VGA Graphics in a VHDL/FPGA Digital Design Course

Darrin M. Hanna and Richard E. Haskell
Department of Electrical and Computer Engineering, Oakland University, Rochester, MI 48309
dmhanna@oakland.edu, haskell@oakland.edu

Abstract - **A junior-level course in digital design using VHDL (VHSIC Hardware Description Language – VHSIC: Very High-Speed Integrated Circuit) has been taught at Oakland University for many years. In this course, students design digital systems using VHDL, and implement the designs on a Xilinx FPGA. In such a course, it is important to have students work on designs that require the capabilities of an FPGA, and can not be implemented easier using a simple microprocessor. To this end, we have found that if students design some type of video game or other video-based device, they must not only design a VGA controller, which operates beyond the speed capabilities of a microprocessor, but they also learn how to implement algorithms directly in hardware. This type of digital design typically uses many of the basic digital building blocks that they learn to design throughout the course. It also requires students to work with timing issues, synchronizing parallel processes using multiple memories, create test benches, and other common embedded issues for a system that is fairly difficult to simulate. The visual graphics that are created as part of the design provides a strong motivation to complete the project.**

Index Terms – Digital Design, FPGA, VHDL, VGA Graphics.

INTRODUCTION

Traditional hardware design courses focus on topics spanning Boolean algebra, the design of basic combinational and sequential elements, and sometimes the design of significant special purpose processors or a microcontroller. In these three- or four-credit courses, there is little time to dwell on any one of these topics, plus most courses strive to have their students create significant hardware in the lab or for an end-of-course project. In a junior-level course at Oakland University, where students learn hardware design using VHDL, we have found that teaching students to design, implement, and expand upon a VGA controller incorporates most hardware design concepts into a single topic, which saves time and provides a continuous example for students to follow over multiple lectures. This topic alone results in a significant boost for students to transition from basic hardware design skills to more advanced skills, which enables them to create impressive, elaborate projects during the last two to three weeks of the course.

MANY HARDWARE DESIGN CONCEPTS: A VGA CONTROLLER

Figure 1 shows a block diagram of the initial VGA controller taught in this course. This includes a module, *VGA_stripes*, that displays stripes of colors on the screen on a Digilent Nexys2 Board supporting 8-bit color (3-bit red, 3-bit green, and 2-bit blue). The VGA controller involves creating horizontal and vertical sync signals using a 25 MHz clock to achieve a 640 x 480 resolution. This requires calculating delays to match those in a standard VGA timing diagram and designing horizontal and vertical counters used to generate the correct timing for both sync signals. Additionally, students must generate a *vidon* signal, which is active high during the visible display region and active low during the retrace and back and front porch regions.

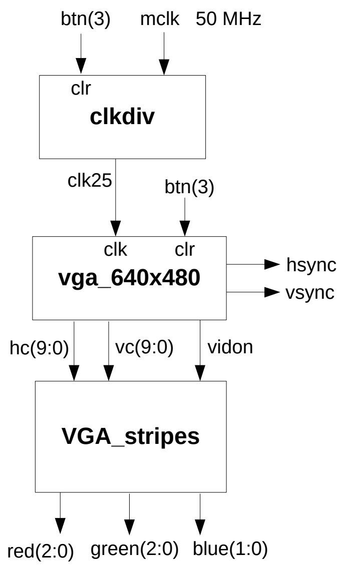


FIGURE 1: VGA CONTROLLER

After achieving stripes on the screen, students learn how to create a bitmap of their initials in a read-only memory and implement a module that uses that bitmap to place their initials on the screen. The switches on the board are used to control the position of their initials on the screen. The next lecture introduces random access memory to be used as video memory. Students are introduced to CoreGen for generating the files necessary to implement a block memory including a *.coe* file for loading a picture into the block RAM as initial data. Of course, this requires creating a *.coe* file from a standard JPEG image. To accomplish this, students use Matlab to parse the image file using built-in Matlab image functions, convert the 8-bit RGB image data to 3-3-2-bit RGB, and create a properly formatted *.coe* file. This also requires investigating the space available in the on-chip block memory for the target FPGA and calculating the picture size that will fit in the available memory.

As a follow on exercise, students add a *bounce* module, shown in Figure 2, which controls the location of the picture on the screen by developing a state machine that turns this project into a screen saver. The starting row and column are incremented or decremented by a constant amount, ΔR and ΔC, until the picture touches one of the viewing area extremes, after which the sign of ΔR or ΔC is changed. The pictures bounce around the screen as shown in Figure 3.

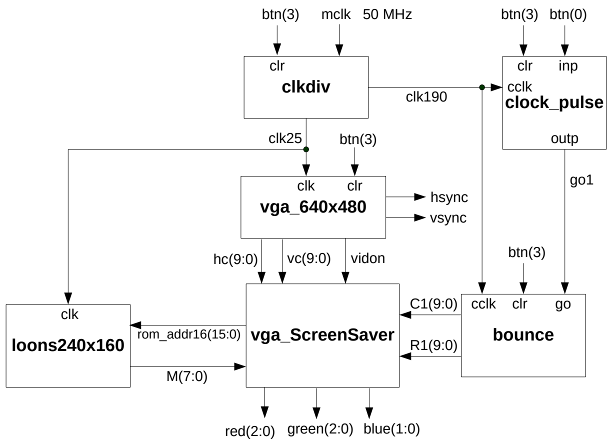


FIGURE 2: A VGA SCREEN SAVER USING AN IMAGE IN BLOCK RAM

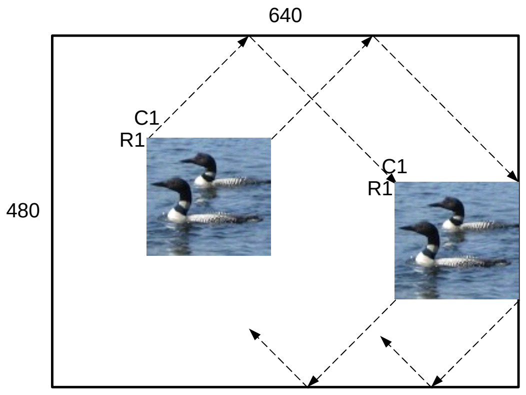


FIGURE 3: MOVING IMAGES LIKE A SCREEN SAVER

The next lecture introduces expanding video memory for storing pixels for an entire screen. Since this will not fit into the block RAM on the Spartan 3E FPGA on the board, we introduce the on-board external cellular memory and flash memory. Video memory introduces interfacing with external memory. This requires several considerations including creating a clock that meets memory timing requirements, implementing a memory controller for reading from memory, and reading and processing two pixels at a time since the memory bus is 16 bits wide. Using these modifications, a 640x480 resolution picture can be displayed on the screen from video memory. The Digilent memory utility is used to download the image to the external on-board memory.

After displaying a 640x480 resolution image, students are challenged to create a state machine that will clear the screen, and another state machine that will plot a dot at a given *x,y* coordinate. This requires creating a special purpose processor for each function, expanding the memory controller to include writing to memory, and considerations for either writing upper and lower bytes separately to the 16-bit memory, or reading the data word in order to change the appropriate byte before writing the new data to memory. These state machines must share the external video memory bus with the VGA controller by writing data on the falling edge of the clock while the controller reads data on the rising edge. Additionally, each special purpose processor must share the memory bus with each other which requires multiplexing. Figure 4 shows a block diagram of this expanded VGA controller suitable for plotting a dot at a location on the screen determined by the switch settings.

From here, students can create special purpose processors to perform other functions such as plotting a line, plotting a circle, plotting a star, and other video-related tasks. This VGA system is somewhat difficult to simulate, given the volume of signal changes required for a single video page. Furthermore, the video memory at this point is stored in the external cellular memory for which there are no simulation models. Therefore, students are exposed to creating test benches that simulates the external cellular memory, port maps in the VGA top level as a unit under test, uses VHDL constructs that are reserved for simulation such as *wait until*, *wait for*, *assert*, *shared variables*, and *file I/O*. The test bench presented simulates the video memory and writes the contents of the memory to a file formatted to present data similar to a VGA monitor; 640 values per line, 480 lines.

All of the VGA material used in this course is covered in a textbook [1] that is available to all students. A Verilog version of this textbook is also available [2]. These books assume prior knowledge of VHDL or Verilog. For those with no prior knowledge of VHDL or Verilog, Digilent is now bundling books [3] [4] with both the BASYS2 and Nexys2 boards, which provide over 75 examples of VHDL or Verilog programs, including VGA examples that can run on either the BASYS2 or Nexys2 boards. These books do not include the examples covered in [1] and [2] of using the external RAM and flash memory, which are available only on the Nexys2 board.

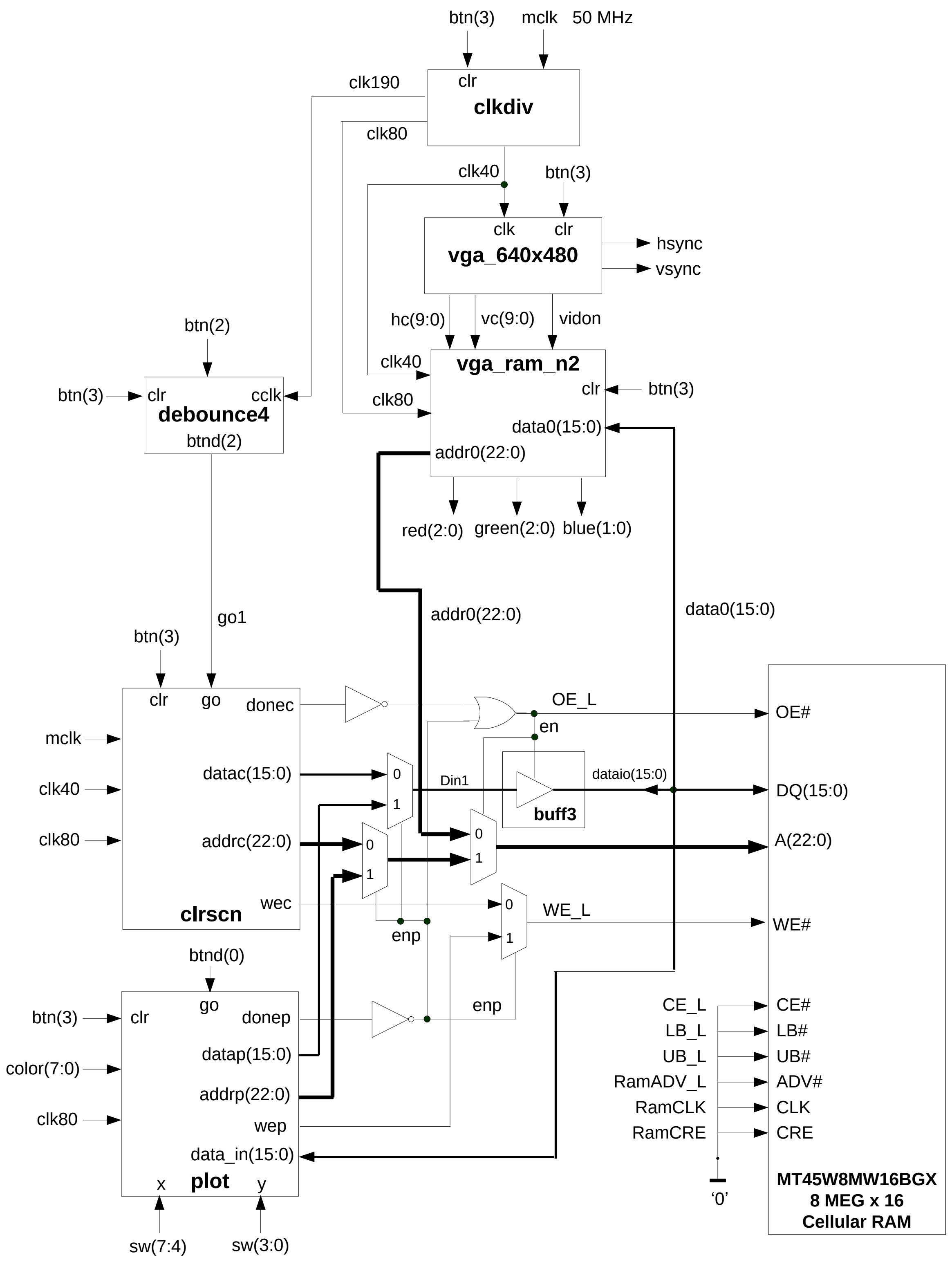
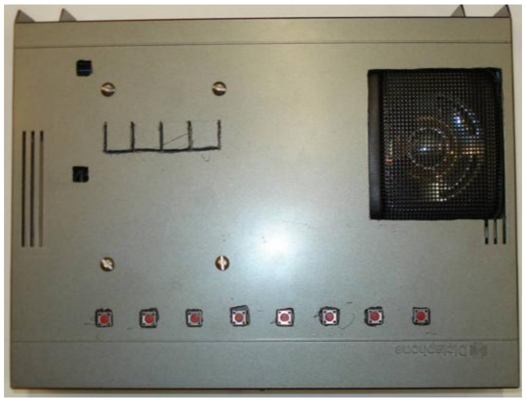


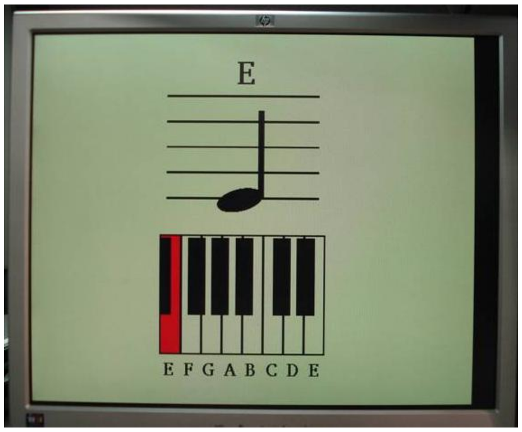
FIGURE 4: A VGA CONTROLLER FOR PLOTTING DOTS USING AN EXTERNAL VIDEO RAM

THREE-WEEK STUDENT DESIGN PROJECTS

After engaging students in these more advanced hardware design topics, students are prepared to create end-of-course projects in groups of three or four that are quite impressive and elaborate given the short period of time in a "first course" in hardware design. Many students choose to create games that use the VGA monitor interactively with video game controllers ranging from home-made controllers to Nintendo joysticks that they wire into the board, and create a controller based on the joysticks timing diagram.

The figures in this section demonstrate some of the typical projects that students have done in this course over the past few years. The Digital Music Player box shown in Figure 5a contains eight pushbuttons corresponding to the eight notes of a musical scale. The project was designed to teach children the musical scale. When a particular button is pressed, the note is sounded though the speaker in the box and the VGA screen displays the proper key and note as shown in Figure 5b.


(a)



(b)

FIGURE 5: DIGITAL MUSIC PLAYER

Figure 6 shows the VGA screen at the beginning of the Hangman game. When the user starts the game, blanks are displayed on the screen for each letter in the mystery word. As correct letters are guessed by pressing a key on a keyboard, connected to the FPGA board through the PS/2 port, they replace the blanks in the mystery word. An incorrect guess will add another part to the body being hanged. The game will end when the correct word is guessed, or when six incorrect guesses complete the hanged body. This project is typical of well-known video games that students have implemented as projects in this class. Other games that students have implemented include Frogger, Jeopardy, The Price is Right, Space Invaders, and Pong, as well as card games such as War and Blackjack.

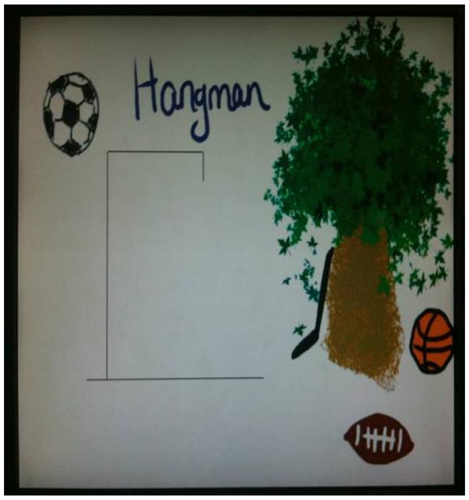


FIGURE 6: HANGMAN

One student group developed a take-off of the popular Wii game of bowling as shown by their screen shot shown in Figure 7. They used an accelerometer that they would swing with their arm to roll the ball down the lane and knock down a certain number of pins, depending on where the ball went.

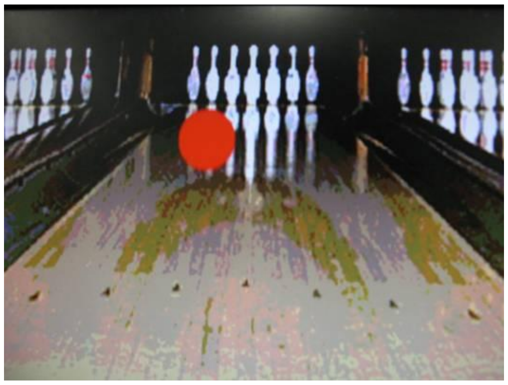


FIGURE 7: BOWLING USING AN ACCELEROMETER

The graphics capability of plotting dots and line using the video RAM was exploited by one group of students that designed a DigiSketcher as shown in Figure 8. Using a mouse that they interfaced through the PS/2 port on the Nexys2 board, the user could sketch lines on the screen.

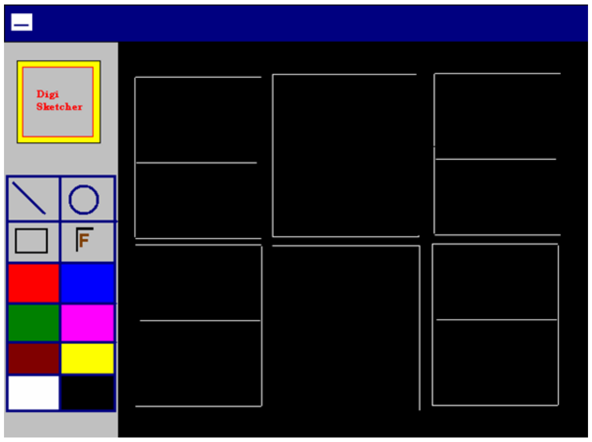


FIGURE 8: DIGISKETCHER

A particularly well-done project that would seem to have commercial potential is the Force-activated Portable Graphic Assistant (FPGA) shown in Figure 9. One of the students in the group played the violin and another played the piano. The problem of having to turn the pages of sheet music while trying to continue to play is solved with this design in which all pages of the sheet music are stored in the 16 Mbytes of flash memory on the Nexys2 board. A page of music is displayed on a standard LCD monitor. A foot pedal is used to flip from one page to the next. The buttons and 7-segment display on the Nexys2 board are used to program the page sequence. The user can easily jump back to earlier pages.

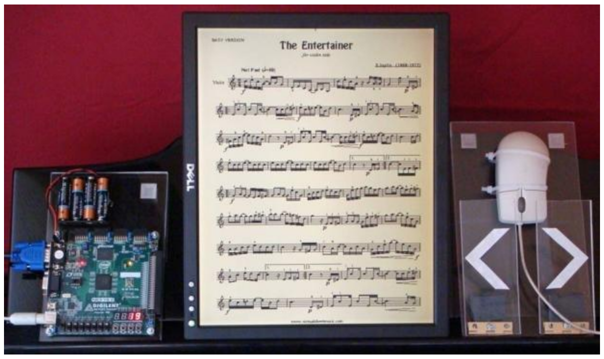


FIGURE 9:
FORCE-ACTIVATED PORTABLE GRAPHIC ASSISTANT (FPGA)

CONCLUSION

Introducing a VGA controller brings the potential to provide experience to students in the most advanced embedded systems design topics in a junior-level course. The controller results in visual accomplishments that are fun and appealing and involves matching timing with external devices, memory controllers, RAM and ROM, special purpose processors, bus multiplexing, using both clock edges, creating test benches, using block memory, considering memory requirements and space available, and converting 8-bit operations to 16-bit busses and vice versa. Many hardware design topics can be introduced to students using a VGA controller in a short period of time. Projects created by students applying this knowledge are sophisticated, impressive, and visual, and most students have a very positive experience. A number of students who have completed this course go on immediately to take graduate courses in digital design as undergraduates. Some have engaged in undergraduate research projects using FPGAs and are more prepared and capable than the graduate students who completed their undergraduate programs at other universities.

REFERENCES

[1] Haskell, R. E., Hanna, D. M, *Learning By Example Using VHDL – Advanced Digital Design With a NEXYS 2 FPGA Board,* LBE Books, Rochester Hills, MI, ISBN 978-0-9801337-4-5, 2008.

[2] Haskell, R. E., Hanna, D. M, *Learning By Example Using Verilog – Advanced Digital Design With a NEXYS 2 FPGA Board,* LBE Books, Rochester Hills, MI, ISBN 978-0-9801337-5-2, 2009.

[3] Haskell, R. E., Hanna, D. M, *Digital Design Using Digilent FPGA Boards – VHDL / Active-HDL Edition,* LBE Books, Rochester Hills, MI, ISBN 978-0-9801337-8-3, 2009.

[4] Haskell, R. E., Hanna, D. M, *Digital Design Using Digilent FPGA Boards – Verilog / Active-HDL Edition,* LBE Books, Rochester Hills, MI, ISBN 978-0-9801337-7-6, 2009.