Final Practical Work Electronics III

Müller, Malena
Diaz, Ian Cruz
Oh, Victor
Lin, Benjamin

November 22, 2018

1 Summary

In this work we were asked to implement a precise chronometer, digital electronics, and a VGA screen of 640 pixels wide and 480 pixels in height. We were provided with a Cyclone IV Field Programmable Gate Array, also known as FPGA, and we had to create the inside digital logic using Verilog Hardware Descriptive Language. The implementation of this chronometer is explained in detail in the following sections.

2 How VGA Works

The functioning of the VGA protocol is very simple, as shown in Figure 1, we only have to analyze 5 of its pins. Whatever the V_SYNC pin receives, corresponds to the vertical synchronization of the screen, while the H_SYNC pin receives what corresponds to its horizontal synchronization. The R, G and B wires correspond to the Red, Geen and Blue colors of the pixel to print. So, if we call T_H to the h_sync period and T_V to the v_sync period, to reference a single pixel as line-column, the formulas are the following:

$$Line = \frac{T_V}{480} * (time\, h_{sync}\, is\, on) \tag{1}$$

$$Column = \frac{T_H}{640} * (time\, v_{sync}\, is\, on) \tag{2}$$

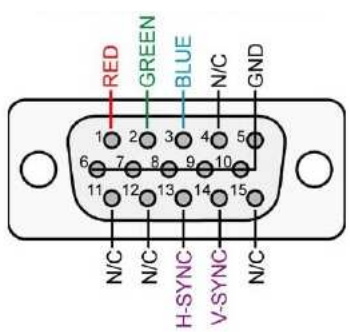

Figure 1: VGA Pins

3 Implementation

To implement the chronometer, we follow the diagram shown on Figure 2, we used a h_sync and v_sync generator module, a Screen Positioning Module, a Time Counter Module, a Timer Module and a binary to a 7 segments display module.

Figure 2: Implementation Diagram

3.1 VGA Module

This module corresponds to the Video Graphics Array (VGA), which is a graphics standard for video display. There are five pins in the VGA connector, that we have to control: the V_SYNC, H_SYNC, R, G and B, shown in Figure 1. We needed to send five signals, one to each of these five different monitor's pins. This signals' generator module was made in Verilog. It outputs the following signals:

-Vsync: The vertical synchronization's signal.

-Hsync: The horizontal synchronization's signal.

-R: The signal corresponding to the red component of a pixel.

-G: The signal corresponding to the green component of a pixel.

-B: The signal corresponding to the blue component of a pixel.

We used a resolution of 640x480. The Vsync signal is responsible of updating the entire display, in each of its periods. In order to do so, the Hsync signal updates one horizontal line of the display in each of the Hsync signal's period. The R, G and B signals update the color of one pixel.
The vertical sync is formed by four parts, in the following order: vertical sync pulse, vertical back porch, vertical display and vertical vertical front porch. As well as the vertical sync signal, the horizontal sync is formed by a horizontal sync pulse, horizontal back porch, horizontal display and horizontal front porch. These signals are 1 only during their sync pulses. For the rest of the parts that form these signals, their value is 0. Moreover, the display colors are only shown during the vertical display time and horizontal display time of these signals. This means that while both the Vsync and Hsync signals are in their display part of the period, the R, G and B signals output the values corresponding to the pixel that has to be shown in that position of the screen. For the rest of the time, the R, G and B signals are 0.
Four Verilog modules were made to generate these signals. The Vsync module, Hsync module, RGB module, and finally the VGA module which includes the previously mentioned three modules.

3.1.1 Vsync Module

This module generates and it outputs the entire Vsync signal, and it also outputs a vertical display sigal which is in 0 only when colors can appear in the corresponding vertical position.

3.1.2 Hsync Module

This module generates and it outputs the entire Hsync signal, and it also outputs a horizontal display signal which is in 0 when colors can appear in the corresponding horizontal position.
The vertical display and the horizontal display signals are not sent to the VGA pins, but they are used by the Screen Positioning Module, in order to know when to update the pixels' colors.

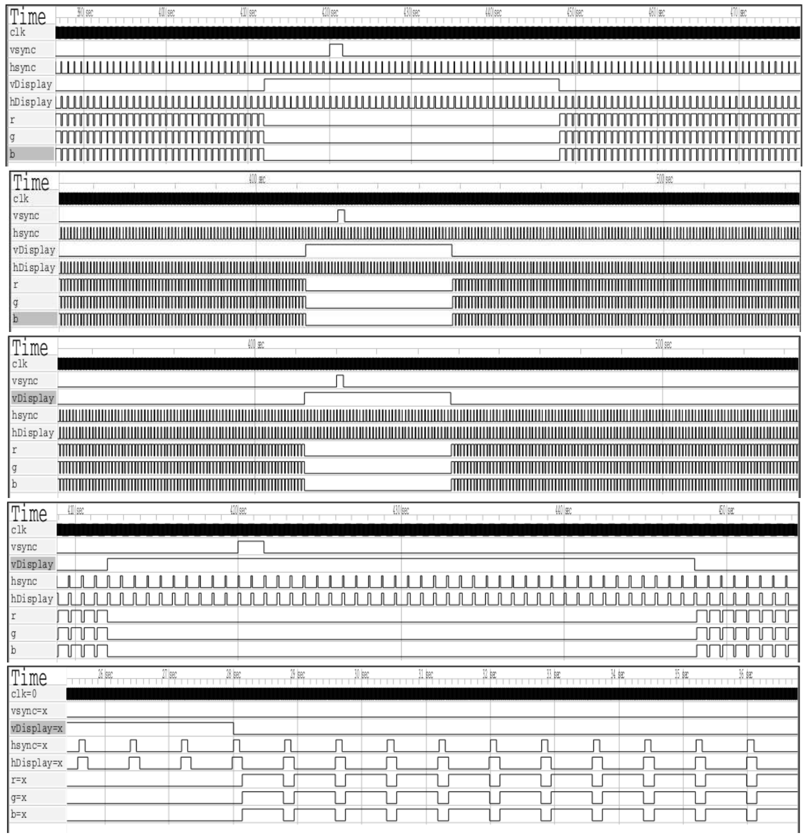

Figure 3: GTK Wave SImulations

In Figure 3 we can see the functioning of the previously mentioned signals. The vertical display and the horizontal display signals are not sent to the VGA pins, but they are used by the Screen Positioning Module, in order to know when to update the pixels' colors.

3.1.3 RGB Module

This module generates the R, G and B output signals.

3.2 Screen Positioning Module

For this module, we received the same signals sent to the VGA (h_sync and v_sync), the BCD digits of the hours, minutes and seconds counted, and a clock working at the same speed as the h_sync clock for each pixel. The outputs of this module, were connected to the H_sync and V_sync Generato Module, and this are the R , G and B colors of the VGA pixels.

3.2.1 Operation

The operation of this module is pretty straight foward, by knowing the h_sync and the v_sync signals, and having the same clock as the h_sync generator, it determines at each moment in wich pixel it is working on. To do this, it utilizes the equations (1) and (2), and by knowing the position, it devides the screen in quadrants for each digit of the chronometer. Whenever it detects that it is on a digit quadrant, it uses the NumTo7Seg module to convert the correspond digit to a series of 7 bits, as if it where a 7 segment display, and by knowing each pixel, it determines which pixel it has to print in white, and which in black.
Finally, it feedbacks the H-sync and V_sync Generator module with the colors processed in this module. A simulation of this module was made in GTK wave before uploading to the FPGA, and as we see on Figure 4, we can conclude that it behaves as expected to.

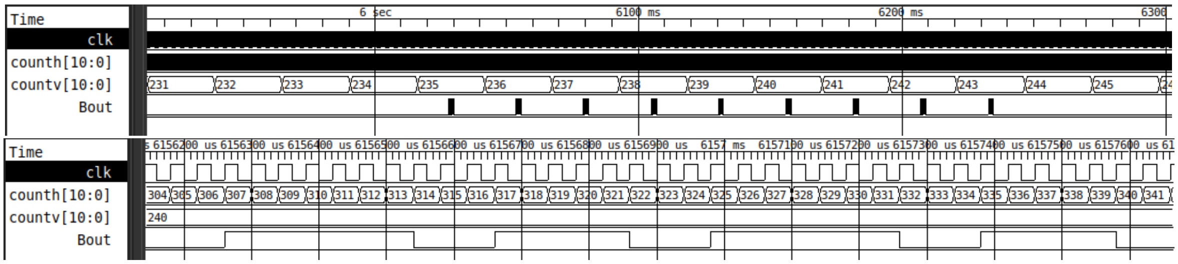

Figure 4: Simulation GTK Wave

3.3 Time Counter Module

To measure times down to the millisecond, a counter module was employed.

3.3.1 Design:

Since the counter is designed to count time units, a clock input was needed. Also, since the main functions on a stopwatch are START/STOP and RESET, the " Enable" and " Reset" inputs were also taken into account in the design. Finally, to properly communicate with the modules for displaying the time, the module was designed to output each digit in BCD format.
Given that the format chosen was HH:mm:ss, 6 BCD output in total were required. In addition, for future implementation, the millisecond digits were also taken into account.
Since the tasks at hand were to:

- Count the number of milliseconds passed
- Calculate the corresponding number of hours, minutes and seconds passed
- Convert each quantity to BCD numbers
- Scale down the clock frequency

The module was designed with 3 sub-modules: a " counter" module, a " watch format" module, a " binary to BCD" module and a " frequency divider" module.

3.3.2 How it works:

First, the frequency divider module converts the FPGA's 50 MHz to the required 1 kHz for the counter module to actually count milliseconds. Second, the counter module, as per the name, counts the number of milliseconds that pass. Since this count needs to reach 100 hours maximum, the size of this counter is 29 bits. Third, the " watch format" module converts the number of millisecons to its respective units: ms to hr, ms to min, and ms to s, taking into account the format in which the data will be presented. Finally, to send this data to the display modules, each quantity is converted into its respective BCD digits. For example, if the " watch format" module returns 59s, the " binary to BCD" module will convert it into a 5 and a 9, each in BCD format.
The RESET and ENABLE inputs only interact with the counter module. The RESET signal will clear the counter and set it back to 0. The ENABLE signal activates or deactivates the counter. In other words, if it is low, the count will not increase.