[[1]](#footnote-1)

Reflex Challenge Game

(Spring 2013)

Nicholas A. Repetti

CPE 410, UNLV

*Abstract*—This game uses VHDL and the DE2 board so it can be played on a VGA Monitor or TV. The parts of the DE2 used will be a push button, four 7-segment displays, and the VGA port.

An object will be moving back and forth across the screen, initially at a slower speed. There are three different "target" zones, that are red, green, and blue in color. Each zone has a respective point value:

Red - 5 pts.

Green - 3 pts.

Blue -- 1 pt.

The object of the game is to stop the object by pressing the push button on the DE2. Whatever zone the object is in when the player presses the button, will earn the player points. For every 10 points the player earns, the object will speed up and the game will become more challenging.

The player's score will be displayed onto four of the 7-segment displays on the DE2 board.

# INTRODUCTION

T

HIS DOCUMENT is a report on my Reflex Challenge Game. The main idea behind the game is to use the VGA in the DE2 board. VGA stands for Video Graphics Array and is a piece of display hardware that is used for displaying graphics to a monitor.

VGA supports a resolution of 640x480 that is actually visible to the screen. This means that there are 640 pixels per line, or row, and 480 lines displayed to the screen. The VGA industry standard operates at a pixel frequency of 25.175MHz, and a screen refresh rate of 60Hz. The operation of the VGA is all based on proper timing.

Because of these timing constraints, the VGA display must be configured properly. The actual resolution is larger than 640x480, however there are certain timing issues, that only allow the display to be active for 640x480. These timing issues are caused by the retrace, and left and right border (called the front and back porch). The video signal will turn

on for 800 pixels, and then it will turn off, and then repeat for the next line. It takes time for the video signal to turn on an off, which happens in the front and back porch, and retrace. Due to these on and off times, only 640 of the 800 pixels are actually visible. The image in figure 1 displays the timing constraints of the display.

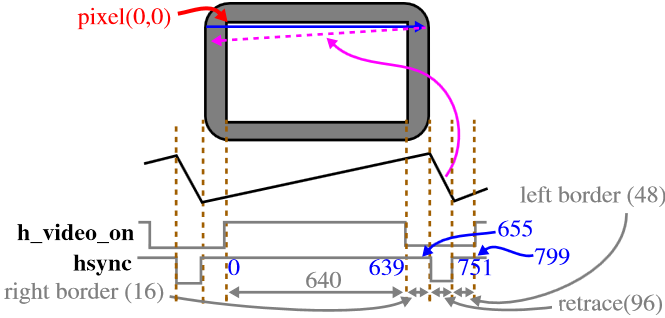


Figure. 1. Timing constraints for the VGA. The display is only on the screen when it is not turning on or off (When it is not in the retrace, front, or back porch). [1]

# Displaying objects to the screen

In order to actually display an object to the screen, we use two counters. One to count the number of pixels per line, and one to keep track of what line we are currently on. We set the counters to operate at 25MHz by splitting the DE2’s internal 50MHz clock. By doing so, the counters will be synced up with the display.

We then use the counters to figure out which location of the screen the counter is currently at, and then draw a RGB pixel at that location.

In order to make the object move, we used a variable as the constraints for the location of the object. Then, using a counter to divide the speed of the clock, we can change the location of the object by incrementing or decrementing by a certain number of pixels, based on the frequency we chose (This is the speed of the object).

# Keeping score

During the game, when the player presses the button, a score is given and then displayed onto the board’s 7-segment displays. To keep track of the score, the location of the object’s center is stored in a variable. When the button is pressed, the variable is checked to see which zone the object is currently in. Then a point value is assigned based on the zone. The score is kept track of in a signal. Every 10 points the player gets, the object will speed up.

To send the score to the 7-segment displays, a decoder function is necessary. I wrote my own, to convert an integer value, from 0 to 9, to a std\_logic\_vector that is sent to the displays.

# OUTCome

The game works pretty well, but it still has a couple of bugs in it. When the object reaches the left or right boarder, it will then travel in the reverse direction. At a certain point, the speed of the object becomes too fast. The object reaches the left wall, but since it is moving so fast the boundary doesn’t register in time, and the object just ignores it, instead of changing direction. Images of the game in action and the working DE2 board, are shown in figures 2 and 3.



# conclusion

Overall the game turned out pretty well. The only real issues I had were in Synthesis with the timing constraints. I found it difficult to move the object. Sometimes it would move to fast that it wouldn’t even show up on the screen, and sometimes it wouldn’t move at all.

I was initially using the WAIT FOR statement, but then realized that statement is ignored during Synthesis. Once I switched to a counter based on the clock signal, it worked fine.

Some improvements to the game I would make would be to fix the boundaries when the ball gets too fast, and to actually display the score to the screen, because it is hard to watch the game on the monitor and look down to the board to see the score.

I enjoyed the entire process of creating the game, and am happy with the result. It really tested my knowledge of VHDL and design processes in general.

References

1. UNM - http://www.ece.unm.edu/~jimp/vhdl\_fpgas/slides/VGA.pdf

Figure. 2. Demonstration of the game being played on a television.

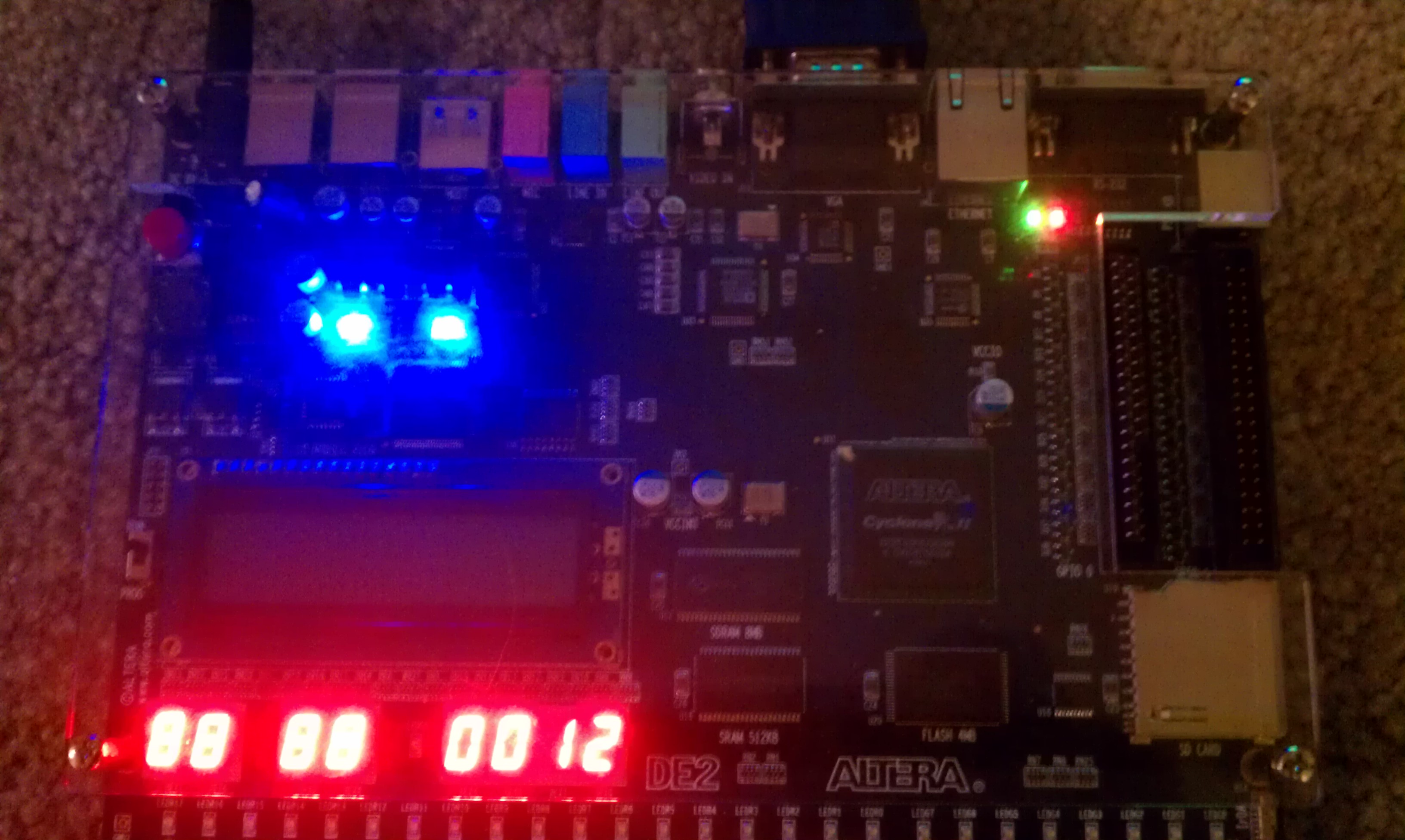


Figure. 3. Demonstration of the score being displayed on the DE2.

1. [↑](#footnote-ref-1)