MUSIC BOX SPECIFICATION

ECSE 323

Lab #5 – Final Report

by
Group 26
Chuan Qin 260562917
Wei Wang 260580783

McGill University
Department of Electronic and Computer Engineering
Tuesday, March. 14th, 2015

Chuan Qin 260562917
Wei Wang 260580783

Table of Contents

Chuan Qin 260562917
Wei Wang 260580783

1. Project Overview

1.1 Objective

The main goal of this lab is to design a music box and implementing on the Altera DE1 board. A full music box system consists of several different components such as note timer circuit, segments decoder circuit, and flash read circuit developed form the previous labs.

1.2 Design Specification

- ROM based storage of 2 songs that can be selected by a switch on the board
- Each song is limited to maximum of 256 notes
- A LED light displays the current beat
- Four 7-segment Displays:
  - Note number
  - Octave
  - Duration
  - Volume
- Three Buttons:
  - Reset Button: initialize the audio interface
  - Start Button: play the song from the beginning
  - Stop Button: stop playing the song
- Ten Switches:
  - Song Selection: selection between both songs from ROM based storage (1 switch)
  - Tempo of song: varies between 46 to 300 beat per minute (bpm) in steps of 4 bpm manipulated by 6 switches
  - Different modes: One-Shot mode, Continuous Looping mode (1 switch)
  - Pause/ Resume: hold the song and resume the song (1 switch)
  - Shift pitch: the song can be transposed by one octave (1 switch)

2. System Functionality

2.1 Description

The music box system requires a finite state machine which is responsible for monitoring different states that control various modules. There are 4 main states: waiting for the start signal to go low (WSL), waiting for the start signal to go high (WSH), waiting for the trigger to go low (WTL), waiting for the trigger to go high (WTH). There are 5 signals (start, stop, pause, looping, and trigger) that determine state transition. The state transition will be explained in further detail later in this section (Figure 2 and 3).

Firstly, the system reads note data from the memory initialization file (.mif) of the song that is stored in ROM. Each note in the song is represented by a set of 16-bit note information words as described in Table 1. Secondly, a note counter generates the address for the ROM storage. It is set to '0' when starting to play the song. The tempo circuit is connected to the note timer system in order to adjust the beat of the song by changing bpm. Furthermore, the counter increases every time the trigger goes high. The trigger signal is determined by the information words of note duration and triplet from ROM. Finally, the information of octave and note pitch generated from the song ROM is sent to the flash read component. Inside the flash read component, there is a component named audio interface which provide communication to an Audio Codec chip on the board. The schematic diagram for the entire music box system is in Appendix B.

Type of Information	Bit
Note Number	0 – 3
Octave Number	4 – 6
Note Duration	7 – 9
Triplet	10
Loudness	11 – 14
End of Sing marker	15

Table 1: Note information words

The entire music box circuit (in Figure 1) named g26_lab5_FSM requires 11 inputs:

- Clock (clk, 1-bit)
- Reset signal (reset, 1-bit)
- Start signal (start, 1-bit)
- Pause signal (pause, 1-bit)
- Stop signal (stop, 1-bit)
- Song number (songnumber, 1-bit)
- Octave shifted (shiftoctave, 1-bit)
- Loop (whetherloop, 1-bit)
- Beat per minute (BPM, 6-bit)
- Flash memory data input (i_data, 7-bit) (unconnected)
- Write configuration data into register when it is asserted (INIT, 1-bit)

And it generates 16 outputs:

- Trigger signal (triggerforled, 1-bit)
- Segment display for note number (segmentsofnotenumber, 7-bit)
- Segments display of octave (segmentsofoctave, 7-bit)
- Segment display of note duration (segmentsofnoteduration, 7-bit)
- Segment display of volume (segmentsofvolume, 7-bit)
- Accessed address in the memory (FL_ADDR, 8-bit)
- Bidirectional bus that pass the data to and from the flash memory (FL_DQ, 8-bit)
- Control signal of flash memory to ensure whether the transformation is complete (FL_CE_N, FL_OE_N, FL_WE_N, FL_RST_N, 1-bit)
- Audio codec master clock (AUD_MCLK, 1-bit)
- Audio bit clock (AUD_BCLK, 1-bit)
- Audio DAC data line (AUD_DACDAT, 1-bit)
- Audio DAC data selection (AUD_DACLRCK, 1-bit)
- Audio serial interface data line (I2C_SDAT, 1-bit)
- Audio serial interface clock (I2C_SCLK, 1-bit)

Each note number refers to a specific musical scale note in Table 2.

Musical Scale	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
Note Number	0	1	2	3	4	5	6	7	8	9	10	11	12-15
Note: the last C note for number 12-15 is one octave higher than the C note corresponding to 0													

Table 2: Note number vs. Musical scale note

2.2 Block Diagram

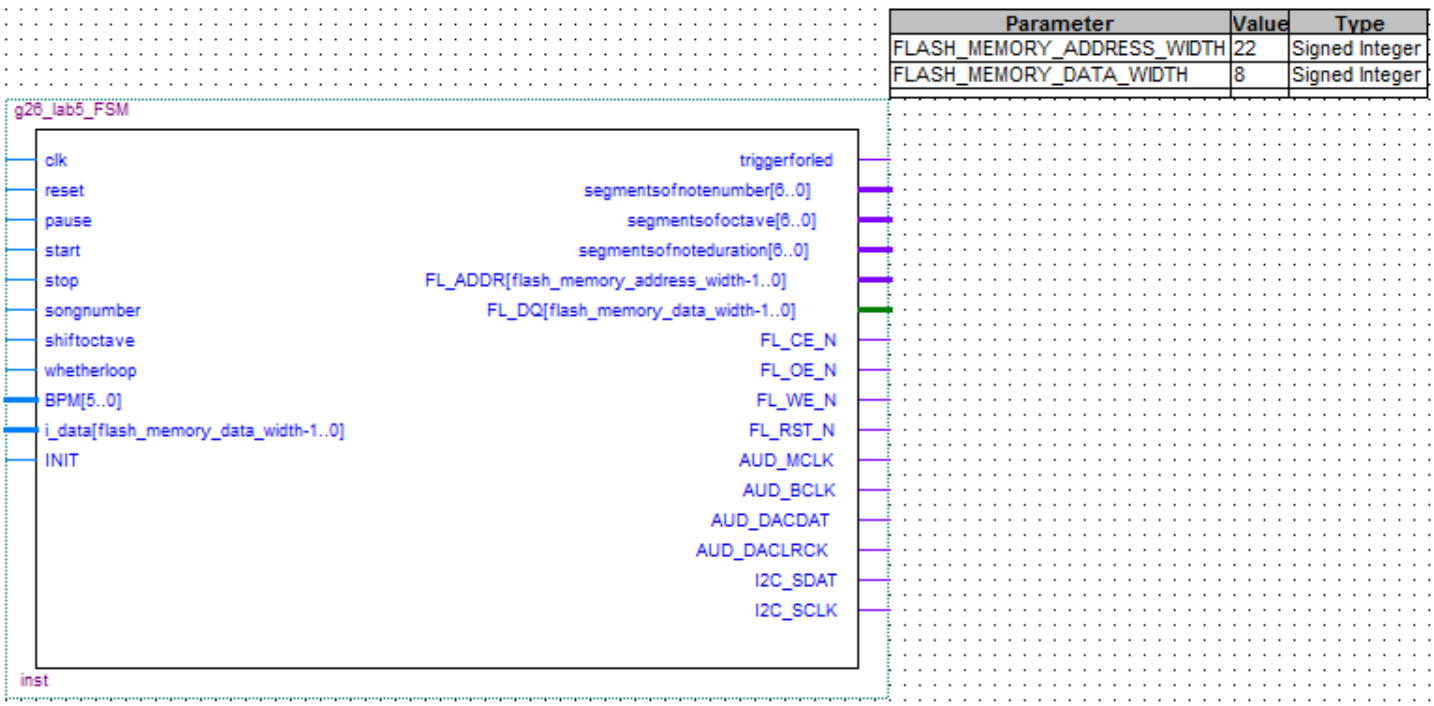


Figure 1: Complete music box system represented as a single block (g26_lab5_FSM.vhd)

Chuan Qin 260562917
Wei Wang 260580783

2.3 Finite State Machine

The music box system initializes the state as WSL (wait for start goes low) and the count as zero once the reset signal is activated. There are 3 buttons and 1 switch on the DE1 board are used to control the inputs such as reset, start, stop, pause, and loop. It is important to know that those buttons are active low for design the finite state machine. The state transition can only occur every time clock is asserted. When the start signal is '0', the states transit from WSL to WSH (wait for start goes high). For the WSH state, if the start signal is '1', it turns to the state WTH (wait for trigger goes high). Or it turns to the state WSL and initializes the count if the stop signal is '0'. Once the system enters the WTH state, the count increase by 1 every time trigger is high and turns to the state WTL (wait for trigger goes low). If the system reaches the end of the song and there is sign for repeating the song, it turns to the state WSL. Also, the systems will automatically turns to the state WSL if the stop signal is '0'. Or the system reset the count to zero if it reaches the end of the song, the looping signal is activated, and stop signal is '1'. For the state WTL, the system returns back to the state WTH if the trigger goes low. Once the stop signal is active, the system will be reset. During the playback of the song, the system jumps between the state WTH and WTL. In addition, we create a temporary variable called whetherstop controlled by the pause switch and stop button in state WSH, WTH, and WTL to notify the system whether it should resume or stop the play. With this additional feature, the users are allow to pause or stop at any time while the song is playing.

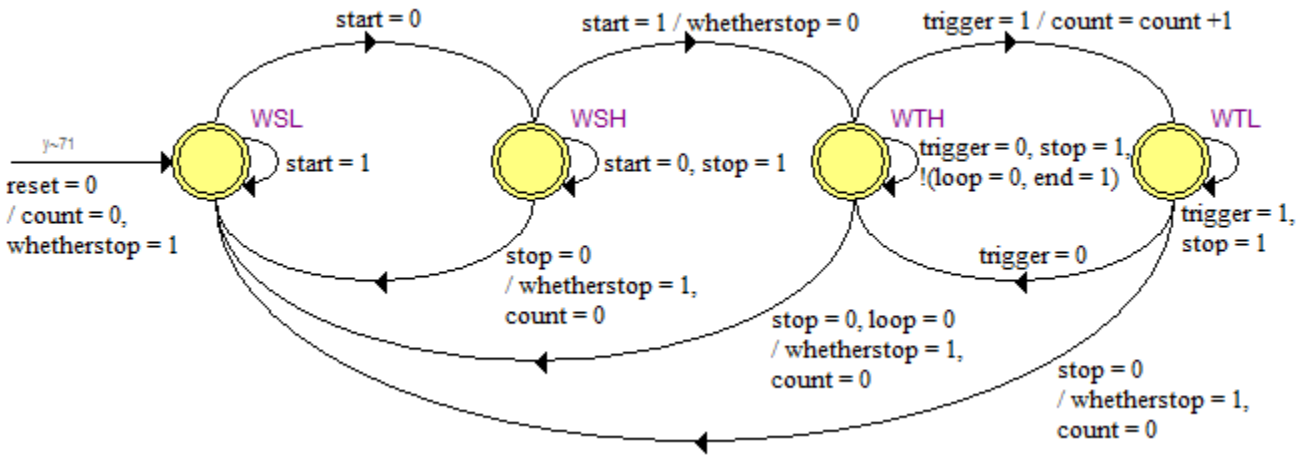


Figure 2: State diagram of the music box system (g26_FSM_simulation.vhd)

	Source State	Destination State	
1	WSH	WSH	(!start).(stop)
2	WSH	WSL	(!start).(!stop)
3	WSH	WTH	(start)
4	WSL	WSH	(!start)
5	WSL	WSL	(start)
6	WTH	WSL	(!stop).(!tempoftrigger) + (stop).(!tempofaddresscounter[1]).(!tempofaddresscounter[5]).(!tempofaddresscounter[6]).(tempofaddresscounter[7]).(!whetherloop).(!tempoftrigger) + (stop).(!tempofaddressco
7	WTH	WTH	(stop).(!tempofaddresscounter[1]).(!tempofaddresscounter[5]).(!tempofaddresscounter[6]).(!tempofaddresscounter[7]).(!tempoftrigger) + (stop).(!tempofaddresscounter[1]).(!tempofaddresscounter[5]).(!
8	WTH	WTL	(tempoftrigger)
9	WTL	WSL	(!stop).(!tempoftrigger)
10	WTL	WTH	(!tempoftrigger)
11	WTL	WTL	(stop).(!tempoftrigger)

Figure 3: The path for each state (g26_FSM_simulation.vhd)

3. Detailed Design Description

In this section, we explain the music box system by three main parts: Note Timer, Flash Read, and Segment Decoder.

3.1 Note Timer Circuit

The note timer circuit takes in the note duration and triplet information, then it generates a note gate signal which goes high for the appropriate time length.

The inputs and the outputs of the circuit (Figure 4) are listed as follows:

- Clock signal input (clk)
- Asynchronous reset input which is active low (reset)
- 1-bit input to pause or resume the play (pause)
- 3-bit input for note duration (note_duration)
- 1-bit input for triplet to turn to different rhythm (triplet)
- 1-bit input indicating the end of the song (playend)
- 8-bit input x representing bpm values (bpm)
- 1-bit output (TRIGGER)

Figure 4: Note timer component block diagram presentation (g26_note_timer_board.vhd)

The note duration is converted to numbers of tempo pulses per note using lookup table. When the triplet input is '1', the number of tempo pulses becomes two-third of a regular note length as in Table 3.

note_duration		0	1	2	3	4	5	6	7
Triplet	0	3	6	12	24	48	96	192	384
	1	2	4	8	16	32	64	128	256

Table 3: Note duration number of tempo pulse Conversion with 2 possible triplet values

For the note timer circuit, the reset is initially zero and the TRIGGER signal is high. Then, the value of count increases on every clock cycle when the tempo_enable signal from the tempo circuit is high. The TRIGGER goes low when the count values reaches zero. However, the TRIGGER is set to high if the count value is equivalent to the number listed in the table above, minus 1. The circuit holds the output high for one tempo pulses and turns low for the remaining ones, and then, the cycles repeat until the pause and playend inputs are active.

3.1.1 Tempo

The tempo circuit is designed to provide the basic tempo for music by computing the function:

$$f(x) = \frac{125000000}{x}$$

The inputs and the outputs of the circuit (Figure 5) are listed as follows:

- Clock signal of 50M Hz input (clk)
- Asynchronous reset input which is active low (reset)
- 8-bit input x representing bpm values (bpm)
- 1-bit beat output (beat)
- 1-bit tempo enable (tempo_enable)
- 5-bit output from the second counter for verifying whether the circuit work properly (beat_output)

In the system, we implement the lookup table approach instead of doing calculation every time there is a new input value for bpm. The range of bpm values is starting from 45 to 300, but the address of the input '0' corresponds to bpm of 45 in the lookup table. For example, bpm = 80 which will be stored at the address 35 has a value of $int\left(\frac{125000000}{80}\right) = 1562500$ in the lookup table.

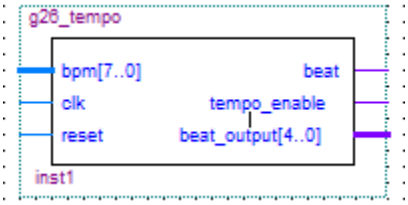


Figure 5: Tempo circuit block diagram presentation (g26_tempo.vhd)

Moreover, each beat is divided into subdivision of 24 bits, allowing multiple notes in one beat. The tempo counter implements 2 lpm_counters that count down to zero. In general, each counter has 5 inputs: clock (clk), clear (aclr), load (sload), count enable (cnt_en), and data value that generate a single output (q). For the output q, the select assignment statement assigns a value of '1' to a variable called temp when the bits for q are all zeros; otherwise, temp is set to be '0'. For both counters, aclr is set to be inverse of reset value.

The first counter loads in a constant value retrieved from the lookup table, and then counts down from that value. The count enable signal is equal to '1', and temp is fed back to sload port.

The second counter counts down from 24, so it requires at least a 5-bit input. This counter loads the value of tempo_enable as an enable control. The most significant bit of the output from this counter is the output beat. Its output goes through the select assignment statement, then ANDED with the tempo_enable value fed back to the sload port of this counter.

The schematic diagram includes the first counter, which is on the left and the second one on the right in the figure shown as below.

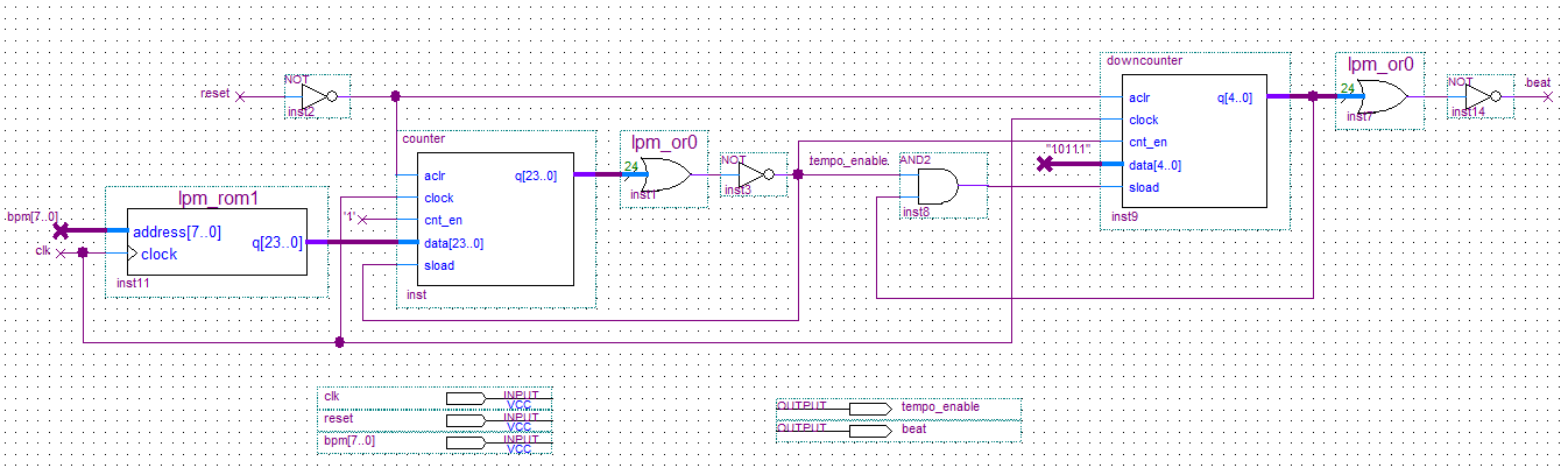


Figure 6: Schematic Diagram of Tempo Counter Architecture

3.2 Flash Read (Figure 7)

The principle of flash read circuit is to build communications between several components, including Altera Flash memory interface IP core module, flash read control module, and audio interface module (The schematic diagram of the circuit is in Appendix A). There are four vdhl code provided by prof. Clark as listed that are used to access flash memory.

1. Altera_UP_Flash_Memory_IP_Core_Standalone.vhd
2. Altera_UP_Flash_Memory_User_Interface.vhd
3. Altera_UP_Flash_Memory_Controller.vhd
4. g26_audio_interface.vhd

Furthermore, the volume of each note is adjusted by multiplying the sample values by the bit string of loudness, and then, it sends the result to the audio interface module. This operation is done by using a build-up lpm multiplier component for signed number because of the type of the sample values.

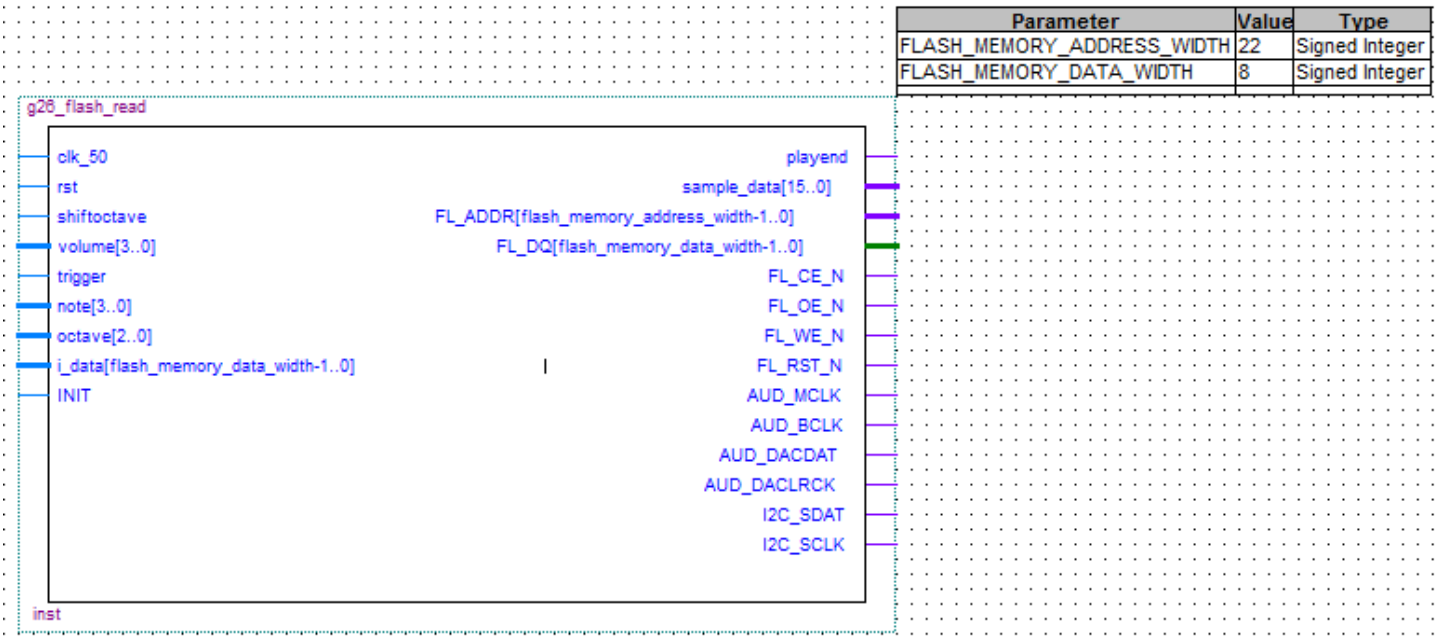


Figure 7: Flash Read Component presentation as a block diagram (g26_flash_read.vhd)

3.2.1 Flash Read Control

The read control module is designed to control the read cooperation of the 4MB flash memory on the board. We use control panel to write a wavefile (Strat Bridge 52.wav) to the Flash Memory chip that allows us to develop the sound of music box.

The circuit (Figure 8) requires 9 inputs:
- Clock signal of 50 MHz (clk_50)
- Asynchronous reset which is active-low (rst)
- Trigger input which can reset the sample address to the beginning (trigger)
- 4-bit number note (note)
- 3-bit number octave controlled by slide switches
- 8-bit input that is the output of the flash memory (odata)
- 1-bit input that indicates the read operation is done (read_done)
- 1-bit input indicating that the next sample should be read (step)
- 1-bit input that controls octave shifting (shiftoctave)

And it outputs 4 values:
- 22-bit address of flash for the flash memory read operation (flash_address)
- 16-bit data sent to the audio codec chip (sample_data)
- 1-bit output which can request a read operation on the flash memory (read_start)

The octave and note inputs decide the frequency of playing back the sample. Once the read start goes high, it will indicate the interface of flash memory to read the data on the specific address given by another output named flash address of the current circuit. When the read operation is complete, the interface outputs a read done signal

to inform the circuit and then it will receive an 8-bit data. After two cycles, the circuit generates a 16-bit sample data which is fed to the audio interface circuit. Every time the input trigger goes high, the flash read control circuit reads the sample data at the beginning of the address. And then, the step input becomes high for generating next data values. If the shiftoctave input is '1', the circuit converts the pitch number to a specific frequency by computing the function $f(N) = f(0) \times 2^{\frac{N}{12}}$ where $f(0) = 515396$ through an inside combinational circuit. Otherwise, the frequency of the note being played is divided by a factor of two.

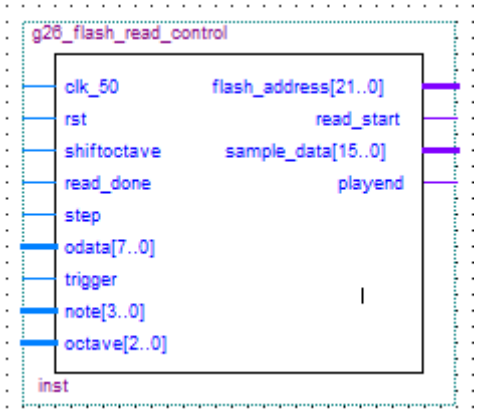


Figure 8: Flash Read Control Circuit (g26_flash_read_control.vhd)

The circuit requires 11 specific states as shown in Table 4. First 5 states are responsible for reading data, and the rest of 6 states are used to read audio sample.

State Number	Purpose of the state	Next State
0	Initialize the reading from the data address	1
1	Read 1st byte of file length	2
2	Verify whether the system has done reading the file (read_done)	3
3	Read 2nd byte of file length	4
4	Verify the read_done data	5
5	Read the 3rd byte of file length	6
6	Wait until step goes low	7
7	Wait for memory to be ready when the step is high	8
8	Read and store 1st byte of data when read_done is high	9
9	Prepare the memory when the read_done is low	10
10	Read and store 2nd byte until the read_done is high	6

Table 4: State Transition of Flash Read Control Circuit

3.3 Segments Decoder (Figure 9)

A typical 7-segment displays on the Altera board consists of 7 individual LED segment in order to produce the required hexadecimal digits from 0 to 9 and A to F respectively. There are 4 digit display with each digit being a 7-segment LED decoder. In addition, the original design for decoder circuit have ripple-blanking capability which means that the leading zeros is forced to be turned off for a multi-digit display. If the system needs to display multi-digits values, we connect the ripple-blanking input port of the current decoder to the ripple-blanking output port of the previous decoder. However, the rightmost LED display must print out all the hexadecimal characters including '0', so the ripple-blanking input is force to be '0'. In fact, the set of decoders can also display the values in decimal by converting the binary value to BCD digits.

For the music box system, each individual decoder takes in bits from the song ROM, and 4 of the decoder are used. Starting from the rightmost, the first decoder takes in bit 0 – bit 3 from the song ROM o print out the note number. The second one gets the octave number information from bit 4 – 6. The third one takes bit 7 – 9 to display the note duration. The last one, also the leftmost one, display the loudness of the playing note by taking in bit 11 – 14. All the ripple-blanking port of the decoders are connected to '0'.

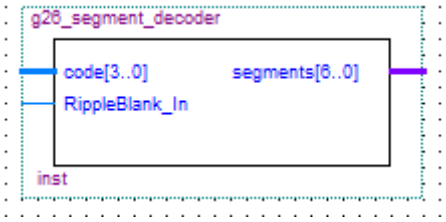


Figure 9: 7-segment Decoder block diagram (g26_segment_decoder.vhd)

4. Testing

4.1 Finite State Machine (FSM) Testing

To run the timing simulation of the finite state machine (g26_lab5_FSM_simulation.vhd), we set the maximum number of count to 29. The clock rate is set to 50M Hz. In the beginning, we set reset signal to '0' as we manually press the reset button. Pressing the buttons that means the signals the corresponding to the buttons is '0' in the diagrams. In order to observe the count number and the state transition, we force the trigger goes high every 1us. In the figure 10, once the reset button is pressed, the system is at state WSL, and after the start button is pressed for few clock cycles, it turns to WSH state. When the start button is released, the system is at WTH state. As shown in the figure 11, when the trigger signal is low, the current state should remain at the state WTH. When the trigger goes high, the state transits to the state WTL. Also, we observe that the count number (refers to addresscounter in all figures) is increased by one when the system detects that the trigger goes from low to high.

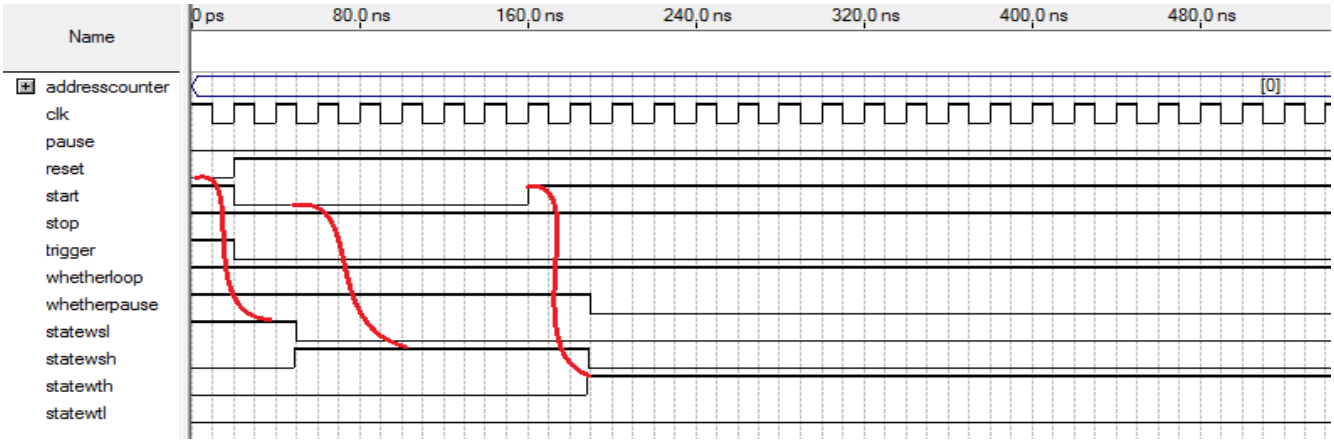


Figure 10: Timing Simulation for FSM when pressing reset and start buttons

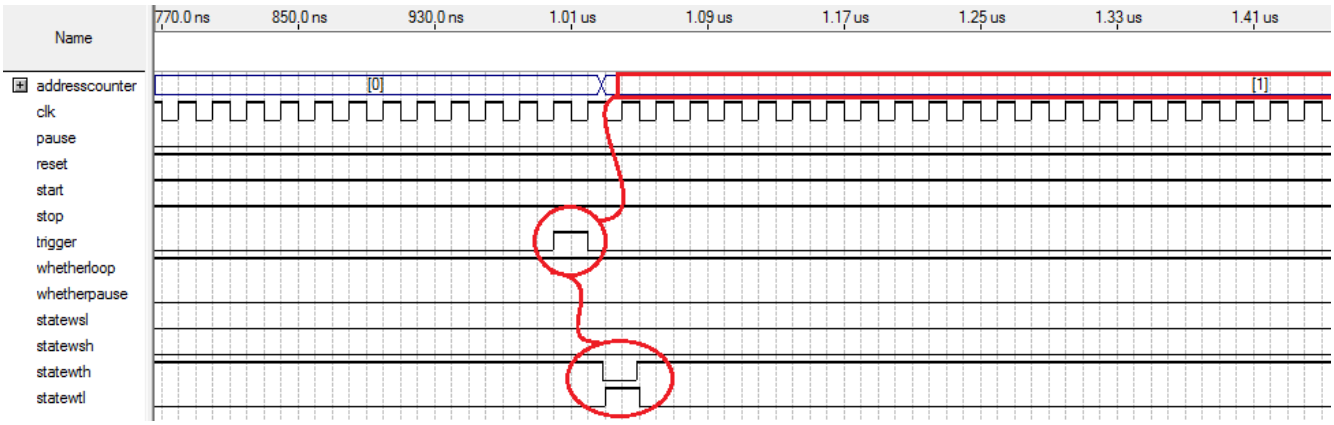


Figure 11: Timing Simulation for FSM when trigger changes

Next, we follows the similar logic to verify the stop signal. Once the stop button is pressed, the system goes back to the WSL state, and the node whetherpause, which is the temporary variable whetherstop ADDED with the pause signal, goes high in the Figure 12. If we press start button again, the state goes to WSH and changes to WTH right after.

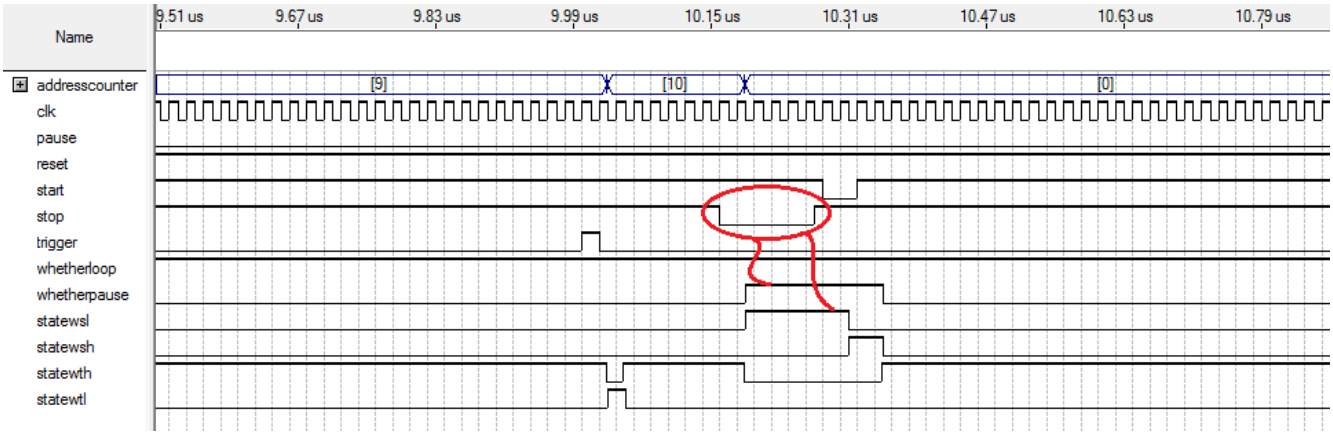


Figure 12: Timing Simulation for FSM when pressing stop button

We assume that the song is played in looping mode as Figure 13. The count number is set back to zero when trigger becomes high after playing the last note. In this case, there are 30 notes in total. When the count number reaches to 29, it goes back to zero for looping mode. However, if he son is played in one-shot mode, the system is at the WSL state and stop playing the song after the count number reaches the last note. The music box will not play any sound before re-pressing the start button. We can also observe from the figure 14 that the systems jumps between two states: WTL, WTH while the song is playing.

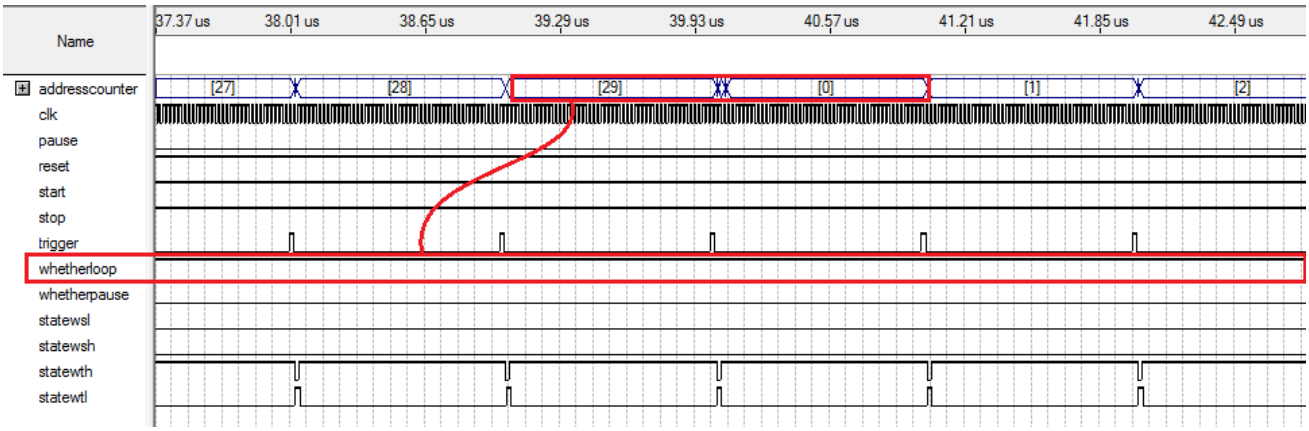


Figure 13: Timing Simulation for FSM when the music box is in looping mode

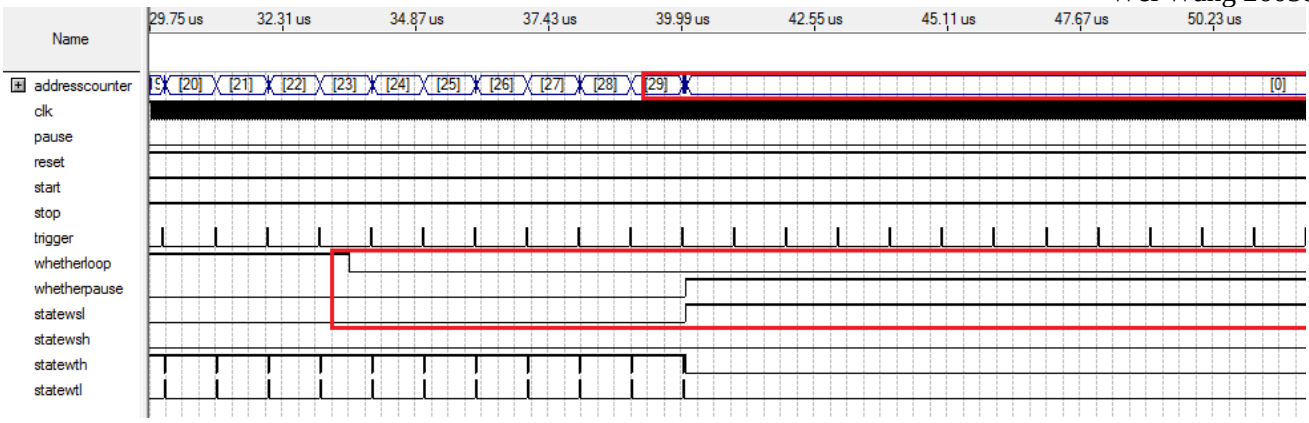


Figure 14: Timing Simulation for FSM when the music box is in one-shot mode

In term of pause switch, the pause signal is equal to '1' when the switch is turned on. In the figure 15, once the pause signal goes high, the whetherpause signal goes high. We can observe that the count number stop increasing until the pause signal goes low. Also, the system stays at the WTH state.

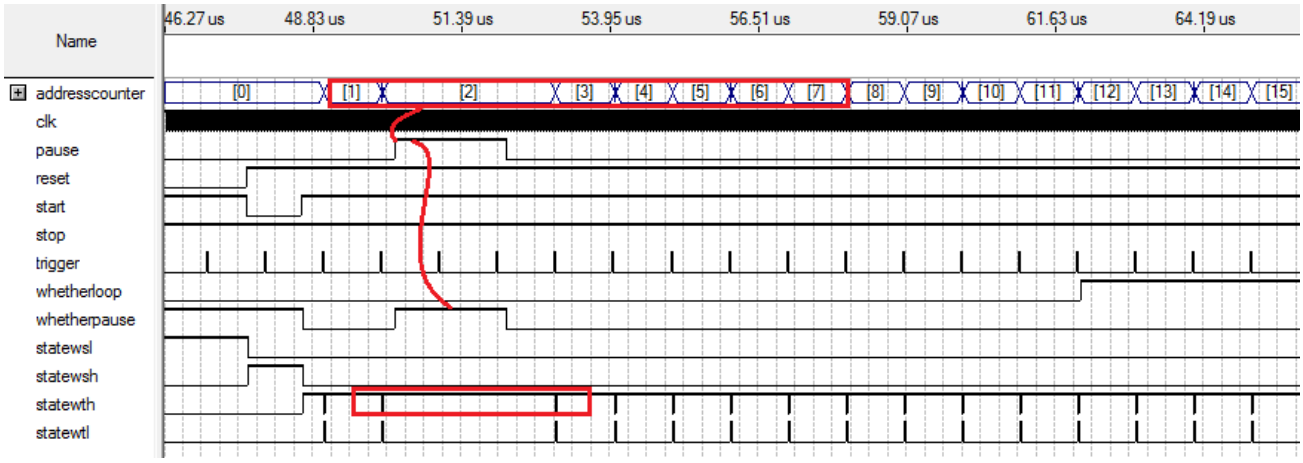


Figure 15: Timing Simulation for FSM when the pause switch is turned on

4.2 Altera DE1 Board Testing

For implementing the music box module on the board, we need to set the switches and the buttons.

Type	Name	Label
Buttons	Reset	Key 2
	Initialization	Key 3
	Start	Key 0
	Stop	Key 1
Switches	Song 1 (Castle in the sky) Song 2 (Für Elise)	SW 0
	Shift Octave	SW 1
	Pause	SW 2
	Looping mode	SW 3
	Beats per minutes	SW 4 - 9
LED light display	Trigger	LEDR0
Segment Displays	Note Duration	HEX 3
	Note Number	HEX 2
	Octave	HEX 1
	Volume	HEX 0

Table 5: Mapping the system's port to the switches and buttons

Note: press the initialization button to clear playback every time starting the DE1 board.

The song chosen by us is Castle in the sky. We modify the song, changing the length, and convert each new note into a 16-bit string. The loudness is set to 1111 throughout the whole song. The note duration and octave changes related to the note number. We listed the first 5 note from the song in the table shown below as an example.

	End	**Loudness**				**Triplet**	**Duration**			**Tempo**	**Octave**			**Note**					**16-bit Binary**
	15	14	13	12	11	10	9	8	7		6	5	4	3	2	1	0		
0	0	1	1	1	1	0	0	1	0	12	1	0	0	1	0	0	1	A	0111100101001001
1	0	1	1	1	1	0	0	1	0	12	1	0	0	1	0	1	1	B	0111100101001011
2	0	1	1	1	1	0	0	1	0	12	1	0	0	0	0	0	0	C	0111100101000000
3	0	1	1	1	1	0	0	1	0	12	1	0	0	0	0	0	0	C	0111100101000000
4	0	1	1	1	1	0	0	1	0	12	1	0	0	0	0	0	0	C	0111100101000000

Table 6: First five notes of the song Castle in the sky

Chuan Qin 260562917
Wei Wang 260580783

5. Timing Analysis

According to the Flow Summary section of the Compilation Report (Figure 16), we ignore the four segment decoder components and keep the rest of the music box architecture. Thus, the entire music box system requires 693 logic elements in total. As using lookup table approach, it means that the system decreases the total number of logic element used. More specifically, it increases the total number of memory bits used which is 12432 in this case. If the system architecture includes the 4 segment decoder, the total amount of logic elements and memory bits change to 716 and 14480, respectively.

Flow Status	Successful - Mon Apr 13 14:29:07 2015
Quartus II Version	9.1 Build 222 10/21/2009 SJ Full Version
Revision Name	lab5
Top-level Entity Name	g26_lab5_FSM
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	693 / 18,752 (4 %)
Total combinational functions	618 / 18,752 (3 %)
Dedicated logic registers	373 / 18,752 (2 %)
Total registers	373
Total pins	64 / 315 (20 %)
Total virtual pins	0
Total memory bits	12,432 / 239,616 (5 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 16: FPGA Flow Summary showing resource utilization of the entire system

First, we use Classic Timing Analyzer for compilation which defines the time delays. In the figure 17, there are three various times such as setup time, clock to output time, and hold time under worst case scenario. Based on the timing simulation of the finite state machine, we observe delays for state transition after receiving the input values. In the figure 18, once the trigger goes high, the state transits from WTH to WTL after the range of 8.12 ns to 9.92 ns. The settling time varies for every transition; therefore, we retrieved another example of time delay from the timing simulation. As in the figure 19, the system turns to the WSH state in the range of 9.0 ns and 9.6 ns after pressing the start buttons. From both case, we notices that the delay values on the timing simulation are not always smaller the worst case of clock to output delay (8.925ns) defined by classic timing analyzer. More specifically, the system requires more setup time compared to clock to output delay and hold time. In the figure 17, the worst case of setup time (10.831 ns). In addition, we add timing constraints related to the clock for the entire music box system. The purpose of adding constraints for the system is to ensure there is no time violation occur during the performance.

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To
1	Worst-case tsu	N/A	None	10.831 ns	songnumber	g26_flash_read:Gate2|g26_flash_read_control:Gate1
2	Worst-case tco	N/A	None	8.925 ns	g26_note_timer_board:Gate1|TRIGGER	triggerforled
3	Worst-case th	N/A	None	0.613 ns	BPM[3]	g26_note_timer_board:Gate1|g26_tempo:Gate1|lpm_
4	Clock Setup: 'clk'	N/A	None	66.83 MHz (period = 14.964 ns)	lpm_rom:crc_table2|altrom:srom|altsyncram:rom_...	g26_flash_read:Gate2|g26_flash_read_control:Gate1
5	Total number of failed paths					

Figure 17: Timing Analyzer Summary for different worst case time delays

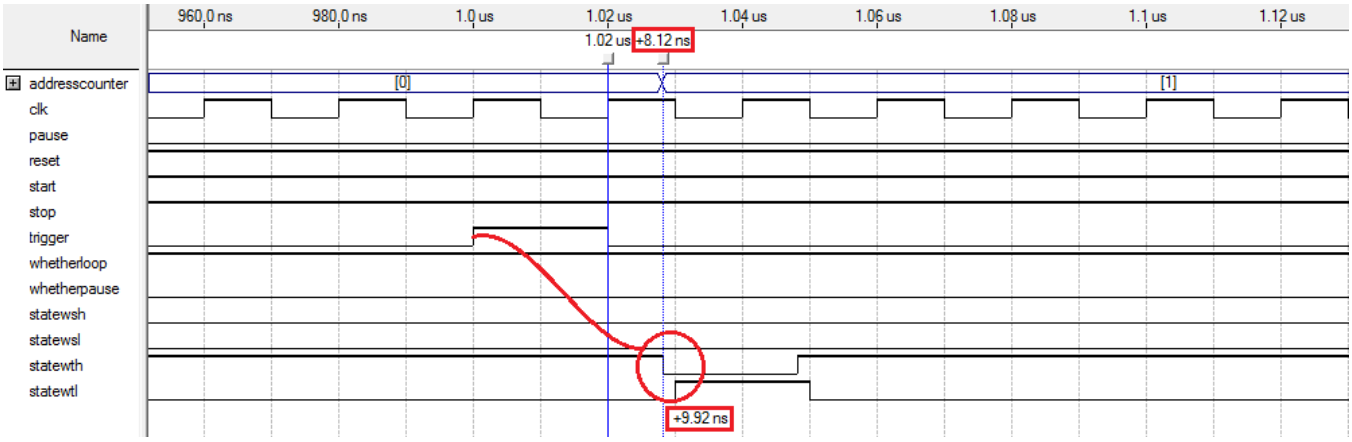


Figure 18: Timing Simulation with time delays for trigger signal

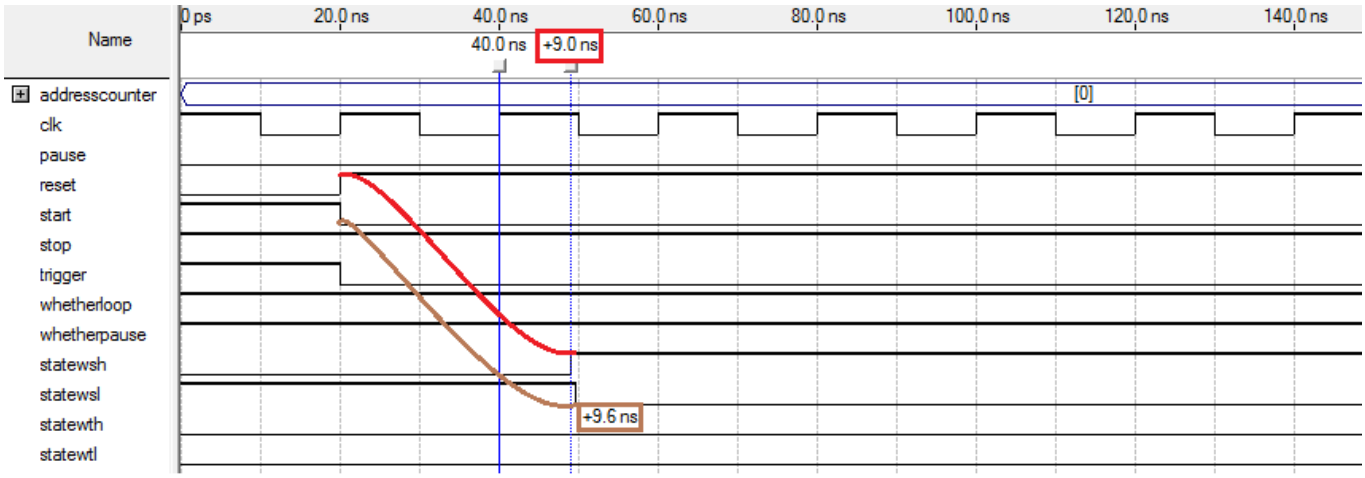


Figure 19: Timing Simulation with time delay (red: reset signal, brown start signal)

Second, after adding the timing constraints, we recompile the circuit using TimeQuest Timing Analyzer. We notice that now the system clock is at correct frequency which is 50M Hz. Fmax value (62.83M Hz) listed in the table below is based on the path where the launch and latch clock are equivalent, and is only reported within a clock domain. We need to ensure that setup timing is met in slow model which indicates the slowest possible

performance for any single path, and then hold timing is met in fast model which indicates fastest possible performance for any single path. Also, slack values must be positive to ensure that the music box operates properly. The setup slack time calculates the difference between the time required for setup and the time for data to arrive. The hold slack time is the time for data arrival minus the time required for hold. For the music box system, the total of all negative slack time is zero.

Clock		Period (ns)	20.000
		Frequency (Hz)	50.0 M
Timing Summaries		Device : EP2C20F484C7N	End Point (TNS)
Slow Model	**Fmax Value (Hz)**	**62.83 M**	**-**
	Setup Slack Value (ns)	**4.083**	**0.000**
	Hold Slack Value (ns)	0.445	0.000
Fast Model	Setup Slack Value (ns)	6.771	0.000
	Hold Slack Value (ns)	**0.215**	**0.000**

Table 7: TimeQuest Timing Analyzer

There is not any unconstrained path for the system without implementing the segment decoder as Figure 20. However, if we implement all four segment displays in the architecture design, the compilation report shows that there are 7 unconstrained output ports for both setup and hold because of the segment display of note number. Also, the Fmax value and setup slack time for slow model become larger than those in Table 7 after putting the segment decoder into the system.

Unconstrained Paths

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	0	0
4	Unconstrained Input Port Paths	0	0
5	Unconstrained Output Ports	0	0
6	Unconstrained Output Port Paths	0	0

Figure 20: Unconstrained path summary for music box without 4 segment displays

6. Conclusion

In this lab, we create a vhdl file called g26_lab5_FSM which requires all the knowledge we had learn from previous labs. We remove few inputs and outputs unused for this lab such as data_size_o from flash_read circuit, and add several ports related to the buttons and switches. The build-in netlist tool in Quartus II helps us to verify whether the components in the system are well connected. The most important part of the system is the finite state machine (FSM). For FSM development, we first draw the state transition diagram, and then translate into vhdl language.

After finishing the code for state transition, we need to load the song information words from the given mif file by implementing lpm block, and we save the data obtained into a signal named tempofsongrom. Each signal is divided into 6 parts to provide different information including note number, note duration, loudness, etc. Before running the programing file onto the DE1 board, we assign the pins to the corresponding input ports by following the user manual.

Then, we begin the testing. In the beginning, we noticed that each note played sounds very short, especially when the bpm value get higher. We believe it is the logic delay of the state machine that makes each note is cut off; however, the idea is wrong. The cause of cut off note problem is the input trigger that we designed in lab 4. By default, the trigger is active low, but we forget to take this factor into consideration. Thus, the music box system plays each note of the song properly.

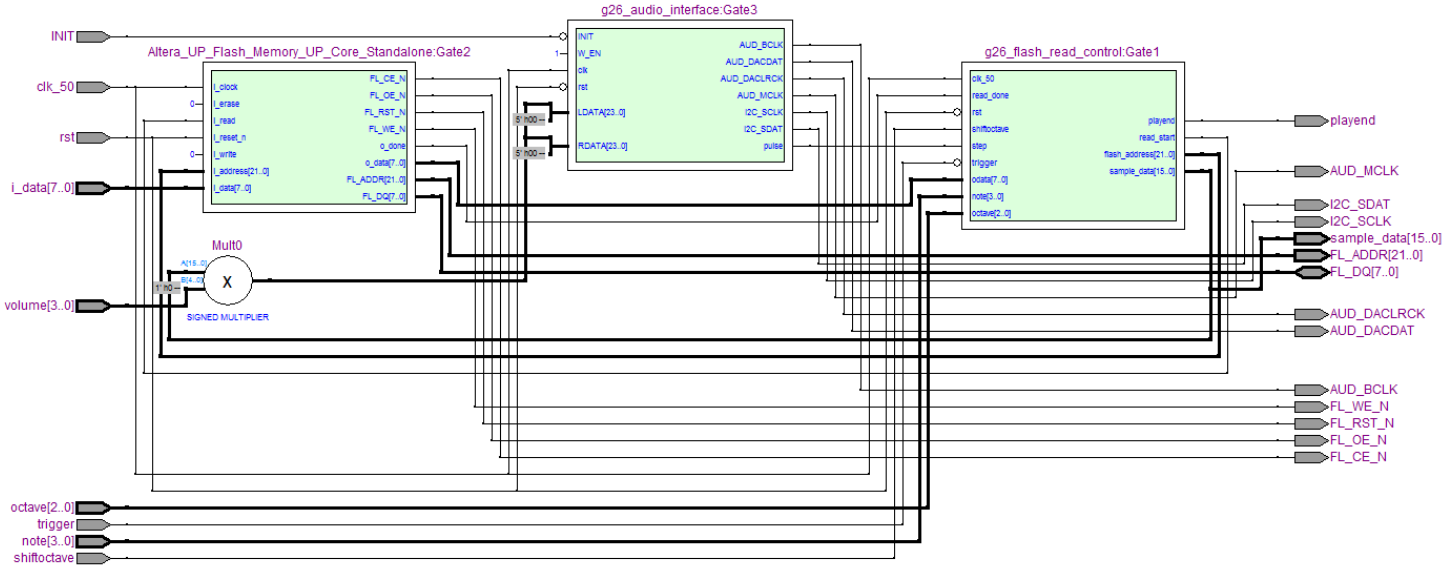
In addition, we suggest two improvements for the current system design which we do not have enough time to do. Every time the stop button is pressed or the pause switch is turned on, the system does not stop playing the song instantaneously, instead it plays one extra note before stopping. Also, the music box system that we design is unable to play more than one note at a time with one single type of sound provided. This is also the reason that we need to adjust the chosen song Castle in the sky.

Here are some suggestion for further development of the music box system for improve the system. The firs solution is to add a new controller called mute which is initially low and becomes high when stop or pause signals are active. Once the mute signal goes high, the system will stop generating the trigger value from the module note timer board. For the second one, we can create multiple flash-read controller. In case the song chosen have two or more notes played at a time, we can create several mif files that contain the note information, and then, let the board play the notes at the same time by transmitting the information of each file to each flash-read controller. Or, we can keep the design of one flash-read controller with multiple audio interfaces. First, we erase the flash memory and divide it into several blocks. Each block is responsible to store one kind of instrument's sound. Second, all the audio interfaces receive the same data inputs from flash-read controller. So, each one should play an instrument's sound from different blocks of flash memory. The sound of instruments can be from piano, violin, cello, or drum.

In conclusion, beside the suggestion of the improvement, we successfully created a music box with two song selections by combining all the modules of previous laboratories. And we gain greater knowledge about VHDL language.

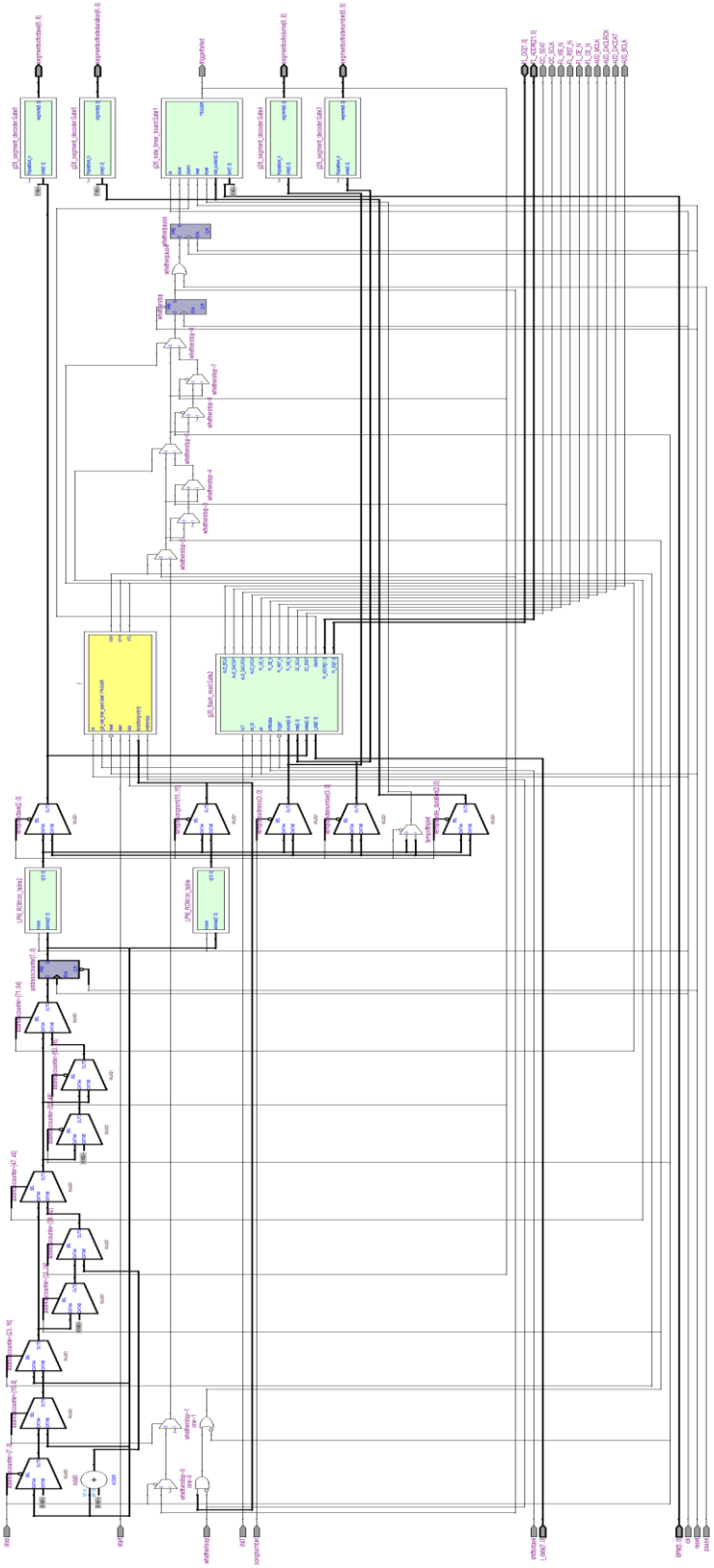
7. Appendices

Appendix A: Schematic diagram of flash read (g26_flash_read)



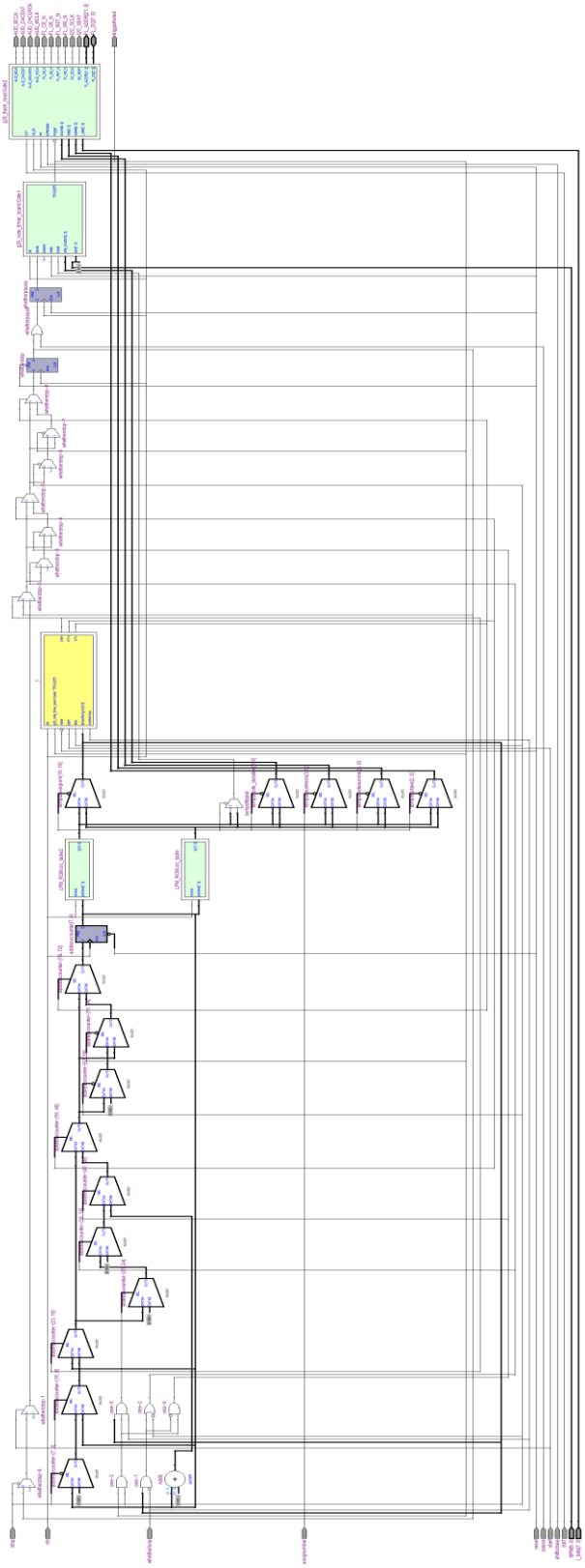
Chuan Qin 260562917
Wei Wang 260580783

Appendix B: Schematic diagram of entire music box with 4 segment displays (g26_lab5_FSM)

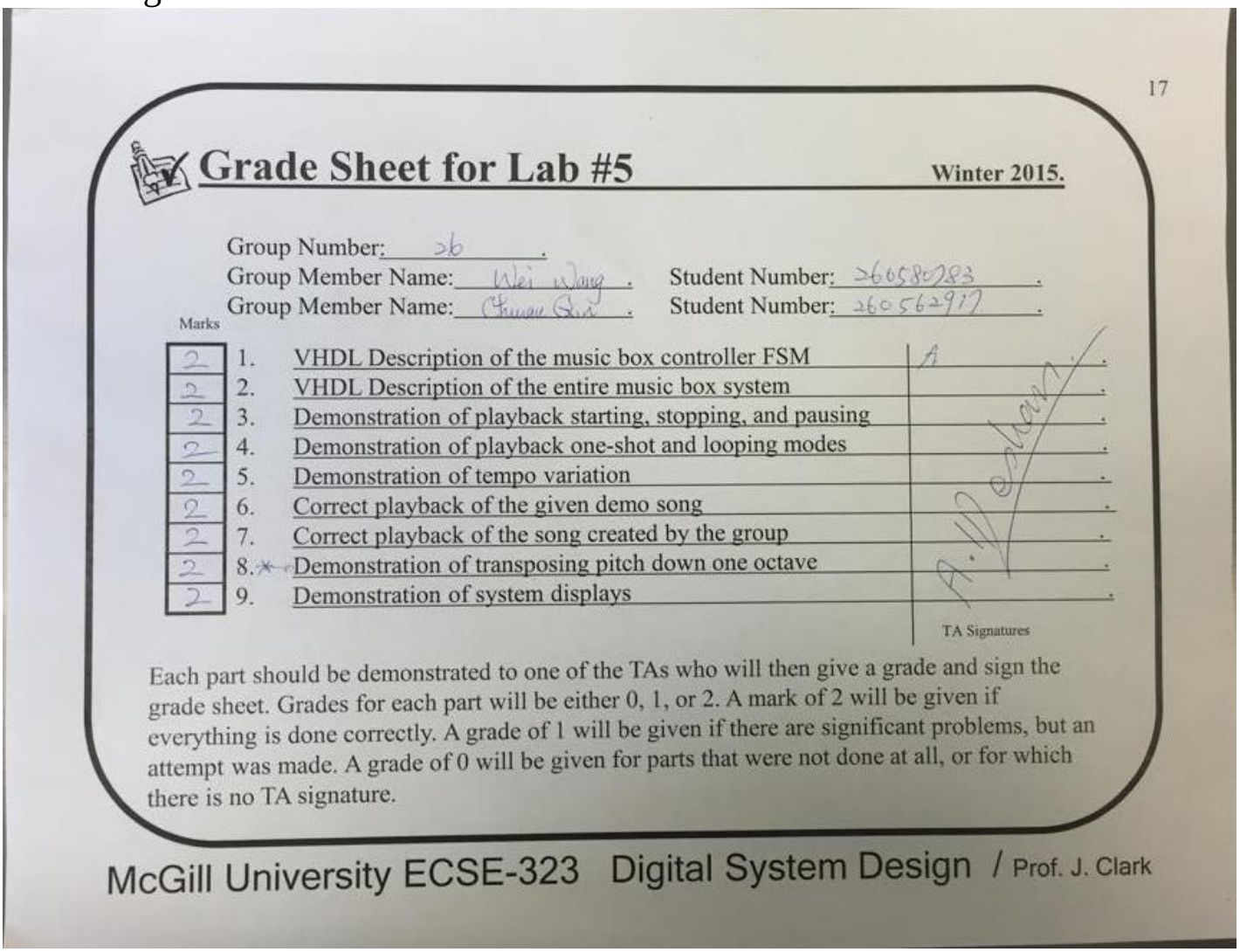


Chuan Qin 260562917
Wei Wang 260580783

Appendix C: Schematic diagram of entire music box without 4 segment displays (g26_lab5_FSM)



8. Grading Sheet



17

Grade Sheet for Lab #5 Winter 2015.

Group Number: 26 .
Group Member Name: Wei Wang . Student Number: 260580783 .
Group Member Name: Chuan Qin . Student Number: 260562917 .

Marks			TA Signatures
2	1.	VHDL Description of the music box controller FSM	A
2	2.	VHDL Description of the entire music box system	
2	3.	Demonstration of playback starting, stopping, and pausing	
2	4.	Demonstration of playback one-shot and looping modes	
2	5.	Demonstration of tempo variation	
2	6.	Correct playback of the given demo song	
2	7.	Correct playback of the song created by the group	
2	8.	Demonstration of transposing pitch down one octave	
2	9.	Demonstration of system displays	

Each part should be demonstrated to one of the TAs who will then give a grade and sign the grade sheet. Grades for each part will be either 0, 1, or 2. A mark of 2 will be given if everything is done correctly. A grade of 1 will be given if there are significant problems, but an attempt was made. A grade of 0 will be given for parts that were not done at all, or for which there is no TA signature.

McGill University ECSE-323 Digital System Design / Prof. J. Clark

9. References

1. J. Clark (2015), Lab 1 to 5 instruction, ECSE-323, McGill University.
2. C. Qin, W. Wang. (2015), Lab 3 and 4 reports, ECSE-323, McGill University.
3. (2009) The Quartus II TimeQuest Timing Analyzer, Altera Corporation
4. (2014) Altera University Program Flash Memory IP Core, Altera Corporation
5. (2006) Development and Education Board- User Manual, Altera Corporation
6. Castle in the sky, http://www.zhaogepu.com/jianpu/128314.html