Rhythm Master

Group Members: Yangyang Zhang (yz4843), Junfeng Zou (jz3850), Hongrui Huang (hh3084)
Target Platform: DE1-SoC FPGA, VGA Display, USB Keyboard, Speaker

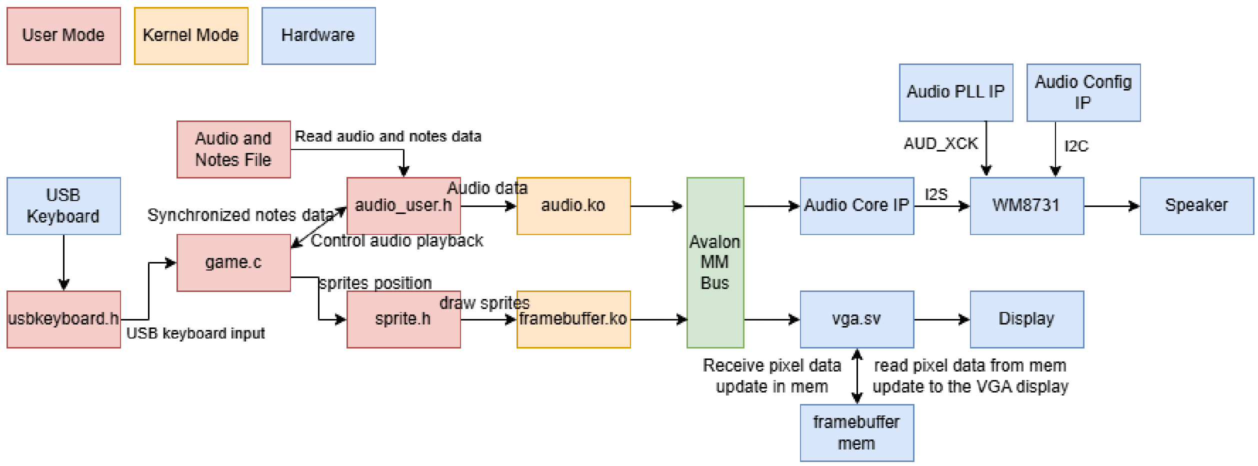
1. Introduction

This document outlines the design of Rhythm Master, a rhythm-based music game where players synchronize key presses with falling notes. The system combines hardware and software components to achieve real-time gameplay, audio synchronization, and user interaction. Key design decisions include:

- **Audio (FPGA):** Implemented using three IP cores — Audio PLL, Audio Config, and Audio Core — which handle sample timing, codec initialization, and audio output via the WM8731 codec through I²S.
- **Display (FPGA + HPS):** VGA output is handled by a custom Verilog module vga.sv, which reads pixel data from a dedicated framebuffer stored in FPGA on-chip memory. The HPS renders graphics (sprites, UI, notes) into a next_frame buffer in HPS. Once rendering is complete, a kernel module (framebuffer.ko) writes next_frame pixel-by-pixel to vga.sv which saves the data into the framebuffer memory. The FPGA continuously reads this memory to drive VGA output.
- **Game Logic & USB Keyboard Input (HPS):** All gameplay logic — including note movement, hit detection, scoring, and menu flow — runs entirely on the HPS side. USB keyboard inputs are handled in software for controlling gameplay.
- **Communication:** The HPS streams 16-bit PCM audio data to the FPGA via the Avalon-MM interface. VGA frame data is written into shared SDRAM, from which the FPGA retrieves it using DMA for real-time video output.

This architecture emphasizes low-latency audio playback and real-time visual synchronization for seamless rhythm gameplay.

2. System Block Diagram



The architecture is split into functional components as follows:

Hardware (FPGA) Components

1. Audio Pipeline:
    - PLL IP: Generates required clock signals.
    - Audio Config IP: Configures WM8731 codec.
    - Audio Core IP: Streams audio data to the codec using I²S protocol.
2. VGA Display Pipeline (vga.sv):
    - Receives pixel data via MMIO from HPS
    - Stores pixel values in a framebuffer located in FPGA on-chip memory
    - Continuously scans framebuffer and generates VGA signals for monitor output

Software (HPS) Components

1. Sprite Renderer: Renders all graphical elements (lanes, notes, UI) into a next_frame buffer in HPS memory.
2. Frame Sender: After rendering, the content of next_frame is copied pixel-by-pixel to a memory-mapped region exposed by the kernel module framebuffer.ko via MMIO.
3. Game Logic: Manages note generation, timing synchronization with audio, grading, and level control.
4. Input Handler: Handles USB keyboard input and maps them to in-game controls.
5. Audio Data Streamer: Transfers audio sample buffers to the FPGA through Avalon-MM interface.

Communication Protocols

1. Audio Streaming: 16-bit PCM samples are pushed from the HPS to the FPGA via Avalon-MM writes to the Audio Core IP's memory-mapped FIFO interface. The software alternates between left and right channel FIFO writes.

2. Video Streaming: The HPS renders each frame into a local next_frame buffer. After rendering, a kernel module (framebuffer.ko) writes the pixel data sequentially to a memory-mapped MMIO region exposed by the custom VGA module (vga.sv). The FPGA stores incoming pixels into its internal framebuffer (on-chip memory), which is continuously scanned and sent to VGA output.
3. Game Events/Input: Entirely managed on HPS; no need for hardware interrupts or polling from the FPGA side.

3. Algorithms

Audio Playback

- FPGA Hardware Pipeline: Audio playback is handled entirely in hardware using IP cores. The FPGA receives PCM samples via Avalon-MM and streams them via I²S to the codec.
- Output Device: The WM8731 codec converts the digital audio stream to analog and sends it to the audio output port.
- Latency Control: Since audio FIFO depth is fixed (2 × 128 × 32-bit), double buffering and real-time streaming on the software side ensure uninterrupted audio playback.

Video Rendering

- Frame Rendering: The HPS renders each frame (sprites, notes, UI) into a local buffer next_frame.
- Frame Transfer: After rendering, next_frame is written pixel-by-pixel to the FPGA via MMIO, using the kernel module framebuffer.ko.
- Framebuffer Storage: The FPGA-side module vga.sv stores the received pixels into a dedicated on-chip framebuffer.
- VGA Output: vga.sv continuously reads the framebuffer and outputs VGA signals at 640×480 resolution, 60 Hz refresh rate.

Game Logic

1. Note Rendering:
    - Pre-processed beat arrays are stored and used to determine when and where notes appear on screen.
    - Notes move vertically with time, synchronized to the audio playback timeline.
2. Hit Detection:
    - There is a determination area at the bottom of the screen. For each note, take the hit detection logic according to its position when its corresponding key is pressed:
    - Classify accuracy:
        - Perfect: Key is pressed when most part of the note is in the determination region
        - Good: Key is pressed when part of the note is in the determination region
        - Miss: Key is not pressed or pressed when the note is not in the determination region at all
3. Scoring:
    - Maintain a counter incremented for combo hits; reset on miss.
    - Final score = Total Hit Count (e.g. Perfect with 5 points; Good with 3 points) + Combo Bonus
    - Level Classification: When a whole song is completed, give a level classification (e.g. S, A, B, C) according to the final score.

4. Resource Budgets

Memory Allocation

Component	Size (Bits)	Quantity	Total (Bits)
Audio FIFO - Left	128 × 32 = 4,096	1	4,096
Audio FIFO - Right	128 × 32 = 4,096	1	4,096
VGA Frame Buffer Memory	640 × 480 × 8 = 2,457,600	1	2,457,600

Logic Elements

Component	Size (Bits)
Audio Core IP	~500-1,500
VGA Module	~1,200–1,800
Avalon-MM / Streaming Interface	~300–700
Control FSMs & Config Registers	~200–500

Total FPGA Memory Utilization:

- Block RAM: ~300 KB
- Logic Elements: ~2,200–5,000

Performance Constraints

Constraint	Target Value	Notes
VGA Refresh Rate	60 Hz (16.7 ms/frame)	Scan loop in vga.sv
Audio Output Latency	< 20 ms	Buffered playback prevents audible delay
HPS Sprite Rendering Time	< 8–10 ms/frame	Followed by pixel-by-pixel MMIO copy

5. The Hardware/Software Interface

All hardware-software interaction is based on memory-mapped I/O (MMIO). The system uses standard IPs for audio, and a custom VGA display pipeline using a software-driven framebuffer communication mechanism. No Avalon streaming or SDRAM buffers are used.

Audio Interface

The audio path is implemented using the Audio Core IP, which interfaces with the WM8731 codec via I²S. Only the playback (sound output) path is enabled.

- Function: Transmit 16-bit PCM audio samples from HPS to FPGA for playback. The IP includes two output FIFOs (left and right channels), each with a depth of 128 × 32-bit words.
- Software Interaction: HPS writes audio samples directly to the left and right FIFO registers in alternating order. No interrupt or DMA is used; polling or buffered writes are performed in software.

Register Map Reference:

Table 1. Audio core register map

Offset in bytes	**Register Name**	**R/W**	**Bit Description**									
			31…24	**23…16**	**15…10**	**9**	**8**	**7…4**	**3**	**2**	**1**	**0**
0	`control`	RW	*(1)*			WI	RI	*(1)*	CW	CR	WE	RE
4	`fifospace`	R	WS LC	WS RC	RA LC			RA RC				
8	`leftdata`	RW *(2)*	Left Data									
12	`rightdata`	RW *(2)*	Right Data									

Notes on Table 1:

(1) Reserved. Read values are undefined. Write zero.

(2) Only reads incoming audio data and writes outgoing audio data.

4.1.1 *Control* Register

Table 2. Control register bits

Bit number	**Bit name**	**Read/Write**	**Description**
0	RE	R/W	Interrupt-enable bit for read interrupts. If the RE bit is set to 1 and both the left and right channel read FIFOs contain data, the Audio core generates an interrupt request (IRQ).
1	WE	R/W	Interrupt-enable bit for write interrupts. If the WE bit is set to 1 and both the left and right channel write FIFOs have space available for more data, the Audio core generates an interrupt request (IRQ).
2	CR	R/W	Clears the Audio core's Input FIFOs, when the bit is 1. Clear remains active until specifically set to zero.
3	CW	R/W	Clears the Audio core's Output FIFOs, when the bit is 1. Clear remains active until specifically set to zero.
8	RI	R	Indicates that a read interrupt is pending.
9	WI	R	Indicates that a write interrupt is pending.

Display Interface

The video output is implemented using a custom Verilog module vga.sv, which maintains a framebuffer in FPGA on-chip memory and drives VGA signals in real time.

- Function: Receive 32-bit packed pixel data from the HPS via MMIO and update the corresponding location in the on-chip framebuffer. The module continuously scans the framebuffer and generates VGA signals at 640×480, 60 Hz.
- Software Interaction: After rendering each frame in software, the HPS sends pixel data to vga.sv using a single 32-bit write per pixel. Each write contains the x and y coordinate, a 4-bit color index, and a write-enable flag. A kernel module (framebuffer.ko) exposes the MMIO interface to user space for efficient pixel transmission.

Register Map:

Offset	Name	Access	Width	Description
0x00	pixel_write	Write-only	32 bits	Packed pixel data (see table* below)

*Video Pixel Write Data Format (32-bit):

Bit Range	Field	Description
31:24	Reserved	(Unused)
23:14	x	X-coordinate (10 bits, 0–639)
13:5	y	Y-coordinate (9 bits, 0–479)
4:1	color_index	4-bit color index (0–15)
0	write_enable	Set to 1 to trigger pixel write