Department of Electric & Electronic Engineering,
Boğaziçi University

EE244 Digital System Design Final Project

MAKE CHAMELEON HAPPY AGAIN

Kerem YURTSEVEN 2020401174
Emirhan ARIKAN 2020401045

Project Advisor : Şenol MUTLU

06/06/2023

Table of Contents

1 INTRODUCTION

In this report, “Make Chameleon Happy Again” project and its modules are explained. One can find the problem deriving the fulcra of the project and the story behind it. Also, main implementation and design utilities are shared under the related sections. The report includes the inferences obtained through the project’s preparation process as well as the conclusion, which is the project’s final results.

2 PROBLEM STATEMENT

Once upon a time, there existed a chameleon named “Digito” always having a smiling face and travelling across the long seas and lands to find divine knowledge. Through its journey, Digito encountered the university and decided to become an electrical and electronics engineer. Yet, this was a controversial choice since Digito’s smile lost after this decision. After the university era of Digito, it was known that Digito only smiles when it disappeared in colors so that it can hide from other engineers. In this project, the problem is to equalize the color of chameleon to background color and make chameleon happy again. The background color is changing every time the color is matched, so the user needs match the new color.

3 RELATED BACKGROUND

The color of the chameleon is controlled via a switch on the FPGA and transmitted to a 620px*480px monitor with 25 MHz pixel clock through a VGA cable. The user can use a push-button to check it and skip to the next level when the colors are matched. Background colors are attained due to a pseudo-random number generator when the colors are the same. VGA cable has the following ports: horizontal synchronizer, vertical synchronizer, 3 bytes for red, 3 bytes for green, and 2 bytes for blue. It draws the chameleon based on vertical and horizontal positions.

4 DESIGN

The project has the design as shown in Figure 1. It consists of 4 input ports, and 5 output ports. Inputs: Clock, Reset, Switch, NextLevel. Outputs: VSync, HSync, Red(2:0), Green(2:0), Blue(2:0).

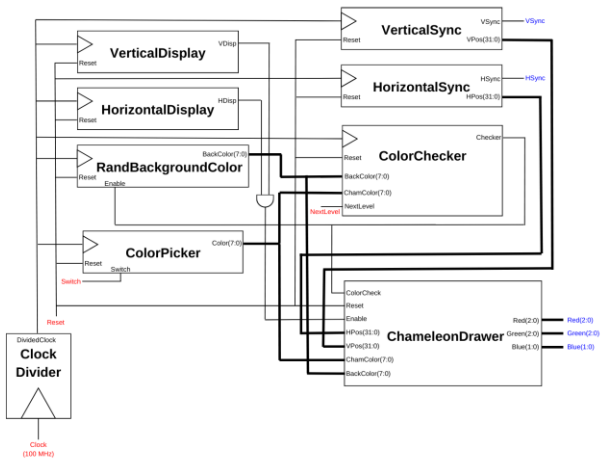

Figure 1: Design of the Project

4.1 Definitions, Acronyms and Abbreviations

Clk: Clock
VSync: Vertical Synchronization
VPos: Vertical Position
VDisp: Vertical Displayer
HSync: Horizontal Synchronization
HPos: Horizontal Position
HDisp: Horizontal Displayer
E: Enable
BackColor: Background Color
ChamColor: Chameleon Color
PRNG: Pseudo-Random Number Generator
Rand: Random

4.2 ClockDivider

Input Port: Clock
Output Port: DividedClock

ClockDivider take the Clock input and divide it by four. Since the FPGA's system clock is 100 MHz and the monitor is drawn by a 25 MHz pixel clock, the input should be divided by four. To achieve this, an integer counter signal, initially 1, is defined and inversed when it is 2.

4.3 VerticalSync

Input Ports: Clk, Reset
Output Port: VSync, VPos

VerticalSync is used to synchronize the monitor vertically. It works for a 620px*480px monitor with 25 MHz pixel clock, and has:

Sync Pulse Time: 16.7 milliseconds and 521 lines
Display Time: 15.36 milliseconds and 480 lines
Pulse Width: 64 microseconds and 2 lines
Front Porch: 320 microseconds and 10 lines
Back Porch: 928 microseconds and 29 lines

Similar to ClockDivider, there is a counter signal to count where it is on a loop and decide the low and high situation of the signal. VSync is low during just pulse width and high otherwise. VPos describes the Y coordinate of the monitor and is obtained by decrementing 31 after dividing by 800, that is because there is a time interval before the display time and clock is between 0 and 416.800, while line is between 0 and 521.

4.4 VerticalDisplay

Input Ports: Clk, Reset
Output Port: VDisp

VerticalDisplayer is coded in a similar manner to VerticalSync, yet it is high just during the Display Time, and low otherwise.

4.5 HorizontalSync

Input Ports: Clk, Reset
Output Ports: HSync, HPos

HorizontalSync is used to synchronize the monitor vertically. It works for a 620px*480px monitor with 25 MHz pixel clock, and has:

Sync Pulse Time: 32 microseconds and 800 clocks
Display Time: 25.6 microseconds and 640 clocks
Pulse Width: 64 microseconds and 96 lines
Front Porch: 320 microseconds and 16 lines
Back Porch: 928 microseconds and 48 lines

Similar to ClockDivider, there is a counter signal to count where it is on a loop and decide the low and high situation of the signal. HSync is low during just pulse width and high otherwise. HPos describes the X coordinate of the monitor and is obtained by decrementing 144, that is because there is a time interval before the display time.

4.6 HorizontalDisplay

Input Ports: Clk, Reset
Output Port: HDisp

HorizontalDisplayer is coded in a similar manner to HorizontalSync, yet it is high just during the Display Time, and low otherwise.

4.7 RandBackgroundColor

Input Ports: Clock, E, Reset
Output Port: BackColor

RandBackgroundColor operates as a background color decider in the project. When E is high, which means chameleon color and background color are same, a new pseudo-random number is generated, and a new color is chosen.

Colors are stored in a 16 unit and 4 bytes ROM, and a color is picked with address, where the address is equal to pseudo-random number generated by PRNG.

4.7.1 PRNG

Input Ports: E, Reset
Output Port: Rand

PRNG is a 4-byte pseudo-random number generator working with the principle of linear-feedback shift register. It indeed does not generate random numbers but act as counter with a predefined order. It takes the first and the last bits of the signal and XOR them. After that, all bits of the signal are left-shifted.

4.8 ColorPicker

Input Ports: Clock, Reset, Switch
Output Port: Color

Color picker is used to change the color of chameleon. There exists a 16 unit and 4-byte ROM in the design to store color options and each switch event picks the next color in addresses.

4.9 ColorChecker

Input Ports: Clock, ChamColor, BackColor, NextLevel
Output Port: Checker

ColorChecker is designed as a logical comparer to check that whether color of the background, BackColor, and color of the chameleon, ChamColor, are the same. When NextLevel button is pressed. ColorChecker operates and checks the colors. If they are the same, then Checker is high for 0.3 seconds, which is controlled by another counter.

4.10 ChameleonDrawer

Input Ports: ColorCheck, Reset, En, HPos, VPos, ChamColor, BackColor
Output Ports: Red, Green, Blue

ChameleonDrawer decides the color of each pixel on the monitor based on the coordinates. HPos and VPos are the X and Y coordinates, and ChamColor is the color of the chameleon, while BackColor is the color of the background.

En decides whether it is the time to display or not based on vertical and horizontal displayers. If both vertical and horizontal displayers are high, then En is also high. Moreover, ColorCheck is used to decide the happiness of the chameleon. If ColorCheck is high then smiling face is drawn to the monitor.

X and Y coordinates and their corresponding colors are given in Figure 2.

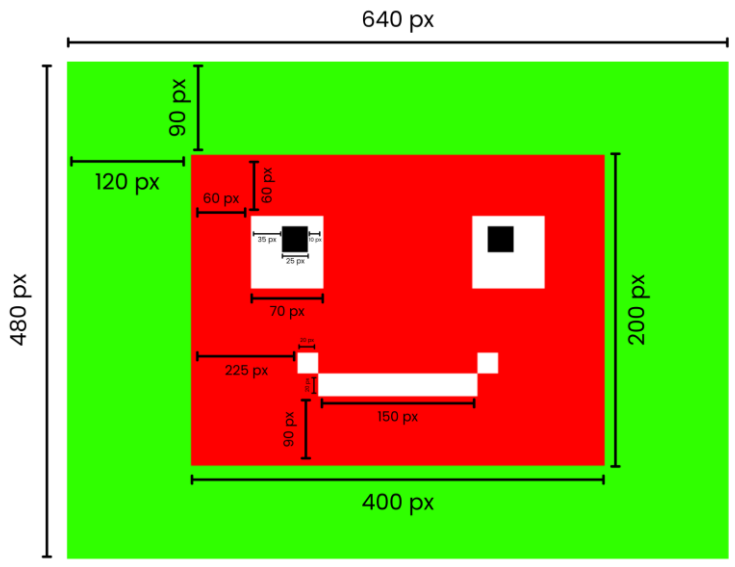

Figure 2: Scale of Drawings

5 RESULTS

As can be seen in Figure 3, the program starts with the different colors for the background and chameleon. After using the switch, the color of the chameleon changes as intended (Figure 4). When the color is matched, the user presses the push button and the chameleon smiles then the background color is changed (Figure 5). After a delay of 0.3 seconds, smile of the chameleon fades away because chameleon and the background color are no longer the same (Figure 6).

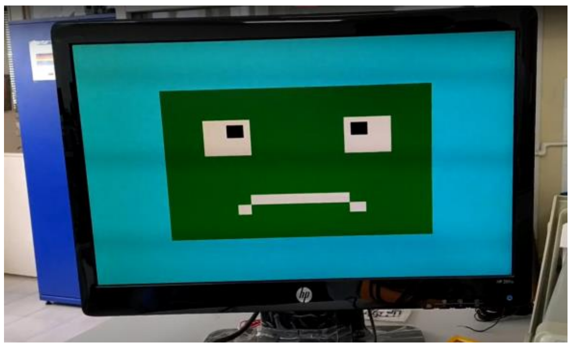
Figure 3: Different Colors

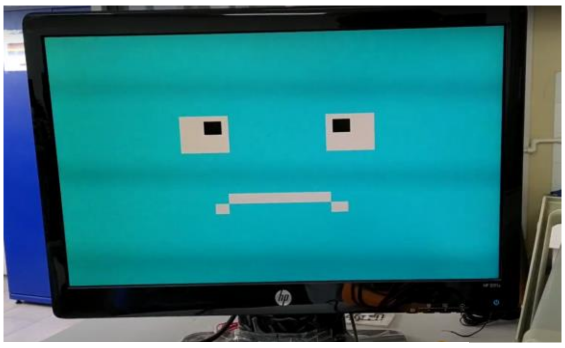
Figure 4: Same Colors

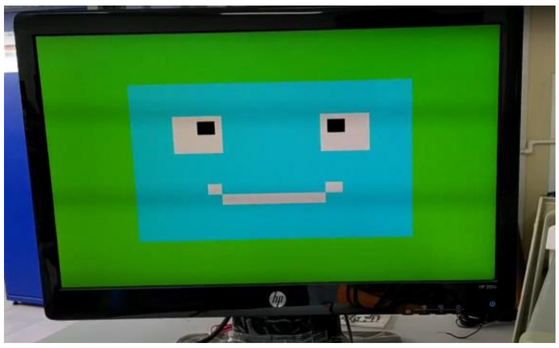
Figure 5: NextLevel has pressed, background color has changed

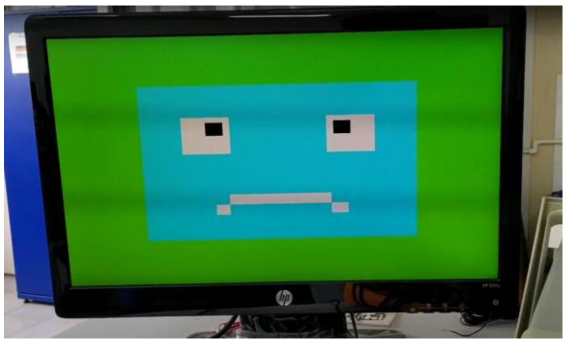
Figure 6: 0.3 seconds delay passed and smile disappears

6 CONCLUSION

At the end of the project, the purpose is achieved, and all desired actions are actualized. By doing so, new competencies like VGA controller, gamification through FPGA, combinational logic in digital design, color visualization on a monitor, and user-interactive real time environments are experienced.

Problems Encountered: A problem we encountered was with ColorChecker. At one point ColorChecker was comparing the background and the chameleons' color so fast that user couldn't see the moment when colors are matched. This problem was probably caused because of asynchronous clocks. So, we added a NextLevel button. This button helped us to have a more user-friendly game while keeping the game's basic purpose.

Another problem was that when colors were same and ColorChecker gave high output, background color changed infinitely times since it had high active enable. In order to overcome this problem, background color was changed at the each rising edge of enable instead of just checking whether the enable is high or not.

Future Work: To achieve a more complex game, this project could be improved by having much more color-changing options. For example, by randomly assigning every bit of the 9-bit color variable, every 256 colors could be obtained. For the chameleons' color, three distinct switches could be assigned for the red, green, and blue colors. While one of the color switches is on, user could use the push buttons for changing the grading of that color. Thus, chameleons' color can vary between 256 colors too.

Another future work is that since this version of the game does not punish the user for choosing the wrong color for the chameleon, a point scoring system can be implemented in the game, where the points are displayed on the seven-segment display.

7 REFERENCES

1. (2021). Retrieved from https://support.xilinx.com/s/article/35078?language=en_US
2. (2022). Retrieved from https://fpgaer.wordpress.com/push-button-debouncer/
3. MISHRA, S., & DAS, M. (2010). Retrieved from https://core.ac.uk/download/53187567.pdf
4. Skycanny, & nand_gates. (2011). Retrieved from https://www.edaboard.com/threads/how-can-i-describe-a-rom-in-vhdl.38052/
5. *VGA driver for FPGA in VHDL*. (2016). *YouTube*. YouTube. Retrieved from https://www.youtube.com/watch?v=eJMYVLPX0no
6. www.fpgakey.com, F. |. (n.d.). Retrieved from https://www.fpgakey.com/tutorial/section589

APPENDIX

A READ ME

An important note for the .ucf file, where the input/output – pin combinations are declared. Since Switch and NextLevel inputs are used with rising_edge, design is interpreted as inappropriate usage of clock. In order to clarify these signals can intentionally be used as clock, following lines are added to the .ucf file. Due to these enforcement, color of the chameleon may not change at very fast inputs

```
NET "Switch"    CLOCK_DEDICATED_ROUTE = TRUE;
NET "NextLevel" CLOCK_DEDICATED_ROUTE = TRUE;
```

B USERS' MANUAL

The color of the chameleon is controlled by the T5 switch and next level action can be done by using A8 push button on the FPGA. To change the color of the chameleon, the user needs to use T5 switch. The rising edge conditions of the switch is checked by the system so that the user can change the color with positive pulse. After the user is confident that the colors are matched, one might use A8 push button to control one's choice of color. If the colors are the same, the chameleon slightly smiles, and a new random background color will be assigned. To reset the system, use switch T10.