California State University, Fullerton

Computer Engineering

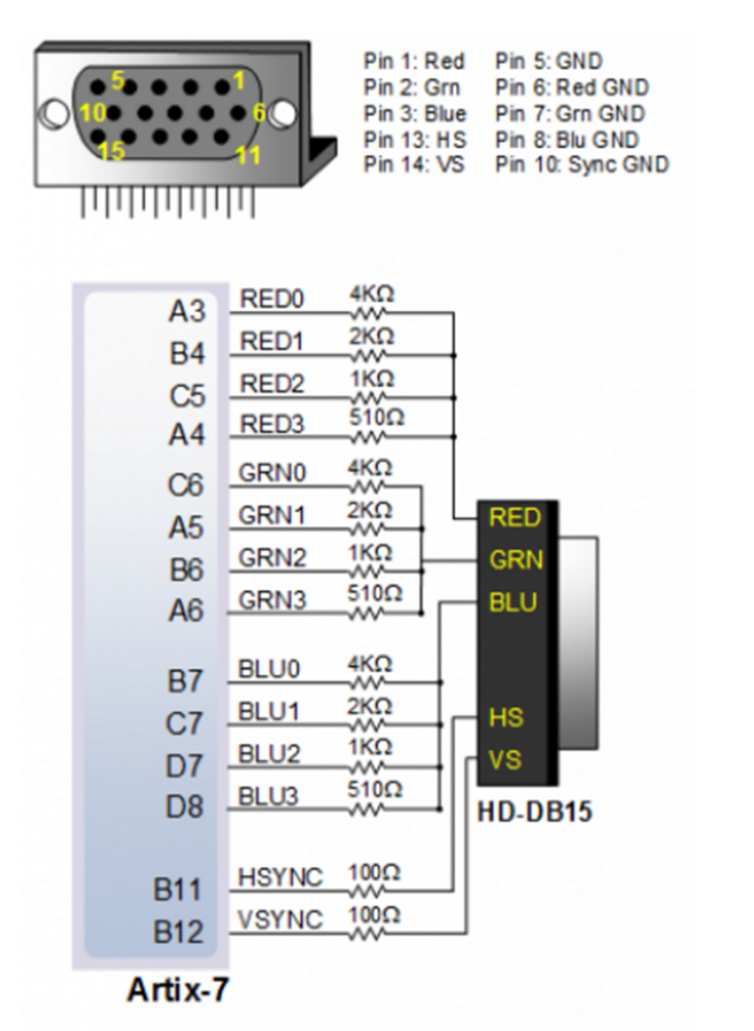
**EGCP 446 – Advanced Digital Design using Verilog HDL**

**(Fall 2019)**

**Lab No 6: VGA Interface**

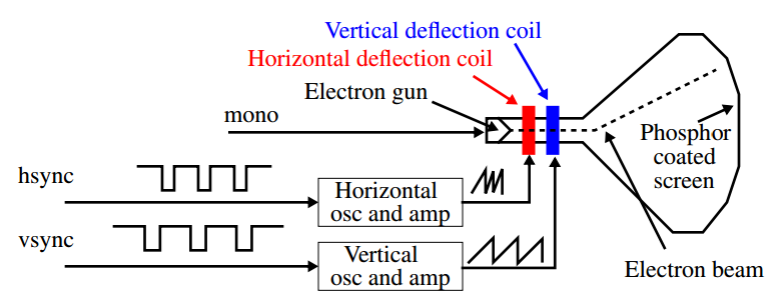
1. **Lab Description**

The Nexys4 DDR board uses 10 FPGA signals to create 8-bit color and two standard sync signals (HS-Horizontal Sync, and VS- sync).



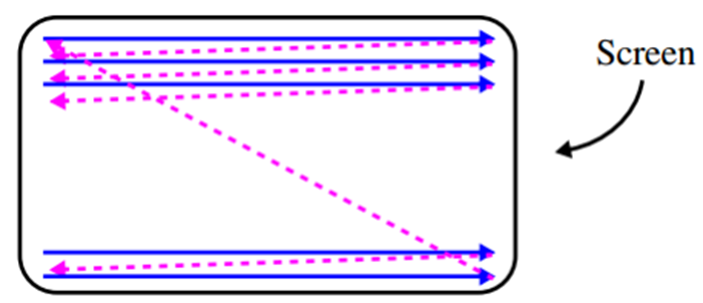
Here we consider an 8 color 640-480 pixel resolution interface for the older monochrome CRT model. The electron gun generates a focused electron beam that strikes the phosphor screen. The intensity of the electron beam and the brightness of the dot are determine by the voltage level of the external video input signal (mono signal). The mono signal is an analog signal whose voltage level is between 0 and 0.7 V.

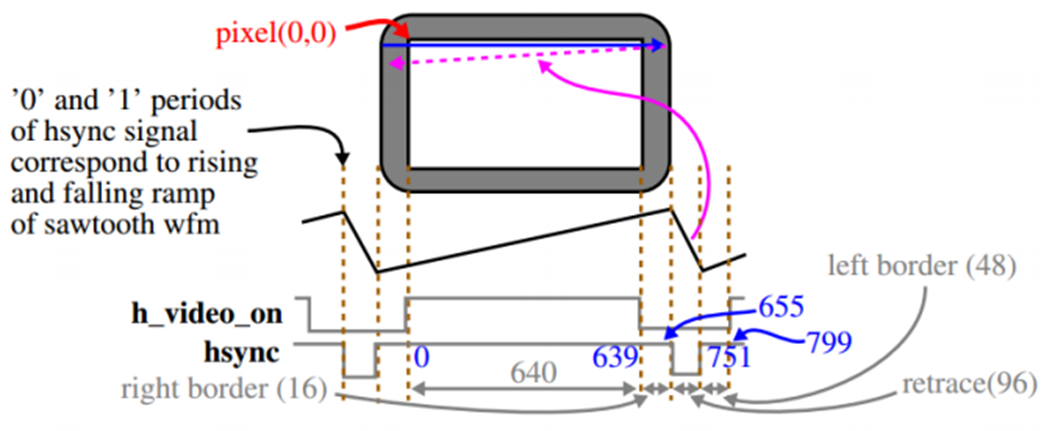
The horizontal and vertical deflection coils produce magnetic fields guide the electron beam to points on the screen.



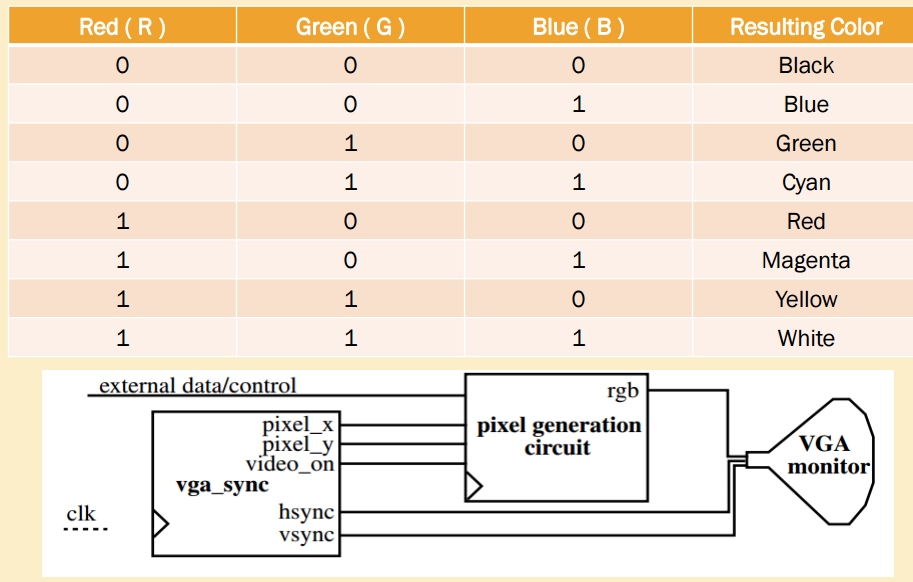
A color CRT is similar except that it has three electron beams, that are projected to the red, green and blue phosphor dots on the screen. The three dots are combined to form a pixel. The three voltage levels determine the intensity of each and therefore the color. The VGA port has five active signals, hsync, vsync, and three video signals for the red, green and blue beams. They are connected to a 15-pin D-subminiature connector.

The video signals are analog signals -- the video controller uses a D-to-A converter to convert the digital output to the appropriate analog level. If video is represented by an N-bit word, it can be converted to 2 𝑁analog levels. Three video signals can generate 2 3𝑁 different colors (called 3N-bit color). If 1-bit is used for each video signal, we get 2 3or 8 colors. If all three video signals are driven from the same 1-bit word, we get black & white.





**Video Controller:**



**Lab No 6 – Task**

* Create a LabProjA project and import the 3 provided Verilog files as well as the provided constraints file.
* Modify the Pong\_graph\_st.v to display a blue horizontal strip of width 40 pixels in the middle of the screen and a green vertical strip of width 20 pixels in the middle of the screen.
* Generate the bitstream and program the FPGA with your design.
* Verify that the hardware’s behavior works as expected.
* Submit all your Verilog code and constrain files. Also, include the picture of your VGA screen.

Submit your Verilog code here

// Listing 13.4

module pong\_top\_st

(

input wire clk, reset,

output wire hsync, vsync,

output wire [2:0] rgb

);

// signal declaration

wire [9:0] pixel\_x, pixel\_y;

wire video\_on, pixel\_tick;

reg [2:0] rgb\_reg;

wire [2:0] rgb\_next;

// body

// instantiate vga sync circuit

vga\_sync vsync\_unit

(.clk(clk), .reset(reset), .hsync(hsync), .vsync(vsync),

.video\_on(video\_on), .p\_tick(pixel\_tick),

.pixel\_x(pixel\_x), .pixel\_y(pixel\_y));

// instantiate graphic generator

pong\_graph\_st pong\_grf\_unit

(.video\_on(video\_on), .pix\_x(pixel\_x), .pix\_y(pixel\_y),

.graph\_rgb(rgb\_next));

// rgb buffer

always @(posedge clk)

if (pixel\_tick)

rgb\_reg <= rgb\_next;

// output

assign rgb = rgb\_reg;

endmodule

// Listing 13.3

module pong\_graph\_st

(

input wire video\_on,

input wire [9:0] pix\_x, pix\_y,

output reg [2:0] graph\_rgb

);

// constant and signal declaration

// x, y coordinates (0,0) to (639,479)

localparam MAX\_X = 640;

localparam MAX\_Y = 480;

//--------------------------------------------

// vertical stripe as a wall

//--------------------------------------------

// wall left, right boundary

localparam WALL\_X\_L = 300;

localparam WALL\_X\_R = 340;

//--------------------------------------------

// right vertical bar

//--------------------------------------------

// bar left, right boundary

localparam BAR\_X\_L = 0;

localparam BAR\_X\_R = 640;

// bar top, bottom boundary

localparam BAR\_Y\_SIZE = 20;

localparam BAR\_Y\_T = MAX\_Y/2-BAR\_Y\_SIZE/2; //204

localparam BAR\_Y\_B = BAR\_Y\_T+BAR\_Y\_SIZE-1;

//--------------------------------------------

// square ball

//--------------------------------------------

localparam BALL\_SIZE = 0;

// ball left, right boundary

localparam BALL\_X\_L = 580;

localparam BALL\_X\_R = BALL\_X\_L+BALL\_SIZE-1;

// ball top, bottom boundary

localparam BALL\_Y\_T = 238;

localparam BALL\_Y\_B = BALL\_Y\_T+BALL\_SIZE-1;

//--------------------------------------------

// object output signals

//--------------------------------------------

wire wall\_on, bar\_on, sq\_ball\_on;

wire [2:0] wall\_rgb, bar\_rgb, ball\_rgb;

// body

//--------------------------------------------

// (wall) left vertical strip

//--------------------------------------------

// pixel within wall

assign wall\_on = (WALL\_X\_L<=pix\_x) && (pix\_x<=WALL\_X\_R);

// wall rgb output

assign wall\_rgb = 3'b001; // blue

//--------------------------------------------

// right vertical bar

//--------------------------------------------

// pixel within bar

assign bar\_on = (BAR\_X\_L<=pix\_x) && (pix\_x<=BAR\_X\_R) &&

(BAR\_Y\_T<=pix\_y) && (pix\_y<=BAR\_Y\_B);

// bar rgb output

assign bar\_rgb = 3'b010; // green

//--------------------------------------------

// square ball

//--------------------------------------------

// pixel within squared ball

assign sq\_ball\_on =

(BALL\_X\_L<=pix\_x) && (pix\_x<=BALL\_X\_R) &&

(BALL\_Y\_T<=pix\_y) && (pix\_y<=BALL\_Y\_B);

assign ball\_rgb = 3'b100; // red

//--------------------------------------------

// rgb multiplexing circuit

//--------------------------------------------

always @\*

if (~video\_on)

graph\_rgb = 3'b000; // blank

else

if (wall\_on)

graph\_rgb = wall\_rgb;

else if (bar\_on)

graph\_rgb = bar\_rgb;

else if (sq\_ball\_on)

graph\_rgb = ball\_rgb;

else

graph\_rgb = 3'b110; // yellow background

endmodule

// Listing 13.1

module vga\_sync

(

input wire clk, reset,

output wire hsync, vsync, video\_on, p\_tick,

output wire [9:0] pixel\_x, pixel\_y

);

// constant declaration

// VGA 640-by-480 sync parameters

localparam HD = 640; // horizontal display area

localparam HF = 48 ; // h. front (left) border

localparam HB = 16 ; // h. back (right) border

localparam HR = 96 ; // h. retrace

localparam VD = 480; // vertical display area

localparam VF = 10; // v. front (top) border

localparam VB = 33; // v. back (bottom) border

localparam VR = 2; // v. retrace

// mod-2 counter

reg mod2\_reg;

wire mod2\_next;

// sync counters

reg [9:0] h\_count\_reg, h\_count\_next;

reg [9:0] v\_count\_reg, v\_count\_next;

// output buffer

reg v\_sync\_reg, h\_sync\_reg;

wire v\_sync\_next, h\_sync\_next;

// status signal

wire h\_end, v\_end, pixel\_tick;

// body

// registers

always @(posedge clk, posedge reset)

if (reset)

begin

mod2\_reg <= 2'b00;

v\_count\_reg <= 0;

h\_count\_reg <= 0;

v\_sync\_reg <= 1'b0;

h\_sync\_reg <= 1'b0;

end

else

begin

mod2\_reg <= mod2\_next;

v\_count\_reg <= v\_count\_next;

h\_count\_reg <= h\_count\_next;

v\_sync\_reg <= v\_sync\_next;

h\_sync\_reg <= h\_sync\_next;

end

// mod-2 circuit to generate 25 MHz enable tick

assign mod2\_next = ~mod2\_reg;

// assign mod2\_next = mod2\_reg + 1'b1;

assign pixel\_tick = mod2\_reg;

// status signals

// end of horizontal counter (799)

assign h\_end = (h\_count\_reg==(HD+HF+HB+HR-1));

// end of vertical counter (524)

assign v\_end = (v\_count\_reg==(VD+VF+VB+VR-1));

// next-state logic of mod-800 horizontal sync counter

always @\*

if (pixel\_tick) // 25 MHz pulse

if (h\_end)

h\_count\_next = 0;

else

h\_count\_next = h\_count\_reg + 1;

else

h\_count\_next = h\_count\_reg;

// next-state logic of mod-525 vertical sync counter

always @\*

if (pixel\_tick & h\_end)

if (v\_end)

v\_count\_next = 0;

else

v\_count\_next = v\_count\_reg + 1;

else

v\_count\_next = v\_count\_reg;

// horizontal and vertical sync, buffered to avoid glitch

// h\_sync\_next asserted between 656 and 751

assign h\_sync\_next = (h\_count\_reg>=(HD+HB) &&

h\_count\_reg<=(HD+HB+HR-1));

// vh\_sync\_next asserted between 490 and 491

assign v\_sync\_next = (v\_count\_reg>=(VD+VB) &&

v\_count\_reg<=(VD+VB+VR-1));

// video on/off

assign video\_on = (h\_count\_reg<HD) && (v\_count\_reg<VD);

// output

assign hsync = h\_sync\_reg;

assign vsync = v\_sync\_reg;

assign pixel\_x = h\_count\_reg;

assign pixel\_y = v\_count\_reg;

assign p\_tick = pixel\_tick;

endmodule

# Xilinx Spartan3 Starter Board (Rev E) UCF file

# by Brian Szmyd

# Updated by Wimbo

# Clock

Net "clk" LOC = "T9";

NET "clk" PERIOD = 20ns; # 20ns = 50Mhz

#Net "Socket" LOC = "D9";

#NET "Socket" PERIOD = ??????? ns;

# User Switches

NET "reset" LOC = "F12";

#NET "SW1" LOC = "G12";

#NET "SW2" LOC = "H14";

#NET "SW3" LOC = "H13";

#NET "SW4" LOC = "J14";

#NET "SW5" LOC = "J13";

#NET "SW6" LOC = "K14";

#NET "SW7" LOC = "K13";

# User Buttons

#NET "BTN0" LOC = "M13";

#NET "BTN1" LOC = "M14";

#NET "BTN2" LOC = "L13";

#NET "BTN3" LOC = "L14";

# LEDs

#NET "LD0" LOC = "K12";

#NET "LD1" LOC = "P14";

#NET "LD2" LOC = "L12";

#NET "LD3" LOC = "N14";

#NET "LD4" LOC = "P13";

#NET "LD5" LOC = "N12";

#NET "LD6" LOC = "P12";

#NET "LD7" LOC = "P11";

# 7 Segment

#NET "AN0" LOC = "D14";

#NET "AN1" LOC = "G14";

#NET "AN2" LOC = "F14";

#NET "AN3" LOC = "E13";

#NET "CA" LOC = "E14";

#NET "CB" LOC = "G13";

#NET "CC" LOC = "N15";

#NET "CD" LOC = "P15";

#NET "CE" LOC = "R16";

#NET "CF" LOC = "F13";

#NET "CG" LOC = "N16";

#NET "CDP" LOC = "P16";

# VGA Signals

NET "rgb<2>" LOC = "R12";

NET "rgb<1>" LOC = "T12";

NET "rgb<0>" LOC = "R11";

NET "hsync" LOC = "R9";

NET "vsync" LOC = "T10";

# RS-232 Port

#NET "RXD" LOC = "T13";

#NET "TXD" LOC = "R13";

#NET "RXD\_A" LOC = "N10";

#NET "TXD\_A" LOC = "T14";

# PS/2 Signals

#NET "PS2D" LOC = "M15";

#NET "PS2C" LOC = "M16";

# 1MB Surface Mounted RAM

# --------------------------------------------

# Address Bus

#NET "A<17>" LOC = "L3";

#NET "A<16>" LOC = "K5";

#NET "A<15>" LOC = "K3";

#NET "A<14>" LOC = "J3";

#NET "A<13>" LOC = "J4";

#NET "A<12>" LOC = "H4";

#NET "A<11>" LOC = "H3";

#NET "A<10>" LOC = "G5";

#NET "A<9>" LOC = "E4";

#NET "A<8>" LOC = "E3";

#NET "A<7>" LOC = "F4";

#NET "A<6>" LOC = "F3";

#NET "A<5>" LOC = "G4";

#NET "A<4>" LOC = "L4";

#NET "A<3>" LOC = "M3";

#NET "A<2>" LOC = "M4";

#NET "A<1>" LOC = "N3";

#NET "A<0>" LOC = "L5";

# Control Signals

#NET "OE\_N" LOC = "K4";

#NET "WE\_N" LOC = "G3";

#NET "CE1\_N" LOC = "P7";

#NET "CE2\_N" LOC = "N5";

#NET "UB1\_N" LOC = "T4";

#NET "UB2\_N" LOC = "R4";

#NET "LB1\_N" LOC = "P6";

#NET "LB2\_N" LOC = "P5";

# Data Signals Chip 1

#NET "IO1<15>" LOC = "R1";

#NET "IO1<14>" LOC = "P1";

#NET "IO1<13>" LOC = "L2";

#NET "IO1<12>" LOC = "J2";

#NET "IO1<11>" LOC = "H1";

#NET "IO1<10>" LOC = "F2";

#NET "IO1<9>" LOC = "P8";

#NET "IO1<8>" LOC = "D3";

#NET "IO1<7>" LOC = "B1";

#NET "IO1<6>" LOC = "C1";

#NET "IO1<5>" LOC = "C2";

#NET "IO1<4>" LOC = "R5";

#NET "IO1<3>" LOC = "T5";

#NET "IO1<2>" LOC = "R6";

#NET "IO1<1>" LOC = "T8";

#NET "IO1<0>" LOC = "N7";

# Data Signals Chip 2

#NET "IO2<15>" LOC = "N1";

#NET "IO2<14>" LOC = "M1";

#NET "IO2<13>" LOC = "K2";

#NET "IO2<12>" LOC = "C3";

#NET "IO2<11>" LOC = "F5";

#NET "IO2<10>" LOC = "G1";

#NET "IO2<9>" LOC = "E2";

#NET "IO2<8>" LOC = "D2";

#NET "IO2<7>" LOC = "D1";

#NET "IO2<6>" LOC = "E1";

#NET "IO2<5>" LOC = "G2";

#NET "IO2<4>" LOC = "J1";

#NET "IO2<3>" LOC = "K1";

#NET "IO2<2>" LOC = "M2";

#NET "IO2<1>" LOC = "N2";

#NET "IO2<0>" LOC = "P2";

