System Design Document for an Audio Jukebox with Corresponding Video Visuals through HDMI

University at Buffalo, The State University at New York

EE478, Fall 2019

Written By	Bradley Golias		**Revision History**			
Engineer	Bradley Golias		**Rev**	**Date**	**Session**	**Approved/Grade**
Engineer	Siddhant Khera		Orig	12/10/19	Mon. 9AM	
Lecture Professor	Christopher Fritz					
Lab Professor	Farah Vandrevala					
University at Buffalo *The State University of New York*						**EE478F19 Final Project**

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

The objective of this final project was to choose a project and implement a hardware system using VHDL. We chose an audio Jukebox with at least 3 songs. For extra credit, we also added pause/stop functionality, 2 toggles to make the audio speed faster and slower. Next, we also added s visual output to the songs. The three screens displayed were a diamond for Twinkle Twinkle Little Star, a present with snow on the bottom for We Wish You a Merry Christmas, and a present with on the bottom with falling snow for Merry Christmas and a Happy New Year. Furthermore, we also implemented pause/stop for the visuals. Finally, we also made two types of pause screens: regular and flashing pause screens for the visuals. There is an included HDMI controller module that was display the shapes and input color in the RGB format. The audio output was achieved using the provided ssm2603 codec. The hardware design is discussed below and was successfully implemented onto the Zybo board.

1.2 Document Scope

This document serves as a record, documentation, and a proof our work. The main purpose of this document is to record and define our work that we did to complete the tasks assigned for the Final Project, Audio Jukebox with Video Visuals through HDMI, through Zybo Z7 Board, in $EE478$ in the Fall 2019 semester. The design covers the functions that were provided by the team of TAs and the professor(s), as well as the functions that we defined in the project. This document discusses the methods we used to setup inputs and outputs, and the implementation on a Zybo Z7 FPGA board.

1.3 Intended Audience

The expected and intended audience for this report is the $EE478$ session Teaching Assistant's, professor Farah Vandrevala, professor Christopher Fritz, along with Siddhant Khera and Bradley Golias in the lab session on Monday at 9am. Any other students part of $EE478$ during the Fall 2019 semester, to whom the material be relevant, are invited to view the document but should keep confidentiality and copyright etiquette in mind when doing so.

2 System Design Overview

2.1 System Block Diagram and Description

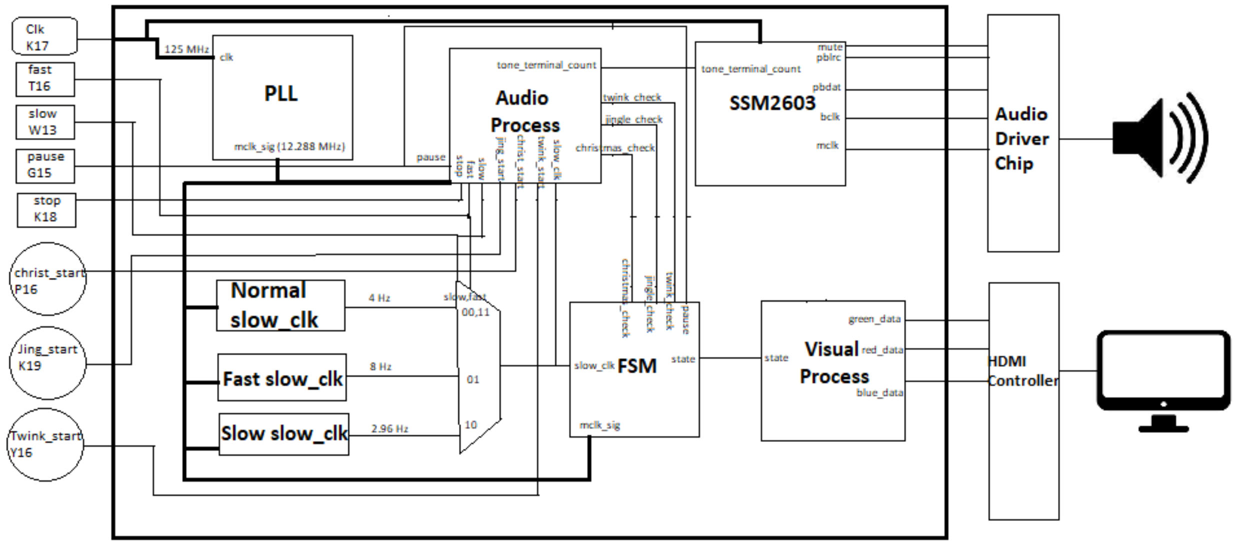

Figure 1: Jukebox Circuit Block Diagram

The Jukebox system uses 3 push-buttons each of which when selected will play a corresponding song and display visuals related to the song.

The jukebox system also includes four slider switches used as inputs. These switches can pause/stop the song and the visuals as well as speeding up or slowing down the song and the visuals. When both slider switches for fast and slow are high this enables dj mode which allows for the user to switch between songs at the place they last played. These operations will be controlled by a clock input that will be adjusted to the desired frequency through the use of a phase-locked loop.

The module will drive a pair of wired headphones to play the songs and a monitor to display the visuals. In depth looks at each block and overall functionality will be described below. An included HDMI controller was used to drive the RGB data on the screen.

2.2 Phase Locked Loop (PLL)

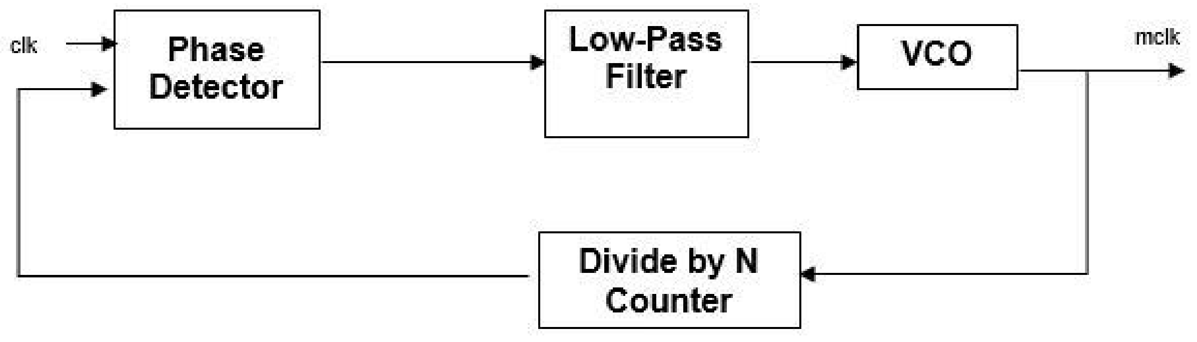

Figure 2: Phase-Locked Loop Block Diagram

Fig. 2 shows the phase-locked loop circuit used in the design of our jukebox. This circuit converts the 125 MHz clock signal of the board into a 12.288 MHz clock. This is done because the SSM2603 chip requires that clock frequency to operate correctly. The PLL was also used to create another clock signal with a frequency of 74.25 MHz for the HDMI controller.

2.3 Slow Clock Circuit Diagram

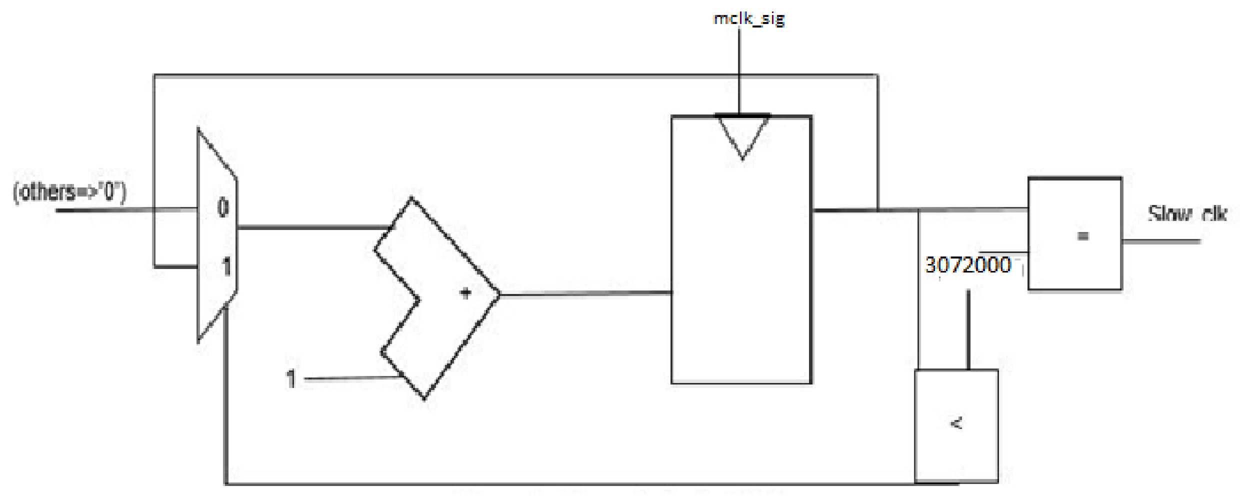

Figure 3: Slow Clock Circuit Diagram

The circuit diagram in Fig. 3 creates a slow clock with a 4 Hz frequency. This is accomplished by incrementing the counter by 1 and then comparing it with the count assigned.

When the counter is equivalent to the count a 1 bit is outputted and the counter is reset to 0. Thus creating a 4 hertz clock frequency. In our design we will use 2 additional slow clocks with different count values of 4,144,000 and 1,536,000 thus creating clocks of 2.96 Hz and 8 Hz.

These slow clocks will be selected using a MUX selected by the two input slider switches fast and slow. When only fast is selected the 8 Hz slow clock will be used to control the speed of the songs and visuals making it faster. When only slow is selected the 2.96 Hz slow clock will be used to control the speed of the songs and visuals making it slower. In any other scenario the normal clock speed of 4 Hz will be used. This circuit element function is to allow the user to control the speed of the songs and visuals.

2.4 Audio Process

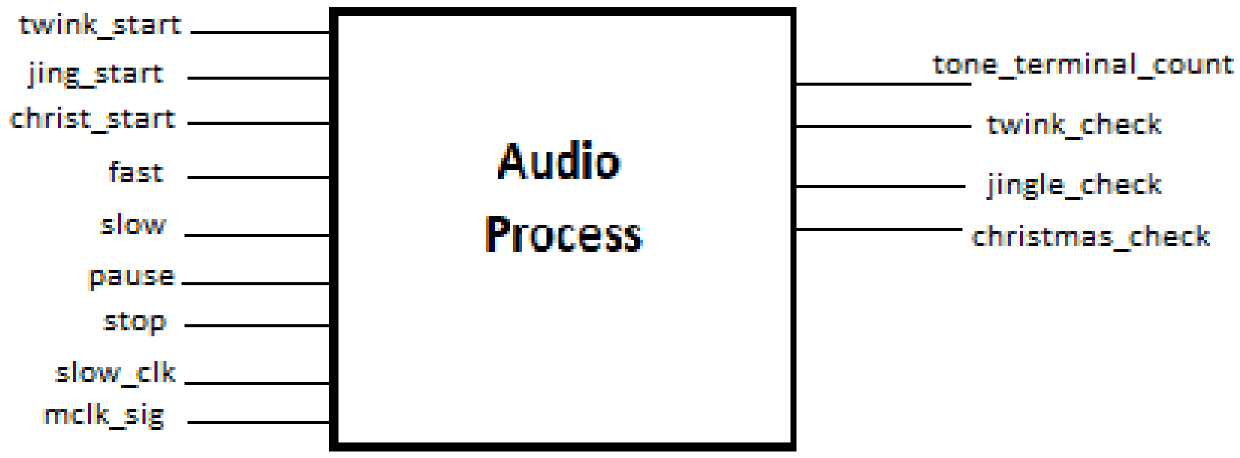

Figure 4: Audio Process Inputs and Outputs

The audio process is a process sensitive to $mclk_{sig}$ and only functions on the rising edge of the clock signal $mclk_{sig}$. The audio process functions by storing 7 bit unsigned notes in an array and then assigning the nth element of the array to the tone terminal count.

The audio process has 3 songs stored in an array which includes Twinkle Twinkle Little Star, Jingle Bells, and We Wish you a Merry Christmas. Each song is selected by the use of an individual push-button twink_start, jing_start, and christ_start. When the button is pressed it will set its corresponding flag signal to 1 and the others to 0. These flag signals are the outputs twink_check, jingle_check, and christmas_check.

When the flag signal is high the corresponding song will play by iterating through each note in the array whenever slow_clk has the value of 1 and assigning it to tone_terminal_count until the end of the song where the array will reset to the zero element of the array and set the corresponding flag low.

However, the song can be stopped by using the stop slider switch which will immediately reset the array to the zeroth element of the array and set the flag low. The song may also be paused using the pause slider switch which will set the flag to 0 but will hold the current position of the array when the slider switch is high, this will allow for the song to resume at the same position it was paused at when the pause slider switch returns low.

Finally, the fast and slow slider switches are used to activate dj mode. When both slider switch inputs are high it allows for each song to be switched between without resetting the previous one allowing for the mixing between songs. The audio process allows for the user to control the songs being played on the jukebox and giving them the ability to pause, stop, and dj the song.

2.5 SSM2603 Chip

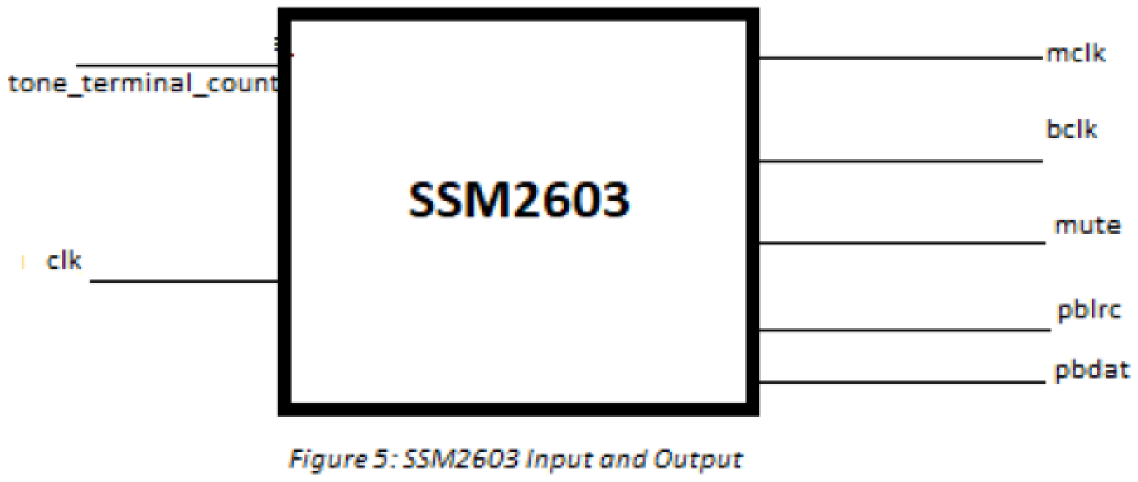

Figure 5: SSM2603 Inputs and Outputs

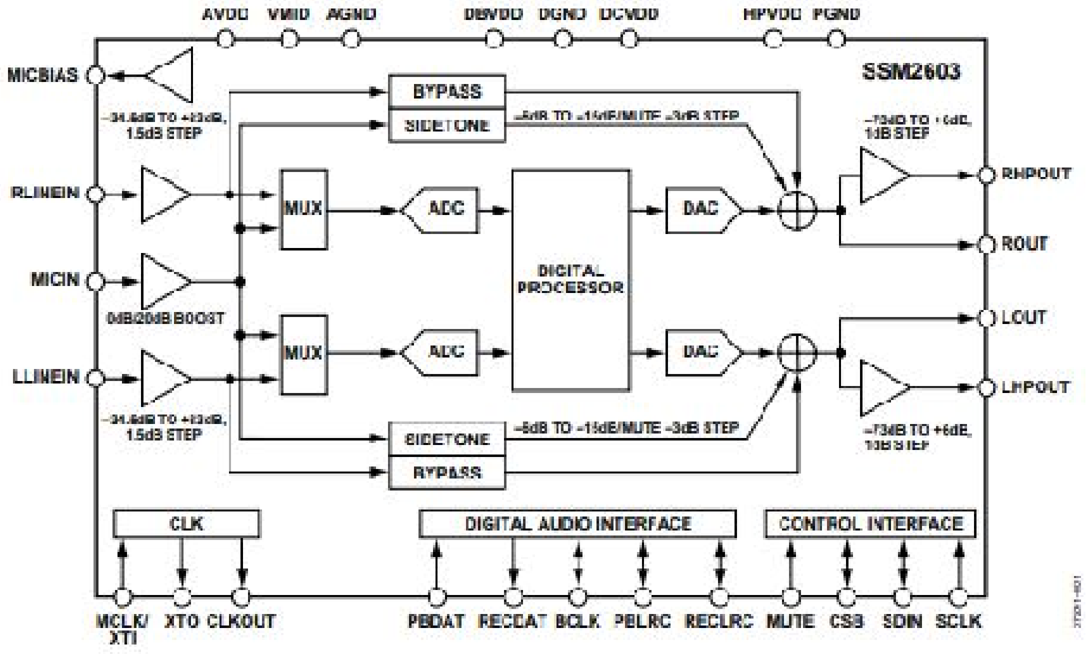

Figure 6: SSM2603 Chip Diagram

The chip diagram of the SSM2603 is shown in Fig. 6, along with its input and output signals in 5. The tone_terminal_count will be converted to l_{data} and r_{data} through the use of a tone counter and a MUX. This data will then be used to create the outputs bclk, mclk, pbdat, pblrc, and mute. This data will then be converted into an audio signal using the audio driver on the board.

2.6 State Machine Diagram/Module

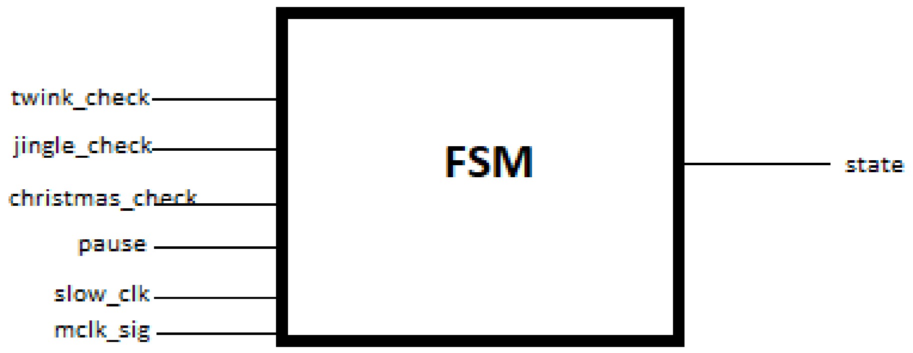

Figure 7: Finite State Machine Inputs and Outputs

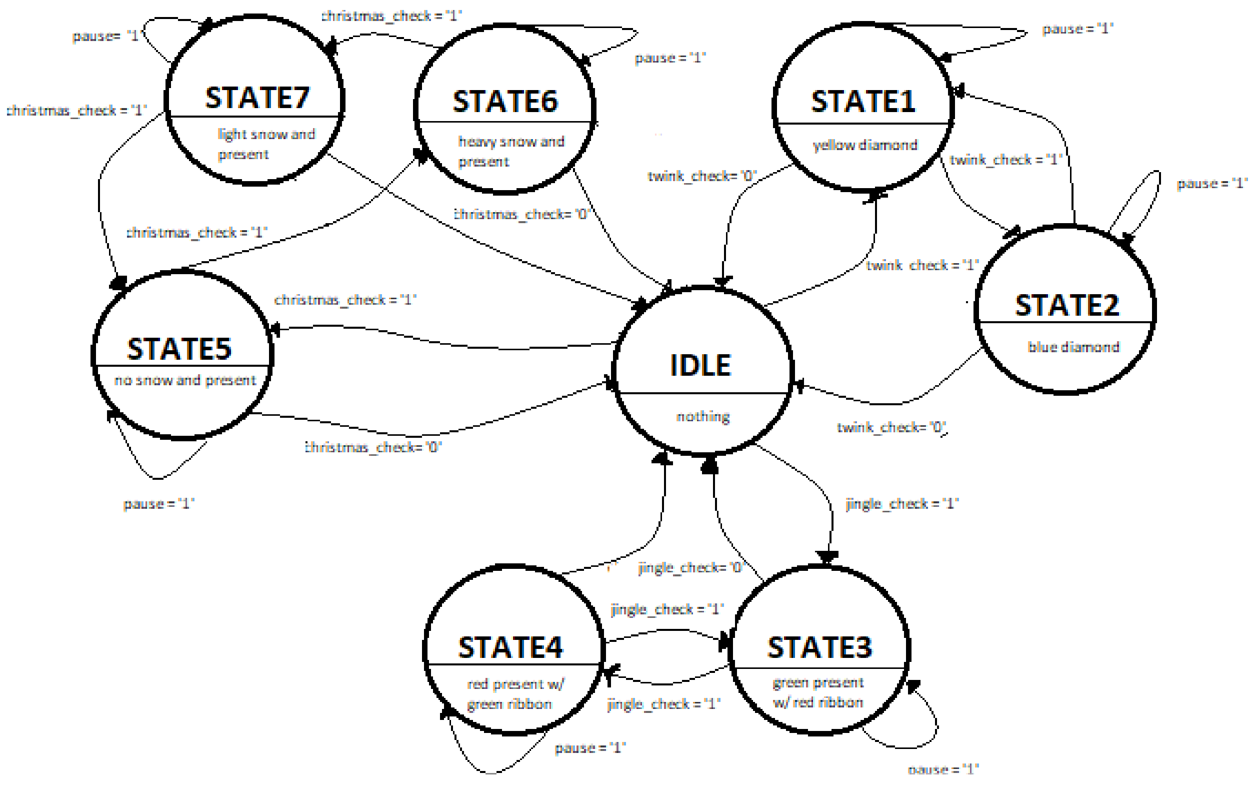

Figure 8: State Machine Diagram

The Finite State Machine module implements the state transition, as shown in Fig. 7 and Fig. 8. This is accomplished using a sequential process that is sensitive to $mclk_{sig}$. Transitions in this diagram will also only occur when the value of $slow_{clk}$ is high.

If no inputs are pressed the state machine will remain in the Idle position where no visuals are displayed. Once an input is selected it will change to the corresponding state where it will rotate between the states corresponding to that song as long as the input flag is still true.

If the flag becomes low, then the state will return to the idle state and display no visuals. If the song is paused, then the visual that the signal is paused on will remain on the screen until the song resumes where it will continue to rotate between the states corresponding to that input. If the stop button is pressed the state will be sent to the idle state because the corresponding input flag will be set low.

The outputs of these states are shown and will be described more in detail in the Visual Process section.

2.7 Visual Process

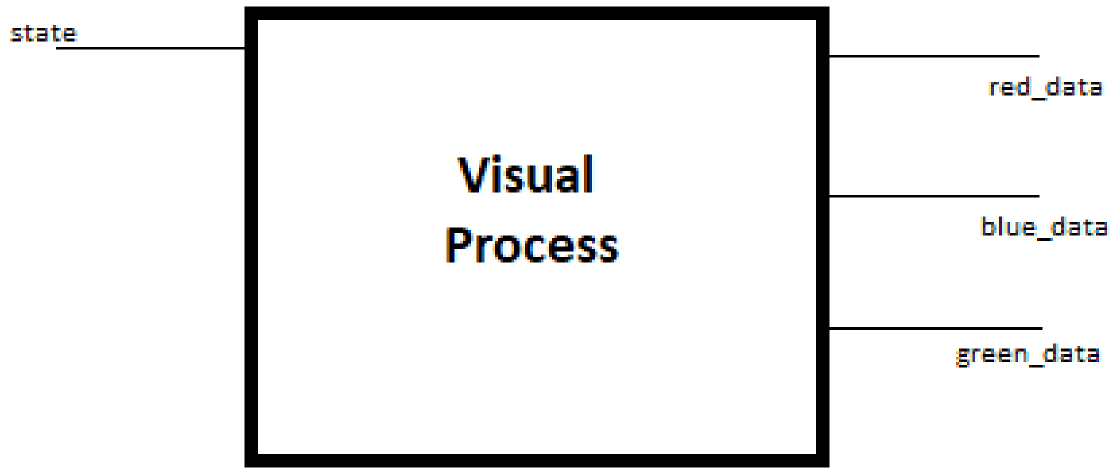

Figure 9: Visual Process Inputs and Outputs

The visual process is a combinational process sensitive to the state, hcount, and vcount. The output of this process is visual data to be displayed on the monitor. When the process is in the idle state no data is sent to the monitor. When in state 1 the screen displays a diamond that is yellow and state 2 is a blue diamond. State 3 displays a green Christmas present with red ribbon sitting on top of snow. State 4 is a red present with green ribbon sitting on top of snow. States 5,6,7 create an animation of snow falling onto the Christmas present and accumulating snow on the ground.

All these states are designed by constraining the values of hcount and vcount so that only color appears in the correct area of the monitor and everywhere else is assigned black. The diamond and rectangle were created by constraining hcount and vcount using 4 different equations of a line to obtain the desired shape. The snow was created by using the modulus function for hcount and vcount. This process defines the states and ultimately outputs the color data that is then driven by the HDMI controller and sent to the monitor to be displayed.

3 Testing and Verification

As discussed below, the results were visually by the students and the professors.

3.1 Testbench and Simulation

A testbench was not set-up to test the design implemented that was implemented for HDMI signaling.

3.2 Objective Verification

The functionality of the Jukebox system with corresponding visuals was verified on the Zybo Z7 development board using the black audio jack connected to a pair of earbuds and the HDMI output connected to a monitor. The three pushbuttons were used as the inputs to start playing the corresponding song and visuals. The four slider switches corresponded to the inputs for pause, stop, fast, and slow.

The first song tested was Twinkle Twinkle Little Star which played when the button corresponding to $twink_{start}$ was pushed. This song played all the way through and stopped once it reached the end of the song. Throughout the song the visuals were displayed and changed between a yellow diamond and a blue diamond when the note of the song changed. The pause functioned accordingly and stored the position of the song and visuals so that when the pause button was unselected it returned to the position it was paused at. The stop functionality also worked so that when the stop was selected the song and visuals stopped and reset so no noise or visuals occurred immediately after it was deselected. The fast slider switch did speed up the tempo of the song as well as the frequency of the change in visuals. The slow slider switch was also effective in slowing down the tempo of Twinkle Twinkle Little Star and the corresponding visual.

These same tests were then done with the other songs Jingle Bells and We Wish you a Merry Christmas. Jingle Bells played as expected with the corresponding visuals switching between a green present with red ribbon sitting on snow and a red present with green ribbon sitting on snow. All additional functionality was tested and verified. We Wish you a Merry Christmas was tested in the same way as the other two and played successfully and the corresponding visuals of making it snow also worked as well as the additional functionality. The final function tested was dj mode which was activated when the slider switches fast and slow were both high. This allowed users to switch between any song without resetting the position of the other songs. This gives the user the ability to mix songs together to create their own song combining the three together. After the extensive testing all songs and visuals functionality were verified to be working.

Glossary

List of Abbreviations

- FSM: Finite State Machine
- CLK: Clock
- VHDL: VHSIC Hardware Description Language
- FPGA: Field Programmable Gate Array
- VHISC: Very High Speed Integrated Circuit
- PLL: Phase-Locked-Loop

Hardware References

- Xilinx Zybo Z7 FPGA — *https://reference.digilentinc.com*
- Xilinx Zybo Z7 Schematic — *https://reference.digilentinc.com*
- Xilinx Vivado Software — *www.xilinx.com*
- UB's Official Color Palette — *www.buffalo.edu*

Note: The websites referenced above are hyperlinked to the actual links used.