# *Assignment -4*
# *Text display on VGA using PS/2 keyboard*

## Introduction

In this assignment, a Xilinx Virtex II Pro FPGA Board with a XC2VP30 device and 896 package has been used. The board includes a 15 pin video DAC connector to support the VGA monitor and a 6 pin PS/2 serial port to support the keyboard.

***Objective:*** The main objective is to learn to use the FPGA to accept input from keyboard and with the help of font rom in .coe file, display the input character on the VGA. Pressing the “Enter” key should clear the screen, only one character needs to be displayed at a time. Pressing any key other than the number keys or the Enter key should display the character “E”. The character should be displayed in the top left corner of the VGA monitor.

***Background:***

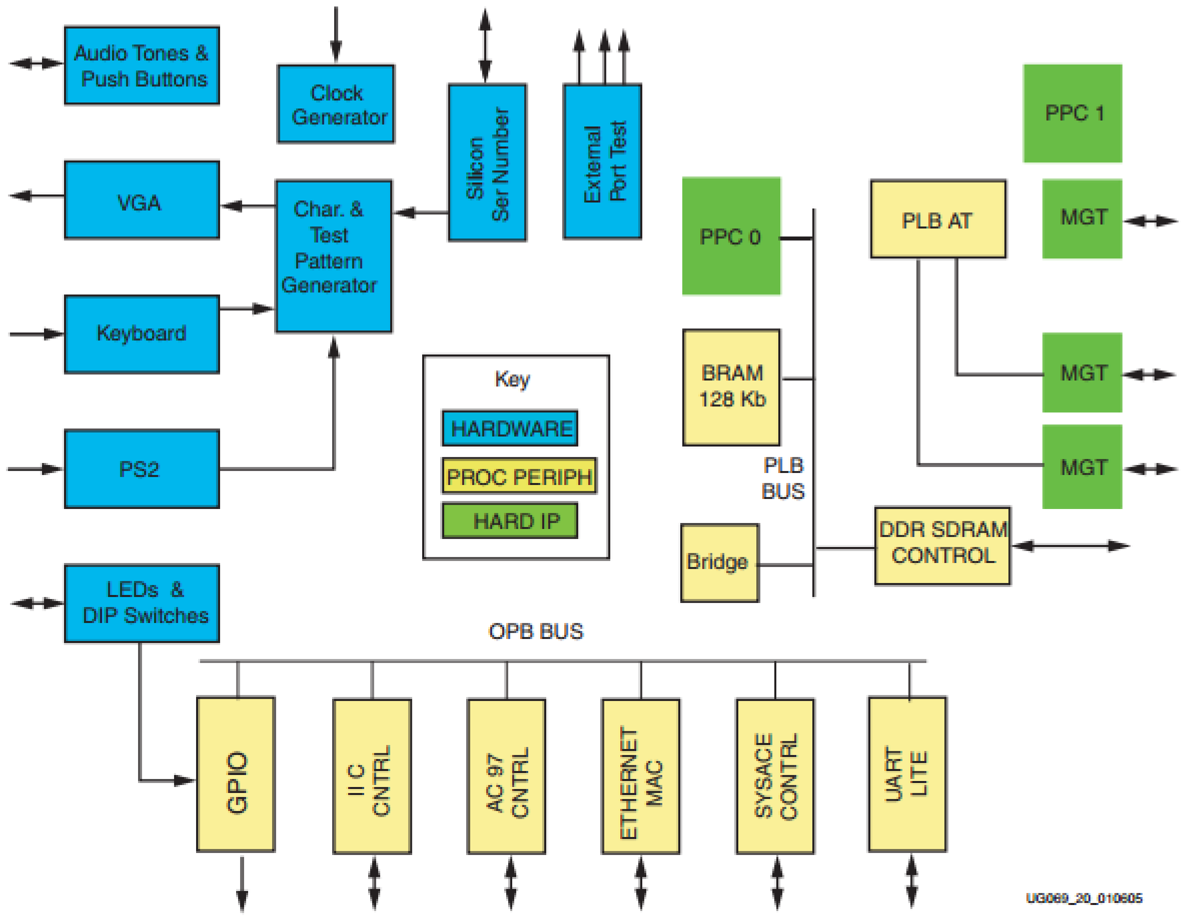


**Fig. 1**: *Block Diagram of Virtex II Pro Development Board* [1]

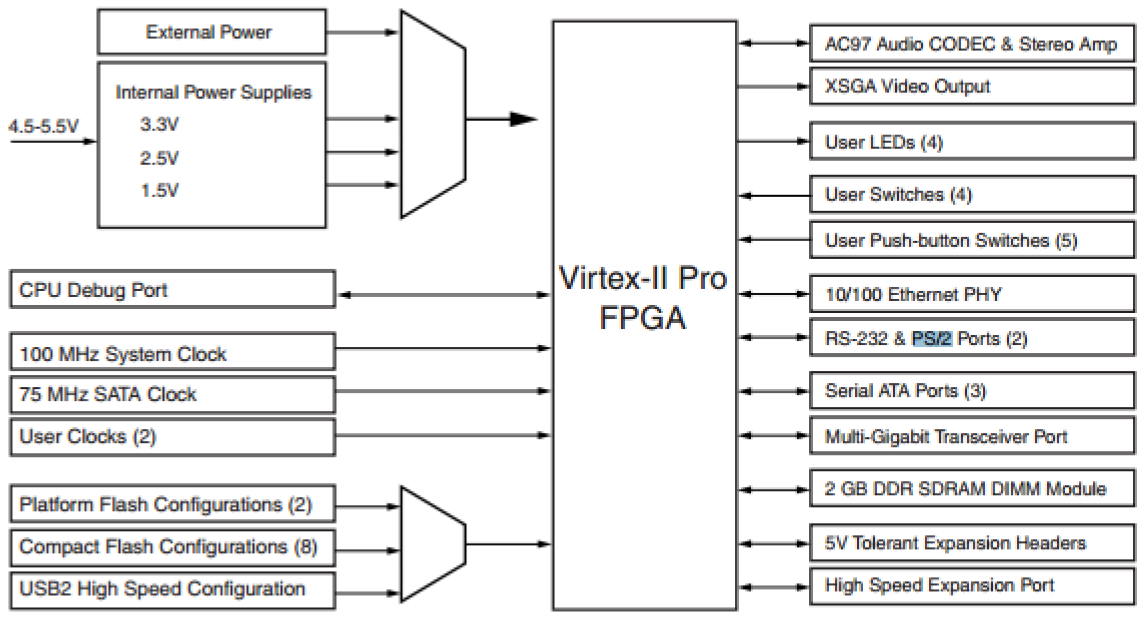

External Power
Internal Power Supplies
4.5-5.5V
3.3V
2.5V
1.5V
CPU Debug Port
100 MHz System Clock
75 MHz SATA Clock
User Clocks (2)
Platform Flash Configurations (2)
Compact Flash Configurations (8)
USB2 High Speed Configuration
Virtex-II Pro FPGA
AC97 Audio CODEC & Stereo Amp
XSGA Video Output
User LEDs (4)
User Switches (4)
User Push-button Switches (5)
10/100 Ethernet PHY
RS-232 & PS/2 Ports (2)
Serial ATA Ports (3)
Multi-Gigabit Transceiver Port
2 GB DDR SDRAM DIMM Module
5V Tolerant Expansion Headers
High Speed Expansion Port

**Fig. 2**: *Block Diagram of Virtex II Pro Development Board* [1]

These are the onboard hardware that we are interested in for this assignment:

- PS/2 Connector
- Clock Generator

**PS/2 Connector:**

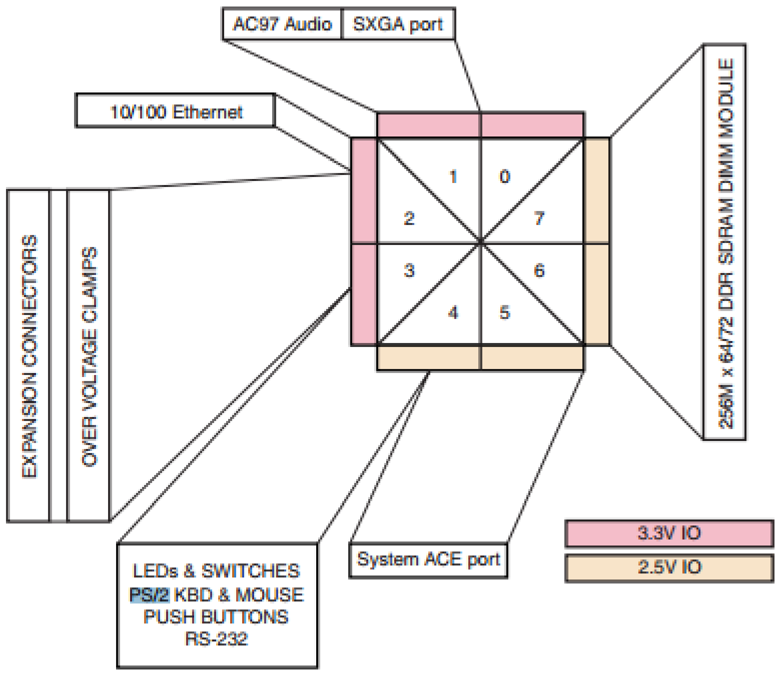
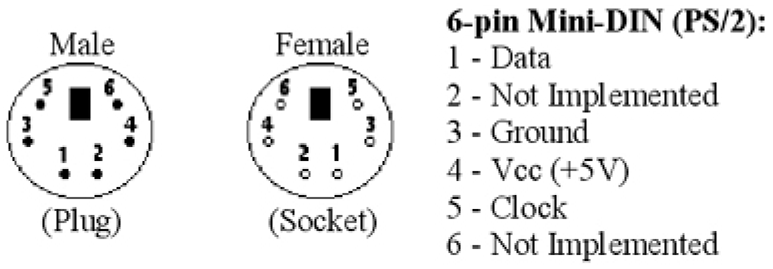


**Fig. 3**: *I/O Bank Connections to Peripheral Devices showing LEDs and PS/2 on #4* [1]



**6-pin Mini-DIN (PS/2):**

1 - Data
2 - Not Implemented
3 - Ground
4 - Vcc (+5V)
5 - Clock
6 - Not Implemented

**Fig. 4**: *Pins on PS/2 connector* [2]

- The development kit has three serial ports on board namely:
  - RS-232 ….. x 1
  - PS/2 ….. x 2

  RS-232 is a 9pin connector used to communicate to a host computer via COM port. There are also two PS/2 ports mainly to support the keyboard and mouse use. System. All of the serial ports are equipped with level-shifting circuits, because the Virtex-II Pro FPGAs cannot interface directly to the voltage levels required by RS-232 or PS/2. [1]

**VGA Connector:**

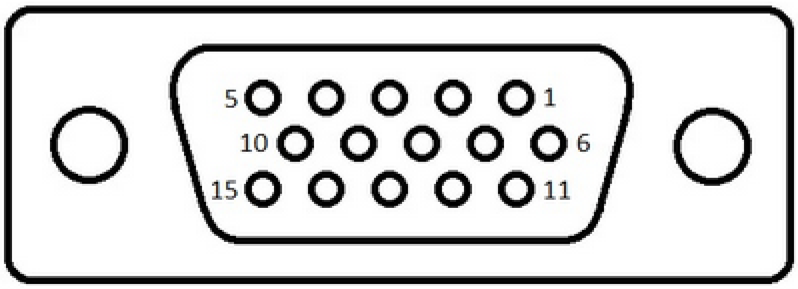


**Fig. 5**: *VGA Female connector* [3]

- *Blank signal*: Overrides the RGB signal and blanks the display output.
- *Pixel clock*: Various clock rates can be created by the software but in this particular case, we use it at a rate of 25MHz since here we are using a 640x480@60Hz monitor.
- *RGB*: These are the three different analog voltage signals that decide the color of each Pixel. The three colors are Red, Green and Blue.

| Red (R) | Green (G) | Blue (B) | Resulting color |
|---|---|---|---|
| 0 | 0 | 0 | black |
| 0 | 0 | 1 | blue |
| 0 | 1 | 0 | green |
| 0 | 1 | 1 | cyan |
| 1 | 0 | 0 | red |
| 1 | 0 | 1 | magenta |
| 1 | 1 | 0 | yellow |
| 1 | 1 | 1 | white |

**Table 1**: *RGB Signals for 7 different colors* [4]

**COE file**

COE files are memory files with extension .coe created with the help of Memory Editor - a tool that specifies memory contents and initialization values for CORE Generator memory cores. These files have one or more memory blocks defining the contents of one or more COE files. For each memory block defined in a CGF file, the Memory Editor generates a separate COE file. [5]

## Procedure

This section of the report provides a brief description of the steps that were followed to implement the assignment. Xilinx ISE 9.2i design software was used for Verilog code write-up, synthesize and implementation and to burn the code to the FPGA kit. We use some fragments of the codes from assignments 2 and 3 to synchronize the keyboard and VGA. Also, the font rom is provided by Dr. Lin.

1. Created a new project in ISE named 'Lab4'. The logic implemented to write the code has been given below.
2. The code is been divided into 4 modules. One to synchronize the keyboard, the output of which is fed as the input to the module that is been used to synchronize the VGA monitor with FPGA development kit. We also use a module to handle the coe file and lastly a top module is used to handle the signal flow within the rest of the modules.
3. *Types of signals used for Keyboard:* There are two signals involved in this procedure, one is the clock signal and the other is the data signal which is received from the keyboard. This signal is sampled at the negative clock i.e. at the falling edge.

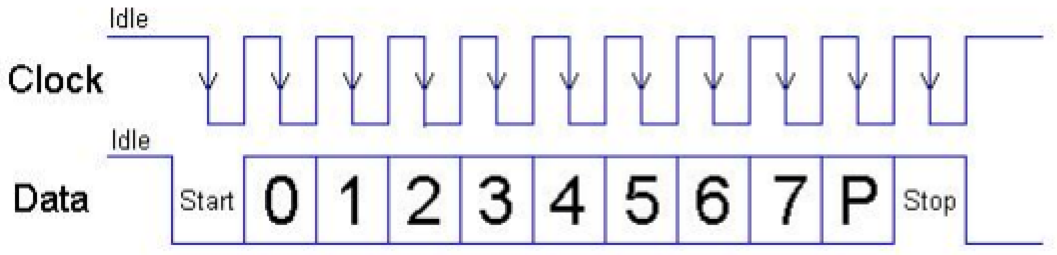


**Fig. 6***: Nature of Data and clock signal* [4]

4. *Reading the font rom:* The used rom has width of 8 bits and a depth of 128. This created using the Xilinx coregent tools and the provided coe file. The clock signal and the address signal are given as inputs to the rom which in turn provides an 8 bit data based on which we choose whether or not to display color on VGA.
   In this ROM, the 4 MSBs of the 7 -bit address are used to identify the character we referred this to as char address, the other three LSBs of the address are used to identify the row within a character pattern which is referred to as row address.

   ***7 bits Address signal of COE → 4 bits of character address + 3 bits of row address***

   With this input, we can obtain 8 bit data to be plotted on the 8x8 area chosen to display the text on VGA.

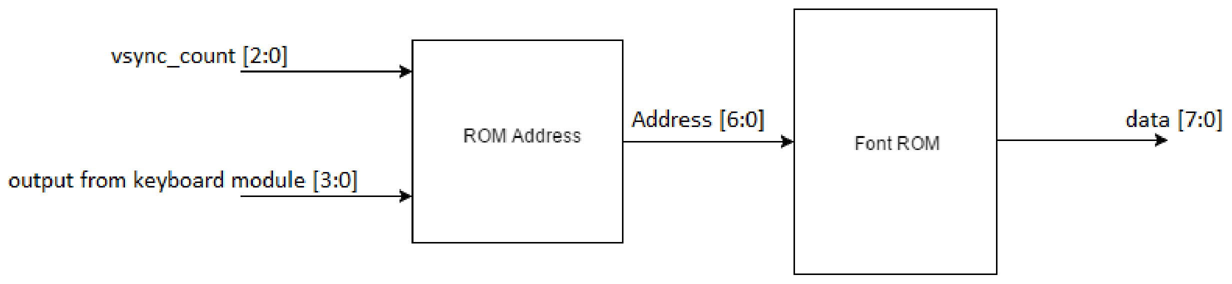


**Fig. 7**: *Two-stage text generation*

5. *Reading the keys:* The data signal from the keyboard is collection of 11 bits in which the main data is stored in the 8 bits starting from the 2nd bit and ending at the 9th bit as shown in the figure above. The rest of the bits indicate the start, end and parity. Each key has a unique code which is sent in the mention 8 bit data set. This data is sent continuously until the key is pressed followed by the ending bit 0.

| Key codes in Hexadecimal | Character Address | KEY |
|---|---|---|
| 45 | 0000 | 0 |
| 16 | 0001 | 1 |
| 1E | 0010 | 2 |
| 26 | 0011 | 3 |
| 25 | 0100 | 4 |
| 2E | 0101 | 5 |
| 36 | 0110 | 6 |
| 3D | 0111 | 7 |
| 3E | 1000 | 8 |
| 46 | 1001 | 9 |
| 5A | 1011 | Enter |

**Table 2**: *Keyboard scan codes* [6]

6. *Using the VGA monitor:* Here, the inputs are the two buttons, pixel clock signal and outputs are the synchronization signals and the signals for RGB.
7. We named the horizontal and vertical synchronization signals as h_synch and v_synch respectively, we also initialize counters for each synch signal in order to identify which pixel is being written. We start from h_counter and v_counter equal to zero, the h_counter counts till the total period of 1 cycle as shown above and then is reset to zero and in the meantime v_counter counts till the total period of 1 cycle of the vertical signal and then is reset to zero conditions are given to make the sync signals either high or low depending on which section of the cycle their respective counter is reached. The limit for each section of cycle for both horizontal and vertical signals where given by Dr. Lin in the handout for assignment 2 but since these were given in terms of seconds, we had to convert them to number of counts by multiplying the given time with pixel

clock used. In my approach, a 25MHz pixel clock was used and the following counter values were implemented.

| Type of refresh cycle | Parameter | Time | count |
|---|---|---|---|
| HORIZONTAL | A | 31.77µs | 795 |
| | B | 3.77 µs | 95 |
| | C | 1.89 µs | 48 |
| | D | 25.17 µs | 640 |
| | E | 0.94 µs | 12 |
| VERTICAL | O | 16.6 ms | 523 |
| | P | 64 µs | 2 |
| | Q | 1.02 ms | 33 |
| | R | 15.25 ms | 480 |
| | S | 0.35 ms | 8 |

**Table 3**: *Count values obtained for the given time using a pixel clock of 25Mhz*

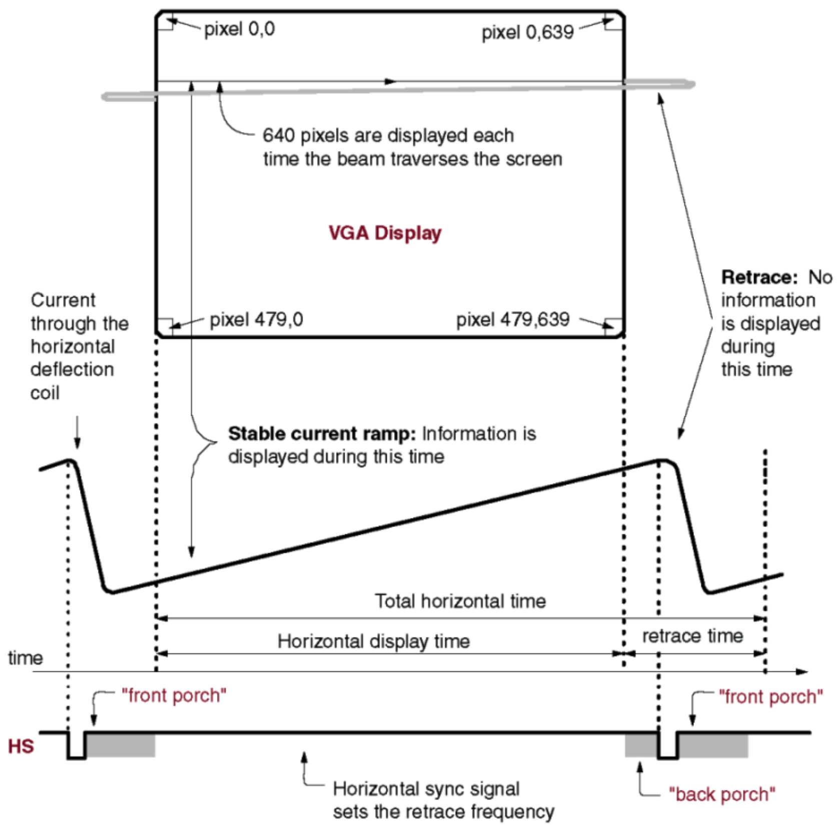

pixel 0,0
pixel 0,639
640 pixels are displayed each time the beam traverses the screen
VGA Display
pixel 479,0
pixel 479,639
Current through the horizontal deflection coil
Retrace: No information is displayed during this time
Stable current ramp: Information is displayed during this time
Total horizontal time
Horizontal display time
retrace time
time
"front porch"
"front porch"
HS
Horizontal sync signal sets the retrace frequency
"back porch"

**Fig. 8:** *Diagram to give an idea of the logic implementation for VGA display* [7]

Once the active video section of the cycle is reached, a case condition is used to display 8 different colors based on the output of the LEDs.

| LED Display | | | Decimal value | R | G | B |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 255 | 255 | 255 |
| 0 | 1 | 0 | 2 | 255 | 0 | 0 |
| 0 | 1 | 1 | 3 | 0 | 255 | 0 |
| 1 | 0 | 0 | 4 | 0 | 0 | 255 |
| 1 | 0 | 1 | 5 | 255 | 255 | 0 |
| 1 | 1 | 0 | 6 | 0 | 255 | 255 |
| 1 | 1 | 1 | 7 | 255 | 0 | 255 |

**Table 4**: *RGB Values based on the output of LEDs*

A white color background and a red color text was used in this assignment.

8. UCF*:* The UCF values were downloaded from [7] used.

| Signal | Value |
|---|---|
| Keyboard Clock | AG2 |
| Keyboard Data | AG1 |
| VGA clock | AJ15 |
| Horizontal synch | B8 |
| Vertical synch | D11 |
| blank | A8 |
| Comp synch | G12 |
| Pixel clock | H12 |
| red[7] | H10 |
| red[6] | C7 |
| red[5] | D7 |
| red[4] | F10 |
| red[3] | F9 |
| red[2] | G9 |
| red[1] | H9 |
| red[0] | G8 |
| green[7] | E11 |
| green[6] | G11 |
| green[5] | H11 |
| green[4] | C8 |
| green[3] | D8 |
| green[2] | D10 |
| green[1] | E10 |
| green[0] | G10 |
| blue[7] | E14 |
| blue[6] | D14 |

| | |
|---|---|
| blue[5] | D13 |
| blue[4] | C13 |
| blue[3] | J15 |
| blue[2] | H15 |
| blue[1] | E15 |
| blue[0] | D15 |

**Table 5**: *UCF Values* [7]

9. Next, we check the syntax form design utilities → check syntax option in the processes window.
10. We then synthesize the written code. If no errors occur, we proceed to the next step. Else, we need to debug the code accordingly.
11. Once the synthesis is finished successfully, we need to implement design.
12. Generate the programing file. The tools for these three steps can be found on the processes window.
13. A bit file is generated in the project folder. With the same name as the name of the project.
14. We now connect the FPGA kit to the system and the keyboard & VGA to the FPGA kit and turn the kit and ON.
15. We use the generated bit file and map it using configure device (iMPACT) tool available in generate programming file tool.
16. The board is now ready for testing.

***Difficult part:***

The most difficult part of the assignment for me was to understand the functionality of the font rom. getting to know how to access each row and then 8 bits of a particular row for character display. FPGA PROTOTYPING BY VERILOG EXAMPLES by Chu was really helpful, it beautifully explains how to divide the address signal for font rom and synchronize the horizontal and vertical synchronization signals of VGA to access the required data signal.

## Results

The results were obtained as expected. 11 characters ie., 0-9 and E were displayed using only the numerical keys below the functions keys on keyboard. Pressing enter key would clear the screen and pressing any other key would display E on the VGA monitor.

## Conclusion

The obtained result supports the desired objective. All keys worked as mentioned above and all digits were displayed within an 8x8 pixel area on the left top corner of the VGA monitor.

## References

[1] https://www.digilentinc.com/Data/Products/XUPV2P/XUPV2P_User_Guide.pdf
[2] http://www.computer-engineering.org/ps2protocol/
[3] http://www.computer-engineering.org/ps2protocol/
[4] http://www.eecs.ucf.edu/~mingjie/EEL5722/EEL5722c%20lab3%20Tutorial.pdf
[5] http://www.xilinx.com/itp/xilinx10/isehelp/cgn_r_memory_editor_overview.htm
[6] https://eewiki.net/pages/viewpage.action?pageId=28278929

[7] http://www.digilentinc.com/Products/Detail.cfm?NavTop=2&NavSub=453&Prod=XUPV2P&CFID=18922620&CFTOKEN=ff23df9a79e4c2ac-061E4AC0-5056-0201-02D0AEB908347149

[8] http://ece.gmu.edu/coursewebpages/ECE/ECE448/S13/viewgraphs/ECE448_lecture7_VGA_1.pdf

[9] http://www.cs.ucr.edu/~jtarango/cs122a_lab4.html

[10] Pong P. Chu, "FPGA PROTOTYPING BY VERILOG EXAMPLES", WILEY PUBLICATIONS, 2008, pp. 341–352.