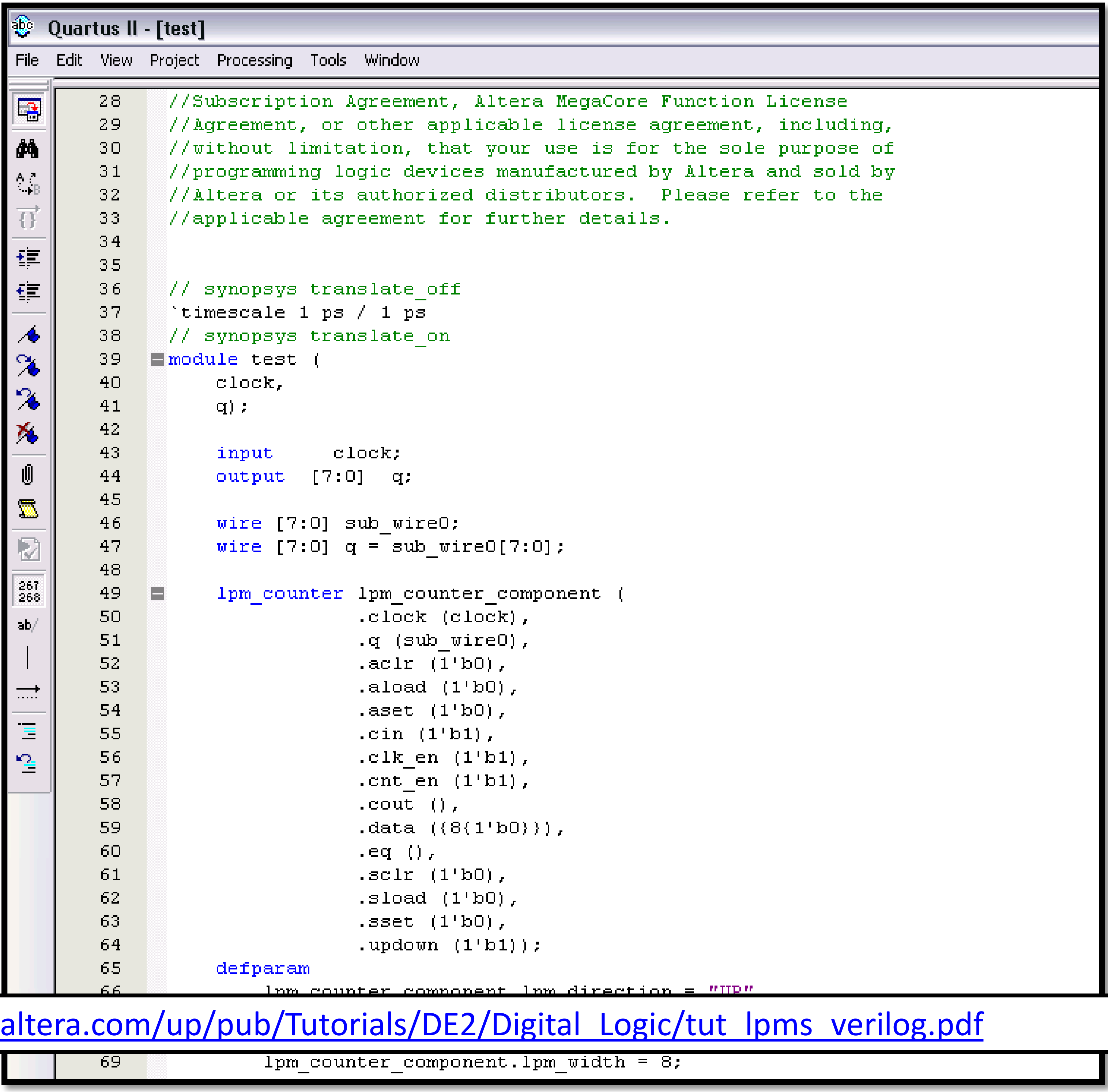
The MegaWizard Plug-In Manager creates the selected files in the following directory:

C:\Documents and Settings\pedjor\Desktop\Copy of DE2_Default\

File	Description
☑ TEST_COUNT.v	Variation file
☐ TEST_COUNT.inc	AHDL Include file
☐ TEST_COUNT.cmp	VHDL component declaration file
☐ TEST_COUNT.bsf	Quartus II symbol file
☐ TEST_COUNT_inst.v	Instantiation template file
☑ TEST_COUNT_bb.v	Verilog HDL black-box file
☑ TEST_COUNT_waveforms.html	Sample waveforms in summary
TEST_COUNT_wave*.jpg	Sample waveform file(s)

Resource Usage
8 lut

Mega Wizard – What you get

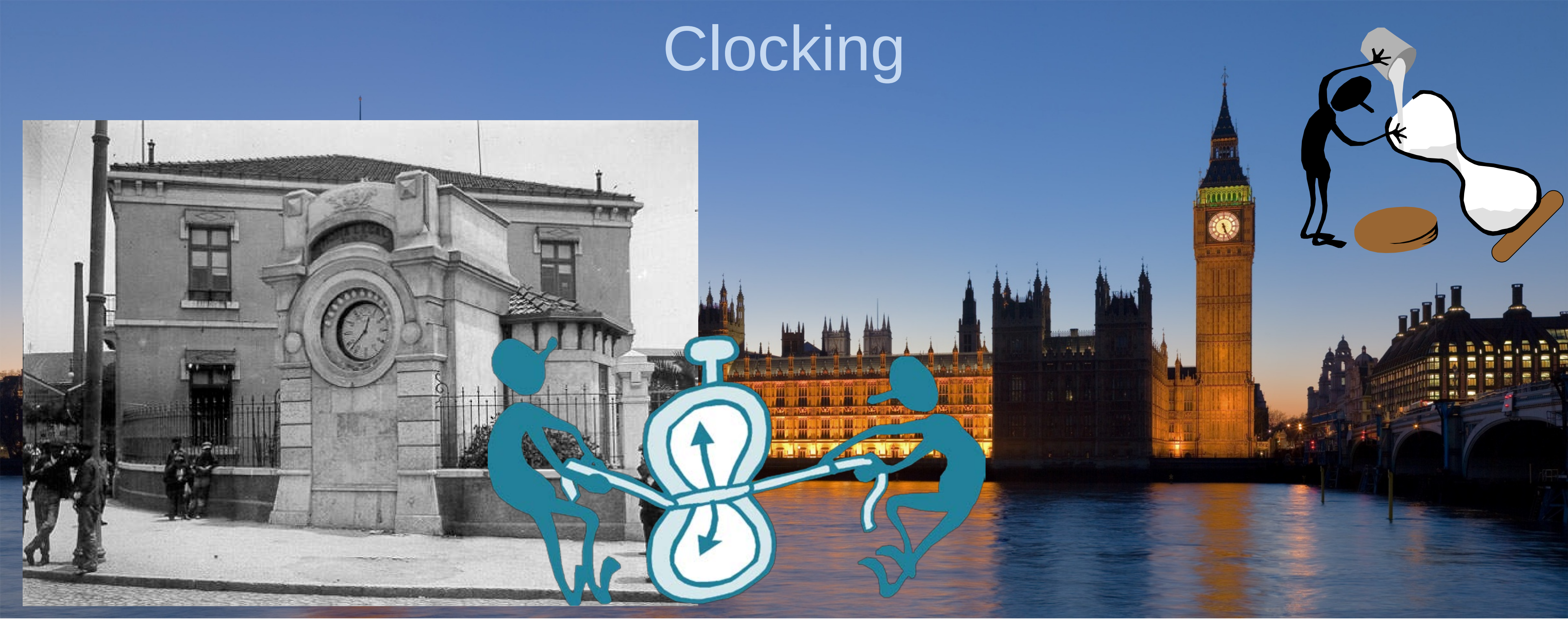


```
//Subscription Agreement, Altera MegaCore Function License
//Agreement, or other applicable license agreement, including,
//without limitation, that your use is for the sole purpose of
//programming logic devices manufactured by Altera and sold by
//Altera or its authorized distributors.  Please refer to the
//applicable agreement for further details.


// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module test (
    clock,
    q);

    input     clock;
    output  [7:0]  q;

    wire [7:0] sub_wire0;
    wire [7:0] q = sub_wire0[7:0];

    lpm_counter lpm_counter_component (
                .clock (clock),
                .q (sub_wire0),
                .aclr (1'b0),
                .aload (1'b0),
                .aset (1'b0),
                .cin (1'b1),
                .clk_en (1'b1),
                .cnt_en (1'b1),
                .cout (),
                .data ({8{1'b0}}),
                .eq (),
                .sclr (1'b0),
                .sload (1'b0),
                .sset (1'b0),
                .updown (1'b1));
    defparam
        lpm_counter_component.lpm_direction = "UP",
        lpm_counter_component.lpm_width = 8;
```

ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital_Logic/tut_lpms_verilog.pdf

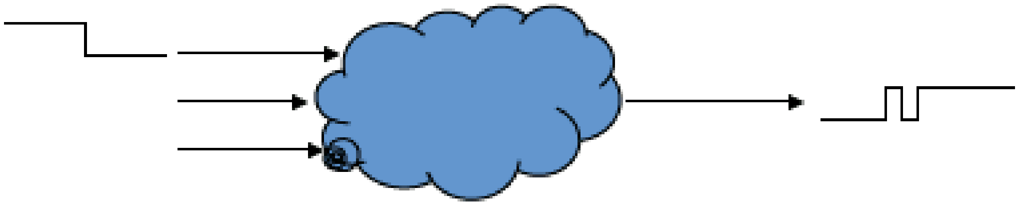
STOP

Clocking

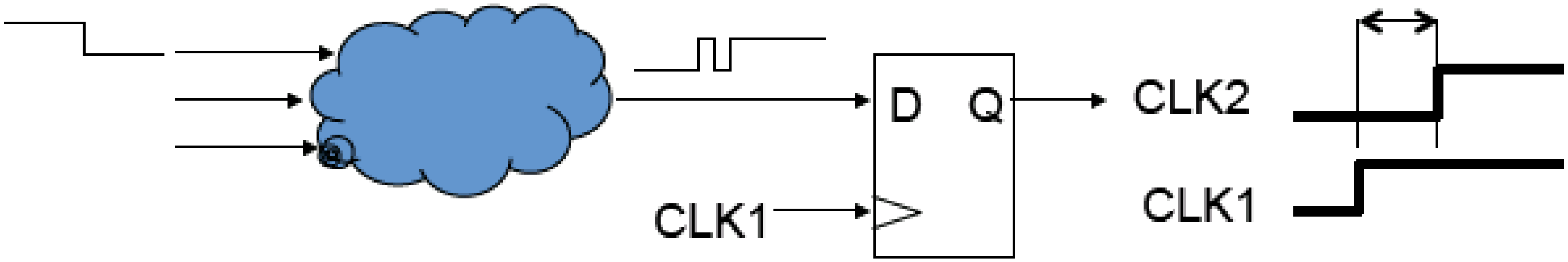


Where should CLK come from?

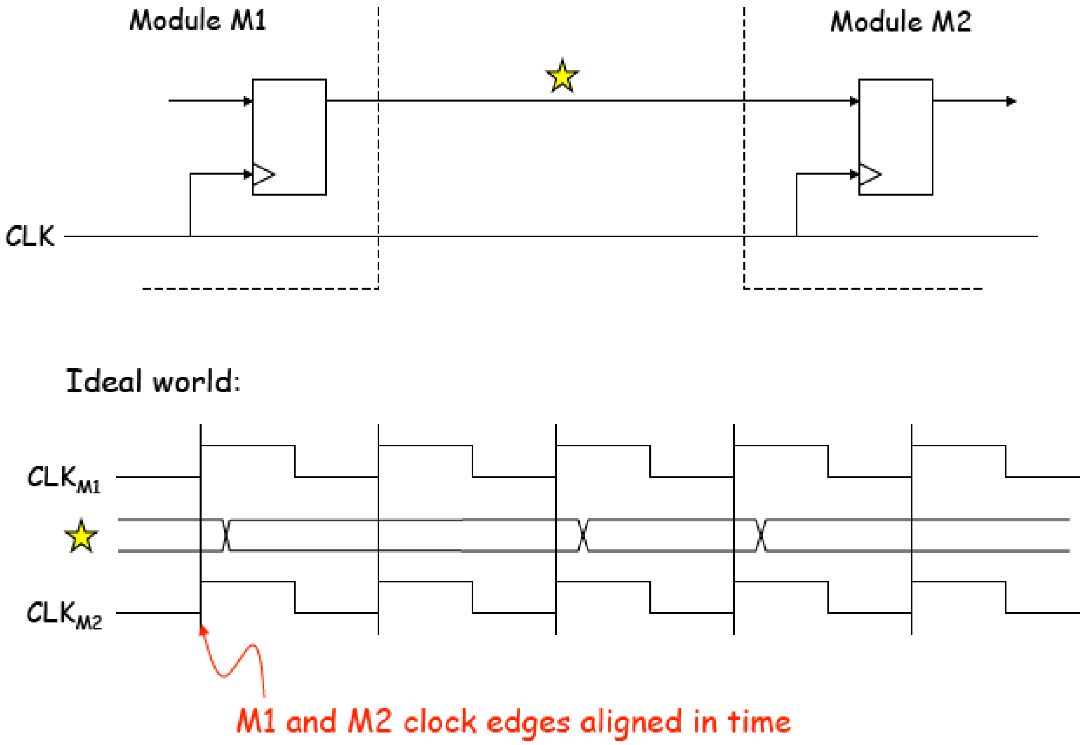
- Option 1: external crystal
  - Stable, known frequency, typically 50% duty cycle
- Option 2: internal signals
  - Option 2A: output of combinational logic



    - No! If inputs to logic change, output may make several transitions before settling to final value → several rising edges, not just one! Hard to design away output glitches…
  - Option 2B: output of a register
    - Okay, but timing of CLK2 won't line up with CLK1



Clocking in and ideal world

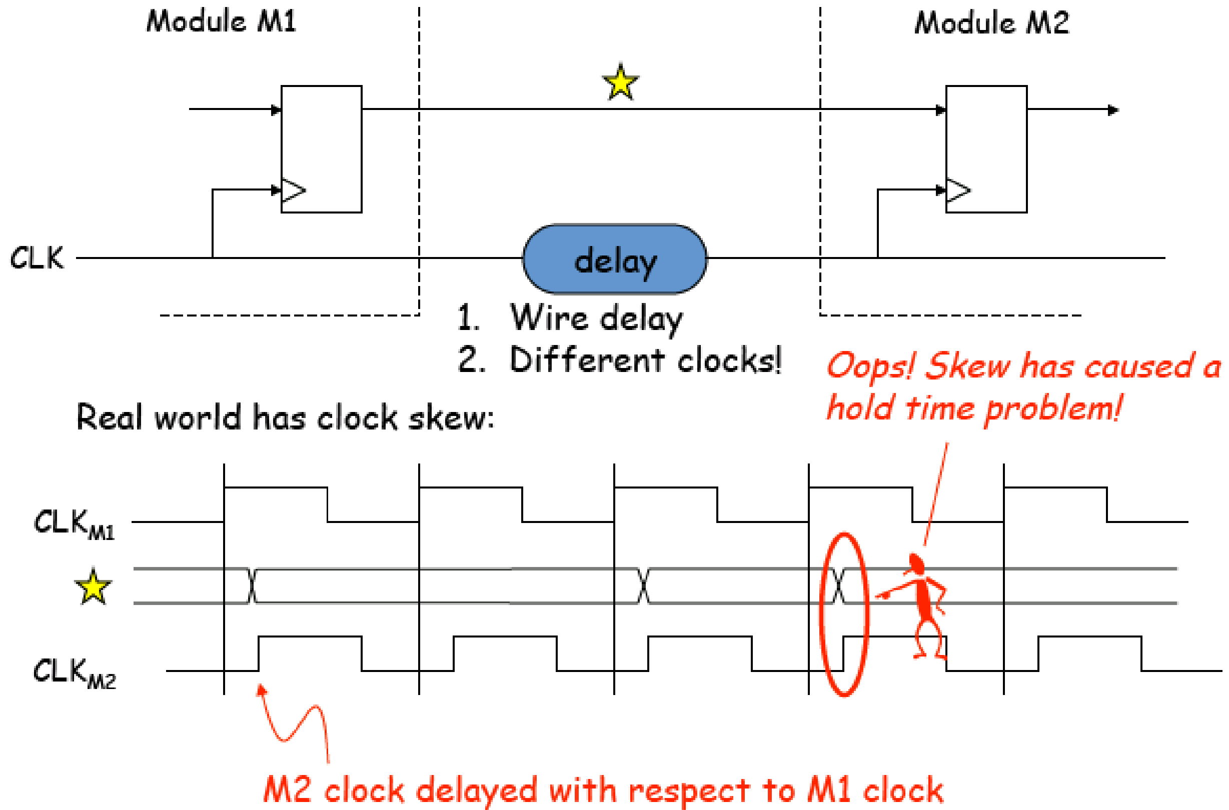


Ideal world:

M1 and M2 clock edges aligned in time

Clocking in a not so perfect reality

FPGAs contain global clock lines that have low skew. Number is limited



1. Wire delay
2. Different clocks!

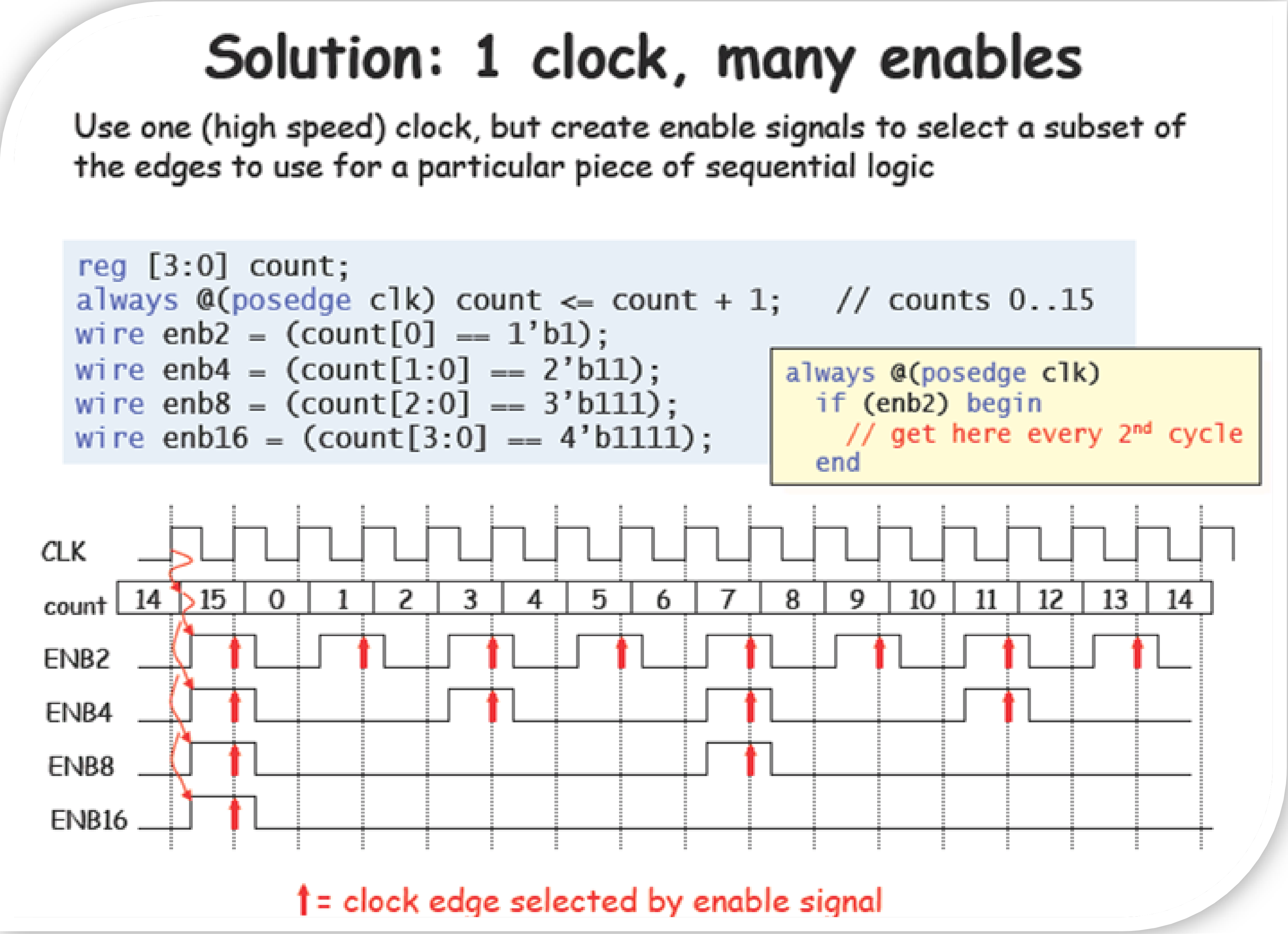
Real world has clock skew:

Oops! Skew has caused a hold time problem!

M2 clock delayed with respect to M1 clock

Clocks with periods multiple of two

```
reg clk2,clk4,clk8,clk16;
always @(posedge clk) clk2 <= ~clk2;
always @(posedge clk2) clk4 <= ~clk4;
always @(posedge clk4) clk8 <= ~clk16;
always @(posedge clk8) clk16 <= ~clk16;
```



Solution: 1 clock, many enables

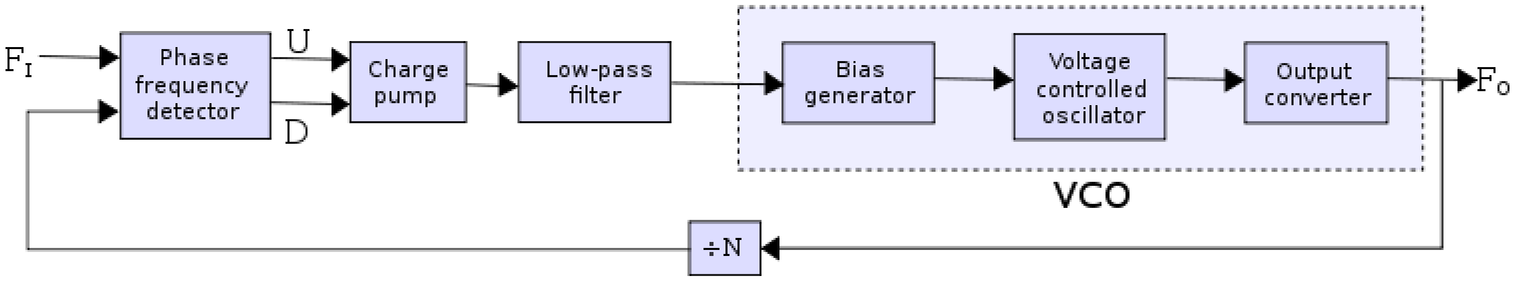
Use one (high speed) clock, but create enable signals to select a subset of the edges to use for a particular piece of sequential logic

```
reg [3:0] count;
always @(posedge clk) count <= count + 1;   // counts 0..15
wire enb2 = (count[0] == 1'b1);
wire enb4 = (count[1:0] == 2'b11);
wire enb8 = (count[2:0] == 3'b111);
wire enb16 = (count[3:0] == 4'b1111);
```

```
always @(posedge clk)
  if (enb2) begin
    // get here every 2nd cycle
  end
```

↑ = clock edge selected by enable signal

The Phase Locked Loop (PLL)



A feature available in Cyclone II devices that employs a phase-locked loop (PLL). The Cyclone II PLL provides advanced multiplication, programmable duty cycle, phase shifting, programmable bandwidth, manual clock switchover, clock outputs driving all networks, and a source synchronous mode.

You can take advantage of the Cyclone II PLL with the altpll megafunction.

Phase-Locked Loops (ALTPLL)

Megafunction User Guide

STOP