Two Large Adjustable Digital Clocks On VGA

Introduction to VHDL

Course Project

151220102061 Mustafa Özçelikörs

151220114057 Metin Kundakçıoğlu

Instructor: Asst. Prof. Dr. Erol Seke

CONTENTS

1. INTRODUCTION

Field Gate Programmable Arrays (FPGAs), in today's technology, are reliable alternatives to microcontrollers. They are used for describing hardware, rather than programming a microcontroller. Since they describe a hardware and does not have a sequential instruction set, they provide parallel operation in system prototyping. Parallel operation provides more speed. Since they are fast, they are widely used in DSP applications like video and sound processing, alongside with military bus applications that require high speed.

FPGA chips are described with HDL (Hardware Description Languages). Two of the most common examples of HDL languages are VHDL and Verilog HDL. In this work, we use VHDL language, which has IEEE compatible libraries and is object oriented, in order to implement two large digital clocks using Video Graphics Array (VGA). In video applications, VGA plays an important role on showing the processed video or even showing an interface. With this project, we design an introductory interface to show two seperate digital clocks that are adjustable.

2. SYSTEM PRELIMINARIES

2.1. XILINX SPARTAN 3E STARTER KIT

The Spartan 3E Starter Board provides a powerful and highly advanced self-contained development platform for designs targeting the Spartan 3E FPGA from Xilinx. It features a 500K gate Spartan 3E FPGA with a 32 bit RISC processor and DDR interfaces [1].

The board also features a Xilinx Platform Flash, USB and JTAG parallel programming interfaces with numerous FPGA configuration options via the onboard Intel StrataFlash and ST Microelectronics Serial Flash. The board is fully compatible with all versions of the Xilinx ISE tools including the free WebPack. The board ships with a power supply and USB cable for programming so designs can be implemented immediately with no hidden costs. The Spartan 3E Starter board is also compatible with the MicroBlaze Embedded Development Kit (EDK) and PicoBlaze from Xilinx [1]. The board is shown in Fig 1.

The Kit has the following features and connectors [1]:

100-pin Hirose FX2 connector
Three 6-pin Pmod connectors
DB15HD VGA
PS/2 keyboard
Two DB9 RS-232 connectors
RJ-45 Ethernet

16-pin header for optional LCD modules
SMA connector for high-speed clock input
Xilinx XC3S500E FPGA
Xilinx XCF04 Platform Flash for storing FPGA configurations
64MB Micron DDR SDRAM
16MB Numonyx StrataFlash
2MB ST Microelectronics Serial Flash
Linear Technologies Power Supplies
Texas Instruments TPS75003 Triple-Supply Power Management IC
SMSC LAN83C185 Ethernet PHY

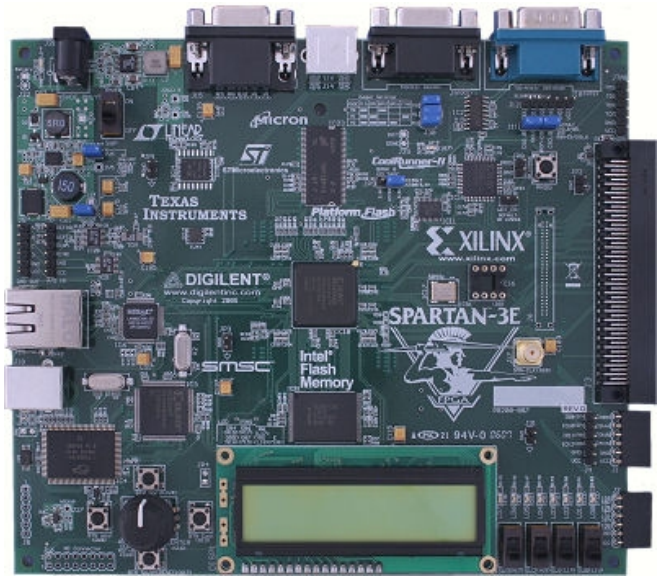


Fig 1. Xilinx Spartan 3E Starter Kit [1]

2.1. VIDEO GRAPHICS ARRAY

Video Graphics Array (VGA) refers specifically to the display hardware first introduced with the IBM PS/2 line of computers in 1987, but through its widespread adoption has also come to mean either an Amplitude Modulated computer display standard, the 15-pin D-subminiature VGA connector or the 640x480 resolution itself [2].

VGA was the last IBM graphics standard to which the majority of PC clone manufacturers conformed, making it the lowest common denominator that virtually all post-1990 PC graphics hardware can be expected to implement. It was officially followed by IBM's Extended Graphics Array (XGA) standard, but was effectively superseded by numerous slightly different extensions to VGA made by clone manufacturers, collectively known as Super VGA [2].

Today, the VGA analog interface is used for high definition video, including resolutions of 1080p and higher. While the transmission bandwidth of VGA is high enough to support even higher resolution playback, there can be picture quality degradation depending on cable quality and length. How discernible this degradation is depends on the individual's eyesight and the display, though it is more noticeable when switching to and from digital inputs like HDMI or DVI [2].

In application, H-SYNC and V-SYNC signals define the horizontal and vertical position of the pixel, whereas RED, GREEN, and BLUE signals are used to create color into the selected pixel. The pinout of a VGA connector is given in Fig 2.

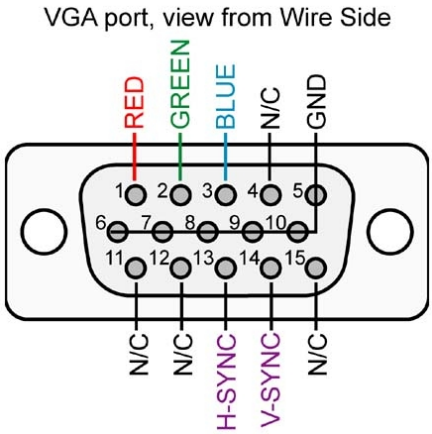


Fig 2. VGA Connector Pinout

2.2. ROTARY ENCODER

A rotary encoder, also called a shaft encoder, is an electro-mechanical device that converts the angular position or motion of a shaft or axle to an analog or digital code [3].

There are two main types: absolute and incremental (relative). The output of absolute encoders indicates the current position of the shaft, making them angle transducers. The output of incremental encoders provides information about the motion of the shaft, which is typically further processed elsewhere into information such as speed, distance, and position[3].

Fig 3. Rotary Encoder

Rotary encoders are used in many applications that require precise shaft unlimited rotation—including industrial controls, robotics, special purpose photographic lenses, computer input devices (such as optomechanical mice and trackballs), controlled stress rheometers, and rotating radar platforms [3]. The waveform to determine the clockwise and counterclockwise rotation in terms of the rising and falling edges of channels A and B in a rotary incoder is given in Fig 4.

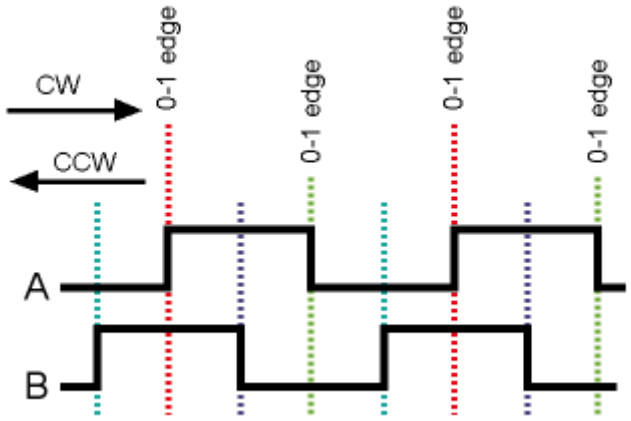
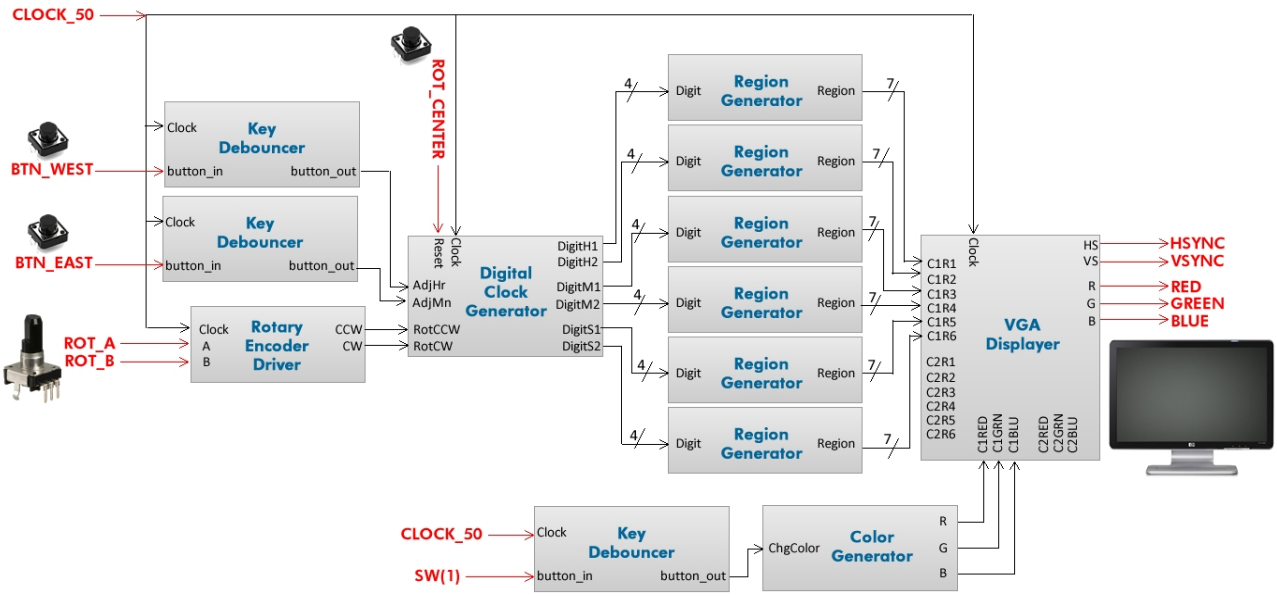


Fig 4. Rotary Encoder Waveform

3. SYSTEM DESIGN & IMPLEMENTATION

During the consideration of the system design, the seperate entity structures for each functional module is designed. This allows us to debug the code easily using test benches for seperate entities. The designed system block diagram (with clock1 connected) is given below



The figure shows only one instance of Digital Clock Generator connected to VGA Displayer module's corresponding region.

RotaryEncoderDriver is the entity that drives the rotary encoder and senses the rising and falling edges of channels A and B in order to output whether the rotation is clockwise, counterclockwise, or whether there is no rotation at all. When an adjust button is hit, the rotary encoder becomes active until the same button is pressed again. This allows users to use 4 config buttons and a rotary encoder to adjust the two clocks.

DigitalClockGenerator is the entity that gets the adjustment buttons and rotary encoder alongside with the 50MHz system clock in order to create 6 digits for the clock (2 digit hour, 2 digit minute, 2 digit seconds).

The way the DigitalClockGenerator creates clock is that it is designed to have a clock divider that is a 1-second counter. The "minutes" register is incremented when "seconds" register reaches 59. Similarly, the "hours" register is incremented until 23 when "minutes" register reached 59. There are also seperate digit registers of these registers which are used for direct interfacing to the RegionGenerator blocks.

The 4-bit digit output from DigitClockGenerator is essentially between "0000"(0) and "1001"(9) showing which number is the corresponding digit. These digit informations are subjected to a decoder which is called RegionGenerator. The RegionGenerator creates a 7-bit data which consists of the regions constructing a 7-Segment number pattern, shown in the Fig 6.

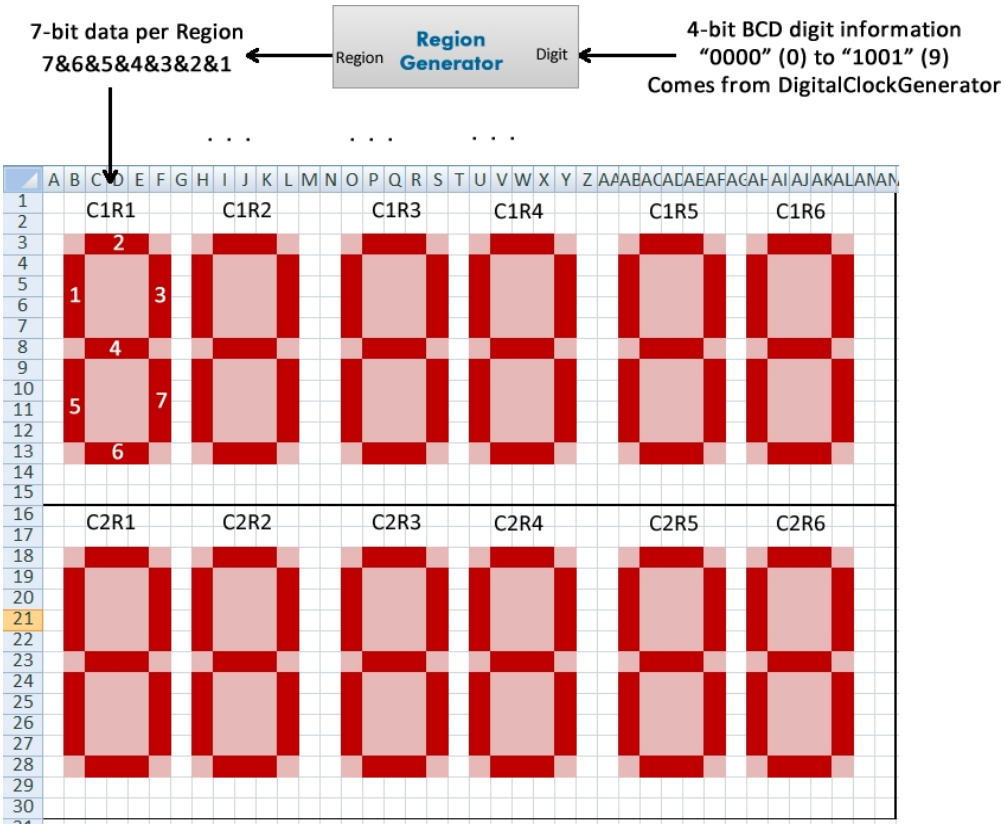


Fig 6. Regions in the Screen (Excel Simulation)

In the Fig 6, the simulation of the pixel blocks are done in Microsoft Excel. In this simulation, each square block are considered to be 16 pixels, which makes horizontal distance 640 pixels, and vertical distance 480 pixels.

It is seen that the display needs to have 12 digit information (6 for clock#1 and 6 for clock#2) coming from RegionGenerator blocks, from their corresponding DigitalClockGenerator entity.

One of the most crucial blocks in the system which displays all the information is VGADisplayer block. This block takes 12 digit information (in 7-bit region info) alongside with R,G,B color signals for each of the clocks. ColorGenerator block is designed to be a basic state machine that provides different color mode each time the ChngColor button is pressed.

The VGADisplayer block has been built onto the classic VGA driver code written in VHDL, which is creating counters for hsync and vsync signals in order to apply R,G,B values when a desired pixel block is reached; using 25MHz of clock in the process. To display the digits, the 12 digits are written into arrays and are indexed through a signal called digIn. The following process determines whether the desired section of the screen is reached in terms of hsync

and vsync counters and the corresponding digIn signal (0 through 12) is produced accordingly.

```
process(clk) is begin
        if(rising_edge(clk)) then
                if(hcntr<97 and vcnt<240)then
                        digIn <= 1;
                elsif(hcntr<193 and vcnt<240)then
                        digIn <= 2;
                elsif(hcntr<305 and vcnt<240)then
                        digIn <= 3;
                elsif(hcntr<401 and vcnt<240)then
                        digIn <= 4;
                elsif(hcntr<513 and vcnt<240)then
                        digIn <= 5;
                elsif(hcntr<609 and vcnt<240)then
                        digIn <= 6;
                elsif(hcntr<97 and vcnt<480)then
                        digIn <= 7;
                elsif(hcntr<193 and vcnt<480)then
                        digIn <= 8;
                elsif(hcntr<305 and vcnt<480)then
                        digIn <= 9;
                elsif(hcntr<401 and vcnt<480)then
                        digIn <= 10;
                elsif(hcntr<513 and vcnt<480)then
                        digIn <= 11;
                elsif(hcntr<609 and vcnt<480)then
                        digIn <= 12;
                else
                        digIn <= 12;
                end if;
        end if;
end process;
```

With the help of the digIn signal, we know which digit to process at a certain time. In an entirely other process, the hsync counter and vsync counter are questioned in order to write R,G,B information, when the corresponding region of the digIn signal is encountered by the hysnc and vsync counters. The following is an example for the first region:

```
if(hcntr>16+hshift(digIn) and hcntr<32+hshift(digIn) and vcnt>48+vshift(digIn) and
vcnt<112+vshift(digIn))then
       if(ClockRegions(digIn)(0)='1')then
              B<=colorarr_B(digIn);
              G<=colorarr_G(digIn);
              R<=colorarr_R(digIn);
       elsif(ClockRegions(digIn)(0)='0')then
              B<='0';
              G<='0';
              R<='0';
       end if;
....
```

In the code above, we check if the counter has reached the first region, which is a rectangular shape with corners (x1,y1)=(16,48) and (y1,y2)=(32,112). We have also offset values when checking this information, since we need to shift the digit to the corresponding digit position using digIn signal indexing.

If this checking is true, we determine if the region information of that digit (which is 1 in this case) is '1', then we send the color information of the digit that is obtained from ColorGenerator entity to R,G,B signals ; if not however, we send '0' for R,G,B signals, indicating that region to be black colored.

This if statement is repeated seven times for seven regions, having different rectangular shapes each time, in order to show the entire clock.

The VGADisplayer entity has also another function that triggers solid shaped square pixel blocks seperating every two digits, every half a second, which is a convention in the digital clocks.

It is important to note that every single key including rotary encoder is debounced using KeyDebouncer blocks, using 5 millisecond of sampling time with three samples.

The application picture is given in Fig 7:

4. CONCLUSION

By using Video Graphics Adapter, one can display anything in a screen. There are a lot of applications of VGA with VHDL and FPGA and this project is successfully achieved. With this project, we were able to display two seperately adjustable digital clocks using VGA.

5. REFERENCES

[1]http://www.digilentinc.com/Products/Detail.cfm?NavTop=2&NavSub=423&Prod=S3EBOARD&CFID=10380660&CFTOKEN=17a9726e697e72e4-305807F2-5056-0201-020149D1BD96AF58

[2]http://en.wikipedia.org/wiki/Video_Graphics_Array

[3] http://en.wikipedia.org/wiki/Rotary_encoder