FPGA VGA Text.
By Quintin Immelman

Introduction

I had to design a PLC for a warping machine which required the operator to enter many variables into the system before the logic controller would evaluate and perform the task. The easiest way (or so I thought) would be to use a VGA monitor as this would allow much more information in one hit. I found information on VGA to be hard to come by. This Doc is meant as an introduction to VGA text and by no means the be all and end all on the subject. I have come up with a more efficient way of text-out but this is beyond the scope of this document.
I have included the method at the end of this document for the curious.

VGA

VGA stands for Video Graphics Array. It encompasses anything from 16 colours to 32 Bit true colour. To produce true colour a DAC is used to alter the voltages on each of the Red, Green and Blue guns to change the intensity of each.
I will be discussing the simple 16 colour VGA. This design was tested on my Xilinx Spartan 3 starter kit as it has a VGA connector already mounted. This method should work for any board as long as you put a VGA connector on. Please remember to always use a 270 ohm resistor in series with the Red, Green and Blue guns on a 3.3V FPGA.

We will be using the six main signals viz.

clk50_in : in std_logic; This is the 50Mhz oscillator found on the board.
Red_out : out std_logic; This will control the Red gun on the monitor.
Green_out : out std_logic; This will control the Green gun on the monitor.
Blue_out : out std_logic; This will control the Blue gun on the monitor.
hs_out : out std_logic; This is the Horizontal Sync.
vs_out : out std_logic); This is the Vertical Sync.

The VGA standard calls for 25 MHz clocking so we need to write code to half the frequency of the 50 MHz on board clock by simply setting up our architecture and defining the process as follows:

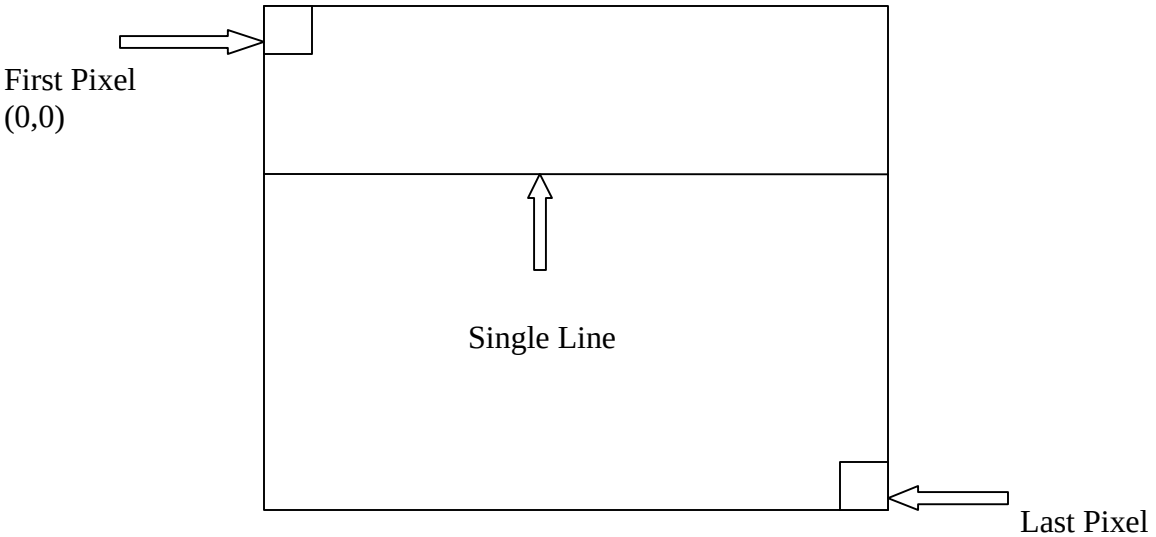
```
process (clk50_in)
begin
        if clk50_in'event and clk50_in='1'
          if (Clk25 = '0')then
            Clk25 <= '1' after 2 ns;
          else
            Clk25 <= '0' after 2 ns;
                  end if;
        end if;
```

end process;

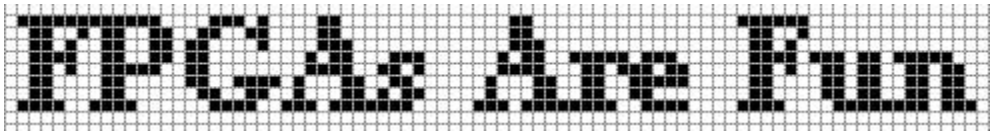
It works as follows: Firstly we check if there is an event on the clocking pin, (either high or low counts as an event). We evaluate further to determine if the event is high. By acting only on high events and ignoring low clocking events we have effectively halved the clocking.

Next step is to check whether our own 25Mhz clock is high or low and invert if required.

That takes care of the timing. The next step is to understand how images are placed on the screen. VGA monitors use a Raster scan. What happens here is the guns travel from the top left of the screen to the bottom right in an ordered manner.



The guns are activated on the rising edge of the Saw-tooth waveform and the retrace begins on the falling edge. To control this we have to bring our hs_ out momentarily low for each front porch. I did this by using a horizontal counter. When the counter reaches 640 I momentarily pull hs_ out low and also reset everything for the next line.
Very interesting you may be thinking," I already know all this but how will all this help me except for placing symmetrical images on the screen?"
Text was a challenge as the standard for text in a 640 x 480 display is an array of 8bits x 8 bits or 8 pixels by 8 pixels.
Now that we understand the theory and the timing done lets proceed.
I define my characters by using a bitmap editor. In this particular case I used Visual Studio.
Define your font size as 8 and type in what you want displayed. This gives a grid for you to determine when to put guns on and when to leave them off. If you'er arty you may not need this tool.



Example F = Line 1:01111111
Line 2:00110010
Line 3:00110100
Line 4:00111100
Line 5:00110100
Line 6:00110000
Line 7:00110000
Line 8:01111000

I wrote a C++ program to convert my text file of zeros and ones into an array.
I have not included this code as it will require you to download .NET V2 Beta which is quite a large file. I have included a text file containing a basic version of the code enabling you to modify and compile it as per your specific requirements. I am still working on a program where you can enter the text and the program will automatically convert the string into an array.
In our Text "FPGAs Are Fun" we have an array size of 82 x 8. Please see FPAGs.txt and the array created in Output.txt.
Now that we have the abovementioned sorted let's get to implement it!

```
process (Clk25)
TYPE Screen_Line1 is ARRAY(0 to 7, 0 to 81) OF std_logic; -- Define Array Type

CONSTANT char_L1 : Screen _Line1 := (( Enter Array Elements Here See VHDL File));
```

We need a separate counter for the array line and one for the array elements:

```
variable Line:integer:=0;
variable Pixel:integer:=0;
```

Now to display text:
Lets Center the Text Horizontal and vertically.

Well we know the screen has 640 pixels per line and 480 lines, our array is 82 by 8 thus 640 / 2= 320 82 / 2 = 41.
320 - 41 = 279 This is our Starting point.
320 + 41 = 361 This is our Ending point.
and 480 / 2 = 240
240 - 4 = 236 This is our starting line.
240 + 4 = 244 This is our ending line.
I started my counter at 144 horizontal thus we add 144 to each horizontal element. this gives us 432 and 505.
So now we check our Horizontal_Counter and Vertical_Counter.

On each high clock of our software 25 MHz clock we process the information.

```
begin
```

```
if Clk25'event and Clk25 = '1' then
        if (Horizontal_Counter >= "0010010000" ) -- 144
        and (Horizontal_Counter < "1100010000" ) -- 784
        and (Vertical_Counter >= "0000100111" ) -- 39
and (Vertical_Counter < "1000000111" ) -- 519
        then
                -- every other combination will be BLACK
                Red_out <= '0';
                Green_out <= '0';
                Blue_out <='0';
-----Line 1
        if (Horizontal_Counter >=  "0110100111" )--423
        and (Horizontal_Counter <= "0111111001")--          505
        and (Vertical_Counter >=       "0011101100") --236
        and (Vertical_Counter <= "0011110011") then -- 243
                if(Pixel <= 81) then --Line 1 Lets make our Text WHITE
                        Red_out <= char_L1(Line, Pixel);
                        Green_out <= char_L1(Line, Pixel);
                        Blue_out <= char_L1(Line, Pixel);
                        Pixel:= Pixel+1;
                 elsif(Pixel >= 82) then  -- All else BLACK
                        Red_out <= '0';
                        Green_out <= '0';
                        Blue_out <= '0';
                end if;
        end if;
        end if;
```

If you want to change the colours, just modify which guns come on. In this example all guns are firing as can be seen by Red_out <= char_L1(Line, Pixel) ect.

What is happening? When the Horizontal counter is between 423 and 505 we check which line of the array to display, we then get the values element for element for that line. 0 indicates the guns are off and a 1 indicates the guns are on. Once we have finished we set all other guns to 0 rendering a black background. When the Vertical counter increases we increase our Line counter. The VGA monitor will refresh at around 60 times per second, so when the vertical counter reaches 480 we reset it and reset the line counter ready for the next update.

Rest of code follows:

```
if (Horizontal_Counter > "0000000000" )
        and (Horizontal_Counter < "0001100001" ) -- 96+1
                then
                        hs_out <= '0';
                else
                hs_out <= '1';
```

```
        end if;
                if (Vertical_Counter > "0000000000" )
        and (Vertical_Counter < "0000000011" ) -- 2+1
                then
                vs_out <= '0';
                else
                vs_out <= '1';
        end if;

                Horizontal_Counter <= Horizontal_Counter+"0000000001";
        if (Horizontal_Counter="1100100000") then
        Vertical_Counter <= Vertical_Counter+"0000000001";
        Horizontal_Counter <= "0000000000";
                        Pixel:= 0;
                        if (Vertical_Counter >=         "0011101100") -- First Line
                          and (Vertical_Counter <= "0011110011") then
                                        if (Line <= 7) then
                                        Line:= Line+1;
                                        elsif (Line >= 8) then
                                         Line:= 0;
                                        end if;
                         end if;
        end if;
        if (Vertical_Counter="1000001001") then
        Vertical_Counter <= "0000000000";
                        Line:= 0;
        end if;
        end if;
end process;
```

For those interested in a more complex but highly efficient method.

Read the Xilinx App694 and download the zip file from their web. This includes a Perl script to create Intel hex files with Bit swapping.
The idea is really quite simple:
Define the entire screen once in a separate file. Use the Perl script to create the bit file. Add this to the user section of your Prom. The Spartan 3 uses just under 1 Mb and finally leaves you with 1 Mb free. The screen takes 640 x 480 pixels which is 307200 bits.
Now for the tricky bit. On start-up the FPGA runs all process once thus it is up to you to slip in a process that will read the user section and buffer it into the onboard RAM. Xilinx does help here as the FPGA dose not reset the EEPROM counter so you do not have to determine your code size and skip through it. From here you can now update the screen from RAM, which by the way is pretty much how your video card on your PC works.
If you have any questions or a better way of doing this please e-mail me on quintin@meds.co.za. Please no SPAM!!!