Name: _____ CM: _____

Name: _____ CM: _____ Start Date: Tuesday, Nov. 4, 2014
Due Date: Thursday, Nov. 20, 2014

ECE333 Fall 2014 Term Project: VGA Driver and FPGA Pong

This is a group project to be performed by groups of two students. The starting point of this project is a pong game written in Verilog from BIG MESS O' WIRES at http://www.bigmessowires.com/2009/06/21/fpga-pong/. The project will port the pong game to the Nexys 3 board from Digilent, Inc., create a new VGA timing controller, and add sounds and other new features to the pong game. The project has intermediate deadlines as well as a final deadline by 12noon on Thursday, November 20 of the final exam week.

I. Deliverables

- *Port the pong game to your Nexys 3 board with rotary encoder. (10%)*
- *Create a VGA CRT controller to replace the video_timer of the pong game. (40%)*
- *Add sounds to the pong game so that different sounds are activated when the paddle moves in one or the other direction or the ball is hit or missed. (10%)*
- *Add a new feature whereby the ball is not released at the start of game or following a "miss" until a pushbutton is pressed. Following the button press, the ball should be released from a random position near the top of the playing field. (15%)*
- *Improve the pong game by adding at least two additional new features to it. (25%)*
- *Demonstrate your project to the instructor before or by the final exam scheduled time slot from 8:00am – 12noon on Thursday, November 20 of the final exam week, which is the deadline for this project. You are encouraged to demonstrate your project early.*
- *Write a one-page memo to explain what new features you have added to the pong game.*
- *Submit annotated simulation waveforms and annotated oscilloscope waveforms for VGA CRT controller to verify that the hsync and vsync signals have appropriate periods, and x and y coordinates are synched.*
- *rar or zip your term project folder for the pong game that contains all source files and submit it to Moodle.*

II. Deadlines

Demonstrate your working circuits on your Nexys3 board to your instructor and submit hard copies of deliverables by the end of the day on each of the following deadlines. The final deadline is during the finals week presentation on Thursday, November 20.

Task	Deadline	Instructor Signature
Porting the pong game to Nexys 3, with rotary encoder	Thursday, November 6	
CRT Controller with simulations and on Nexys 3	Thursday, November 13	
Sounds and random ball release	Thursday, November 20	
Two additional features and project demonstration	Thursday, November 20	

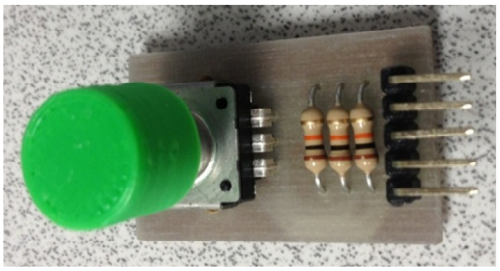
III. Specifications and Objectives of Five Tasks

This section describes specifications and objectives.

III.1 To port the pong from reference [2] to Nexys 3 board. (10%)

Make a folder called TermProjectPhase1. Copy the pong game to this folder. Rename the pong top level module, PongNexys3. Port the pong game to your Nexys 3 board. Use a rotary encoder to control the paddle. Change the colors of the ball and paddle to make them more appealing. Here are what you need to do.

(1) Add a clock of 50MHz.
(2) Change the colors to 3 bits for RED and GREEN and 2 bits for BLUE. You will need to change the output wires as well as the color generation assignments in the game module to the right bits.
(3) Remove as many warnings in the pong circuit as possible.
(4) Create an ucf file for Nexys 3. Do not worry about the driving capabilities such as IOSTANDARD, DRIVE, SLEW, etc. The UCF file with them would still work.



The Encoder has five pins. They are from right to left, power, ground, A, B and pushbutton. The pushbutton has no function. When you plug the encoder on a connector, say, connector A, here are how the pins are connected: JA6=power, JA5=Ground, JA4=A, JA3=B.

III.2 To create a portable VGA CRT controller (40%)

Make a folder called TermProjectPhase2. Download PongDriverTemplate2014fall.rar folder from Moodle. Copy the source files in the template to your new folder. Change the top level module to PongWithCRTdriver.v. Create your CRT driver. Add your names, campus mail numbers, start date and finish date in the header of each file you have revised.

This VGA CRT controller assumes 640x480 resolution, 60Hz screen refreshing rate, and 25MHz CRT module clock speed and generates the following hsync and vsync signals as well as x and y positions. Notice that active video starts at zero count and the sync pulses appear at the end of active video plus front porch. This is to match the video timer of the original pong that generates vsync and hsync as follows.

```
hsync <= ~(xpos > 664 && xpos <= 759);  xposition<=xcount;  // active for 96 clocks
vsync <= ~(ypos == 490 || ypos == 491);  yposition<=ycount;  // active for lines 490 and 491
```

800x520

Total Pixels	Active Video	Front Porch	Synch Pulse	Back porch
T_s	T_{disp}	T_{fp}	T_{pw}	T_{bp}
800	640	25	95	40
31.77 µs	25.42 µs	0.9928 µs	3.772 µs	1.589 µs

The vsync specification is as follows.

Total Pixels	Active Video	Front Porch	Synch Pulse	Back Porch
T_s	T_{disp}	T_{fp}	T_{pw}	T_{bp}
520	480	9	2	29
16.5 ms	15.238 ms	0.286 ms	0.0635 ms	0.921 ms

The following timing waveforms show the synch pulses at the start of the counts. The labels are defined as follows: B – hsync pulse; C – Back Porch; D – Active Line Display; E – Front Porch; P – vsync pulse; Q – Back Porch; R – Number of Lines; S – Front Porch.

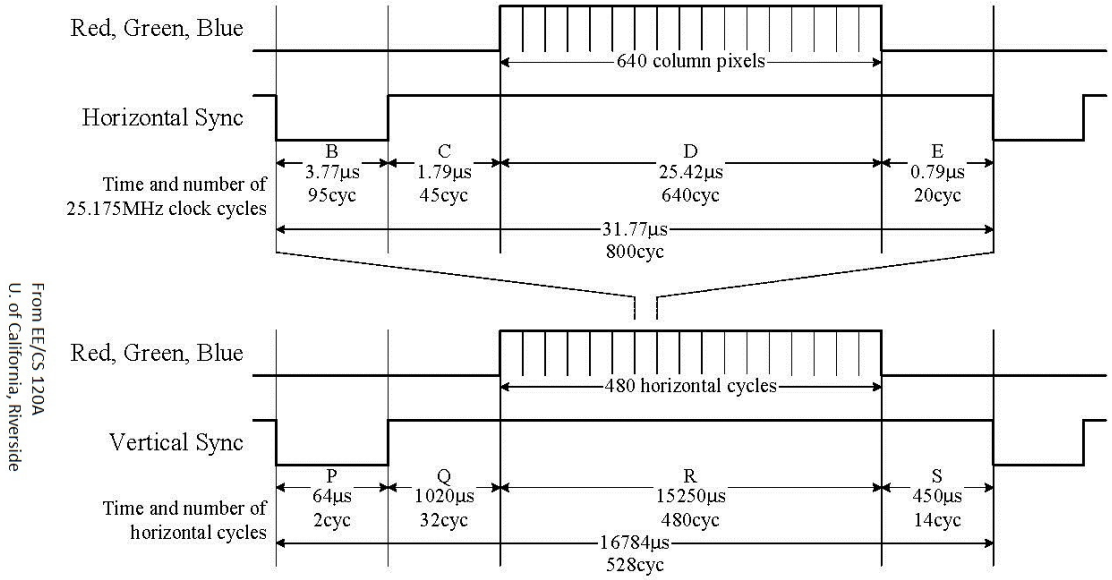


Figure 2. Horizontal and vertical synchronization signals timing diagram for a 25.175MHz clock.

The CRT controller must be composed of three modules: clock generation module to generate a 25MHz control clock from 100MHz system clock; hsync module and vsync module. The clock generation module could take any input system clock to generate a 25MHz CRT control clock. However, 100MHz system is used for this project. Notice that this clock is not really a clock for flip-flops but a control signal to synchronized timing signals and counts. The vsync module will generate 480 active video line synch pulses and additional 40 line synch pulses for front, back porches and vsynch signal. The hsynch module will generate 680 active pixel synch pulses and additional 119 pixel synch pulses for front, back porches and hsynch signal.

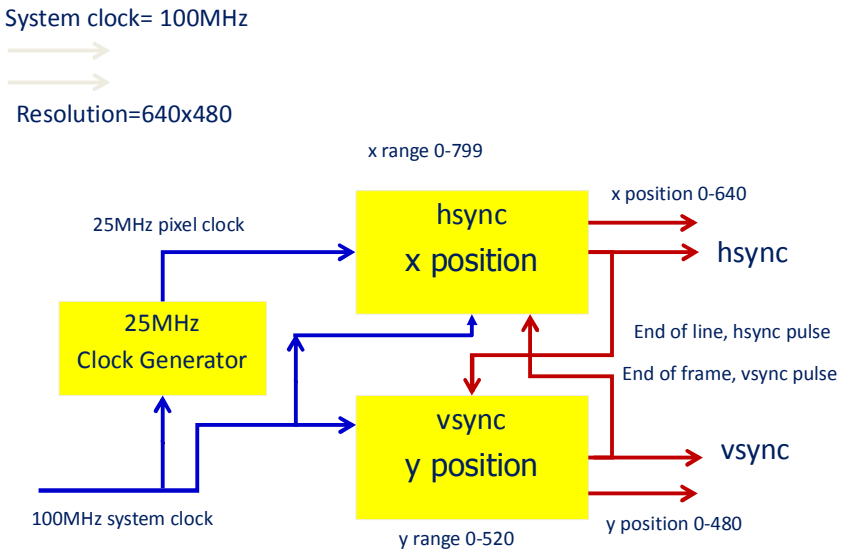
The vsynch pulse could be used to synchronize the hsync although this is not a must. hsync works without being synchronized with vsynch. End of Line from hsynch is used to increase y position.

Test your CRT controller by replacing the video_timer of the original pong game with your CRT controller. Here is an example definition for the CRT controller and its top level schematic. You will also need to change the following statements in the game module:

//wire top = (visible && ypos <= 4); //new to run at 100MHz

//wire left = (visible && xpos <= 4); //new to run at 100MHz

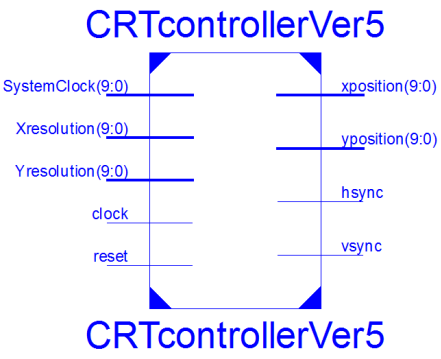
Internal Structure of the CRT Controller for 640x480 (800x520)

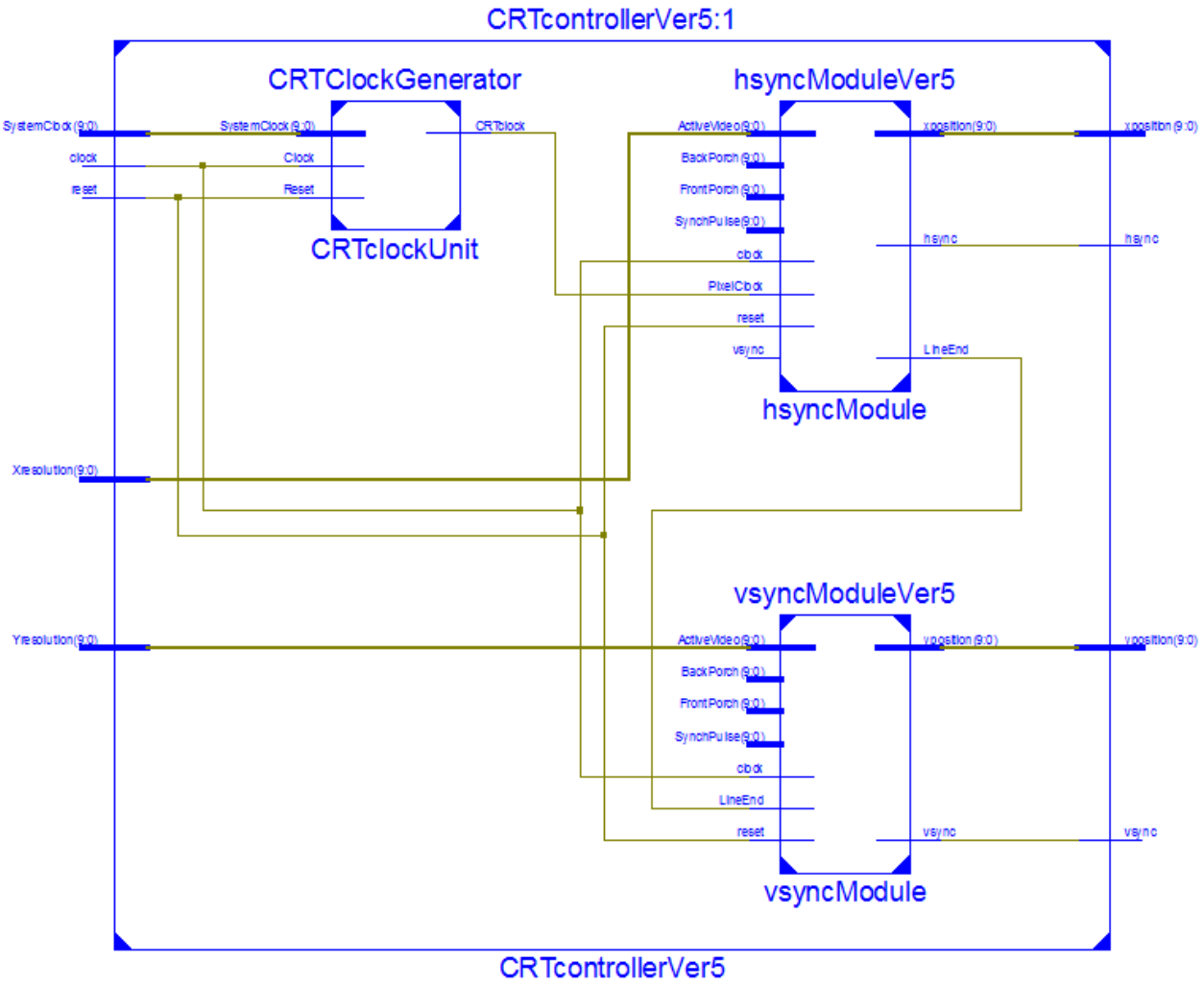


Lecture 21 VGA Driver ECE333 Fall

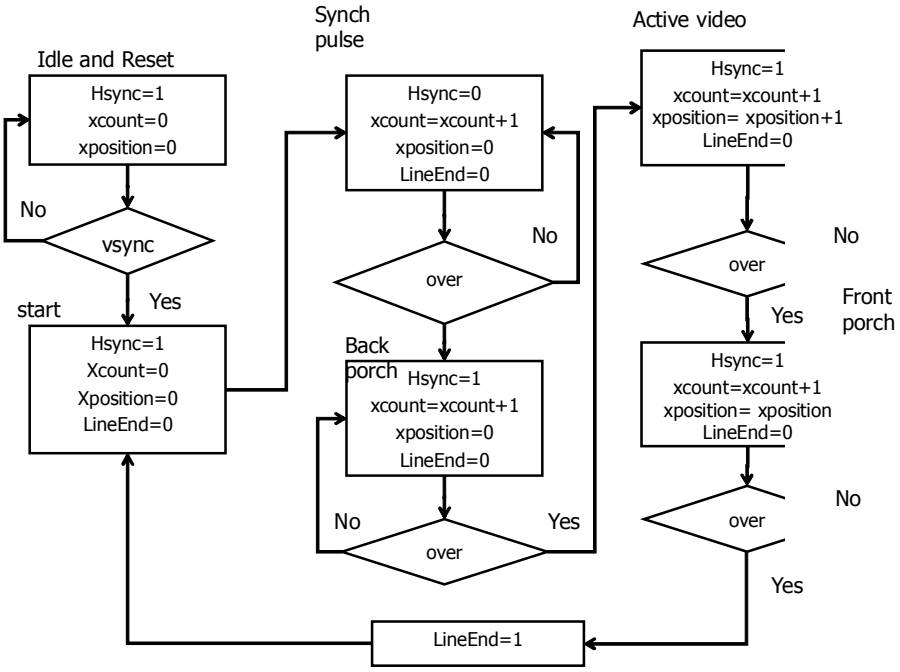
```
module CRTcontrollerVer5(Xresolution, Yresolution, SystemClock, hsync, vsync, xposition,
yposition, reset, clock);
parameter ResolutionSize=10, SystemClockSize=10;
input [ResolutionSize-1:0] Xresolution, Yresolution;
input [SystemClockSize-1:0] SystemClock;
input reset, clock;
output hsync, vsync;
output [ResolutionSize-1:0] xposition, yposition;
//hsync is generated after active video and front porch from >664 to >=759
parameter hSynchPulse=10'd95, hFrontPorch=10'd25, hBackPorch=10'd40; //hsynch=799
//vsync is generated after active video and front porch from >664 to >=759
parameter vSynchPulse=10'd2, vFrontPorch=10'd9, vBackPorch=10'd29; //vsynch=520

wire PixelClock;
//module CRTClockGenerator(SystemClock, CRTclock, Reset, Clock);
CRTClockGenerator CRTclockUnit(SystemClock, PixelClock, reset, clock);
wire LineEnd;
//module hsyncModuleVer5(vsync, PixelClock, SynchPulse, BackPorch, ActiveVideo,
FrontPorch, hsync, LineEnd, xposition, reset, clock);
hsyncModuleVer5 hsyncModule(1'b1, PixelClock, hSynchPulse, hBackPorch,  Xresolution,
hFrontPorch, hsync, LineEnd, xposition, reset, clock);
//module vsyncModuleVer4(hsynchpulse, SynchPulse, FrontPorch, ActiveVideo, BackPorch,
vsync, yposition, reset, clock);
vsyncModuleVer5 vsyncModule(LineEnd, vSynchPulse, vFrontPorch, Yresolution,
vBackPorch, vsync, yposition, reset, clock);
endmodule
```





Here is an ASM chart for the CRT controller where the synch pulses are generated at the beginning of the counts. It is not hard to change it to generate synch pulses at the end of the counts if needed.



III.3 Adding sounds to the pong game with the CRT controller (10%)

Make a folder called TermProjectPhase3. Copy all source files from TermProjectPhase2 to this folder. Change the top level module change to PongwithSound.v. Change the game module to GamewithSoundButton.v.

Add sounds to the pong game with the CRT controller by incorporating the PlaySound.v module provided by the instructor. Different sounds are activated when the paddles moves to right or left or the ball is hit or missed. You could add other sounds or music if you want to.

III.4 Add feature for pushbutton-controlled random ball release (15%)

Modify the game in TermProjectPhase3 so that the ball is not release immediately upon starting. Instead, it should wait for a pushbutton to be pressed, then release, or "serve", the ball from a random x-position in the topmost row of the playing field.

Under the default game action, when the ball is missed, the screen flashes but the ball continues to bounce off the lower wall and continue. Modify this functionality so that after a "miss", a pushbutton must be pressed to serve the ball again, just as in the start state.

III.5 Improving the pong game with your VGA driver (25%)

Make a folder called TermProjectPhase4. Copy all source files from TermProjectPhase3 to this folder. Change the top level module change to PongwithnewFeatures.v. Change the game module to NewGame.

You are required to add at least two new features to improve the pong game with your VGA driver and Nexys 3 board. Possible ideas include but are not limited to the following:

- *Making the paddle jump;*
- *Adding more paddles;*
- *Adding more balls;*
- *Making LEDs flash;*

- *Making the ball move faster or randomly,*
- *Adding melodies to the actions;*
- *Adding blocks in the middle of the screen;*
- *Adding more colors; etc.*

You are free to include any new features.

IV. Appendix: Differences between video_timer() and CRTcontroller()

There are at least three major differences between the new CRTcontroller() and video_timer() from the original pong. (1) The video_timer() uses the 25MHz clock as its system clock. (2) The original pong ties video_timer() and game() closely so that the game() also uses the 25MHz clock for all of its timing definitions The original pong would not work if clock frequency is not 50MHz.

Both the new CRTcontroller() and video_timer() generate synch signals at the end of x and y counts. Here how video_timer() produces them:

```
reg hsync, vsync;
always @(posedge clk25) begin
   hsync <= ~(xpos > 664 && xpos <= 759);  // active for 96 clocks
   vsync <= ~(ypos == 490 || ypos == 491);   // active for lines 490 and 491
end
```

video_timer() outputs x and y coordinates from 0 to the end counts. It does not generate just active pixel coordinates.

If you want to match your CRT signals with those from video_timer(), you can change your CTRcontroller() to generate hsync and vsync at the end of x and y counts. You may also just output xcount and ycount by doing the following.

assign xposition=xcount; in hsyncModule.

assign yposition=ycount; in vsyncModule.

This way, your CRT controller may work better with the game module.

You may also need to adjust timing parameters to make the brightness better.

V. References

1. Video on youtube: http://www.youtube.com/watch?v=vehgjOFRGjM.
2. The starting point pong website: http://www.bigmessowires.com/2009/06/21/fpga-pong/.
3. Pong tutorial on fpga4fun: http://www.fpga4fun.com/PongGame.html.
4. VGA information: http://martin.hinner.info/vga/.
5. Nexys 3 Board from Digilent: http://www.digilentinc.com/Products/Detail.cfm?Prod=NEXYS3.

VI. VGA Standard and Nexys 3 Implementation