Project 2 Report

The Frog Flap (Game Emulator)

By

Thomas Zoldowski and Izik Janusz

EGR 426 – Integrated Circuit Systems Design

Section #10

Date Submitted: March 20, 2025

Instructor: Dr. Parikh

1.0 Objective

The objective of this project was to design and implement an FPGA-based system on the Spartan-7 board that produced HDMI signals with an LFSR for ‘random’ image generation. The input sources use the onboard buttons and produce a 640x480 hdmi signal output that works on any typical monitor.

2.0 Introduction

Project 2 was essentially an emulation of the Frog Flapper game from the Wario Party Games. The buttons scroll the background where images are pasted on top using a merger module. The characters use a sprite map to create animation while allowing the background is moving.

3.0 Equipment

Table 1 below outlines the equipment used for this laboratory experiment:

Table 1. Equipment/ Apparatus

FPGA Board	
Spartan-7 xc7s50csga324-1	1
Components	
Micro-USB Cable	1
Software	
Vivado 2023.3	

4.0 System Overview

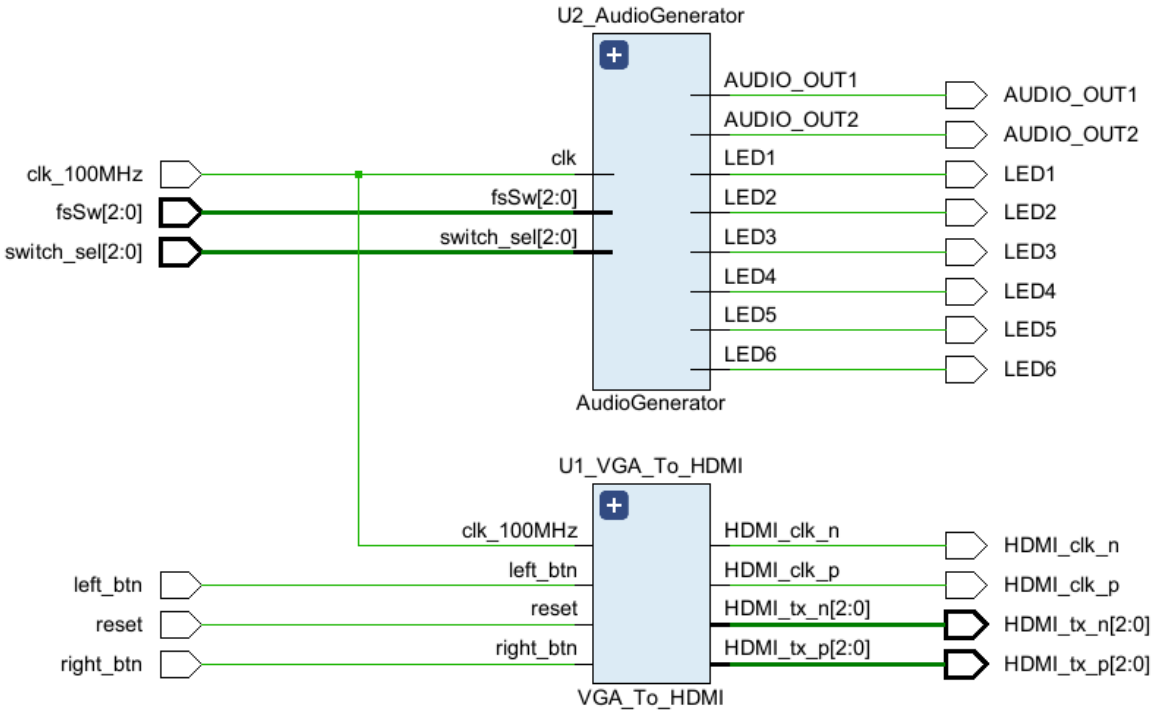

Figure 1. Top-Level Schematic

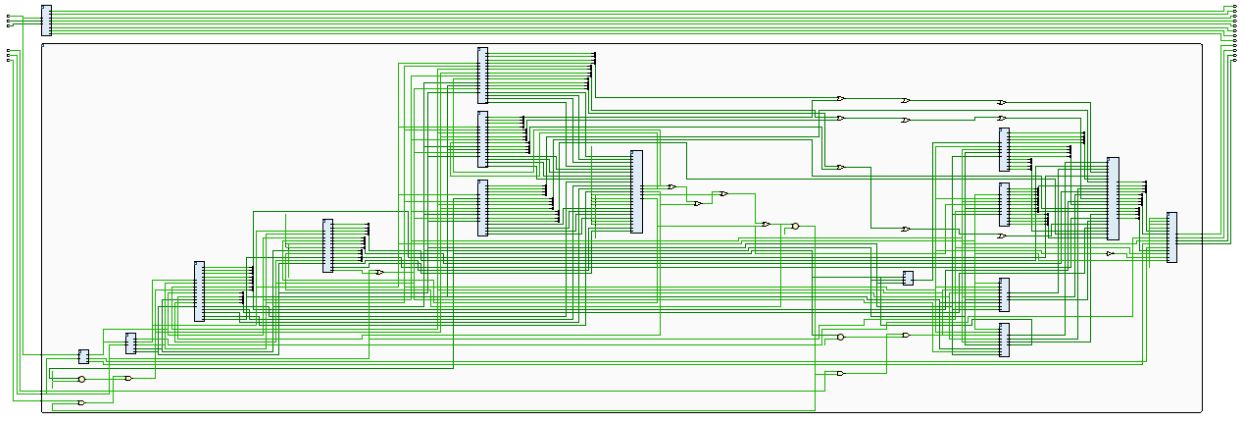

Figure 2. VGA_To_HDMI Schematic

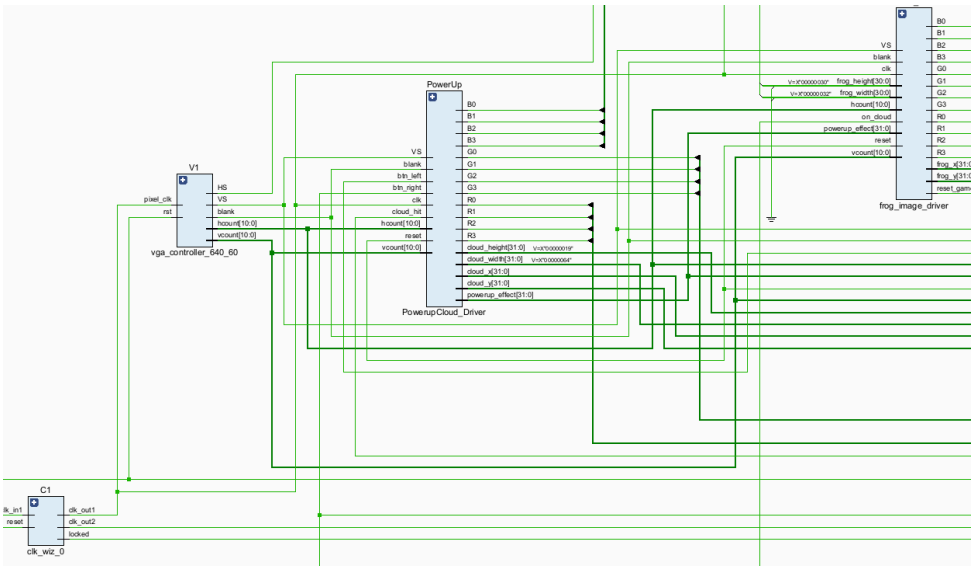

Figure 3. VGA_To_HDMI Schematic zoomed 1

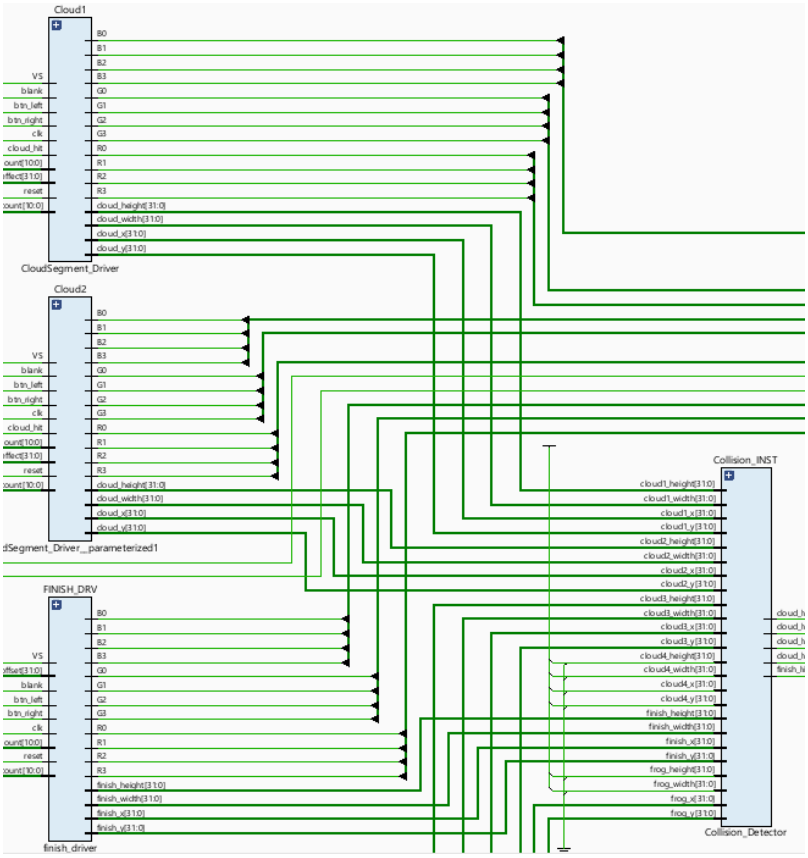

Figure 4. VGA_To_HDMI Schematic zoomed 2

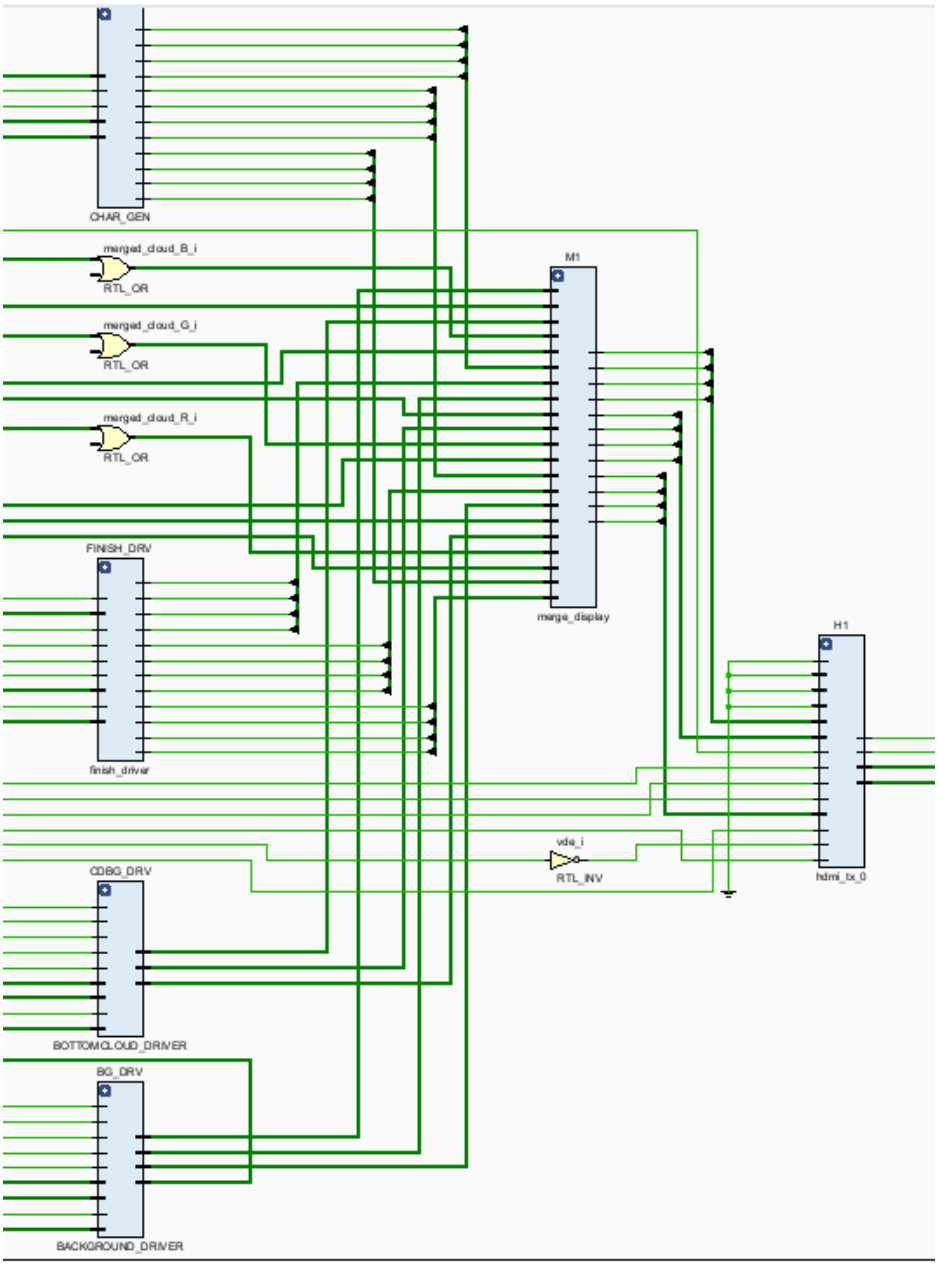

Figure 5. VGA_To_HDMI Schematic zoomed 3

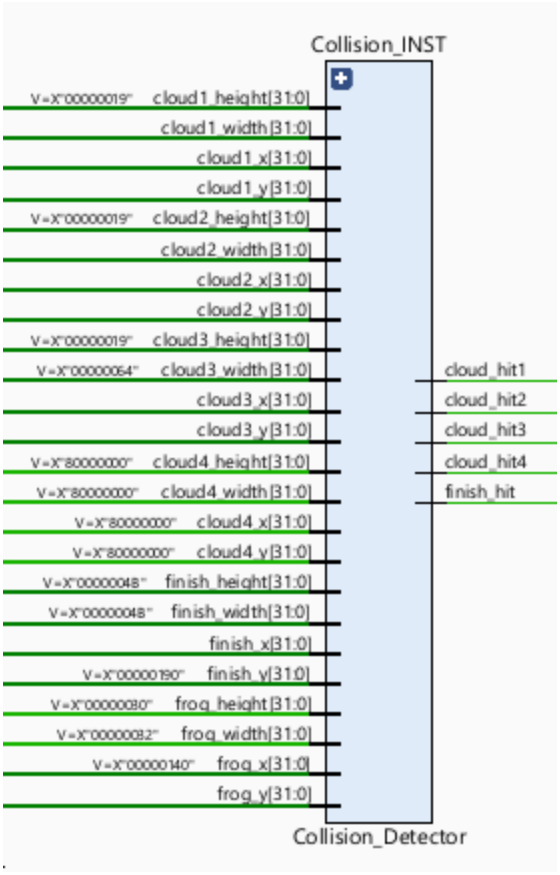

Figure 6. VGA_To_HDMI Schematic zoomed 4

5.0 Module Descriptions

5.1 Background Driver Module

A ROM holds the background data in a .coe file with a size of 400x60. This image is much smaller than the screen which uses a lot less resources than storing a full 640x480 image. Each pixel has 12 bits associated with it as well so the image file eats lots of resources if given the opportunity. The driver then scales the image by 8 to reach the correct height to fill the entire screen by repeating inputs. This is called upscaling and we decided to do it for the background only since it was so large. There is a tradeoff as the screen becomes a little more blurry but that is okay since the game sprites will be sharp and it will cause the player to focus on the sprites instead of the background. The driver outputs the pixel data using the ROM but it also scrolls using the button inputs to shift the background.

5.2 VGA Controller Module

The VGA module was essentially an IP from Dr Parikh that outputs the Hysnc, and Vsync lines needed to populate the HDMI IP so it can generate the HDMI signal correctly.

5.3 Clock Wizard IP

The clock wizard IP makes clock outputs from the resources on board to output the requested clock speeds. The clocks are used to drive the other modules.

5.4 HDMI IP(Real Digital)

The HDMI module takes the VGA video signals from the VGA module and the pixel data then puts them onto a dedicated HDMI driver IC. The IC then sends the HDMI equivalent out to the monitor. The timing all comes from the vga module but the pixel data comes from user defined modules.

5.5 Block ROMs (Vivado IP)

The Block Roms are held in the .coe files, which hold information containing both video pixel values and audio signals. In order to reduce ROM usage some images were upscaled like the background.

5.6 Cloud Driver Modules

The cloud drivers are very similar to the background driver but include much more logic. The cloud generator uses an LSFR to make random clouds appear along with the movement. It also includes logic to handle collision for animation purposes. There are 3 sets of cloud drivers, one for each color which includes the logic for speeding up the game which are mostly the same but use a different image.

5.7 Character Driver Module

The character driver takes in the sprite maps held in ROM and plays them based on the button inputs and the states of the character's movement. This creates animated actions the character expresses while moving through the game.

5.8 Collision Module

The collision module takes in image hitbox data from the driver modules and outputs a 1/0 hit state which goes back into the driver modules for updates.

5.9 Text Module

The text module is a version of the given template that outputs the word score statically. The distance traveled then is shown on the score counter as the character goes further to the right.

5.10 Merger Module

The merger module takes in the pixel data from all drivers and using a priority scheme it determines what pixels should appear on top of one another. This usually follows the background is default and anything else goes on top of it.

5.11 Audio Module (Extra Credit)

After experimenting for a long time with both a Matlab-generated coe file and a PWM/PDM approach. The coe file was taking 10 36k bit Bram for a small sampling rate of 40khz. This quickly could take up more than half the resources if sampled too high. It also sounded mediocre and essentially was a poorly quantized PCM output. The way the audio now works is using a sin lookup table to generate the sin waves in a digital format without loss, the low notes however cannot be played due to the limitations of quantization of low frequencies so to offset that we added in variable sampling speed. Additionally, there is the option of swapping the music being played with switches 0/1/2/3 and changing the sample rate with 15/14/13/12 making 16 music tracks and 16 different sampling rates from 5k hz to 50khz.

5.12 Use of Animation, RNG, and Motion(Extra Credit?)

The character driver and cloud drivers contain logic to animate and move the objects. The character animation uses multiple frames on landing and moving vertically to create animation. The clouds themselves also have a LFSR that creates random movements and spawning.

6.0 Conclusion

The lab overall was a great experience learning how VGA signals work alongside how FPGA logic can be used to make games and pretty much make anything you want. The project itself showed how games in the past could’ve been made and what problems they encountered when making them.