CSEE 4840 Embedded Systems - Project Design Document

Project Title: Air Hockey

Group Members:
Gerald Zhao - zz3427
Derrick Chen - dc3494
Zening Wang - zw3161
Xuepeng Han - xh2718

Spring 2026

Contents

1 Introduction

This project involves the implementation of a real-time, two-player digital Air Hockey arcade game on an FPGA SoC platform. To emulate the experience of a traditional arcade, the system is designed to output a 2D top-down perspective to a large flat-panel display laid horizontally. Two players will stand on opposite sides and compete against each other. The gameplay includes continuous paddle control, 2D vector collision physics, synchronized audio feedback for impacts and scoring, and a match conclusion when either player reaches seven points.

Fundamentally, the system operates as a distributed algorithm divided into a software-driven game engine and a hardware-driven presentation. The software, running in Linux, will act as the brain. It will poll independent raw USB mouse data, calculate floating-point inelastic collision physics, and manage the game state. The hardware will act as the canvas and speaker. It guarantees a 60 Hz timing requirement of VGA video generation and audio, translating software-provided coordinates into on-screen pixels without interrupting the processor's physics calculations.

The remainder of the document will detail the technical implementations of the system:

- **System Block Diagram:** Illustration of the high-level architecture and dataflow.
- **Algorithms:** Details the floating-point vector mathematics and physics used for inelastic collisions (puck-to-paddle, puck-to-wall) and the state-machine logic for scoring.
- **Resource Budgets:** Estimates the logical elements, memory blocks, and processing bandwidth required to sustain the game in a 60 Hz loop.
- **The Hardware/Software Interface:** Defines the strict memory-mapped register contract and VSYNC communication protocol that bridges the software and hardware.

2 System Block Diagram

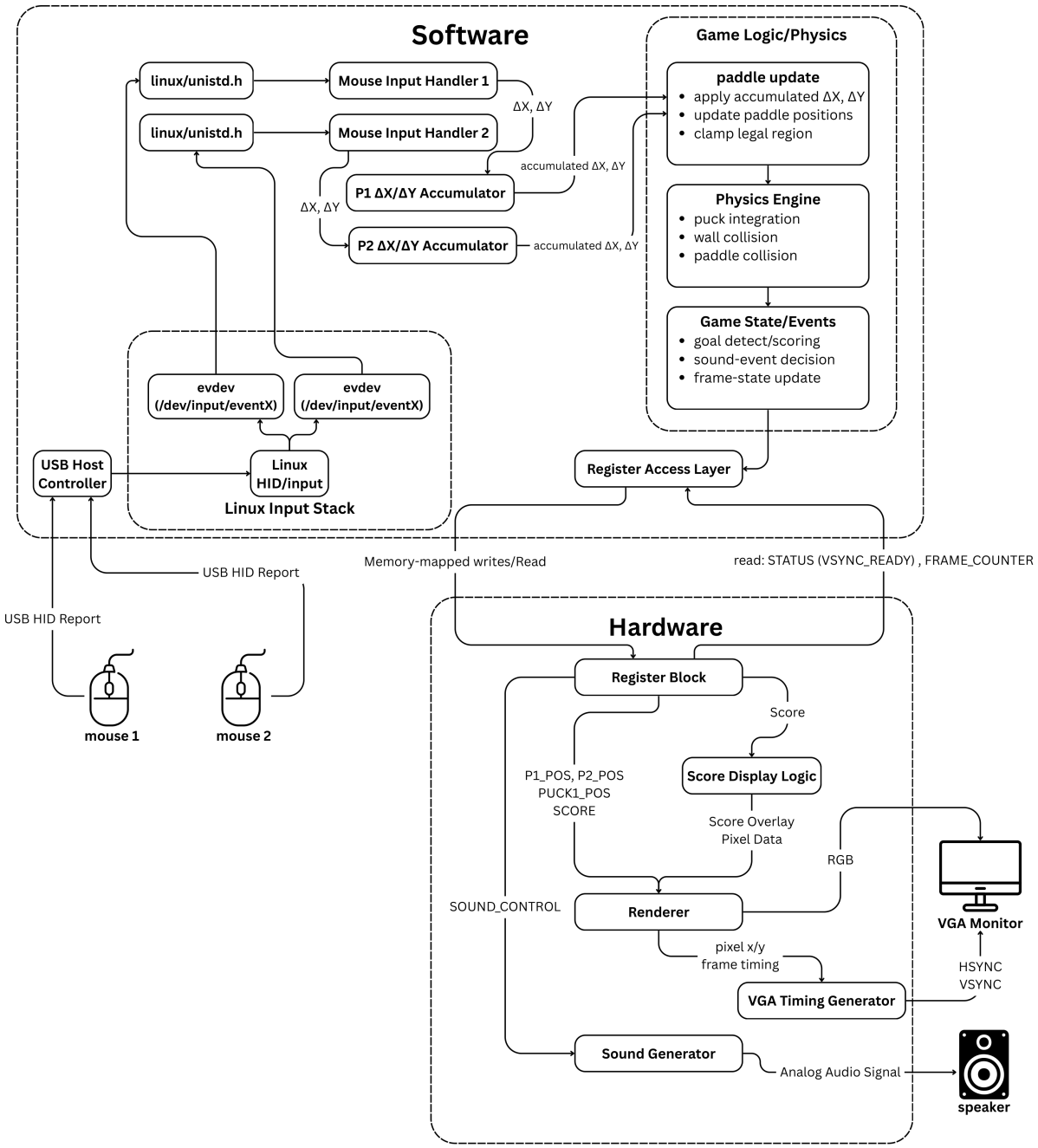

Figure 1: System Block Diagram

The system features two main components:

- **Software Stack:** Linux Input Stack (USB HID Report → evdev → libevdev), Mouse Input Handlers, Position Accumulators, Physics Engine (puck integration, wall/paddle collision), Game State/Events (goal detect, sound events), and the Register Access Layer.
- **Hardware Subsystem:** Register Block, Score Display Logic, Renderer, VGA Timing Generator, and Sound Generator.

3 Algorithms

This system operates as a distributed architecture. The complex game logic and calculations are computed in the software. Meanwhile, the pixel rendering and audio processing are executed as hardware algorithms on the FPGA.

The game loop runs at a fixed 60 Hz (16.67 ms). We parameterize the puck's movement using a time variable t, where $t = 0$ is the start of the frame, and $t = 1$ is the end of the frame.

Our game will be displayed on a 640×480 pixel screen. The origin (0, 0) will be located at the top-left corner of the screen. Y will increment downwards, while X increments towards the right. The wall boundaries will have a thickness of 10 pixels, effectively giving us a playable area of 620×460.

3.1 Uninterrupted Movement

During any given frame, the puck travels along a line. We define its position as a function of t where $0 \leq t \leq 1$.

$$X_{puck}(t) = X_{old} + V_x t$$
$$Y_{puck}(t) = Y_{old} + V_y t$$

Assuming that no collisions occur during this time frame, the puck's final position is simply when $t = 1$, thus sending

$$(X_{new}, Y_{new}) = (X_{puck}(1), Y_{puck}(1))$$

as the new coordinate.

3.2 Colliding with a Wall

The puck is bounded by the lines $Y = 10$, $Y = 470$, $X = 10$, and $X = 630$. Assuming that we know the puck's current position (X_{old}, Y_{old}) as well as its current velocity (V_x, V_y), we can calculate what happens when it collides with a wall. Note that we are also assuming the puck does not collide with any paddle during this time frame. This means we can effectively represent a collision against the wall as two separate cases of uninterrupted movement.

For example, the bottom wall is the boundary $Y = 470$. A collision here occurs when

$$Y_{old} + V_y t_c = 470 - r_{puck} \tag{1}$$

Solving for the time of collision, we find

$$t_c = \frac{470 - r_{puck} - Y_{old}}{V_y} \tag{2}$$

The point of collision is then

$$X_{collision} = X_{old} + V_x t_c$$
$$Y_{collision} = Y_{old} + V_y t_c$$

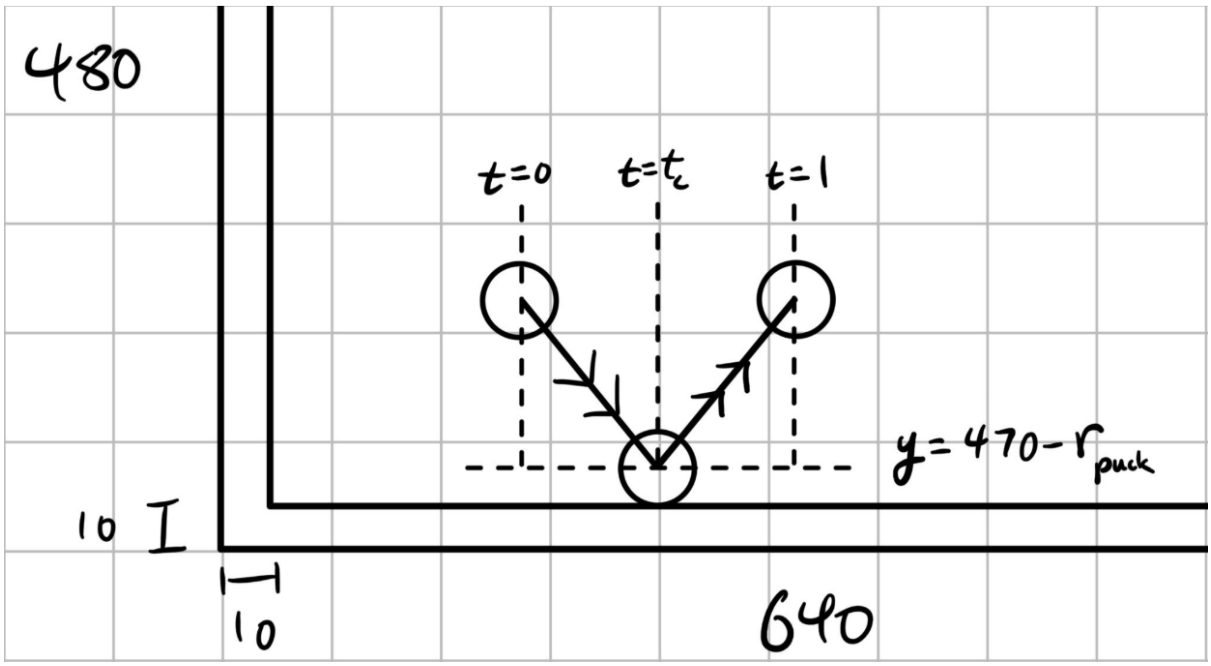

Figure 2: Bouncing Off a Wall

We apply a coefficient of restitution, C_R, to simulate energy loss during the collision. The velocity in the Y direction also flips signs. This is simply

$$V_{x_{new}} = V_{x_{old}} C_R$$
$$V_{y_{new}} = -V_{y_{old}} C_R$$

Finally, the puck moves away from the wall after the collision, with $t_{remaining} = 1 - t_c$

$$X_{new} = X_{collision} + V_{x_{new}} t_{remaining}$$
$$Y_{new} = Y_{collision} + V_{y_{new}} t_{remaining}$$

For the other three walls, the calculation is relatively the same.

- **Top Wall:** Check $Y_{old} + V_y t_c = 10 + r_{puck}$, flipping V_y.
- **Left Wall:** Check $X_{old} + V_x t_c = 10 + r_{puck}$, flipping V_x.
- **Right Wall:** Check $X_{old} + V_x t_c = 630 - r_{puck}$, flipping V_x.

3.3 Colliding with a Paddle

3.3.1 Time of Collision

To find the time of the collision between the puck and the paddle, we can approach this by making the paddle "stationary" relative to the puck. In other words, we calculate the relative position and velocity of the puck from the paddle's perspective. Note that this is only to solve for t_c and that the resulting velocities after the collision will be calculated differently.

$$\begin{aligned}\Delta P_x &= X_{puck_{old}} - X_{paddle_{old}} \\ \Delta P_y &= Y_{puck_{old}} - Y_{paddle_{old}} \\ \Delta V_x &= V_{x_{puck_{old}}} - V_{x_{paddle_{old}}} \\ \Delta V_y &= V_{y_{puck_{old}}} - V_{y_{paddle_{old}}}\end{aligned}$$

A collision between the puck and paddle occurs if the squared distance between their center points is equal to the square of their combined radii: $R_{sum} = R_{puck} + R_{paddle}$.

If we track the puck's relative movement over time, we can find the exact time t_c of the collision. The relative position between the two centers at any given time is

$$P(t) = \Delta P + \Delta V \cdot t \tag{3}$$

The collision occurs when this magnitude is equal to the combined radii

$$||P(t)|| = ||\Delta P + \Delta V \cdot t|| = R_{sum} \tag{4}$$

By looking at the squared distance, we can see that

$$||\Delta P + \Delta V \cdot t||^2 = (\Delta P + \Delta V \cdot t)^2 = (R_{sum})^2 \tag{5}$$

Expanding, we find

$$(\Delta V \cdot \Delta V)t^2 + 2(\Delta P \cdot \Delta V)t + (\Delta P \cdot \Delta P) - (R_{sum})^2 = 0 \tag{6}$$

Let

$$\begin{aligned}A &= (\Delta V_x)^2 + (\Delta V_y)^2 \\ B &= 2 \times [(\Delta P_x \cdot \Delta V_x) + (\Delta P_y \cdot \Delta V_y)] \\ C &= (\Delta P_x)^2 + (\Delta P_y)^2 - (R_{sum})^2\end{aligned}$$

Then we can solve this quadratic equation, finding

$$t_c = \frac{-B - \sqrt{B^2 - 4AC}}{2A} \tag{7}$$

Similarly to the case of colliding with the wall, if $0 \leq t_c \leq 1$ the collision occurs within the current time frame.

3.3.2 Resulting Velocities

In order to calculate the resulting velocity after the collision, we will treat the paddle as an object with infinite mass. This approach is reasonable since the "push-back" felt on the paddle by the puck will never affect the player. Consider the equations for conservation of momentum and restitution:

$$m_{puck}v_{puck_{old}} + m_{paddle}v_{paddle_{old}} = m_{puck}v_{puck_{new}} + m_{paddle}v_{paddle_{new}} \tag{8}$$

$$e = \frac{v_{paddle_{new}} - v_{puck_{new}}}{v_{puck_{old}} - v_{paddle_{old}}} \tag{9}$$

Note that v is a vector. Additionally, e measures the elasticity of the collision. We will use these two known equations to solve for two variables, $v_{puck_{new}}$ and $v_{paddle_{new}}$.

$$v_{paddle_{new}} - v_{puck_{new}} = e(v_{puck_{old}} - v_{paddle_{old}})$$

$$v_{puck_{new}} = v_{paddle_{new}} - e(v_{puck_{old}} - v_{paddle_{old}}) \tag{10}$$

Plugging this into the momentum equation:

$$m_{puck}v_{puck_{old}} + m_{paddle}v_{paddle_{old}} = m_{puck}(v_{paddle_{new}} - e(v_{puck_{old}} - v_{paddle_{old}})) + m_{paddle}v_{paddle_{new}}$$

$$m_{puck}v_{puck_{old}} + m_{paddle}v_{paddle_{old}} = (m_{puck} + m_{paddle})v_{paddle_{new}} - m_{puck}e(v_{puck_{old}} - v_{paddle_{old}})$$

$$m_{puck}v_{puck_{old}} + m_{paddle}v_{paddle_{old}} + m_{puck}e(v_{puck_{old}} - v_{paddle_{old}}) = (m_{puck} + m_{paddle})v_{paddle_{new}}$$

$$v_{paddle_{new}} = \frac{m_{puck}v_{puck_{old}} + m_{paddle}v_{paddle_{old}} + m_{puck}e(v_{puck_{old}} - v_{paddle_{old}})}{m_{puck} + m_{paddle}}$$

Dividing the top and bottom by m_{paddle}, we get

$$v_{paddle_{new}} = \frac{\frac{m_{puck}}{m_{padde}}v_{puck_{old}} + v_{paddle_{old}} + \frac{m_{puck}}{m_{padde}}e(v_{puck_{old}} - v_{paddle_{old}})}{\frac{m_{puck}}{m_{padde}} + 1}$$

Since the paddle is being treated as infinite mass, we take $m_{paddle} \to \infty$, meaning $\frac{m_{puck}}{m_{padde}} \to 0$.

This leaves us with

$$v_{paddle_{new}} = v_{paddle_{old}} \tag{11}$$

To find $v_{puck_{new}}$, we simply go back to our earlier equation:

$$v_{puck_{new}} = (1 + e)v_{paddle_{old}} - e(v_{puck_{old}}) \tag{12}$$

Note that if our system is completely elastic, we take $e = 1$, giving us

$$v_{puck_{new}} = 2v_{paddle_{old}} - v_{puck_{old}}$$

$$v_{paddle_{new}} = v_{paddle_{old}}$$

3.3.3 Post-Collision Vector

We first calculate the absolute global coordinates of both objects at the exact moment of impact.

$$
\begin{aligned}
X_{puck_{impact}} &= X_{puck_{old}} + V_{x_{puck_{old}}} \cdot t_c \\
Y_{puck_{impact}} &= Y_{puck_{old}} + V_{y_{puck_{old}}} \cdot t_c \\
X_{paddle_{impact}} &= X_{paddle_{old}} + V_{x_{paddle_{old}}} \cdot t_c \\
Y_{paddle_{impact}} &= Y_{paddle_{old}} + V_{y_{paddle_{old}}} \cdot t_c
\end{aligned}
$$

The collision force acts along the position vector pointing from the paddle's center to the puck's center. We define this as the position vector N, where

$$
\begin{aligned}
N_x &= X_{puck_{impact}} - X_{paddle_{impact}} \\
N_y &= Y_{puck_{impact}} - Y_{paddle_{impact}}
\end{aligned}
$$

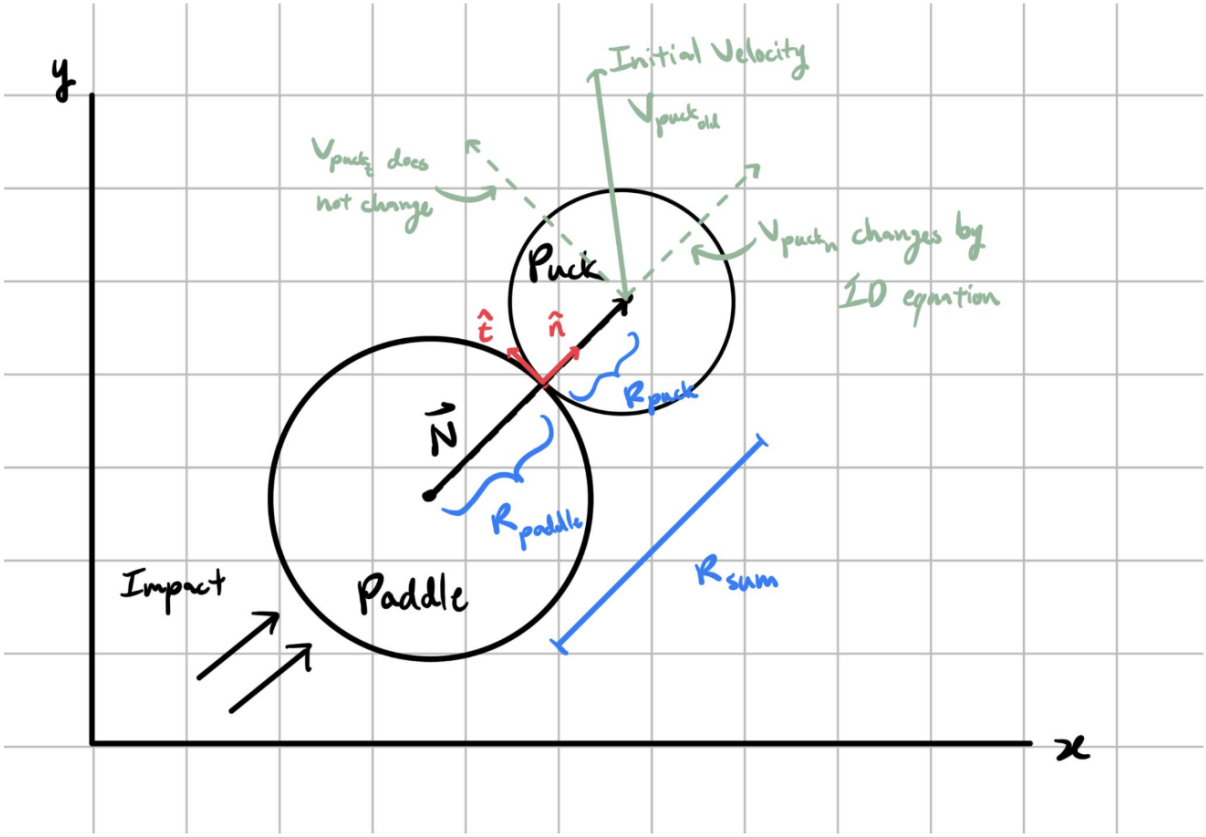

Figure 3: Bouncing Off a Paddle

The magnitude of N is equal to R_{sum}. This means we can find a normal unit vector $\hat{n}$ where $\hat{n_x} = \frac{N_x}{R_{sum}}$ and $\hat{n_y} = \frac{N_y}{R_{sum}}$.

The tangent vector $\hat{t}$ is the vector orthogonal to n. It is defined as $\hat{t_x} = -\hat{n_y}$ and $\hat{t_y} = \hat{n_x}$.

The equation we derived earlier only works in 1D, meaning it's used for head-on collisions. Since our game is in 2D, we can create a temporary coordinate system for the collision:

1. Normal axis $\hat{n}$: our new x-axis, cuts through the centers of both circles and the point of collision.
2. Tangent axis $\hat{t}$: our new y-axis, perpendicular to the normal axis.

Along the normal axis, the system behaves exactly as a 1D head-on collision, meaning we can use the formula that we derived earlier. Along the tangential axis, because there is no friction, the interaction does not change the velocity in this direction.

Let u be any velocity prior to impact, and v be any velocity after impact. Then,

$$\begin{aligned} u_{puck_n} &= (V_{x_{puck_{old}}} \cdot \hat{n_x}) + (V_{y_{puck_{old}}} \cdot \hat{n_y}) \\ u_{puck_t} &= (V_{x_{puck_{old}}} \cdot \hat{t_x}) + (V_{y_{puck_{old}}} \cdot \hat{t_y}) \\ u_{paddle_n} &= (V_{x_{paddle_{old}}} \cdot \hat{n_x}) + (V_{y_{paddle_{old}}} \cdot \hat{n_y}) \end{aligned}$$

We can now see that

$$\begin{aligned} v_{puck_n} &= 2u_{paddle_n} - u_{puck_n} \\ v_{puck_t} &= u_{puck_t} \end{aligned}$$

Converting this back to global X and Y coordinates, we have

$$\begin{aligned} v_{puck_x} &= (v_{puck_n} \cdot \hat{n_x}) + (v_{puck_t} \cdot \hat{t_x}) \\ v_{puck_y} &= (v_{puck_n} \cdot \hat{n_y}) + (v_{puck_t} \cdot \hat{t_y}) \end{aligned}$$

Now that we've established a new velocity vector for the puck, we know that the puck travels along this path for the rest of the time frame $t_{remaining} = 1 - t_c$.

The final coordinates are then

$$\begin{aligned} X_{puck_{new}} &= X_{puck_{impact}} + v_{puck_x} t_{remaining} \\ Y_{puck_{new}} &= Y_{puck_{impact}} + v_{puck_y} t_{remaining} \end{aligned}$$

3.4 Scoring a Goal

Along with the standard bouncing-off-a-wall logic, one consideration that we must make along the left and right walls is when a goal is scored. For example, if we decide that our goals are 120 pixels large, then the goal openings exist between $Y = 180$ and $Y = 300$. We calculate the hypothetical Y-coordinate of the puck at the exact moment it intersects the left boundary plane ($X = 10$).

$$t_c = \frac{10 + r_{puck} - X_{old}}{V_x} \tag{13}$$

$$Y_{collision} = Y_{old} + V_y t_c \tag{14}$$

To handle the edge case of hitting the corner of the goal, we model the top and bottom corners as stationary dummy paddles with $R_{post} = 10$ pixels and infinite mass. The centers are placed at $(0, 170)$ and $(0, 310)$. Consider the following cases:

1. **Flat Wall:** If $Y_{collision} \leq 170$ or $Y_{collision} \geq 310$, the puck has collided with the wall. Apply standard wall collision.
2. **Goal Zone:** If $180 + r_{puck} < Y_{collision} < 300 - r_{puck}$, the puck has completely cleared both rounded corners and has made a goal.
3. **Top Goal Post:** If $170 < Y_{collision} \leq 180 + r_{puck}$, the puck won't strike the wall but will instead intersect the rounded edge of the top post. Apply paddle logic.
4. **Bottom Goal Post:** If $300 - r_{puck} \leq Y_{collision} < 310$, the puck is on a trajectory to hit the bottom goal post. Apply paddle logic.

4 Resource Budgets

4.1 Resource Budgets Table

Category	Time (s)	Size (bits)
Puck hitting wall	0.05	10,024
Puck hitting paddle	0.06	13,368
Score	0.5	70,216
Total Sound Memory Budget (Bits)		**93,608**

For rendering the paddles and puck, the hardware will use a purely computational approach. For each pixel, the hardware evaluates the condition $x^2 + y^2 \leq r^2$, where x and y are the pixel's offsets from the object's center coordinate.

If the condition is satisfied, the pixel is rendered in the object's color, and as a result no on-chip block RAM is needed for graphics storage. All visual elements including the paddles, puck, rink boundaries, and center line are drawn procedurally through comparator and multiplier logic, leaving the on-chip memory budget available exclusively for audio sample storage.

Rather than saving the images of 0 through 7 in memory, we've decided to implement logic to display the correct score for each player. We use a simple 7-segment display case-logic that decides which segments light up depending on the SCORE register.

Because sound is a expensive resource, we opted to reserve all of our memory for sound clips. The computational trade-off is not expensive, as this calculation should be simple for an FPGA-board. There are enough multipliers on the FPGA board to do the calculation comfortably.

5 The Hardware/Software Interface

5.1 Interface Design Overview

5.1.1 Interface Philosophy

The hardware/software interface for the air hockey system is implemented as a **memory-mapped register interface**. The interface is designed to match the hardware/software partition: software handles controller input, game logic, collision physics, score updates, and sound-event decisions, while hardware handles VGA output, on-screen rendering, frame timing, and sound playback. The interface is state-based rather than command-based. Software does not send low-level drawing commands to hardware. Instead, once per frame, software writes the current visible game state into a small set of registers, and hardware uses those values directly to render the next frame.

5.1.2 Information Passed Between Software and Hardware

For a holistic view, the information transferred from software to hardware consists of the current game state:

1. Player 1, 2 paddles center positions
2. Puck center position
3. Current score
4. Sound event request for the current frame

The information transferred from hardware back to software consists of status and synchronization information:

1. Whether the display is currently in a safe update interval
2. Whether the display is enabled and active
3. A frame counter for debugging and validation

This interface reflects the division of responsibility: software decides what the game state is, while hardware decides how that state is displayed and sounded.

5.1.3 Register Organization

All interface registers are 32 bits wide and are memory-mapped into the processor address space. The register set is divided into three functional groups:

1. **Control/status registers**, used for reset, enable, and synchronization
2. **Game-state registers**, used to communicate object positions and score
3. **Sound-control registers**, used to request playback of collision or goal sounds

5.1.4 Coordinate Encoding and Frame Data Format

The paddle and puck positions are communicated as center coordinates. Each object uses one 32-bit register, with: bits $[15:0]$ storing the x-coordinate, and bits $[31:16]$ storing the y-coordinate.

Although the visible screen resolution only requires 10 bits for x and 9 bits for y at 640×480 resolution, 16 bits are allocated to each coordinate for consistency.

The hardware renderer interprets these coordinates as the centers of circular objects. Paddle radius and puck radius are fixed constants inside the hardware design rather than configurable through the interface.

5.1.5 Frame Update Model

The game operates on a **per-frame update model**. Software computes the next frame's game state and writes it into the hardware registers once per display frame. A software update cycle is:

1. Read both mice and compute updated paddle positions
2. Update puck motion and collision physics
3. Update score and determine whether a sound event occurred
4. Wait for the next safe hardware update interval
5. Write the new positions, score, and sound event to the memory-mapped registers
6. Repeat for the next frame

Under this model, hardware always renders from a stable set of register values corresponding to one complete game-state update.

5.1.6 Synchronization Mechanism

Synchronization between software and hardware is implemented through memory-mapped polling rather than hardware interrupts. The hardware exposes a VSYNC_READY bit in the STATUS register. This bit is asserted during the vertical blanking interval and de-asserted while visible pixels are being drawn.

Software polls VSYNC_READY and only writes updated game-state registers when this bit is asserted. After completing the register writes, software waits for VSYNC_READY to return to 0 before beginning the next update cycle.

A FRAME_COUNTER register is also provided for debugging and validation. This register increments once per completed frame and allows software to confirm that the display pipeline is advancing correctly.

5.2 Register Map Overview

This table summarizes the memory-mapped register interface between the software and the FPGA air hockey peripheral. All registers are 32 bits wide, and offsets are given relative to the base address.

Offset	Register Name	Access	Purpose
0x00	STATUS	R	Hardware status bits (e.g., VSYNC_READY)
0x04	SOUND_CONTROL	W	Sound event command
0x08	P1_POS	W	Player 1 paddle center position
0x0C	P2_POS	W	Player 2 paddle center position
0x10	PUCK_POS	W	Puck center position
0x14	SCORE	W	Player 1 and Player 2 scores

5.3 Registers Definition

5.3.1 5.3.1 STATUS Register (0x00)

This register is written by hardware and read by software.

Bit(s)	Name	Meaning
[0]	VSYNC_READY	1 during vertical blanking interval, 0 otherwise
[31:1]	Reserved	Reads as 0

VSYNC_READY is the most important status bit for synchronization. Software polls this bit to determine when to write the next frame's game state. When VSYNC_READY == 1, software may safely write the next frame's game-state registers without risking mid-frame tearing. Hardware updates this register continuously as part of the display timing logic.

5.3.2 5.3.2 SOUND_CONTROL Register (0x04)

This register is used by software to request sound playback. Software decides **what happened** in the game, while hardware decides **how it sounds**.

Bit(s)	Name	Meaning
[2:0]	SOUND_EVENT	Encoded sound event ID
[31:3]	Reserved	Must be written as 0

The SOUND_EVENT field is defined as follows:

0 - no sound; 1 - puck hit wall; 2 - puck hit paddle; 3- goal scored.

When software detects one of these events, it writes the appropriate event code into SOUND_EVENT. The hardware sound generator immediately produces the corresponding sound effect. In the case of a new code being written while a sound is being played, the new sound event will take place immediately, cutting off the old sound.

5.3.3 5.3.3 PLAYER_1 Position Register (0x08)

This register stores the center coordinates of Player 1's paddle.

Bit(s)	Name	Meaning
[15:0]	P1_X	x-coordinate of Player 1 paddle center
[31:16]	P1_Y	y-coordinate of Player 1 paddle center

When software writes P1_POS, it updates the center coordinate of Player 1's paddle for the next rendered frame. The lower 16 bits are interpreted as the x-coordinate and the upper 16 bits as the y-coordinate. These coordinates are measured in screen pixels relative to the top-left corner of the visible display region, with x increasing to the right and y increasing downward.

The renderer interprets this value as the center of a solid circular paddle with a fixed radius of 20 pixels. For each pixel being drawn, the renderer compares the current pixel location against the stored center coordinate and paddle radius. If the pixel lies within the circle, the renderer outputs the paddle color for that pixel; otherwise, it continues evaluating other scene elements.

If a value larger than the maximum display width or height is passed in, the hardware will accept the value and draw the paddle without checking the boundary. So, if the paddle is being drawn off screen, there will be no visible paddle on screen or only the parts of the paddle being drawn. So the software should always check bounds before passing in values to the register.

5.3.4 5.3.4 PLAYER_2 Position Register (0x0C)

This register stores the center coordinates of Player 2's paddle.

Bit(s)	Name	Meaning
[15:0]	P2_X	x-coordinate of Player 2 paddle center
[31:16]	P2_Y	y-coordinate of Player 2 paddle center

Same as 5.3.3 Player 1 Position Register.

5.3.5 5.3.5 PUCK_POSITION Register (0x10)

This register stores the center coordinates of the puck.

Bit(s)	Name	Meaning
[15:0]	PUCK_X	x-coordinate of Puck center
[31:16]	PUCK_Y	y-coordinate of Puck center

Same as 5.3.3 Player 1 Position Register, except radius is 10 pixels.

5.3.6 5.3.6 SCORE Register (0x14)

This register stores the current score.

Bit(s)	Name	Meaning
[2:0]	SCORE_P1	Player 1 score
[5:3]	SCORE_P2	Player 2 score
[31:6]	Reserved	Must be written as 0

When software writes SCORE, it updates the numeric scores shown on the screen for Player 1 and Player 2. The lower score field is interpreted as Player 1's score and the upper score field as Player 2's score. These values are not used for game logic inside hardware; they are only used for display.

The score display logic reads this register during rendering and converts the stored values into on-screen digits in the score region of the display. For pixels that fall within the score-display area, the score logic determines whether the current pixel belongs to one of the active digit segments or digit bitmap elements, and the renderer overlays the score graphics onto the scene.

6 6. Input Path and Protocol

6.1 6.1 Software Stack

The two player controllers are standard USB HID mice connected to the board's USB host ports. The project will use the normal Linux USB stack, Linux HID driver, and Linux input subsystem rather than parsing USB packets directly in custom FPGA logic. In userspace, the game program will read the mice through the `evdev` interface exposed as `/dev/input/eventX` device files.

The software path is: **USB mouse → USB host controller → Linux HID/input drivers → /dev/input/eventX → game software**

Linux device drivers communicate to the hardware, including via USB, and generate events, while the `evdev` handler passes those hardware-independent events to userspace. It also states that `evdev` is the preferred interface for userspace to consume user input.

6.2 6.2 USB-Level Protocol

The endpoint is polled by the host controller at the interval specified by the endpoint descriptor's polling interval field. USB HID therefore uses host-driven polling for mouse input rather than device-initiated transmission.

For a HID boot mouse, the raw mouse input report format is fixed for the first three bytes. The HID 1.11 specification states that the first three bytes of the report for a boot mouse are:

- **Byte 0:** Button bits
- **Byte 1:** X displacement (relative X motion)
- **Byte 2:** Y displacement (relative Y motion)

6.3 6.3 Linux Event Conversion

The project software will not read raw USB HID bytes directly. Linux's HID and input layers will translate the USB HID reports into `evdev` events. Those events are delivered to userspace as `struct input_event` records read from `/dev/input/eventX`. The exact structure used by Linux is:

```
struct input_event {
    struct timeval time;
    unsigned short type; // event class
    unsigned short code; // specific axis / button
    int value; // event payload
};
```

For a simple mouse movement with no button change, the game software will receive event records of this form:

1. `type = EV_REL, code = REL_X, value = dx`
2. `type = EV_REL, code = REL_Y, value = dy`

3. `type = EV_SYN, code = SYN_REPORT, value = 0`

6.4 6.4 Linux Event Conversion

The design uses **two independent mice**, one for each player. Each mouse will be opened as a separate `evdev` device file, for example `/dev/input/eventA` and `/dev/input/eventB`. The software will keep these device streams separate so that each player's motion updates only one paddle.

This is one reason `evdev` is chosen here over the aggregated `/dev/input/mice` interface: `evdev` preserves per-device event streams, while `/dev/input/mice` is a shared aggregate mouse interface.

6.5 6.5 Input Rate Versus Game Update Rate

The mouse input rate and the paddle update rate are not the same quantity. At the USB layer, input reports arrive according to the mouse endpoint polling interval. At the Linux userspace layer, the program receives `evdev` event records whenever the kernel posts new translated mouse events. The exact report rate is therefore determined by the mouse device and USB polling interval, not by the game frame rate.

The game itself runs at **60 Hz**, synchronized to the display frame rate. Therefore, the software does **not** update paddle positions every time an individual mouse event arrives. Instead, it accumulates motion deltas between frames, then applies the accumulated movement once per frame during the synchronized game update.

For each mouse, the software maintains motion accumulators:

- Accumulated x delta
- Accumulated y delta

Each time an `evdev` event is received:

- If type == EV_REL and code == REL_X, the value is added to that player's x accumulator
- If type == EV_REL and code == REL_Y, the value is added to that player's y accumulator
- If type == EV_SYN and code == SYN_REPORT, that event packet is complete

Once per display frame, the game loop performs the following sequence:

1. Wait until VSYNC_READY == 1
2. Apply the accumulated deltas to Player 1 and Player 2 paddle positions
3. Clamp both paddles to their legal movement regions
4. Clear the mouse-delta accumulators
5. Write to registers
6. Wait until VSYNC_READY == 0
7. Continue collecting input for the next frame

CSEE 4840 Embedded Systems - Contributions and Reflections

Project Title: Air Hockey

Group Members:
Gerald Zhao - zz3427
Derrick Chen - dc3494
Zening Wang - zw3161
Xuepeng Han - xh2718

Spring 2026

Contents

1 Division of Labor

Member	Responsibilities
Derrick Chen	Physics engine and game logic: wall and paddle collision time solvers, 2D post-collision velocity calculation, iterative multi-collision frame loop, goal-post handling, goal detection, score tracking, and the main game loop state machine.
Gerald Zhao	Linux kernel driver and software I/O: platform driver, /dev/air_hockey misc device, six ioctl handlers, VSYNC polling loop, register write wrappers, mouse device discovery, event accumulation, axis remapping, and paddle overlap prevention.
Zening Wang	Score display and audio: seven-segment digit overlay, sound event pulse logic, ROM-based sample player for three sound effects, I2S audio serialiser, and WM8731 codec I2C initialisation.
Xuepeng Han	VGA hardware peripheral: Avalon MM register file, VSYNC_READY signal, VGA timing generator, rink geometry rendering (walls, goal openings, centre line, arcs), and procedural puck/paddle rendering using per-pixel r^2 comparators.

2 Lessons Learned

2.1 Plan the register map before writing any code

The hardware/software interface in this project is a small set of memory-mapped registers. Every field—its offset, width, bit layout, and read/write direction—needs to be written down and agreed on before either side starts implementation. In our case even a small ambiguity, such as which 16-bit half of a register holds X versus Y, would have caused silent rendering bugs that are hard to trace across the hardware/software boundary. Fixing the register map early let both sides develop independently without constant back-and-forth.

2.2 Know the FPGA resource budget before writing RTL

FPGA resources such as block RAM and DSP multipliers are fixed in quantity and must be planned ahead. Choices that look simple can have large resource costs: rendering circles procedurally eliminates the need for a frame buffer but consumes multiplier blocks, and storing audio samples in ROM eats into the same on-chip memory that score bitmaps would use. We found it useful to keep a rough budget table listing each subsystem's expected block RAM and multiplier usage and checking it against the device's total before synthesis. Finding an overrun during place-and-route, with a deadline approaching, gives very little room to fix things.

2.3 Get the hardware/software synchronization working first

The VSYNC handshake—where software waits for the blanking interval before writing registers—looks simple but is easy to get wrong. We recommend bringing up this synchronization loop first, with constant dummy values in the registers, and confirming it works before connecting any game logic. Once the sync is solid, both sides have a stable base to build on. A register write that lands during an active scanline produces a one-frame tear that only shows up at certain puck positions and is difficult to reproduce consistently.

2.4 Work through the physics edge cases on paper first

The collision math has more cases than it first appears. The quadratic solver for puck–paddle collision time t_c can produce a negative discriminant (no collision), two roots both in $[0, 1]$ (take the smaller one), or a root outside the frame (ignore it). On top of that, a fast puck can hit a wall and then a paddle within the same 16.67 ms frame, which requires looping over the remaining time after each resolved collision. We also needed to handle glancing contacts with the rounded goal-post corners. Writing out every case and testing them in a simple desktop simulation before touching the FPGA saved a significant amount of debugging time on the board, where the only observable is a pixel moving incorrectly on the VGA output.

3 Advice for Future Projects

3.1 Lock down the register interface on day one

Before writing any RTL or driver code, produce a register table with every field's offset, bit range, direction, and reset value, and treat it as a shared contract. Any change should require the whole team to agree. A one-bit mistake in field encoding breaks both the kernel driver and the synthesized hardware at the same time, and finding it during final integration is far more painful than spending an extra hour on the spec at the start.

3.2 Build a resource budget spreadsheet before synthesis

List every subsystem alongside its estimated block RAM consumption (in 18 Kb units), DSP block usage, and rough logic element count. Check the total against the device's capacity early and leave a margin of around 20%. Synthesis and place-and-route can take many minutes per run, so discovering an overrun late in the project is expensive in both time and stress.

3.3 Bring up synchronization before anything else

Implement the VSYNC polling loop with dummy constant register values first, and verify with a logic analyser or oscilloscope that no writes land during active pixel output. Only then connect the real game state. This gives both the hardware and software sides a known-good integration point to build from.

3.4 Simulate the physics on a laptop before running on the board

Write a small desktop program that feeds the physics engine a sweep of starting conditions and prints the results. Cover the edge cases: puck near each wall, puck already touching a paddle, puck fast enough to cross the arena in one frame, and puck heading into a goal-post corner. Fixing these cases in a terminal is far faster than diagnosing them from VGA output on the FPGA board.

Air Hockey Project Code Appendix

Contents

1 Software: Application

1.1 Makefile

```
#
    ================================================================================

# Root Makefile for project-hockey-sw
#
# Purpose:
#   Build userspace app/test programs and optionally the
    kernel module.
#
# Design choice:
#   We do NOT build the kernel module by default anymore.
#   This avoids rebuilding driver/air_hockey.ko when only
    userspace files
#   changed.
#
# Main targets:
#   make                   -> build common userspace programs
    only
#   make module            -> build kernel module only
#   make tests             -> build all tests
#   make app/main          -> build main app only
#   make tests/test_mouse -> build one specific test only
#   make clean             -> clean module + userspace outputs
#
    ================================================================================


KERNEL_SOURCE := /usr/src/linux-headers-$(shell uname -r)

CC := cc
CFLAGS := -I driver -I app -Wall -Wextra -g

#
    --------------------------------------------------------------------------------

# Userspace programs
#
    --------------------------------------------------------------------------------


```

```
30 TEST_BINS := tests/test_positions tests/test_mouse tests/
      test_starting_pos tests/test_score_display tests/
      test_sound
31 APP_BINS := app/main
32
33 .PHONY: default all module tests apps clean help
34
35 # Default target:
36 # Build userspace only, not the kernel module.
37 default: all
38
39 all: tests apps
40
41 #
      ---------------------------------------------------------------------
42 # Kernel module target
43 #
      ---------------------------------------------------------------------
44
45 module:
46         $(MAKE) -C driver KERNEL_SOURCE=$(KERNEL_SOURCE)
47
48 #
      ---------------------------------------------------------------------
49 # Build all userspace tests
50 #
      ---------------------------------------------------------------------
51
52 tests: $(TEST_BINS)
53
54 #
      ---------------------------------------------------------------------
55 # Build all userspace apps
56 #
      ---------------------------------------------------------------------
57
58 apps: $(APP_BINS)
59
```

```
60 #
       ------------------------------------------------------------------------
61 # Individual userspace build rules
62 #
63 # Any userspace file that uses game_io_* must link with app/
       game_io.c.
64 #
       ------------------------------------------------------------------------
65
66 tests/test_positions: tests/test_positions.c app/game_io.c
       app/game_io.h driver/air_hockey.h
67         $(CC) $(CFLAGS) tests/test_positions.c app/game_io.c
               -o tests/test_positions
68
69 tests/test_mouse: tests/test_mouse.c app/game_io.c app/
       game_io.h driver/air_hockey.h
70         $(CC) $(CFLAGS) tests/test_mouse.c app/game_io.c -o
               tests/test_mouse
71
72 tests/test_starting_pos: tests/test_starting_pos.c app/
       game_io.c app/game_io.h driver/air_hockey.h
73         $(CC) $(CFLAGS) tests/test_starting_pos.c app/game_io
               .c -o tests/test_starting_pos
74
75 tests/test_score_display: tests/test_score_display.c app/
       game_io.c app/game_io.h driver/air_hockey.h
76         $(CC) $(CFLAGS) tests/test_score_display.c app/
               game_io.c -o tests/test_score_display
77
78 tests/test_sound: tests/test_sound.c app/game_io.c app/
       game_io.h driver/air_hockey.h
79         $(CC) $(CFLAGS) tests/test_sound.c app/game_io.c -o
               tests/test_sound
80
81
82 app/main: app/main.c \
83           app/game_io.c \
84           app/physics_engine.c \
85           app/game_sim.c \
86           app/game_input.c \
87           app/game_render.c \
88           app/game_score.c \
89           app/game_io.h \
```

```
90            app/physics_engine.h \
91            app/game_config.h \
92            app/game_sim.h \
93            app/game_input.h \
94            app/game_render.h \
95            app/game_score.h \
96            driver/air_hockey.h
97          $(CC) $(CFLAGS) app/main.c app/game_io.c app/
                physics_engine.c app/game_sim.c app/game_input.c
                app/game_render.c app/game_score.c -o app/main -lm
98
99  #
        ------------------------------------------------------------------------

100 # Clean everything
101 #
        ------------------------------------------------------------------------

102
103 clean:
104         $(MAKE) -C driver KERNEL_SOURCE=$(KERNEL_SOURCE)
                clean
105         rm -f $(TEST_BINS) $(APP_BINS)
106
107 help:
108         @echo "Targets:"
109         @echo "  make                  - build userspace
                programs only"
110         @echo "  make module           - build kernel module
                only"
111         @echo "  make tests            - build all tests"
112         @echo "  make apps             - build all apps"
113         @echo "  make app/main         - build main app only"
114         @echo "  make tests/test_mouse - build only
                test_mouse"
115         @echo "  make tests/test_starting_pos - build only
                test_starting_pos"
116         @echo "  make help             - show this help
                message"
117         @echo "  make clean            - remove generated
                files"
```

1.2 game_config.h

```
#ifndef GAME_CONFIG_H
#define GAME_CONFIG_H

// Mouse orientation mapping
// 0 = normal monitor orientation
// 1 = monitor laid flat / mouse axes swapped
#define MOUSE_SWAP_XY 1

// Per-player direction signs after optional X/Y swap.
// Change these between 1 and -1 until each paddle feels correct.
#define P1_MOUSE_X_SIGN  -1
#define P1_MOUSE_Y_SIGN  1

#define P2_MOUSE_X_SIGN  1
#define P2_MOUSE_Y_SIGN  -1

// Screen geometry
#define SCREEN_WIDTH    640.0
#define SCREEN_HEIGHT   480.0

// Outer rink wall geometry from hardware Verilog
#define WALL_LEFT       10.0
#define WALL_RIGHT      629.0
#define WALL_TOP        10.0
#define WALL_BOTTOM     469.0
#define WALL_WIDTH      4.0

// Inner playable ice area from hardware:
// Verilog uses:
// px >= WL + WW && px <= WR - WW
// py >= WT + WW && py <= WB - WW
#define PLAY_LEFT       (WALL_LEFT + WALL_WIDTH)
#define PLAY_RIGHT      (WALL_RIGHT - WALL_WIDTH)
#define PLAY_TOP        (WALL_TOP + WALL_WIDTH)
#define PLAY_BOTTOM     (WALL_BOTTOM - WALL_WIDTH)

// Goal opening from hardware Verilog
#define GOAL_TOP        190.0
#define GOAL_BOTTOM     290.0

// Object radii from hardware Verilog
// PADDLE_R2 = 900  -> radius 30
```

```
43 // PUCK_R2   = 400  -> radius 20
44 #define PADDLE_RADIUS   30.0
45 #define PUCK_RADIUS     20.0
46 #define POST_RADIUS     10.0
47
48 // Starting positions
49 #define P1_START_X      100.0
50 #define P1_START_Y      240.0
51
52 #define P2_START_X      540.0
53 #define P2_START_Y      240.0
54
55 #define PUCK_START_X    320.0
56 #define PUCK_START_Y    240.0
57
58 #define PUCK_START_X_P1 440.0
59 #define PUCK_START_X_P2 200.0
60
61 // Paddle movement bounds
62 // These keep the whole paddle inside the visible ice, not
       inside the border.
63 #define P1_X_MIN        (PLAY_LEFT + PADDLE_RADIUS)
64 #define P1_X_MAX        (320.0 - PADDLE_RADIUS)
65
66 #define P2_X_MIN        (320.0 + PADDLE_RADIUS)
67 #define P2_X_MAX        (PLAY_RIGHT - PADDLE_RADIUS)
68
69 #define PADDLE_Y_MIN    (PLAY_TOP + PADDLE_RADIUS)
70 #define PADDLE_Y_MAX    (PLAY_BOTTOM - PADDLE_RADIUS)
71
72 // Puck movement bounds
73 // These keep the whole puck inside the visible ice.
74 #define PUCK_X_MIN      (PLAY_LEFT + PUCK_RADIUS)
75 #define PUCK_X_MAX      (PLAY_RIGHT - PUCK_RADIUS)
76 #define PUCK_Y_MIN      (PLAY_TOP + PUCK_RADIUS)
77 #define PUCK_Y_MAX      (PLAY_BOTTOM - PUCK_RADIUS)
78
79 // Gameplay tuning
80 #define MOUSE_SENSITIVITY 0.8
81 #define MAX_PADDLE_SPEED  20.0
82 #define MAX_PUCK_SPEED    40.0
83
84 #define RESTITUTION       0.6
85 #define WALL_RESTITUTION  0.7
86 #define PADDLE_RESTITUTION 0.3
```

```
87 #define POST_RESTITUTION  0.7
88
89 // Goal detection threshold.
90 // Score only after puck has visibly entered the goal/off-
       screen area.
91 // Keep this based on outer wall, not PLAY_LEFT/PLAY_RIGHT.
92 #define LEFT_GOAL_SCORE_X   (WALL_LEFT)    //
93 #define RIGHT_GOAL_SCORE_X  (WALL_RIGHT)   //
94
95 // Simulation tuning
96 #define MAX_BOUNCES 3
97 #define MIN_FRAME_TIME_REMAINING 0.001
98 #define MIN_COLLISION_TIME 1e-6
99
100 // Score limits
101 #define MAX_SCORE 7
102
103 #endif
```

1.3 game_input.c

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <linux/input.h>

#include "game_input.h"
#include "game_config.h"

static double clamp_double(double val, double lo, double hi)
{
    if (val < lo) return lo;
    if (val > hi) return hi;
    return val;
}

int open_mouse_device(const char *path)
{
    int fd = open(path, O_RDONLY | O_NONBLOCK);

    if (fd < 0) {
        perror(path);
        return -1;
    }

    return fd;
}

#define INPUT_BY_PATH_DIR "/dev/input/by-path"
#define MAX_MOUSE_DEVICES 8

static int compare_mouse_paths(const void *a, const void *b)
{
    const char *pa = *(const char * const *)a;
    const char *pb = *(const char * const *)b;
    return strcmp(pa, pb);
}

int open_two_mouse_devices(int *mouse_fd1, int *mouse_fd2)
{
    DIR *dir;
```

```
    struct dirent *entry;
    char *mouse_paths[MAX_MOUSE_DEVICES];
    int mouse_count = 0;

    *mouse_fd1 = -1;
    *mouse_fd2 = -1;

    dir = opendir(INPUT_BY_PATH_DIR);
    if (dir == NULL) {
        perror("opendir /dev/input/by-path");
        return -1;
    }

    while ((entry = readdir(dir)) != NULL) {
        if (strstr(entry->d_name, "event-mouse") != NULL) {
            char full_path[512];

            if (mouse_count >= MAX_MOUSE_DEVICES) {
                break;
            }

            snprintf(full_path, sizeof(full_path), "%s/%s",
                     INPUT_BY_PATH_DIR, entry->d_name);

            mouse_paths[mouse_count] = strdup(full_path);
            if (mouse_paths[mouse_count] == NULL) {
                perror("strdup");
                closedir(dir);

                for (int i = 0; i < mouse_count; i++) {
                    free(mouse_paths[i]);
                }

                return -1;
            }

            mouse_count++;
        }
    }

    closedir(dir);

    if (mouse_count < 2) {
        fprintf(stderr, "Found only %d mouse device(s). Need
           2.\n", mouse_count);
```

```
88
89          for (int i = 0; i < mouse_count; i++) {
90              fprintf(stderr, "Found mouse: %s\n", mouse_paths[
                    i]);
91              free(mouse_paths[i]);
92          }
93
94          return -1;
95      }
96
97      qsort(mouse_paths, mouse_count, sizeof(char *),
            compare_mouse_paths);
98
99      fprintf(stderr, "Using mouse 1: %s\n", mouse_paths[0]);
100     fprintf(stderr, "Using mouse 2: %s\n", mouse_paths[1]);
101
102     *mouse_fd1 = open_mouse_device(mouse_paths[0]);
103     *mouse_fd2 = open_mouse_device(mouse_paths[1]);
104
105     for (int i = 0; i < mouse_count; i++) {
106         free(mouse_paths[i]);
107     }
108
109     if (*mouse_fd1 < 0 || *mouse_fd2 < 0) {
110         if (*mouse_fd1 >= 0) close(*mouse_fd1);
111         if (*mouse_fd2 >= 0) close(*mouse_fd2);
112
113         *mouse_fd1 = -1;
114         *mouse_fd2 = -1;
115
116         return -1;
117     }
118
119     return 0;
120 }
121
122 void poll_mouse_and_update_paddle(int mouse_fd,
123                                   GameObject *p,
124                                   const GameObject *puck,
125                                   double x_min,
126                                   double x_max,
127                                   double y_min,
128                                   double y_max,
129                                   int x_sign,
130                                   int y_sign)
```

```
{
    struct input_event ev;
    int dx = 0;
    int dy = 0;

    // Collect all mouse movement since last frame
    while (read(mouse_fd, &ev, sizeof(ev)) == sizeof(ev)) {
        if (ev.type == EV_REL) {
            if (ev.code == REL_X) {
                dx += ev.value;
            } else if (ev.code == REL_Y) {
                dy += ev.value;
            }
        }
    }

    int mapped_dx = dx;
    int mapped_dy = dy;

    #if MOUSE_SWAP_XY
    mapped_dx = dy;
    mapped_dy = dx;
    #endif

    mapped_dx *= x_sign;
    mapped_dy *= y_sign;

    // convert mouse movement to paddle movement, with
       sensitivity and max speed limits
    double move_x = mapped_dx * MOUSE_SENSITIVITY;
    double move_y = mapped_dy * MOUSE_SENSITIVITY;

    move_x = clamp_double(move_x, -MAX_PADDLE_SPEED,
       MAX_PADDLE_SPEED);
    move_y = clamp_double(move_y, -MAX_PADDLE_SPEED,
       MAX_PADDLE_SPEED);

    // If there is no movement, do not update velocity to
       allow puck to push stationary paddle
    if (dx == 0 && dy == 0) {
        // No movement, do not update velocity
        p->vel.x = 0.0;
        p->vel.y = 0.0;
        return;
    }
```

```
172
173     // Save current paddle position before moving
174     double old_x = p->pos.x;
175     double old_y = p->pos.y;
176
177     // Compute where paddle would move to if we apply the
            full mouse movement, and clamp to arena bounds
178     double proposed_x = clamp_double(old_x + move_x, x_min,
            x_max);
179     double proposed_y = clamp_double(old_y + move_y, y_min,
            y_max);
180
181     // Compute distance from current paddle position to puck
            position
182     double old_dx = old_x - puck->pos.x;
183     double old_dy = old_y - puck->pos.y;
184     double old_dist2 = old_dx * old_dx + old_dy * old_dy;
185
186     // Compute distance from proposed paddle position to puck
187     double proposed_dx = proposed_x - puck->pos.x;
188     double proposed_dy = proposed_y - puck->pos.y;
189     double proposed_dist2 = proposed_dx * proposed_dx +
            proposed_dy * proposed_dy;
190
191     // Minimum legal center to center distance between paddle
             and puck to avoid overlap
192     double min_dist = p->radius + puck->radius;
193     double min_dist2 = min_dist * min_dist;
194
195     if (proposed_dist2 < min_dist2) {
196         int moving_deeper_into_puck = proposed_dist2 <=
                old_dist2;
197
198         if(moving_deeper_into_puck) {
199             p->vel.x = 0.0;
200             p->vel.y = 0.0;
201             return;
202         }
203     }
204
205     // Movement is legal, commit the proposed position and
            velocity
206     p->pos.x = proposed_x;
207     p->pos.y = proposed_y;
208
```

```
209     p->vel.x = proposed_x - old_x;
210     p->vel.y = proposed_y - old_y;
211 }
```

1.4 game_input.h

```
#ifndef GAME_INPUT_H
#define GAME_INPUT_H

#include "physics_engine.h"

int open_mouse_device(const char *path);

int open_two_mouse_devices(int *mouse_fd1, int *mouse_fd2);

void poll_mouse_and_update_paddle(int mouse_fd,
                                  GameObject *p,
                                  const GameObject *puck,
                                  double x_min,
                                  double x_max,
                                  double y_min,
                                  double y_max,
                                  int x_sign,
                                  int y_sign);

#endif
```

1.5 game_io.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>

#include "game_io.h"

/*
 * File descriptor for /dev/air_hockey.
 *
 * Hidden inside this file so the rest of the app does not need to care
 * about device file management.
 */
static int g_air_hockey_fd = -1;

/*
 * Internal helper:
 * send one position ioctl using the shared air_hockey_pos_arg_t wrapper.
 */
static void write_pos(unsigned long cmd, unsigned short x, unsigned short y)
{
    air_hockey_pos_arg_t arg;

    arg.pos.x = x;
    arg.pos.y = y;

    if (ioctl(g_air_hockey_fd, cmd, &arg)) {
        perror("game_io: position ioctl failed");
        exit(1);
    }
}

/*
 * Internal helper:
 * send one score ioctl.
 */
static void write_score(unsigned char p1, unsigned char p2)
{
    air_hockey_score_arg_t arg;
```

```
41
42      arg.score.p1 = p1;
43      arg.score.p2 = p2;
44
45      if (ioctl(g_air_hockey_fd, AIR_HOCKEY_WRITE_SCORE, &arg))
            {
46          perror("game_io: score ioctl failed");
47          exit(1);
48      }
49  }
50
51  /*
52   * Internal helper:
53   * send one sound ioctl.
54   */
55  static void write_sound(unsigned char event_id)
56  {
57      air_hockey_sound_arg_t arg;
58
59      arg.sound_event = event_id;
60
61      if (ioctl(g_air_hockey_fd, AIR_HOCKEY_WRITE_SOUND, &arg))
            {
62          perror("game_io: sound ioctl failed");
63          exit(1);
64      }
65  }
66
67  int game_io_open(const char *device_path)
68  {
69      if (device_path == NULL)
70          device_path = "/dev/air_hockey";
71
72      g_air_hockey_fd = open(device_path, O_RDWR);
73      if (g_air_hockey_fd == -1) {
74          perror("game_io: open(/dev/air_hockey) failed");
75          return -1;
76      }
77
78      return 0;
79  }
80
81  void game_io_close(void)
82  {
83      if (g_air_hockey_fd != -1) {
```

```
84          close(g_air_hockey_fd);
85          g_air_hockey_fd = -1;
86      }
87  }
88
89  unsigned int game_io_read_status(void)
90  {
91      air_hockey_status_t st;
92
93      if (g_air_hockey_fd == -1) {
94          fprintf(stderr, "game_io: device not open\n");
95          exit(1);
96      }
97
98      if (ioctl(g_air_hockey_fd, AIR_HOCKEY_READ_STATUS, &st))
            {
99          perror("game_io: AIR_HOCKEY_READ_STATUS failed");
100         exit(1);
101     }
102
103     return st.status;
104 }
105
106 void game_io_wait_for_vsync(void)
107 {
108     //If already in VBlank, wait for it to end first (
            prevents accidental double-sync if this is called
            multiple times per frame)
109     while ((game_io_read_status() &
            AIR_HOCKEY_STATUS_VSYNC_READY) != 0) {
110         usleep(50);
111     }
112     //wait for the next VBlank to begin.
113     while ((game_io_read_status() &
            AIR_HOCKEY_STATUS_VSYNC_READY) == 0) {
114         usleep(50);
115     }
116 }
117
118 void game_io_set_p1(unsigned short x, unsigned short y)
119 {
120     write_pos(AIR_HOCKEY_WRITE_P1_POS, x, y);
121 }
122
123 void game_io_set_p2(unsigned short x, unsigned short y)
```

```
124 {
125     write_pos(AIR_HOCKEY_WRITE_P2_POS, x, y);
126 }
127
128 void game_io_set_puck(unsigned short x, unsigned short y)
129 {
130     write_pos(AIR_HOCKEY_WRITE_PUCK_POS, x, y);
131 }
132
133 void game_io_set_score(unsigned char p1, unsigned char p2)
134 {
135     write_score(p1, p2);
136 }
137
138 void game_io_set_sound(unsigned char event_id)
139 {
140     write_sound(event_id);
141 }
142
143 void game_io_submit_frame(const game_io_frame_t *frame)
144 {
145     if (frame == NULL)
146         return;
147
148     /*
149      * The intent is that main.c prepares one frame of
            logical state, then
150      * calls this once during VSYNC.
151      *
152      * Order mostly does not matter much for the current
            hardware, but grouping
153      * all writes here keeps the main loop clean and makes
            the per-frame update
154      * policy explicit.
155      */
156     game_io_set_p1(frame->p1_x, frame->p1_y);
157     game_io_set_p2(frame->p2_x, frame->p2_y);
158     game_io_set_puck(frame->puck_x, frame->puck_y);
159     game_io_set_score(frame->score_p1, frame->score_p2);
160     game_io_set_sound(frame->sound_event);
161 }
```

1.6 game_io.h

```c
#ifndef GAME_IO_H
#define GAME_IO_H

/*
 * app/game_io.h
 *
 * Userspace wrapper around the kernel driver /dev/air_hockey.
 *
 * This is NOT the kernel driver.
 * This is a small helper layer used by app/main.c and tests so they do not
 * have to call ioctl() directly everywhere.
 *
 * Responsibilities:
 *   - open / close the device
 *   - read status register
 *   - wait for VSYNC
 *   - send logical puck / paddle / score / sound updates
 */

#include "../driver/air_hockey.h"

/* Open /dev/air_hockey (or another device path if needed). Returns 0 on success. */
int game_io_open(const char *device_path);

/* Close the device if open. Safe to call more than once. */
void game_io_close(void);

/* Read raw STATUS register value through the kernel driver. */
unsigned int game_io_read_status(void);

/* Busy-wait until VSYNC_READY is asserted. */
void game_io_wait_for_vsync(void);

/* Write one logical paddle/puck position. */
void game_io_set_p1(unsigned short x, unsigned short y);
void game_io_set_p2(unsigned short x, unsigned short y);
void game_io_set_puck(unsigned short x, unsigned short y);

/* Write score register. */
```

```
40 void game_io_set_score(unsigned char p1, unsigned char p2);
41
42 /* Write one sound event code. */
43 void game_io_set_sound(unsigned char event_id);
44
45 /*
46  * Push all visible game objects for one frame.
47  *
48  * This is just a convenience helper so main.c can update all
        visible state
49  * together in one place.
50  */
51 typedef struct {
52     unsigned short p1_x, p1_y;
53     unsigned short p2_x, p2_y;
54     unsigned short puck_x, puck_y;
55     unsigned char score_p1, score_p2;
56     unsigned char sound_event;
57 } game_io_frame_t;
58
59 void game_io_submit_frame(const game_io_frame_t *frame);
60
61 #endif
```

1.7 game_render.c

```
#include "game_render.h"
#include "game_io.h"

void write_to_vga_registers(GameObject *puck,
                            GameObject *p1,
                            GameObject *p2,
                            int p1_score,
                            int p2_score,
                            unsigned char sound_event)
{
    game_io_set_p1((unsigned short)p1->pos.x, (unsigned short
       )p1->pos.y);
    game_io_set_p2((unsigned short)p2->pos.x, (unsigned short
       )p2->pos.y);
    game_io_set_puck((unsigned short)puck->pos.x, (unsigned
       short)puck->pos.y);
    game_io_set_score((unsigned char)p1_score, (unsigned char
       )p2_score);
    game_io_set_sound(sound_event);
}
```

1.8 game_render.h

```
#ifndef GAME_RENDER_H
#define GAME_RENDER_H

#include "physics_engine.h"

void write_to_vga_registers(GameObject *puck,
                            GameObject *p1,
                            GameObject *p2,
                            int p1_score,
                            int p2_score,
                            unsigned char sound_event);

#endif
```

1.9 game_score.c

```c
#include <stdio.h>

#include "game_score.h"
#include "game_config.h"

GoalResult check_goal(const GameObject *puck)
{
    int in_goal_y =
        puck->pos.y >= GOAL_TOP &&
        puck->pos.y <= GOAL_BOTTOM;

    if (!in_goal_y) {
        return GOAL_NONE;
    }

    /*
     * Do NOT score when puck merely touches the inner wall boundary.
     * Score only after the puck has traveled into/behind the goal.
     */
    if (puck->pos.x <= LEFT_GOAL_SCORE_X) {
        return GOAL_P2_SCORED;
    }

    if (puck->pos.x >= RIGHT_GOAL_SCORE_X) {
        return GOAL_P1_SCORED;
    }

    return GOAL_NONE;
}

void reset_after_goal(GameObject *puck,
                      GameObject *p1,
                      GameObject *p2,
                      double x_puck_start)
{
    puck->pos.x = x_puck_start;
    puck->pos.y = PUCK_START_Y;
    puck->vel.x = 0.0;
    puck->vel.y = 0.0;
    puck->radius = PUCK_RADIUS;

```

```
    p1->pos.x = P1_START_X;
    p1->pos.y = P1_START_Y;
    p1->vel.x = 0.0;
    p1->vel.y = 0.0;
    p1->radius = PADDLE_RADIUS;

    p2->pos.x = P2_START_X;
    p2->pos.y = P2_START_Y;
    p2->vel.x = 0.0;
    p2->vel.y = 0.0;
    p2->radius = PADDLE_RADIUS;
}

// Returns 1 if P1 scored, returns 2 if P2 scored. 0 of no
   score
int handle_score_update(GameObject *puck,
                        GameObject *p1,
                        GameObject *p2,
                        int *p1_score,
                        int *p2_score)
{
    GoalResult goal = check_goal(puck);

    if (goal == GOAL_P1_SCORED) {
        if (*p1_score < MAX_SCORE) {
            (*p1_score)++;
        }

        fprintf(stderr, "[GOAL] P1 SCORED! New score: P1=%d, P2=%d\n", *p1_score, *p2_score);
        return 1;
    } else if (goal == GOAL_P2_SCORED) {
        if (*p2_score < MAX_SCORE) {
            (*p2_score)++;
        }

        fprintf(stderr, "[GOAL] P2 SCORED! New score: P1=%d, P2=%d\n", *p1_score, *p2_score);
        return 2;
    }

    return goal != GOAL_NONE;
}
```

1.10 game_score.h

```
#ifndef GAME_SCORE_H
#define GAME_SCORE_H

#include "physics_engine.h"

typedef enum {
    GOAL_NONE = 0,
    GOAL_P1_SCORED,
    GOAL_P2_SCORED
} GoalResult;

GoalResult check_goal(const GameObject *puck);

void reset_after_goal(GameObject *puck,
                      GameObject *p1,
                      GameObject *p2,
                      double x_puck_start);

int handle_score_update(GameObject *puck,
                        GameObject *p1,
                        GameObject *p2,
                        int *p1_score,
                        int *p2_score);

#endif
```

1.11 game_sim.c

```
#include <stdio.h>
#include <math.h>
#include <float.h>

#include "game_sim.h"
#include "physics_engine.h"
#include "game_config.h"
#include "../driver/air_hockey.h"

extern int debug_physics;

typedef enum {
    COLLISION_NONE = 0,
    COLLISION_WALL,
    COLLISION_P1,
    COLLISION_P2,
    COLLISION_LEFT_TOP_POST,
    COLLISION_LEFT_BOTTOM_POST,
    COLLISION_RIGHT_TOP_POST,
    COLLISION_RIGHT_BOTTOM_POST
} CollisionType;

static double min_time(double a, double b)
{
    return (a < b) ? a : b;
}

static void print_time(const char *name, double t)
{
    if (t == DBL_MAX || isinf(t)) {
        printf("%s=INF␣", name);
    } else {
        printf("%s=%.6f␣", name, t);
    }
}

/*
 * Center of puck is inside the visual goal opening.
 * This is used to decide whether the vertical goal wall is
    open.
 */
static int puck_center_inside_goal_opening(double y)
{
```

```
43      return y >= GOAL_TOP && y <= GOAL_BOTTOM;
44  }
45
46  static void clamp_puck_to_arena(GameObject *puck)
47  {
48      int in_goal_y = puck_center_inside_goal_opening(puck->pos
            .y);
49
50      /*
51       * If puck is inside the goal opening and moving outward,
52       * allow it to pass beyond the left/right playable edge.
53       */
54      int exiting_left_goal =
55          in_goal_y &&
56          puck->vel.x < 0.0 &&
57          puck->pos.x <= PLAY_LEFT + puck->radius;
58
59      int exiting_right_goal =
60          in_goal_y &&
61          puck->vel.x > 0.0 &&
62          puck->pos.x >= PLAY_RIGHT - puck->radius;
63
64      if (!exiting_left_goal && !exiting_right_goal) {
65          if (puck->pos.x < PLAY_LEFT + puck->radius) {
66              puck->pos.x = PLAY_LEFT + puck->radius;
67          } else if (puck->pos.x > PLAY_RIGHT - puck->radius) {
68              puck->pos.x = PLAY_RIGHT - puck->radius;
69          }
70      }
71
72      if (puck->pos.y < PLAY_TOP + puck->radius) {
73          puck->pos.y = PLAY_TOP + puck->radius;
74      } else if (puck->pos.y > PLAY_BOTTOM - puck->radius) {
75          puck->pos.y = PLAY_BOTTOM - puck->radius;
76      }
77  }
78
79  static void clamp_object_speed(GameObject *obj, double
        max_speed)
80  {
81      double speed = sqrt(obj->vel.x * obj->vel.x + obj->vel.y
            * obj->vel.y);
82
83      if (speed > max_speed) {
84          double scale = max_speed / speed;
```

```
85          obj->vel.x *= scale;
86          obj->vel.y *= scale;
87      }
88  }
89
90
91  unsigned char simulate_frame(GameObject *puck,
92                       GameObject *p1,
93                       GameObject *p2)
94  {
95      double t_remaining = 1.0;
96      int bounce_count = 0;
97      unsigned char sound_event = AIR_HOCKEY_SOUND_NONE;
98
99      while (t_remaining > MIN_FRAME_TIME_REMAINING &&
100            bounce_count < MAX_BOUNCES) {
101
102         double t_wall = get_wall_collision_time(puck);
103
104         double t_p1 = getPaddleCollisionTime(puck, p1);
105         double t_p2 = getPaddleCollisionTime(puck, p2);
106
107
108         // get goal post time
109         GameObject left_top_post = {{PLAY_LEFT, GOAL_TOP},
               {0.0, 0.0}, 0.1};
110         GameObject left_bottom_post = {{PLAY_LEFT,
               GOAL_BOTTOM}, {0.0, 0.0}, 0.1};
111         GameObject right_top_post = {{PLAY_RIGHT, GOAL_TOP},
               {0.0, 0.0}, 0.1};
112         GameObject right_bottom_post = {{PLAY_RIGHT,
               GOAL_BOTTOM}, {0.0, 0.0}, 0.1};
113
114         double t_post1, t_post2;
115
116         if (puck->pos.x < (PLAY_LEFT + PLAY_RIGHT) / 2) {
117             // near left goal, check collision with left goal
                    posts
118             t_post1 = getPaddleCollisionTime(puck, &
                   left_top_post);
119             t_post2 = getPaddleCollisionTime(puck, &
                   left_bottom_post);
120         } else {
121             // near right goal, check collision with right
                   goal posts
```

```
122             t_post1 = getPaddleCollisionTime(puck, &
                    right_top_post);
123             t_post2 = getPaddleCollisionTime(puck, &
                    right_bottom_post);
124         }
125
126         double t_c = min_time(min_time(t_wall, min_time(t_p1,
                t_p2)), min_time(t_post1, t_post2));
127
128         CollisionType collision_type = COLLISION_NONE;
129
130         if (t_c == t_wall) {
131             collision_type = COLLISION_WALL;
132         } else if (t_c == t_p1) {
133             collision_type = COLLISION_P1;
134         } else if (t_c == t_p2) {
135             collision_type = COLLISION_P2;
136         } else if (t_c == t_post1) {
137             if (puck->pos.x < (PLAY_LEFT + PLAY_RIGHT) / 2) {
138                 collision_type = COLLISION_LEFT_TOP_POST;
139             } else {
140                 collision_type = COLLISION_RIGHT_TOP_POST;
141             }
142         } else if (t_c == t_post2) {
143             if (puck->pos.x < (PLAY_LEFT + PLAY_RIGHT) / 2) {
144                 collision_type = COLLISION_LEFT_BOTTOM_POST;
145             } else {
146                 collision_type = COLLISION_RIGHT_BOTTOM_POST;
147             }
148         } else {
149             collision_type = COLLISION_NONE;
150         }
151
152         if (t_c < MIN_COLLISION_TIME) {
153             t_c = MIN_COLLISION_TIME;
154         }
155
156         if (t_c >= t_remaining || t_c == DBL_MAX || isinf(t_c
                )) {
157             puck->pos.x += puck->vel.x * t_remaining;
158             puck->pos.y += puck->vel.y * t_remaining;
159
160             t_remaining = 0.0;
161         } else {
162             puck->pos.x += puck->vel.x * t_c;
```

```
163             puck->pos.y += puck->vel.y * t_c;
164
165             if (collision_type == COLLISION_P1) {
166                 sound_event = AIR_HOCKEY_SOUND_HIT_PADDLE;
167                 applyPaddleCollision(puck, p1,
                        PADDLE_RESTITUTION);
168             } else if (collision_type == COLLISION_P2) {
169                 sound_event = AIR_HOCKEY_SOUND_HIT_PADDLE;
170                 applyPaddleCollision(puck, p2,
                        PADDLE_RESTITUTION);
171             } else if (collision_type == COLLISION_WALL) {
172                 sound_event = AIR_HOCKEY_SOUND_HIT_WALL;
173                 applyWallBounce(puck, WALL_RESTITUTION);
174             }  else if (collision_type ==
                    COLLISION_LEFT_BOTTOM_POST) {
175                 sound_event = AIR_HOCKEY_SOUND_HIT_WALL;
176                 applyPaddleCollision(puck, &left_bottom_post,
                        WALL_RESTITUTION);
177             } else if (collision_type ==
                    COLLISION_LEFT_TOP_POST) {
178                 sound_event = AIR_HOCKEY_SOUND_HIT_WALL;
179                 applyPaddleCollision(puck, &left_top_post,
                        WALL_RESTITUTION);
180             } else if (collision_type ==
                    COLLISION_RIGHT_BOTTOM_POST) {
181                 sound_event = AIR_HOCKEY_SOUND_HIT_WALL;
182                 applyPaddleCollision(puck, &right_bottom_post
                        , WALL_RESTITUTION);
183             } else if (collision_type ==
                    COLLISION_RIGHT_TOP_POST) {
184                 sound_event = AIR_HOCKEY_SOUND_HIT_WALL;
185                 applyPaddleCollision(puck, &right_top_post,
                        WALL_RESTITUTION);
186             }
187
188             clamp_object_speed(puck, MAX_PUCK_SPEED);
189
190             t_remaining -= t_c;
191             bounce_count++;
192         }
193     }
194
195     if (bounce_count >= MAX_BOUNCES && debug_physics) {
196         fprintf(stderr,
197                 "simulate_frame:␣MAX_BOUNCES␣reached,␣"
```

```
198                 "puck␣pos=(%.2f,␣%.2f)␣vel=(%.2f,␣%.2f)\n",
199                 puck->pos.x, puck->pos.y,
200                 puck->vel.x, puck->vel.y);
201     }
202     return sound_event;
203 }
```

1.12 game_sim.h

```
#ifndef GAME_SIM_H
#define GAME_SIM_H

#include "physics_engine.h"

unsigned char simulate_frame(GameObject *puck,
                     GameObject *p1,
                     GameObject *p2);

#endif
```

1.13 main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <errno.h>
#include <math.h>
#include <float.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <fcntl.h>


#include "physics_engine.h"
#include "game_io.h"
#include "game_config.h"
#include "game_render.h"
#include "game_sim.h"
#include "game_input.h"
#include "game_score.h"

int debug_physics = 0; // Set to 1 to enable detailed physics debug prints


int main(int argc, char *argv[]) {

    // 1. Initialize Game Objects
    GameObject p1 = {{100.0, 240.0}, {0.0, 0.0}, PADDLE_RADIUS};
    GameObject p2 = {{540.0, 240.0}, {0.0, 0.0}, PADDLE_RADIUS};
    GameObject puck = {{320.0, 240.0}, {0.0, 0.0}, PUCK_RADIUS};

    // if command line argument puck is provided, set initial puck velocity
    // command line argument format: puckx pucky puckvelx puckvely
    if (argc == 5) {
        debug_physics = 1;
        puck.pos.x = atof(argv[1]);
        puck.pos.y = atof(argv[2]);
        puck.vel.x = atof(argv[3]);
```

```
38          puck.vel.y = atof(argv[4]);
39          printf("[main] initial puck state from command line:
                pos=(%.2f, %.2f) vel=(%.2f, %.2f)\n",
40                  puck.pos.x, puck.pos.y, puck.vel.x, puck.vel.y
                        );
41      }
42
43
44      int p1_score = 0;
45      int p2_score = 0;
46      int game_state = 0; // 0 = playing, 1 = goal scored/
            waiting for serve
47
48      printf("Starting Embedded Air Hockey Engine...\n");
49      if (game_io_open("/dev/air_hockey") != 0) {
50          fprintf(stderr, "Failed to open /dev/air_hockey\n");
51          return 1;
52      }
53
54      // Scan /dev/input/by-path for event-mouse devices and
            open the first two
55      int mouse_fd1, mouse_fd2;
56      if (open_two_mouse_devices(&mouse_fd1, &mouse_fd2) != 0)
            {
57          fprintf(stderr, "Failed to open two mouse devices\n")
                ;
58          game_io_close();
59          return 1;
60      }
61
62      unsigned char sound_event = AIR_HOCKEY_SOUND_NONE;
63
64      // 2. Main Game Loop
65      while (game_state != 2) {
66
67          // Wait for VGA to finish drawing the current frame
                (60 Hz sync)
68          game_io_wait_for_vsync();
69
70          // STEP 1: Read input nonblocking, and update paddle
71          poll_mouse_and_update_paddle(mouse_fd1, &p1, &puck,
                P1_X_MIN, P1_X_MAX, PADDLE_Y_MIN, PADDLE_Y_MAX,
                P1_MOUSE_X_SIGN, P1_MOUSE_Y_SIGN);
72          poll_mouse_and_update_paddle(mouse_fd2, &p2, &puck,
                P2_X_MIN, P2_X_MAX, PADDLE_Y_MIN, PADDLE_Y_MAX,
```

```
            P2_MOUSE_X_SIGN, P2_MOUSE_Y_SIGN);
73
74          // STEP 2: Simulate puck
75          // Run physics engine for this frame
76          if (game_state == 0) {
77              sound_event = simulate_frame(&puck, &p1, &p2);
78          }
79
80          // STEP 3: Detect goal / update game_state / score
81          // If goal: update scores, reset puck to center,
                game_state = 1
82          int score_updater = handle_score_update(&puck, &p1, &
                p2, &p1_score, &p2_score);
83          if (score_updater) {
84              sound_event = AIR_HOCKEY_SOUND_GOAL;
85              game_state = 1;
86          }
87
88          // STEP 4: Push all state to hardware
89          // Send updated coordinates to the Verilog VGA driver
90          write_to_vga_registers(&puck, &p1, &p2, p1_score,
                p2_score, sound_event);
91
92          // STEP 5: Reset after a goal has been scored
93          if (score_updater == 1) {
94              usleep(250000);
95              reset_after_goal(&puck, &p1, &p2, PUCK_START_X_P1
                    );
96          } else if (score_updater == 2) {
97              usleep(250000);
98              reset_after_goal(&puck, &p1, &p2, PUCK_START_X_P2
                    );
99          }
100
101         if (game_state == 1) {
102             game_state = 0;
103             if (p1_score >= MAX_SCORE || p2_score >=
                    MAX_SCORE) {
104                 printf("Game␣over!␣Final␣score:␣P1=%d,␣P2=%d\
                        n", p1_score, p2_score);
105                 for (int i = 3; i > 0; i--) {
106                     printf("Restarting␣game␣in␣%d...\n", i);
107                     write_to_vga_registers(&puck, &p1, &p2, i
                            , i, AIR_HOCKEY_SOUND_GOAL);
108                     usleep(1000000);
```

```
                }
                printf("Game restarted!\n");
                p1_score = 0;
                p2_score = 0;
                game_state = 0;
            }
        }
    }

    close(mouse_fd1);
    close(mouse_fd2);
    game_io_close(); //close the device when done
    return 0;
}
```

1.14 physics_engine.c

```
#include "physics_engine.h"
#include "game_config.h"
#include <float.h> // For DBL_MAX (infinity)

double get_wall_collision_time(const GameObject *puck) {
    double t_min = DBL_MAX;
    double t;

    // Check Left Wall
    if (puck->vel.x < 0) {
        t = (PLAY_LEFT + puck->radius - puck->pos.x) / puck->
            vel.x;
        if(t>=0){
            double hit_y = puck->pos.y + puck->vel.y * t;
            if(!(hit_y >= GOAL_TOP && hit_y <= GOAL_BOTTOM)){
                // If collision with left wall is outside
                    goal area, consider it a valid wall
                    collision
                if (t < t_min) t_min = t;
            }
        }
    }
    // Check Right Wall
    else if (puck->vel.x > 0) {
        t = (PLAY_RIGHT - puck->radius - puck->pos.x) / puck
            ->vel.x;
        if(t>=0){
            double hit_y = puck->pos.y + puck->vel.y * t;
            if(!(hit_y >= GOAL_TOP && hit_y <= GOAL_BOTTOM)){
                // If collision with right wall is outside
                    goal area, consider it a valid wall
                    collision
                if (t < t_min) t_min = t;
            }
        }
    }

    // Check Top Wall
    if (puck->vel.y < 0) {
        t = (PLAY_TOP + puck->radius - puck->pos.y) / puck->
            vel.y;
        if (t >= 0 && t < t_min) t_min = t;
    }
```

```
37      // Check Bottom Wall
38      else if (puck->vel.y > 0) {
39          t = (PLAY_BOTTOM - puck->radius - puck->pos.y) / puck
                ->vel.y;
40          if (t >= 0 && t < t_min) t_min = t;
41      }
42
43      return t_min;
44  }
45
46  // Paddle Collision Time (The Quadratic Solver)
47  double getPaddleCollisionTime(const GameObject *puck, const
        GameObject *paddle) {
48      // Relative position (Delta P)
49      double dp_x = puck->pos.x - paddle->pos.x;
50      double dp_y = puck->pos.y - paddle->pos.y;
51
52      // Relative velocity (Delta V)
53      double dv_x = puck->vel.x - paddle->vel.x;
54      double dv_y = puck->vel.y - paddle->vel.y;
55
56      double r_sum = puck->radius + paddle->radius;
57
58      // A, B, and C for the quadratic equation At^2 + Bt + C =
            0
59      double A = (dv_x * dv_x) + (dv_y * dv_y);
60      double B = 2.0 * ((dp_x * dv_x) + (dp_y * dv_y));
61      double C = (dp_x * dp_x) + (dp_y * dp_y) - (r_sum * r_sum
            );
62
63      // If relative velocity is near 0 (A < EPS), they will
            never collide
64      if (A < 1e-10) return DBL_MAX;
65
66      // Check if they are currently overlapping and moving
            towards each other
67      double closing = (dp_x * dv_x) + (dp_y * dv_y);
68
69      // If objects are already overlapping (C <= 0), collision
            is happening right now
70      if (C <= 0.0){
71          if (closing < 0) {
72              // They are moving towards each other while
                    overlapping, treat as immediate collision
73              return 0.0;
```

```
74          } else {
75              // They are moving apart while overlapping, treat
                    as no future collision
76              return DBL_MAX;
77          }
78      }
79
80      // Calculate discriminant to check if paths intersect
81      double discriminant = (B * B) - (4.0 * A * C);
82
83      // If discriminant is negative, the paths never cross
84      if (discriminant < 0.0) return DBL_MAX;
85
86      // We only want the first intersection (- sqrt), not the
           exit (+ sqrt)
87      double t_c = (-B - sqrt(discriminant)) / (2.0 * A);
88
89      // If t_c is negative, the collision happened in the past
90      if (t_c < 0.0) return DBL_MAX;
91
92      return t_c;
93  }
94
95  // 2D Post-Collision Velocities
96  void applyPaddleCollision(GameObject *puck, const GameObject
       *paddle, double restitution) {
97      // Define the Normal Vector (Line of Action)
98      double Nx = puck->pos.x - paddle->pos.x;
99      double Ny = puck->pos.y - paddle->pos.y;
100
101     // Calculate magnitude to normalize the vector
102     double distance = sqrt((Nx * Nx) + (Ny * Ny));
103
104     // Prevent division by zero just in case of weird
           floating point overlap
105     if (distance == 0.0) return;
106
107     double nx = Nx / distance;
108     double ny = Ny / distance;
109
110     // Define the Tangent Vector (Perpendicular to Normal)
111     double tx = -ny;
112     double ty = nx;
113
```

```
114     // Project Velocities onto Normal and Tangent Axes (Dot
            Products)
115     double U_puck_n = (puck->vel.x * nx) + (puck->vel.y * ny)
            ;
116     double U_puck_t = (puck->vel.x * tx) + (puck->vel.y * ty)
            ;
117 
118     double U_paddle_n = (paddle->vel.x * nx) + (paddle->vel.y
            * ny);
119 
120     // If the puck is already moving away from the paddle
            along the normal,
121     if (U_puck_n >= U_paddle_n) return;
122 
123     // Apply 1D Infinite Mass Collision Formula
124     double V_puck_n_new = (1.0 + restitution) * U_paddle_n -
            (restitution * U_puck_n);
125 
126     // Tangent velocity is unchanged (frictionless)
127     double V_puck_t_new = U_puck_t;
128 
129     // Convert Back to Global X and Y Coordinates
130     puck->vel.x = (V_puck_n_new * nx) + (V_puck_t_new * tx);
131     puck->vel.y = (V_puck_n_new * ny) + (V_puck_t_new * ty);
132 }
133 
134 void applyWallBounce(GameObject *puck, double restitution) {
135     const double EPS = 0.001;
136 
137     double left_bound   = PLAY_LEFT + puck->radius;
138     double right_bound  = PLAY_RIGHT - puck->radius;
139     double top_bound    = PLAY_TOP + puck->radius;
140     double bottom_bound = PLAY_BOTTOM - puck->radius;
141 
142     // Optional safety clamp
143     if (restitution < 0.0) restitution = 0.0;
144     if (restitution > 1.0) restitution = 1.0;
145 
146     // Left wall
147     if (puck->pos.x <= left_bound + EPS && puck->vel.x < 0) {
148         // If within goal area, do not apply wall bounce
149         if (puck->pos.y >= GOAL_TOP && puck->pos.y <=
                GOAL_BOTTOM) {
150             return;
151         }
```

```
152         puck->pos.x = left_bound + EPS;
153         puck->vel.x = -puck->vel.x * restitution;
154     }
155 
156     // Right wall
157     else if (puck->pos.x >= right_bound - EPS && puck->vel.x
        > 0) {
158         // If within goal area, do not apply wall bounce
159         if (puck->pos.y >= GOAL_TOP && puck->pos.y <=
            GOAL_BOTTOM) {
160             return;
161         }
162         puck->pos.x = right_bound - EPS;
163         puck->vel.x = -puck->vel.x * restitution;
164     }
165 
166     // Top wall
167     if (puck->pos.y <= top_bound + EPS && puck->vel.y < 0) {
168         puck->pos.y = top_bound + EPS;
169         puck->vel.y = -puck->vel.y * restitution;
170     }
171 
172     // Bottom wall
173     else if (puck->pos.y >= bottom_bound - EPS && puck->vel.y
        > 0) {
174         puck->pos.y = bottom_bound - EPS;
175         puck->vel.y = -puck->vel.y * restitution;
176     }
177 }
```

1.15 physics_engine.h

```
#ifndef PHYSICS_ENGINE_H
#define PHYSICS_ENGINE_H

#include <math.h>

typedef struct {
    double x;
    double y;
} Vec2D;

typedef struct {
    Vec2D pos;
    Vec2D vel;
    double radius;
} GameObject;

double get_wall_collision_time(const GameObject *puck);
double getPaddleCollisionTime(const GameObject *puck, const
   GameObject *paddle);
void applyPaddleCollision(GameObject *puck, const GameObject
   *paddle, double restitution);
void applyWallBounce(GameObject *puck, double restitution);

#endif
```

2 Software: Drivers

2.1 air_hockey.c

```
/*
 * air_hockey.c
 *
 * Kernel driver for the FPGA air hockey peripheral.
 *
 * Author: Gerald Zhao  Apr-30-2026
 *
 * Role of this file:
 *   - bind to the FPGA peripheral through the device tree
 *   - map the peripheral's memory-mapped register region
 *   - expose a simple device file: /dev/air_hockey
 *   - receive ioctl requests from userspace
 *   - translate logical userspace data into packed 32-bit hardware register
 *     writes
 *
 * This driver hides:
 *   - the peripheral's physical base address
 *   - the raw register packing details
 *     so the userspace program can be neater
 *
 * Userspace only needs to know:
 *   - /dev/air_hockey
 *   - ioctl command numbers
 *   - shared structs from air_hockey.h
 */

#include <linux/module.h>
#include <linux/platform_device.h>
#include <linux/miscdevice.h>
#include <linux/fs.h>
#include <linux/io.h>
#include <linux/of.h>
#include <linux/of_address.h>
#include <linux/uaccess.h>

#include "air_hockey.h"

/*
 * The device name is used in two places:
 *   1. misc device registration -> creates /dev/air_hockey
```

```
41  *   2. kernel log messages
42  */
43 #define DRIVER_NAME "air_hockey"
44
45 /*
46  * Register offsets relative to the peripheral base address.
47  * THIS MUST MATCH HARDWARE SIDE
48  *
49  * Hardware register map:
50  *   0x00 STATUS
51  *   0x04 SOUND_CONTROL
52  *   0x08 P1_POS
53  *   0x0C P2_POS
54  *   0x10 PUCK_POS
55  *   0x14 SCORE
56  */
57 #define REG_STATUS          0x00
58 #define REG_SOUND_CONTROL   0x04
59 #define REG_P1_POS          0x08
60 #define REG_P2_POS          0x0C
61 #define REG_PUCK_POS        0x10
62 #define REG_SCORE           0x14
63
64 /*
65  * Helper macro: convert register offset into mapped I/O
        address.
66  *
67  * dev.virtbase is the kernel's virtual mapping of the device
        register region.
68  * Adding an offset gives the location of one specific
        register.
69  *
70  * dev.virtbase is memory-mapped I/O space, so access must go
        through iowrite32/ioread32.
71  */
72 #define REG32_ADDR(base, off) ((base) + (off))
73
74 /*
75  * Pack logical position from user program into one 32-bit
        hardware register.
76  * note: this is used for both puck and paddle positions,
        they share the same format
77  * Register format:
78  *   bits [15:0]  = x
79  *   bits [31:16] = y
```

```
80  */
81  static inline unsigned int pack_pos(const air_hockey_pos_t *p
        )
82  {
83      return (((unsigned int)p->y) << 16) | ((unsigned int)p->x
            );
84  }
85
86  /*
87   * Pack logical scores into the SCORE register.
88   *
89   * SCORE register:
90   *   bits [2:0] = player 1 score
91   *   bits [5:3] = player 2 score
92   *   bits [31:6] = 0
93   */
94  static inline unsigned int pack_score(const
        air_hockey_score_t *s)
95  {
96      return (((unsigned int)(s->p2 & 0x7)) << 3) |
97             ((unsigned int)(s->p1 & 0x7));
98  }
99
100 /*
101  * Pack one sound event into the SOUND_CONTROL register.
102  *
103  * SOUND_CONTROL register:
104  *   bits [2:0] = sound event ID
105  *   bits [31:3] = 0
106  */
107 static inline unsigned int pack_sound(unsigned char
        sound_event)
108 {
109     return ((unsigned int)(sound_event & 0x7));
110 }
111
112 /*
113  * Driver-private state - only driver can see this instance
114  * This stores both:
115  *   1. the physical resource description (dev.res)
116  *   2. the mapped I/O base pointer used for actual register
         access
117  */
118 struct air_hockey_dev {
```

```
119     struct resource res; // stores the physical addr of the
            registers, for ownership, allocation, release
120     void __iomem *virtbase; // used for actual iowrite32 and
            ioread32 register access
121     // this maps the physical addr to kernel virtual memory
122 } dev; // this instantiate a shared file scope object
123 
124 /*
125  * Write helpers: write position registers.
126  */
127 static void write_p1_pos(const air_hockey_pos_t *p)
128 {
129     iowrite32(pack_pos(p), REG32_ADDR(dev.virtbase,
            REG_P1_POS));
130 }
131 
132 static void write_p2_pos(const air_hockey_pos_t *p)
133 {
134     iowrite32(pack_pos(p), REG32_ADDR(dev.virtbase,
            REG_P2_POS));
135 }
136 
137 static void write_puck_pos(const air_hockey_pos_t *p)
138 {
139     iowrite32(pack_pos(p), REG32_ADDR(dev.virtbase,
            REG_PUCK_POS));
140 }
141 
142 static void write_score(const air_hockey_score_t *s)
143 {
144     iowrite32(pack_score(s), REG32_ADDR(dev.virtbase,
            REG_SCORE));
145 }
146 
147 static void write_sound(unsigned char sound_event)
148 {
149     iowrite32(pack_sound(sound_event), REG32_ADDR(dev.
            virtbase, REG_SOUND_CONTROL));
150 }
151 
152 /*
153  * Read helper: read STATUS register.
154  * This is exposed through AIR_HOCKEY_READ_STATUS so
        userspace can poll
155  * VSYNC_READY and synchronize updates to vertical blank.
```

```
156  */
157 static unsigned int read_status(void)
158 {
159     return ioread32(REG32_ADDR(dev.virtbase, REG_STATUS));
160 }
161
162 /*
163  * ioctl handler
164  *   1. userspace calls ioctl(fd, cmd, arg)
165  *   2. Linux dispatches here through .unlocked_ioctl
166  */
167 static long air_hockey_ioctl(struct file *f, unsigned int cmd
        , unsigned long arg)
168 {
169     air_hockey_pos_arg_t pos_arg;
170     air_hockey_score_arg_t score_arg;
171     air_hockey_sound_arg_t sound_arg;
172     air_hockey_status_t status_arg;
173
174     switch (cmd) {
175
176     case AIR_HOCKEY_WRITE_P1_POS:
177         /*
178          * Copy logical position from userspace into a kernel
                -local struct.
179          * copy_from_user is in uaccess.h and returns 0 on
                success, nonzero on failure.
180          */
181         if (copy_from_user(&pos_arg,
182                            (air_hockey_pos_arg_t *)arg,
183                            sizeof(air_hockey_pos_arg_t)))
184             return -EACCES; // failure and return immediately
185
186         write_p1_pos(&pos_arg.pos);
187         break;
188
189     case AIR_HOCKEY_WRITE_P2_POS:
190         if (copy_from_user(&pos_arg,
191                            (air_hockey_pos_arg_t *)arg,
192                            sizeof(air_hockey_pos_arg_t)))
193             return -EACCES;
194
195         write_p2_pos(&pos_arg.pos);
196         break;
197
```

```
198     case AIR_HOCKEY_WRITE_PUCK_POS:
199         if (copy_from_user(&pos_arg,
200                            (air_hockey_pos_arg_t *)arg,
201                            sizeof(air_hockey_pos_arg_t)))
202             return -EACCES;
203 
204         write_puck_pos(&pos_arg.pos);
205         break;
206 
207     case AIR_HOCKEY_WRITE_SCORE:
208         if (copy_from_user(&score_arg,
209                            (air_hockey_score_arg_t *)arg,
210                            sizeof(air_hockey_score_arg_t)))
211             return -EACCES;
212 
213         write_score(&score_arg.score);
214         break;
215 
216     case AIR_HOCKEY_WRITE_SOUND:
217         if (copy_from_user(&sound_arg,
218                            (air_hockey_sound_arg_t *)arg,
219                            sizeof(air_hockey_sound_arg_t)))
220             return -EACCES;
221 
222         write_sound(sound_arg.sound_event);
223         break;
224 
225     case AIR_HOCKEY_READ_STATUS:
226         /*
227          * reads hardware register
228          * fills a kernel-local result struct
229          * copies that result back to userspace
230          */
231         status_arg.status = read_status();
232 
233         if (copy_to_user((air_hockey_status_t *)arg,
234                          &status_arg,
235                          sizeof(air_hockey_status_t)))
236             return -EACCES;
237         break;
238 
239     default:
240         /*
241          * if ioctl command is not valid
242          */
```

```
243         return -EINVAL;
244     }
245     return 0;
246 }
247 
248 /*
249  * File operations table for /dev/air_hockey
250  * This binds the air_hockey_ioctl to linux when userspace
        calls ioctl()
251  */
252 static const struct file_operations air_hockey_fops = {
253     .owner          = THIS_MODULE,
254     .unlocked_ioctl = air_hockey_ioctl,
255 };
256 
257 /*
258  * misc device descriptor
259  *  This register the driver as a misc device to linux
260  * This is the object that tells Linux:
261  *   - create a misc char device
262  *   - use this device name
263  *   - use this file_operations table
264  *
265  * That essentially creates /dev/air_hockey gets
266  */
267 static struct miscdevice air_hockey_misc_device = {
268     .minor = MISC_DYNAMIC_MINOR,
269     .name  = DRIVER_NAME,
270     .fops  = &air_hockey_fops,
271 };
272 
273 /*
274  * Probe callback
275  *
276  * Called by Linux when the driver is matched to a platform
        device node
277  * from the device tree.
278  *
279  *   1. create /dev/air_hockey
280  *   2. fetch the hardware resource address range from the
        device tree
281  *   3. reserve the region
282  *   4. map it into kernel virtual address space
283  *
```

```
 * After this succeeds, the driver can perform register reads/writes.
 */
static int __init air_hockey_probe(struct platform_device *pdev)
{
    int ret;

    /*
     * Register misc device first so userspace can open /dev/air_hockey
     * once probe succeeds.
     */
    ret = misc_register(&air_hockey_misc_device);
    if (ret)
        return ret;

    /*
     * Fetch the physical resource range from the device tree node
     * corresponding to this peripheral.
     */
    ret = of_address_to_resource(pdev->dev.of_node, 0, &dev.res);
    if (ret) {
        ret = -ENOENT;
        goto out_deregister; // if failed, return immediatly and redo previous step
    }

    /*
     * Reserve ownership of the physical I/O memory region.
     */
    if (request_mem_region(dev.res.start,
                           resource_size(&dev.res),
                           DRIVER_NAME) == NULL) {
        ret = -EBUSY;
        goto out_deregister; // redo previous steps
    }

    /*
     * Map the device's register space into kernel virtual memory so
     * iowrite32/ioread32 can use it.
     */
```

```
322     dev.virtbase = of_iomap(pdev->dev.of_node, 0);
323     if (dev.virtbase == NULL) {
324         ret = -ENOMEM;
325         goto out_release_mem_region; // redo previous steps
326     }
327 
328     /*
329      * Initialization writes.
330      *
331      * Defaults so the display starts in a known state.
332      */
333     {
334         air_hockey_pos_t p1   = { .x = 100, .y = 240 };
335         air_hockey_pos_t p2   = { .x = 540, .y = 240 };
336         air_hockey_pos_t puck = { .x = 320, .y = 240 };
337         air_hockey_score_t score = { .p1 = 0, .p2 = 0 };
338 
339         write_p1_pos(&p1);
340         write_p2_pos(&p2);
341         write_puck_pos(&puck);
342         write_score(&score);
343         write_sound(AIR_HOCKEY_SOUND_NONE);
344     }
345 
346     return 0;
347 
348 out_release_mem_region:
349     release_mem_region(dev.res.start, resource_size(&dev.res)
        );
350 out_deregister:
351     misc_deregister(&air_hockey_misc_device);
352     return ret;
353 }
354 
355 /*
356  * Remove callback
357  *
358  * Undo what probe() did.
359  */
360 static int air_hockey_remove(struct platform_device *pdev)
361 {
362     iounmap(dev.virtbase);
363     release_mem_region(dev.res.start, resource_size(&dev.res)
        );
364     misc_deregister(&air_hockey_misc_device);
```

```
365     return 0;
366 }
367
368 /*
369  * Device tree match table.
370  *
371  * IMPORTANT:
372  * The compatible string here must match the one generated by
        hardware
373  * side (through the DT metadata in vga_ball_hw.tcl or its
       renamed equivalent).
374  *
375  * If the strings do not match, Linux will not bind this
       driver to the device.
376  *
377  */
378
379 //  Which "compatible" string(s) to search for in the Device
       Tree
380 #ifdef CONFIG_OF
381 static const struct of_device_id air_hockey_of_match[] = {
382     { .compatible = "csee4840,vga_ball-1.0" },
383     {},
384 };
385 MODULE_DEVICE_TABLE(of, air_hockey_of_match);
386 #endif
387
388 /*
389  * Platform driver object
390  */
391 static struct platform_driver air_hockey_driver = {
392     .driver = {
393         .name = DRIVER_NAME,
394         .owner = THIS_MODULE,
395         .of_match_table = of_match_ptr(air_hockey_of_match),
396     },
397     .remove = __exit_p(air_hockey_remove),
398 };
399
400 /*
401  * Called when the module is loaded.
402  *
403  * This does NOT directly initialize hardware registers
       itself.
404  * It registers the platform driver and lets Linux call the
```

```
405  * probe callback when a matching device tree node is found.
406  */
407 static int __init air_hockey_init(void)
408 {
409     pr_info(DRIVER_NAME ":␣init\n");
410     return platform_driver_probe(&air_hockey_driver,
        air_hockey_probe);
411     /* passing in function name as argument let's it decays
        into a
412      * pointer to the function
413      */
414 }
415
416 /*
417  * Called when the module is unloaded.
418  */
419 static void __exit air_hockey_exit(void)
420 {
421     platform_driver_unregister(&air_hockey_driver);
422     pr_info(DRIVER_NAME ":␣exit\n");
423 }
424
425 module_init(air_hockey_init);
426 module_exit(air_hockey_exit);
427
428 MODULE_LICENSE("GPL");
429 MODULE_AUTHOR("Gerald␣Zhao␣/␣adapted␣from␣CSEE␣4840␣lab␣
    skeleton");
430 MODULE_DESCRIPTION("FPGA␣air␣hockey␣driver");
```

2.2 air_hockey.h

```
#ifndef AIR_HOCKEY_H
#define AIR_HOCKEY_H

/*
 * air_hockey.h
 *
 * Shared interface between:
 *   1. userspace programs (test app / game logic)
 *   2. kernel driver (air_hockey.c)
 *
 * This file should be included by BOTH sides so they agree on:
 *   - ioctl command numbers
 *   - struct layouts
 *   - semantic meaning of each request
 *
 * Design rule:
 *   Userspace deals in logical game data (x, y, scores, sound event IDs).
 *   The driver is responsible for packing those values into 32-bit hardware
 *   register writes.
 */

#include <linux/ioctl.h>

/*
 * AIR_HOCKEY_MAGIC
 *
 * ioctl command families need a "magic" identifying character so command
 * numbers do not collide with unrelated drivers.
 *
 * This can be almost any value, but it should stay fixed once chosen.
 */
#define AIR_HOCKEY_MAGIC 'q'

/*
 * Logical 2D position in screen pixel coordinates.
 *
 * Coordinates are measured relative to the top-left corner of the visible
```

```
38  * display region:
39  *   - x increases to the right
40  *   - y increases downward
41  *
42  * These are logical values used by userspace and driver code.
43  * The driver packs them into one 32-bit register:
44  *
45  *   bits [15:0]  = x
46  *   bits [31:16] = y
47  *
48  * We use unsigned short because the hardware register map allocates 16 bits
49  * per coordinate.
50  */
51 typedef struct {
52     unsigned short x;
53     unsigned short y;
54 } air_hockey_pos_t;
55
56 /*
57  * Score structure used by userspace.
58  *
59  * The hardware SCORE register is packed as:
60  *   bits [2:0] = player 1 score
61  *   bits [5:3] = player 2 score
62  *
63  * We keep this as a logical struct here so userspace does not need to
64  * manually pack bitfields.
65  */
66 typedef struct {
67     unsigned char p1;
68     unsigned char p2;
69 } air_hockey_score_t;
70
71 /*
72  * Status returned by the driver when userspace wants to read hardware state.
73  *
74  * For the current design, we only expose the raw STATUS register value.
75  * Userspace can inspect bit 0 to check VSYNC_READY.
76  *
```

```
77  * Later, if you want, you can expand this struct to also
        include:
78  *   - frame counter
79  *   - sound busy
80  *   - debug flags
81  * without changing the general driver pattern.
82  */
83 typedef struct {
84     unsigned int status;
85 } air_hockey_status_t;
86
87 /*
88  * Separate argument wrappers for ioctl commands.
89  *
90  * Why wrap instead of passing air_hockey_pos_t directly?
91  * Mostly for consistency and future-proofing:
92  *   - easier to expand later
93  *   - makes each ioctl's payload semantics clearer
94  *   - mirrors the lab3 style where one argument struct was
        passed
95  */
96 typedef struct {
97     air_hockey_pos_t pos;
98 } air_hockey_pos_arg_t;
99
100 typedef struct {
101     air_hockey_score_t score;
102 } air_hockey_score_arg_t;
103
104 typedef struct {
105     unsigned char sound_event;
106 } air_hockey_sound_arg_t;
107
108 /*
109  * Sound event IDs understood by both userspace and driver.
110  *
111  * These values are what the hardware sound block expects in
        the
112  * SOUND_CONTROL register's low bits.
113  *
114  * Keep these synchronized with the hardware documentation.
115  */
116 #define AIR_HOCKEY_SOUND_NONE        0
117 #define AIR_HOCKEY_SOUND_HIT_WALL    1
118 #define AIR_HOCKEY_SOUND_HIT_PADDLE  2
```

```
119 #define AIR_HOCKEY_SOUND_GOAL        3
120
121 /*
122  * ioctl command definitions
123  *
124  * _IOW = userspace writes data into the driver
125  * _IOR = userspace reads data back from the driver
126  *
127  * Each command number should stay stable once userspace and
        driver are both
128  * using it.
129  */
130
131 /* Write player 1 paddle position to hardware */
132 #define AIR_HOCKEY_WRITE_P1_POS \
133     _IOW(AIR_HOCKEY_MAGIC, 1, air_hockey_pos_arg_t)
134
135 /* Write player 2 paddle position to hardware */
136 #define AIR_HOCKEY_WRITE_P2_POS \
137     _IOW(AIR_HOCKEY_MAGIC, 2, air_hockey_pos_arg_t)
138
139 /* Write puck position to hardware */
140 #define AIR_HOCKEY_WRITE_PUCK_POS \
141     _IOW(AIR_HOCKEY_MAGIC, 3, air_hockey_pos_arg_t)
142
143 /* Write score display values to hardware */
144 #define AIR_HOCKEY_WRITE_SCORE \
145     _IOW(AIR_HOCKEY_MAGIC, 4, air_hockey_score_arg_t)
146
147 /* Write one sound event code to hardware */
148 #define AIR_HOCKEY_WRITE_SOUND \
149     _IOW(AIR_HOCKEY_MAGIC, 5, air_hockey_sound_arg_t)
150
151 /* Read back hardware STATUS register through the driver */
152 #define AIR_HOCKEY_READ_STATUS \
153     _IOR(AIR_HOCKEY_MAGIC, 6, air_hockey_status_t)
154
155 /*
156  * STATUS register bit definitions
157  *
158  * Userspace can use these helpers after reading status.
159  */
160 #define AIR_HOCKEY_STATUS_VSYNC_READY  (1u << 0)
161
162 #endif
```

2.3 Makefile (Driver)

```
#
    ================================================================================

# driver/Makefile
#
# Purpose:
#   Build the Linux kernel module for the FPGA air hockey
    driver.
#

# This Makefile supports two modes:
#
#   Mode A: kernel build system mode
#     When the Linux kernel recursively invokes this Makefile
    , it defines
#     KERNELRELEASE. In that case, do NOT run normal targets.
#     Simply tell the kernel which module object to build.
#
#   Mode B: normal user-invoked mode
#     When you or the root Makefile call:
#         make -C driver
#     KERNELRELEASE is not defined yet, so this Makefile
    invokes the kernel
#     build system
#
    ================================================================================



#
    --------------------------------------------------------------------------------

# Mode A: being called from inside the Linux kernel build
    system
ifneq ($(KERNELRELEASE),)

# Tell Kbuild which module object to build.
#
# air_hockey.o is compiled from air_hockey.c and then linked
    into air_hockey.ko
#
# Result:
#   air_hockey.ko   <- loadable kernel module
obj-m := air_hockey.o
```

```
33
34 #
   ------------------------------------------------------------------------
35 # Mode B: normal user-invoked mode
36 else
37
38 # Allow the parent/root Makefile to pass KERNEL_SOURCE in.
39 # If not passed, fall back to the currently running kernel
   version.
40 KERNEL_SOURCE ?= /usr/src/linux-headers-$(shell uname -r)
41
42 # Absolute path to this driver directory.
43 #
44 # The kernel build system needs this so it knows which
   external module
45 # directory to compile.
46 PWD := $(shell pwd)
47
48 # PHONY targets are logical commands, not real files.
49 .PHONY: default all clean help
50
51 # Default target:
52 # Build the kernel module.
53 default: all
54
55 all:
56         $(MAKE) -C $(KERNEL_SOURCE) SUBDIRS=$(PWD) modules
57
58 # Clean target:
59 # Ask the kernel build system to remove generated module-
   build outputs.
60 #
61 # This removes files such as:
62 #   *.o
63 #   *.ko
64 #   *.mod.c
65 #   Module.symvers
66 #   modules.order
67 # and other generated Kbuild artifacts in this directory.
68 clean:
69         $(MAKE) -C $(KERNEL_SOURCE) SUBDIRS=$(PWD) clean
70
71 # Optional helper target.
72 help:
```

```
73          @echo "driver/Makefile␣targets:"
74          @echo "␣␣make␣␣␣␣␣␣␣␣-␣build␣air_hockey.ko"
75          @echo "␣␣make␣clean␣␣␣-␣remove␣generated␣module␣files"
76
77  endif
```

3 Software: Tests

3.1 test_one_mouse.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/input.h>

#include "../driver/air_hockey.h"

/* Area bounds (wall + puck radius = 10+4+10 = 24) */
#define X_MIN 24
#define X_MAX 615
#define Y_MIN 24
#define Y_MAX 455

/*
 * Global file descriptor for the device.
 *
 * This is the handle returned by open("/dev/air_hockey", ...).
 * It is how userspace refers to the driver after opening the device.
 */
static int air_hockey_fd;

/*
 * Helper: read STATUS register through the driver.
 *
 * Returns the raw status value on success.
 * Exits on failure because this is a simple test utility, not production code.
 */
static uint32_t read_status(void)
{
    air_hockey_status_t st;

    if (ioctl(air_hockey_fd, AIR_HOCKEY_READ_STATUS, &st)) {
        perror("ioctl(AIR_HOCKEY_READ_STATUS) failed");
        exit(1);
    }
```

```
39
40      return st.status;
41  }
42
43  static int clamp_mouse(int val, int lo, int hi)
44  {
45      if (val < lo) return lo;
46      if (val > hi) return hi;
47      return val;
48  }
49
50  /*
51   * Helper: wait until VSYNC_READY is asserted.
52   *
53   * Why do this?
54   *   The design intent is for software to update visible game
         -state registers
55   *   during vertical blanking, which reduces the chance of
         visible tearing.
56   *
57   * For an MVP, polling is simple and good enough.
58   */
59  static void wait_for_vsync_ready(void)
60  {
61      while ((read_status() & AIR_HOCKEY_STATUS_VSYNC_READY) ==
             0) {
62          /* busy-wait */
63      }
64  }
65
66  /*
67   * Helper: write one logical position to a specific ioctl
         command.
68   *
69   * This keeps the repeated userspace pattern simple:
70   *   fill wrapper struct
71   *   call ioctl
72   *   check error
73   */
74  static void write_pos(unsigned long cmd, uint16_t x, uint16_t
         y)
75  {
76      air_hockey_pos_arg_t arg;
77
78      arg.pos.x = x;
```

```
79      arg.pos.y = y;
80
81      if (ioctl(air_hockey_fd, cmd, &arg)) {
82          perror("position ioctl failed");
83          exit(1);
84      }
85  }
86
87  /*
88   * Helper: write score values.
89   */
90  static void write_score(uint8_t p1, uint8_t p2)
91  {
92      air_hockey_score_arg_t arg;
93
94      arg.score.p1 = p1;
95      arg.score.p2 = p2;
96
97      if (ioctl(air_hockey_fd, AIR_HOCKEY_WRITE_SCORE, &arg)) {
98          perror("ioctl(AIR_HOCKEY_WRITE_SCORE) failed");
99          exit(1);
100     }
101 }
102
103 /*
104  * Helper: write sound event.
105  */
106 static void write_sound(uint8_t event_id)
107 {
108     air_hockey_sound_arg_t arg;
109
110     arg.sound_event = event_id;
111
112     if (ioctl(air_hockey_fd, AIR_HOCKEY_WRITE_SOUND, &arg)) {
113         perror("ioctl(AIR_HOCKEY_WRITE_SOUND) failed");
114         exit(1);
115     }
116 }
117
118 /*
119  * Convenience wrappers so call sites are easier to read.
120  */
121 static void set_p1(uint16_t x, uint16_t y)
122 {
123     write_pos(AIR_HOCKEY_WRITE_P1_POS, x, y);
```

```
124 }
125
126 static void set_p2(uint16_t x, uint16_t y)
127 {
128     write_pos(AIR_HOCKEY_WRITE_P2_POS, x, y);
129 }
130
131 static void set_puck(uint16_t x, uint16_t y)
132 {
133     write_pos(AIR_HOCKEY_WRITE_PUCK_POS, x, y);
134 }
135
136
137 int main(int argc, char *argv[])
138 {
139     static const char filename[] = "/dev/air_hockey";
140
141     const char *dev_mouse = (argc > 1) ? argv[1] : "/dev/
        input/event0";
142
143
144     printf("air_hockey␣test_positions␣starting\n");
145
146     /*
147      * Open the driver-exposed device file.
148      *
149      * This is not a normal data file.
150      * It is the userspace handle to the kernel driver.
151      */
152     air_hockey_fd = open(filename, O_RDWR);
153     if (air_hockey_fd == -1) {
154         perror("open(/dev/air_hockey)␣failed");
155         return 1;
156     }
157
158     /*
159      * Initialize a visible starting scene.
160      */
161     wait_for_vsync_ready();
162     set_p1(100, 240);
163     set_p2(540, 240);
164     set_puck(320, 240);
165     write_score(0, 0);
166     write_sound(AIR_HOCKEY_SOUND_NONE);
167
```

```
168     sleep(1);
169
170     struct input_event ev;
171     int mouse_fd;
172     int x = 320, y = 240;
173
174     mouse_fd = open(dev_mouse, O_RDONLY);
175     if (mouse_fd == -1) {
176         fprintf(stderr, "could␣not␣open␣%s\n", dev_mouse);
177         return -1;
178     }
179
180     printf("Air␣Hockey␣started␣  ␣move␣mouse␣to␣control␣puck
        \n");
181     printf("Using␣mouse␣device:␣%s\n", dev_mouse);
182     set_p1(x, y);
183
184
185     while (read(mouse_fd, &ev, sizeof(ev)) == sizeof(ev)) {
186         if (ev.type == EV_REL) {
187             if (ev.code == REL_X)
188                 x = clamp_mouse(x + ev.value, X_MIN, X_MAX);
189             else if (ev.code == REL_Y)
190                 y = clamp_mouse(y + ev.value, Y_MIN, Y_MAX);
191             wait_for_vsync_ready();
192             set_p1(x, y);
193         }
194     }
195
196
197     close(mouse_fd);
198     close(air_hockey_fd);
199     return 0;
200 }
```

3.2 test_positions.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>

#include "../driver/air_hockey.h"

/*
 * Global file descriptor for the device.
 *
 * This is the handle returned by open("/dev/air_hockey", ...).
 * It is how userspace refers to the driver after opening the device.
 */
static int air_hockey_fd;

/*
 * Helper: read STATUS register through the driver.
 *
 * Returns the raw status value on success.
 * Exits on failure because this is a simple test utility, not production code.
 */
static uint32_t read_status(void)
{
    air_hockey_status_t st;

    if (ioctl(air_hockey_fd, AIR_HOCKEY_READ_STATUS, &st)) {
        perror("ioctl(AIR_HOCKEY_READ_STATUS) failed");
        exit(1);
    }

    return st.status;
}

/*
 * Helper: wait until VSYNC_READY is asserted.
 *
 * Why do this?
```

```
 *   The design intent is for software to update visible game-state registers
 *   during vertical blanking, which reduces the chance of visible tearing.
 *
 * For an MVP, polling is simple and good enough.
 */
static void wait_for_vsync_ready(void)
{
    while ((read_status() & AIR_HOCKEY_STATUS_VSYNC_READY) == 0) {
        /* busy-wait */
    }
}

/*
 * Helper: write one logical position to a specific ioctl command.
 *
 * This keeps the repeated userspace pattern simple:
 *   fill wrapper struct
 *   call ioctl
 *   check error
 */
static void write_pos(unsigned long cmd, uint16_t x, uint16_t y)
{
    air_hockey_pos_arg_t arg;

    arg.pos.x = x;
    arg.pos.y = y;

    if (ioctl(air_hockey_fd, cmd, &arg)) {
        perror("position ioctl failed");
        exit(1);
    }
}

/*
 * Helper: write score values.
 */
static void write_score(uint8_t p1, uint8_t p2)
{
    air_hockey_score_arg_t arg;

```

```
80      arg.score.p1 = p1;
81      arg.score.p2 = p2;
82
83      if (ioctl(air_hockey_fd, AIR_HOCKEY_WRITE_SCORE, &arg)) {
84          perror("ioctl(AIR_HOCKEY_WRITE_SCORE) failed");
85          exit(1);
86      }
87  }
88
89  /*
90   * Helper: write sound event.
91   */
92  static void write_sound(uint8_t event_id)
93  {
94      air_hockey_sound_arg_t arg;
95
96      arg.sound_event = event_id;
97
98      if (ioctl(air_hockey_fd, AIR_HOCKEY_WRITE_SOUND, &arg)) {
99          perror("ioctl(AIR_HOCKEY_WRITE_SOUND) failed");
100         exit(1);
101     }
102 }
103
104 /*
105  * Convenience wrappers so call sites are easier to read.
106  */
107 static void set_p1(uint16_t x, uint16_t y)
108 {
109     write_pos(AIR_HOCKEY_WRITE_P1_POS, x, y);
110 }
111
112 static void set_p2(uint16_t x, uint16_t y)
113 {
114     write_pos(AIR_HOCKEY_WRITE_P2_POS, x, y);
115 }
116
117 static void set_puck(uint16_t x, uint16_t y)
118 {
119     write_pos(AIR_HOCKEY_WRITE_PUCK_POS, x, y);
120 }
121
122 int main(void)
123 {
124     static const char filename[] = "/dev/air_hockey";
```

```
125
126     printf("air_hockey test_positions starting\n");
127
128     /*
129      * Open the driver-exposed device file.
130      *
131      * This is not a normal data file.
132      * It is the userspace handle to the kernel driver.
133      */
134     air_hockey_fd = open(filename, O_RDWR);
135     if (air_hockey_fd == -1) {
136         perror("open(/dev/air_hockey) failed");
137         return 1;
138     }
139
140     /*
141      * Initialize a visible starting scene.
142      */
143     wait_for_vsync_ready();
144     set_p1(100, 240);
145     set_p2(540, 240);
146     set_puck(320, 240);
147     write_score(0, 0);
148     write_sound(AIR_HOCKEY_SOUND_NONE);
149
150     sleep(1);
151
152     /*
153      * Move puck to a few fixed points so we can confirm the
           entire path:
154      * userspace -> ioctl -> driver -> hardware register ->
           renderer.
155      */
156     wait_for_vsync_ready();
157     set_puck(160, 120);
158     sleep(1);
159
160     wait_for_vsync_ready();
161     set_puck(480, 120);
162     sleep(1);
163
164     wait_for_vsync_ready();
165     set_puck(480, 360);
166     sleep(1);
167
```

```
168     wait_for_vsync_ready();
169     set_puck(160, 360);
170     sleep(1);
171 
172     wait_for_vsync_ready();
173     set_puck(320, 240);
174     sleep(1);
175 
176     /*
177      * Move paddles too, so you confirm each independent
            register path.
178      */
179     wait_for_vsync_ready();
180     set_p1(140, 200);
181     set_p2(500, 280);
182     sleep(1);
183 
184     wait_for_vsync_ready();
185     set_p1(140, 280);
186     set_p2(500, 200);
187     sleep(1);
188 
189     /*
190      * Test score and sound path once.
191      */
192     wait_for_vsync_ready();
193     write_score(1, 2);
194     write_sound(AIR_HOCKEY_SOUND_GOAL);
195     sleep(1);
196 
197     close(air_hockey_fd);
198     printf("air_hockey␣test_positions␣done\n");
199 
200     return 0;
201 }
```

3.3 test_score_display.c

```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

#include "../app/game_io.h"

static void show_score(unsigned char p1, unsigned char p2,
    int seconds)
{
    fprintf(stderr, "[test_score_display] setting score P1=%u
        P2=%u\n", p1, p2);

    /*
     * Match the main app behavior: wait for VSYNC before
         updating visible state.
     */
    game_io_wait_for_vsync();

    game_io_set_score(p1, p2);

    sleep(seconds);
}

int main(void)
{
    fprintf(stderr, "Starting score display test...\n");

    if (game_io_open("/dev/air_hockey") != 0) {
        fprintf(stderr, "Failed to open /dev/air_hockey,
            errno=%d\n", errno);
        return 1;
    }

    /*
     * First clear both scores.
     */
    show_score(0, 0, 2);

    /*
     * Test obvious individual values.
     */
    show_score(1, 0, 2);
    show_score(0, 1, 2);
```

```
    show_score(3, 5, 2);
    show_score(7, 7, 2);

    /*
     * Count P1 from 0 to 7 while P2 stays 0.
     */
    for (unsigned char i = 0; i <= 7; i++) {
        show_score(i, 0, 1);
    }

    /*
     * Count P2 from 0 to 7 while P1 stays 0.
     */
    for (unsigned char i = 0; i <= 7; i++) {
        show_score(0, i, 1);
    }

    /*
     * Count both together.
     */
    for (unsigned char i = 0; i <= 7; i++) {
        show_score(i, i, 1);
    }

    /*
     * Leave display at 0-0 when done.
     */
    show_score(0, 0, 2);

    game_io_close();

    fprintf(stderr, "Score␣display␣test␣complete.\n");
    return 0;
}
```

3.4 test_sound.c

```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

#include "../app/game_io.h"

static void play_sound_event(unsigned char event_id, const
    char *name)
{
    fprintf(stderr, "[test_sound] sending %s, event_id=%u\n",
            name, event_id);

    /*
     * Wait for a clean frame boundary before sending the
        event.
     * This matches how the main game writes frame state.
     */
    game_io_wait_for_vsync();

    game_io_set_sound(event_id);

    /*
     * Give hardware time to play the sound before sending
        the next one.
     * Adjust if your sound effect is longer/shorter.
     */
    sleep(1);
}

int main(void)
{
    fprintf(stderr, "Starting sound test through game_io +
       kernel driver...\n");

    if (game_io_open("/dev/air_hockey") != 0) {
        fprintf(stderr, "Failed to open /dev/air_hockey,
           errno=%d\n", errno);
        return 1;
    }

    fprintf(stderr, "Opened /dev/air_hockey successfully.\n")
       ;

```

```
38      /*
39       * Send NONE first to clear/idle the sound command.
40       */
41      play_sound_event(AIR_HOCKEY_SOUND_NONE, "NONE");
42
43      /*
44       * Test each real sound effect individually.
45       */
46      play_sound_event(AIR_HOCKEY_SOUND_HIT_WALL, "HIT_WALL");
47      play_sound_event(AIR_HOCKEY_SOUND_HIT_PADDLE, "HIT_PADDLE
            ");
48      play_sound_event(AIR_HOCKEY_SOUND_GOAL, "GOAL");
49
50      /*
51       * Repeat a few times so it is easy to hear.
52       */
53      for (int i = 0; i < 3; i++) {
54          fprintf(stderr, "[test_sound] repeat round %d\n", i +
                1);
55
56          play_sound_event(AIR_HOCKEY_SOUND_HIT_WALL, "HIT_WALL
                ");
57          play_sound_event(AIR_HOCKEY_SOUND_HIT_PADDLE, "
                HIT_PADDLE");
58          play_sound_event(AIR_HOCKEY_SOUND_GOAL, "GOAL");
59      }
60
61      /*
62       * Return to no sound.
63       */
64      play_sound_event(AIR_HOCKEY_SOUND_NONE, "NONE");
65
66      game_io_close();
67
68      fprintf(stderr, "Sound test complete.\n");
69      return 0;
70  }
```

3.5 test_starting_pos.c

```
#include <stdio.h>
#include <stdbool.h>
#include <errno.h>
#include <math.h>
#include <float.h>
#include <string.h>
#include <unistd.h>

#include "../app/physics_engine.h"
#include "../app/game_io.h"

void write_to_vga_registers(GameObject *puck,
                            GameObject *p1,
                            GameObject *p2,
                            int p1_score,
                            int p2_score,
                            unsigned char sound_event)
{
    printf("[write_to_vga_registers]\n");
    printf("␣␣P1␣␣␣GameObject=(%.2f,␣%.2f)␣->␣ushort=(%hu,␣%
       hu)\n",
           p1->pos.x, p1->pos.y,
           (unsigned short)p1->pos.x, (unsigned short)p1->pos
              .y);

    printf("␣␣P2␣␣␣GameObject=(%.2f,␣%.2f)␣->␣ushort=(%hu,␣%
       hu)\n",
           p2->pos.x, p2->pos.y,
           (unsigned short)p2->pos.x, (unsigned short)p2->pos
              .y);

    printf("␣␣Puck␣GameObject=(%.2f,␣%.2f)␣->␣ushort=(%hu,␣%
       hu)\n",
           puck->pos.x, puck->pos.y,
           (unsigned short)puck->pos.x, (unsigned short)puck
              ->pos.y);

    printf("␣␣Score=(%d,␣%d),␣sound=%u\n\n",
           p1_score, p2_score, sound_event);

    game_io_set_p1((unsigned short)p1->pos.x, (unsigned short
       )p1->pos.y);
```

```
36      game_io_set_p2((unsigned short)p2->pos.x, (unsigned short
            )p2->pos.y);
37      game_io_set_puck((unsigned short)puck->pos.x, (unsigned
            short)puck->pos.y);
38      game_io_set_score((unsigned char)p1_score, (unsigned char
            )p2_score);
39      game_io_set_sound(sound_event);
40  }
41
42  int main(void)
43  {
44      printf("Starting␣GameObject␣->␣game_io␣hardware␣path␣test
            ...\n");
45
46      GameObject p1 = {{100.0, 240.0}, {0.0, 0.0}, 20.0};
47      GameObject p2 = {{540.0, 240.0}, {0.0, 0.0}, 20.0};
48      GameObject puck = {{320.0, 240.0}, {-20.0, -15.0}, 10.0};
49
50      int p1_score = 0;
51      int p2_score = 0;
52
53      if (game_io_open("/dev/air_hockey") != 0) {
54          fprintf(stderr, "Failed␣to␣open␣/dev/air_hockey,␣
                errno=%d\n", errno);
55          return 1;
56      }
57
58      printf("Opened␣/dev/air_hockey␣successfully.\n");
59
60      printf("Test␣1:␣same␣initial␣positions␣as␣main\n");
61      game_io_wait_for_vsync();
62      write_to_vga_registers(&puck, &p1, &p2, p1_score,
            p2_score, AIR_HOCKEY_SOUND_NONE);
63      sleep(3);
64
65      printf("Test␣2:␣puck␣at␣legal␣top-left\n");
66      puck.pos.x = 20.0;
67      puck.pos.y = 20.0;
68      p1_score = 1;
69      game_io_wait_for_vsync();
70      write_to_vga_registers(&puck, &p1, &p2, p1_score,
            p2_score, AIR_HOCKEY_SOUND_NONE);
71      sleep(3);
72
73      printf("Test␣3:␣puck␣at␣legal␣top-right\n");
```

```
74      puck.pos.x = 620.0;
75      puck.pos.y = 20.0;
76      p2_score = 1;
77      game_io_wait_for_vsync();
78      write_to_vga_registers(&puck, &p1, &p2, p1_score,
           p2_score, AIR_HOCKEY_SOUND_NONE);
79      sleep(3);
80
81      printf("Test␣4:␣puck␣at␣legal␣bottom-right\n");
82      puck.pos.x = 620.0;
83      puck.pos.y = 460.0;
84      p1_score = 2;
85      game_io_wait_for_vsync();
86      write_to_vga_registers(&puck, &p1, &p2, p1_score,
           p2_score, AIR_HOCKEY_SOUND_NONE);
87      sleep(3);
88
89      printf("Test␣5:␣puck␣at␣legal␣bottom-left\n");
90      puck.pos.x = 20.0;
91      puck.pos.y = 460.0;
92      p2_score = 2;
93      game_io_wait_for_vsync();
94      write_to_vga_registers(&puck, &p1, &p2, p1_score,
           p2_score, AIR_HOCKEY_SOUND_NONE);
95      sleep(3);
96
97      printf("Test␣6:␣move␣paddles␣too\n");
98      puck.pos.x = 320.0;
99      puck.pos.y = 240.0;
100     p1.pos.x = 80.0;
101     p1.pos.y = 120.0;
102     p2.pos.x = 560.0;
103     p2.pos.y = 360.0;
104     game_io_wait_for_vsync();
105     write_to_vga_registers(&puck, &p1, &p2, 3, 3,
           AIR_HOCKEY_SOUND_NONE);
106     sleep(3);
107
108     printf("Test␣7:␣sweep␣puck␣left␣to␣right,␣same␣write␣path
           ␣as␣main\n");
109     p1.pos.x = 100.0;
110     p1.pos.y = 240.0;
111     p2.pos.x = 540.0;
112     p2.pos.y = 240.0;
113     puck.pos.y = 240.0;
```

```
114
115     for (double x = 20.0; x <= 620.0; x += 20.0) {
116         puck.pos.x = x;
117         game_io_wait_for_vsync();
118         write_to_vga_registers(&puck, &p1, &p2, 4, 4,
                AIR_HOCKEY_SOUND_NONE);
119         usleep(100000);
120     }
121
122     printf("Test␣8:␣intentional␣bad␣negative␣coordinate␣
            conversion\n");
123     puck.pos.x = -20.0;
124     puck.pos.y = 240.0;
125     game_io_wait_for_vsync();
126     write_to_vga_registers(&puck, &p1, &p2, 9, 9,
            AIR_HOCKEY_SOUND_NONE);
127     sleep(3);
128
129     game_io_close();
130     printf("Test␣complete.\n");
131     return 0;
132 }
```

4 Hardware & System

4.1 Makefile (Hardware)

```
SYSTEM = soc_system

TCL = $(SYSTEM).tcl
QSYS = $(SYSTEM).qsys
SOPCINFO = $(SYSTEM).sopcinfo
QIP = $(SYSTEM)/synthesis/$(SYSTEM).qip
HPS_PIN_TCL = $(SYSTEM)/synthesis/submodules/
   hps_sdram_p0_pin_assignments.tcl
HPS_PIN_MAP = hps_sdram_p0_all_pins.txt
QPF = $(SYSTEM).qpf
QSF = $(SYSTEM).qsf
SDC = $(SYSTEM).sdc
SRF = $(SYSTEM).srf

BOARD_INFO = $(SYSTEM)_board_info.xml
DTS = $(SYSTEM).dts
DTB = $(SYSTEM).dtb

SOF = output_files/$(SYSTEM).sof
RBF = output_files/$(SYSTEM).rbf

SOFTWARE_DIR = software

BSP_DIR = $(SOFTWARE_DIR)/spl_bsp
BSP_SETTINGS = $(BSP_DIR)/settings.bsp
PRELOADER_SETTINGS_DIR = hps_isw_handoff/soc_system_hps_0
PRELOADER_MAKEFILE = $(BSP_DIR)/Makefile
PRELOADER_MKPIMAGE = $(BSP_DIR)/preloader-mkpimage.bin

UBOOT_IMAGE = $(BSP_DIR)/uboot-socfpga/u-boot.img

# We had used Git branch "socfpga-4.19", but that disappeared
# Tag v4.19 seems to be the final version of the 4.19 kernel

KERNEL_REPO = https://github.com/altera-fpga/linux-socfpga.
   git
KERNEL_TAG = v4.19
DEFAULT_CONFIG = socfpga_defconfig
CROSS = env CROSS_COMPILE=arm-altera-eabi- ARCH=arm

KERNEL_DIR = $(SOFTWARE_DIR)/linux-socfpga
```

```
40 KERNEL_CONFIG = $(KERNEL_DIR)/.config
41 ZIMAGE = $(KERNEL_DIR)/arch/arm/boot/zImage
42
43 TARFILES = Makefile \
44         $(TCL) \
45         $(QSYS) \
46         $(SYSTEM)_top.sv \
47         $(BOARD_INFO) \
48         $(SRF) \
49         ip/intr_capturer/intr_capturer.v \
50         ip/intr_capturer/intr_capturer_hw.tcl \
51         vga_ball.sv
52
53 TARFILE = lab3-hw.tar.gz
54
55 # project
56 #
57 # Run the topmost tcl script to generate the initial project
       files
58 #
59 # This also adds the pin constraints for the sdram subsystem,
       which
60 # Platform Designer places in in the HPS_PIN_TCL file.
61 #
62 # However, to run that, quartus_map has to run once to
       prepare the netlist
63 # enough so that the HPS_PIN_TCL script can run
64
65 .PHONY : project
66 project : $(QPF) $(QSF) $(SDC)
67
68 $(QPF) $(QSF) $(SDC) : $(TCL) $(HPS_PIN_TCL)
69         quartus_sh -t $(TCL)
70         quartus_map $(SYSTEM)
71         quartus_sta -t $(HPS_PIN_TCL) $(SYSTEM)
72
73 # qsys
74 #
75 # From the .qsys file, generate the .sopcinfo, .qip, and
       directory
76 # (named according to the system) with all the Verilog files,
       etc.
77
78 .PHONY : qsys
79 qsys : $(SOPCINFO)
```

```
80
81  $(SOPCINFO) $(QIP) $(HPS_PIN_TCL) $(SYSTEM)/ $(
        PRELOADER_SETTINGS_DIR) : $(QSYS)
82          rm -rf $(SOPCINFO) $(SYSTEM)/
83          qsys-generate $(QSYS) --synthesis=VERILOG
84
85  # quartus
86  #
87  # Run Quartus on the Qsys-generated files
88  #
89  # Note that for this to succeed, quartus_map has to be run
        once
90  # on the project so that quartus_sta can be run on the
        HPS_PIN_TCL
91  # script to set up the proper constraints for all the SDRAM
        pins
92
93  .PHONY : quartus
94  quartus : $(SOF)
95
96  $(SOF) $(HPS_PIN_MAP) : $(QIP) $(QPF) $(QSF) $(HPS_PIN_TCL)
97          quartus_sh --flow compile $(QPF)
98
99  # rbf
100 #
101 # Convert the .sof file (for programming through the USB
        blaster)
102 # to an .rbf file to be placed on an SD card and written by u
        -boot
103 .PHONY : rbf
104 rbf : $(RBF)
105
106 $(RBF) : $(SOF)
107         quartus_cpf -c $(SOF) $(RBF)
108
109 # dtb
110 #
111 # Use the .sopcinfo file to generate a device tree blob file
112 # with information about the memory map of the peripherals
113 .PHONY : dtb
114 dtb : $(DTB)
115
116 $(DTB) : $(DTS)
117         @which dtc || (echo "dtc␣not␣found.␣␣Did␣you␣run␣
              embedded_command_shell.sh?"; exit 1)
```

```
118         dtc -I dts -O dtb -o $(DTB) $(DTS)
119
120 $(DTS) : $(SOPCINFO) $(BOARD_INFO)
121         @which sopc2dts || (echo "sopc2dts␣not␣found.␣␣Did␣
                you␣run␣embedded_command_shell.sh?"; exit 1)
122         sopc2dts --input $(SOPCINFO) \
123                 --output $(DTS) \
124                 --type dts \
125                 --board $(BOARD_INFO) \
126                 --clocks
127
128 # preloader
129 #
130 # Builds the SPL's preloader-mkpimage.bin image file, which
       should
131 # be written to the "magic" 3rd parition on the SD card
132 # in software/spl_bsp
133 #
134 # Requires the embedded_command_shell.sh script to have run
       so the compiler,
135 # etc is available
136 #
137 .PHONY : preloader
138 preloader : $(PRELOADER_MKPIMAGE)
139
140 $(PRELOADER_MKPIMAGE) : $(PRELOADER_MAKEFILE) $(BSP_SETTINGS)
141         $(MAKE) -C $(BSP_DIR)
142
143 $(BSP_SETTINGS) $(PRELOADER_MAKEFILE) : $(
       PRELOADER_SETTINGS_DIR)
144         mkdir -p $(BSP_DIR)
145         bsp-create-settings \
146           --type spl \
147           --bsp-dir $(BSP_DIR) \
148           --settings $(BSP_SETTINGS) \
149           --preloader-settings-dir $(PRELOADER_SETTINGS_DIR)
                \
150           --set spl.boot.FAT_SUPPORT 1
151
152 # uboot
153 #
154 # Build the bootloader
155
156 .PHONY : uboot
157 uboot : $(UBOOT_IMAGE)
```

```
158
159 $(UBOOT_IMAGE) : $(PRELOADER_MAKEFILE) $(BSP_SETTINGS)
160         $(MAKE) -C $(BSP_DIR) uboot
161
162 # kernel-download
163 #
164 #   Clone the Linux kernel repository
165 #
166 # kernel-config
167 #
168 #   Set up the default kernel configuration
169 #
170 # kernel-menuconfig
171 #
172 #   (Optional) Access the kernel configuration menu to make
        further
173 #   adjustments about which modules are included
174 #
175 # zimage
176 #
177 #   Compile the kernel
178
179 .PHONY : download-kernel config-kernel zimage
180 kernel-download : $(KERNEL_DIR)
181 kernel-config : $(KERNEL_CONFIG)
182 kernel-menuconfig :
183         $(CROSS) $(MAKE) -C $(KERNEL_DIR) menuconfig
184 zimage : $(ZIMAGE)
185
186 $(KERNEL_DIR) :
187         mkdir -p $(KERNEL_DIR)
188         git clone --branch $(KERNEL_TAG) --single-branch --
            depth 1 \
189                 $(KERNEL_REPO) $(KERNEL_DIR)
190
191 # Configure the kernel.  Start from a provided default,
192 #
193 # Turn off version checking (makes it easier to compile
        kernel
194 # modules and not have them complain about version)
195 #
196 # Turn on large file (+2TB) support, which the ext4
        filesystem
197 # requires by default (it will not be able to mount the root
198 # filesystem read/write otherwise)
```

```
199 $(KERNEL_CONFIG) : $(KERNEL_DIR)
200         $(CROSS) $(MAKE) -C $(KERNEL_DIR) $(DEFAULT_CONFIG)
201         $(KERNEL_DIR)/scripts/config --file $(KERNEL_CONFIG)
               \
202           --disable CONFIG_LOCALVERSION_AUTO \
203           --enable CONFIG_LBDAF \
204           --disable CONFIG_XFS_FS \
205           --disable CONFIG_GFS2_FS \
206           --disable CONFIG_TEST_KMOD
207
208 # Compile the kernel
209
210 $(ZIMAGE) : $(KERNEL_CONFIG)
211         $(CROSS) $(MAKE) -C $(KERNEL_DIR) LOCALVERSION=
               zImage
212
213 # tar
214 #
215 # Build soc_system.tar.gz
216
217 .phony : tar
218 tar : $(TARFILE)
219
220 $(TARFILE) : $(TARFILES)
221         tar zcfC $(TARFILE) .. $(TARFILES:%=lab3-hw/%)
222
223 .phony : hockey
224 hockey:
225         ./build_hockey.sh
226
227 # clean
228 #
229 # Remove all generated files
230
231 .PHONY : clean quartus-clean qsys-clean project-clean
232 clean : quartus-clean qsys-clean project-clean dtb-clean
       preloader-clean \
233         uboot-clean
234
235 project-clean :
236         rm -rf $(QPF) $(QSF) $(SDC)
237
238 qsys-clean :
239         rm -rf $(SOPCINFO) $(QIP) $(SYSTEM)/ .qsys_edit \
240         hps_isw_handoff/ hps_sdram_p0_summary.csv
```

```
241
242 quartus-clean :
243         rm -rf  $(SOF) output_files db incremental_db $(
                SYSTEM).qdf \
244         c5_pin_model_dump.txt $(HPS_PIN_MAP)
245
246 dtb-clean :
247         rm -rf $(DTS) $(DTB)
248
249 preloader-clean :
250         rm -rf $(BSP_DIR)
251
252 uboot-clean :
253         rm -rf $(BSP_DIR)/uboot-socfpga
254
255 kernel-clean :
256         rm -rf $(KERNEL_DIR)
257
258 config-clean :
259         rm -rf $(KERNEL_CONFIG)
```

4.2 build_hockey.sh

```
# build_hockey.sh
# Gerald Zhao zz3427
# Purpose:
#   Run the original Quartus/Qsys build flow inside a temporary sandbox
#   directory so the real project folder stays clean.
#
#   1. Deletes any old temporary build sandbox.
#   2. Creates fresh .hockey_build/ and artifacts/ directories.
#   3. Copies the current project source files into .hockey_build/.
#   4. Runs the normal hardware build flow:
#         make qsys
#         make quartus
#         make rbf
#         make dtb   (after loading the embedded tools environment)
#   5. Copies only the final useful outputs back to artifacts/:
#         soc_system.rbf
#         soc_system.dtb
#         soc_system.dts (if generated)
#
# Run directly with ./build_hockey.sh or through "make hockey"
#
# Result:
#   - project source folder stays clean
#   - temporary junk lives in .hockey_build/
#   - final outputs are collected in artifacts/


#!/usr/bin/env bash
set -euo pipefail

SYSTEM="soc_system"
SRC_DIR="$(pwd)"
BUILD_ROOT="${SRC_DIR}/.hockey_build"
ARTIFACT_DIR="${SRC_DIR}/artifacts"

echo "[1/7] Cleaning old build sandbox"
rm -rf "${BUILD_ROOT}"
```

```
mkdir -p "${BUILD_ROOT}"
mkdir -p "${ARTIFACT_DIR}"

echo "[2/7] Copying source into sandbox"
rsync -a \
  --exclude '.git' \
  --exclude '.hockey_build' \
  --exclude 'artifacts' \
  --exclude 'db' \
  --exclude 'incremental_db' \
  --exclude 'output_files' \
  --exclude 'soc_system' \
  --exclude '*.qpf' \
  --exclude '*.qsf' \
  --exclude '*.sdc' \
  --exclude '*.sopcinfo' \
  --exclude '*.dts' \
  --exclude '*.dtb' \
  --exclude '*.rbf' \
  --exclude '.qsys_edit' \
  "${SRC_DIR}/" "${BUILD_ROOT}/"

cd "${BUILD_ROOT}"

echo "[3/7] Running Qsys generation"
make qsys

echo "[4/7] Running Quartus compile"
make quartus

echo "[5/7] Generating RBF"
make rbf

echo "[6/7] Generating DTB from existing sandbox"

if [ ! -d "${BUILD_ROOT}" ]; then
  echo "Error: sandbox ${BUILD_ROOT} does not exist"
  echo "Run the earlier build steps first."
  exit 1
fi

EMBEDDED_SHELL="$(find /tools/intel -name
   embedded_command_shell.sh -print -quit 2>/dev/null)"

if [ -z "${EMBEDDED_SHELL}" ]; then
```

```
  echo "Error: could not find embedded_command_shell.sh under /tools/intel"
  exit 1
fi

echo "[6.1/7] Using embedded shell: ${EMBEDDED_SHELL}"
echo "[6.2/7] Working in: ${BUILD_ROOT}"
echo "[6.3/7] Open the embedded shell manually if this fails"

echo "[6.4/7] Running DTB build in one command"
bash -lc 'cd "'"${BUILD_ROOT}"'" && "'"${EMBEDDED_SHELL}"'" <<'\''EOS'\''
make dtb
exit
EOS'

echo "[7/7] Copying artifacts back"
mkdir -p "${ARTIFACT_DIR}"

if [ -f "${BUILD_ROOT}/output_files/${SYSTEM}.rbf" ]; then
  cp -f "${BUILD_ROOT}/output_files/${SYSTEM}.rbf" "${ARTIFACT_DIR}/${SYSTEM}.rbf"
  echo "Copied ${ARTIFACT_DIR}/${SYSTEM}.rbf"
else
  echo "Warning: ${BUILD_ROOT}/output_files/${SYSTEM}.rbf not found"
fi

if [ -f "${BUILD_ROOT}/${SYSTEM}.dtb" ]; then
  cp -f "${BUILD_ROOT}/${SYSTEM}.dtb" "${ARTIFACT_DIR}/${SYSTEM}.dtb"
  echo "Copied ${ARTIFACT_DIR}/${SYSTEM}.dtb"
else
  echo "Warning: ${BUILD_ROOT}/${SYSTEM}.dtb not found"
fi

if [ -f "${BUILD_ROOT}/${SYSTEM}.dts" ]; then
  cp -f "${BUILD_ROOT}/${SYSTEM}.dts" "${ARTIFACT_DIR}/${SYSTEM}.dts"
  echo "Copied ${ARTIFACT_DIR}/${SYSTEM}.dts"
fi

echo "Done."
```

4.3 soc_system_board_info.xml

```
<BoardInfo pov="hps_0_arm_a9_0">
<!--
This file is intended to be used when building device trees
for the Altera Cyclone5 SOC Development Kits.
This board info file and hps_clock_info.xml are required input
to sopc2dts to create a device tree suitable for the 3.9 version
of the Linux kernel.  One typically executes sopc2dts as follows:

        sopc2dts -i soc_system.sopcinfo -b soc_system_board_info.xml
             -b hps_clock_info.xml -o soc_system.dts

-->

<DTAppend name="model" type="string" parentlabel="" val="Altera SOCFPGA Cyclone V"/>
<DTAppend name="compatible" parentlabel="" >
        <val type="string">altr,socfpga-cyclone5</val>
        <val type="string">altr,socfpga</val>
</DTAppend>
<IRQMasterIgnore className="intr_capturer"/>
<IRQMasterIgnore className="d_irq"/>

<DTAppend name="next-level-cache" type="phandle" parentlabel="hps_0_arm_a9_0" val="hps_0_L2"/>
<DTAppend name="next-level-cache" type="phandle" parentlabel="hps_0_arm_a9_1" val="hps_0_L2"/>

<DTAppend name="cache-unified" type="bool" parentlabel="hps_0_L2" val="true"/>
<DTAppend name="arm,tag-latency" parentlabel="hps_0_L2">
        <val type="number">1</val>
        <val type="number">1</val>
        <val type="number">1</val>
</DTAppend>
<DTAppend name="arm,data-latency" parentlabel="hps_0_L2">
        <val type="number">2</val>
        <val type="number">1</val>
        <val type="number">1</val>
</DTAppend>
```

```
36
37 <DTAppend name="interrupt-controller" parentlabel="
      hps_0_gpio0"/>
38 <DTAppend name="#interrupt-cells" type="number" parentlabel="
      hps_0_gpio0" val="2"/>
39
40 <DTAppend name="interrupt-controller" parentlabel="
      hps_0_gpio1"/>
41 <DTAppend name="#interrupt-cells" type="number" parentlabel="
      hps_0_gpio1" val="2"/>
42
43 <DTAppend name="interrupt-controller" parentlabel="
      hps_0_gpio2"/>
44 <DTAppend name="#interrupt-cells" type="number" parentlabel="
      hps_0_gpio2" val="2"/>
45
46 <DTAppend name="clock-frequency" type="number" parentlabel="
      hps_0_uart0" val="100000000"/>
47 <DTAppend name="clock-frequency" type="number" parentlabel="
      hps_0_uart1" val="100000000"/>
48 <DTAppend name="clock-frequency" type="number" parentlabel="
      hps_0_timer0" val="100000000"/>
49 <DTAppend name="clock-frequency" type="number" parentlabel="
      hps_0_timer1" val="100000000"/>
50 <DTAppend name="clock-frequency" type="number" parentlabel="
      hps_0_timer2" val="25000000"/>
51 <DTAppend name="clock-frequency" type="number" parentlabel="
      hps_0_timer3" val="25000000"/>
52
53 <DTAppend name="cpu1-start-addr" type="hex" parentlabel="
      hps_0_sysmgr" val="0xffd080c4"/>
54 <DTAppend name="compatible" type="string" parentlabel="
      hps_0_sysmgr" val="syscon" action="add"/>
55 <DTAppend name="compatible" type="string" parentlabel="
      hps_0_rstmgr" val="syscon" action="add"/>
56
57 <DTAppend name="speed-mode" type="number" parentlabel="
      hps_0_i2c0" val="0"/>
58 <DTAppend name="status" type="string" parentlabel="hps_0_i2c1
      " val="disabled"/>
59 <DTAppend name="status" type="string" parentlabel="hps_0_i2c2
      " val="disabled"/>
60 <DTAppend name="status" type="string" parentlabel="hps_0_i2c3
      " val="disabled"/>
61
```

```
62 <DTAppend name="phy-mode" type="string" parentlabel="
      hps_0_gmac1" val="rgmii"/>
63 <DTAppend name="clock-names" type="string" parentlabel="
      hps_0_gmac1" val="stmmaceth"/>
64 <DTAppend name="clocks" type="phandle" parentlabel="
      hps_0_gmac1" val="emac1_clk"/>
65 <DTAppend name="phy-addr" type="hex" parentlabel="hps_0_gmac1
      " val="0xffffffff"/>
66
67 <DTAppend name="micrel-ksz9021rlrn-clk-skew" type="hex"
      parentlabel="hps_0_gmac1" val="0xa0e0"/>
68 <DTAppend name="micrel-ksz9021rlrn-rx-skew" type="hex"
      parentlabel="hps_0_gmac1" val="0x0"/>
69
70
71 <!-- Added by sedwards -->
72 <DTAppend name="altr,sysmgr-syscon" parentlabel="hps_0_gmac1"
      >
73   <val type="phandle">hps_0_sysmgr</val>
74   <val type="hex">0x60</val>
75   <val type="hex">0x2</val>
76 </DTAppend>
77 <DTAppend name="reset-names" type="string" parentlabel="
      hps_0_gmac1" val="stmmaceth"/>
78 <DTAppend name="resets" parentlabel="hps_0_gmac1">
79   <val type="hex">0x21</val>
80   <val type="hex">0x21</val>
81 </DTAppend>
82 <DTAppend name="txd0-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x0"/>
83 <DTAppend name="txd1-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x0"/>
84 <DTAppend name="txd2-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x0"/>
85 <DTAppend name="txd3-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x0"/>
86 <DTAppend name="rxd0-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x1a4"/>
87 <DTAppend name="rxd1-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x1a4"/>
88 <DTAppend name="rxd2-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x1a4"/>
89 <DTAppend name="rxd3-skew-ps"  parentlabel="hps_0_gmac1" type
      ="hex" val="0x1a4"/>
```

```
90  <DTAppend name="txen-skew-ps"  parentlabel="hps_0_gmac1" type
       ="hex" val="0x0"/>
91  <DTAppend name="txc-skew-ps"  parentlabel="hps_0_gmac1" type=
       "hex" val="0x744"/>
92  <DTAppend name="rxdv-skew-ps"  parentlabel="hps_0_gmac1" type
       ="hex" val="0x1a4"/>
93  <DTAppend name="rxc-skew-ps"  parentlabel="hps_0_gmac1" type=
       "hex" val="0x690"/>
94
95
96
97
98  <DTAppend name="status" type="string" parentlabel="
       hps_0_gmac0" val="disabled"/>
99  <DTAppend name="status" type="string" parentlabel="
       hps_0_uart1" val="disabled"/>
100 <DTAppend name="status" type="string" parentlabel="hps_0_usb0
       " val="disabled"/>
101 <DTAppend name="status" type="string" parentlabel="
       hps_0_nand0" val="disabled"/>
102 <DTAppend name="clocks" type="phandle" parentlabel="
       hps_0_nand0" val="nand_clk"/>
103
104 <I2CBus master="hps_0_i2c0">
105         <I2CChip addr="0x28" label="lcd" name="newhaven,nhd
               -0216k3z-nsw-bbw"></I2CChip>
106         <I2CChip addr="0x51" label="eeprom" name="atmel,24c32
               "></I2CChip>
107 </I2CBus>
108 <DTAppend name="height" type="number" parentlabel="lcd" val="
       2"/>
109 <DTAppend name="width" type="number" parentlabel="lcd" val="
       16"/>
110 <DTAppend name="brightness" type="number" parentlabel="lcd"
       val="8"/>
111 <DTAppend name="pagesize" type="number" parentlabel="eeprom"
       val="32"/>
112 <DTAppend name="clocks"  parentlabel="hps_0_sdmmc" >
113 <val type="phandle">l4_mp_clk</val>
114 <val type="phandle">sdmmc_clk</val>
115 </DTAppend>
116 <DTAppend name="clock-names"  parentlabel="hps_0_sdmmc" >
117 <val type="string">biu</val>
118 <val type="string">ciu</val>
119 </DTAppend>
```

```
120 <DTAppend name="#address-cells" type="number" parentlabel="
    hps_0_sdmmc" val="1"/>
121 <DTAppend name="#size-cells" type="number" parentlabel="
    hps_0_sdmmc" val="0"/>
122 <DTAppend name="supports-highspeed"  parentlabel="hps_0_sdmmc
    " />
123 <DTAppend name="broken-cd" type="bool" parentlabel="
    hps_0_sdmmc" val="true"/>
124 <DTAppend name="compatible" type="string" parentlabel="
    hps_0_sdmmc" val="altr,socfpga-dw-mshc" action="replace"/>
125 <DTAppend name="altr,dw-mshc-ciu-div" type="number"
    parentlabel="hps_0_sdmmc" val="3"/>
126 <DTAppend name="altr,dw-mshc-sdr-timing"  parentlabel="
    hps_0_sdmmc" >
127 <val type="number">0</val>
128 <val type="number">3</val>
129 </DTAppend>
130 <DTAppend name="slot@0" type="node" parentlabel="hps_0_sdmmc"
     newlabel="slot_0"/>
131 <DTAppend name="reg" type="number" parentlabel="slot_0" val="
    0"/>
132 <DTAppend name="bus-width" type="number" parentlabel="slot_0"
     val="4"/>
133
134 <DTAppend name="master-ref-clk" type="number" parentlabel="
    hps_0_qspi" val="400000000"/>
135 <DTAppend name="ext-decoder" type="number" parentlabel="
    hps_0_qspi" val="0"/>
136 <DTAppend name="n25q00@0" type="node" parentlabel="hps_0_qspi
    " newlabel="flash0"/>
137 <DTAppend name="#address-cells" type="number" parentlabel="
    hps_0_qspi" val="1"/>
138 <DTAppend name="#size-cells" type="number" parentlabel="
    hps_0_qspi" val="0"/>
139 <DTAppend name="#address-cells" type="number" parentlabel="
    flash0" val="1"/>
140 <DTAppend name="#size-cells" type="number" parentlabel="
    flash0" val="1"/>
141 <DTAppend name="compatible" type="string" parentlabel="flash0
    " val="n25q00"/>
142 <DTAppend name="reg" type="number" parentlabel="flash0" val="
    0"/>
143 <DTAppend name="spi-max-frequency" type="number" parentlabel=
    "flash0" val="100000000"/>
```

```
144 <DTAppend name="page-size" type="number" parentlabel="flash0"
        val="256"/>
145 <DTAppend name="block-size" type="number" parentlabel="flash0
    " val="16"/>
146 <DTAppend name="m25p,fast-read" type="bool" parentlabel="
    flash0" val="true"/>
147 <DTAppend name="read-delay" type="number" parentlabel="flash0
    " val="4"/>
148 <DTAppend name="tshsl-ns" type="number" parentlabel="flash0"
    val="50"/>
149 <DTAppend name="tsd2d-ns" type="number" parentlabel="flash0"
    val="50"/>
150 <DTAppend name="tchsh-ns" type="number" parentlabel="flash0"
    val="4"/>
151 <DTAppend name="tslch-ns" type="number" parentlabel="flash0"
    val="4"/>
152 <DTAppend name="partition@0" type="node" parentlabel="flash0"
        newlabel="part0"/>
153 <DTAppend name="label" type="string" parentlabel="part0" val=
    "Flash␣0␣Raw␣Data"/>
154 <DTAppend name="reg" parentlabel="part0" >
155 <val type="hex">0x0</val>
156 <val type="hex">0x800000</val>
157 </DTAppend>
158 <DTAppend name="partition@800000" type="node" parentlabel="
    flash0" newlabel="part1"/>
159 <DTAppend name="label" type="string" parentlabel="part1" val=
    "Flash␣1␣jffs2␣Filesystem"/>
160 <DTAppend name="reg" parentlabel="part1">
161 <val type="hex">0x800000</val>
162 <val type="hex">0x800000</val>
163 </DTAppend>
164
165 <DTAppend name="pmu0" type="node" parentlabel="sopc0"
    newlabel="pmu"/>
166 <DTAppend name="#address-cells" type="number" parentlabel="
    pmu" val="1"/>
167 <DTAppend name="#size-cells" type="number" parentlabel="pmu"
    val="1"/>
168 <DTAppend name="compatible" type="string" parentlabel="pmu"
    val="arm,cortex-a9-pmu"/>
169 <DTAppend name="interrupt-parent" type="phandle" parentlabel=
    "pmu" val="hps_0_arm_gic_0"/>
170 <DTAppend name="interrupts" parentlabel="pmu">
171 <val type="number">0</val>
```

```
<val type="number">176</val>
<val type="number">4</val>
<val type="number">0</val>
<val type="number">177</val>
<val type="number">4</val>
</DTAppend>
<DTAppend name="ranges" type="bool" parentlabel="pmu" val="
   true"/>

<DTAppend name="cti0@ff118000" type="node" parentlabel="pmu"
   newlabel="cti0"/>
<DTAppend name="compatible" type="string" parentlabel="cti0"
   val="arm,coresight-cti"/>
<DTAppend name="reg" parentlabel="cti0">
<val type="hex">0xff118000</val>
<val type="hex">0x100</val>
</DTAppend>

<DTAppend name="cti0@ff119000" type="node" parentlabel="pmu"
   newlabel="cti1"/>
<DTAppend name="compatible" type="string" parentlabel="cti1"
   val="arm,coresight-cti"/>
<DTAppend name="reg" parentlabel="cti1">
<val type="hex">0xff119000</val>
<val type="hex">0x100</val>
</DTAppend>

<DTAppend name="fpgabridge@0" type="node" parentlabel="sopc0"
    newlabel="fpgabridge0"/>
<DTAppend name="compatible" type="string" parentlabel="
   fpgabridge0" val="altr,socfpga-hps2fpga-bridge"/>
<DTAppend name="label" type="string" parentlabel="fpgabridge0
   " val="hps2fpga"/>
<DTAppend name="clocks" type="phandle" parentlabel="
   fpgabridge0" val="l4_main_clk"/>

<DTAppend name="fpgabridge@1" type="node" parentlabel="sopc0"
    newlabel="fpgabridge1"/>
<DTAppend name="compatible" type="string" parentlabel="
   fpgabridge1" val="altr,socfpga-lwhps2fpga-bridge"/>
<DTAppend name="label" type="string" parentlabel="fpgabridge1
   " val="lwhps2fpga"/>
<DTAppend name="clocks" type="phandle" parentlabel="
   fpgabridge1" val="l4_main_clk"/>

```

```
<DTAppend name="fpgabridge@2" type="node" parentlabel="sopc0"
    newlabel="fpgabridge2"/>
<DTAppend name="compatible" type="string" parentlabel="
   fpgabridge2" val="altr,socfpga-fpga2hps-bridge"/>
<DTAppend name="label" type="string" parentlabel="fpgabridge2
   " val="fpga2hps"/>
<DTAppend name="clocks" type="phandle" parentlabel="
   fpgabridge2" val="l4_main_clk"/>

<DTAppend name="l3regs@0xff800000" type="node" parentlabel="
   sopc0" newlabel="l3regs"/>
<DTAppend name="compatible"  parentlabel="l3regs" >
<val type="string">altr,l3regs</val>
<val type="string">syscon</val>
</DTAppend>
<DTAppend name="reg"  parentlabel="l3regs" >
<val type="hex">0xff800000</val>
<val type="hex">0x1000</val>
</DTAppend>


<DTAppend name="sdrctl@0xffc25000" type="node" parentlabel="
   sopc0" newlabel="sdctrl"/>
<DTAppend name="compatible"  parentlabel="sdctrl" >
<val type="string">altr,sdr-ctl</val>
<val type="string">syscon</val>
</DTAppend>
<DTAppend name="reg"  parentlabel="sdctrl" >
<val type="hex">0xffc25000</val>
<val type="hex">0x1000</val>
</DTAppend>



<Chosen>
        <Bootargs val="console=ttyS0,115200"></Bootargs>
</Chosen>
</BoardInfo>
```

4.4 soc_system.tcl

```
# Generate Quartus project files for the DE1-SoC board
#
# Stephen A. Edwards, Columbia University

# Invoke as
#
# quartus_sh -t soc_system.tcl

set project "soc_system"

# Top-level SystemVerilog file should be <project>_top.sv,
   with Verilog module
# <project>_top in it

set systemVerilogSource "${project}_top.sv"
set qip "${project}/synthesis/${project}.qip"

project_new $project -overwrite

foreach {name value} {
    FAMILY "Cyclone V"
    DEVICE 5CSEMA5F31C6

    PROJECT_OUTPUT_DIRECTORY output_files

    CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS
       REGULAR IO"

    NUM_PARALLEL_PROCESSORS 8

} { set_global_assignment -name $name $value }

set_global_assignment -name TOP_LEVEL_ENTITY "${project}_top"

foreach filename $systemVerilogSource {
    set_global_assignment -name SYSTEMVERILOG_FILE $filename
}

foreach filename $qip {
    set_global_assignment -name QIP_FILE $filename
}

# FPGA pin assignments
```

```
42
43 foreach {pin port} {
44     PIN_AJ4 ADC_CS_N
45     PIN_AK4 ADC_DIN
46     PIN_AK3 ADC_DOUT
47     PIN_AK2 ADC_SCLK
48
49     PIN_K7 AUD_ADCDAT
50     PIN_K8 AUD_ADCLRCK
51     PIN_H7 AUD_BCLK
52     PIN_J7 AUD_DACDAT
53     PIN_H8 AUD_DACLRCK
54     PIN_G7 AUD_XCK
55
56     PIN_AA16 CLOCK2_50
57     PIN_Y26 CLOCK3_50
58     PIN_K14 CLOCK4_50
59     PIN_AF14 CLOCK_50
60
61     PIN_AK14 DRAM_ADDR[0]
62     PIN_AH14 DRAM_ADDR[1]
63     PIN_AG15 DRAM_ADDR[2]
64     PIN_AE14 DRAM_ADDR[3]
65     PIN_AB15 DRAM_ADDR[4]
66     PIN_AC14 DRAM_ADDR[5]
67     PIN_AD14 DRAM_ADDR[6]
68     PIN_AF15 DRAM_ADDR[7]
69     PIN_AH15 DRAM_ADDR[8]
70     PIN_AG13 DRAM_ADDR[9]
71     PIN_AG12 DRAM_ADDR[10]
72     PIN_AH13 DRAM_ADDR[11]
73     PIN_AJ14 DRAM_ADDR[12]
74     PIN_AF13 DRAM_BA[0]
75     PIN_AJ12 DRAM_BA[1]
76     PIN_AF11 DRAM_CAS_N
77     PIN_AK13 DRAM_CKE
78     PIN_AH12 DRAM_CLK
79     PIN_AG11 DRAM_CS_N
80     PIN_AK6 DRAM_DQ[0]
81     PIN_AJ7 DRAM_DQ[1]
82     PIN_AK7 DRAM_DQ[2]
83     PIN_AK8 DRAM_DQ[3]
84     PIN_AK9 DRAM_DQ[4]
85     PIN_AG10 DRAM_DQ[5]
86     PIN_AK11 DRAM_DQ[6]
```

```
    PIN_AJ11 DRAM_DQ[7]
    PIN_AH10 DRAM_DQ[8]
    PIN_AJ10 DRAM_DQ[9]
    PIN_AJ9 DRAM_DQ[10]
    PIN_AH9 DRAM_DQ[11]
    PIN_AH8 DRAM_DQ[12]
    PIN_AH7 DRAM_DQ[13]
    PIN_AJ6 DRAM_DQ[14]
    PIN_AJ5 DRAM_DQ[15]
    PIN_AB13 DRAM_LDQM
    PIN_AE13 DRAM_RAS_N
    PIN_AK12 DRAM_UDQM
    PIN_AA13 DRAM_WE_N

    PIN_AA12 FAN_CTRL

    PIN_J12 FPGA_I2C_SCLK
    PIN_K12 FPGA_I2C_SDAT

    PIN_AC18 GPIO_0[0]
    PIN_Y17  GPIO_0[1]
    PIN_AD17 GPIO_0[2]
    PIN_Y18  GPIO_0[3]
    PIN_AK16 GPIO_0[4]
    PIN_AK18 GPIO_0[5]
    PIN_AK19 GPIO_0[6]
    PIN_AJ19 GPIO_0[7]
    PIN_AJ17 GPIO_0[8]
    PIN_AJ16 GPIO_0[9]
    PIN_AH18 GPIO_0[10]
    PIN_AH17 GPIO_0[11]
    PIN_AG16 GPIO_0[12]
    PIN_AE16 GPIO_0[13]
    PIN_AF16 GPIO_0[14]
    PIN_AG17 GPIO_0[15]
    PIN_AA18 GPIO_0[16]
    PIN_AA19 GPIO_0[17]
    PIN_AE17 GPIO_0[18]
    PIN_AC20 GPIO_0[19]
    PIN_AH19 GPIO_0[20]
    PIN_AJ20 GPIO_0[21]
    PIN_AH20 GPIO_0[22]
    PIN_AK21 GPIO_0[23]
    PIN_AD19 GPIO_0[24]
    PIN_AD20 GPIO_0[25]
```

```
    PIN_AE18 GPIO_0[26]
    PIN_AE19 GPIO_0[27]
    PIN_AF20 GPIO_0[28]
    PIN_AF21 GPIO_0[29]
    PIN_AF19 GPIO_0[30]
    PIN_AG21 GPIO_0[31]
    PIN_AF18 GPIO_0[32]
    PIN_AG20 GPIO_0[33]
    PIN_AG18 GPIO_0[34]
    PIN_AJ21 GPIO_0[35]

    PIN_AB17 GPIO_1[0]
    PIN_AA21 GPIO_1[1]
    PIN_AB21 GPIO_1[2]
    PIN_AC23 GPIO_1[3]
    PIN_AD24 GPIO_1[4]
    PIN_AE23 GPIO_1[5]
    PIN_AE24 GPIO_1[6]
    PIN_AF25 GPIO_1[7]
    PIN_AF26 GPIO_1[8]
    PIN_AG25 GPIO_1[9]
    PIN_AG26 GPIO_1[10]
    PIN_AH24 GPIO_1[11]
    PIN_AH27 GPIO_1[12]
    PIN_AJ27 GPIO_1[13]
    PIN_AK29 GPIO_1[14]
    PIN_AK28 GPIO_1[15]
    PIN_AK27 GPIO_1[16]
    PIN_AJ26 GPIO_1[17]
    PIN_AK26 GPIO_1[18]
    PIN_AH25 GPIO_1[19]
    PIN_AJ25 GPIO_1[20]
    PIN_AJ24 GPIO_1[21]
    PIN_AK24 GPIO_1[22]
    PIN_AG23 GPIO_1[23]
    PIN_AK23 GPIO_1[24]
    PIN_AH23 GPIO_1[25]
    PIN_AK22 GPIO_1[26]
    PIN_AJ22 GPIO_1[27]
    PIN_AH22 GPIO_1[28]
    PIN_AG22 GPIO_1[29]
    PIN_AF24 GPIO_1[30]
    PIN_AF23 GPIO_1[31]
    PIN_AE22 GPIO_1[32]
    PIN_AD21 GPIO_1[33]
```

```
177     PIN_AA20 GPIO_1[34]
178     PIN_AC22 GPIO_1[35]
179
180     PIN_AE26 HEX0[0]
181     PIN_AE27 HEX0[1]
182     PIN_AE28 HEX0[2]
183     PIN_AG27 HEX0[3]
184     PIN_AF28 HEX0[4]
185     PIN_AG28 HEX0[5]
186     PIN_AH28 HEX0[6]
187
188     PIN_AJ29 HEX1[0]
189     PIN_AH29 HEX1[1]
190     PIN_AH30 HEX1[2]
191     PIN_AG30 HEX1[3]
192     PIN_AF29 HEX1[4]
193     PIN_AF30 HEX1[5]
194     PIN_AD27 HEX1[6]
195
196     PIN_AB23 HEX2[0]
197     PIN_AE29 HEX2[1]
198     PIN_AD29 HEX2[2]
199     PIN_AC28 HEX2[3]
200     PIN_AD30 HEX2[4]
201     PIN_AC29 HEX2[5]
202     PIN_AC30 HEX2[6]
203
204     PIN_AD26 HEX3[0]
205     PIN_AC27 HEX3[1]
206     PIN_AD25 HEX3[2]
207     PIN_AC25 HEX3[3]
208     PIN_AB28 HEX3[4]
209     PIN_AB25 HEX3[5]
210     PIN_AB22 HEX3[6]
211
212     PIN_AA24 HEX4[0]
213     PIN_Y23  HEX4[1]
214     PIN_Y24  HEX4[2]
215     PIN_W22  HEX4[3]
216     PIN_W24  HEX4[4]
217     PIN_V23  HEX4[5]
218     PIN_W25  HEX4[6]
219
220     PIN_V25  HEX5[0]
221     PIN_AA28 HEX5[1]
```

```
    PIN_Y27  HEX5[2]
    PIN_AB27 HEX5[3]
    PIN_AB26 HEX5[4]
    PIN_AA26 HEX5[5]
    PIN_AA25 HEX5[6]

    PIN_AA30 IRDA_RXD
    PIN_AB30 IRDA_TXD

    PIN_AA14 KEY[0]
    PIN_AA15 KEY[1]
    PIN_W15  KEY[2]
    PIN_Y16  KEY[3]

    PIN_V16 LEDR[0]
    PIN_W16 LEDR[1]
    PIN_V17 LEDR[2]
    PIN_V18 LEDR[3]
    PIN_W17 LEDR[4]
    PIN_W19 LEDR[5]
    PIN_Y19 LEDR[6]
    PIN_W20 LEDR[7]
    PIN_W21 LEDR[8]
    PIN_Y21 LEDR[9]

    PIN_AD7 PS2_CLK
    PIN_AD9 PS2_CLK2
    PIN_AE7 PS2_DAT
    PIN_AE9 PS2_DAT2

    PIN_AB12 SW[0]
    PIN_AC12 SW[1]
    PIN_AF9  SW[2]
    PIN_AF10 SW[3]
    PIN_AD11 SW[4]
    PIN_AD12 SW[5]
    PIN_AE11 SW[6]
    PIN_AC9  SW[7]
    PIN_AD10 SW[8]
    PIN_AE12 SW[9]

    PIN_H15 TD_CLK27
    PIN_D2  TD_DATA[0]
    PIN_B1  TD_DATA[1]
    PIN_E2  TD_DATA[2]
```

```
    PIN_B2  TD_DATA[3]
    PIN_D1  TD_DATA[4]
    PIN_E1  TD_DATA[5]
    PIN_C2  TD_DATA[6]
    PIN_B3  TD_DATA[7]
    PIN_A5  TD_HS
    PIN_F6  TD_RESET_N
    PIN_A3  TD_VS

    PIN_A13 VGA_R[0]
    PIN_C13 VGA_R[1]
    PIN_E13 VGA_R[2]
    PIN_B12 VGA_R[3]
    PIN_C12 VGA_R[4]
    PIN_D12 VGA_R[5]
    PIN_E12 VGA_R[6]
    PIN_F13 VGA_R[7]

    PIN_J9  VGA_G[0]
    PIN_J10 VGA_G[1]
    PIN_H12 VGA_G[2]
    PIN_G10 VGA_G[3]
    PIN_G11 VGA_G[4]
    PIN_G12 VGA_G[5]
    PIN_F11 VGA_G[6]
    PIN_E11 VGA_G[7]

    PIN_B13 VGA_B[0]
    PIN_G13 VGA_B[1]
    PIN_H13 VGA_B[2]
    PIN_F14 VGA_B[3]
    PIN_H14 VGA_B[4]
    PIN_F15 VGA_B[5]
    PIN_G15 VGA_B[6]
    PIN_J14 VGA_B[7]

    PIN_A11 VGA_CLK
    PIN_B11 VGA_HS
    PIN_D11 VGA_VS
    PIN_F10 VGA_BLANK_N
    PIN_C10 VGA_SYNC_N
} {
    set_location_assignment $pin -to $port
    set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -
       to $port
```

```
311 }
312
313 # HPS assignments
314
315 # 3.3-V LVTTL pins
316 foreach port {
317     HPS_CONV_USB_N
318     HPS_ENET_GTX_CLK
319     HPS_ENET_INT_N
320     HPS_ENET_MDC
321     HPS_ENET_MDIO
322     HPS_ENET_RX_CLK
323     HPS_ENET_RX_DATA[0]
324     HPS_ENET_RX_DATA[1]
325     HPS_ENET_RX_DATA[2]
326     HPS_ENET_RX_DATA[3]
327     HPS_ENET_RX_DV
328     HPS_ENET_TX_DATA[0]
329     HPS_ENET_TX_DATA[1]
330     HPS_ENET_TX_DATA[2]
331     HPS_ENET_TX_DATA[3]
332     HPS_ENET_TX_EN
333     HPS_GSENSOR_INT
334     HPS_I2C1_SCLK
335     HPS_I2C1_SDAT
336     HPS_I2C2_SCLK
337     HPS_I2C2_SDAT
338     HPS_I2C_CONTROL
339     HPS_KEY
340     HPS_LED
341     HPS_LTC_GPIO
342     HPS_SD_CLK
343     HPS_SD_CMD
344     HPS_SD_DATA[0]
345     HPS_SD_DATA[1]
346     HPS_SD_DATA[2]
347     HPS_SD_DATA[3]
348     HPS_SPIM_CLK
349     HPS_SPIM_MISO
350     HPS_SPIM_MOSI
351     HPS_SPIM_SS
352     HPS_UART_RX
353     HPS_UART_TX
354     HPS_USB_CLKOUT
355     HPS_USB_DATA[0]
```

```
356     HPS_USB_DATA[1]
357     HPS_USB_DATA[2]
358     HPS_USB_DATA[3]
359     HPS_USB_DATA[4]
360     HPS_USB_DATA[5]
361     HPS_USB_DATA[6]
362     HPS_USB_DATA[7]
363     HPS_USB_DIR
364     HPS_USB_NXT
365     HPS_USB_STP
366 } {
367     set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -
            to $port
368 }
369
370 # There are a lot of settings for the HPS_DDR3 interface not
        listed here.
371 # Instead, the
372 #
373 # soc_system/synthesis/submodules/
        hps_sdram_p0_pin_assignments.tcl
374 #
375 # script generated by qsys adds that information.  However,
        quartus_map
376 # must be run before this .tcl script may run because the
        script
377 # relies on being able to look at the (HPS) netlist to
        determine which
378 # pins to constrain
379
380 set sdcFilename "${project}.sdc"
381
382 set_global_assignment -name SDC_FILE $sdcFilename
383
384 set sdcf [open $sdcFilename "w"]
385 puts $sdcf {
386     foreach {clock port} {
387         clock_50_1 CLOCK_50
388         clock_50_2 CLOCK2_50
389         clock_50_3 CLOCK3_50
390         clock_50_4 CLOCK4_50
391     } {
392         create_clock -name $clock -period 20ns [get_ports
                $port]
393     }
```

```
394
395     create_clock -name clock_27_1 -period 37 [get_ports
            TD_CLK27]
396
397     derive_pll_clocks -create_base_clocks
398     derive_clock_uncertainty
399 }
400 close $sdcf
401
402 project_close
```

4.5 soc_system.qsys

```
<?xml version="1.0" encoding="UTF-8"?>
<system name="$${FILENAME}">
 <component
   name="$${FILENAME}"
   displayName="$${FILENAME}"
   version="1.0"
   description=""
   tags=""
   categories="System" />
 <parameter name="bonusData"><![CDATA[bonusData
{
   element clk_0
   {
      datum _sortIndex
      {
         value = "0";
         type = "int";
      }
   }
   element hps_0
   {
      datum _sortIndex
      {
         value = "1";
         type = "int";
      }
   }
   element soc_system
   {
      datum _originalDeviceFamily
      {
         value = "Cyclone V";
         type = "String";
      }
   }
   element soc_system
   {
      datum _originalDeviceFamily
      {
         value = "Cyclone V";
         type = "String";
      }
   }
```

```
   element soc_system
   {
      datum _originalDeviceFamily
      {
         value = "Cyclone V";
         type = "String";
      }
   }
   element soc_system
   {
      datum _originalDeviceFamily
      {
         value = "Cyclone V";
         type = "String";
      }
   }
   element soc_system
   {
      datum _originalDeviceFamily
      {
         value = "Cyclone V";
         type = "String";
      }
   }
   element soc_system
   {
      datum _originalDeviceFamily
      {
         value = "Cyclone V";
         type = "String";
      }
   }
   element vga_ball_0
   {
      datum _sortIndex
      {
         value = "2";
         type = "int";
      }
   }
}
]]></parameter>
 <parameter name="clockCrossingAdapter" value="HANDSHAKE" />
 <parameter name="device" value="5CSEMA5F31C6" />
 <parameter name="deviceFamily" value="Cyclone V" />
```

```
<parameter name="deviceSpeedGrade" value="6" />
<parameter name="fabricMode" value="QSYS" />
<parameter name="generateLegacySim" value="false" />
<parameter name="generationId" value="0" />
<parameter name="globalResetBus" value="false" />
<parameter name="hdlLanguage" value="VERILOG" />
<parameter name="hideFromIPCatalog" value="false" />
<parameter name="lockedInterfaceDefinition" value="" />
<parameter name="maxAdditionalLatency" value="1" />
<parameter name="projectName" value="" />
<parameter name="sopcBorderPoints" value="false" />
<parameter name="systemHash" value="0" />
<parameter name="testBenchDutName" value="" />
<parameter name="timeStamp" value="0" />
<parameter name="useTestBenchNamingPattern" value="false" />
<instanceScript></instanceScript>
<interface name="clk" internal="clk_0.clk_in" type="clock" dir="end" />
<interface name="hps" internal="hps_0.hps_io" type="conduit" dir="end" />
<interface name="hps_ddr3" internal="hps_0.memory" type="conduit" dir="end" />
<interface name="reset" internal="clk_0.clk_in_reset" type="reset" dir="end" />
<interface name="sound" internal="vga_ball_0.sound" type="conduit" dir="end" />
<interface name="vga" internal="vga_ball_0.vga" type="conduit" dir="end" />
<module name="clk_0" kind="clock_source" version="21.1" enabled="1">
 <parameter name="clockFrequency" value="50000000" />
 <parameter name="clockFrequencyKnown" value="true" />
 <parameter name="inputClockFrequency" value="0" />
 <parameter name="resetSynchronousEdges" value="NONE" />
</module>
<module name="hps_0" kind="altera_hps" version="21.1" enabled="1">
 <parameter name="ABSTRACT_REAL_COMPARE_TEST" value="false" />
 <parameter name="ABS_RAM_MEM_INIT_FILENAME" value="meminit" />
 <parameter name="ACV_PHY_CLK_ADD_FR_PHASE" value="0.0" />
 <parameter name="AC_PACKAGE_DESKEW" value="false" />
 <parameter name="AC_ROM_USER_ADD_0" value="0_0000_0000_0000" />
```

```
123   <parameter name="AC_ROM_USER_ADD_1" value="0_0000_0000_1000
          " />
124   <parameter name="ADDR_ORDER" value="0" />
125   <parameter name="ADD_EFFICIENCY_MONITOR" value="false" />
126   <parameter name="ADD_EXTERNAL_SEQ_DEBUG_NIOS" value="false"
          />
127   <parameter name="ADVANCED_CK_PHASES" value="false" />
128   <parameter name="ADVERTIZE_SEQUENCER_SW_BUILD_FILES" value
          ="false" />
129   <parameter name="AFI_DEBUG_INFO_WIDTH" value="32" />
130   <parameter name="ALTMEMPHY_COMPATIBLE_MODE" value="false"
          />
131   <parameter name="AP_MODE" value="false" />
132   <parameter name="AP_MODE_EN" value="0" />
133   <parameter name="AUTO_DEVICE_SPEEDGRADE" value="6" />
134   <parameter name="AUTO_PD_CYCLES" value="0" />
135   <parameter name="AUTO_POWERDN_EN" value="false" />
136   <parameter name="AVL_DATA_WIDTH_PORT" value
          ="32,32,32,32,32,32" />
137   <parameter name="AVL_MAX_SIZE" value="4" />
138   <parameter name="BONDING_OUT_ENABLED" value="false" />
139   <parameter name="BOOTFROMFPGA_Enable" value="false" />
140   <parameter name="BSEL" value="1" />
141   <parameter name="BSEL_EN" value="false" />
142   <parameter name="BYTE_ENABLE" value="true" />
143   <parameter name="C2P_WRITE_CLOCK_ADD_PHASE" value="0.0" />
144   <parameter name="CALIBRATION_MODE" value="Skip" />
145   <parameter name="CALIB_REG_WIDTH" value="8" />
146   <parameter name="CAN0_Mode" value="N/A" />
147   <parameter name="CAN0_PinMuxing" value="Unused" />
148   <parameter name="CAN1_Mode" value="N/A" />
149   <parameter name="CAN1_PinMuxing" value="Unused" />
150   <parameter name="CFG_DATA_REORDERING_TYPE" value="
          INTER_BANK" />
151   <parameter name="CFG_REORDER_DATA" value="true" />
152   <parameter name="CFG_TCCD_NS" value="2.5" />
153   <parameter name="COMMAND_PHASE" value="0.0" />
154   <parameter name="CONTROLLER_LATENCY" value="5" />
155   <parameter name="CORE_DEBUG_CONNECTION" value="EXPORT" />
156   <parameter name="CPORT_TYPE_PORT">Bidirectional,
          Bidirectional,Bidirectional,Bidirectional,Bidirectional,
          Bidirectional</parameter>
157   <parameter name="CSEL" value="0" />
158   <parameter name="CSEL_EN" value="false" />
159   <parameter name="CTI_Enable" value="false" />
```

```
<parameter name="CTL_AUTOPCH_EN" value="false" />
<parameter name="CTL_CMD_QUEUE_DEPTH" value="8" />
<parameter name="CTL_CSR_CONNECTION" value="INTERNAL_JTAG" />
<parameter name="CTL_CSR_ENABLED" value="false" />
<parameter name="CTL_CSR_READ_ONLY" value="1" />
<parameter name="CTL_DEEP_POWERDN_EN" value="false" />
<parameter name="CTL_DYNAMIC_BANK_ALLOCATION" value="false" />
<parameter name="CTL_DYNAMIC_BANK_NUM" value="4" />
<parameter name="CTL_ECC_AUTO_CORRECTION_ENABLED" value="false" />
<parameter name="CTL_ECC_ENABLED" value="false" />
<parameter name="CTL_ENABLE_BURST_INTERRUPT" value="false" />
<parameter name="CTL_ENABLE_BURST_TERMINATE" value="false" />
<parameter name="CTL_HRB_ENABLED" value="false" />
<parameter name="CTL_LOOK_AHEAD_DEPTH" value="4" />
<parameter name="CTL_SELF_REFRESH_EN" value="false" />
<parameter name="CTL_USR_REFRESH_EN" value="false" />
<parameter name="CTL_ZQCAL_EN" value="false" />
<parameter name="CUT_NEW_FAMILY_TIMING" value="true" />
<parameter name="DAT_DATA_WIDTH" value="32" />
<parameter name="DEBUGAPB_Enable" value="false" />
<parameter name="DEBUG_MODE" value="false" />
<parameter name="DEVICE_DEPTH" value="1" />
<parameter name="DEVICE_FAMILY_PARAM" value="" />
<parameter name="DISABLE_CHILD_MESSAGING" value="false" />
<parameter name="DISCRETE_FLY_BY" value="true" />
<parameter name="DLL_SHARING_MODE" value="None" />
<parameter name="DMA_Enable">No,No,No,No,No,No,No,No</parameter>
<parameter name="DQS_DQSN_MODE" value="DIFFERENTIAL" />
<parameter name="DQ_INPUT_REG_USE_CLKN" value="false" />
<parameter name="DUPLICATE_AC" value="false" />
<parameter name="ED_EXPORT_SEQ_DEBUG" value="false" />
<parameter name="EMAC0_Mode" value="N/A" />
<parameter name="EMAC0_PTP" value="false" />
<parameter name="EMAC0_PinMuxing" value="Unused" />
<parameter name="EMAC1_Mode" value="RGMII" />
<parameter name="EMAC1_PTP" value="false" />
<parameter name="EMAC1_PinMuxing" value="HPS I/O Set 0" />
<parameter name="ENABLE_ABS_RAM_MEM_INIT" value="false" />
<parameter name="ENABLE_BONDING" value="false" />
```

```
199   <parameter name="ENABLE_BURST_MERGE" value="false" />
200   <parameter name="ENABLE_CTRL_AVALON_INTERFACE" value="true"
          />
201   <parameter name="ENABLE_DELAY_CHAIN_WRITE" value="false" />
202   <parameter name="ENABLE_EMIT_BFM_MASTER" value="false" />
203   <parameter name="ENABLE_EXPORT_SEQ_DEBUG_BRIDGE" value="
          false" />
204   <parameter name="ENABLE_EXTRA_REPORTING" value="false" />
205   <parameter name="ENABLE_ISS_PROBES" value="false" />
206   <parameter name="ENABLE_NON_DESTRUCTIVE_CALIB" value="false
          " />
207   <parameter name="ENABLE_NON_DES_CAL" value="false" />
208   <parameter name="ENABLE_NON_DES_CAL_TEST" value="false" />
209   <parameter name="ENABLE_SEQUENCER_MARGINING_ON_BY_DEFAULT"
          value="false" />
210   <parameter name="ENABLE_USER_ECC" value="false" />
211   <parameter name="EXPORT_AFI_HALF_CLK" value="false" />
212   <parameter name="EXTRA_SETTINGS" value="" />
213   <parameter name="F2H_AXI_CLOCK_FREQ" value="50000000" />
214   <parameter name="F2H_SDRAM0_CLOCK_FREQ" value="100" />
215   <parameter name="F2H_SDRAM1_CLOCK_FREQ" value="100" />
216   <parameter name="F2H_SDRAM2_CLOCK_FREQ" value="100" />
217   <parameter name="F2H_SDRAM3_CLOCK_FREQ" value="100" />
218   <parameter name="F2H_SDRAM4_CLOCK_FREQ" value="100" />
219   <parameter name="F2H_SDRAM5_CLOCK_FREQ" value="100" />
220   <parameter name="F2SCLK_COLDRST_Enable" value="false" />
221   <parameter name="F2SCLK_DBGRST_Enable" value="false" />
222   <parameter name="F2SCLK_PERIPHCLK_Enable" value="false" />
223   <parameter name="F2SCLK_PERIPHCLK_FREQ" value="0" />
224   <parameter name="F2SCLK_SDRAMCLK_Enable" value="false" />
225   <parameter name="F2SCLK_SDRAMCLK_FREQ" value="0" />
226   <parameter name="F2SCLK_WARMRST_Enable" value="false" />
227   <parameter name="F2SDRAM_Type" value="" />
228   <parameter name="F2SDRAM_Width" value="" />
229   <parameter name="F2SINTERRUPT_Enable" value="false" />
230   <parameter name="F2S_Width" value="2" />
231   <parameter name="FIX_READ_LATENCY" value="8" />
232   <parameter name="FORCED_NON_LDC_ADDR_CMD_MEM_CK_INVERT"
          value="false" />
233   <parameter name="FORCED_NUM_WRITE_FR_CYCLE_SHIFTS" value
          ="0" />
234   <parameter name="FORCE_DQS_TRACKING" value="AUTO" />
235   <parameter name="FORCE_MAX_LATENCY_COUNT_WIDTH" value="0"
          />
```

```
236   <parameter name="FORCE_SEQUENCER_TCL_DEBUG_MODE" value="
         false" />
237   <parameter name="FORCE_SHADOW_REGS" value="AUTO" />
238   <parameter name="FORCE_SYNTHESIS_LANGUAGE" value="" />
239   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC0_RX_CLK_IN" value
         ="100" />
240   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC0_TX_CLK_IN" value
         ="100" />
241   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC1_RX_CLK_IN" value
         ="100" />
242   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC1_TX_CLK_IN" value
         ="100" />
243   <parameter
244      name="
            FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC_PTP_REF_CLOCK"
245      value="100" />
246   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C0_SCL_IN" value
         ="100" />
247   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C1_SCL_IN" value
         ="100" />
248   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C2_SCL_IN" value
         ="100" />
249   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C3_SCL_IN" value
         ="100" />
250   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_SPIS0_SCLK_IN" value
         ="100" />
251   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_SPIS1_SCLK_IN" value
         ="100" />
252   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_USB0_CLK_IN" value
         ="100" />
253   <parameter name="
         FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_USB1_CLK_IN" value
         ="100" />
```

```
254   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC0_GTX_CLK" value
         ="125" />
255   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC0_MD_CLK" value
         ="2.5" />
256   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC1_GTX_CLK" value
         ="125" />
257   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC1_MD_CLK" value
         ="2.5" />
258   <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C0_CLK
         " value="100" />
259   <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C1_CLK
         " value="100" />
260   <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C2_CLK
         " value="100" />
261   <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C3_CLK
         " value="100" />
262   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_QSPI_SCLK_OUT" value
         ="100" />
263   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_SDIO_CCLK" value="100"
          />
264   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_SPIM0_SCLK_OUT" value
         ="100" />
265   <parameter name="
         FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_SPIM1_SCLK_OUT" value
         ="100" />
266   <parameter name="GPIO_Enable">No,No,No,No,No,No,No,No,No,
         Yes,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
         No,No,No,No,No,No,No,No,Yes,No,No,No,No,Yes,No,No,No,No,
         No,No,No,Yes,No,No,No,No,Yes,Yes,No,No,No,No,No,No,Yes,
         No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No
         ,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
         No</parameter>
267   <parameter name="GP_Enable" value="false" />
268   <parameter name="H2F_AXI_CLOCK_FREQ" value="50000000" />
269   <parameter name="H2F_CTI_CLOCK_FREQ" value="100" />
270   <parameter name="H2F_DEBUG_APB_CLOCK_FREQ" value="100" />
271   <parameter name="H2F_LW_AXI_CLOCK_FREQ" value="50000000" />
272   <parameter name="H2F_TPIU_CLOCK_IN_FREQ" value="100" />
```

```
<parameter name="HARD_EMIF" value="true" />
<parameter name="HCX_COMPAT_MODE" value="false" />
<parameter name="HHP_HPS" value="true" />
<parameter name="HHP_HPS_SIMULATION" value="false" />
<parameter name="HHP_HPS_VERIFICATION" value="false" />
<parameter name="HLGPI_Enable" value="false" />
<parameter name="HPS_PROTOCOL" value="DDR3" />
<parameter name="I2C0_Mode" value="I2C" />
<parameter name="I2C0_PinMuxing" value="HPS I/O Set 0" />
<parameter name="I2C1_Mode" value="I2C" />
<parameter name="I2C1_PinMuxing" value="HPS I/O Set 0" />
<parameter name="I2C2_Mode" value="N/A" />
<parameter name="I2C2_PinMuxing" value="Unused" />
<parameter name="I2C3_Mode" value="N/A" />
<parameter name="I2C3_PinMuxing" value="Unused" />
<parameter name="INCLUDE_BOARD_DELAY_MODEL" value="false"
   />
<parameter name="INCLUDE_MULTIRANK_BOARD_DELAY_MODEL" value
   ="false" />
<parameter name="IS_ES_DEVICE" value="false" />
<parameter name="LOANIO_Enable">No,No,No,No,No,No,No,No,No,
   No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No
   ,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
   No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No
   ,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
   No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No</
   parameter>
<parameter name="LOCAL_ID_WIDTH" value="8" />
<parameter name="LRDIMM_EXTENDED_CONFIG">0
   x0000000000000000</parameter>
<parameter name="LWH2F_Enable" value="true" />
<parameter name="MARGIN_VARIATION_TEST" value="false" />
<parameter name="MAX_PENDING_RD_CMD" value="32" />
<parameter name="MAX_PENDING_WR_CMD" value="16" />
<parameter name="MEM_ASR" value="Manual" />
<parameter name="MEM_ATCL" value="Disabled" />
<parameter name="MEM_AUTO_LEVELING_MODE" value="true" />
<parameter name="MEM_BANKADDR_WIDTH" value="3" />
<parameter name="MEM_BL" value="OTF" />
<parameter name="MEM_BT" value="Sequential" />
<parameter name="MEM_CK_PHASE" value="0.0" />
<parameter name="MEM_CK_WIDTH" value="1" />
<parameter name="MEM_CLK_EN_WIDTH" value="1" />
<parameter name="MEM_CLK_FREQ" value="400.0" />
<parameter name="MEM_CLK_FREQ_MAX" value="800.0" />
```

```
  <parameter name="MEM_COL_ADDR_WIDTH" value="10" />
  <parameter name="MEM_CS_WIDTH" value="1" />
  <parameter name="MEM_DEVICE" value="MISSING_MODEL" />
  <parameter name="MEM_DLL_EN" value="true" />
  <parameter name="MEM_DQ_PER_DQS" value="8" />
  <parameter name="MEM_DQ_WIDTH" value="32" />
  <parameter name="MEM_DRV_STR" value="RZQ/7" />
  <parameter name="MEM_FORMAT" value="DISCRETE" />
  <parameter name="MEM_GUARANTEED_WRITE_INIT" value="false"
     />
  <parameter name="MEM_IF_BOARD_BASE_DELAY" value="10" />
  <parameter name="MEM_IF_DM_PINS_EN" value="true" />
  <parameter name="MEM_IF_DQSN_EN" value="true" />
  <parameter name="MEM_IF_SIM_VALID_WINDOW" value="0" />
  <parameter name="MEM_INIT_EN" value="false" />
  <parameter name="MEM_INIT_FILE" value="" />
  <parameter name="MEM_MIRROR_ADDRESSING" value="0" />
  <parameter name="MEM_NUMBER_OF_DIMMS" value="1" />
  <parameter name="MEM_NUMBER_OF_RANKS_PER_DEVICE" value="1"
     />
  <parameter name="MEM_NUMBER_OF_RANKS_PER_DIMM" value="1" />
  <parameter name="MEM_PD" value="DLL off" />
  <parameter name="MEM_RANK_MULTIPLICATION_FACTOR" value="1"
     />
  <parameter name="MEM_ROW_ADDR_WIDTH" value="15" />
  <parameter name="MEM_RTT_NOM" value="RZQ/4" />
  <parameter name="MEM_RTT_WR" value="RZQ/4" />
  <parameter name="MEM_SRT" value="Normal" />
  <parameter name="MEM_TCL" value="11" />
  <parameter name="MEM_TFAW_NS" value="30.0" />
  <parameter name="MEM_TINIT_US" value="500" />
  <parameter name="MEM_TMRD_CK" value="4" />
  <parameter name="MEM_TRAS_NS" value="35.0" />
  <parameter name="MEM_TRCD_NS" value="13.75" />
  <parameter name="MEM_TREFI_US" value="7.8" />
  <parameter name="MEM_TRFC_NS" value="260.0" />
  <parameter name="MEM_TRP_NS" value="13.75" />
  <parameter name="MEM_TRRD_NS" value="7.7" />
  <parameter name="MEM_TRTP_NS" value="7.5" />
  <parameter name="MEM_TWR_NS" value="15.0" />
  <parameter name="MEM_TWTR" value="4" />
  <parameter name="MEM_USER_LEVELING_MODE" value="Leveling"
     />
  <parameter name="MEM_VENDOR" value="JEDEC" />
  <parameter name="MEM_VERBOSE" value="true" />
```

```
350   <parameter name="MEM_VOLTAGE" value="1.5V DDR3" />
351   <parameter name="MEM_WTCL" value="8" />
352   <parameter name="MPU_EVENTS_Enable" value="false" />
353   <parameter name="MRS_MIRROR_PING_PONG_ATSO" value="false"
        />
354   <parameter name="MULTICAST_EN" value="false" />
355   <parameter name="NAND_Mode" value="N/A" />
356   <parameter name="NAND_PinMuxing" value="Unused" />
357   <parameter name="NEXTGEN" value="true" />
358   <parameter name="NIOS_ROM_DATA_WIDTH" value="32" />
359   <parameter name="NUM_DLL_SHARING_INTERFACES" value="1" />
360   <parameter name="NUM_EXTRA_REPORT_PATH" value="10" />
361   <parameter name="NUM_OCT_SHARING_INTERFACES" value="1" />
362   <parameter name="NUM_OF_PORTS" value="1" />
363   <parameter name="NUM_PLL_SHARING_INTERFACES" value="1" />
364   <parameter name="OCT_SHARING_MODE" value="None" />
365   <parameter name="P2C_READ_CLOCK_ADD_PHASE" value="0.0" />
366   <parameter name="PACKAGE_DESKEW" value="false" />
367   <parameter name="PARSE_FRIENDLY_DEVICE_FAMILY_PARAM" value
        ="" />
368   <parameter name="PARSE_FRIENDLY_DEVICE_FAMILY_PARAM_VALID"
        value="false" />
369   <parameter name="PHY_CSR_CONNECTION" value="INTERNAL_JTAG"
        />
370   <parameter name="PHY_CSR_ENABLED" value="false" />
371   <parameter name="PHY_ONLY" value="false" />
372   <parameter name="PINGPONGPHY_EN" value="false" />
373   <parameter name="PLL_ADDR_CMD_CLK_DIV_PARAM" value="0" />
374   <parameter name="PLL_ADDR_CMD_CLK_FREQ_PARAM" value="0.0"
        />
375   <parameter name="PLL_ADDR_CMD_CLK_FREQ_SIM_STR_PARAM" value
        ="" />
376   <parameter name="PLL_ADDR_CMD_CLK_MULT_PARAM" value="0" />
377   <parameter name="PLL_ADDR_CMD_CLK_PHASE_PS_PARAM" value="0"
         />
378   <parameter name="PLL_ADDR_CMD_CLK_PHASE_PS_SIM_STR_PARAM"
        value="" />
379   <parameter name="PLL_AFI_CLK_DIV_PARAM" value="0" />
380   <parameter name="PLL_AFI_CLK_FREQ_PARAM" value="0.0" />
381   <parameter name="PLL_AFI_CLK_FREQ_SIM_STR_PARAM" value=""
        />
382   <parameter name="PLL_AFI_CLK_MULT_PARAM" value="0" />
383   <parameter name="PLL_AFI_CLK_PHASE_PS_PARAM" value="0" />
384   <parameter name="PLL_AFI_CLK_PHASE_PS_SIM_STR_PARAM" value
        ="" />
```

```
385   <parameter name="PLL_AFI_HALF_CLK_DIV_PARAM" value="0" />
386   <parameter name="PLL_AFI_HALF_CLK_FREQ_PARAM" value="0.0"
         />
387   <parameter name="PLL_AFI_HALF_CLK_FREQ_SIM_STR_PARAM" value
         ="" />
388   <parameter name="PLL_AFI_HALF_CLK_MULT_PARAM" value="0" />
389   <parameter name="PLL_AFI_HALF_CLK_PHASE_PS_PARAM" value="0"
          />
390   <parameter name="PLL_AFI_HALF_CLK_PHASE_PS_SIM_STR_PARAM"
         value="" />
391   <parameter name="PLL_AFI_PHY_CLK_DIV_PARAM" value="0" />
392   <parameter name="PLL_AFI_PHY_CLK_FREQ_PARAM" value="0.0" />
393   <parameter name="PLL_AFI_PHY_CLK_FREQ_SIM_STR_PARAM" value
         ="" />
394   <parameter name="PLL_AFI_PHY_CLK_MULT_PARAM" value="0" />
395   <parameter name="PLL_AFI_PHY_CLK_PHASE_PS_PARAM" value="0"
         />
396   <parameter name="PLL_AFI_PHY_CLK_PHASE_PS_SIM_STR_PARAM"
         value="" />
397   <parameter name="PLL_C2P_WRITE_CLK_DIV_PARAM" value="0" />
398   <parameter name="PLL_C2P_WRITE_CLK_FREQ_PARAM" value="0.0"
         />
399   <parameter name="PLL_C2P_WRITE_CLK_FREQ_SIM_STR_PARAM"
         value="" />
400   <parameter name="PLL_C2P_WRITE_CLK_MULT_PARAM" value="0" />
401   <parameter name="PLL_C2P_WRITE_CLK_PHASE_PS_PARAM" value
         ="0" />
402   <parameter name="PLL_C2P_WRITE_CLK_PHASE_PS_SIM_STR_PARAM"
         value="" />
403   <parameter name="PLL_CLK_PARAM_VALID" value="false" />
404   <parameter name="PLL_CONFIG_CLK_DIV_PARAM" value="0" />
405   <parameter name="PLL_CONFIG_CLK_FREQ_PARAM" value="0.0" />
406   <parameter name="PLL_CONFIG_CLK_FREQ_SIM_STR_PARAM" value
         ="" />
407   <parameter name="PLL_CONFIG_CLK_MULT_PARAM" value="0" />
408   <parameter name="PLL_CONFIG_CLK_PHASE_PS_PARAM" value="0"
         />
409   <parameter name="PLL_CONFIG_CLK_PHASE_PS_SIM_STR_PARAM"
         value="" />
410   <parameter name="PLL_DR_CLK_DIV_PARAM" value="0" />
411   <parameter name="PLL_DR_CLK_FREQ_PARAM" value="0.0" />
412   <parameter name="PLL_DR_CLK_FREQ_SIM_STR_PARAM" value="" />
413   <parameter name="PLL_DR_CLK_MULT_PARAM" value="0" />
414   <parameter name="PLL_DR_CLK_PHASE_PS_PARAM" value="0" />
```

```
415   <parameter name="PLL_DR_CLK_PHASE_PS_SIM_STR_PARAM" value
          ="" />
416   <parameter name="PLL_HR_CLK_DIV_PARAM" value="0" />
417   <parameter name="PLL_HR_CLK_FREQ_PARAM" value="0.0" />
418   <parameter name="PLL_HR_CLK_FREQ_SIM_STR_PARAM" value="" />
419   <parameter name="PLL_HR_CLK_MULT_PARAM" value="0" />
420   <parameter name="PLL_HR_CLK_PHASE_PS_PARAM" value="0" />
421   <parameter name="PLL_HR_CLK_PHASE_PS_SIM_STR_PARAM" value
          ="" />
422   <parameter name="PLL_LOCATION" value="Top_Bottom" />
423   <parameter name="PLL_MEM_CLK_DIV_PARAM" value="0" />
424   <parameter name="PLL_MEM_CLK_FREQ_PARAM" value="0.0" />
425   <parameter name="PLL_MEM_CLK_FREQ_SIM_STR_PARAM" value=""
          />
426   <parameter name="PLL_MEM_CLK_MULT_PARAM" value="0" />
427   <parameter name="PLL_MEM_CLK_PHASE_PS_PARAM" value="0" />
428   <parameter name="PLL_MEM_CLK_PHASE_PS_SIM_STR_PARAM" value
          ="" />
429   <parameter name="PLL_NIOS_CLK_DIV_PARAM" value="0" />
430   <parameter name="PLL_NIOS_CLK_FREQ_PARAM" value="0.0" />
431   <parameter name="PLL_NIOS_CLK_FREQ_SIM_STR_PARAM" value=""
          />
432   <parameter name="PLL_NIOS_CLK_MULT_PARAM" value="0" />
433   <parameter name="PLL_NIOS_CLK_PHASE_PS_PARAM" value="0" />
434   <parameter name="PLL_NIOS_CLK_PHASE_PS_SIM_STR_PARAM" value
          ="" />
435   <parameter name="PLL_P2C_READ_CLK_DIV_PARAM" value="0" />
436   <parameter name="PLL_P2C_READ_CLK_FREQ_PARAM" value="0.0"
          />
437   <parameter name="PLL_P2C_READ_CLK_FREQ_SIM_STR_PARAM" value
          ="" />
438   <parameter name="PLL_P2C_READ_CLK_MULT_PARAM" value="0" />
439   <parameter name="PLL_P2C_READ_CLK_PHASE_PS_PARAM" value="0"
          />
440   <parameter name="PLL_P2C_READ_CLK_PHASE_PS_SIM_STR_PARAM"
          value="" />
441   <parameter name="PLL_SHARING_MODE" value="None" />
442   <parameter name="PLL_WRITE_CLK_DIV_PARAM" value="0" />
443   <parameter name="PLL_WRITE_CLK_FREQ_PARAM" value="0.0" />
444   <parameter name="PLL_WRITE_CLK_FREQ_SIM_STR_PARAM" value=""
          />
445   <parameter name="PLL_WRITE_CLK_MULT_PARAM" value="0" />
446   <parameter name="PLL_WRITE_CLK_PHASE_PS_PARAM" value="0" />
447   <parameter name="PLL_WRITE_CLK_PHASE_PS_SIM_STR_PARAM"
          value="" />
```

```
448   <parameter name="POWER_OF_TWO_BUS" value="false" />
449   <parameter name="PRIORITY_PORT" value="1,1,1,1,1,1" />
450   <parameter name="QSPI_Mode" value="N/A" />
451   <parameter name="QSPI_PinMuxing" value="Unused" />
452   <parameter name="RATE" value="Full" />
453   <parameter name="RDIMM_CONFIG" value="0000000000000000" />
454   <parameter name="READ_DQ_DQS_CLOCK_SOURCE" value="
         INVERTED_DQS_BUS" />
455   <parameter name="READ_FIFO_SIZE" value="8" />
456   <parameter name="REFRESH_BURST_VALIDATION" value="false" />
457   <parameter name="REFRESH_INTERVAL" value="15000" />
458   <parameter name="REF_CLK_FREQ" value="25.0" />
459   <parameter name="REF_CLK_FREQ_MAX_PARAM" value="0.0" />
460   <parameter name="REF_CLK_FREQ_MIN_PARAM" value="0.0" />
461   <parameter name="REF_CLK_FREQ_PARAM_VALID" value="false" />
462   <parameter name="S2FCLK_COLDRST_Enable" value="false" />
463   <parameter name="S2FCLK_PENDINGRST_Enable" value="false" />
464   <parameter name="S2FCLK_USER0CLK_Enable" value="false" />
465   <parameter name="S2FCLK_USER1CLK_Enable" value="true" />
466   <parameter name="S2FCLK_USER1CLK_FREQ" value="100.0" />
467   <parameter name="S2FCLK_USER2CLK" value="5" />
468   <parameter name="S2FCLK_USER2CLK_Enable" value="false" />
469   <parameter name="S2FCLK_USER2CLK_FREQ" value="100.0" />
470   <parameter name="S2FINTERRUPT_CAN_Enable" value="false" />
471   <parameter name="S2FINTERRUPT_CLOCKPERIPHERAL_Enable" value
         ="false" />
472   <parameter name="S2FINTERRUPT_CTI_Enable" value="false" />
473   <parameter name="S2FINTERRUPT_DMA_Enable" value="false" />
474   <parameter name="S2FINTERRUPT_EMAC_Enable" value="false" />
475   <parameter name="S2FINTERRUPT_FPGAMANAGER_Enable" value="
         false" />
476   <parameter name="S2FINTERRUPT_GPIO_Enable" value="false" />
477   <parameter name="S2FINTERRUPT_I2CEMAC_Enable" value="false"
          />
478   <parameter name="S2FINTERRUPT_I2CPERIPHERAL_Enable" value="
         false" />
479   <parameter name="S2FINTERRUPT_L4TIMER_Enable" value="false"
          />
480   <parameter name="S2FINTERRUPT_NAND_Enable" value="false" />
481   <parameter name="S2FINTERRUPT_OSCTIMER_Enable" value="false
         " />
482   <parameter name="S2FINTERRUPT_QSPI_Enable" value="false" />
483   <parameter name="S2FINTERRUPT_SDMMC_Enable" value="false"
         />
```

```
484   <parameter name="S2FINTERRUPT_SPIMASTER_Enable" value="
         false" />
485   <parameter name="S2FINTERRUPT_SPISLAVE_Enable" value="false
         " />
486   <parameter name="S2FINTERRUPT_UART_Enable" value="false" />
487   <parameter name="S2FINTERRUPT_USB_Enable" value="false" />
488   <parameter name="S2FINTERRUPT_WATCHDOG_Enable" value="false
         " />
489   <parameter name="S2F_Width" value="2" />
490   <parameter name="SDIO_Mode" value="4-bit Data" />
491   <parameter name="SDIO_PinMuxing" value="HPS I/O Set 0" />
492   <parameter name="SEQUENCER_TYPE" value="NIOS" />
493   <parameter name="SEQ_MODE" value="0" />
494   <parameter name="SKIP_MEM_INIT" value="true" />
495   <parameter name="SOPC_COMPAT_RESET" value="false" />
496   <parameter name="SPEED_GRADE" value="7" />
497   <parameter name="SPIM0_Mode" value="N/A" />
498   <parameter name="SPIM0_PinMuxing" value="Unused" />
499   <parameter name="SPIM1_Mode" value="Single Slave Select" />
500   <parameter name="SPIM1_PinMuxing" value="HPS I/O Set 0" />
501   <parameter name="SPIS0_Mode" value="N/A" />
502   <parameter name="SPIS0_PinMuxing" value="Unused" />
503   <parameter name="SPIS1_Mode" value="N/A" />
504   <parameter name="SPIS1_PinMuxing" value="Unused" />
505   <parameter name="STARVE_LIMIT" value="10" />
506   <parameter name="STM_Enable" value="false" />
507   <parameter name="SYS_INFO_DEVICE_FAMILY" value="Cyclone V"
         />
508   <parameter name="TEST_Enable" value="false" />
509   <parameter name="TIMING_BOARD_AC_EYE_REDUCTION_H" value
         ="0.0" />
510   <parameter name="TIMING_BOARD_AC_EYE_REDUCTION_SU" value
         ="0.0" />
511   <parameter name="TIMING_BOARD_AC_SKEW" value="0.03" />
512   <parameter name="TIMING_BOARD_AC_SLEW_RATE" value="1.0" />
513   <parameter name="TIMING_BOARD_AC_TO_CK_SKEW" value="0.0" />
514   <parameter name="TIMING_BOARD_CK_CKN_SLEW_RATE" value="2.0"
          />
515   <parameter name="TIMING_BOARD_DELTA_DQS_ARRIVAL_TIME" value
         ="0.0" />
516   <parameter name="TIMING_BOARD_DELTA_READ_DQS_ARRIVAL_TIME"
         value="0.0" />
517   <parameter name="TIMING_BOARD_DERATE_METHOD" value="AUTO"
         />
```

```
518   <parameter name="TIMING_BOARD_DQS_DQSN_SLEW_RATE" value
         ="2.0" />
519   <parameter name="TIMING_BOARD_DQ_EYE_REDUCTION" value="0.0"
          />
520   <parameter name="TIMING_BOARD_DQ_SLEW_RATE" value="1.0" />
521   <parameter name="TIMING_BOARD_DQ_TO_DQS_SKEW" value="0.0"
         />
522   <parameter name="TIMING_BOARD_ISI_METHOD" value="AUTO" />
523   <parameter name="TIMING_BOARD_MAX_CK_DELAY" value="0.03" />
524   <parameter name="TIMING_BOARD_MAX_DQS_DELAY" value="0.02"
         />
525   <parameter name="TIMING_BOARD_READ_DQ_EYE_REDUCTION" value
         ="0.0" />
526   <parameter name="TIMING_BOARD_SKEW_BETWEEN_DIMMS" value
         ="0.05" />
527   <parameter name="TIMING_BOARD_SKEW_BETWEEN_DQS" value
         ="0.08" />
528   <parameter name="TIMING_BOARD_SKEW_CKDQS_DIMM_MAX" value
         ="0.16" />
529   <parameter name="TIMING_BOARD_SKEW_CKDQS_DIMM_MIN" value
         ="0.09" />
530   <parameter name="TIMING_BOARD_SKEW_WITHIN_DQS" value="0.01"
          />
531   <parameter name="TIMING_BOARD_TDH" value="0.0" />
532   <parameter name="TIMING_BOARD_TDS" value="0.0" />
533   <parameter name="TIMING_BOARD_TIH" value="0.0" />
534   <parameter name="TIMING_BOARD_TIS" value="0.0" />
535   <parameter name="TIMING_TDH" value="65" />
536   <parameter name="TIMING_TDQSCK" value="255" />
537   <parameter name="TIMING_TDQSCKDL" value="1200" />
538   <parameter name="TIMING_TDQSCKDM" value="900" />
539   <parameter name="TIMING_TDQSCKDS" value="450" />
540   <parameter name="TIMING_TDQSH" value="0.35" />
541   <parameter name="TIMING_TDQSQ" value="125" />
542   <parameter name="TIMING_TDQSS" value="0.25" />
543   <parameter name="TIMING_TDS" value="30" />
544   <parameter name="TIMING_TDSH" value="0.2" />
545   <parameter name="TIMING_TDSS" value="0.2" />
546   <parameter name="TIMING_TIH" value="140" />
547   <parameter name="TIMING_TIS" value="180" />
548   <parameter name="TIMING_TQH" value="0.38" />
549   <parameter name="TIMING_TQHS" value="300" />
550   <parameter name="TIMING_TQSH" value="0.4" />
551   <parameter name="TPIUFPGA_Enable" value="false" />
552   <parameter name="TPIUFPGA_alt" value="false" />
```

```
553   <parameter name="TRACE_Mode" value="N/A" />
554   <parameter name="TRACE_PinMuxing" value="Unused" />
555   <parameter name="TRACKING_ERROR_TEST" value="false" />
556   <parameter name="TRACKING_WATCH_TEST" value="false" />
557   <parameter name="TREFI" value="35100" />
558   <parameter name="TRFC" value="350" />
559   <parameter name="UART0_Mode" value="No Flow Control" />
560   <parameter name="UART0_PinMuxing" value="HPS I/O Set 0" />
561   <parameter name="UART1_Mode" value="N/A" />
562   <parameter name="UART1_PinMuxing" value="Unused" />
563   <parameter name="USB0_Mode" value="N/A" />
564   <parameter name="USB0_PinMuxing" value="Unused" />
565   <parameter name="USB1_Mode" value="SDR" />
566   <parameter name="USB1_PinMuxing" value="HPS I/O Set 0" />
567   <parameter name="USER_DEBUG_LEVEL" value="1" />
568   <parameter name="USE_AXI_ADAPTOR" value="false" />
569   <parameter name="USE_FAKE_PHY" value="false" />
570   <parameter name="USE_MEM_CLK_FREQ" value="false" />
571   <parameter name="USE_MM_ADAPTOR" value="true" />
572   <parameter name="USE_SEQUENCER_BFM" value="false" />
573   <parameter name="WEIGHT_PORT" value="0,0,0,0,0,0" />
574   <parameter name="WRBUFFER_ADDR_WIDTH" value="6" />
575   <parameter name="can0_clk_div" value="1" />
576   <parameter name="can1_clk_div" value="1" />
577   <parameter name="configure_advanced_parameters" value="
        false" />
578   <parameter name="customize_device_pll_info" value="false"
        />
579   <parameter name="dbctrl_stayosc1" value="true" />
580   <parameter name="dbg_at_clk_div" value="0" />
581   <parameter name="dbg_clk_div" value="1" />
582   <parameter name="dbg_trace_clk_div" value="0" />
583   <parameter name="desired_can0_clk_mhz" value="100.0" />
584   <parameter name="desired_can1_clk_mhz" value="100.0" />
585   <parameter name="desired_cfg_clk_mhz" value="97.368421" />
586   <parameter name="desired_emac0_clk_mhz" value="250.0" />
587   <parameter name="desired_emac1_clk_mhz" value="250.0" />
588   <parameter name="desired_gpio_db_clk_hz" value="32000" />
589   <parameter name="desired_l4_mp_clk_mhz" value="100.0" />
590   <parameter name="desired_l4_sp_clk_mhz" value="100.0" />
591   <parameter name="desired_mpu_clk_mhz" value="800.0" />
592   <parameter name="desired_nand_clk_mhz" value="12.5" />
593   <parameter name="desired_qspi_clk_mhz" value="400.0" />
594   <parameter name="desired_sdmmc_clk_mhz" value="200.0" />
595   <parameter name="desired_spi_m_clk_mhz" value="200.0" />
```

```
596   <parameter name="desired_usb_mp_clk_mhz" value="200.0" />
597   <parameter name="device_name" value="5CSEMA5F31C6" />
598   <parameter name="device_pll_info_manual">{320000000
         1600000000} {320000000 1000000000} {800000000 400000000
         400000000}</parameter>
599   <parameter name="eosc1_clk_mhz" value="25.0" />
600   <parameter name="eosc2_clk_mhz" value="25.0" />
601   <parameter name="gpio_db_clk_div" value="6249" />
602   <parameter name="l3_mp_clk_div" value="1" />
603   <parameter name="l3_sp_clk_div" value="1" />
604   <parameter name="l4_mp_clk_div" value="1" />
605   <parameter name="l4_mp_clk_source" value="1" />
606   <parameter name="l4_sp_clk_div" value="1" />
607   <parameter name="l4_sp_clk_source" value="1" />
608   <parameter name="main_pll_c3" value="3" />
609   <parameter name="main_pll_c4" value="3" />
610   <parameter name="main_pll_c5" value="15" />
611   <parameter name="main_pll_m" value="63" />
612   <parameter name="main_pll_n" value="0" />
613   <parameter name="nand_clk_source" value="2" />
614   <parameter name="periph_pll_c0" value="3" />
615   <parameter name="periph_pll_c1" value="3" />
616   <parameter name="periph_pll_c2" value="1" />
617   <parameter name="periph_pll_c3" value="19" />
618   <parameter name="periph_pll_c4" value="4" />
619   <parameter name="periph_pll_c5" value="9" />
620   <parameter name="periph_pll_m" value="79" />
621   <parameter name="periph_pll_n" value="1" />
622   <parameter name="periph_pll_source" value="0" />
623   <parameter name="qspi_clk_source" value="1" />
624   <parameter name="quartus_ini_hps_emif_pll" value="false" />
625   <parameter
626      name="
         quartus_ini_hps_ip_enable_all_peripheral_fpga_interfaces
         "
627      value="false" />
628   <parameter name="quartus_ini_hps_ip_enable_bsel_csel" value
         ="false" />
629   <parameter
630      name="
         quartus_ini_hps_ip_enable_emac0_peripheral_fpga_interface
         "
631      value="false" />
632   <parameter
```

```
633       name="
              quartus_ini_hps_ip_enable_low_speed_serial_fpga_interfaces
              "
634       value="false" />
635    <parameter name="quartus_ini_hps_ip_enable_test_interface"
          value="false" />
636    <parameter name="quartus_ini_hps_ip_f2sdram_bonding_out"
          value="false" />
637    <parameter name="quartus_ini_hps_ip_fast_f2sdram_sim_model"
           value="false" />
638    <parameter name="quartus_ini_hps_ip_suppress_sdram_synth"
          value="false" />
639    <parameter name="sdmmc_clk_source" value="2" />
640    <parameter name="show_advanced_parameters" value="false" />
641    <parameter name="show_debug_info_as_warning_msg" value="
          false" />
642    <parameter name="show_warning_as_error_msg" value="false"
          />
643    <parameter name="spi_m_clk_div" value="0" />
644    <parameter name="usb_mp_clk_div" value="0" />
645    <parameter name="use_default_mpu_clk" value="true" />
646  </module>
647  <module name="vga_ball_0" kind="vga_ball" version="1.0"
       enabled="1" />
648  <connection
649    kind="avalon"
650    version="21.1"
651    start="hps_0.h2f_lw_axi_master"
652    end="vga_ball_0.avalon_slave_0">
653    <parameter name="arbitrationPriority" value="1" />
654    <parameter name="baseAddress" value="0x0000" />
655    <parameter name="defaultConnection" value="false" />
656  </connection>
657  <connection kind="clock" version="21.1" start="clk_0.clk"
       end="vga_ball_0.clock" />
658  <connection
659    kind="clock"
660    version="21.1"
661    start="clk_0.clk"
662    end="hps_0.f2h_axi_clock" />
663  <connection
664    kind="clock"
665    version="21.1"
666    start="clk_0.clk"
667    end="hps_0.h2f_axi_clock" />
```

```
 <connection
   kind="clock"
   version="21.1"
   start="clk_0.clk"
   end="hps_0.h2f_lw_axi_clock" />
 <connection
   kind="reset"
   version="21.1"
   start="clk_0.clk_reset"
   end="vga_ball_0.reset" />
 <interconnectRequirement for="$system" name="qsys_mm.
    clockCrossingAdapter" value="HANDSHAKE" />
 <interconnectRequirement for="$system" name="qsys_mm.
    enableEccProtection" value="FALSE" />
 <interconnectRequirement for="$system" name="qsys_mm.
    insertDefaultSlave" value="FALSE" />
 <interconnectRequirement for="$system" name="qsys_mm.
    maxAdditionalLatency" value="1" />
</system>
```

4.6 soc_system_top.sv

```
1  //
       =============================================================
2  // Copyright (c) 2013 by Terasic Technologies Inc.
3  //
       =============================================================
4  //
5  // Modified 2019 by Stephen A. Edwards
6  //
7  // Permission:
8  //
9  //   Terasic grants permission to use and modify this code
       for use
10 //   in synthesis for all Terasic Development Boards and
       Altera
11 //   Development Kits made by Terasic.  Other use of this
       code,
12 //   including the selling ,duplication, or modification of
       any
13 //   portion is strictly prohibited.
14 //
15 // Disclaimer:
16 //
17 //   This VHDL/Verilog or C/C++ source code is intended as a
       design
18 //   reference which illustrates how these types of functions
        can be
19 //   implemented.  It is the user's responsibility to verify
       their
20 //   design for consistency and functionality through the use
        of
21 //   formal verification methods.  Terasic provides no
       warranty
22 //   regarding the use or functionality of this code.
23 //
24 //
       =============================================================
25 //
26 //  Terasic Technologies Inc
27
```

```
28 //  9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu
       City, 30070. Taiwan
29 //
30 //
31 //                          web: http://www.terasic.com/
32 //                          email: support@terasic.com
33 module soc_system_top(
34
35  ///////// ADC /////////
36  inout          ADC_CS_N,
37  output         ADC_DIN,
38  input          ADC_DOUT,
39  output         ADC_SCLK,
40
41  ///////// AUD /////////
42  input          AUD_ADCDAT,
43  inout          AUD_ADCLRCK,
44  //inout          AUD_BCLK,
45  output         AUD_BCLK,
46  output         AUD_DACDAT,
47  //inout          AUD_DACLRCK,
48  output         AUD_DACLRCK,
49  output         AUD_XCK,
50
51  ///////// CLOCK2 /////////
52  input          CLOCK2_50,
53
54  ///////// CLOCK3 /////////
55  input          CLOCK3_50,
56
57  ///////// CLOCK4 /////////
58  input          CLOCK4_50,
59
60  ///////// CLOCK /////////
61  input          CLOCK_50,
62
63  ///////// DRAM /////////
64  output [12:0] DRAM_ADDR,
65  output [1:0]  DRAM_BA,
66  output         DRAM_CAS_N,
67  output         DRAM_CKE,
68  output         DRAM_CLK,
69  output         DRAM_CS_N,
70  inout [15:0]  DRAM_DQ,
71  output         DRAM_LDQM,
```

```
72  output          DRAM_RAS_N,
73  output          DRAM_UDQM,
74  output          DRAM_WE_N,
75
76  ///////// FAN /////////
77  output          FAN_CTRL,
78
79  ///////// FPGA /////////
80  output          FPGA_I2C_SCLK,
81  inout           FPGA_I2C_SDAT,
82
83  ///////// GPIO /////////
84  inout [35:0]    GPIO_0,
85  inout [35:0]    GPIO_1,
86
87  ///////// HEX0 /////////
88  output [6:0]    HEX0,
89
90  ///////// HEX1 /////////
91  output [6:0]    HEX1,
92
93  ///////// HEX2 /////////
94  output [6:0]    HEX2,
95
96  ///////// HEX3 /////////
97  output [6:0]    HEX3,
98
99  ///////// HEX4 /////////
100 output [6:0]    HEX4,
101
102 ///////// HEX5 /////////
103 output [6:0]    HEX5,
104
105 ///////// HPS /////////
106 inout           HPS_CONV_USB_N,
107 output [14:0]   HPS_DDR3_ADDR,
108 output [2:0]    HPS_DDR3_BA,
109 output          HPS_DDR3_CAS_N,
110 output          HPS_DDR3_CKE,
111 output          HPS_DDR3_CK_N,
112 output          HPS_DDR3_CK_P,
113 output          HPS_DDR3_CS_N,
114 output [3:0]    HPS_DDR3_DM,
115 inout [31:0]    HPS_DDR3_DQ,
116 inout [3:0]     HPS_DDR3_DQS_N,
```

```
117 inout [3:0]   HPS_DDR3_DQS_P,
118 output        HPS_DDR3_ODT,
119 output        HPS_DDR3_RAS_N,
120 output        HPS_DDR3_RESET_N,
121 input         HPS_DDR3_RZQ,
122 output        HPS_DDR3_WE_N,
123 output        HPS_ENET_GTX_CLK,
124 inout         HPS_ENET_INT_N,
125 output        HPS_ENET_MDC,
126 inout         HPS_ENET_MDIO,
127 input         HPS_ENET_RX_CLK,
128 input [3:0]   HPS_ENET_RX_DATA,
129 input         HPS_ENET_RX_DV,
130 output [3:0]  HPS_ENET_TX_DATA,
131 output        HPS_ENET_TX_EN,
132 inout         HPS_GSENSOR_INT,
133 inout         HPS_I2C1_SCLK,
134 inout         HPS_I2C1_SDAT,
135 inout         HPS_I2C2_SCLK,
136 inout         HPS_I2C2_SDAT,
137 inout         HPS_I2C_CONTROL,
138 inout         HPS_KEY,
139 inout         HPS_LED,
140 inout         HPS_LTC_GPIO,
141 output        HPS_SD_CLK,
142 inout         HPS_SD_CMD,
143 inout [3:0]   HPS_SD_DATA,
144 output        HPS_SPIM_CLK,
145 input         HPS_SPIM_MISO,
146 output        HPS_SPIM_MOSI,
147 inout         HPS_SPIM_SS,
148 input         HPS_UART_RX,
149 output        HPS_UART_TX,
150 input         HPS_USB_CLKOUT,
151 inout [7:0]   HPS_USB_DATA,
152 input         HPS_USB_DIR,
153 input         HPS_USB_NXT,
154 output        HPS_USB_STP,
155
156 ///////// IRDA /////////
157 input         IRDA_RXD,
158 output        IRDA_TXD,
159
160 ///////// KEY /////////
161 input [3:0]   KEY,
```

```
162
163  ///////// LEDR /////////
164  output [9:0]  LEDR,
165
166  ///////// PS2 /////////
167  inout         PS2_CLK,
168  inout         PS2_CLK2,
169  inout         PS2_DAT,
170  inout         PS2_DAT2,
171
172  ///////// SW /////////
173  input [9:0]   SW,
174
175  ///////// TD /////////
176  input         TD_CLK27,
177  input [7:0]   TD_DATA,
178  input         TD_HS,
179  output        TD_RESET_N,
180  input         TD_VS,
181
182
183  ///////// VGA /////////
184  output [7:0]  VGA_B,
185  output        VGA_BLANK_N,
186  output        VGA_CLK,
187  output [7:0]  VGA_G,
188  output        VGA_HS,
189  output [7:0]  VGA_R,
190  output        VGA_SYNC_N,
191  output        VGA_VS
192 );
193
194
195    logic sound_valid;
196    logic [2:0] sound_code;
197    logic signed [15:0] audio_sample;
198
199    wire sound_trigger = sound_valid;
200
201    soc_system soc_system0(
202      .clk_clk                       ( CLOCK_50 ),
203      .reset_reset_n                 ( 1'b1 ),
204
205      .hps_ddr3_mem_a                ( HPS_DDR3_ADDR ),
206      .hps_ddr3_mem_ba               ( HPS_DDR3_BA ),
```

```
207       .hps_ddr3_mem_ck              ( HPS_DDR3_CK_P ),
208       .hps_ddr3_mem_ck_n            ( HPS_DDR3_CK_N ),
209       .hps_ddr3_mem_cke             ( HPS_DDR3_CKE ),
210       .hps_ddr3_mem_cs_n            ( HPS_DDR3_CS_N ),
211       .hps_ddr3_mem_ras_n           ( HPS_DDR3_RAS_N ),
212       .hps_ddr3_mem_cas_n           ( HPS_DDR3_CAS_N ),
213       .hps_ddr3_mem_we_n            ( HPS_DDR3_WE_N ),
214       .hps_ddr3_mem_reset_n         ( HPS_DDR3_RESET_N ),
215       .hps_ddr3_mem_dq              ( HPS_DDR3_DQ ),
216       .hps_ddr3_mem_dqs             ( HPS_DDR3_DQS_P ),
217       .hps_ddr3_mem_dqs_n           ( HPS_DDR3_DQS_N ),
218       .hps_ddr3_mem_odt             ( HPS_DDR3_ODT ),
219       .hps_ddr3_mem_dm              ( HPS_DDR3_DM ),
220       .hps_ddr3_oct_rzqin           ( HPS_DDR3_RZQ ),
221
222       .hps_hps_io_emac1_inst_TX_CLK ( HPS_ENET_GTX_CLK ),
223       .hps_hps_io_emac1_inst_TXD0   ( HPS_ENET_TX_DATA[0] ),
224       .hps_hps_io_emac1_inst_TXD1   ( HPS_ENET_TX_DATA[1] ),
225       .hps_hps_io_emac1_inst_TXD2   ( HPS_ENET_TX_DATA[2] ),
226       .hps_hps_io_emac1_inst_TXD3   ( HPS_ENET_TX_DATA[3] ),
227       .hps_hps_io_emac1_inst_RXD0   ( HPS_ENET_RX_DATA[0] ),
228       .hps_hps_io_emac1_inst_MDIO   ( HPS_ENET_MDIO  ),
229       .hps_hps_io_emac1_inst_MDC    ( HPS_ENET_MDC   ),
230       .hps_hps_io_emac1_inst_RX_CTL ( HPS_ENET_RX_DV ),
231       .hps_hps_io_emac1_inst_TX_CTL ( HPS_ENET_TX_EN ),
232       .hps_hps_io_emac1_inst_RX_CLK ( HPS_ENET_RX_CLK ),
233       .hps_hps_io_emac1_inst_RXD1   ( HPS_ENET_RX_DATA[1]  ),
234       .hps_hps_io_emac1_inst_RXD2   ( HPS_ENET_RX_DATA[2]  ),
235       .hps_hps_io_emac1_inst_RXD3   ( HPS_ENET_RX_DATA[3]  ),
236
237       .hps_hps_io_sdio_inst_CMD     ( HPS_SD_CMD          ),
238       .hps_hps_io_sdio_inst_D0      ( HPS_SD_DATA[0]      ),
239       .hps_hps_io_sdio_inst_D1      ( HPS_SD_DATA[1]      ),
240       .hps_hps_io_sdio_inst_CLK     ( HPS_SD_CLK          ),
241       .hps_hps_io_sdio_inst_D2      ( HPS_SD_DATA[2]      ),
242       .hps_hps_io_sdio_inst_D3      ( HPS_SD_DATA[3]      ),
243
244       .hps_hps_io_usb1_inst_D0      ( HPS_USB_DATA[0]     ),
245       .hps_hps_io_usb1_inst_D1      ( HPS_USB_DATA[1]     ),
246       .hps_hps_io_usb1_inst_D2      ( HPS_USB_DATA[2]     ),
247       .hps_hps_io_usb1_inst_D3      ( HPS_USB_DATA[3]     ),
248       .hps_hps_io_usb1_inst_D4      ( HPS_USB_DATA[4]     ),
249       .hps_hps_io_usb1_inst_D5      ( HPS_USB_DATA[5]     ),
250       .hps_hps_io_usb1_inst_D6      ( HPS_USB_DATA[6]     ),
251       .hps_hps_io_usb1_inst_D7      ( HPS_USB_DATA[7]     ),
```

```
252     .hps_hps_io_usb1_inst_CLK     ( HPS_USB_CLKOUT      ),
253     .hps_hps_io_usb1_inst_STP     ( HPS_USB_STP         ),
254     .hps_hps_io_usb1_inst_DIR     ( HPS_USB_DIR         ),
255     .hps_hps_io_usb1_inst_NXT     ( HPS_USB_NXT         ),
256
257     .hps_hps_io_spim1_inst_CLK    ( HPS_SPIM_CLK  ),
258     .hps_hps_io_spim1_inst_MOSI   ( HPS_SPIM_MOSI ),
259     .hps_hps_io_spim1_inst_MISO   ( HPS_SPIM_MISO ),
260     .hps_hps_io_spim1_inst_SS0    ( HPS_SPIM_SS   ),
261
262     .hps_hps_io_uart0_inst_RX     ( HPS_UART_RX     ),
263     .hps_hps_io_uart0_inst_TX     ( HPS_UART_TX     ),
264
265     .hps_hps_io_i2c0_inst_SDA     ( HPS_I2C1_SDAT     ),
266     .hps_hps_io_i2c0_inst_SCL     ( HPS_I2C1_SCLK     ),
267
268     .hps_hps_io_i2c1_inst_SDA     ( HPS_I2C2_SDAT     ),
269     .hps_hps_io_i2c1_inst_SCL     ( HPS_I2C2_SCLK     ),
270
271     .hps_hps_io_gpio_inst_GPIO09  ( HPS_CONV_USB_N ),
272     .hps_hps_io_gpio_inst_GPIO35  ( HPS_ENET_INT_N ),
273     .hps_hps_io_gpio_inst_GPIO40  ( HPS_LTC_GPIO ),
274
275     .hps_hps_io_gpio_inst_GPIO48  ( HPS_I2C_CONTROL ),
276     .hps_hps_io_gpio_inst_GPIO53  ( HPS_LED ),
277     .hps_hps_io_gpio_inst_GPIO54  ( HPS_KEY ),
278     .hps_hps_io_gpio_inst_GPIO61  ( HPS_GSENSOR_INT ),
279 .vga_r (VGA_R),
280 .vga_g (VGA_G),
281 .vga_b (VGA_B),
282 .vga_clk (VGA_CLK),
283 .vga_hs (VGA_HS),
284 .vga_vs (VGA_VS),
285 .vga_blank_n (VGA_BLANK_N),
286 .vga_sync_n (VGA_SYNC_N),
287
288 .sound_sound_valid (sound_valid),
289 .sound_sound_code  (sound_code)
290   );
291
292     // The following quiet the "no driver" warnings for output
293     // pins and should be removed if using any of these
        peripherals
294
295     assign ADC_CS_N = SW[1] ? SW[0] : 1'bZ;
```

```
296     assign ADC_DIN = SW[0];
297     assign ADC_SCLK = SW[0];
298
299     assign AUD_ADCLRCK = 1'bZ;
300
301   sample_player multi_sound_player (
302     .clk50(CLOCK_50),
303     .reset(1'b0),
304     .trigger(sound_trigger),
305     .sound_code(sound_code),
306     .sample(audio_sample)
307   );
308
309
310 // Simple AUX/audio-jack test output
311   simple_audio_out audio_out (
312     .clk50(CLOCK_50),
313     .reset(1'b0),
314     .audio_sample(audio_sample),
315     .AUD_XCK(AUD_XCK),
316     .AUD_BCLK(AUD_BCLK),
317     .AUD_DACLRCK(AUD_DACLRCK),
318     .AUD_DACDAT(AUD_DACDAT)
319   );
320
321     assign DRAM_ADDR = { 13{ SW[0] } };
322     assign DRAM_BA = { 2{ SW[0] } };
323     assign DRAM_DQ = SW[1] ? { 16{ SW[0] } } : { 16{ 1'bZ } };
324     assign {DRAM_CAS_N, DRAM_CKE, DRAM_CLK, DRAM_CS_N,
325             DRAM_LDQM, DRAM_RAS_N, DRAM_UDQM, DRAM_WE_N} = {
                8{SW[0]} };
326
327     assign FAN_CTRL = SW[0];
328
329     wm8731_config audio_codec_config (
330      .clk50(CLOCK_50),
331      .reset(1'b0),
332      .I2C_SCLK(FPGA_I2C_SCLK),
333      .I2C_SDAT(FPGA_I2C_SDAT)
334   );
335
336     assign GPIO_0 = SW[1] ? { 36{ SW[0] } } : { 36{ 1'bZ } };
337     assign GPIO_1 = SW[1] ? { 36{ SW[0] } } : { 36{ 1'bZ } };
338
339     assign HEX0 = { 7{ SW[1] } };
```

```
340     assign HEX1 = { 7{ SW[2] } };
341     assign HEX2 = { 7{ SW[3] } };
342     assign HEX3 = { 7{ SW[4] } };
343     assign HEX4 = { 7{ SW[5] } };
344     assign HEX5 = { 7{ SW[6] } };
345
346     assign IRDA_TXD = SW[0];
347
348     assign LEDR = { 10{SW[7]} };
349
350     assign PS2_CLK = SW[1] ? SW[0] : 1'bZ;
351     assign PS2_CLK2 = SW[1] ? SW[0] : 1'bZ;
352     assign PS2_DAT = SW[1] ? SW[0] : 1'bZ;
353     assign PS2_DAT2 = SW[1] ? SW[0] : 1'bZ;
354
355     assign TD_RESET_N = SW[0];
356
357
358 endmodule
359
360
361 module simple_audio_out(
362     input  logic clk50,
363     input  logic reset,
364     input  logic signed [15:0] audio_sample,
365
366     output logic AUD_XCK,
367     output logic AUD_BCLK,
368     output logic AUD_DACLRCK,
369     output logic AUD_DACDAT
370 );
371
372     logic [9:0] div;
373     logic bclk_prev;
374     logic [4:0] bit_pos;
375     logic signed [15:0] frame_sample;
376
377     assign AUD_XCK  = div[1];   // 12.5 MHz
378     assign AUD_BCLK = div[4];   // 1.5625 MHz
379
380     wire bclk_falling = bclk_prev && !div[4];
381
382     always_ff @(posedge clk50 or posedge reset) begin
383         if (reset) begin
384             div <= 10'd0;
```

```
385            bclk_prev <= 1'b0;
386            bit_pos <= 5'd0;
387            frame_sample <= 16'sd0;
388            AUD_DACLRCK <= 1'b0;
389            AUD_DACDAT <= 1'b0;
390         end else begin
391            div <= div + 10'd1;
392            bclk_prev <= div[4];
393
394            if (bclk_falling) begin
395               if (bit_pos == 5'd0) begin
396                  frame_sample <= audio_sample;
397               end
398
399               if (bit_pos < 5'd16) begin
400                  AUD_DACLRCK <= 1'b0; // left channel
401                  AUD_DACDAT <= 1'b0;  // mute right channel to
                        remove static
402               end else begin
403                  AUD_DACLRCK <= 1'b1; // right channel
404                  AUD_DACDAT <= frame_sample[31 - bit_pos];
405               end
406
407               if (bit_pos == 5'd31)
408                  bit_pos <= 5'd0;
409               else
410                  bit_pos <= bit_pos + 5'd1;
411            end
412         end
413      end
414
415   endmodule
416
417
418   // WM8731 audio codec I2C configuration
419   module wm8731_config(
420      input  logic clk50,
421      input  logic reset,
422
423      output logic I2C_SCLK,
424      inout        I2C_SDAT
425   );
426
427      localparam [6:0] CODEC_ADDR = 7'h1A;
428      localparam integer CLK_DIV = 250;
```

```
429
430     logic [15:0] clk_count;
431     logic tick;
432
433     always_ff @(posedge clk50 or posedge reset) begin
434        if (reset) begin
435           clk_count <= 16'd0;
436           tick <= 1'b0;
437        end else begin
438           if (clk_count == CLK_DIV - 1) begin
439              clk_count <= 16'd0;
440              tick <= 1'b1;
441           end else begin
442              clk_count <= clk_count + 16'd1;
443              tick <= 1'b0;
444           end
445        end
446     end
447
448     // Open-drain style I2C data line
449     logic sdat_out_en;
450     logic sdat_out;
451
452     assign I2C_SDAT = sdat_out_en ? sdat_out : 1'bZ;
453
454     logic scl;
455
456     assign I2C_SCLK = scl;
457
458     logic [3:0] rom_index;
459     logic [15:0] rom_data;
460
461     always_comb begin
462        case (rom_index)
463           4'd0: rom_data = {7'd15, 9'h000}; // reset
464           4'd1: rom_data = {7'd6,  9'h000}; // power up
                 everything
465           4'd2: rom_data = {7'd2,  9'h075}; // left headphone
                 volume
466           4'd3: rom_data = {7'd3,  9'h060}; // right headphone
                  volume
467           4'd4: rom_data = {7'd4,  9'h012}; // analog path:
                 DAC selected
468           4'd5: rom_data = {7'd5,  9'h000}; // digital path
469           4'd6: rom_data = {7'd7,  9'h000}; // I2S, 16-bit
```

```
470         4'd7: rom_data = {7'd8,  9'h000}; // normal mode
                sampling
471         4'd8: rom_data = {7'd9,  9'h001}; // activate codec
472         default: rom_data = 16'h0000;
473      endcase
474   end
475
476   typedef enum logic [4:0] {
477      ST_IDLE,
478      ST_START_1,
479      ST_START_2,
480      ST_SEND_ADDR,
481      ST_ACK_ADDR,
482      ST_SEND_HIGH,
483      ST_ACK_HIGH,
484      ST_SEND_LOW,
485      ST_ACK_LOW,
486      ST_STOP_1,
487      ST_STOP_2,
488      ST_DONE
489   } state_t;
490
491   state_t state;
492
493   logic [7:0] byte_to_send;
494   logic [2:0] bit_index;
495   logic phase;
496   logic [15:0] wait_count;
497
498   always_ff @(posedge clk50 or posedge reset) begin
499      if (reset) begin
500         state <= ST_IDLE;
501         rom_index <= 4'd0;
502         scl <= 1'b1;
503         sdat_out_en <= 1'b1;
504         sdat_out <= 1'b1;
505         byte_to_send <= 8'd0;
506         bit_index <= 3'd7;
507         phase <= 1'b0;
508         wait_count <= 16'd0;
509      end else if (tick) begin
510         case (state)
511
512            ST_IDLE: begin
513               scl <= 1'b1;
```

```
514                  sdat_out_en <= 1'b1;
515                  sdat_out <= 1'b1;
516                  wait_count <= wait_count + 16'd1;
517
518                  // Small startup delay
519                  if (wait_count == 16'd1000) begin
520                     wait_count <= 16'd0;
521                     state <= ST_START_1;
522                  end
523               end
524
525               // I2C start: SDA goes low while SCL high
526               ST_START_1: begin
527                  scl <= 1'b1;
528                  sdat_out_en <= 1'b1;
529                  sdat_out <= 1'b0;
530                  state <= ST_START_2;
531               end
532
533               ST_START_2: begin
534                  scl <= 1'b0;
535                  byte_to_send <= {CODEC_ADDR, 1'b0}; // write
536                  bit_index <= 3'd7;
537                  phase <= 1'b0;
538                  state <= ST_SEND_ADDR;
539               end
540
541               ST_SEND_ADDR: begin
542                  if (phase == 1'b0) begin
543                     scl <= 1'b0;
544                     sdat_out_en <= 1'b1;
545                     sdat_out <= byte_to_send[bit_index];
546                     phase <= 1'b1;
547                  end else begin
548                     scl <= 1'b1;
549                     phase <= 1'b0;
550                     if (bit_index == 3'd0)
551                        state <= ST_ACK_ADDR;
552                     else
553                        bit_index <= bit_index - 3'd1;
554                  end
555               end
556
557               ST_ACK_ADDR: begin
558                  if (phase == 1'b0) begin
```

```
559                     scl <= 1'b0;
560                     sdat_out_en <= 1'b0; // release SDA for ACK
561                     phase <= 1'b1;
562                  end else begin
563                     scl <= 1'b1;
564                     phase <= 1'b0;
565                     byte_to_send <= rom_data[15:8];
566                     bit_index <= 3'd7;
567                     state <= ST_SEND_HIGH;
568                  end
569               end
570
571               ST_SEND_HIGH: begin
572                  if (phase == 1'b0) begin
573                     scl <= 1'b0;
574                     sdat_out_en <= 1'b1;
575                     sdat_out <= byte_to_send[bit_index];
576                     phase <= 1'b1;
577                  end else begin
578                     scl <= 1'b1;
579                     phase <= 1'b0;
580                     if (bit_index == 3'd0)
581                        state <= ST_ACK_HIGH;
582                     else
583                        bit_index <= bit_index - 3'd1;
584                  end
585               end
586
587               ST_ACK_HIGH: begin
588                  if (phase == 1'b0) begin
589                     scl <= 1'b0;
590                     sdat_out_en <= 1'b0;
591                     phase <= 1'b1;
592                  end else begin
593                     scl <= 1'b1;
594                     phase <= 1'b0;
595                     byte_to_send <= rom_data[7:0];
596                     bit_index <= 3'd7;
597                     state <= ST_SEND_LOW;
598                  end
599               end
600
601               ST_SEND_LOW: begin
602                  if (phase == 1'b0) begin
603                     scl <= 1'b0;
```

```
604                     sdat_out_en <= 1'b1;
605                     sdat_out <= byte_to_send[bit_index];
606                     phase <= 1'b1;
607                  end else begin
608                     scl <= 1'b1;
609                     phase <= 1'b0;
610                     if (bit_index == 3'd0)
611                        state <= ST_ACK_LOW;
612                     else
613                        bit_index <= bit_index - 3'd1;
614                  end
615               end
616
617               ST_ACK_LOW: begin
618                  if (phase == 1'b0) begin
619                     scl <= 1'b0;
620                     sdat_out_en <= 1'b0;
621                     phase <= 1'b1;
622                  end else begin
623                     scl <= 1'b1;
624                     phase <= 1'b0;
625                     state <= ST_STOP_1;
626                  end
627               end
628
629               // I2C stop: SDA goes high while SCL high
630               ST_STOP_1: begin
631                  scl <= 1'b0;
632                  sdat_out_en <= 1'b1;
633                  sdat_out <= 1'b0;
634                  state <= ST_STOP_2;
635               end
636
637               ST_STOP_2: begin
638                  scl <= 1'b1;
639                  sdat_out_en <= 1'b1;
640                  sdat_out <= 1'b1;
641
642                  if (rom_index == 4'd8) begin
643                     state <= ST_DONE;
644                  end else begin
645                     rom_index <= rom_index + 4'd1;
646                     state <= ST_START_1;
647                  end
648               end
```

```
649
650             ST_DONE: begin
651                scl <= 1'b1;
652                sdat_out_en <= 1'b1;
653                sdat_out <= 1'b1;
654             end
655
656             default: begin
657                state <= ST_IDLE;
658             end
659
660          endcase
661       end
662    end
663
664 endmodule
665
666 module sample_player(
667    input  logic clk50,
668    input  logic reset,
669    input  logic trigger,
670    input  logic [2:0] sound_code,
671
672    output logic signed [15:0] sample
673 );
674
675    localparam integer SAMPLE_DIV = 1024;
676
677    localparam integer WALL_COUNT   = 3826;
678    localparam integer PADDLE_COUNT = 9516;
679    localparam integer SCORE_COUNT  = 26785;
680
681    logic [15:0] wall_rom   [0:WALL_COUNT-1];
682    logic [15:0] paddle_rom [0:PADDLE_COUNT-1];
683    logic [15:0] score_rom  [0:SCORE_COUNT-1];
684
685    logic [31:0] div_count;
686    logic [15:0] index;
687    logic [15:0] current_count;
688    logic [2:0] current_code;
689    logic playing;
690
691    initial begin
692       $readmemh("sounds/wall.mem", wall_rom);
693       $readmemh("sounds/paddle.mem", paddle_rom);
```

```
694        $readmemh("sounds/score.mem", score_rom);
695     end
696
697     always_ff @(posedge clk50 or posedge reset) begin
698        if (reset) begin
699           div_count <= 32'd0;
700           index <= 16'd0;
701           sample <= 16'sd0;
702           playing <= 1'b0;
703           current_code <= 3'd0;
704           current_count <= 16'd0;
705        end else begin
706           if (trigger) begin
707              playing <= 1'b1;
708              index <= 16'd0;
709              div_count <= 32'd0;
710              current_code <= sound_code;
711
712              case (sound_code)
713                 3'd1: begin
714                    current_count <= WALL_COUNT;
715                    sample <= wall_rom[0];
716                 end
717
718                 3'd2: begin
719                    current_count <= PADDLE_COUNT;
720                    sample <= paddle_rom[0];
721                 end
722
723                 3'd3: begin
724                    current_count <= SCORE_COUNT;
725                    sample <= score_rom[0];
726                 end
727
728                 default: begin
729                    current_count <= PADDLE_COUNT;
730                    sample <= paddle_rom[0];
731                 end
732              endcase
733           end else if (playing) begin
734              if (div_count == SAMPLE_DIV - 1) begin
735                 div_count <= 32'd0;
736
737                 case (current_code)
738                    3'd1: sample <= wall_rom[index];
```

```
                    3'd2: sample <= paddle_rom[index];
                    3'd3: sample <= score_rom[index];
                    default: sample <= 16'sd0;
                endcase

                if (index == current_count - 1) begin
                    playing <= 1'b0;
                    sample <= 16'sd0;
                end else begin
                    index <= index + 16'd1;
                end
            end else begin
                div_count <= div_count + 32'd1;
            end
        end else begin
            sample <= 16'sd0;
        end
    end
   end

endmodule
```

4.7 vga_ball.sv

```
/*
 * VGA Air Hockey peripheral
 *
 * Register map, 32-bit Avalon MM slave, word addressing:
 *
 *   Verilog Address   Byte Offset   Dir   Description
 *   0                 0x00          R     STATUS:
 *                                         bit[0] = VSYNC_READY
 *
 *   1                 0x04          W     SOUND_CONTROL:
 *                                         bits[2:0] = SOUND_EVENT
 *
 *   2                 0x08          W     P1_POS:
 *                                         bits[15:0]  = P1_X
 *                                         bits[31:16] = P1_Y
 *
 *   3                 0x0C          W     P2_POS:
 *                                         bits[15:0]  = P2_X
 *                                         bits[31:16] = P2_Y
 *
 *   4                 0x10          W     PUCK_POS:
 *                                         bits[15:0]  = PUCK_X
 *                                         bits[31:16] = PUCK_Y
 *
 *   5                 0x14          W     SCORE:
 *                                         bits[2:0] = SCORE_P1
 *                                         bits[5:3] = SCORE_P2
 */

module vga_ball(
   input  logic        clk,
   input  logic        reset,

```

```
34      // Avalon MM slave
35      input  logic        chipselect,
36      input  logic        write,
37      input  logic [2:0]  address,
38      input  logic [31:0] writedata,
39      output logic [31:0] readdata,
40
41      // VGA
42      output logic [7:0]  VGA_R, VGA_G, VGA_B,
43      output logic        VGA_CLK, VGA_HS, VGA_VS,
44                          VGA_BLANK_n,
45      output logic        VGA_SYNC_n,
46
47      output logic        SOUND_VALID,
48      output logic [2:0]  SOUND_CODE
49
50  );
51
52
53      logic [10:0] hcount;
54      logic [9:0]  vcount;
55
56      vga_counters counters(.clk50(clk), .*);
57
58      logic [9:0] px, py;
59      assign px = hcount[10:1];
60      assign py = vcount;
61
62      // Hardware/software interface registers
63      logic [2:0] sound_event;
64
65      logic [9:0] p1_x, p1_y;
66      logic [9:0] p2_x, p2_y;
67
68      //puck position register
69      logic [9:0] puck_x, puck_y;
70
71      logic [2:0] score_p1, score_p2;
72
73      always_ff @(posedge clk or posedge reset) begin
74         if (reset) begin
75            sound_event <= 3'd0;
76
77            // Player 1 paddle reset position
78            p1_x <= 10'd120;
```

```
79          p1_y <= 10'd240;
80
81          // Player 2 paddle reset position
82          p2_x <= 10'd520;
83          p2_y <= 10'd240;
84
85          // Puck register exists but is not drawn yet
86          puck_x <= 10'd320;
87          puck_y <= 10'd240;
88
89          score_p1 <= 3'd0;
90          score_p2 <= 3'd0;
91       end else if (chipselect && write) begin
92          case (address)
93
94             // 0x04: SOUND_CONTROL
95             3'd1: begin
96                sound_event <= writedata[2:0];
97             end
98
99             // 0x08: P1_POS
100            3'd2: begin
101               p1_x <= writedata[9:0];
102               p1_y <= writedata[25:16];
103            end
104
105            // 0x0C: P2_POS
106            3'd3: begin
107               p2_x <= writedata[9:0];
108               p2_y <= writedata[25:16];
109            end
110
111            // 0x10: PUCK_POS
112            3'd4: begin
113               puck_x <= writedata[9:0];
114               puck_y <= writedata[25:16];
115            end
116
117            // 0x14: SCORE
118            3'd5: begin
119               score_p1 <= writedata[2:0];
120               score_p2 <= writedata[5:3];
121            end
122
123            default: begin
```

```
124             end
125          endcase
126       end
127    end
128
129    // Avalon read logic
130    logic vsync_ready;
131    assign vsync_ready = (vcount >= 10'd480);
132
133    always_comb begin
134       readdata = 32'd0;
135
136       case (address)
137          // 0x00: STATUS
138          3'd0: begin
139             readdata = {31'd0, vsync_ready};
140          end
141
142          default: begin
143             readdata = 32'd0;
144          end
145       endcase
146    end
147
148    // Rink geometry
149    localparam WL = 10;
150    localparam WR = 629;
151    localparam WT = 10;
152    localparam WB = 469;
153
154    localparam WW = 4;
155
156    localparam GT = 190;
157    localparam GB = 290;
158
159    // Centre circles (two concentric rings)
160    localparam CCR_LO = 24'd756;   // inner red  ring: r
       30, lo = 27.5^2
161    localparam CCR_HI = 24'd1056;  //                  hi =
       32.5^2
162    localparam CCB_LO = 24'd2756;  // outer blue ring: r
       55, lo = 52.5^2
163    localparam CCB_HI = 24'd3306;  //                  hi =
       57.5^2
164
```

```
    // Goal arcs: circle centred at left/right wall mid, r =
       90 +/- 3 px
    localparam ARC_LO = 24'd7569;  // 87^2
    localparam ARC_HI = 24'd8649;  // 93^2

    // Paddle dome: 2-level radial gradient (r^2 thresholds)
    localparam PAD_L1 = 24'd196;    // r <=  14  bright
    localparam PAD_L2 = 24'd256;    // r <= 16  inside border
    localparam PAD_L3 = 24'd784;  // r <= 28  dark
    localparam PAD_R2 = 24'd900;  // r <= 30  border

    // Puck dome: 2-level radial gradient (r^2 thresholds)
    localparam PUCK_L1 = 24'd256;   // r <=  16  bright
    localparam PUCK_L2 = 24'd289;  // r <= 17  inside border
    localparam PUCK_L3 = 24'd361;  // r <= 19  dark
    localparam PUCK_R2 = 24'd400;  // r <= 20  border

    // Score display positions
    localparam SCORE_Y  = 10'd30;
    localparam P1_SCORE_X = 10'd145;  // top of left side
    localparam P2_SCORE_X = 10'd465;  // top of right side

    localparam DIGIT_W = 10'd30;
    localparam DIGIT_H = 10'd50;

    // Centre circle distance squared
    logic signed [11:0] cdx, cdy;
    logic [23:0] cdist2;

    assign cdx    = $signed({2'b00, px}) - 12'sd320;
    assign cdy    = $signed({2'b00, py}) - 12'sd240;
    assign cdist2 = cdx * cdx + cdy * cdy;

    // Paddle 1 distance squared
    logic signed [11:0] p1dx, p1dy;
    logic [23:0] p1dist2;

    assign p1dx    = $signed({2'b00, px}) - $signed({2'b00,
       p1_x});
    assign p1dy    = $signed({2'b00, py}) - $signed({2'b00,
       p1_y});
    assign p1dist2 = p1dx * p1dx + p1dy * p1dy;

    // Paddle 2 distance squared
    logic signed [11:0] p2dx, p2dy;
```

```
207     logic [23:0] p2dist2;
208
209     assign p2dx    = $signed({2'b00, px}) - $signed({2'b00,
            p2_x});
210     assign p2dy    = $signed({2'b00, py}) - $signed({2'b00,
            p2_y});
211     assign p2dist2 = p2dx * p2dx + p2dy * p2dy;
212
213     // Puck distance squared
214     logic signed [11:0] puck_dx, puck_dy;
215     logic [23:0] puck_dist2;
216
217     assign puck_dx    = $signed({2'b00, px}) - $signed({2'b00,
            puck_x});
218     assign puck_dy    = $signed({2'b00, py}) - $signed({2'b00,
            puck_y});
219     assign puck_dist2 = puck_dx * puck_dx + puck_dy * puck_dy;
220
221
222     // Goal arc distance squared (centre = wall mid-point, r =
            90 px)
223     logic signed [11:0] lax, lay;
224     logic [23:0] larc_dist2;
225     assign lax       = $signed({2'b00, px}) - 12'sd10;
226     assign lay       = $signed({2'b00, py}) - 12'sd240;
227     assign larc_dist2 = lax * lax + lay * lay;
228
229     logic signed [11:0] rax, ray;
230     logic [23:0] rarc_dist2;
231     assign rax       = $signed({2'b00, px}) - 12'sd629;
232     assign ray       = $signed({2'b00, py}) - 12'sd240;
233     assign rarc_dist2 = rax * rax + ray * ray;
234
235     always_ff @(posedge clk or posedge reset) begin
236        if (reset) begin
237           SOUND_VALID <= 1'b0;
238           SOUND_CODE  <= 3'd0;
239        end else begin
240           SOUND_VALID <= 1'b0;
241
242           if (chipselect && write && address == 3'd1 &&
                  writedata[2:0] != 3'd0) begin
243              SOUND_VALID <= 1'b1;
244              SOUND_CODE  <= writedata[2:0];
245           end
```

```
246         end
247     end
248
249
250
251     // Seven-segment score digit, supports 0-7
252     // local x range: 0..29
253     // local y range: 0..49
254     function automatic logic score_digit_on(
255         input logic [2:0] digit,
256         input logic [9:0] x,
257         input logic [9:0] y
258     );
259         logic a, b, c, d, e, f, g;
260         logic seg_a, seg_b, seg_c, seg_d, seg_e, seg_f, seg_g;
261     begin
262         // Segment geometry
263         seg_a = (y < 10'd5)  && (x >= 10'd5)  && (x < 10'd25);
264         seg_b = (x >= 10'd25) && (x < 10'd30) && (y >= 10'd5)
                && (y < 10'd23);
265         seg_c = (x >= 10'd25) && (x < 10'd30) && (y >= 10'd27)
                && (y < 10'd45);
266         seg_d = (y >= 10'd45) && (y < 10'd50) && (x >= 10'd5)
                && (x < 10'd25);
267         seg_e = (x < 10'd5)  && (y >= 10'd27) && (y < 10'd45);
268         seg_f = (x < 10'd5)  && (y >= 10'd5)  && (y < 10'd23);
269         seg_g = (y >= 10'd23) && (y < 10'd28) && (x >= 10'd5)
                && (x < 10'd25);
270
271         // Default: all segments off
272         a = 1'b0;
273         b = 1'b0;
274         c = 1'b0;
275         d = 1'b0;
276         e = 1'b0;
277         f = 1'b0;
278         g = 1'b0;
279
280         case (digit)
281             3'd0: begin a=1; b=1; c=1; d=1; e=1; f=1; g=0; end
282             3'd1: begin a=0; b=1; c=1; d=0; e=0; f=0; g=0; end
283             3'd2: begin a=1; b=1; c=0; d=1; e=1; f=0; g=1; end
284             3'd3: begin a=1; b=1; c=1; d=1; e=0; f=0; g=1; end
285             3'd4: begin a=0; b=1; c=1; d=0; e=0; f=1; g=1; end
286             3'd5: begin a=1; b=0; c=1; d=1; e=0; f=1; g=1; end
```

```
287           3'd6: begin a=1; b=0; c=1; d=1; e=1; f=1; g=1; end
288           3'd7: begin a=1; b=1; c=1; d=0; e=0; f=0; g=0; end
289           default: begin a=0; b=0; c=0; d=0; e=0; f=0; g=0;
                  end
290       endcase
291
292       score_digit_on =
293           (a && seg_a) ||
294           (b && seg_b) ||
295           (c && seg_c) ||
296           (d && seg_d) ||
297           (e && seg_e) ||
298           (f && seg_f) ||
299           (g && seg_g);
300   end
301   endfunction
302
303
304   // Score display pixel detection
305   logic p1_score_on, p2_score_on;
306
307   always_comb begin
308       p1_score_on = 1'b0;
309       p2_score_on = 1'b0;
310
311       if (px >= P1_SCORE_X && px < P1_SCORE_X + DIGIT_W &&
312           py >= SCORE_Y    && py < SCORE_Y + DIGIT_H) begin
313           p1_score_on = score_digit_on(score_p1, px -
                  P1_SCORE_X, py - SCORE_Y);
314       end
315
316       if (px >= P2_SCORE_X && px < P2_SCORE_X + DIGIT_W &&
317           py >= SCORE_Y    && py < SCORE_Y + DIGIT_H) begin
318           p2_score_on = score_digit_on(score_p2, px -
                  P2_SCORE_X, py - SCORE_Y);
319       end
320   end
321
322
323   // VGA renderer
324   always_comb begin
325       {VGA_R, VGA_G, VGA_B} = 24'h000000;
326
327       if (VGA_BLANK_n) begin
328
```

```
329             // 1. Dark board border
330             {VGA_R, VGA_G, VGA_B} = 24'h222222;
331
332             // 2. Ice surface inside walls
333             if (px >= WL + WW && px <= WR - WW &&
334                 py >= WT + WW && py <= WB - WW)
335                {VGA_R, VGA_G, VGA_B} = 24'hF0F4FF;
336
337             // 3. Goal slot openings in wall
338             if (px >= WL && px < WL + WW && py > GT && py < GB)
339                {VGA_R, VGA_G, VGA_B} = 24'hCC1414;
340             if (px > WR - WW && px <= WR && py > GT && py < GB)
341                {VGA_R, VGA_G, VGA_B} = 24'h1432CC;
342
343             // 4. Goal arcs (ring centred on wall mid, r = 90 px
                   )
344             if (larc_dist2 >= ARC_LO && larc_dist2 <= ARC_HI &&
                   px >= WL + WW)
345                {VGA_R, VGA_G, VGA_B} = 24'hCC1414;
346             if (rarc_dist2 >= ARC_LO && rarc_dist2 <= ARC_HI &&
                   px <= WR - WW)
347                {VGA_R, VGA_G, VGA_B} = 24'h1432CC;
348
349             // 5. Centre line (red)
350             if ((px == 319 || px == 320) &&
351                 py >= WT + WW && py <= WB - WW)
352                {VGA_R, VGA_G, VGA_B} = 24'hCC2020;
353
354             // 6. Centre circles: inner red ring, outer blue
                   ring
355             if (cdist2 >= CCR_LO && cdist2 <= CCR_HI)
356                {VGA_R, VGA_G, VGA_B} = 24'hCC2020;
357             if (cdist2 >= CCB_LO && cdist2 <= CCB_HI)
358                {VGA_R, VGA_G, VGA_B} = 24'h2020CC;
359
360             // 7. Puck: 2-level gray radial dome
361             if (puck_dist2 <= PUCK_R2) begin
362                if      (puck_dist2 <= PUCK_L1) {VGA_R, VGA_G,
                      VGA_B} = 24'h484848;
363                else if (puck_dist2 <= PUCK_L2) {VGA_R, VGA_G,
                      VGA_B} = 24'h000000;
364                else if (puck_dist2 <= PUCK_L3) {VGA_R, VGA_G,
                      VGA_B} = 24'h484848;
365                else                            {VGA_R, VGA_G,
                      VGA_B} = 24'h000000;
```

```
366           end
367
368           // 8. Paddle 1: 2-level red radial dome
369           if (p1dist2 <= PAD_R2) begin
370              if      (p1dist2 <= PAD_L1) {VGA_R, VGA_G, VGA_B}
                     = 24'hFF2200;
371              else if (p1dist2 <= PAD_L2) {VGA_R, VGA_G, VGA_B}
                     = 24'h000000;
372              else if (p1dist2 <= PAD_L3) {VGA_R, VGA_G, VGA_B}
                     = 24'hCC1100;
373              else                        {VGA_R, VGA_G, VGA_B
                    } = 24'h000000;
374           end
375
376           // 9. Paddle 2: 2-level blue radial dome
377           if (p2dist2 <= PAD_R2) begin
378              if      (p2dist2 <= PAD_L1) {VGA_R, VGA_G, VGA_B}
                     = 24'h0033FF;
379              else if (p2dist2 <= PAD_L2) {VGA_R, VGA_G, VGA_B}
                     = 24'h000000;
380              else if (p2dist2 <= PAD_L3) {VGA_R, VGA_G, VGA_B}
                     = 24'h0020CC;
381              else                        {VGA_R, VGA_G, VGA_B
                    } = 24'h000000;
382           end
383
384           // 10. Score display: red for P1, blue for P2
385           if (p1_score_on) {VGA_R, VGA_G, VGA_B} = 24'hFF2200;
386           if (p2_score_on) {VGA_R, VGA_G, VGA_B} = 24'h0033FF;
387        end
388     end
389
390  endmodule
391
392  // VGA timing generator     unchanged from lab3 skeleton
393  module vga_counters(
394     input  logic         clk50, reset,
395     output logic [10:0]  hcount,
396     output logic [9:0]   vcount,
397     output logic         VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_n,
           VGA_SYNC_n);
398
399     parameter HACTIVE      = 11'd 1280,
400               HFRONT_PORCH = 11'd 32,
401               HSYNC        = 11'd 192,
```

```
402                HBACK_PORCH  = 11'd 96,
403                HTOTAL       = HACTIVE + HFRONT_PORCH + HSYNC +
                      HBACK_PORCH;
404
405       parameter VACTIVE      = 10'd 480,
406                 VFRONT_PORCH = 10'd 10,
407                 VSYNC        = 10'd 2,
408                 VBACK_PORCH  = 10'd 33,
409                 VTOTAL       = VACTIVE + VFRONT_PORCH + VSYNC +
                      VBACK_PORCH;
410
411       logic endOfLine;
412
413       always_ff @(posedge clk50 or posedge reset)
414          if (reset)          hcount <= 0;
415          else if (endOfLine) hcount <= 0;
416          else                hcount <= hcount + 11'd1;
417
418       assign endOfLine = hcount == HTOTAL - 1;
419
420       logic endOfField;
421
422       always_ff @(posedge clk50 or posedge reset)
423          if (reset)          vcount <= 0;
424          else if (endOfLine)
425             if (endOfField)  vcount <= 0;
426             else             vcount <= vcount + 10'd1;
427
428       assign endOfField = vcount == VTOTAL - 1;
429
430       assign VGA_HS = !( (hcount[10:8] == 3'b101) &
431                          !(hcount[7:5] == 3'b111));
432       assign VGA_VS = !( vcount[9:1] == (VACTIVE + VFRONT_PORCH)
              / 2);
433
434       assign VGA_SYNC_n  = 1'b0;
435
436       assign VGA_BLANK_n = !( hcount[10] & (hcount[9] | hcount
             [8]) ) &
437                            !( vcount[9] | (vcount[8:5] == 4'
                                  b1111) );
438
439       assign VGA_CLK = hcount[0];
440
441    endmodule
```

4.8 vga_ball_hw.tcl

```
# TCL File Generated by Component Editor 21.1
# Thu May 07 20:50:39 EDT 2026
# DO NOT MODIFY


#
# vga_ball "VGA Ball" v1.0
#  2026.05.07.20:50:39
#
#

#
# request TCL package from ACDS 16.1
#
package require -exact qsys 16.1


#
# module vga_ball
#
set_module_property DESCRIPTION ""
set_module_property NAME vga_ball
set_module_property VERSION 1.0
set_module_property INTERNAL false
set_module_property OPAQUE_ADDRESS_MAP true
set_module_property AUTHOR ""
set_module_property DISPLAY_NAME "VGA Ball"
set_module_property INSTANTIATE_IN_SYSTEM_MODULE true
set_module_property EDITABLE true
set_module_property REPORT_TO_TALKBACK false
set_module_property ALLOW_GREYBOX_GENERATION false
set_module_property REPORT_HIERARCHY false


#
# file sets
#
add_fileset QUARTUS_SYNTH QUARTUS_SYNTH "" ""
set_fileset_property QUARTUS_SYNTH TOP_LEVEL vga_ball
set_fileset_property QUARTUS_SYNTH
   ENABLE_RELATIVE_INCLUDE_PATHS false
set_fileset_property QUARTUS_SYNTH ENABLE_FILE_OVERWRITE_MODE
    false
```

```
42 add_fileset_file vga_ball.sv SYSTEM_VERILOG PATH vga_ball.sv
      TOP_LEVEL_FILE
43
44
45 #
46 # parameters
47 #
48
49
50 #
51 # module assignments
52 #
53 set_module_assignment embeddedsw.dts.group vga
54 set_module_assignment embeddedsw.dts.name vga_ball
55 set_module_assignment embeddedsw.dts.vendor csee4840
56
57
58 #
59 # display items
60 #
61
62
63 #
64 # connection point clock
65 #
66 add_interface clock clock end
67 set_interface_property clock clockRate 0
68 set_interface_property clock ENABLED true
69 set_interface_property clock EXPORT_OF ""
70 set_interface_property clock PORT_NAME_MAP ""
71 set_interface_property clock CMSIS_SVD_VARIABLES ""
72 set_interface_property clock SVD_ADDRESS_GROUP ""
73
74 add_interface_port clock clk clk Input 1
75
76
77 #
78 # connection point reset
79 #
80 add_interface reset reset end
81 set_interface_property reset associatedClock clock
82 set_interface_property reset synchronousEdges DEASSERT
83 set_interface_property reset ENABLED true
84 set_interface_property reset EXPORT_OF ""
85 set_interface_property reset PORT_NAME_MAP ""
```

```
set_interface_property reset CMSIS_SVD_VARIABLES ""
set_interface_property reset SVD_ADDRESS_GROUP ""

add_interface_port reset reset reset Input 1


#
# connection point vga
#
add_interface vga conduit end
set_interface_property vga associatedClock clock
set_interface_property vga associatedReset ""
set_interface_property vga ENABLED true
set_interface_property vga EXPORT_OF ""
set_interface_property vga PORT_NAME_MAP ""
set_interface_property vga CMSIS_SVD_VARIABLES ""
set_interface_property vga SVD_ADDRESS_GROUP ""

add_interface_port vga VGA_B b Output 8
add_interface_port vga VGA_BLANK_n blank_n Output 1
add_interface_port vga VGA_CLK clk Output 1
add_interface_port vga VGA_G g Output 8
add_interface_port vga VGA_HS hs Output 1
add_interface_port vga VGA_R r Output 8
add_interface_port vga VGA_SYNC_n sync_n Output 1
add_interface_port vga VGA_VS vs Output 1


#
# connection point avalon_slave_0
#
add_interface avalon_slave_0 avalon end
set_interface_property avalon_slave_0 addressUnits WORDS
set_interface_property avalon_slave_0 associatedClock clock
set_interface_property avalon_slave_0 associatedReset reset
set_interface_property avalon_slave_0 bitsPerSymbol 8
set_interface_property avalon_slave_0
   burstOnBurstBoundariesOnly false
set_interface_property avalon_slave_0 burstcountUnits WORDS
set_interface_property avalon_slave_0 explicitAddressSpan 0
set_interface_property avalon_slave_0 holdTime 0
set_interface_property avalon_slave_0 linewrapBursts false
set_interface_property avalon_slave_0
   maximumPendingReadTransactions 0
```

```
128 set_interface_property avalon_slave_0
       maximumPendingWriteTransactions 0
129 set_interface_property avalon_slave_0 readLatency 0
130 set_interface_property avalon_slave_0 readWaitTime 1
131 set_interface_property avalon_slave_0 setupTime 0
132 set_interface_property avalon_slave_0 timingUnits Cycles
133 set_interface_property avalon_slave_0 writeWaitTime 0
134 set_interface_property avalon_slave_0 ENABLED true
135 set_interface_property avalon_slave_0 EXPORT_OF ""
136 set_interface_property avalon_slave_0 PORT_NAME_MAP ""
137 set_interface_property avalon_slave_0 CMSIS_SVD_VARIABLES ""
138 set_interface_property avalon_slave_0 SVD_ADDRESS_GROUP ""
139
140 add_interface_port avalon_slave_0 chipselect chipselect Input
        1
141 add_interface_port avalon_slave_0 write write Input 1
142 add_interface_port avalon_slave_0 address address Input 3
143 add_interface_port avalon_slave_0 writedata writedata Input
       32
144 add_interface_port avalon_slave_0 readdata readdata Output 32
145 set_interface_assignment avalon_slave_0 embeddedsw.
       configuration.isFlash 0
146 set_interface_assignment avalon_slave_0 embeddedsw.
       configuration.isMemoryDevice 0
147 set_interface_assignment avalon_slave_0 embeddedsw.
       configuration.isNonVolatileStorage 0
148 set_interface_assignment avalon_slave_0 embeddedsw.
       configuration.isPrintableDevice 0
149
150
151 #
152 # connection point sound
153 #
154 add_interface sound conduit end
155 set_interface_property sound associatedClock clock
156 set_interface_property sound associatedReset ""
157 set_interface_property sound ENABLED true
158 set_interface_property sound EXPORT_OF ""
159 set_interface_property sound PORT_NAME_MAP ""
160 set_interface_property sound CMSIS_SVD_VARIABLES ""
161 set_interface_property sound SVD_ADDRESS_GROUP ""
162
163 add_interface_port sound SOUND_VALID sound_valid Output 1
164 add_interface_port sound SOUND_CODE sound_code Output 3
```