Pop’n FPGA Rhythm Game — Design Document

Team Member

- Vince-Arvin Magno (vm2787)

1. Introduction & Core Concept

This project is a hardware/software co-designed rhythm game inspired by the arcade game Pop’n Music. The system parses a beatmap, plays synchronized audio through the onboard Wolfson WM8731 codec, and challenges the player to press keys on a USB keyboard as falling notes cross a judgment line on a VGA display.

The design demonstrates real-time I/O handling, hardware-accelerated sprite rendering at 640 × 480 at 60 Hz, and precise timing synchronization between a Linux userspace process running on the ARM Cortex-A9 HPS and custom FPGA fabric on the Cyclone V DE1-SoC.

The hardware is responsible for VGA timing generation, lane background rendering, falling note sprites, and button-glow effects—all at 60 frames/s without per-frame software involvement. The software handles beatmap parsing, timer synchronization with audio playback, USB keyboard input via Linux evdev, hit/miss detection, score tracking, and per-frame writes of note Y-coordinates to FPGA registers over the HPS-to-FPGA bridge.

2. System Block Diagram

The system is divided into a software layer on the ARM HPS and a hardware accelerator in the FPGA fabric. Communication between them occurs through memory-mapped registers on two separate HPS-to-FPGA bridges:

Heavyweight AXI bridge (0xC0000000): Carries the game display registers (popn_engine_0). The software writes note Y-coordinates and glow masks here at 60 Hz.

Lightweight AXI bridge (0xFF200000): Carries the audio subsystem (Audio core, Audio/Video Config core). The software writes PCM audio samples here for playback through the codec.

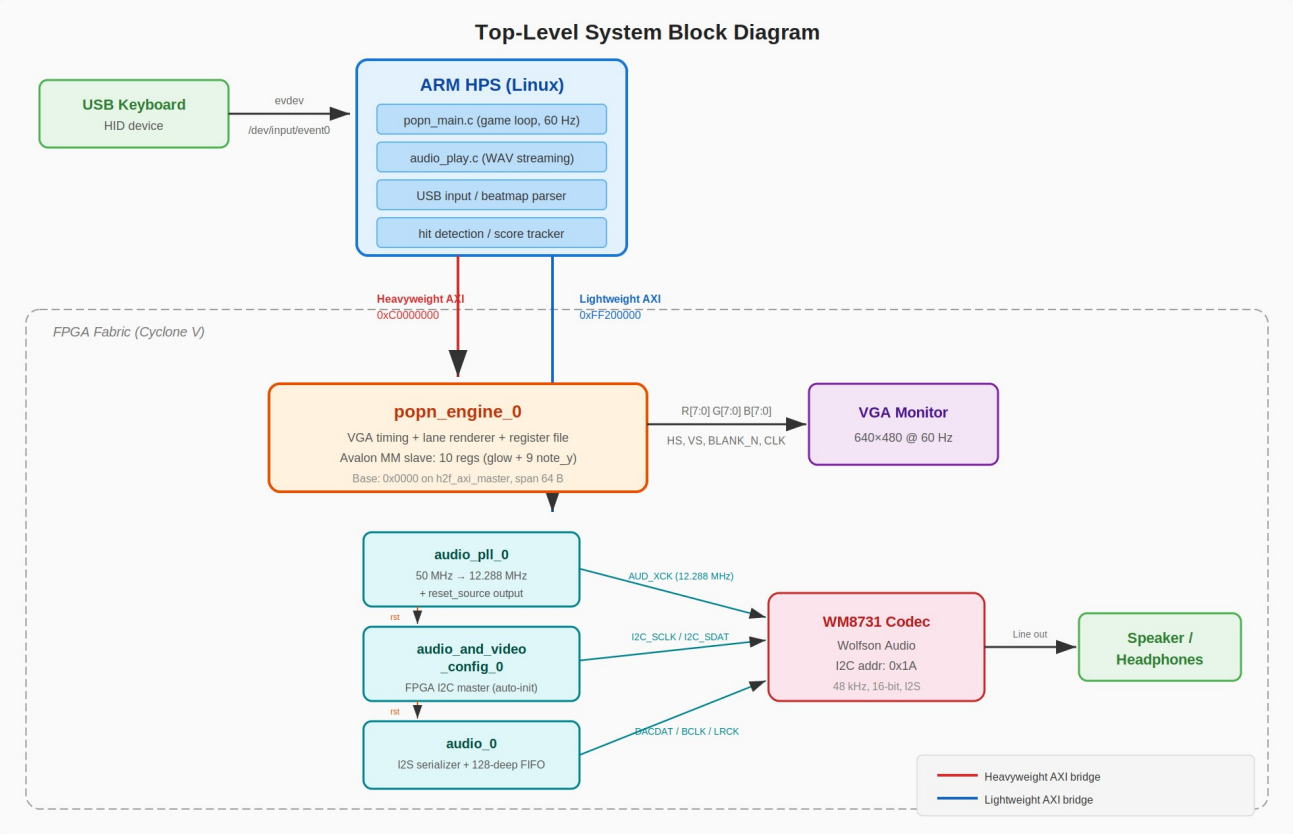

Figure 1: Top-Level System Block Diagram

2.1 Platform Designer (Qsys) Component Map

Component	Bridge	Base Address	Span
popn_engine_0	h2f_axi_master (heavyweight)	0x0000_0000	64 B (0x00–0x3F)
audio_and_video_config_0	h2f_lw_axi_master (lightweight)	0x0000_0000	16 B (0x00–0x0F)
audio_0	h2f_lw_axi_master (lightweight)	0x0000_0010	16 B (0x10–0x1F)
audio_pll_0	Internal clock source	N/A	50 MHz → 12.288 MHz
hps_0	HPS hard processor system	N/A	DDR3, USB, SD, UART

3. Hardware-Software Interface: Register Map

The software communicates with the FPGA through memory-mapped registers. It uses mmap() on /dev/mem to access the physical bridge addresses, then reads and writes 32-bit words at specific offsets.

3.1 popn_engine_0 — Game Display Controller (at 0xC0000000)

All registers are 32-bit, write-only from the HPS. The software writes these at 60 Hz to update the display. The hardware reads them combinationally each pixel clock to determine what to draw.

Byte Offset	Name	Bits Used	Description
0x00	lane_glow_mask	[8:0]	Bitmask: bit i = 1 means lane i is glowing (key held). Written on every key press/release.
0x04	note_y[0]	[9:0]	Y-coordinate of the falling note in lane 0. Range 0–479. Value 0x1FF = no active note.
0x08	note_y[1]	[9:0]	Y-coordinate of the falling note in lane 1.
0x0C	note_y[2]	[9:0]	Lane 2 note Y.
0x10	note_y[3]	[9:0]	Lane 3 note Y.
0x14	note_y[4]	[9:0]	Lane 4 (center) note Y.
0x18	note_y[5]	[9:0]	Lane 5 note Y.
0x1C	note_y[6]	[9:0]	Lane 6 note Y.
0x20	note_y[7]	[9:0]	Lane 7 note Y.
0x24	note_y[8]	[9:0]	Lane 8 note Y.

3.2 audio_0 — Audio Core (at 0xFF200010)

Read/write from the HPS. The software polls fifospace before writing each stereo sample pair.

Byte Offset	Name	Access	Description
0x00	control	R/W	[3] CW: clear write FIFO. [2] CR: clear read FIFO. [1] WI: write interrupt enable. [0] RI: read interrupt enable.
0x04	fifospace	Read	[31:24] WSLC (write-space left). [23:16] WSRC (write-space right). [15:8] RALC. [7:0] RARC. 0x80 = 128 slots free (empty).
0x08	leftdata	R/W	Write: push a 32-bit signed sample into the left-channel write FIFO.
0x0C	rightdata	R/W	Write: push a 32-bit signed sample into the right-channel write FIFO.

3.3 audio_and_video_config_0 — I2C Configuration (at 0xFF200000)

This core automatically sends eight I2C register writes to the WM8731 codec (address 0x1A) on system reset. It configures the codec for a 48 kHz sample rate, I2S format, and DAC output. The

status register at byte offset 0x0C reads 0xFF when all eight commands have been sent. The core drives FPGA_I2C_SCLK and FPGA_I2C_SDAT, which are FPGA fabric pins (not the HPS I2C controller) routed directly to the codec.

4. Hardware Design: popn_engine Module

4.1 Module Interface

```
module popn_engine (
    input  logic        clk,           // 50 MHz from Qsys clk_0
    input  logic        reset,         // From Qsys reset controller
    input  logic [3:0]  avs_address,    // Avalon MM slave address (word index)
    input  logic        avs_write,      // Avalon write strobe
    input  logic [31:0] avs_writedata,  // Avalon write data
    output logic [7:0]  vga_r, vga_g, vga_b,  // 8-bit RGB to DAC
    output logic        vga_clk,        // 25 MHz pixel clock to VGA DAC
    output logic        vga_hs, vga_vs,  // Horizontal/vertical sync
    output logic        vga_blank_n,    // Active display region
    output logic        vga_sync_n      // Sync-on-green (unused)
);
```

4.2 Internal Block Diagram

The popn_engine module contains the following sub-blocks. All logic is defined within a single file, popn_engine.sv; no separate instantiated submodules are used.

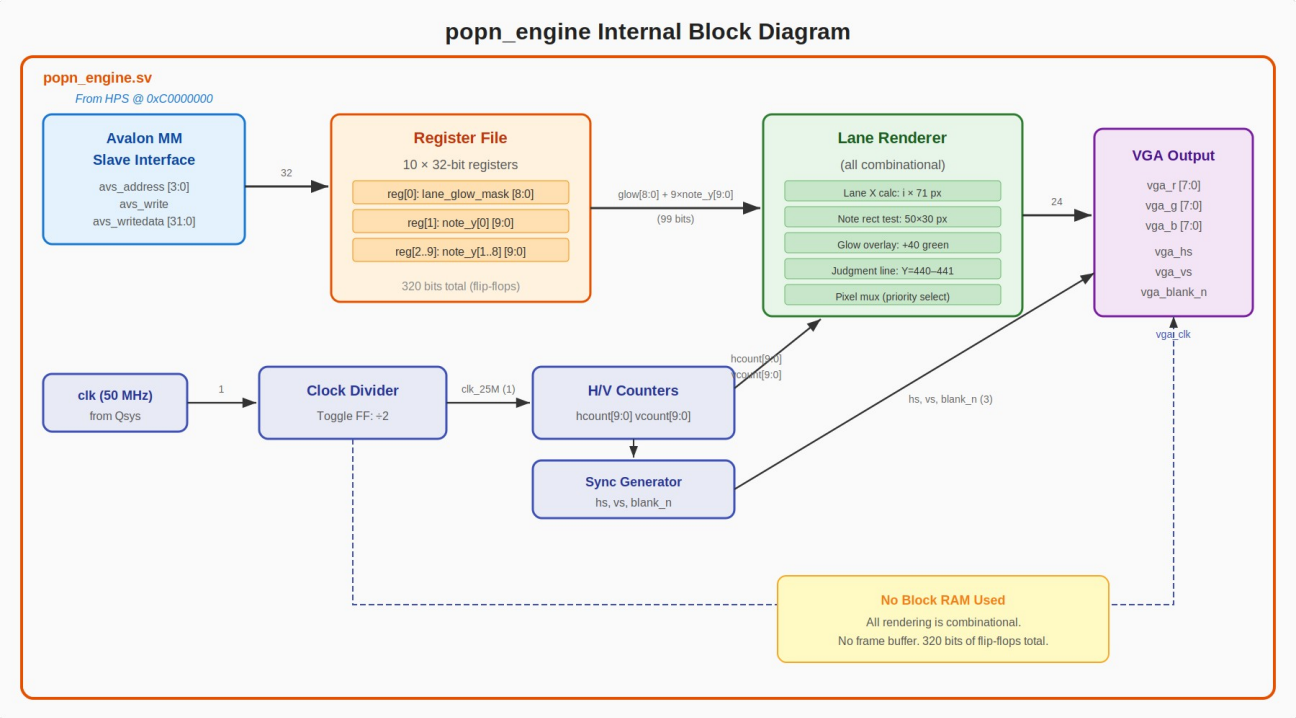

Figure 2: popn_engine Module Internal Block Diagram

Sub-Block	Type	Width	Function
Clock divider	Sequential (toggle FF)	1 bit	Divides 50 MHz clk to 25 MHz pixel clock (clk_25M) using a single always_ff toggle.
H/V counters	Sequential	10 bits each	hcount: 0–799. vcount: 0–524. Standard 640×480 @ 60 Hz VGA timing.
Register file	Sequential	32 bits × 10	10 registers written by HPS via Avalon MM. Addr 0 = lane_glow_mask[8:0]. Addr 1–9 = note_y[0..8][9:0].
Lane X calculator	Combinational	10 bits	lane_x[i] = i × 71. Evenly divides 640 px across 9 lanes.
Note rectangle test	Combinational	1 bit	True if pixel (hcount, vcount) is inside a 50×30 px rectangle at (lane_x[i], note_y[i]).
Glow overlay	Combinational	1 bit	If lane_glow_mask[i] is set, the lane background brightens (+40 to green channel).
Judgment line	Combinational	1 bit	White 2 px horizontal line at Y = 440–441 spanning all 9 lanes.
Pixel mux	Combinational	24 bits (RGB)	Priority: note sprite > glow > judgment line > lane background > black.

4.3 VGA Timing

Parameter	Horizontal	Vertical
Active pixels	640	480
Front porch	16	10
Sync pulse	96	2
Back porch	48	33
Total	800	525

Pixel clock: 25 MHz. Frame rate: 60.0 Hz. Active display: blank_n is high only when hcount < 640 and vcount < 480.

4.4 Memory Usage

The popn_engine module uses no block RAM or on-chip memory. All state is held in 10 registers (320 bits of flip-flops). The lane renderer is purely combinational: pixel color is computed from (hcount, vcount) and the register file contents on every pixel clock cycle. No frame buffer is required because the display is procedurally generated in real time.

5. Audio Subsystem

Audio playback uses three Altera University Program IP cores instantiated in Platform Designer, all connected through the HPS lightweight AXI bridge at 0xFF200000.

Audio PLL (audio_pll_0): Generates 12.288 MHz master clock from the 50 MHz reference. Output drives AUD_XCK pin (PIN_G7). Configuration: 50 MHz × 29 / 2 / 59 = 12.288135 MHz.

Audio/Video Config (audio_and_video_config_0): FPGA-side I2C master that auto-initializes the WM8731 codec on reset. Drives FPGA_I2C_SCLK (PIN_J12) and FPGA_I2C_SDAT (PIN_K12). These are FPGA fabric pins, not the HPS I2C controller.

Audio Core (audio_0): Dual-channel 128-entry FIFO with I2S serializer. Software pushes 32-bit signed samples via MMIO. Hardware serializes them as AUD_DACDAT synchronized to AUD_BCLK and AUD_DACLRCK.

5.1 Clock and Reset Hierarchy

The audio subsystem follows the reset sequence below to ensure proper initialization:

- audio_pll_0.ref_clk connects to clk_0 (50 MHz). audio_pll_0.ref_reset connects to clk_0.clk_reset.
- audio_pll_0.reset_source stays asserted until the PLL achieves lock at 12.288 MHz.
- Both audio_0.reset and audio_and_video_config_0.reset are driven by audio_pll_0.reset_source, ensuring they do not exit reset until MCLK is stable.
- The top-level reset_reset_n is driven by KEY[0] (active-low pushbutton), providing a physical reset capability.

5.2 Audio Pin Assignments

Signal	Pin	Direction	Function
AUD_XCK	PIN_G7	Output	12.288 MHz master clock to codec
AUD_BCLK	PIN_H7	Input (bidir)	Bit clock generated by codec
AUD_DACLRCK	PIN_H8	Input (bidir)	DAC left/right clock from codec
AUD_DACDAT	PIN_J7	Output	Serial audio data to codec DAC
AUD_ADCLRCK	PIN_K8	Input (bidir)	ADC left/right clock from codec
AUD_ADCDAT	PIN_K7	Input	Serial audio data from codec ADC
FPGA_I2C_SCLK	PIN_J12	Output	I2C clock to codec config registers
FPGA_I2C_SDAT	PIN_K12	Bidir	I2C data to/from codec config

6. Software Design

6.1 Software Components

Component	File	Interface	Description
Main game loop	popn_main.c	mmap @ 0xC0000000	Runs at 60 Hz. Reads input, advances note positions, writes FPGA registers, checks hit timing.
USB input handler	popn_main.c	Linux evdev (O_NONBLOCK)	Reads /dev/input/eventX. Maps KEY_A through KEY_SEMICOLON to lanes 0–8.
Beatmap parser	popn_main.c	fopen / fscanf	Reads timestamped note schedule from a text file at startup into a sorted array.
Audio playback	audio_play.c	mmap @ 0xFF200000	Parses WAV header, streams 16-bit PCM samples to audio FIFO, polling

Component	File	Interface	Description
			fifospace.
Hit detection	popn_main.c	In-process logic	On key press: abs(note_y[lane] − 440) < 20 → PERFECT; < 40 → GOOD; else MISS.
Score tracker	popn_main.c	printf to terminal	Accumulates PERFECT / GOOD / MISS counts and combo counter.

6.2 Key Mapping

Nine keys are mapped to the nine lanes, mirroring the physical layout of a Pop'n Music arcade controller:

Key	Linux evdev Code	Lane
A	KEY_A (30)	0 (leftmost)
S	KEY_S (31)	1
D	KEY_D (32)	2
F	KEY_F (33)	3
Space	KEY_SPACE (57)	4 (center)
J	KEY_J (36)	5
K	KEY_K (37)	6
L	KEY_L (38)	7
;	KEY_SEMICOLON (39)	8 (rightmost)

6.3 Audio Playback Protocol

The audio_play program opens a WAV file, parses the RIFF/fmt/data headers, and then enters a tight polling loop:

```
for each sample pair (L, R) in the WAV data:
  do { st = audio[1]; }                  // read fifospace
  while (((st >> 24) & 0xFF) == 0 ||          // WSLC > 0?
       ((st >> 16) & 0xFF) == 0);          // WSRC > 0?
  audio[2] = (uint32_t)L;                 // push left sample
  audio[3] = (uint32_t)R;                 // push right sample
```

The I2S serializer drains the FIFO at 48 kHz. Back-pressure is handled by the polling loop: when the FIFO is full, the software spins until space opens.

7. File Formats

7.1 Beatmap File (.txt)

Plain text, one note per line. Each line contains a timestamp in milliseconds and a lane index (0–8), separated by whitespace:

```
1200  4
1400  2
1400  6
1800  0
```

The parser reads all entries at startup into a time-sorted array. During gameplay, the game loop spawns a falling note when the game clock reaches each entry's timestamp.

7.2 WAV Audio File

Standard RIFF WAV format. Supported: 16-bit signed PCM, mono or stereo. Preferred sample rate: 48 kHz (matches the WM8731 default configuration).

8. C Interface Declarations

These constants define the software interface to the hardware:

```
/* popn_main.c - Game display registers */
#define HW_BRIDGE_BASE   0xC0000000
#define HW_BRIDGE_SPAN   0x04000000
#define GLOW_REG        0    /* word offset: lane_glow_mask */
#define NOTE_Y_BASE     1    /* word offset: note_y[0] */
#define NUM_LANES       9
#define JUDGE_Y         440
#define PERFECT_WINDOW   20   /* pixels */
#define GOOD_WINDOW      40   /* pixels */
```

```
/* audio_play.c - Audio FIFO registers */
#define LW_BRIDGE_BASE   0xFF200000
#define LW_BRIDGE_SPAN   0x00100000
#define AUDIO_BASE      0x10  /* byte offset from LW bridge */
/* audio[0]=control  audio[1]=fifospace */
/* audio[2]=leftdata audio[3]=rightdata */
```

9. Verilog Module Hierarchy

Every module instantiated in the design is listed below:

```
soc_system_top.sv                (top-level entity for Quartus)
 +-- soc_system                  (Qsys/Platform Designer generated)
 |    +-- hps_0                  HPS: Cortex-A9, DDR3, USB, SD, UART
 |    +-- popn_engine_0          Custom: VGA renderer + game registers
 |    |    (internal)            clk_25M divider, hcount/vcount,
 |    |                         register file, lane renderer, pixel mux
 |    +-- audio_pll_0            Altera PLL: 50 MHz -> 12.288 MHz
 |    +-- audio_and_video_config_0  I2C master for WM8731 auto-init
 |    +-- audio_0                I2S serializer + dual 128-deep FIFO
```

10. Build and Deployment

Full build sequence from source to running on the board:

```
# On the workstation:
cd ~/Downloads/lab3-project/
qsys-generate soc_system.qsys --synthesis=VERILOG
quartus_sh --flow compile soc_system
quartus_cpf -c output_files/soc_system.sof output_files/soc_system.rbf
cp output_files/soc_system.rbf /run/media/vm2787/A26C-17FF/
sync && udisksctl unmount -b /dev/sda1
```

```
# On the DE1-SoC board after boot:
./memwrite FFD05014 0            # release HPS-to-FPGA bridges
./popn /dev/input/event0         # launch the game
./audio_play song.wav &          # start audio in background
```