Worcester Polytechnic Institute

ECE 574: Modeling and Synthesis of Digital Systems using Verilog and VHDL

Design and Implementation (in VHDL) of a VGA Display and Light Sensor to run on the Nexys A7 Board

by

Vladimir Vakhter

Laboratory Report

10/07/2019

Dr. Yarkin Doroz
ECE Department
Professor

Adhitya Athyur Ganesh
RBE Program
Tutor

Table of Contents

1. Introduction

This project [1] is aimed to design, implement and verify the following interfaces:

- *Seven-segment Display Interface* that allows demonstrating the input 16-bits binary number in the hexadecimal format from “0000” to “FFFF” on the four seven-segment indicators.
- *VGA Display Monitor Interface* that allows demonstrating on the VGA monitor six different graphical patterns selected by three slider-switches.
- *Light Sensor SPI Interface* that allows reading the 8-bits digital light sensor data from Digilent Pmod ALS Module [2] and displaying this data in the decimal format on the two seven-segment indicators.

Both *VGA Display Monitor Interface* and *Light Sensor SPI Interface* utilize *Seven-segment Display Interface* as a component of their architectures.

Both *VGA Display Monitor Interface* and *Light Sensor SPI Interface* utilize the Xilinx IP-core Generator for the Mixed-Mode Clock Manager (MMCM) that is used to synthesize the clock signal of 25MHz from the system clock signal of 100MHz.

VGA Display Monitor Interface uses the existing IP-core for Digilent VGA Controller [3].

2. Methods

In this chapter, a general overview of the approaches to solution will be provided.

This project consists of multiple independent parts. Therefore, initially multiple small specialized projects were created, tested and debugged. Then, these projects were combined to solve the tasks stated at [1].

2.1. Seven-segment Display Interface

Seven-segment Display Interface controls the anodes and the cathodes of a seven-segment LED display in order to illuminate a specific pattern. More information on the operation of the seven-segment LED display may be found at [4].

Seven-segment Display Interface is built with the use of two entities:

1. decoder (Fig. 1);
2. seven_seg_4 (Fig. 2).

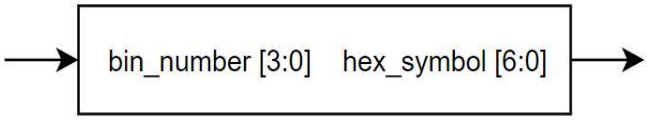

Fig. 1 – decoder entity diagram: bin_number – 4-bits binary value,
hex_symbol – hexadecimal representation of bin_number on the seven-segment display

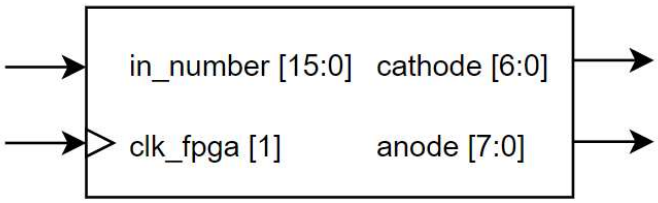

Fig. 2 – seven_seg_4 entity diagram: in_number – 16-bits binary value to be displayed,
clk_fpga – system clock of 100MHz, cathode – 7-bits output bus controlling the cathodes,
anode - 8-bits output bus controlling the anodes

The module *decoder* converts a binary number to the hexadecimal symbol to be displayed on a seven-segment indicator. Each hexadecimal symbol is a special cathode pattern designed in conformity with the Nexys A7 Reference Manual [4] as it is shown on the state diagram (Fig. 3).

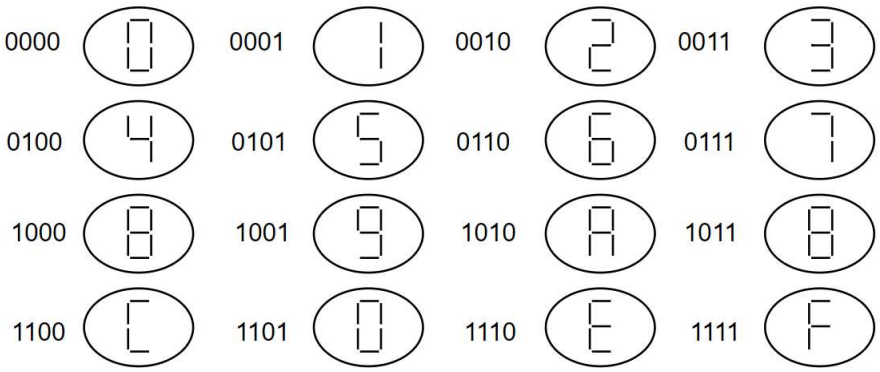

Fig. 3 – *decoder* entity state diagram

(on the left – input binary numbers, on the right – cathode patterns)

The module *seven_seg_4* represents the 4-digit seven-segment display. Display update rate should be higher than the human eye's reaction (around 45Hz) [4]. In the implemented design, the update rate of 95.4Hz was obtained from the FPGA clock of 100MHz with the use of the divider $1/2^{20}$ (the 20-bits counter). So, each of the four digits was illuminated during 1 / (4 · 95.4Hz) = 2.5ms.

seven_seg_4 entity includes *decoder* entity in the form of four elements (Fig. 4).

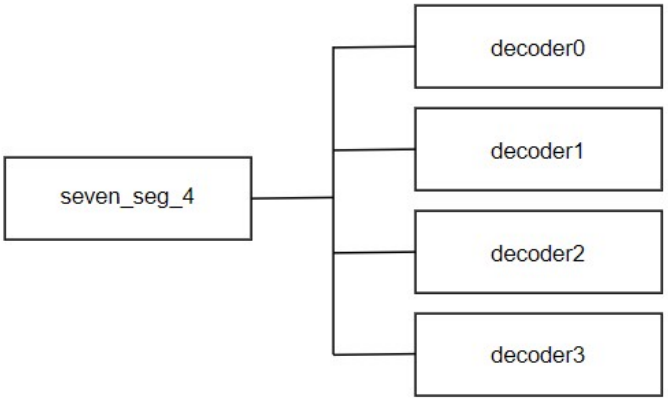

Fig. 4 – Seven-segment Display Interface Components' hierarchy

For the purpose of tests, the slider-switches were used to enter various numbers in the range from "0000" to "FFFF" and display them on the seven-segment display.

2.2. VGA Display Monitor Interface

VGA Display Monitor Interface controls the VGA port of the Nexys A7 board [4].

VGA Display Monitor Interface is built with the use of four entities:

1. vga_display (Fig. 5);
2. clk_manager (MMCM) (Fig. 6);
3. seven_seg_4 (Fig. 2);
4. vga_controller_640_60 (Fig. 7).

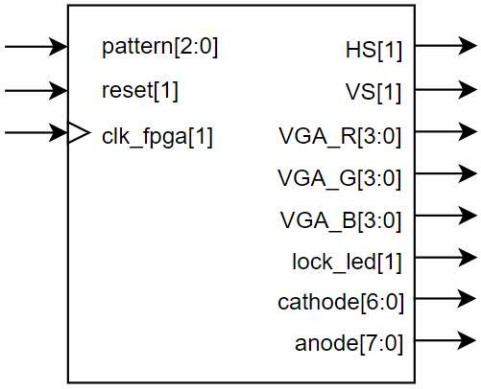

Fig. 5 - vga_display entity diagram:

pattern – 3-bits input that defines which pattern will be displayed on the screen;

rst – reset input, clk_fpga – system clock of 100MHz, HS/VS – horizontal/vertical synch pulse,

VGA_R/VGA_G/VGA_B – 4-bits red/green/blue-color component of VGA,

lock_led – output that is active when the clock output of MMCM is valid,

cathode – 7-bits vector controlling cathodes, anode - 8-bits vector controlling anodes

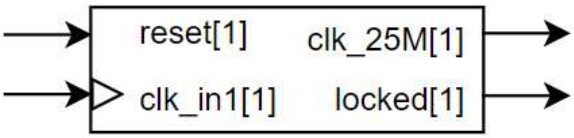

Fig. 6 - clk_manager entity diagram:

reset – reset input, clk_in1 – system clock of 100MHz input, clk_25M – output of 25MHz,

locked – output that is active when clk_25M is valid

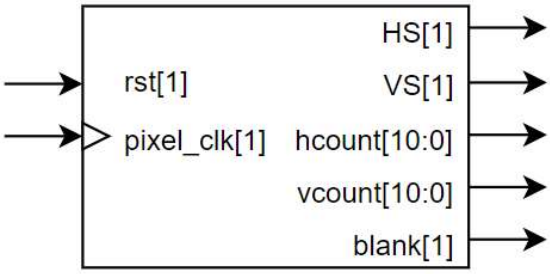

Fig. 7 - vga_controller_640_60 entity diagram:

rst – reset input, pixel_clk – input for 25MHz clock generated by MMCM,

HS/VS – horizontal/vertical synch pulse, hcount/vcount – horizontal/vertical count (position)

of the currently displayed pixel, blank – output which is active when pixel is not in visible area

vga_display entity includes *clk_manager*, *seven_seg_4* and *vga_controller_640_60* entities in the form of three elements (Fig. 8).

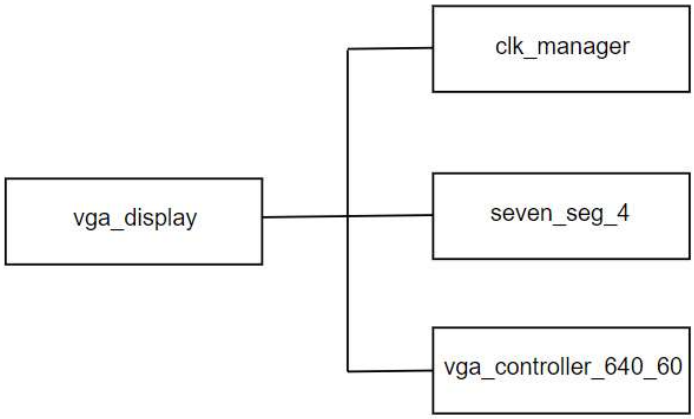

Fig. 8 - VGA Display Interface Components' hierarchy

clk_manager is an instance of the MMCM IP core [5].

vga_controller_640_60 is an instance of the existing IP-core for Digilent VGA Controller. This module generates the video synch pulses for the monitor to enter 640x480@60Hz resolution state. It also provides horizontal and vertical counters for the currently displayed pixel and a blank signal that is active when the pixel is not inside the visible screen and the color outputs should be reset to zero [3].

vga_display shows different patterns based on the position of the 3 slider-switches - J15 (the least significant bit – LSB), L16, M13 (the most significant bit – MSB):

- "000" – completely red screen;
- "001" – blue and yellow vertical bars each 32 pixels wide;

For the next combinations of the slider switches, it is assumed that the VGA screen is divided into blocks 24 pixels high and 32 pixels wide. Overall, 20 x 20 blocks. (x,y) position of a block is equal to (x,y) position of its left upper corner. Two out of four seven-segment indicators are used to display the current x-position of the block in decimal (00 to 19):

- “010” – a white colored block placed at a starting position of (2, 8);
- “011” – a red colored block placed at a starting position of (11, 11);
- “100” – a yellow colored block placed at a starting position of (14, 3);
- “101” – a blue colored block placed at a starting position of (19, 18).

The state diagram of *vga_display* entity is presented on the Fig. 9.

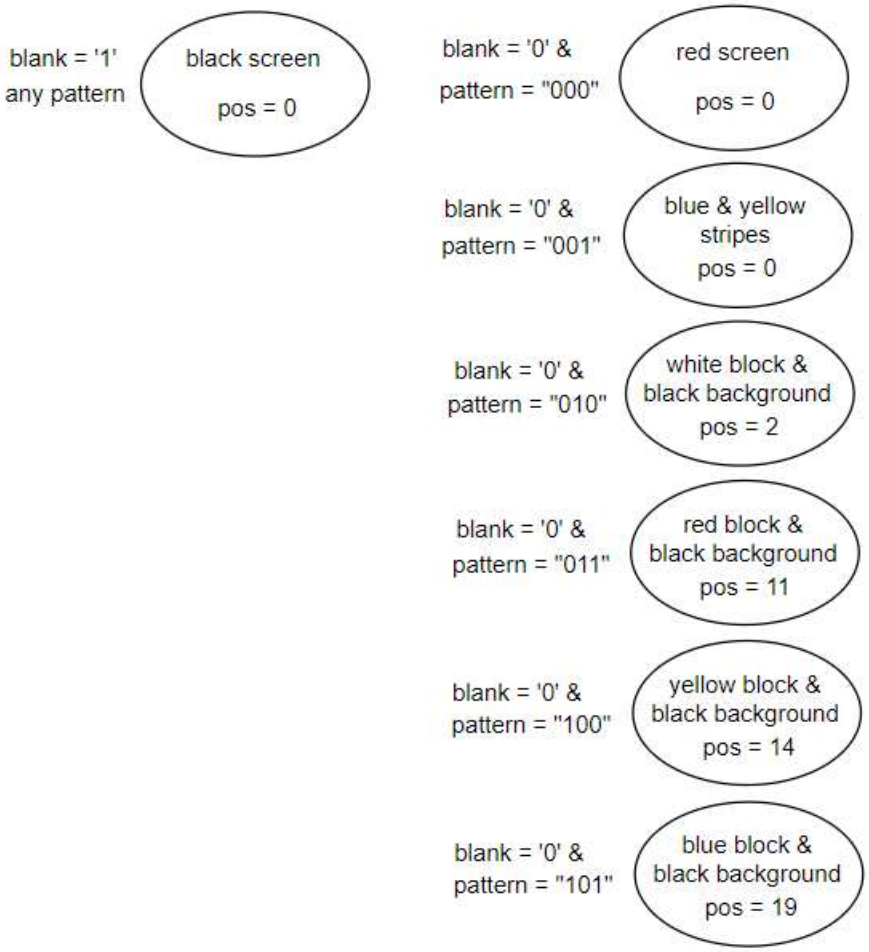

Fig. 9 – *vga_display* entity state diagram

(on the left – inputs, on the right – states, where pos – x-position of a block)

2.3. Light Sensor SPI Interface

Light Sensor SPI Interface is built to control the data exchange between the Nexys A7 board [4] and the PmodALS light sensor [6] utilizing Serial Peripheral Interface (SPI) [7].

Light Sensor SPI Interface is built with the use of three entities:

1. pmodals (Fig. 10);
2. clk_manager (MMCM) (Fig. 6);
3. seven_seg_4 (Fig. 2).

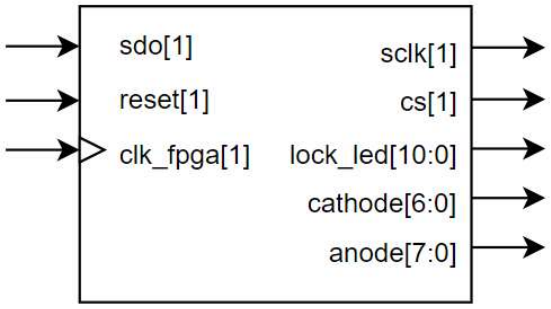

Fig. 10 – pmodals entity diagram:

sdo – digital data input from the sensor, reset – reset input, clk_fpga – system clock of 100MHz,

sclk - digital clock output (1MHz), cs - chip selected output (10Hz, active low),

lock_led – output that is active when the clock output of MMCM is valid,

cathode – 7-bits vector controlling cathodes, anode - 8-bits vector controlling anodes

pmodals entity includes *clk_manager* and *seven_seg_4* in the form of two elements (Fig. 11).

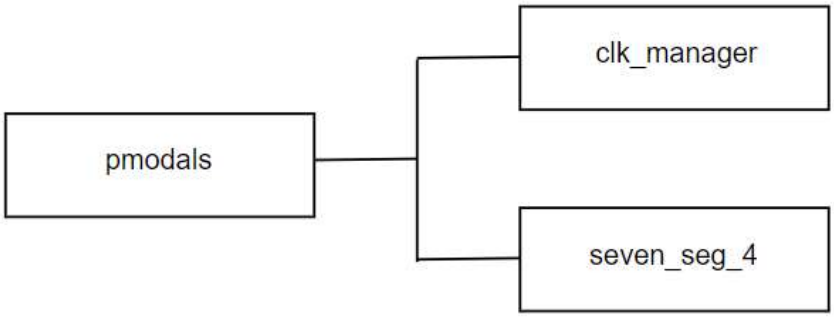

Fig. 11 - Light Sensor SPI Interface Components' hierarchy

3. Oscillograms

In this chapter, the waveforms of CS, SDO and SCLK signals for *Light Sensor SPI Interface* will be shown.

The data from the light sensor is presented as a set of 15 bits where the first 3 bits are leading zeros, the bits 4-11 are the light level with the MSB first, and the last 4 bits are trailing zeros. However, 16 SCLK (clock) pulses are associated with one operation of the Digilent Pmod ALS Module [2], [6].

In order to capture a new light sensor value (SDO signal), each 100ms (10Hz), CS signal was set low during 16 clock cycles of SCLK signal (1MHz) (Fig. 12, Fig. 13).

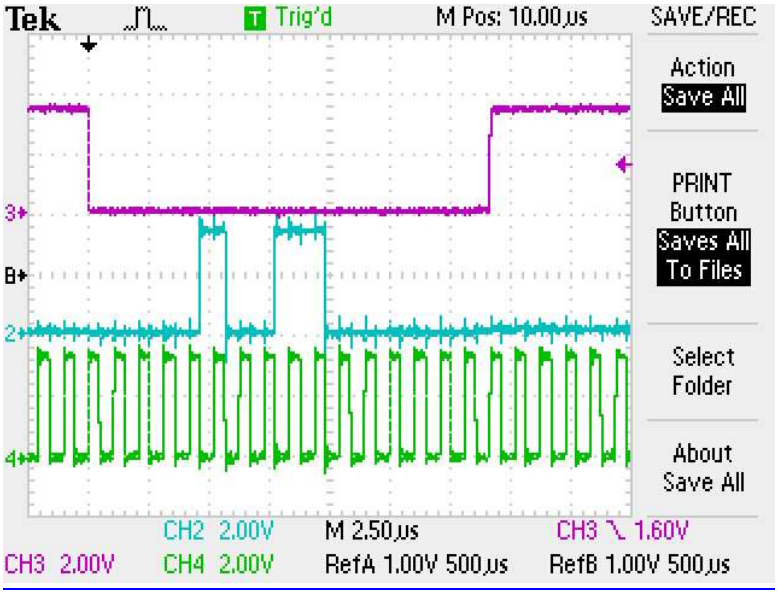

***Fig. 12** – SPI Transfer Oscillogram: top to bottom – CS, SDO, SCLK correspondingly (CS at 10Hz, SCLK at 1MHz, light sensor value = 0x4C = 0b01001100)*

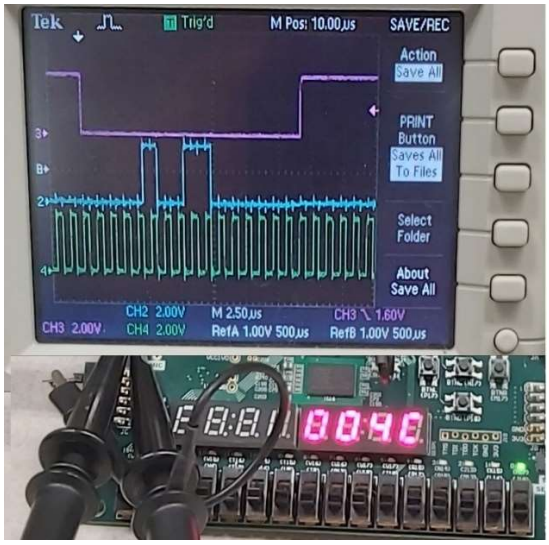

***Fig. 13** - SPI Transfer Oscillogram and the Value on the Seven Segment Display*

4. FPGA Resource Usage

In this chapter, the information on the FPGA resource usage for each of the interfaces will be provided.

4.1. Seven-segment Display Interface

Seven-segment Display Interface utilizes 0.07% of lookup tables (LUT), 0.02% of flip-flops (FF), and 15.24% of Input/Output Blocks (IO) (Fig. 14).

Resource	Utilization	Available	Utilization %
LUT	43	63400	0.07
FF	20	126800	0.02
IO	32	210	15.24

Fig. 14 – Utilization of resources by Seven-segment Display Interface

4.2. VGA Display Monitor Interface

VGA Display Monitor Interface utilizes 0.09% of LUT, 0.04% of FF, 16.67% of IO, and 16.67% of MMCM (**Fig. *15***).

Resource	Utilization	Available	Utilization %
LUT	60	63400	0.09
FF	45	126800	0.04
IO	35	210	16.67
MMCM	1	6	16.67

Fig. 15 – Use of resources by VGA Display Monitor Interface

4.3. Light Sensor SPI Interface

VGA Display Monitor Interface utilizes 0.07% of LUT, 0.05% of FF, 10.00% of IO, and 16.67% of MMCM (**Fig. *16***).

Resource	Utilization	Available	Utilization %
LUT	45	63400	0.07
FF	63	126800	0.05
IO	21	210	10.00
MMCM	1	6	16.67

Fig. 16 – Use of resources by Light Sensor SPI Interface

5. Warning messages

In this chapter, the information on the warning messages for each of the interfaces will be provided.

For all the interfaces, the configuration bank voltage select (CFGBVS) was set equal to VCCO and the configuration voltage (CONFIG_VOLTAGE) was set equal to 3.3V by adding the following lines to the constraints (.xdc) files:

- set_property CFGBVS VCCO [current_design]
- set_property CONFIG_VOLTAGE 3.3 [current_design]

It allowed avoiding the warning "[DRC CFGBVS-1] Missing CFGBVS and CONFIG_VOLTAGE".

5.1 Seven-segment Display Interface

The following four warning messages appeared during the synthesis of *Seven-segment Display Interface*:

- [Synth 8-3917] design seven_seg_4 has ports anode[7] driven by constant 1
- [Synth 8-3917] design seven_seg_4 has ports anode[6] driven by constant 1
- [Synth 8-3917] design seven_seg_4 has ports anode[5] driven by constant 1
- [Synth 8-3917] design seven_seg_4 has ports anode[4] driven by constant 1

These warnings show that the anodes from 7 down to 4 are constantly driven high. It was made deliberately, because in the design only four out of eight seven-segments indicators were used. According to [4], the anodes are active-low. Hence, in order to switch the elder four digits off, these outputs were tied to the high voltage level.

5.2. VGA Display Monitor Interface

The design inherited the warnings for *Seven-segment Display Interface.*

5.3. Light Sensor SPI Interface

The design inherited the warnings for *Seven-segment Display Interface.*

6. Conclusions

During this project, three interfaces were developed and tested:

1. *Seven-segment Display Interface*;
2. *VGA Display Monitor Interface;*
3. *Light Sensor SPI Interface.*

The design of these interfaces demanded the application of a broad range of approaches:

- Decomposition of a complex project into multiple smaller independent parts in order to easier design, test, debug, and reuse these specified modules;
- Utilization of IP cores in order to accelerate the development cycle;

- Implementation of standard elements such as counters, decoders, and shift registers.

A few problems faced in implementation:

- An error during the synthesis of the MMCM IP core. This problem was resolved by the explicit setting of special attributes inside the architecture definition which told the synthesis tool that this IP core was used as a black box (see, for instance, the code for VGA Display Monitor Interface in the **Error! Reference source not found.**.
- One latch was inferred by the design. This issue was resolved by assigning default values for all the outputs.

The following lessons learned from the project:

- All the necessary ports of the underlying elements should be mapped to the top hierarchical module.
- One should pay thorough attention to the warning messages (for instance, “[Synth 8-327] inferring latch for variable…”, or “[Synth 8-614] signal 'X' is read in the process but is not in the sensitivity list”).
- The sensitivity list of a process should include all the signals tracked in this process.
- All outputs should have default values in order to avoid latches in combinational logic.
- Special attributes should be explicitly set inside the architecture definition when the MMCM IP core is used, because it is a black box for the synthesis tool.

Also, the basics of SPI Protocol were learned and applied.

As an organizational improvement, it could be suggested to use private repositories (for example, on GitHub [8]) to store the source code, because when directly copying, some indents are lost. Also, some tools that allow generating documentation from source code (e.g., Doxygen [9]) could be learned and applied.

References

[1] D. Y. Doroz, "Design and implementation (in VHDL) of a VGA Display and Light Sensor to run on the Nexys4DDR board," [Online]. Available: http://users.wpi.edu/~ydoroz/ece574/assignments/ece574_2019_VGA_Light_Sensor.pdf. [Accessed 7 Oct. 2019].

[2] "Pmod ALS: Ambient Light Sensor," Digilent, [Online]. Available: https://store.digilentinc.com/pmod-als-ambient-light-sensor/. [Accessed 7 Oct. 2019].

[3] "VGA Controller," Digilent, [Online]. Available: http://users.wpi.edu/~ydoroz/ece574/tutorials/vga_controller_640_60.vhd. [Accessed 7 Oct. 2019].

[4] "Nexys A7 FPGA Board Reference Manual," Digilent, [Online]. Available: https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual. [Accessed 7 Oct. 2019].

[5] J. Duckworth, "MMCM Tutorial," [Online]. Available: http://users.wpi.edu/~ydoroz/ece574/tutorials/MMCM%20Vivado%20example%20VHDL.pdf. [Accessed 7 Oct. 2019].

[6] "PmodALS User Guide," Digilent, [Online]. Available: https://reference.digilentinc.com/pmod/pmod/als/user_guide. [Accessed 7 Oct. 2019].

[7] "SPI," Digilent, [Online]. Available: https://reference.digilentinc.com/learn/fundamentals/communication-protocols/spi/start?redirect=1. [Accessed 7 Oct. 2019].

[8] "GitHub," [Online]. Available: https://github.com/.

[9] "Doxygen," [Online]. Available: http://www.doxygen.nl/.

[10] "Master XDC Files," Digilent, [Online]. Available: https://github.com/Digilent/digilent-xdc/. [Accessed 7 Oct. 2019].

[11] "Nexys A7 Schematic," Digilent, [Online]. Available: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7-sch.pdf. [Accessed 7 Oct. 2019].

[12] "Nexys-A7-100T Out-of-Box Demo," Digilent, [Online]. Available: https://github.com/Digilent/Nexys-A7-100T-OOB. [Accessed 7 Oct. 2019].

[13] "PmodALS data sheet," [Online]. Available: http://users.wpi.edu/~ydoroz/ece574/tutorials/PmodALS_data.pdf. [Accessed 7 Oct. 2019].