ECE 492 Capstone Project

2013W

Virtual Guitar Gloves

Summary: Play a virtual guitar in the air by using two gloves fitted with tactile sensors and an accelerometer.

Group Members:

Elysia Jong	ejong@ualberta.ca
Qingyue Zhou	qingyue@ualberta.ca

Abstract

Terasic’s V.1.6 DE2_SD_Card_Audio demonstration was used as a base for this project. By default, it waited for an SD card to be inserted into the FPGA’s port, then played all the files within it in sequence through audio out. The files are assumed to be of .wav format with a 48 kHz sampling rate. It also displayed on the 7-segment and LEDs. The project had no documentation and the files were written in Verilog.

This demo project was modified to only play one file at a time, depending on which tactile sensor was pressed and only if the accelerometer detected a strum action from the user. The LCD was programmed to display whether the system was in note or chord mode and which octave scale was selected. The tactile sensors, in essence, were simple on/off switches with hardware debouncing implemented. The digital accelerometer communicates through the onboard GPIO pins to a second I2C bus, separated from the I2C bus used for the audio CODEC to avoid any possibility of conflict. All sounds played are pre-recorded files stored in the SD card. External components are all wired to the Altera DE2 via the GPIO pins.

Table of Content

Functional Requirement

Specification:

Our virtual guitar consists of 2 gloves. There are 7 tactile sensors on the left hand, which are used for 7 notes (A to G). There are 4 sensors on the right: 2 used to go up/down octaves, 1 to switch from playing notes to chords or vice versa, and a reset to the original mode (notes with default octave). There is also one accelerometer on the right hand to detect a strum from the player. By combining the glove's tap sensor and accelerometer signals, we can realize a guitar playing without an actual guitar.

The player is assumed to be right-handed. Once the player is touching their left thumb to a left sensor and a strum action from the right hand is detected, the speakers will then play the certain note/chord the player intended to play. This is meant to reflect the playing of an actual guitar, where the player would hold the fretboard with their left hand and play the strings with their right. If the player waves his/her right hand, but no tap sensor is engaged, nothing will happen.

The 2 gloves have a wired connection to the Altera DE2 FPGA board through the GPIO pins. Speakers are connected to Line Out for audio output. The guitar sounds (A-G, notes/chords) are saved in an SD card. Filtering is to be applied to the sound files to obtain the different octaves.

How to use:

1 <u>Playing a note:</u>
   - a user must press and hold their left thumb to 1 left finger note ("A" - "G")
      - i this is to mimic holding the fretboard on a real guitar
   - b user moves right hand downward in a strumming action
   - c simultaneous combination of both a. and b. required

2 <u>Changing between single notes and chords:</u>
   - a user needs to press right thumb to "MS" and release (no hold)

3 <u>Changing octaves:</u>
   - a user must press their right thumb (no hold) to either:
      - i ↓: octave down
      - ii ↑: octave up
   - b play a note normally as in 1.

4 <u>Resetting to default octave and note mode:</u>
   - a user needs to press right thumb to "R" and release (no hold)

Goals achieved:

The tactile sensors, combined with hardware debouncing, had no issues. The left hand sensors correspond to the correct notes/chords and the right hand's mode switch (for selecting notes/chords), octave up, octave down, and reset buttons were functional. The LCD also correctly displayed whether the user was in note or chord mode and which octave. The threshold on the accelerometer was set so that it would not recognize motion if the user moved their hand too slowly. Due to the significant amount of time it took to interface the accelerometer, the filtering used to achieve the octaves was not done. The original sound files were run through an audio editor called Audacity 2.0.3, where the frequencies were changed, new sound files generated and then stored in the SD card. This meant there were 42 files instead of the intended 14 files for guitar sounds.

The loading time required for the SD card is not noticeable. When running from flash memory, powering up the FPGA and instantly attempting to play a sound will work without any apparent delay. Originally, there was noise in the output sound. The saved .wav files are from an acoustic guitar, but the output from the FPGA sounded synthesized and in a minor scale. This was corrected by changing the Nios II processor clock from 50 MHz to 100 MHz.

Design and Description of Operation

There are 11 tap sensors for the 7 notes (A-G) and 4 modes (R, MS, Octave ↑, Octave ↓) placed as shown in the image below. Each sensor is located on a finger joint section, so that the player may bend their fingers. All unused finger joints may be used for future expansion. The accelerometer will be attached on the back of the right hand near the wrist.

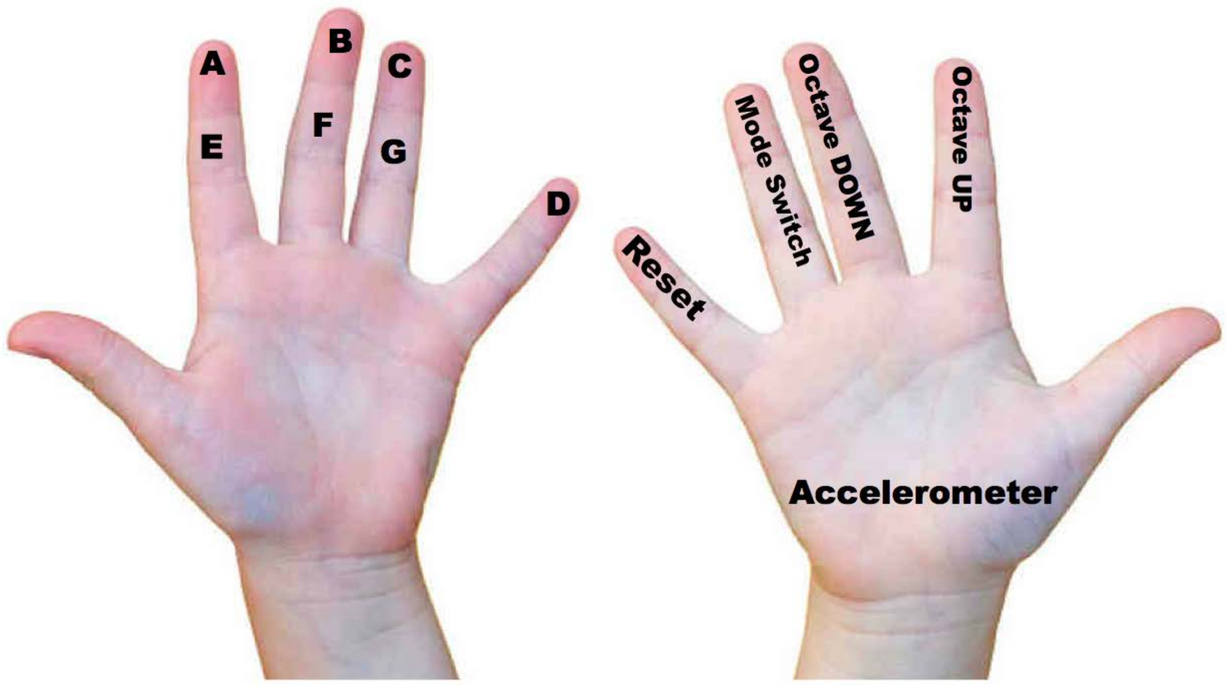


Image 1: Placement of sensors modified from http://media.npr.org/assets/news/2010/01/27/hands-ef12ac2473025791c19e4478b5ca0078c2fa1942-s6-c10.jpg

Flow chart:

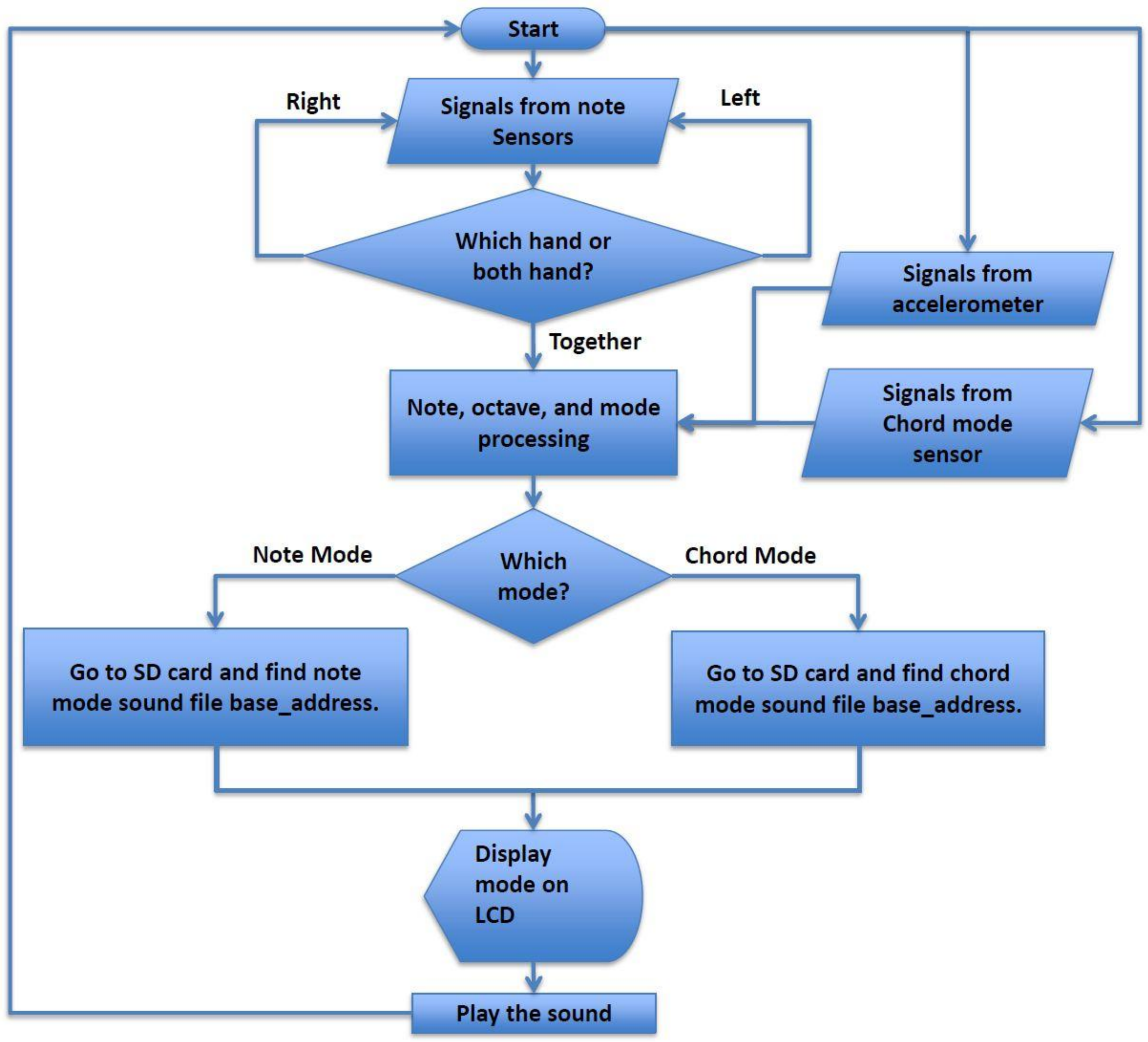

Start
Right
Signals from note Sensors
Left
Which hand or both hand?
Signals from accelerometer
Together
Note, octave, and mode processing
Signals from Chord mode sensor
Note Mode
Which mode?
Chord Mode
Go to SD card and find note mode sound file base_address.
Go to SD card and find chord mode sound file base_address.
Display mode on LCD
Play the sound

Image 2: Software design flow

Input to output functionality:

1 Tap sensor and accelerometer signals detected from GPIO pins
2 Based on input, read sound file from SD card
3 Output sound file to speakers connected to FPGA

Algorithms/Tasks:

"task_which_note" simply polls for user input on the left glove through an infinite while loop. It keeps track of the BASE_ADDRESS of the tactile sensors using IORD_PIO_DATA (BASE_ADDR,0) on the GPIO pins. The tactile sensors, using the internal ~25kohm resistors on the DE2, are wired to be active high. Therefore, once a 0 value is detected in the

BASE_ADDRESS, it corresponds to a tactile sensor being depressed. The sector that should be read from the SD card is then updated depending on which sensor it was. While polling is generally wasteful for CPU cycles, this system has no need to do any other critical computation or tasks other than to respond to the user's actions.

Interrupts and queues are used to handle the signals from the right glove, as these are used to change the modes of play available, versus the actual selection of notes. The thought was that these buttons will be less often pressed and so there is less need for polling. The current state of the system is displayed on the LCD by "taskModeDisplay."

"task_accelerometer" initializes the I2C bus as well as initializing the accelerometer registers. The values in these registers configure how it will function. For this project, it is set to detect freefall in the x-axis (1D) from the accelerometer's point of view. This corresponds to the axis perpendicular to the earth's surface.

"task_sd_play" gathers all the information from the other tasks and actually plays the .wav file from the SD card to the Line Out audio jack. The range of sectors to play, determined from "task_which_note," is read one sector at a time and stored in a FIFO buffer that is fed to audio out.

Bill of Materials

Other than the items listed below, a standard wall outlet (120 V, 60 Hz) is required to power the Altera DE2 board and external speakers if headphones are not used.

Qty	Part Name	Unit Cost (CAD)	Total Cost (CAD)
1	Altera/Terasic DE2 development board http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html	$517.72	$517.72
2	Ribbon cable (40 & 50-pin)	$10.00	$20.00
1	Perf board	$7.50	$7.50
1 roll	Wire wrap	$10.00	$10.00
4	Capacitors (0.1 uF)	$0.25	$1.00
1 kit	Resistors (used 4x 180 ohm, 4x 1 kohm)	$7.95	$7.95
2	Headers (40 & 50-pin)	$0.20	$0.40
1	Speakers/headphones	$20.00	$20.00
1	Thin wool gloves (pair)	$5.00	$5.00

1	2GB SD Card	$10.00	$10.00
1	**Accelerometer (MMA8452Q)** Digital output via I2C bus. Low voltage to interface (1.6-3.6V). ±2g/4g/8g 3-axis. Package is about the size of a quarter. 1.95-3.6V supply. 2 programmable interrupt pins. Supplier: https://www.sparkfun.com/products/10955 Datasheet: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Accelerometers/MMA8452Q.pdf	$9.95	$9.95
11	**Tactile sensor (450-1649-ND)** Non-illuminated, top-actuated, 6.00mm x 6.00mm, off-mom micro switch. Actuator height is 4.30mm with 0.25mm switch travel and through-hole mounting. Max contact rating at 0.05A @ 12VDC, minimum at 10uA @ 1VDC. Supplier: http://www.digikey.ca/product-detail/en/FSM2JH/450-1649-ND/1632535 Datasheet: http://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=1825910&DocType=Customer+Drawing&DocLang=English	$0.05	$0.55
	TOTAL		**$610.07**

Reusable Design Units

1. The Altera DE0-Nano board could be used in place of the Altera DE2. There is an accelerometer already built into the DE0 FPGA and it is only 3x2 inches compared to the 8x6 inches DE2. However, it does not have a audio out jack or an SD card port. A USB-bus can be attached to the USB port for more connections. The DE0 also only has a 2-pin external header for power which can be subject to a lot of movement, especially if the board is attached directly to our glove. The DE2 has many more features and functions that could be additions to our project, or they may be unused. If the latter, the DE0 may be a more appropriate choice.

2. The Terasic Technologies Inc.V.1.6 “DE2_SD_Card_Audio” demonstration project contains solutions for both the SD card and audio out. The SD card portion can be substituted with reusable design unit 3. The audio can be substituted with Altera University’s Audio IP Core. An application note has been written for it here: https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/audio_altera_university_ip_cores/

3. For the SD card, the application note “SD_Card_Interfacing” created by Jason Brown & Brady Thornton (Group 12) of Winter 2013 can be used as a substitute plan: https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/SD_card_interfacing/

Datasheet

User Perspective Diagram

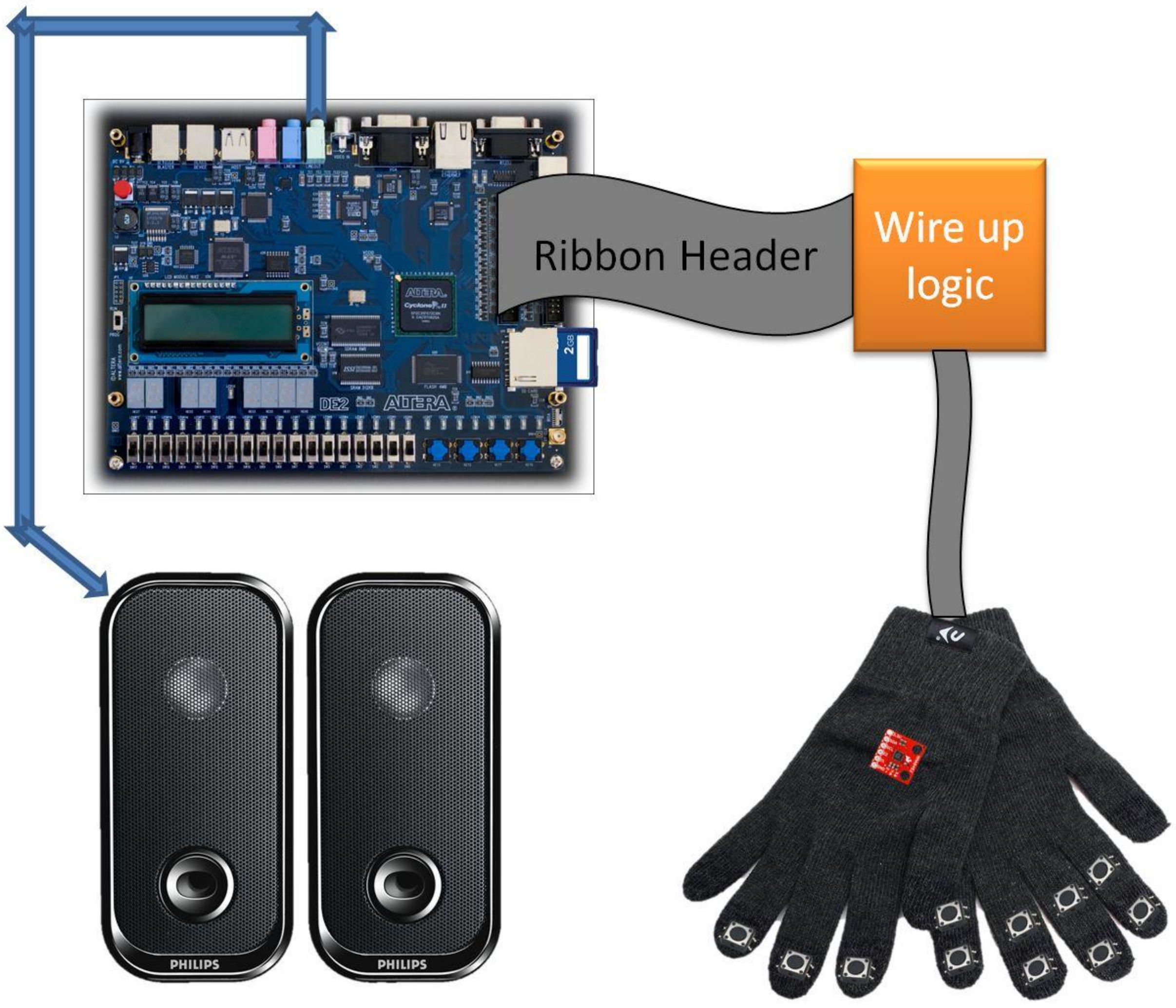


Image 3: User perspective diagram

User I/O Signals:

All GPIO inputs are driven by +3.3 V and are active low.

Signal	GPIO Pin	Description	Type

GPIO_0[10]	13	Note A tactile sensor on left index tip	Input
GPIO_0[12]	15	Note B tactile sensor on left middle tip	Input
GPIO_0[14]	17	Note C tactile sensor on left ring tip	Input
GPIO_0[16]	19	Note D tactile sensor on left pinky tip	Input
GPIO_0[18]	21	Note E tactile sensor on left index tip below Note A	Input
GPIO_0[20]	23	Note F tactile sensor on left middle tip below Note B	Input
GPIO_0[22]	25	Note G tactile sensor on left ring tip below Note C	Input
GPIO_0[6]	7	Reset tactile sensor on right pinky tip	Input
GPIO_0[4]	5	Mode Switch tactile sensor on right ring tip	Input
GPIO_0[0]	1	Octave Up tactile sensor on right index tip	Input
GPIO_0[2]	3	Octave Down tactile sensor on right middle tip	Input
GPIO_0[11]	14	SDA of accelerometer (communicate with I2C bus)	Input
GPIO_0[13]	16	SCL of accelerometer (communicate with I2C bus)	Output
GPIO_0[15]	18	INT2 of accelerometer (communicate with I2C bus)	Input
SD Card	N/A	Holds .wav files	Input
Line-Out	N/A	Audio out (green)	Output
LCD	N/A	Display the current mode (note/chord)	Output
7-Segment	N/A	SD card sector display (right-most 4)	Output

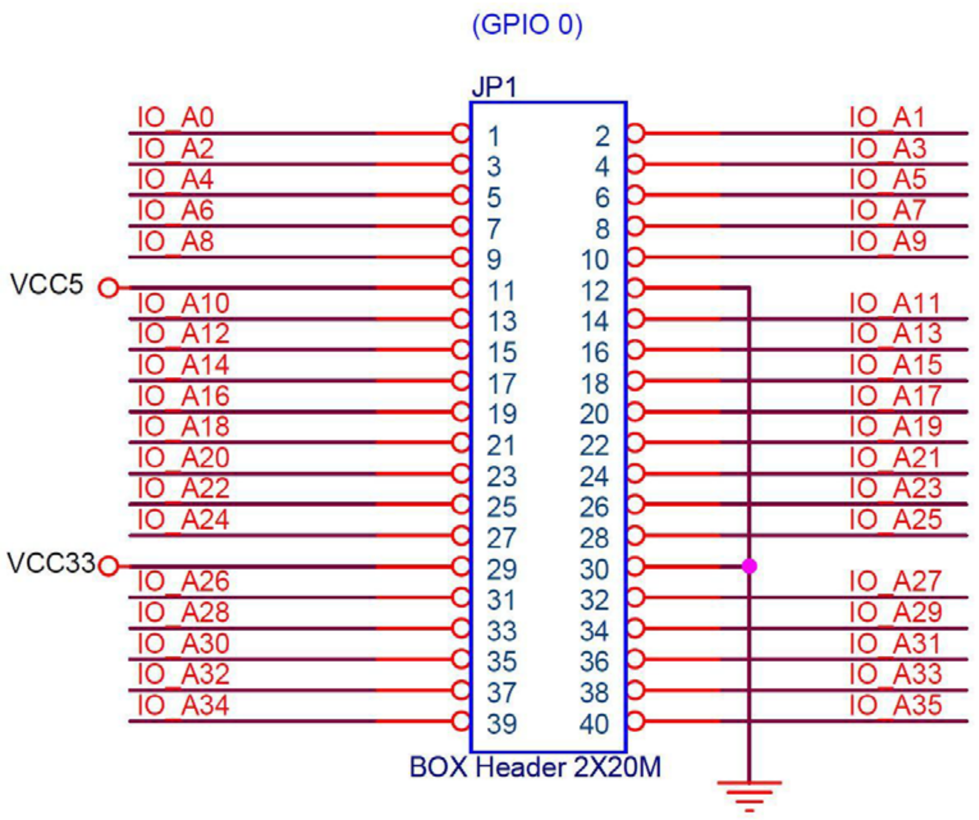
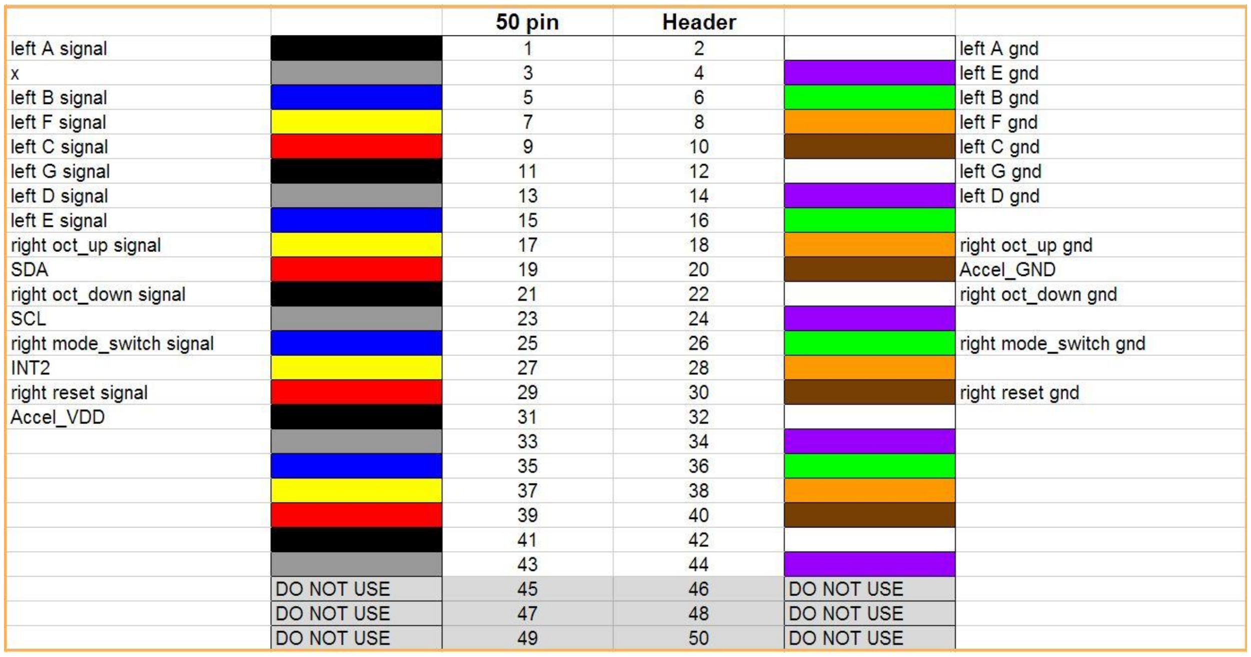


Image 4: GPIO pin schematic

Ribbon Header Mapping



		50 pin	Header		
left A signal	⬛	1	2	⬜	left A gnd
x	▨	3	4	🟪	left E gnd
left B signal	🟦	5	6	🟩	left B gnd
left F signal	🟨	7	8	🟧	left F gnd
left C signal	🟥	9	10	🟫	left C gnd
left G signal	⬛	11	12	⬜	left G gnd
left D signal	▨	13	14	🟪	left D gnd
left E signal	🟦	15	16	🟩	
right oct_up signal	🟨	17	18	🟧	right oct_up gnd
SDA	🟥	19	20	🟫	Accel_GND
right oct_down signal	⬛	21	22	⬜	right oct_down gnd
SCL	▨	23	24	🟪	
right mode_switch signal	🟦	25	26	🟩	right mode_switch gnd
INT2	🟨	27	28	🟧	
right reset signal	🟥	29	30	🟫	right reset gnd
Accel_VDD	⬛	31	32	⬜	
	▨	33	34	🟪	
	🟦	35	36	🟩	
	🟨	37	38	🟧	
	🟥	39	40	🟫	
	⬛	41	42	⬜	
	▨	43	44	🟪	
	DO NOT USE	45	46	DO NOT USE	
	DO NOT USE	47	48	DO NOT USE	
	DO NOT USE	49	50	DO NOT USE	

Image 5: Ribbon Header Mapping

Accelerometer Interface:

Signal	Description	Type
sda_padoen_o	data enable	in
scl_i2c_to_accel	i2c clock to accelerometer	out
sda_i2c_to_accel	i2c data to accelerometer	inout
scl_i2c_to_bus	i2c clock to i2c bus	in
sda_i2c_to_bus	i2c data to i2c bus	inout
interrupt2	int2 on accelerometer	in
avalon_slave_address	avalon_slave.address	in
avalon_slave_readdata	avalon_slave.readdata	out
avalon_slave_writedata	avalon_slave.writedata	in
avalon_slave_write_n	avalon_slave.write	in
avalon_slave_chipselect	avalon_slave.chipselect	in
avalon_slave_byteenable	avalon_slave.byteenable	in
avalon_slave_read	avalon_slave.read	in

SD Card Interface:

Signal	Description	Type
Data2	Data signal 1	out
Data3	Data signal 2	out
CMD I/O	Input and output command	inout
GND	Supply voltage negative	in
VDD	Supply voltage positive	in
CLK	Clock signal	in
Data0	Data signal 0	out
Data1	Data signal 1	out

External Circuitry Schematic:

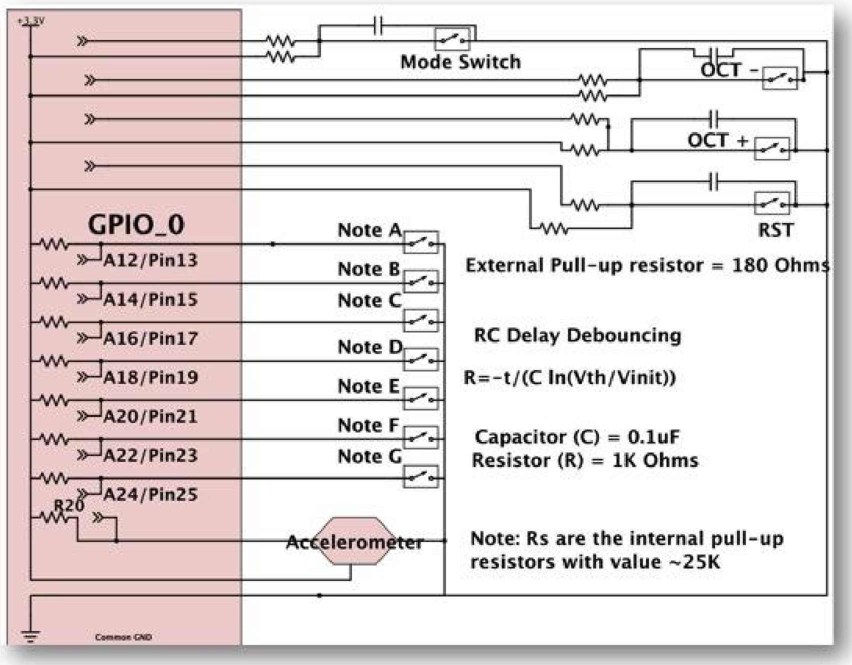


Image 6: Schematic

Power:

The below values were obtained with a DE2 power measurement wiring harness connected to the system:

Idle: 0.447A, 9.10V
Peak: 0.457A, 9.10V
Power $= I^2R = (0.447 \text{ to } 0.457)^2\ 9.10 = 1.818 \text{ to } 1.901$ [W]

Background Reading

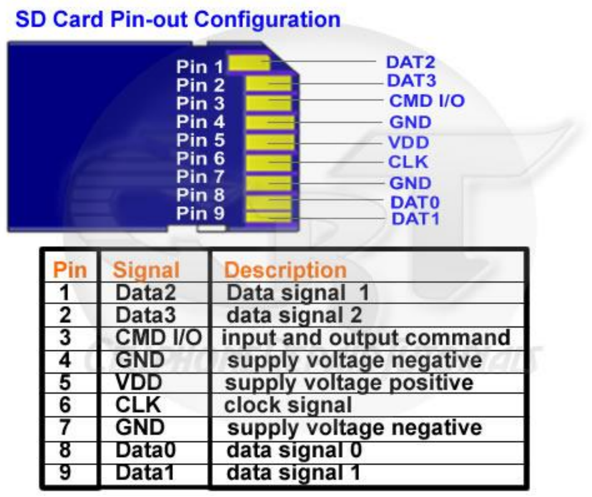
1. Implementation of SD card music player using Altera DE2-70:
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5957486&contentType=Conference+Publications&searchField%3DSearch_All%26queryText%3DAltera+de2

This article is dedicated to show the implementation of a music player build based on Altera DE2-70 board. It shows us very detailed design procedures and requirements. Although it is built on a different board, we can still get some decent ideas from it. For instance, their project is using I2C protocol to configure the audio chip working in master mode. The SD card must be 16-bit FAT format system, and for each sector, it has 512 bytes data. Therefore the main program is going to read 512 bytes each time, write the data to the DAC FIFO in the audio controller, and then enter the next loop. In our design, we should know base/starting address of each sound file, and how many loops we are going to read for each of them.

2. The Reading/Writing SD Card System Based on FPGA

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5635615&contentType=Conference+Publications&searchField%3DSearch_All%26queryText%3DSD+card

The information provided in this article explains how to interface the SD card to an FPGA, specifically the Altera DE2-70 using QuartusII and NiosII. The pins of the SD card are as follows:

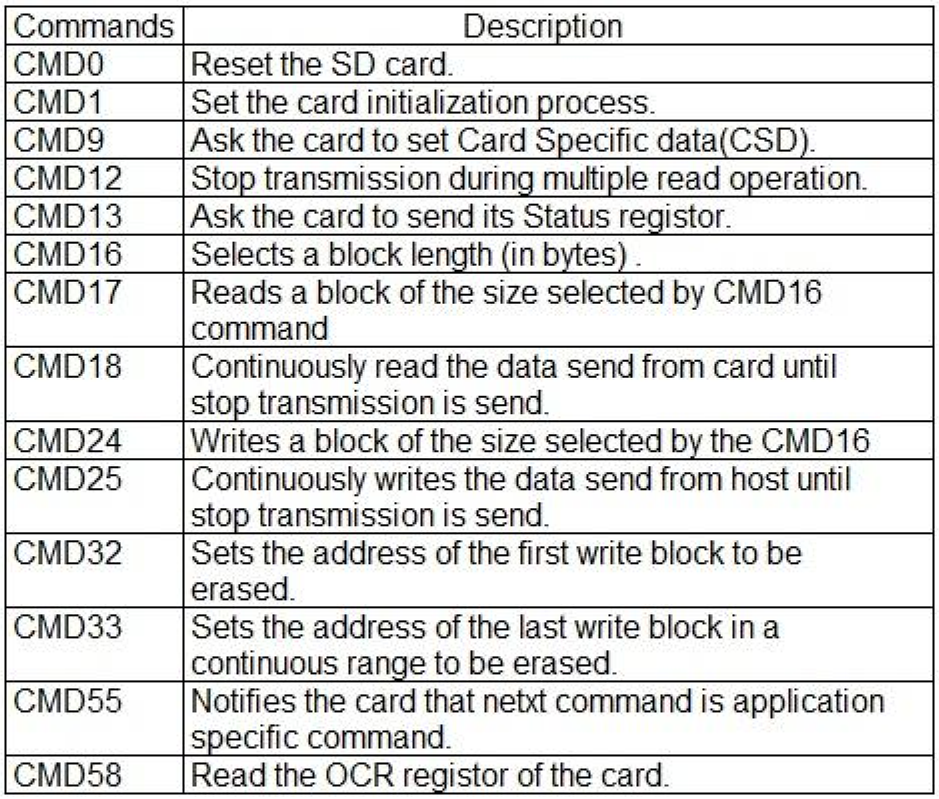


Pin	Signal	Description
1	Data2	Data signal 1
2	Data3	data signal 2
3	CMD I/O	input and output command
4	GND	supply voltage negative
5	VDD	supply voltage positive
6	CLK	clock signal
7	GND	supply voltage negative
8	Data0	data signal 0
9	Data1	data signal 1

Image7: SD card interface

http://3.bp.blogspot.com/_8JZhVVmpICU/TH_Pxa19MHI/AAAAAAAAApg/pgSppwx0gY8/s1600/SD+card+pinout.jpg

These pins are interfaced in Quartus II as memory-mapped slaves. The article also discusses the FAT16 file system as well as the commands for the SD card operation:



Commands	Description
CMD0	Reset the SD card.
CMD1	Set the card initialization process.
CMD9	Ask the card to set Card Specific data(CSD).
CMD12	Stop transmission during multiple read operation.
CMD13	Ask the card to send its Status registor.
CMD16	Selects a block length (in bytes) .
CMD17	Reads a block of the size selected by CMD16 command
CMD18	Continuously read the data send from card until stop transmission is send.
CMD24	Writes a block of the size selected by the CMD16
CMD25	Continuously writes the data send from host until stop transmission is send.
CMD32	Sets the address of the first write block to be erased.
CMD33	Sets the address of the last write block in a continuous range to be erased.
CMD55	Notifies the card that netxt command is application specific command.
CMD58	Read the OCR registor of the card.

http://dcodelab.files.wordpress.com/2012/11/command.jpg

Furthermore, SD card initialization, data reading, data writing, and data erasing, were explained. The information was extremely similar to what we are doing for our project (based of off Terasic's demo.) and multiple example projects we have found have used the same methods for SD card communication (1-bit SPI). Most examples only did reading, so the data writing and data erasing details could possibly be utilized for further functional expansion of our guitar gloves.

3. 基于 FPGA 的 I_2C 控制器的实现及其在音频解码中的应用 (The application of I2C controller, based on FPGA, in audio codec perspective)

http://read.pudn.com/downloads143/doc/comm/624483/基于 FPGA 的 I_2C 控制器的实现及其在音频编解码中应用.pdf

In our project, we are playing sound files from the SD card to the audio codec, and the transmission is realized by I2C bus. This article provides the information about how to realize communication by using I2C bus in DE2 board. It explains how I2C controller work in detail by commenting the verilog codes provided by Altera. Thus this article is really helpful for us to understand the functionality, and performance of I2C bus so that we can embed our accelerometer into our design.

4. Altera Quartus II Tutorial

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6278363&contentType=Books+%26+eBooks&searchField%3DSearch_All%26queryText%3DAltera+Quartus+II+Tutorial

This is a tutorial article for Altera Quartus II. It includes several functions of Quartus II. We found it is very useful under the section "B6. Making Pin Assignment". It helped us to complete our application notes, which is about applying internal weak pull-up resistor in Quartus II.

Software design

The software design is based on Nios II, uC/OS-II. All the signals are coming from the GPIO on the DE2 board, and interrupt handlers are used to deal with the different signals. In addition, we need 2 tasks to manage the filtering and sound playing. The following is the task and interrupt descriptions:

Interrupt name	**Description**
isr_Rst	It will handle the RESET signal comes from the right pinky tip sensor. Inside this interrupt handler, the mode, octave flags would be reseted to default values. Post a message queue, which TaskModeDisplay would pending, so that TaskModeDisplay can display correct MODE and OCTAVE LEVEL on the LCD.
isr_Octup	Increasing the octave level by 1. Post a message queue, which TaskModeDisplay would pending, so that TaskModeDisplay can display correct MODE and OCTAVE LEVEL on the LCD.
isr_Octdown	Decreasing the octave level by 1. Post a message queue, which TaskModeDisplay would pending, so that TaskModeDisplay can display correct MODE and OCTAVE LEVEL on the LCD.
isr_ModeSwitch	In this interrupt handler, it would raise a flag for a specific MODE and post a message queue, which TaskModeDisplay would pending, so that TaskModeDisplay can display correct MODE on the LCD.

Table 2: Interrupt description

Task name	**Description**
task_which_note	This task has task priority 1. Pending on a message queue, which would be posted once the strumming signal detected by the accelerometer. It also monitor which note/chord in which octave level the user pressed, if a strumming signal detected without any note/chord pressing, then do nothing.
task_sd_play	This task has task priority 4. Its main responsibility is going to receiving

	all the messages from other tasks and analyze the signals to perform the next correct move. It is going to monitor the sensors' signals from the player's left hand by polling. There are seven tap sensors/switches on the left hand glove. We could distinguish all those signals by assigning them to different pins on the GPIO on DE2 board. It must be able to detect the "useless" signals from other tasks. For instance, if the player only press one of the note sensors on left hand glove without any strumming movement on the right hand glove, SD_Play should never play a note or chord in this case. The SD card is formatted as 16-bit FAT, and it has 512 bytes for each segment of memory.
TaskModeDisplay	This task has task priority 2. It display whatever current mode and octave level on the LCD.
task_acceleromete r	This task has task priority 3. This task would keep monitoring the status of the accelerometer. If it detects the strumming signal, then it would post to a message queue, which is pending in task_which_note.

Table 3: Task description

BLOCK DIAGRAM:

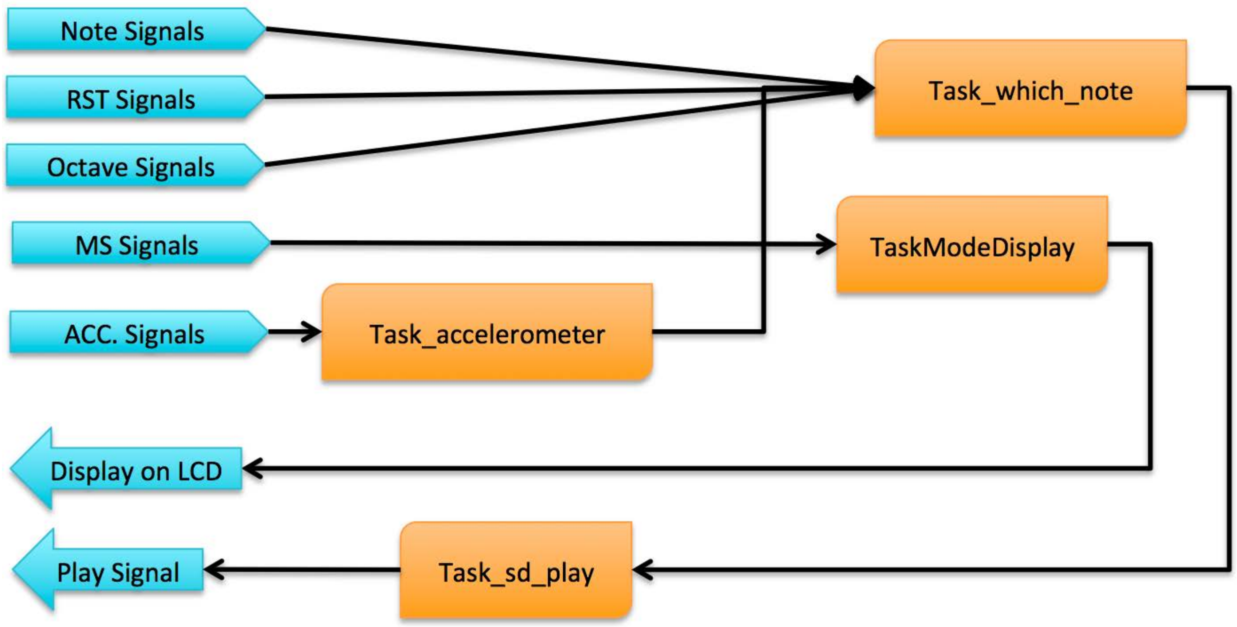


Image 8: Software block diagram

List of Libraries:

Name	**Description**
"altera_avalon_pio_regs.h"	Provides a hardware interface that allows software to access the PIO.
"altera_avalon_lcd_16207_regs.h" "altera_avalon_lcd_16207.h"	Provides a hardware interface that allows software to access the two (2) internal 8-bit registers in an Optrex model 16207 (or equivalent) character LCD display (the kind shipped with the Nios Development Kit, 2 rows x 16 columns).
"basic_io.h"	Basic I/O.
"LCD.h"	User defined functions/macros for LCD. For example, LCD_Init(); lcd_wirte_cmd(base, data).

"SD_Card.h"	User defined functions/macros for SD Card, and some commands for SD Card.
	The macros contains SD card set I/O directions, output High/Low, and input read. Functions:
	SD_card_init(void)//initialize SD card.
	SD_read_lba(BYTE *buff, UINT32 lba, UINT32 seccnt)//reads seccnt number of sector in SD card.
	send_cmd(BYTE *in)//sends the commands *in.
	Predefined commands:
	const int cmd0 [5] = {0x40,0x00,0x00,0x00,0x00}; /* Reset SD Card */
	const int cmd55[5] = {0x77,0x00,0x00,0x00,0x00}; /* Next CMD is ASC */
	const int cmd2 [5] = {0x42,0x00,0x00,0x00,0x00}; /* Asks to send the CID numbers */
	const int cmd3 [5] = {0x43,0x00,0x00,0x00,0x00}; /* Send RCA */
	const int cmd7 [5] = {0x47,0x00,0x00,0x00,0x00}; /* Select one card, put it into Transfer State */
	const int cmd9 [5] = {0x49,0x00,0x00,0x00,0x00}; /* Ask send CSD */
	const int cmd10[5] = {0x4a,0x00,0x00,0x00,0x00}; /* Ask send CID */
	const int cmd16[5] = {0x50,0x00,0x00,0x02,0x00}; /* Select a block length */
	const int cmd17[5] = {0x51,0x00,0x00,0x00,0x00}; /* Read a single block */
	const int acmd6 [5] = {0x46,0x00,0x00,0x00,0x02}; /* SET BUS WIDTH */

	const int cmd24[5] = {0x58,0x00,0x00,0x00,0x00}; /* Write a single block */ const int acmd41[5] = {0x69,0x0f,0xf0,0x00,0x00}; /* Active Card's ini process */ const int acmd42[5] = {0x6A,0x0f,0xf0,0x00,0x00}; /* Disable pull up on Dat3 */ const int acmd51[5] = {0x73,0x00,0x00,0x00,0x00}; /* Read SCR(Configuration Reg) */ note: we are not using all of these commands.

Test plan

Software:

Test	**Complete/Results**
Detect SD card	Yes
Detect speakers	Yes
Memory leaks after 30 minutes	Yes - No leaks
Polling works/sufficient on A-G signals	Yes - Immediate response
ISR works on single press signals (chord change, octave up/down, reset)	Yes
ISR works on accelerometer	Yes
Able to flash project to board	Yes

Table 5: Software test plans

Hardware:

Test	**Complete/Results**
Notes A-G play	Yes
Chords A-G play	Yes
Increase octave works on all notes	Yes

Decrease octave works on all notes	Yes
Reset sends system to note mode and default octave	Yes
No sound plays if no tactile sensors are engaged but accelerometer detects	Yes
No failure if user holds the chord/up octave/down octave/reset buttons instead of pressing them	Yes
Playing one note and another in quick succession should cut off the first note (no overlap)	Yes
Response time and performance	Yes - Note selection is instant to a human's reaction time. Accelerometer threshold may need to be varied slightly per user for comfort.

Table 6: Hardware test plans

Results of Experiments and Characterization

Strum Detection:

Due to the difficulty in interfacing the accelerometer, a backup distance sensor was used in its place temporarily. The sensor had a maximum 30 cm range for detection of movement. It was still small and light enough for the purpose of this system, but had to be positioned more specifically to catch the user's right arm strums. There is also the option of attaching the sensor to the player's waist. An issue is that sensor will not differentiate between up and down arm movements, and so a user may unintentionally play twice as they raise their arm to strum again. However, the delay required between detections is extremely small and not noticeable even when one is waving their arm as fast as possible.

In the end, the accelerometer was chosen as the detection device because it had a myriad of extra features the distance sensor did not. This is discussed in the Appendix - Future Work. The accelerometer also did not need special positioning or have any range issues as it was attached and wired directly onto the right glove. For the I2C interfacing to work with the DE2 board, however, there is a 150 ms delay between the interrupts fired upon detection.

CPU Clock/Performance:

A major issue noted during testing was that the sound coming out from the speakers was distorted. For example, what should have been a major "C" note came out as a minor "A." The .wav files were acoustic guitar sounds, but most notes sounded synthesized. The audio CODEC was configured for a 48 kHz playback, and so the .wav files were generated as such. Changing the playback frequency on both the files and the CODEC either made the output worse, or did not have any noticeable effect. The FIFO buffer size for playing the sounds was also investigated. No overflow/underflow was found and changing the size of the buffer had no noticeable effect.

The Nios II processor was running on a 50 MHz clock. It was found that using a 100 MHz clock for the CPU and all components, except for the SDRAM, solved the audio distortion issue. Also, when flashing the project to the DE2 board, all *_syslib items must go to SDRAM. Placing the .text and .rodata sections into flash will significantly slow down the audio output.

References

[1] E. Lunty, K. Brooks, P. Roland. "Audio_Codec_G2/." Internet: http://www.ece.ualberta.ca/~elliott/ece492/appnotes/2012w/Audio_Codec_G2/, Apr. 13, 2012 [Jan. 28, 2013].

[2] Altera Corporation. "Altera University Program Secure Data Card IP Core." Internet: ftp://ftp.altera.com/up/pub/Altera_Material/12.0/University_Program_IP_Cores/Memory/SD_Card_Interface_for_SoPC_Builder.pdf, May 2012 [Jan. 26, 2013].

[3] Altera Corporation. "Audio Core for Altera DE-Series Boards." Internet: ftp://ftp.altera.com/up/pub/Altera_Material/12.0/University_Program_IP_Cores/Audio_Video/Audio.pdf, May 2012 [Jan. 26, 2013].

[4] Altera Corporation. "Audio/Video Configuration Core for DE-Series Boards." Internet: ftp://ftp.altera.com/up/pub/Altera_Material/12.0/University_Program_IP_Cores/Audio_Video/Audio_and_Video_Config.pdf, May 2012 [Jan. 26, 2013].

[5] R. Wong, V. Santhanagopalan. "ECE 5760 - Final Project Music Player." Internet: http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/f2010/vs327_rw363/WAV_player/ECE%205760.htm, 2010 [Jan. 27, 2013].

[6] Z. Lai, Z. Liu, M. Li, Q. Yuan. "MP3 Player." Internet: http://www.cs.columbia.edu/~sedwards/classes/2010/4840/reports/KH.pdf, 2010 [Jan. 27, 2013].

[7] Terasic Technologies. "DE2 Music Synthesizer Source Code." Internet: http://www.terasic.com.tw/cgi-

bin/page/archive.pl?Language=English&CategoryNo=165&No=30&PartNo=4, Nov. 16, 2006 [Feb. 2, 2013]

[8] N. Minderman. “niosII_microc_lab1.vhd.” Internet: https://eclass.srv.ualberta.ca/mod/resource/view.php?id=529057, Jan. 8, 2013 [Jan. 26, 2013]

[9] Terasic Technologies. “DE2_system_v1.6.zip.” Internet: https://eclass.srv.ualberta.ca/mod/resource/view.php?id=232128, July 19, 2006 [Feb. 3, 2013]

[10] C. Smart, T. Davis. “G8_Accelerometer/.” Internet: https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/G8_Accelerometer/, Mar. 25, 2013 [Apr. 10, 2013]

[11] The Audacity Team. “Audacity 2.0.3” Internet: http://audacity.sourceforge.net/about/credits, Mar. 20, 2013 [Jan. 21, 2013]

Appendices

Quick Start Manual

Hardware:

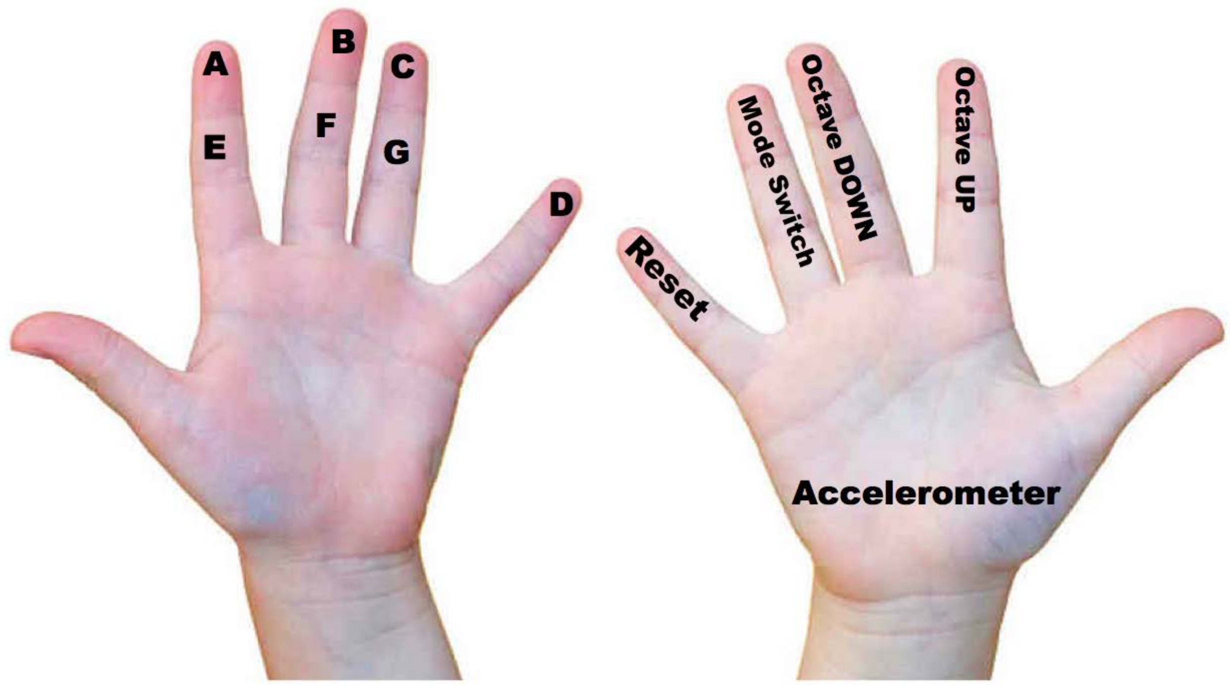


Image 9: Placement of sensors modified from http://media.npr.org/assets/news/2010/01/27/hands-ef12ac2473025791c19e4478b5ca0078c2fa1942-s6-c10.jpg

Overview:

Left-hand: 7 notes/chords (depending on which MODE) A-G.
Right-hand:

- Reset: reset to default setting (note mode with 0 octave).
- Mode Switch: switch between note and chord mode.
- Octave Up: increasing the octave by 1 level.
- Octave Down: decreasing the octave by 1 level.
- Accelerometer: strumming detection.

Steps to start from flashed board:

1. Plug in power block to Altera DE2 Board.
2. Connect the 40-pin (grey) header to the GPIO_0 port on the DE2 board.
3. Connect the speaker to the line-out port on the DE2 board and power on the speaker.
4. Turn on the DE2 board (red button).
5. Follow steps in “Overview” and “How to play” sections

Software:

Steps for non-volatile version:

1. Navigate to the directory of this project.
2. Run ./scripts/launch_quartus.sh in the terminal.
3. Compile the code in Quartus task section.
4. Flash the compiled .sof file to the DE2 board (JTAG port).
5. Run ./scripts/launch_niosII.sh in the terminal.
6. Run the project as NIOS II Hardware.

Steps for volatile version:

1. Navigate to the directory of this project.
2. Run ./scripts/launch_quartus.sh in the terminal.
3. Compile the code in Quartus task section.
4. Turn the board off and move switch 19 on the DE2 from RUN to PROG

4. Flash the compiled .pof file to the DE2 board (Active Serial Programming).

5. After the DE2 has been flashed, turn off the board.

6. Move SW19 to RUN and turn on the DE2 board again.

7. Run ./scripts/launch_niosII.sh in the terminal.

8. In *_syslib, ensure all components in System Library - Linker Script are pointing to SDRAM

9. Flash the project to the Cyclone-II processor

How to play:

Steps are the same for both non-volatile and volatile versions:

1. Put on the gloves - accelerometer is on the right hand.
2. Hold one of the notes/chord (depending on which mode) on the left hand.
3. Strum the right hand, there should be output to the speaker.
4. Press once the octave up/down to increase/decrease the octave.
5. Press reset once to reset the guitar to default setting (note mode with 0 octave level).
6. Enjoy the virtual guitar gloves.

Future Work

Backup Plans/Options:

There was much difficulty with interfacing the digital accelerometer (MMA8452Q). It communicates to an I2C bus, of which there is one on the DE2 board. This bus is physically wired to the audio CODEC (Wolfson WM8731) on the board, which uses I2C as well. As including an external component to the I2C bus proved to be troublesome, a distance sensor

(SHARP GP2D15 F 52) was successfully implemented as a backup. This sensor requires a 5V power supply and outputs a digital signal (0 for non-detection, 1 for detection). Thus, it was very easy to interface and power it through the GPIO pins on the DE2. The accelerometer was still chosen as the main option for strumming detection as it provided much more features that can be used to extend upon this project.

Due to the accelerometer delay, there was not enough time to implement filtering on the saved .wav files to achieve different octaves. Doing so would have saved much more space in the SD card. The backup plan implemented was to use an audio editor (Audacity 2.0.3) to change the frequency of the original files and generate new .wavs to be saved.

Extensions:

For further development, a wireless implementation can be applied for the two gloves via Bluetooth or some other transmitter. Additional features can include modes for sharp and/or flat notes, different instrument sounds, and consideration for left-handed players. The accelerometer has many more features that can be taken advantage of as well. For example, it is capable of detecting negative and positive acceleration, which can be used to differentiate between an up-strum and a down-strum. These sound different on a guitar. Future work can aim to accommodate having multiple note buttons pressed at once. One may wish to also completely remove the need for tactile sensors and use some sort of resistive or capacitive touch detection.

Hardware documentation

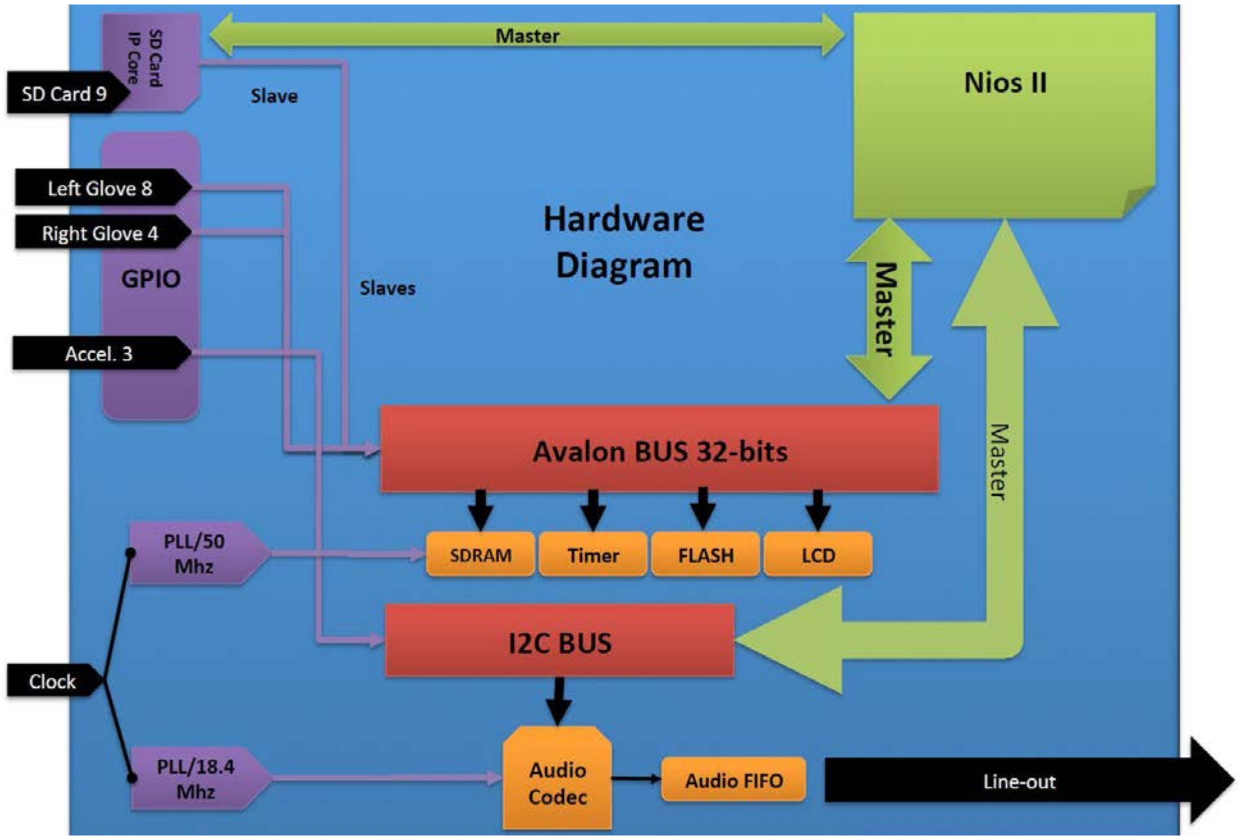


Image 10: Hardware block diagram

Source code section:

Block diagram of interactions between processes:

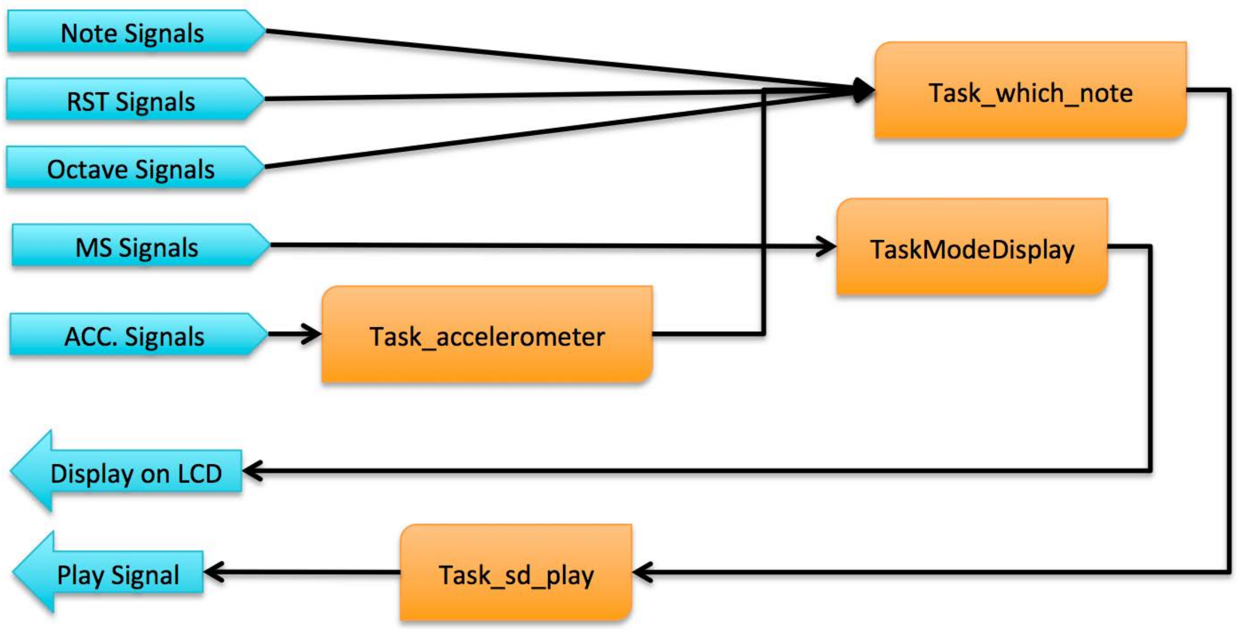


Image 11: Software block diagram

Index to source code:

Status: -**N**ot compiled successfully
-**C**ompiled without errors
-**E**xecuted or otherwise demonstrated
-**T**ested and passed

Source File (Created)	**Description**	**Status**
hello_ucosii.c	Main C file modified from Terasic's "DE2_SD_Card_Audio." Plays particular sound on particular tap sensor press and accelerometer signal.	T
basic_io.h	From Terasic's demo. Holds functions for GPIO, 7-segment display, and sleep/delays.	T
LCD.h/LCD.c	From Terasic's demo. Contains code for initializing and controlling the LCD.	T
SD_Card.h	From Terasic's demo. Contains code for initializing and reading from the SD card.	T

terasic_includes.h	From Terasic's demo. Links to other header files.	T
i2c_ctrl.h/i2c_ctrl.c	From Reference [10], controller for the accelerometer I2C bus.	T
accelerometer.h/ accelerometer.c	From Reference [10], controller for the accelerometer.	T
DE2_SD_Card_Audio.v	Top level Verilog file modified from Terasic's "DE2_SD_Card_Audio."	T
opencores_i2c_master.vhd	Top level VHDL file of i2c bus modified by Troy.	T

Table 7: Created source files

Source File (Auto-Generated)	**Description**	**Status**
Accel_Control.v	SOPC builder - accelerometer component	T
Audio_0.v	from Terasic	T
AUDIO_DAC_FIFO.v	from Terasic	T
Audio_PLL.v	from Terasic	T
button_pio.v	from Terasic	T
clock_0.v	from Terasic	T
clock_1.v	from Terasic	T
cpu_0.v	from Terasic	T
FIFO_16_256.v	from Terasic	T
I2C_AV_Config.v	from Terasic	T
I2C_Controller.v	from Terasic	T
I2C_to_GPIO.v	from Terasic	T
jtag_uart_0.v	from Terasic	T
MReset.v	SOPC builder - reset tap sensor	T
MS.v	SOPC builder - mode switch tap sensor	T
noteA.v	SOPC builder - tap sensor	T

noteB.v	SOPC builder - tap sensor	T
noteC.v	SOPC builder - tap sensor	T
noteD.v	SOPC builder - tap sensor	T
noteE.v	SOPC builder - tap sensor	T
noteF.v	SOPC builder - tap sensor	T
noteG.v	SOPC builder - tap sensor	T
Oct_DOWN.v	SOPC builder - octave down tap sensor	T
Oct_UP.v	SOPC builder - octave up tap sensor	T
SD_CLK.v	from Terasic	T
SD_CMD.v	from Terasic	T
SD_DAT.v	from Terasic	T
sdram_0.v	from Terasic	T
SDRAM_PLL.v	from Terasic	T
SEG7_Display.v	from Terasic	T
SEG7_LUT.v	from Terasic	T
sram_0.v	from Terasic	T
SRAM_16Bit_512K.v	from Terasic	T
switch_pio.v	from Terasic	T
system_0.v	from Terasic	T
timer_0.v	from Terasic	T
system_0_clock_0.v	from Terasic	T
onchip_memory2_0.v	SOPC builder - onchip memory.	T

Table 8: Auto-generated source files