Engs 31 / CoSc 56

Lab 4: VGA

Due Week of April 28

Learning objectives

1. Implement a VGA controller to generate appropriate timing signals to a VGA monitor.

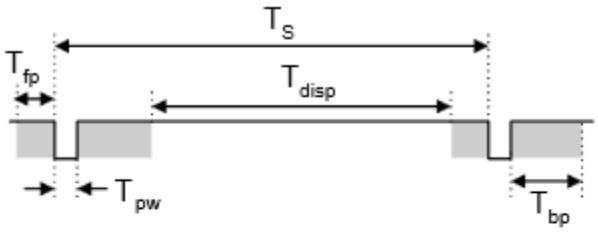
2. Display a provided test pattern on the display.

Introduction

The BASYS2 board uses 10 FPGA signals to create a VGA port with 8 bit 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 eight signal levels on the red and green VGA signals, and four on blue (the human eye is less sensitive to blue levels). This circuit produces video color signals that proceed in equal increments between 0V (fully off) and 0.7V (fully on). Using this circuit, 256 different colors can be displayed, one for each unique 8-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.

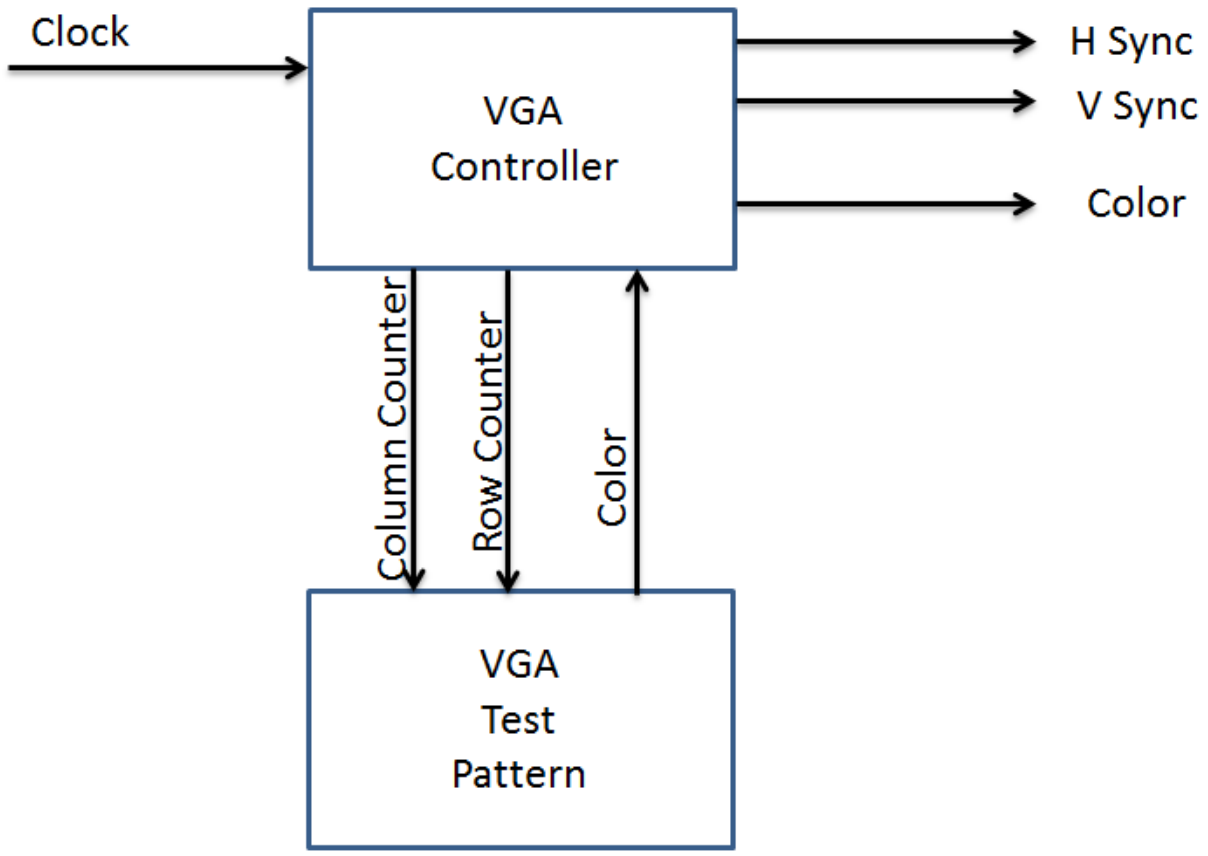
A VGA controller circuit must generate the HS and VS timings signals and coordinate the delivery of video data based on the pixel clock. The pixel clock defines the time available to display one pixel of information. The VS signal defines the "refresh" frequency of the display, or the frequency at which all information on the display is redrawn. The minimum refresh frequency is a function of the display's phosphor and electron beam intensity, with practical refresh frequencies falling in the 50Hz to 120Hz range. The number of lines to be displayed at a given refresh frequency defines the horizontal "retrace" frequency. For a 640-pixel by 480-
row display using a 25MHz pixel clock and 60 +/-1Hz refresh, the signal timings shown in the table below can be derived. Timings for sync pulse width and front and back porch intervals (porch intervals are the pre-and post-sync pulse times during which information cannot be displayed) are based on observations taken from actual VGA displays

A VGA controller circuit decodes the output of a horizontal-sync counter driven by the pixel clock to generate HS signal timings. This counter can be used 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 this counter can be used to locate any given row.



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

In this lab, you will create a "VGA controller" that will generate all of the signals needed to display a known test pattern on the VGA display. The test_pattern code (which is supplied to you) will receive the row counter and column counter and output the color code needed to display the test pattern.



BEFORE COMING TO LAB

Reading

- Review BASYS2 Reference Manual concerning VGA timing.

Part I: Prelab

1. Generate VGA controller.

Create a new project (i.e. lab4_vga_*your initials*) and create your VGA controller. Your VGA controller should output HS and VS signals matching the timing shown above as well as a row counter value and a column counter value. These values will be used to generate the colors needed to display a test pattern. Your design should include two counters. The first to count the 640 columns plus the front porch, back porch and sync pulse as you move across the row. When this counter reaches the end, it should enable the second counter to count the 480 rows plus sync pulse time (equal to three more rows). These counter values will be fed to the test pattern code for color generation. These counters also allow for creation of the VS and HS

signals for the VGA monitor. The pixel clock that you should use is 25 MHz which the supplied clock on the BASYS2 board.

2. Simulate

Using a testbench which only needs to contain a clock, simulate your VGA controller. Ensure that the sync signals meet the timing shown.

IN THE LAB

Part II: Implementation and testing

1. **When you get the simulation working, show it to one of the lab staff and obtain a signature on your cover sheet.**

2. Integrate the vga_test_pattern code into your design and re-simulate. You should have a top level to your design with two components: your vga controller and the test pattern code. The color signal gets mapped as follows:

   Color (7 downto 5) => red (2 downto 0)
   Color (4 downto 2) => green(2 downto 0)
   Color (1 downto 0) => blue(2 downto 1)

   Once you get the simulation working, show it to one of the lab staff and obtain a signature on your cover sheet.

3. Synthesize you design and configure the FPGA on the BASYS2 board. Be sure to modify (un-comment) your .ucf file to include the interface to VGA Monitor.

4. Connect the VGA monitor cable to the BASYS2 board and display the test pattern on the monitor. Be sure that the colors are correct. **Show one of the lab staff and obtain a signature on your cover sheet.**

AFTER THE LAB

Submit the following documents, with your Cover Sheet on top:

(1) TA cover sheet, attached as a cover page to your report.

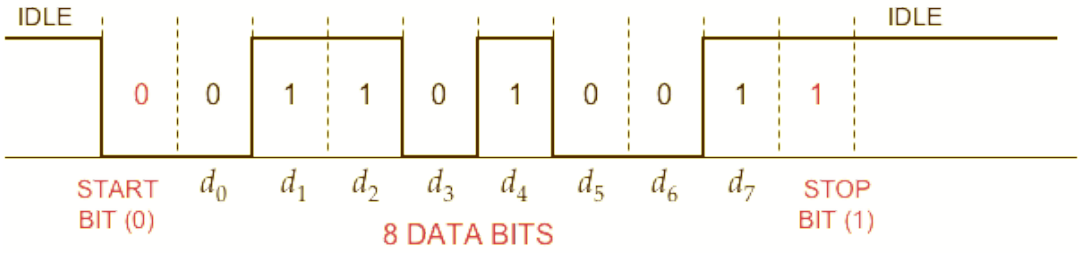
(2) VHDL source for your VGA controller.

(3) A printout of you simulation of the VGA controller.

Discuss the obstacles you encountered, how you overcame them, and what you learned in the process. Make sure that each page is clearly labeled with your name and a descriptive caption or title. Staple these pages together.

In the RS-232 asynchronous serial communications protocol, an 8-bit byte (typically an ASCII-encoded character) is transmitted as shown in the picture below. When nothing is being sent (the line is quiescent), the signal is at a logical "1" level. The byte is preceded by a "0", called the *start bit*, and concluded by a "1", called the *stop bit*. A parity bit may optionally be inserted between the most significant data bit and the stop bit, and there may be an additional stop bit. But this packet format, "8 bit, no parity, 1 stop bit", is common and the one we will use in this lab.

The data packets are transmitted without an accompanying clock (hence, "asynchronous"), so the synchronization must be recovered in the receiver. It is assumed that the receiver knows the rate at which the transmitter is sending bits. The reciprocal of the width of a single bit is called the baud rate. In Lab 5 we will use 115,200. The receiver is supplied with a second clock, called the baud rate generator(brg), that runs at 16 times the baud rate, i.e., there are 16 brg pulses within each bit time.



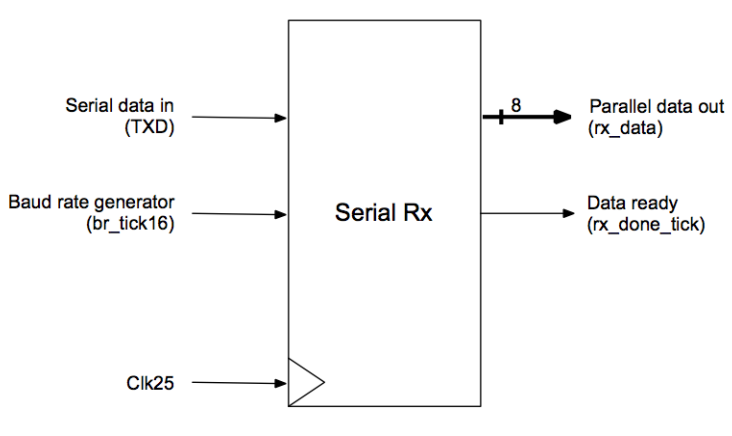
Baud rate generator

Design a counter in VHDL that counts from 0 to M–1 (or from M–1 down to 0) and repeats, emitting a pulse for one master clock (Clk50) cycle every time the count is M–1 (for an up counter) or 0 (for a down counter). Choose M so that the frequency of the pulse train is 16 ×115,200=1,843,200 Hz when the master clock frequency is 25 MHz. (For example, if M = 2 then the frequency will be 12.5 MHz.) Design this counter as its own entity and architecture in its own file, e.g., brg.vhd. Make BAUDRATE a constant and calculate M from it, similar to the way MAXCLKDIV was calculated from NCLKCDIV. This will make it easier to change the baud rate later.

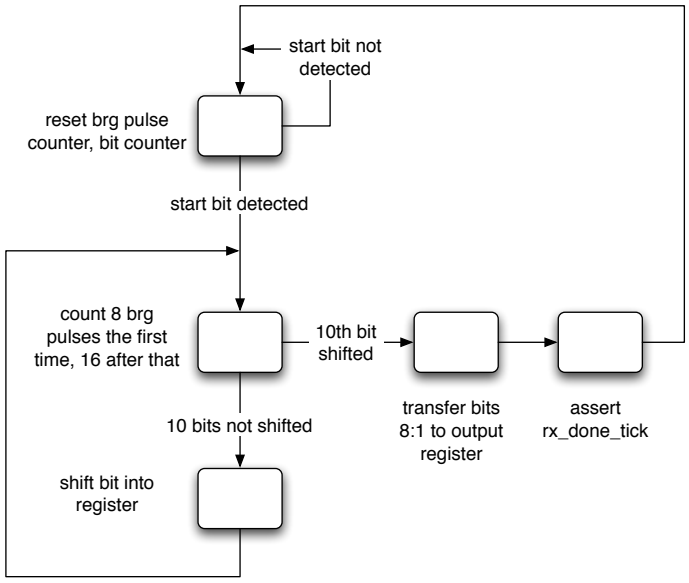
Shift register and data register

Design a 10-bit shift register in VHDL. It should have a CLEAR input for synchronous resetting, and a SHIFT input that enables the register to shift right on a clock edge. The serial input is through bit 9, the msb. It will be clocked with the 25 MHz master clock. In lab, the data byte will be parallel-loaded from bits 8...1 of the shift register into an 8-bit output register. With this register the receiver output will remain stable while the next byte is being received into the shift register. In practice this output register might be the input to a FIFO.

The reciprocal of the temporal width of a single bit is called the *baud rate*. Common baud rates are 9600 and 19,200. The transmitter is simply a shift register that takes in the 8-bit data, adds a start bit and stop bit, and shifts the result out serially at the baud rate. The data packets are transmitted without an accompanying clock (hence, "asynchronous"), so the clock must be recovered in the receiver.



We may surmise that, at least, the receiver consists of a 10-bit shift register and a state machine to control the shift register. The baud rate generator is shown as external to the receiver block, because it is also used by the transmitter block. The operation of the receiver is described by the following high-level state machine.



- Watch the serial input line, waiting for it to drop from 1 to 0, signifying the beginning of a packet.
- Start counting pulses from the baud rate generator (br_tick16). After eight pulses, the center of the start bit has been identified. Shift this value (expected to be 0) into the shift register.
- Next, count sixteen baud rate generator pulses, which locates the center of the first data bit. Shift this value into the register. Repeat this step until the stop bit is shifted into the register.
- The middle eight bits of the register (8 down to 1) contain the data byte. Transfer this byte into an output register. This second register ensures that the output doesn't fluctuate while the next packet is being received.

- Assert the rx_done_tick pulse for one master clock cycle to signal that the byte is available, then return to the initial state and wait for the next start bit.

The key components of the receiver's datapath are the baud rate generator (brg) the shift register, and a double-flop synchronizer (Vahid, pp 147-149).

You will also need, for the controller:

• a counter to count 8 brg pulses before the first shift and 16 brg pulses between shifts (this can be the same counter if you work it right)
• a counter to count the number of bits that have been shifted into the register

Design the controller FSM in VHDL to include the serial data stream and the TIMEOUT signals from the two counters as inputs. For outputs, include control signals for the shift register, the data register, and the two counters, and the data ready signal rx_done_tick.

For debugging in the lab, plan to bring the input data stream, the brg pulse stream, the shift register's SHIFT signal, and rx_done_tick out to pins on the FPGA board. These signals were invaluable in helping me debug my design.

In the lab, the input data stream will come from a PC running PuTTy, and the 8 output bits from the receiver will be sent two places: to the multiplexed seven-segment display you built this week, and also to the serial transmitter (I will supply) for looping back up the RS-232 cable to the PC. When your design is complete, you will see the ASCII code for characters you type appear on the seven-segment display, and you will also see the characters echo back to PuTTy.

Lab 4: VGA

Cover Sheet

Instructions: Bring this sheet with you to the lab. Have a TA or instructor initial your progress as you complete the in-lab assignments. Attach this sheet to the **front** of your lab report.

Your name: _____

Lab partner's name: _____Section_____

Assignment	**TA signature**	**Date**
Simulation of VGA Controller	_____	_____
Simulation VGA Controller with Test Pattern	_____	_____
Test Pattern Displayed on VGA Monitor	_____	_____