BILKENT UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT
CS-223 DIGITAL DESIGN
PROJECT FINAL REPORT

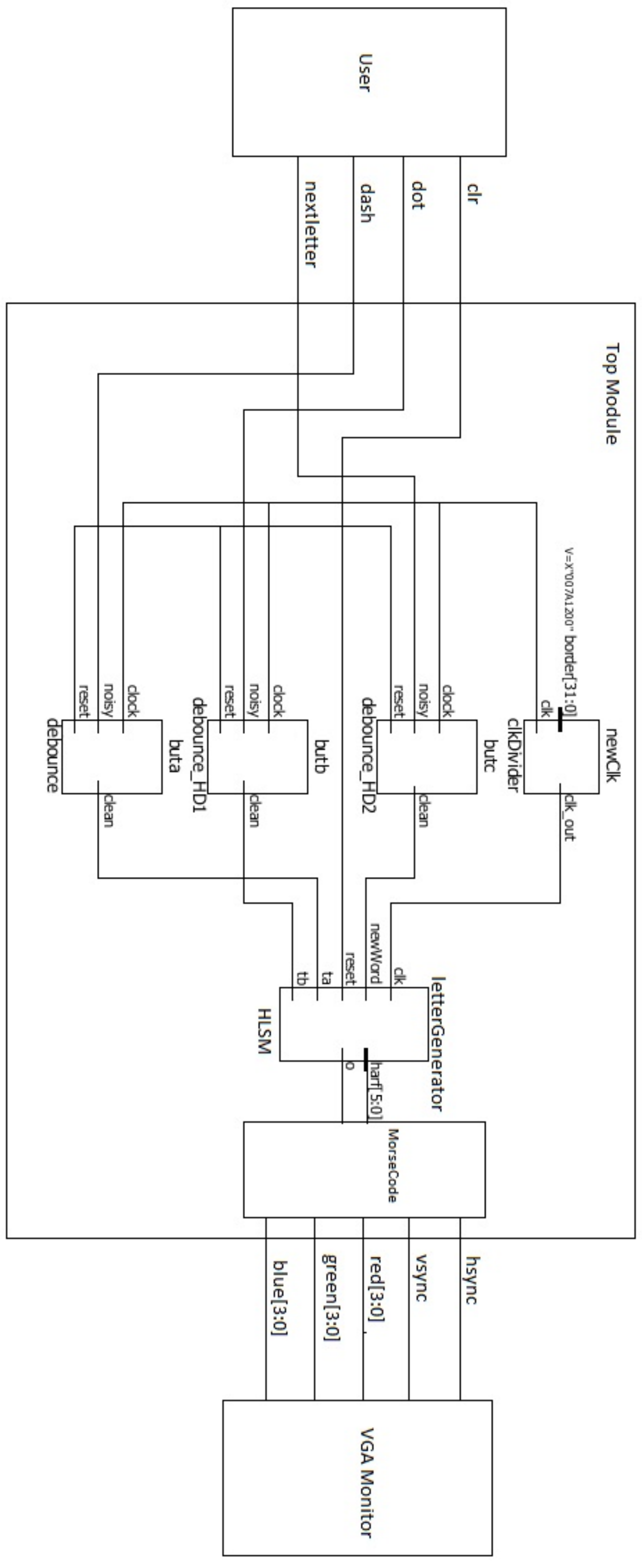
BASYSOS

Emre BAŞAR
21503907 223-2

Emre GÜRÇAY
21401645 223-2

25/12/2016

Block Diagram



Explanation of Modules

Module clkDivider

This module has 2 inputs called clock and border and one output called clk_out. We determine the border value according to our calculations. It has one counter logic inside. Every positive edge of basys3's regular clock our counter increments by 1. When it reaches our border value our clk_out becomes 1, and in the following clock cycle it becomes 0 again. We simply used this module for dividing clock values for HLSM and led outputs.

Module Debouncer

This module has 2 inputs clock and noisy and outputs clean. This module allows us to send proper button signal to our HLSM. Simply, regular button signals have noise and it sends too many signals according to our clock cycles so with this module, we decreased it to only one signal. It has its own clock divider inside. As a summary it cleans the button signal from noises and outputs a clean signal.

Source: MIT

Module LetterGenerator:

We have one HLSM. Since alphabet has 26 letters, this HLSM has 27 states (26 letters and 1 blank state). It takes 5 inputs. One of them is clock input which is divided at top module with usşng clkDivider module. The other 4 inputs are buttons called ta,tb,nextLetter. One of the button (tb) is for dash, the other button (ta) is for dot and the other button (nextLetter) is for sending signal for changing the letter index in top module for changing the index to the other letter index in the 6bit vector array by changing our other output "O" to 1(This logic will be explained in more detail in the top module section). Input clear is for resetting the current letter. HLSM has 6 bit output called Letter. As it is written before it has 26 states for each letter. It changes states according to button input. If there is no input, it stays on the same state. For example State5 is the letter “e”. When user presses ta button which means dot in morse alphabet, 6 bit letter variable becomes the binary number which represents “E” in our state encodings because in Morse alphabet “E” is represented with 1 dash. If user does not press anything, it stays on current state.

MorseCode Module

This module is our top module. It has 3 button inputs for generating letters with using letterGenerator module as stated in the previous module description. The other input is basys3 fpga’s regular clock input. It is too fast for our letterGenerator module so we divide it with using our clkDivider module. This module has 2 functions called draw and letter. Letter function takes 3 inputs, 6bit variable letter, x coordinate and y coordinate. It calls the draw function according to the inputs. For example, it takes the 6 bit input letter, finds the encoding which is same with it calls draw function for drawing the letter in the specified x and y coordinates. Draw function takes 5 inputs xStart,yStart,xEnd,yEnd and 12-bit color. It assigns a value to the top modules 4 bit outputs, red, green and blue from the 12-bit color input which is assigned to white in letter function, for printing the specified coordinates to white. The inputs for coordinates come from letter function so it prints what came from letter function. Letter function has 27 different cases for letters and a blank letter. The input buttons dot dash and next word are sending their signals to the debouncer module. This fixes the timing issue for the buttons and the outputs from these 3 debouncers sends signal to the letterGenerator when the button dash,dot or nextletter button is pressed. We have a register called sentence for having a memory for our letterGenerate outputs. We have one register called i for keeping the current index for letterGenerate output. When our letterGenerator module outputs a letter, it is written on the screen, when user presses nextLetter button letterGenerator outputs 1 and this output increases the index register by one and now letterGenerator will start to output at second index of sentence register. This module has another clockDivider for dividing regular basys3 clock for vga output. Vga works on 50Mhz so we have divided the basys3's clock to 2. This module has vga codes also. Vsync and Hsync synchronizes the Vga into 640x480 pixels.

Timing and Delays

VGA Clock: 50 MHz

Debouncer Clock: 27 MHz

LetterGenerator Module: 12.5 MHz

Data path delay 4.172 ns (logic 1472ns %35.282) route 2.700ns (64.718%))

Destination clock delay 4.856ns (14.856-10.000)

Source Clock Delay 5.156ns

References

a) We have a debouncer for our project. Which normalizes the input sent from input. When we found this code it was a Verilog code we change We used this code in our Debouncer module with a little change. We changed the value of clock cycle of this debouncer in order to use in our code properly.

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-111-introductory-digital-systems-laboratory-spring-2006/labs/debounce.v

b) We found a Verilog code at GitHub for displaying a message to VGA display. The code had two main functions in one module. When we wrote the constraints for the Verilog code it displayed letters of alphabet but it was not in the form we want. So we used the two functions of this code in our Morse Code module after translating it to

SystemVerilog. When we used the code we we changed the letter function in to the form we want. At first when we found the code it was displaying the letters instantly but we needed to display message as we got Morse inputs from the user. So we changed a lot the letters function. The other function called draw was determining the boundary cases of each letter to VGA Monitor. But as we needed different sizes for letters we needed to change draw function as well. We found this code from https://github.com/bdeloeste/Character-Table-VGA-Display

c) We did not use any other hardware device from outside Bilkent. We were planning to give sound output but as we had not have enough time we did not use any other device.

Appendices

a) We have debouncer module in our debounce.sv file.

We clkDivder module in out clkDivider.sv file.

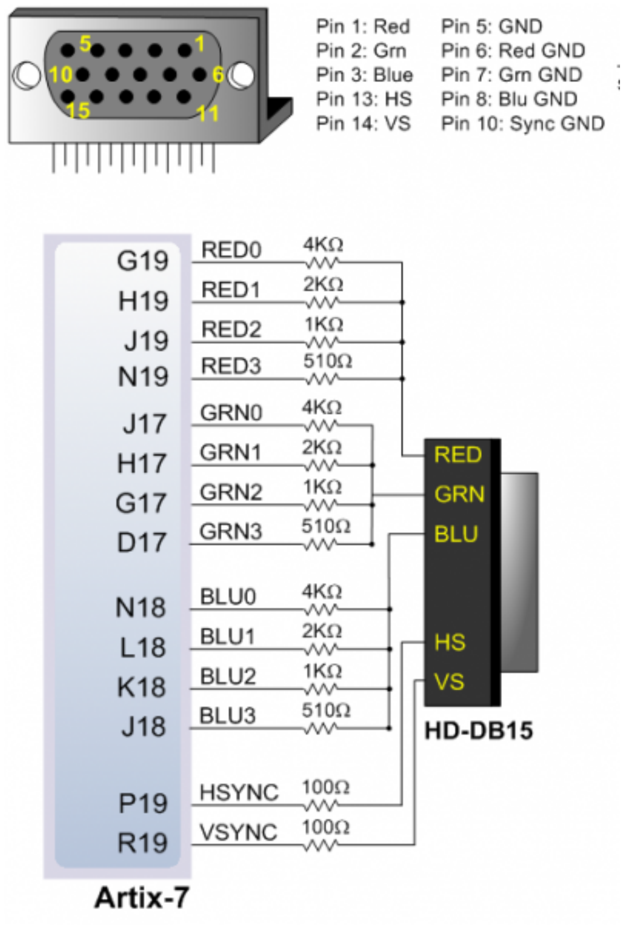
We have a HLSM called letterGenerator in our letterGenerator.sv file.

We have a top module called MorseCode in our MorseCode.sv file.

b) VGA Port Data Sheet

The Basys3 board uses 14 FPGA signals to create a VGA port with 4 bits-per-color and the two standard sync signals (HS – Horizontal Sync, and VS – Vertical Sync). The color signals use resistor-divider circuits that work in conjunction with the 75-ohm termination resistance of the VGA display to create 16 signal levels each on the red, green, and blue VGA signals.

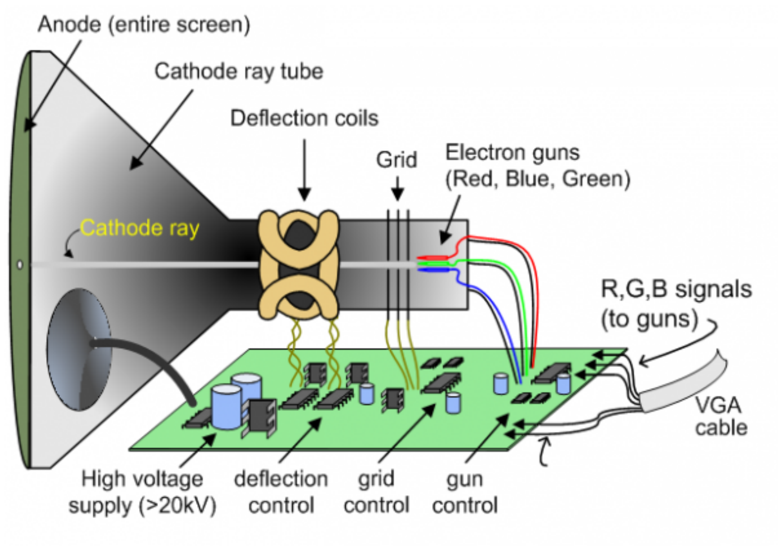
This circuit, shown in the below diagram , produces video color signals that proceed in equal increments between 0V (fully off) and 0.7V (fully on). Using this circuit, 4096 different colors can be displayed, one for each unique 12-bit pattern. A video controller circuit must be created in the FPGA to drive the sync and color signals with the correct timing in order to produce a working display system.



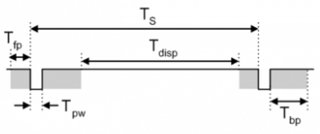
How CRT Displays Work- Theory

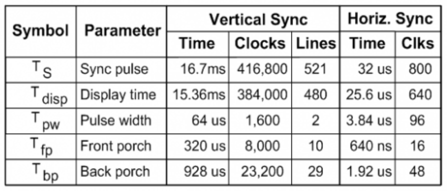
CRT-based VGA displays use amplitude-modulated moving electron beams (or cathode rays) to display information on a phosphor-coated screen. LCD displays use an array of switches that can impose a voltage across a small amount of liquid crystal, thereby changing light permittivity through the crystal on a pixel-by-pixel basis. Although the following description is limited to CRT displays, LCD displays have evolved to use the same signal timings as CRT displays (so the “signals” discussion below pertains to both CRTs and LCDs). Color

CRT displays use three electron beams (one for red, one for blue, and one for green) to energize the phosphor that coats the inner side of the display end of a cathode ray tube.



Electron beams emanate from “electron guns” which are finely-pointed heated cathodes placed in close proximity to a positively charged annular plate called a “grid.” The electrostatic force imposed by the grid pulls rays of energized electrons from the cathodes, and those rays are fed by the current that flows into the cathodes. These particle rays are initially accelerated towards the grid, but they soon fall under the influence of the much larger electrostatic force that results from the entire phosphor-coated display surface of the CRT being charged to 20kV (or more). The rays are focused to a fine beam as they pass through the center of the grids, and then they accelerate to impact on the phosphor-coated display surface. The phosphor surface glows brightly at the impact point, and it continues to glow for several hundred microseconds after the beam is removed. The larger the current fed into the cathode, the brighter the phosphor will glow. Between the grid and the display surface, the beam passes through the neck of the CRT where two coils of wire produce orthogonal electromagnetic fields. Because cathode rays are composed of charged particles (electrons), they can be deflected by these magnetic fields. Current waveforms are passed through the coils to produce magnetic fields that interact with the cathode rays and cause them to transverse the display surface in a "raster" pattern, horizontally from left





Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

A VGA controller circuit, such as the one diagramed in the below, decodes the output of a horizontal-sync counter driven by the pixel clock to generate HS signal timings. You can use this counter to locate any pixel location on a given row. Likewise, the output of a vertical-sync counter that increments with each HS pulse can be used to generate VS signal timings, and you can use this counter to locate any given row. These two continually running counters can be used to form an address into video RAM. No time relationship between the onset of the HS pulse and the onset of the VS pulse is specified, so you can arrange the counters to easily form video RAM addresses, or to minimize decoding logic for sync pulse generation.

Datasheet is obtained from: https://reference.digilentinc.com/basys3/refmanual