**Student #1**: Xie Xianghui (Xianghui.xie@ufl.edu)
**Student #2**: Kurtis J. Gnapp (kurtisgnapp@ufl.edu)
**Student #3**: Benjamin Hernandez (bhernandez2@ufl.edu)
**Student #4**: Alex Santiago (santiagorod.alex@ufl.edu)

# SoC Final Project: Pong Game with Addition of Audio Feedback

## Game Playing Sequence

At startup or after a manual reset, the system initializes and plays an introductory song ("Mary Had a Little Lamb"). After the song finishes, the Pong game begins and continues until a player reaches **9 points**. Once a player wins, both scores reset to **0**, and the game restarts **immediately without replaying** the introduction. Pressing the reset button at any time clears the scores and restarts the full sequence, including the song.

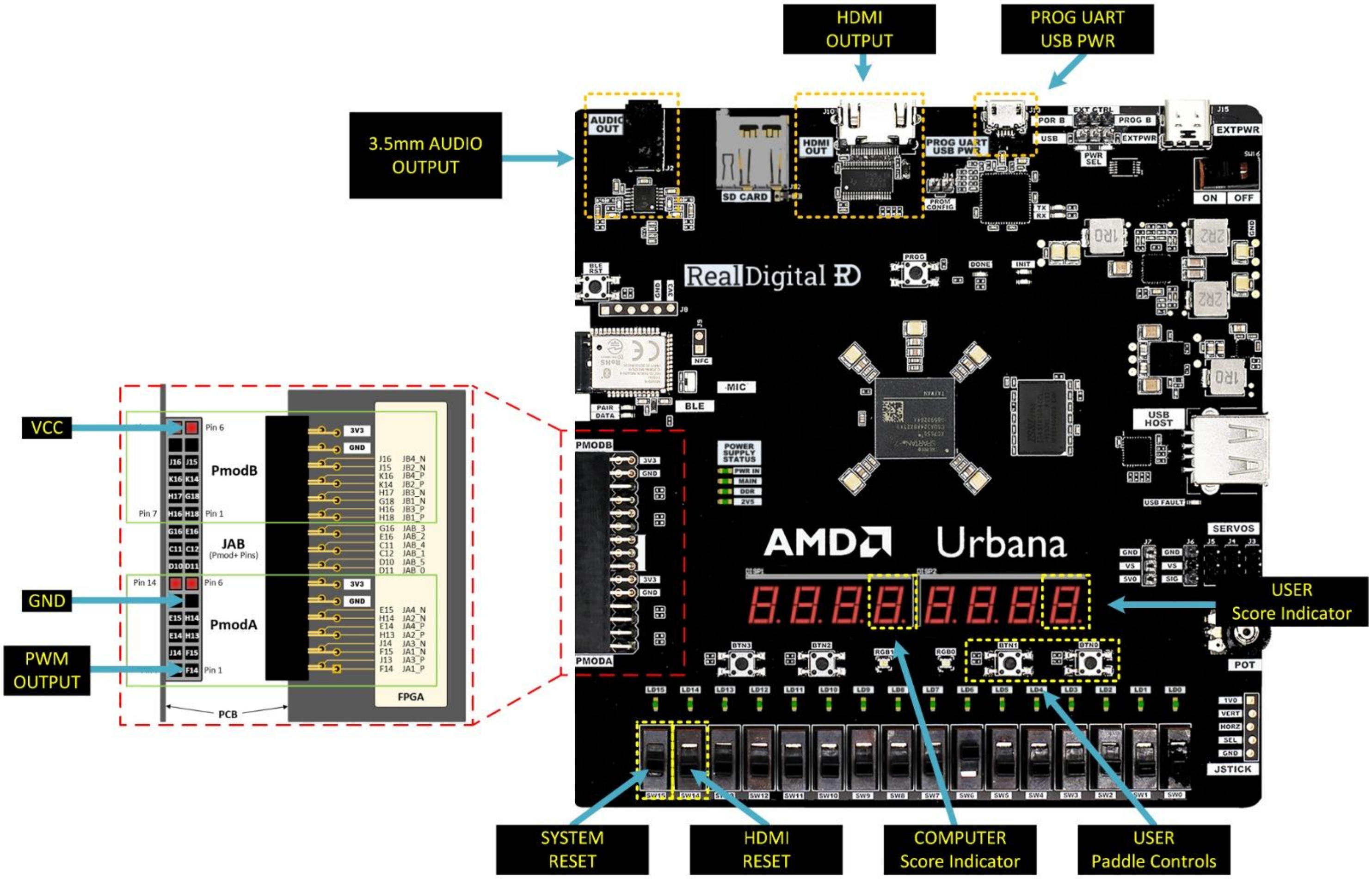


**Figure 1:** AMD Urbana development board layout, highlighting the key interfaces used in the project: audio output, HDMI video output, paddle control switches, score display via 7-segments, system reset, and more.

## Task #1

### Problems Faced:

- How to ensure the audio tables were generating the expected audio. We generated extra .wav files to hear the output. We also created .h and .mem files for use in Tasks #3 and #4. To ensure the integrity across file formats, we performed frequency analysis (FFT) using a Python script.

**What We Learned:**

- We learned how to create audio tables with a specific sample rate and 8-bit resolution. This involved encoding amplitude values from a known waveform (like a sine wave or melody tone) into discrete 8-bit integers sampled at fixed time intervals. These samples are then used to approximate audio through a PWM signal.

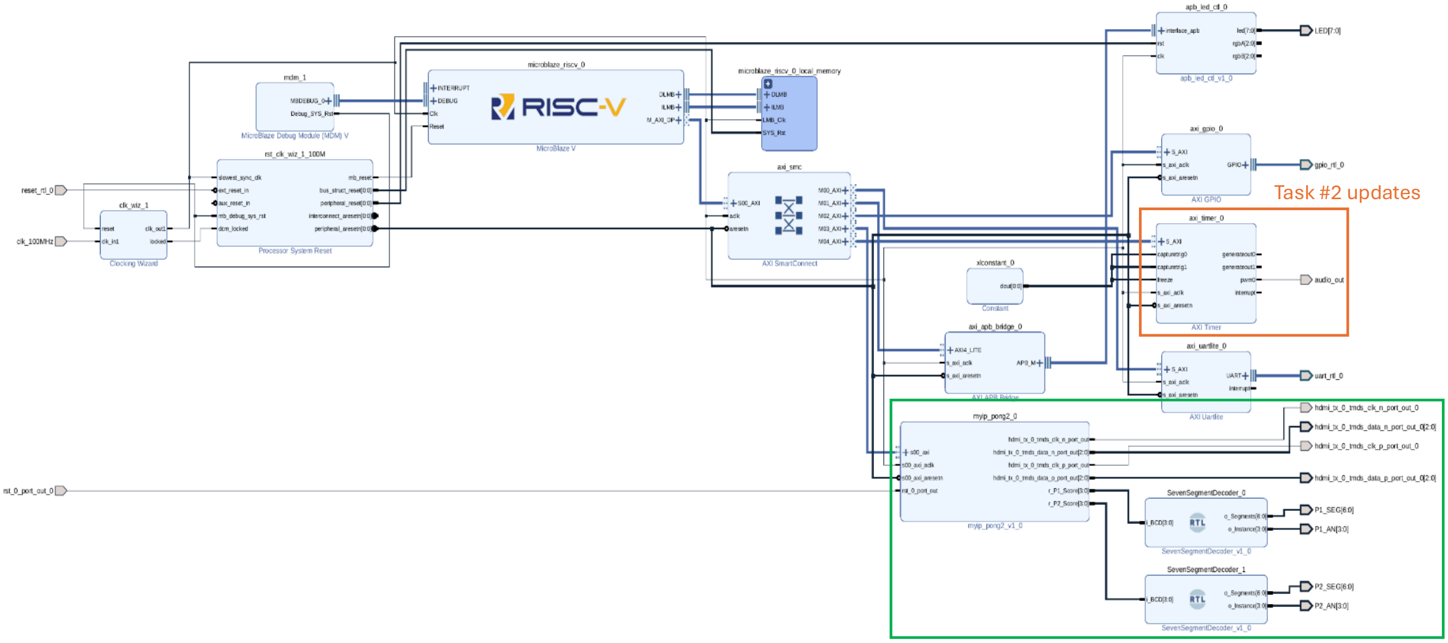
**Task #2**

**Problems Faced:**

- The AXI Timer IP initially didn’t produce the expected PWM output. To debug, we redirected the output to onboard LEDs and used UART (Putty) to print internal states. XSDB (Xilinx System Debugger) was also used to confirm register values and help isolate timing/setup issues.

**What We Learned:**

- We learned how to configure and operate the AXI Timer IP. Specifically, we controlled the TCSR0/TCSR1 (Control/Status Registers), TLR0/TLR1 (Load Registers), and TCR0/TCR1 (Counter Registers). These registers allowed us to set the PWM frequency (via period), duty cycle (via high time), enable reload (ARHT), and start/stop the signal (ENT, LOAD). We also learned to route this PWM output to the board’s 3.5mm audio jack interface.



Task #2 updates

Task #3 updates

**Figure 2:** Vivado block diagram illustrating the complete system integration for the Pong game, incorporating MicroBlaze RISC-V, AXI Timer for PWM-based audio output, AXI GPIO for paddle control, AXI UART for serial debugging, and 7-segment display logic for score output. This diagram reflects the implemented architecture for both Task #2 and Task #3.

**Task #3**

**Problems Faced:**

- The PWM signal couldn't drive the buzzer reliably due to its current requirements. Despite increasing drive strength and adjusting slew rate, the output was weak. Using an oscilloscope, we observed signal attenuation. We resolved this by adding a simple BJT-based driver circuit to buffer the signal.
- Another issue was that game logic introduced delays when playing audio. We optimized logic and removed redundant checks to improve responsiveness.
- We also fixed game issues like extra rounds being played after a win by modifying the Pong game base code.

**What We Learned:**

- We learned that PWM can approximate analog audio by modulating the duty cycle at a much higher frequency than the audio signal itself (e.g., using a 500 kHz PWM carrier to reproduce sub-8 kHz audio). We developed an understanding of the relationship between sample rate, duty cycle, and output frequency. Additionally, we learned how to synchronize audio playback with in-game events to trigger sounds at the correct times.

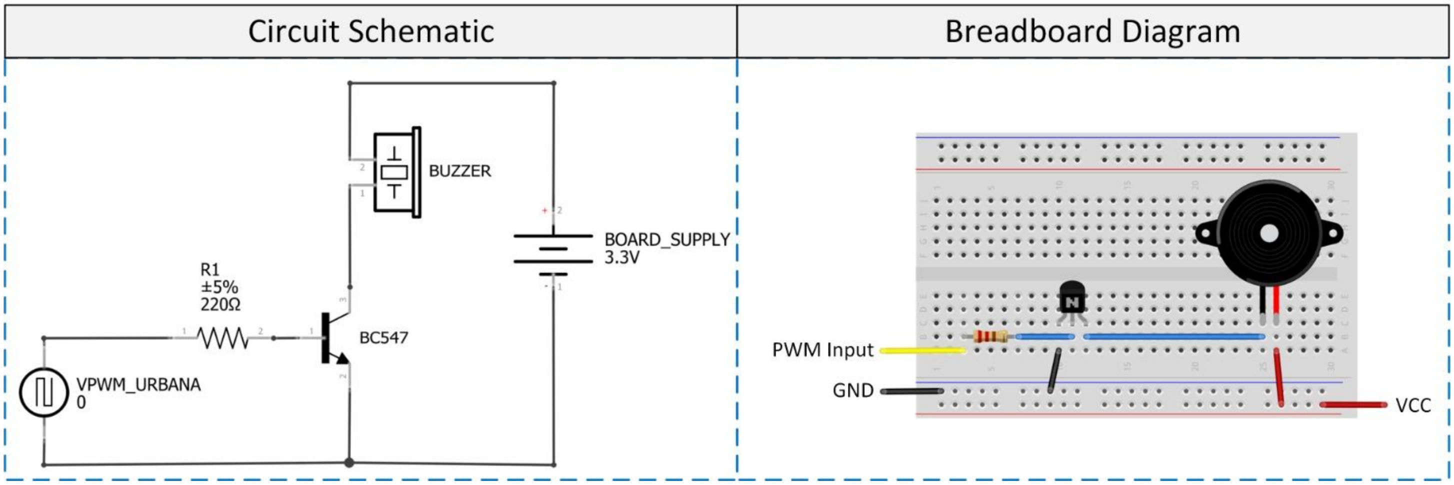


**Figure 3:** External buzzer drive circuit using a BC547 BJT transistor to amplify the PWM signal generated by the FPGA. This approach compensates for the limited drive strength of the direct FPGA output to the buzzer.