CSEE4840 Embedded Systems
Video Game: Monster Casino

By Zongwei Zhen(zz3083), Chenzhi Lu(cl4407), Shaokun Feng(sf3209)
Professor: Stephen A. Edwards

Outline

1. Introduction
2. Sysytem Design
3. Hardware Design
4. Software Design
5. Improvements

Final Project

Part 1 – Introduction

Part 1 – Introduction

In this project, our group designed a single-player and online dual-mode battle game based on SoC (System on Chip) called Monster Casino. This game combines elements of casino slot machines with monster capture and monster battle. The objective is to obtain monsters from the slot machine, nurture them, join the battle with wild monster, and finally use the monster to beat boss monster or play against other player via online and win the game.

The FPGA is responsible for reading images, driving the screen, playing background music. The software communicates with the FPGA through Avalon bus to complete the design of the game logic, send user controlling message and internet transmitting.

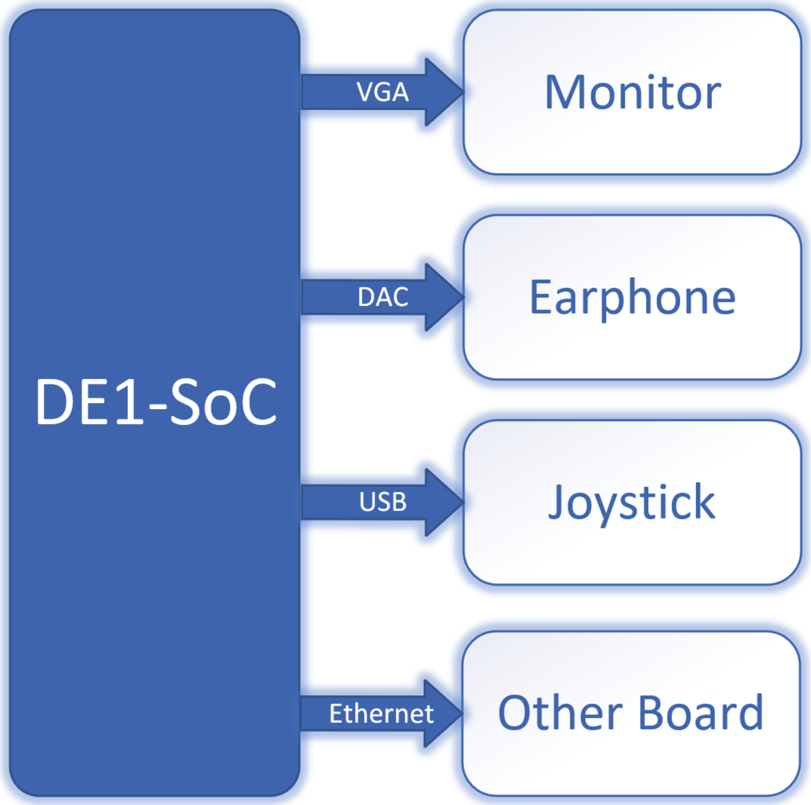
Final Project

Part 2 – System Design

Part 2 – System Design

Peripherals Design Overview

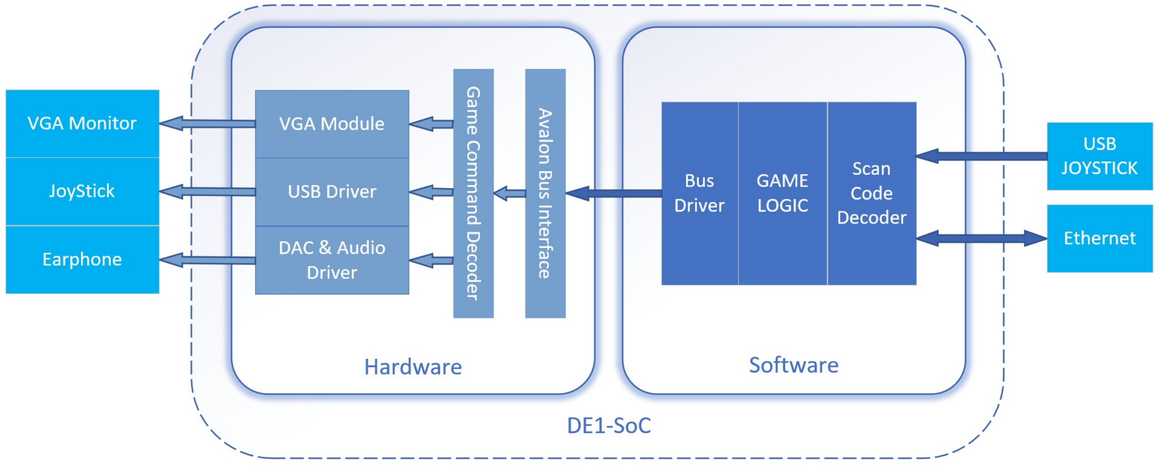
As the main board of DE1-SoC, the peripherals connect with screen monitor, earphone, joystick and other boards.



Part 2 – System

System Design Overview

The design involves a USB joystick that sends scan codes to a System-on-Chip (SoC), which decodes these into game commands. These commands drive game logic, which controls image and audio outputs. Visuals are displayed via VGA after pixel data is fetched from ROM. Audio tones, managed in Verilog, are generated through frequency division, played via a DAC, and output through headphones using an onboard amplifier. Additionally, game data is transmitted over Ethernet for online gameplay.



Final Project

Part 3 – Hardware Design

Part 3 - Hardare Design

General Bus

In the design, we used 8*32 bits different registers.

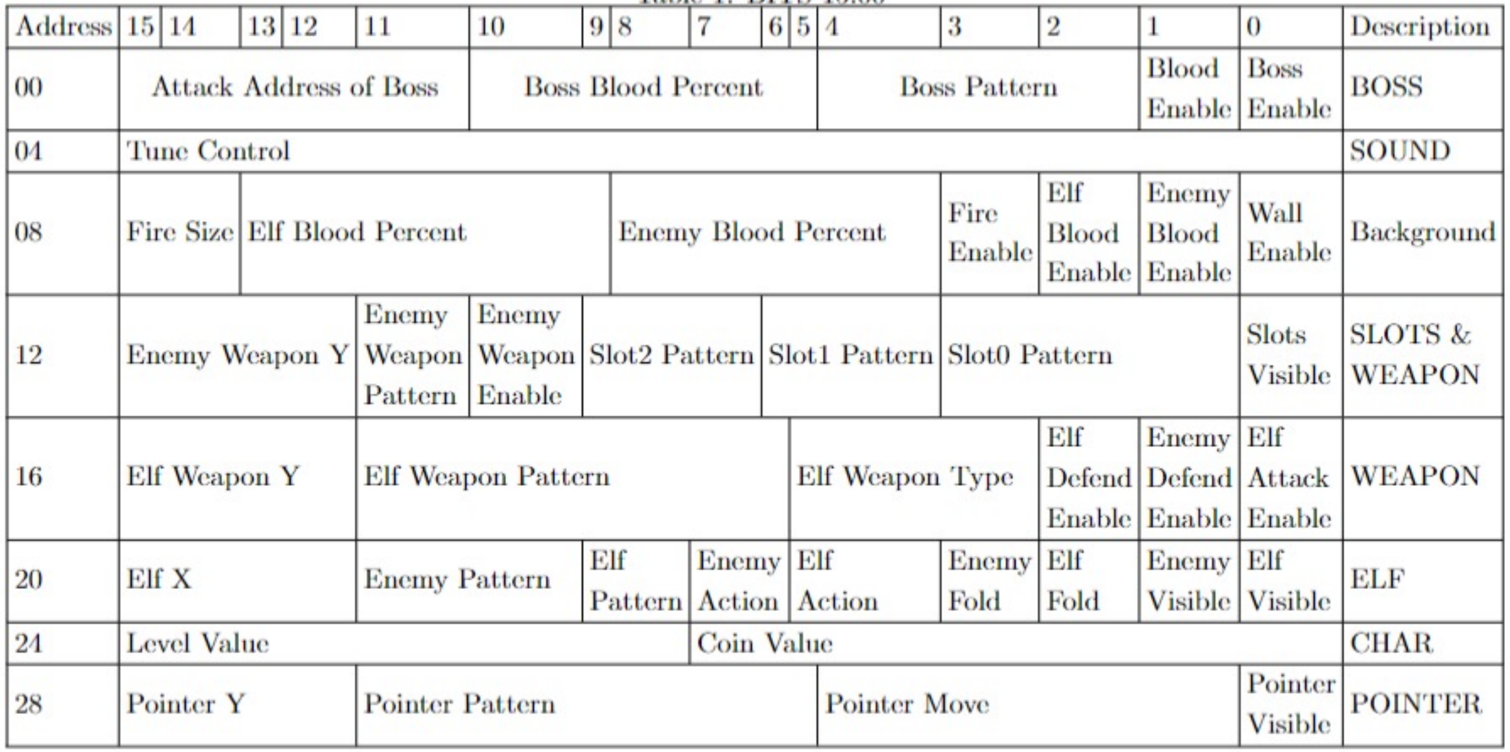


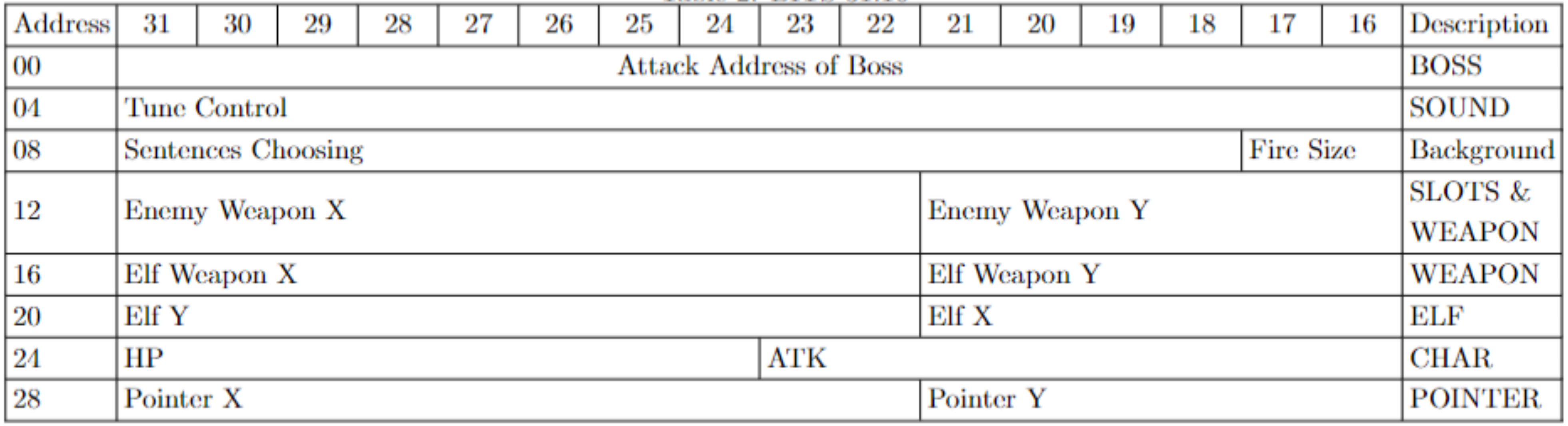
Table 1: BITS 15:00

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Description
00	Attack Address of Boss					Boss Blood Percent						Boss Pattern			Blood Enable	Boss Enable	BOSS
04	Tune Control																SOUND
08	Fire Size		Elf Blood Percent					Enemy Blood Percent					Fire Enable	Elf Blood Enable	Enemy Blood Enable	Wall Enable	Background
12	Enemy Weapon Y				Enemy Weapon Pattern	Enemy Weapon Enable	Slot2 Pattern			Slot1 Pattern			Slot0 Pattern			Slots Visible	SLOTS & WEAPON
16	Elf Weapon Y				Elf Weapon Pattern						Elf Weapon Type			Elf Defend Enable	Enemy Defend Enable	Elf Attack Enable	WEAPON
20	Elf X				Enemy Pattern		Elf Pattern		Enemy Action		Elf Action		Enemy Fold	Elf Fold	Enemy Visible	Elf Visible	ELF
24	Level Value								Coin Value								CHAR
28	Pointer Y				Pointer Pattern							Pointer Move				Pointer Visible	POINTER

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Part 3 - Hardare Design

Table 2: BITS 31:16



Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	Description
00	Attack Address of Boss																BOSS
04	Tune Control																SOUND
08	Sentences Choosing														Fire Size		Background
12	Enemy Weapon X										Enemy Weapon Y						SLOTS & WEAPON
16	Elf Weapon X										Elf Weapon Y						WEAPON
20	Elf Y										Elf X						ELF
24	HP								ATK								CHAR
28	Pointer X										Pointer Y						POINTER

Part 3 - Hardare Design

Graphics

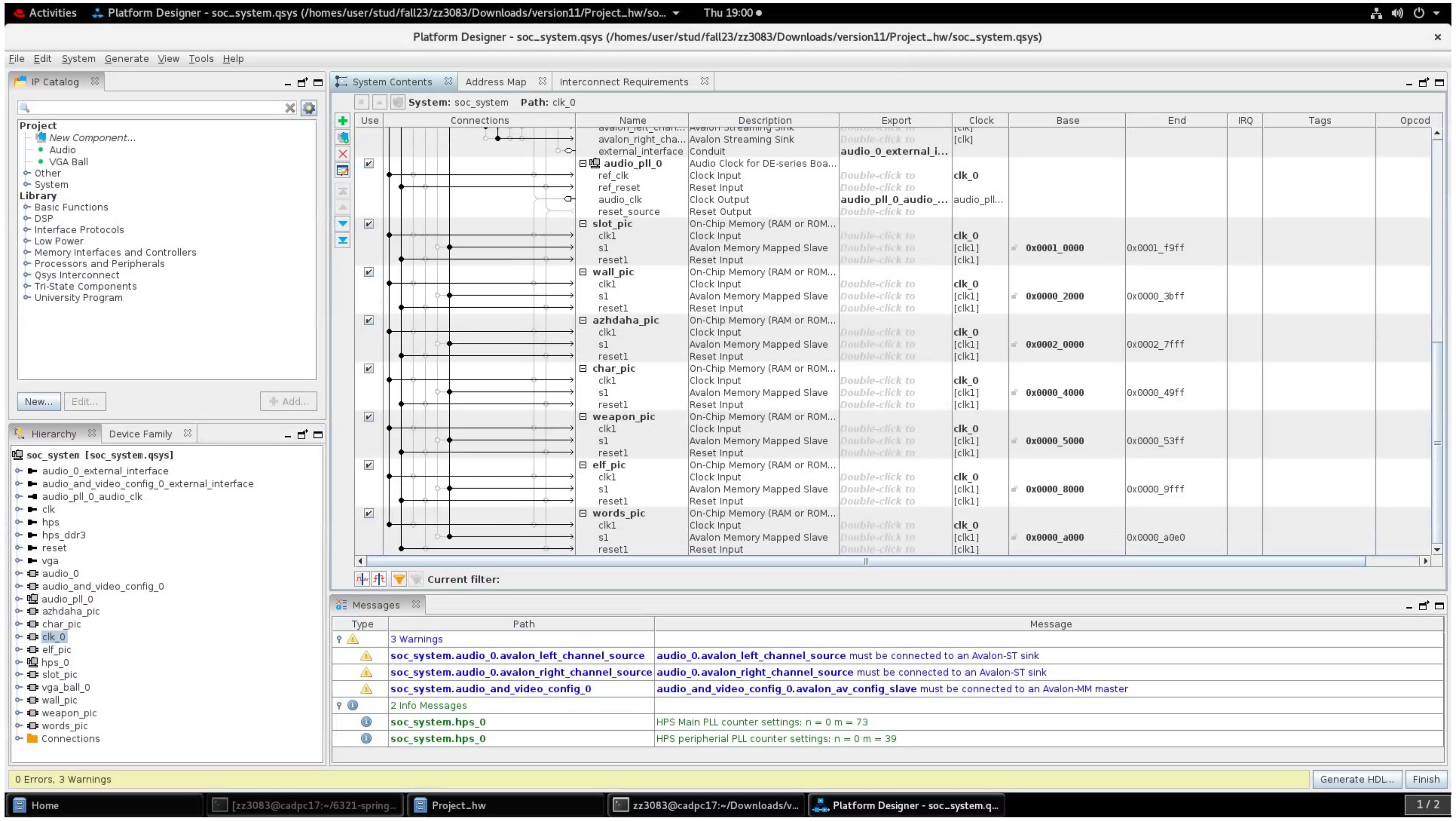
In our design, graphics are implemented through two primary methods:

- ROM Storage for Basic Graphics: Key figures like elves, bosses, and backgrounds are stored in ROM. Due to ROM's size limitations, RGB888 format (3 bytes per pixel) is not used. Instead, images are downsampled using Matlab, reducing color depth to one byte per pixel, which references a color table. Matlab is also used to resize images and convert them to .mif format, suitable for storage and retrieval via an Intel IP block.

- Hard-coded Graphics in SystemVerilog: Certain elements such as elf and enemy weapons, and protective circles are hardcoded directly in SystemVerilog, bypassing the need for ROM storage.

Part 3 - Hardare Design

Picture ROM Design Overview


Platform Designer - soc_system.qsys (/homes/user/stud/fall23/zz3083/Downloads/version11/Project_hw/soc_system.qsys)

System: soc_system   Path: clk_0

Use	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcod
	avalon_right_cha...	Avalon Streaming Sink	Double-click to	[clk]					
	external_interface	Conduit	audio_0_external_i...						
✓	audio_pll_0	Audio Clock for DE-series Boa...							
	ref_clk	Clock Input	Double-click to	clk_0					
	ref_reset	Reset Input	Double-click to						
	audio_clk	Clock Output	audio_pll_0_audio_...	audio_pll...					
	reset_source	Reset Output	Double-click to						
✓	slot_pic	On-Chip Memory (RAM or ROM...							
	clk1	Clock Input	Double-click to	clk_0					
	s1	Avalon Memory Mapped Slave	Double-click to	[clk1]	0x0001_0000	0x0001_f9ff			
	reset1	Reset Input	Double-click to	[clk1]					
✓	wall_pic	On-Chip Memory (RAM or ROM...							
	clk1	Clock Input	Double-click to	clk_0					
	s1	Avalon Memory Mapped Slave	Double-click to	[clk1]	0x0000_2000	0x0000_3bff			
	reset1	Reset Input	Double-click to	[clk1]					
✓	azhdaha_pic	On-Chip Memory (RAM or ROM...							
	clk1	Clock Input	Double-click to	clk_0					
	s1	Avalon Memory Mapped Slave	Double-click to	[clk1]	0x0002_0000	0x0002_7fff			
	reset1	Reset Input	Double-click to	[clk1]					
✓	char_pic	On-Chip Memory (RAM or ROM...							
	clk1	Clock Input	Double-click to	clk_0					
	s1	Avalon Memory Mapped Slave	Double-click to	[clk1]	0x0000_4000	0x0000_49ff			
	reset1	Reset Input	Double-click to	[clk1]					
✓	weapon_pic	On-Chip Memory (RAM or ROM...							
	clk1	Clock Input	Double-click to	clk_0					
	s1	Avalon Memory Mapped Slave	Double-click to	[clk1]	0x0000_5000	0x0000_53ff			
	reset1	Reset Input	Double-click to	[clk1]					
✓	elf_pic	On-Chip Memory (RAM or ROM...							
	clk1	Clock Input	Double-click to	clk_0					
	s1	Avalon Memory Mapped Slave	Double-click to	[clk1]	0x0000_8000	0x0000_9fff			
	reset1	Reset Input	Double-click to	[clk1]					
✓	words_pic	On-Chip Memory (RAM or ROM...							
	clk1	Clock Input	Double-click to	clk_0					
	s1	Avalon Memory Mapped Slave	Double-click to	[clk1]	0x0000_a000	0x0000_a0e0			
	reset1	Reset Input	Double-click to	[clk1]					

Hierarchy: soc_system [soc_system.qsys]
- audio_0_external_interface
- audio_and_video_config_0_external_interface
- audio_pll_0_audio_clk
- clk
- hps
- hps_ddr3
- reset
- vga
- audio_0
- audio_and_video_config_0
- audio_pll_0
- azhdaha_pic
- char_pic
- clk_0
- elf_pic
- hps_0
- slot_pic
- vga_ball_0
- wall_pic
- weapon_pic
- words_pic
- Connections

Messages

Type	Path	Message
	3 Warnings	
⚠	soc_system.audio_0.avalon_left_channel_source	audio_0.avalon_left_channel_source must be connected to an Avalon-ST sink
⚠	soc_system.audio_0.avalon_right_channel_source	audio_0.avalon_right_channel_source must be connected to an Avalon-ST sink
⚠	soc_system.audio_and_video_config_0	audio_and_video_config_0.avalon_av_config_slave must be connected to an Avalon-MM master
	2 Info Messages	
ⓘ	soc_system.hps_0	HPS Main PLL counter settings: n = 0 m = 73
ⓘ	soc_system.hps_0	HPS peripheral PLL counter settings: n = 0 m = 39

0 Errors, 3 Warnings

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Part 3 - Hardare Design

Audio

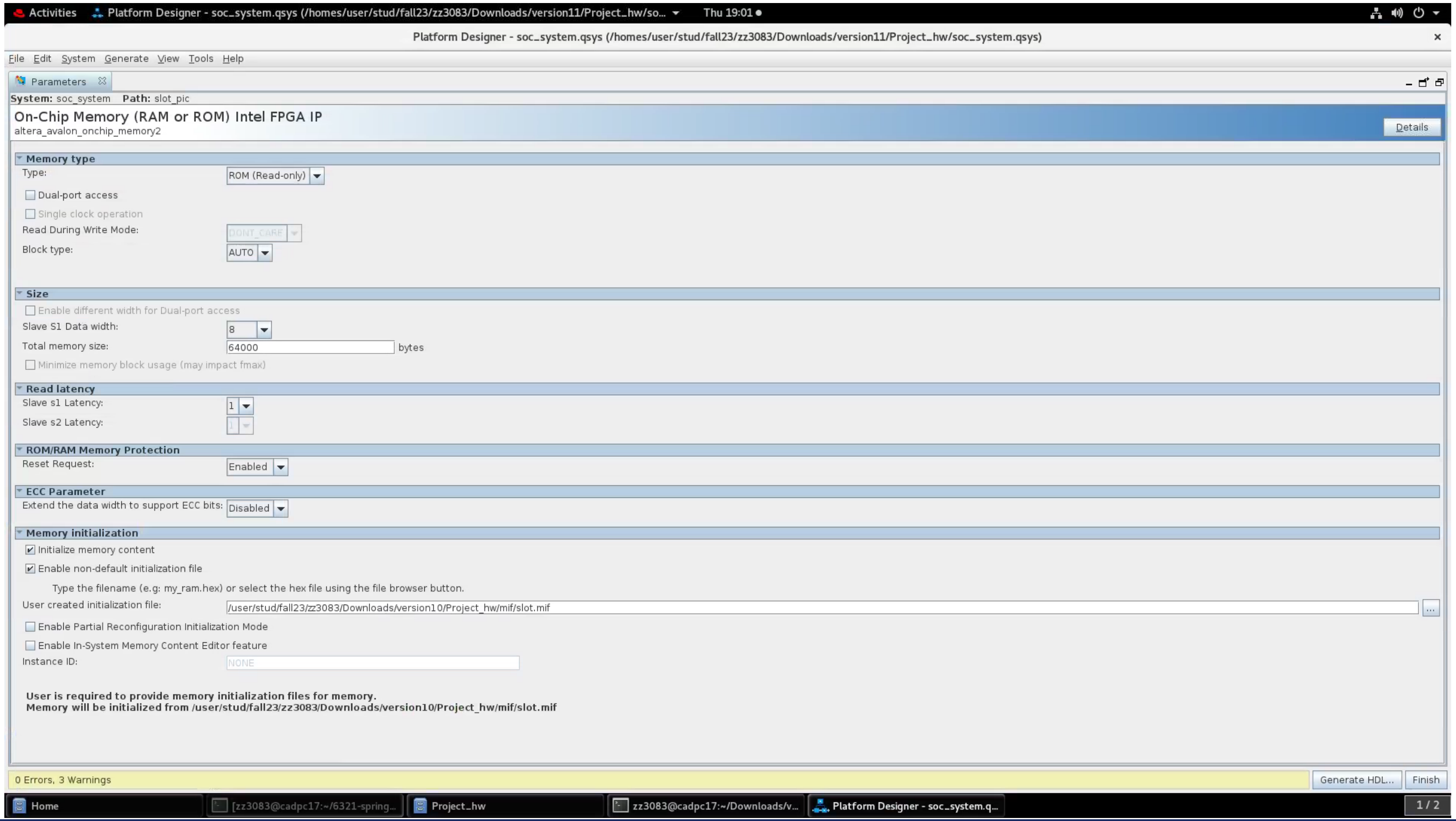
We used MIDI files from Pokémon-related websites, converting these into 16-bit audio waves using MATLAB, and stored them as MIF files, limited to 8192 bytes due to storage constraints.

For continuous background music, we generated tones using Verilog, controlled by signals from the Avalon bus. Audio output was managed using the WM8731 audio module and IP blocks, with communication facilitated by a FIFO system between the audio module and the VGA-ball.

The system operates at an 8kHz sampling rate with a 16-bit width, regulated by a frequency-reducing counter. Background music is input into the system via the Avalon bus to a specific register, allowing for software control of the melody.

Part 3 - Hardare Design

Music ROM IP Block


Activities Platform Designer - soc_system.qsys (/homes/user/stud/fall23/zz3083/Downloads/version11/Project_hw/so... Thu 19:01

Platform Designer - soc_system.qsys (/homes/user/stud/fall23/zz3083/Downloads/version11/Project_hw/soc_system.qsys)

File Edit System Generate View Tools Help

Parameters

System: soc_system **Path:** slot_pic

On-Chip Memory (RAM or ROM) Intel FPGA IP
altera_avalon_onchip_memory2

Memory type

Type: ROM (Read-only)
Dual-port access
Single clock operation
Read During Write Mode: DONT_CARE
Block type: AUTO

Size

Enable different width for Dual-port access
Slave S1 Data width: 8
Total memory size: 64000 bytes
Minimize memory block usage (may impact fmax)

Read latency

Slave s1 Latency: 1
Slave s2 Latency: 1

ROM/RAM Memory Protection

Reset Request: Enabled

ECC Parameter

Extend the data width to support ECC bits: Disabled

Memory initialization

☑ Initialize memory content
☑ Enable non-default initialization file

Type the filename (e.g: my_ram.hex) or select the hex file using the file browser button.

User created initialization file: /user/stud/fall23/zz3083/Downloads/version10/Project_hw/mif/slot.mif

Enable Partial Reconfiguration Initialization Mode
Enable In-System Memory Content Editor feature
Instance ID: NONE

User is required to provide memory initialization files for memory.
Memory will be initialized from /user/stud/fall23/zz3083/Downloads/version10/Project_hw/mif/slot.mif

0 Errors, 3 Warnings

Details
Generate HDL...
Finish
Home
[zz3083@cadpc17:~/6321-spring...
Project_hw
zz3083@cadpc17:~/Downloads/v...
Platform Designer - soc_system.q...
1 / 2

Part 3 - Hardare Design

Rom picutre

In the slot machine, our 5 images together occupy 60000 bytes of ROM. The boss's 4 action images occupy 49152 bytes of ROM. The elf's two images occupy 8192 bytes of ROM. The background images, including two states of wall and floor images, mirrored two flame images, and one reward symbol, occupy 7168 bytes of ROM. Finally, characters occupy 265 bytes, and numbers occupy 320 bytes of ROM. The overall occupancy is less than the ROM limit on the board.

Table 3: ROM Picture

NAME	Graphic	Size(bits)	Color Bits	#Required	Total Size(bytes)
Background		32 x 32	8	7	7168
Slot		100 x 128	8	5	60,000
Boss		96 x 128	8	4	49,152
Elf/Enemy		64 x 64	8	2	8,192
Font	PRESS	5 x 8	1	53	265
Numbers	003	16 x 16	1	10	320

Final Project

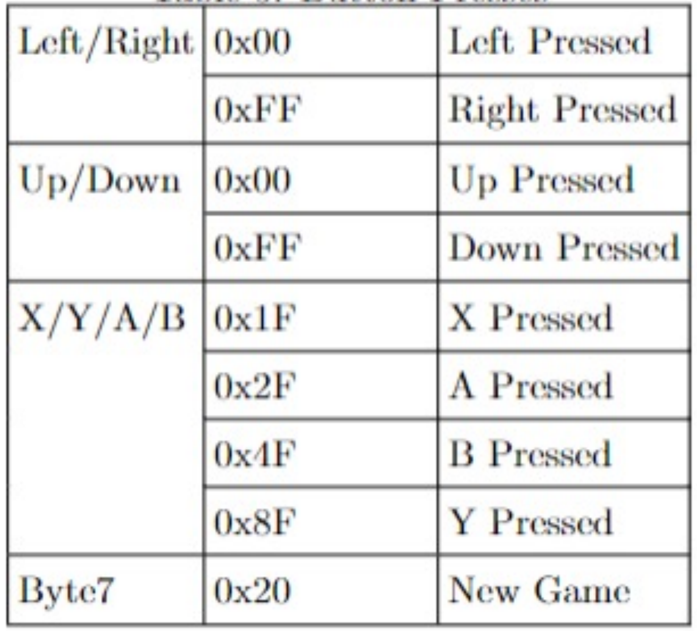
Part 4 – Sofware Design

Part 4 - Software Design

User Input

Our project involves connecting controllers to the software program via USB input. We use the Linux libusb-1.0 C library to read USB scan codes, and the interface is initialized using the keyboard function. Players can control the pointer and elf movement using the arrow keys, stop the slot machine and defend during battle with the B key, and attack and capture elves during battle with the A key. The slot is reset using the Y key, and the menu is navigated with the X key.

Table 5: Button Presses



Left/Right	0x00	Left Pressed
	0xFF	Right Pressed
Up/Down	0x00	Up Pressed
	0xFF	Down Pressed
X/Y/A/B	0x1F	X Pressed
	0x2F	A Pressed
	0x4F	B Pressed
	0x8F	Y Pressed
Byte7	0x20	New Game

Part 4 - Software Design

Online Battle

The real-time online battle system implemented in this project allows players to engage in dynamic combat scenarios over a network, offering a seamless multiplayer experience through TCP communication.

Communication Protocol:

•Initial Setup:

- Host: Listens on a predefined port for TCP connections.
- Client: Connects and sends initial settings via a handshake message.
- Acknowledgment: Host confirms the connection, transitioning both to battle mode.

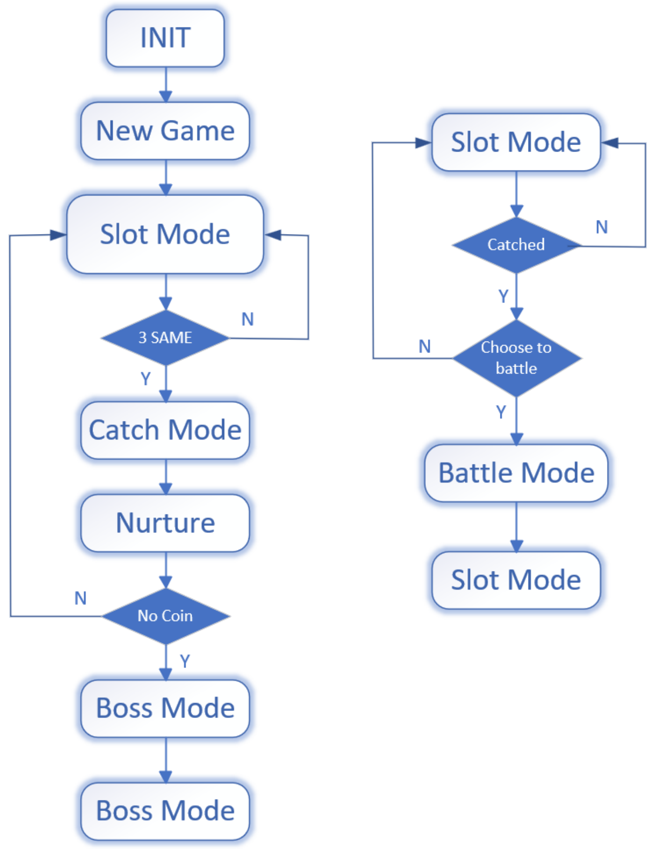
•During Battle:

- Host to Client: Sends continuous updates on game state and player health.
- Client to Host: Sends current player actions for synchronization.

Part 4 - Software Design

Game Logic

After the game starts, enter slot machine mode. Then, enter the Catch mode to capture the elf by judging whether three identical slots appear. After capturing, return to slot mode. Once the number of allowed coin flips is reduced to 0, boss mode is entered. After successfully capturing the elf, players can choose to enter battle mode from the menu. Players can choose to fight wild monsters, fight bosses, or fight other players online.



Part 4 - Software Design

Game Logic

Now let's welcome Shaokun Feng to give a live demonstration about detailed game logic.

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Final Project

Part 5 – Improvements

Part 5 - Improvements

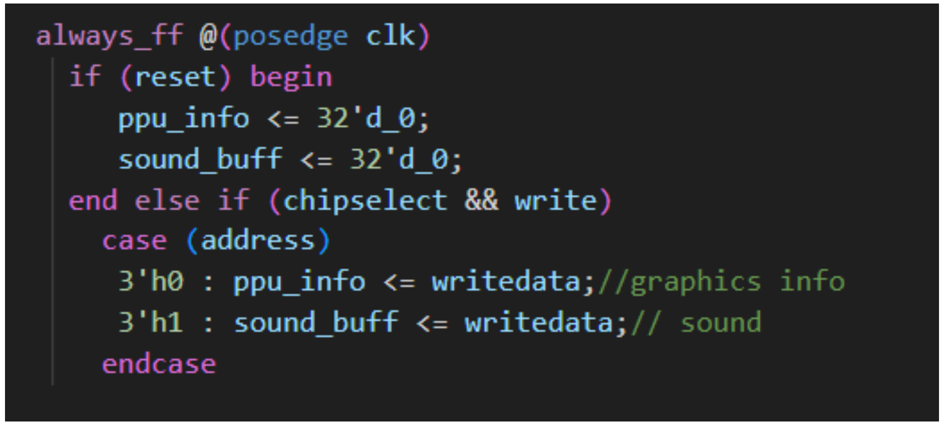
Compared with the last time we met with the professor, our project has improved significantly.

• We reconstructed the underlying display logic
• We greatly enriched the game logic
• We added an online battle mode so that the game can support multiplayers, increasing the fun and interactivity of the game

Part 5 - Improvements

Reconstruction the underlying display logic

In the past we only used two register addresses, one for image and one for sound. There are complex classification discussions on the display of images (just like different types of MIPS instructions), which eventually leads to timing problems in the entire SOC system. The images on the screen will flicker and be unclear. Now after our reconstruction, we use ROM to store images and use 8 32-bit registers to interact with the bus. Now our images are clear and stable, which greatly improves the user experience.



```
always_ff @(posedge clk)
  if (reset) begin
     ppu_info <= 32'd_0;
     sound_buff <= 32'd_0;
  end else if (chipselect && write)
    case (address)
     3'h0 : ppu_info <= writedata;//graphics info
     3'h1 : sound_buff <= writedata;// sound
    endcase
```

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Contribution

In this project, every member of the team played an integral role. Zongwei is responsible for the design of the entire game hardware, including images and music, as well as the framework design of the software part, and the writing of the report; Chenzhi is responsible for the partially design of the hardware music, the game logic of the game software, the presentation, the finalization of the paper, and the final debugging process of the game; Shaokun was responsible for the game design of the online battle, the game software design, the writing of the paper, and the final debugging of the game.

Contribution

Among us, we would like to sincerely thank Zongwei for his efforts. After the first meeting with the professor, he made a decisive decision to reconstruct our underlying hardware display logic code. As we all know, hardware reconstruction is very time-consuming. Zongwei spent three nights in the laboratory and finally completed the code reconstruction. His spirit inspired the rest of us and allowed us to complete the task within the limited time. Due to campus restrictions, the final days of the project were extremely difficult, but every member of the team never gave up and helped each other, and finally achieved today's results. This is an unforgettable time.

Acknowledgments

Finally, we would like to express our gratitude to the professor Edward and all teaching assistants of CSEE 4840 for their valuable opinions and selfless help. We will use the knowledge we have learned to make further progress in the future!

THANK YOU