

ECE 385

Spring 2022

Final Project - Lab Report

FPGA Sound Board

Ritvik Gandesiri & Ankit Veerendra

ritvikg2 & ankitv2

Introduction

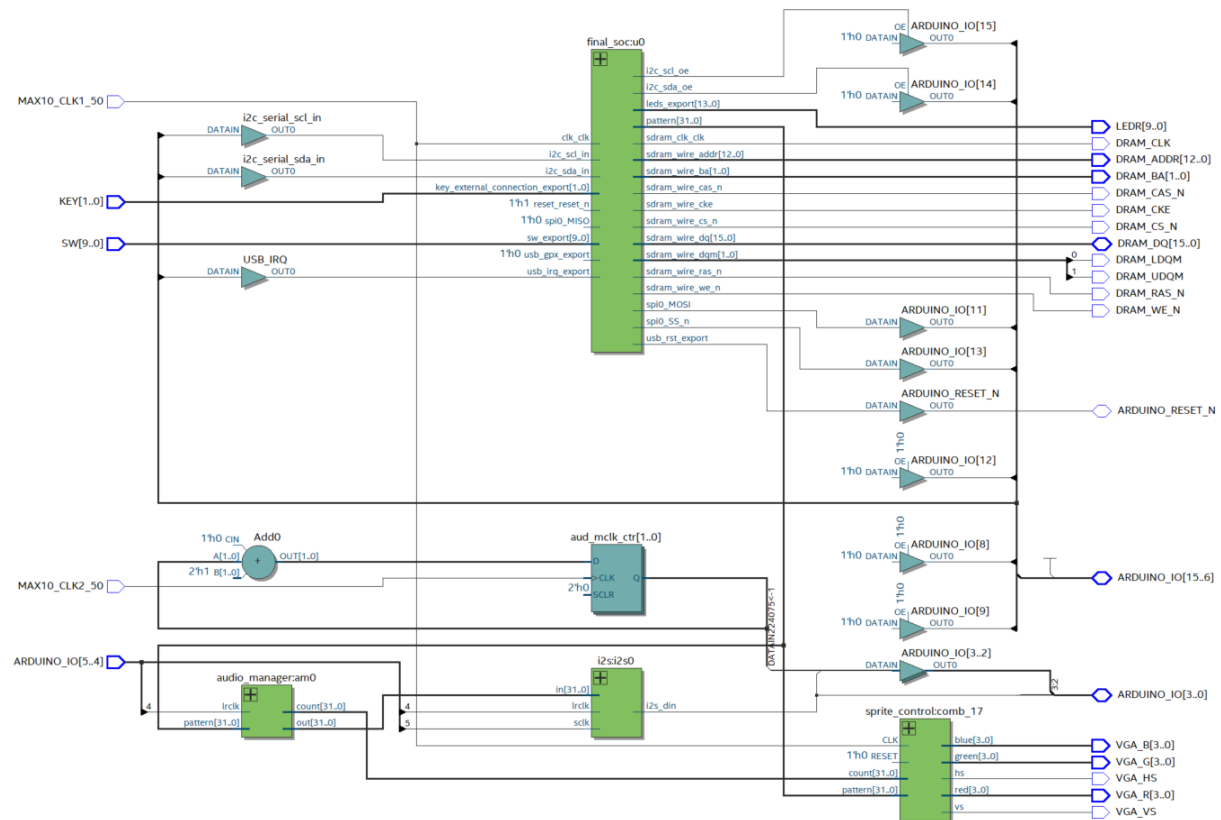
In this project, we attempt to make a rudimentary drum kit audio sequencer, in the vein of GarageBand, using solely the FPGA. The left two switches on the board can be used to select between 1 of 4 sounds, and the remaining 8 dictate the count at which they will be played. Pressing the top button on the FPGA saves this pattern to this sound. And you will immediately begin to hear it play. On the screen, there is a grid of 4x8 colored cells that show the current pattern for each sound at the same time, and as the audio loops across the 8 counts, a yellow bar scrolls across the screen to show you where in the loop you are playing. When you want to reset all, press the bottom button on the FPGA and start all over again.

When you make a selection with the switches, and press the button to save it, it is stored on to registers on the FPGA as 32 bits total.

The SGTL5000 codec provides the serial bit clock, and the left-right clock needed to output audio. In order to combine and sequence the audio, we used an 8-state state machine, wherein at every positive edge of the left-right clock, we would increment the address for the next chunk of audio, and when the 14th bit of the address went from 1 to 0, it moved to the next state. In each state, we simply multiply the value of the pattern at that point (1 or 0) by the bits that represent the audio, and sum all together.

In order to determine whether or not to draw a pixel for a cell depending on a pattern, We take the DrawX and DrawY, and divide by 80 and 60, respectively, to give us a cell of 8 in the x axis and 8 in the y axis. We can know if that beat in the pattern is active by checking in the 32-bit array for $\text{pattern}[(8 * \text{CellY}) + (7 - \text{CellX})]$. If it is true, then we check within the sprite itself to see if that X and Y position has a foreground or background pixel by using $\text{DrawX} \% 80$ and $\text{DrawY} \% 60$.

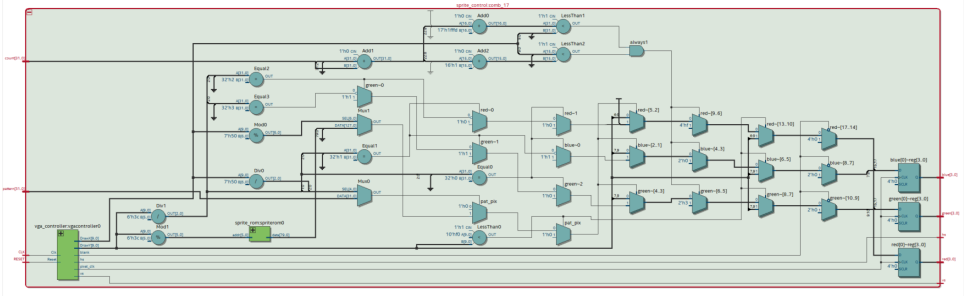
Top-Level Block Diagram



Module Descriptions

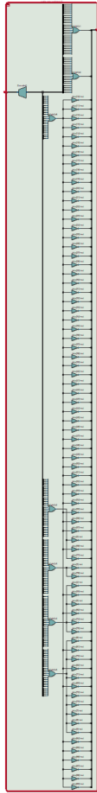
Module	audiorom
Inputs	logic [13:0] addr
Outputs	logic [31:0] data_Out
Description	Initialize memory from a text file that holds the audio for its respective sound
Purpose	When given a 14-bit address, packages the 8-bits at that address into 32 bits and outputs to the audio manager
RTL Diagram	

Module	i2s
Inputs	logic sclk, lrclk, logic [31:0] in
Outputs	i2s_din
Description	At every negative edge of the sclk outputs a single bit out At every edge of lrclk, load in a new set of 32 bits
Purpose	Receives 32 bits of data form the audio manager, and outputs serial data output to the SGTL5000 codec
RTL Diagram	<p>The RTL diagram, titled 'i2s:i2s0', illustrates the internal logic of the module. It features several key components and connections:</p> <ul style="list-style-type: none">Inputs: A 32-bit data input 'in[31:0]' and two clock signals, 'sclk' and 'lrclk'.Registers: Two 32-bit registers, 'l_reg[31:0]' and 'r_reg[31:0]', are used to store the input data. They are loaded on the rising edge of 'lrclk' (indicated by a triangle on the clock pin) with a constant '32'h0' as the next clock edge.Shift Registers: Two shift registers, 'l_out' and 'r_out', are used to output the data serially. They are clocked by 'sclk' and have an enable signal 'ENA' set to '1'h0'.Output: The serial outputs of the shift registers are combined using a 2-to-1 multiplexer to produce the final output 'i2s_din'.

Module	sprite_control
Inputs	logic [31:0] pattern int count logic CLK, RESET
Outputs	logic [3:0] red, green, blue, logic hs, vs
Description	Sets the colors of the pixels based on the current state of the pattern saved and the place audio is currently playing at
Purpose	When different patterns are set, the VGA controller changes the pattern shown on the screen to reflect that, as well as shows a yellow line along where the current audio is playing, to make it easier to see the current state of the beat at any moment
RTL Diagram	

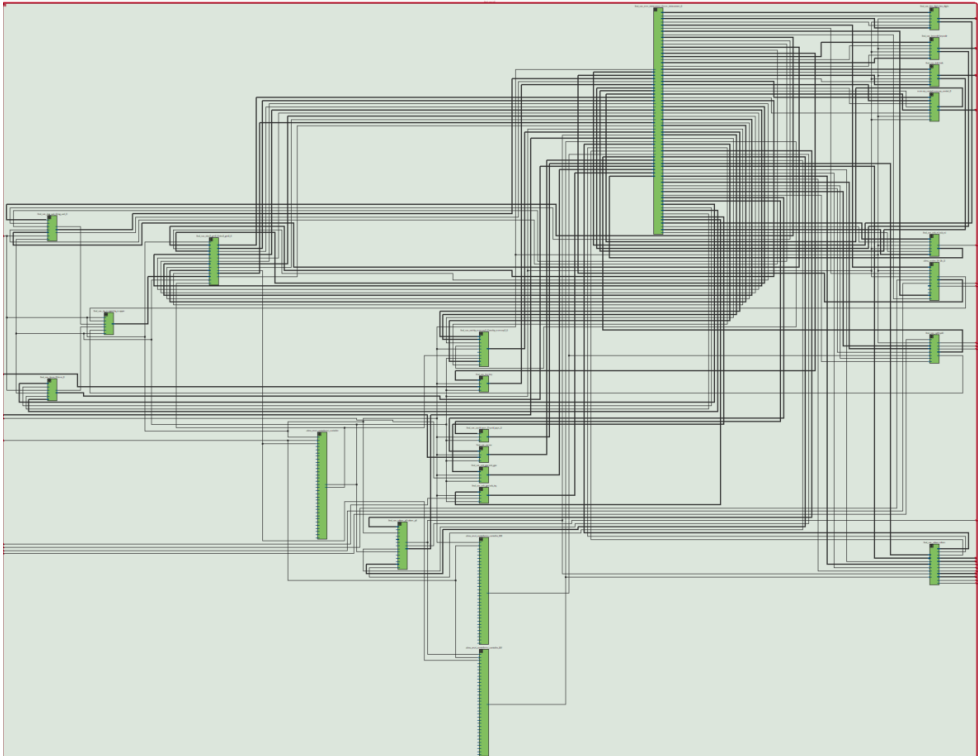
Module	vga_controller
Inputs	Clk, Reset
Outputs	logic hs, vs, pixel_clk, blank, sync [9:0] DrawX, DrawY
Description	Sets the clock for when a new pixel is drawn, and outputs the position of that pixel
Purpose	The purpose of this module is to help translate the logic behind the ball position to actual colors for each RGB value, by iterating through every every pixel between each Clock cycle
RTL Diagram	<p>The RTL diagram illustrates the internal logic of the vga_controller module. It features several registers: <code>hs-reg0</code> (10'h0), <code>vs-reg0</code> (1'h0), <code>hc[9:0]</code> (10'h0), and <code>vc[9:0]</code> (10'h0). The module includes two adders, <code>Add0</code> and <code>Add1</code>, which take 10-bit inputs and produce 10-bit outputs. There are also eight comparators: <code>LessThan0</code>, <code>LessThan1</code>, <code>LessThan2</code>, <code>LessThan3</code>, <code>Equal0</code>, <code>Equal1</code>, <code>Equal2</code>, and <code>Equal3</code>. The logic is controlled by <code>Clk</code> and <code>Reset</code> signals. The outputs are <code>hs</code> (1'h0), <code>vs</code> (1'h0), <code>pixel_clk</code> (10'h0), <code>blank</code> (1'h0), <code>sync</code> (1'h0), <code>DrawX[9:0]</code> (10'h0), and <code>DrawY[9:0]</code> (10'h0).</p>

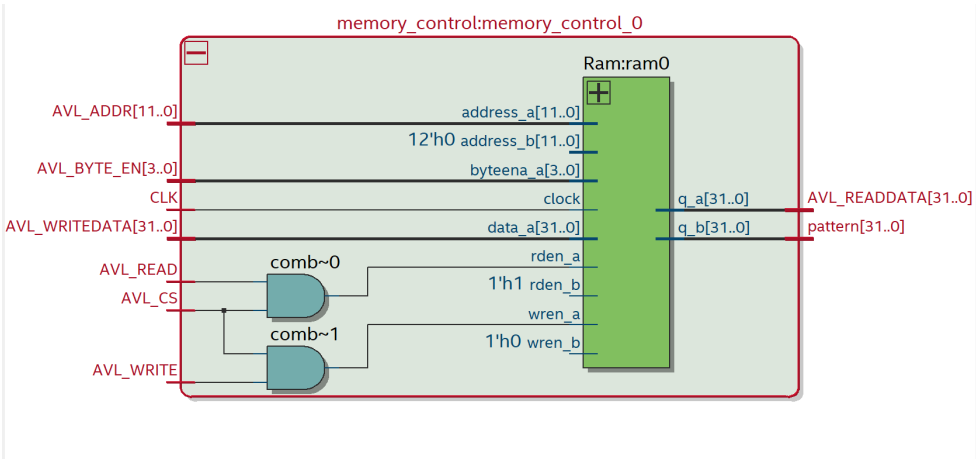
Module	audio_manager
Inputs	lrclk, [31:0] pattern
Outputs	[31:0] out int count
Description	Processes the current state of the pattern and the data from the individual audioroms to create the combined audio
Purpose	Outputs the 32 bits of audio to the i2s module which serially outputs it to be heard
RTL Diagram	

Module	sprite_rom
Inputs	[5:0] addr
Outputs	[79:0] data
Description	Stores data to create a single rounded rectangular cell
Purpose	Depending on the position of DrawX and DrawY, we use sprite_rom to know if the color of a character is the foreground or background
RTL Diagram	

Module	final_soc
Inputs	clk_clk, reset_reset_n, spi0_MISO, usb_gpx_export, usb_irq_export [1:0] key_external_connection_export
Outputs	[15:0] hex_digits_export [7:0] keycode_export [3:0] leds_export sdram_clk_clk, usb_rst_export [12:0] sdram_wire_addr [1:0] sdram_wire_ba sdram_wire_cas_n, sdram_wire_cke, sdram_wire_cs_n [15:0] sdram_wire_dq [1:0] sdram_wire_dqm sdram_wire_ras_n, sdram_wire_we_n spi0_MOSI, spi0_SCLK, spi0_SS_n [31:0] pattern
Description	Verilog module generated by the Platform Designer to connect various peripherals and components to the NIOS II
Purpose	In order to execute C code we need to connect the NIOS II CPU to the FPGA and USB and LEDs and internal memory, and this module does that.

RTL Diagram



Module	memory_control
Inputs	CLK, RESET AVL_READ, AVL_WRITE, AVL_CS [3:0] AVL_BYTE_EN [9:0] AVL_ADDR [31:0] AVL_WRITEDATA
Outputs	[31:0] AVL_READDATA [31:0] pattern
Description	Interfaces with the NIOS II to read stored data, and outputs the pattern stored in on-chip memory
Purpose	Interacts with the on-chip memory to store data onto on-chip memory, and returns that data to the rest of the soc
RTL Diagram	 <p>The RTL diagram illustrates the internal structure of the <code>memory_control</code> module. It features a block representing on-chip memory (<code>Ram:ram0</code>) with two data ports, <code>q_a[31..0]</code> and <code>q_b[31..0]</code>. The module's inputs are connected to internal signals as follows: <code>AVL_ADDR[11..0]</code> is connected to <code>address_a[11..0]</code>; <code>AVL_BYTE_EN[3..0]</code> is connected to <code>byteena_a[3..0]</code>; <code>CLK</code> is connected to the <code>clock</code> input of the memory block; <code>AVL_WRITEDATA[31..0]</code> is connected to <code>data_a[31..0]</code>. The control logic consists of two combinational blocks, <code>comb~0</code> and <code>comb~1</code>. <code>comb~0</code> takes <code>AVL_READ</code> and <code>AVL_CS</code> as inputs and produces <code>rden_a</code> (set to <code>1'h1</code>) and <code>wren_a</code>. <code>comb~1</code> takes <code>AVL_WRITE</code> and <code>AVL_CS</code> as inputs and produces <code>rden_b</code> (set to <code>1'h1</code>) and <code>wren_b</code> (set to <code>1'h0</code>). The memory block's <code>rden_a</code> and <code>wren_a</code> inputs are connected to the outputs of <code>comb~0</code>, while its <code>rden_b</code> and <code>wren_b</code> inputs are connected to the outputs of <code>comb~1</code>. The output of the memory's <code>q_a[31..0]</code> port is connected to <code>AVL_READDATA[31..0]</code>, and the output of the <code>q_b[31..0]</code> port is connected to <code>pattern[31..0]</code>.</p>

Platform Designer Modules

<input checked="" type="checkbox"/>	clk_0	Clock Source	clk	exported				
	clk_in	Clock Input	clk					
	clk_in_reset	Reset Input	reset	clk_0				
	clk	Clock Output	Double-click to Double-click to					
	clk_reset	Reset Output	Double-click to Double-click to					
<input checked="" type="checkbox"/>	nios2_gen2...	Nios II Processor						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	data_master	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	instruction_m...	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	irq	Interrupt Receiver	Double-click to Double-click to	[clk]				
	debug_reset_r...	Reset Output	Double-click to Double-click to	[clk]				
	debug_mem...	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	custom_instru...	Custom Instruction Master	Double-click to Double-click to	[clk]				
<input checked="" type="checkbox"/>	onchip_mem...	On-Chip Memory (RAM o...						
	clk1	Clock Input	Double-click to Double-click to	clk_0				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk1]				
	reset1	Reset Input	Double-click to Double-click to	[clk1]				
<input checked="" type="checkbox"/>	sw	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	sw					
<input checked="" type="checkbox"/>	hex_digits	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	hex_digits					
<input checked="" type="checkbox"/>	key	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	key_external_c...					
<input checked="" type="checkbox"/>	leds	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	leds					
<input checked="" type="checkbox"/>	sdram	SDRAM Controller Intel F...						
	clk	Clock Input	Double-click to Double-click to	sdram...				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	wire	Conduit	sdram_wire					
<input checked="" type="checkbox"/>	sdram_pll	ALTPLL Intel FPGA IP						
	indk_interface...	Clock Input	Double-click to Double-click to	clk_0				
	indk_interface...	Reset Input	Double-click to Double-click to	[indk_i...				
	pll_slave	Avalon Memory Mapped ...	Double-click to Double-click to	[indk_i...				
	c0	Clock Output	Double-click to Double-click to	sdram...				
	c1	Clock Output	sdram_clk					
<input checked="" type="checkbox"/>	sysid_qsqs_0	System ID Peripheral Inte...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	control_slave	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
<input checked="" type="checkbox"/>	jtag_uart_0	JTAG UART Intel FPGA IP						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	avalon_jtag_sl...	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	irq	Interrupt Sender	Double-click to Double-click to	[clk]				
<input checked="" type="checkbox"/>	usb_irq	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	usb_irq					
<input checked="" type="checkbox"/>	usb_gpx	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	usb_gpx					
<input checked="" type="checkbox"/>	usb_rst	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	usb_rst					
<input checked="" type="checkbox"/>	timer_0	Interval Timer Intel FPGA...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	irq	Interrupt Sender	Double-click to Double-click to	[clk]				
<input checked="" type="checkbox"/>	spi0	SPI (3 Wire Serial) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	spi_control_port	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	irq	Interrupt Sender	Double-click to Double-click to	[clk]				
	external	Conduit	spi0					
<input checked="" type="checkbox"/>	keycode	PIO (Parallel I/O) Intel F...						
	clk	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clk]				
	s1	Avalon Memory Mapped ...	Double-click to Double-click to	[clk]				
	external_conn...	Conduit	keycode					
<input checked="" type="checkbox"/>	memory_cont...	memory_control						
	avl_mm_slave	Avalon Memory Mapped ...	Double-click to Double-click to	[clock]				
	clock	Clock Input	Double-click to Double-click to	clk_0				
	reset	Reset Input	Double-click to Double-click to	[clock]				
	pattern	Conduit	pattern					
<input checked="" type="checkbox"/>	i2c_0	Avalon I2C (Master) Intel...						
	clock	Clock Input	Double-click to Double-click to	clk_0				
	reset_sink	Reset Input	Double-click to Double-click to	[clock]				
	interrupt_sender	Interrupt Sender	Double-click to Double-click to	[clock]				
	csr	Avalon Memory Mapped ...	Double-click to Double-click to	[clock]				
	i2c_serial	Conduit	i2c					

Component	memory_control
Input	CLK, RESET, AVL_READ, AVL_WRITE, AVL_CS, AVL_BYTE_EN, AVL_ADDR, AVL_WRITEDATA, AVL_READDATA,
Conduit	[31:0] pattern
Description	Interfaces with the NIOS II to read stored data, and outputs the current pattern

Component	nios2_gen2_0
Input	clk, reset, irq, debug_mem_slave
Output	data_master, instruction_master, debug_reset_request
Conduit	clk, reset
Description	The NIOS-II Processor, using I/O to allow the processor to interface, interact and execute the instructions

Component	onchip_memory2_0
Input	clk1, s1, reset1
Description	The memory module where we allocate on-chip memory

Component	SDRAM
Input	CLK, reset, s1
Conduit	sdram_wire
Description	Interfaces with the LEDs and switches

Component	sdram_pll
Input	inclk_interface, inclk_interface_reset, pll_slave
Output	c0
Conduit	sdram_clk
Description	Generates the clock that connects to the SDRAM, and allows for the required delay for read/write

Component	sysid_qsys_0
Input	clk, reset, control_slave
Description	System ID checker to prevent incompatibility between hardware and software

Component	jtag_uart_0
Input	clk, reset, avalon_jtag_slave
Output	irq
Description	Helps communicate with VGA monitor

Component	usb_irq
Input	clk, reset, s1
Conduit	usb_irq
Description	Used to communicate with keyboard and VGA

Component	usb_gpx
Input	clk, reset, s1
Conduit	usb_gpx
Description	Used to communicate with keyboard and VGA

Component	usb_rst
Input	clk, reset, s1
Conduit	usb_rst
Description	Used to communicate with keyboard and VGA

Component	timer_0
Input	clk, reset, s1
Output	irq
Description	Acts as timer for MAX

Component	spi0
Input	clk, reset, spi_control_port
Output	irq
Conduit	spi0
Description	3-wire serial comm to connect the USB data to the NIOS II

Component	keycode
Input	clk, reset, s1
Conduit	keycode
Description	PIO module for the FPGA switches to connect the input of the switches to the NIOS II

Component	hex_digits
Input	clk, reset, s1
Conduit	hex_digits
Description	PIO module for the FPGA Hex Displays to connect the displays to the NIOS II

Component	leds
Input	clk, reset, s1
Conduit	leds
Description	PIO module for the FPGA LEDs to connect the LEDs to the NIOS II

Component	key
Input	clk, reset, s1
Conduit	key_external_connection
Description	PIO module to connect the NIOS II to the input value of Key[0] and Key[1] at every rising edge

Component	i2c_0
Input	clk, reset, csr
Conduit	i2c
Description	PIO module to connect the NIOS II to the input value of Key[0] and Key[1] at every rising edge

Performance

LUT	21548
DSP	N/A
Memory (BRAM)	221184
Flip-Flop	2938
Frequency	115.31 MHz
Static Power	96.18 mW
Dynamic Power	0.58 mW
Total Power	105.87 mW

Conclusion

1. Discuss the functionality of your design. If parts of your design did not work, discuss what could be done to fix it.
 - a. Near everything on the design functioned as it should. We did not implement MicroSD card memory, and it would be interesting to see if that made compile times noticeably faster, and allowed for even further selection of sounds.
Moreover, while 8-bit quality wav files sufficed, it would be interesting to try with 16-bit wav files, to improve the fidelity of the audio
Currently, we were limited to only 8 beats of sounds, and 4 sounds to use, but with a more developed UI, that could be changed to give even more variety to the user.
2. Was there anything ambiguous, incorrect, or unnecessarily difficult in the given materials that can be improved for next semester? You can also specify what we did right, so it does not get changed.
 - a. I think that more written documentation on using i2s would be of great help to future students.