ECE532: Digital Systems Design
Final Project Group Report

Air Drums

Submission Date: April 7, 2025

Group 20:
Raymond Huynh
Leo Li
Joanne Tan
Xinyu Zhang

1.0 Overview

This section describes the Air Drums project, including the background and motivation, goal, block diagram, and IP blocks used.

1.1 Background and Motivation

Physical drums are noisy and not portable. The project aims to design an augmented-reality (AR)-like drum simulator that can be used anywhere. This project, inspired by [1] and [2], is an interactive system on Nexys Video Board where a user can use physical drumsticks to “hit” virtual drums on a computer monitor. The team, interested in creating something interactive and fun, found this concept easy and intuitive and therefore decided to build this Air Drums project for ECE532.

1.2 Goal

The project entails moving two drum sticks wired to the Nexys board into certain spatial zones, analogous to hitting drum sticks onto drums, to produce a corresponding sound from the board (figure 1). The SPI-based accelerometer attached to the front of the drum stick will take the hand movement velocity data to the board. If the downward acceleration exceeds 2*g*, where $g = 9.81\ \text{m/s}^2$, a hit is registered, triggering the sound effect and displaying a green highlight on the drum; the image data read from the camera is used to determine the position of the drumstick and thus location of the hit. Colour detection used to locate the drumsticks with a red tip and a green tip, respectively.

1.3 Block diagram

Figure 1 below shows the high-level block diagram of the design.

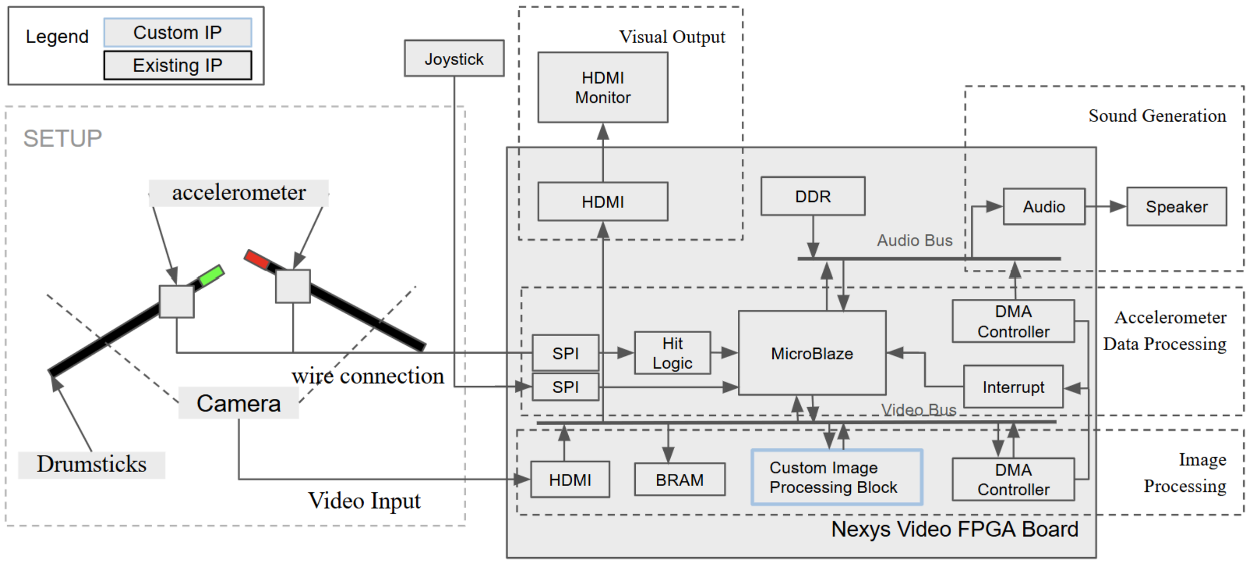

Fig. 1. Air Drums high-level block diagram.

1.4 IP Overview

Figure 2 below shows the detailed block diagram of the design with every IP block. Many of the IPs are sourced from demo projects available on Digilent with the exception of one custom IP block used for bit detection using a stream from a camera. The purple one is the custom IP block; the yellow ones serve for Hdmi input and output, and video streaming; the orange ones are for acceleration data retrieving; and the green ones serve for audio output. TheMore details about these IP blocks can be found in section [4.0 Description of Blocks](#).

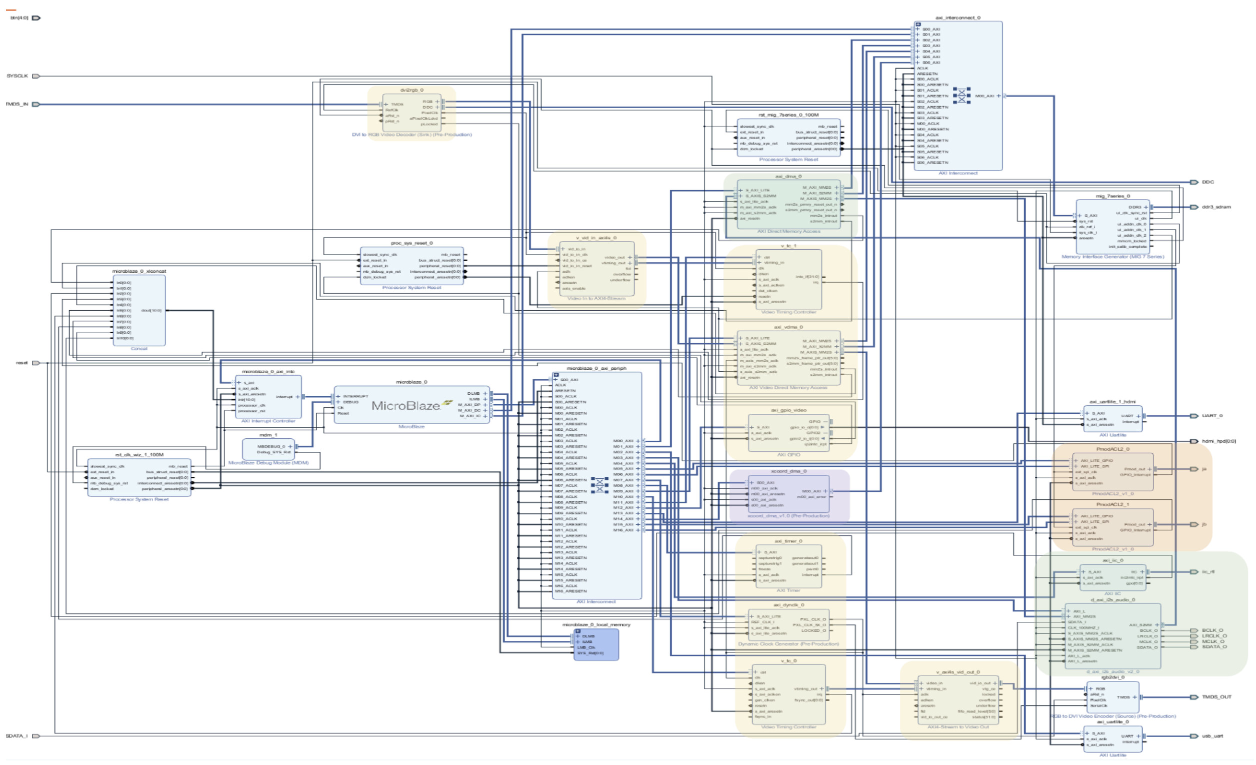

Fig. 2. Air Drum Vivado block diagram.

2.0 Outcome

This section describes the final outcome of the project, possible improvements, changes if the project were to be restarted, and notes for developers taking over the project.

2.1 Result

The final features of the project that the team was able to achieve was the integration of accelerometer, sound, and HDMI output. The custom pixel detection IP failed to detect red and green pixels while integrated into the design despite that the IP worked when implemented by itself. The final design had two drumsticks with accelerometers connected to the Nexys Video Board with long jumper wires as planned. However, there were only two drums on the screen as opposed to the planned four. Also, instead of using the positions of the drumsticks to calculate which drum is hit, the C code was written such that the left drumstick always triggers the left drum and the right drumstick the right drum for the sake of the Final Demo. The background was also black as opposed to a livestream of the drum player. The two current sounds are of a cymbal and of a bass drum, which played as expected, although with the bass sound quieter than the cymbal.

2.2 Future Work and Improvements

The project is near-complete with the critical bug that the custom pixel detection IP reports that no red or green coordinates are found. Removing the .cache and .runs files and regenerating the bitstream will likely resolve this issue. This is because it is hypothesized that Vivado synthesized an older, cached revision of the IP rather than the most recent, bug-free one.

Improvements to the project include that new custom IPs can be added to draw the drum animations more quickly onto the frame buffer to make overlaying the drums on top of a live video stream possible. Currently, the hit animation of every drum is pre-drawn into a separate static frame buffer; the hit animation is played by displaying each of the separate frame buffers. Adding more than two drums and drum sounds to the project would make the project more similar to an actual drum set that a user would expect.

2.3 Changes If the Project Were to Restart

If the team could start over, we would not attempt to use a camera peripheral and would go straight to using the webcam from a laptop. This way, the design and testing of the image processing custom IP (simple bit colour detection) would be able to start earlier, and allow for more integration time. Furthermore, searching for a working HDMI project from a previous working project and not starting with trying to convert the 2018.2 HDMI demo project to 2018.3 would also have saved anguish and time.

2.4 Notes For Developers Taking Over the Project

The design's current bugs and possible improvements are listed in section 2.2 Future Work and Improvements.

3.0 Project Schedule

This section compares the proposed milestones with the actual work done and discusses the reasons for the discrepancies.

3.1 Projected versus Actual Milestones

Table 1: Milestone Schedule (Projected vs. Actual)

Milestone	Projected	Actual
#1	Gather and read the documentations of all required PMODs.	Gather and read the documentations of all required PMODs.

#2	Complete block diagram on Vivado. Complete unit testing completed for the accelerometer, joystick, and flash drive.	Accelerometer data processed correctly to detect drum hits on DDR board. Retrieved .wav file from personal computer using Ethernet connection via TCP, can see the bytes received printing. Output video on a monitor using the Nexys Video HDMI example project 2018.2. Drew block diagram with AXI IIC.
#3	Have accelerometer data processed correctly to detect drum hits. Have VGA/HDMI camera capturing frames and sending them to DESL desktop.	Setup and tested all accelerometer and joystick data reading on the DDR board and Nexys Video board using polling. Stored audio files in DDR3 memory. Running into problems with HDMI, upgrading IP blocks from 2018.2 to 2018.3. Recreated the example HDMI project from scratch in 2018.3 - but bitstream generation is failing and some properties could not be ported from 2018.2 to 2018.3. Camera frame capturing using the assigned camera was not achieved - finding documentation for this peripheral was difficult and the example project was not outputting anything to the VGA monitor.
#4	Complete the image processing block so the positions of drum hits can be correctly recognized.	Speaker now plays audio as expected. Implemented left and right button presses to play different sounds. Successfully migrated Nexys Video HDMI demo to Vivado 2018.3, can now use this reference project for the creation of UI visuals on the monitor. Switched to OV7670 camera, can successfully read and write to registers of the camera.

#5	Implement drum hit sound generation and animation on the HDMI monitor whenever the user hits the air drum.	ACL2 and sound generation integrated together, also tested ACL2 attached to long jumper wires (eventually want to attach the ACL2's on a drumstick) and works with no problems. A sound is successfully played when the accelerometer detects a hit. Integration of HDMI with integrated sound+ACL2 in progress. Animation on the HDMI monitor is not implemented when the user hits the air drum. Implemented GUI for drum hit animations for HDMI output (independently tested). Functioning red pixel x-coordinate detection using software using laptop webcam.
#6	Implement joystick volume control. Finalize GUI.	Joystick was not implemented (stretch goal). GUI was finalized. Debugging integrated project of HDMI with integrated sound+ACL2. HDMI not outputting anything but sound+ACL2 still working. Pixel detection with custom IP block still under debug - x-coordinate value is staying constant despite a moving red object in the frame. Shortly after this milestone report, the custom IP block was verified and tested independently and was functional.

3.2 Discussion

The projected milestones were ambitious compared to what was achieved each week. While each functionality of the project encountered delays, most components of the project were pieced together right before demo day. The bring up for HDMI and image processing took significantly longer than expected. Integration of HDMI with the integrated sound and ACL2 project was difficult, requiring more than 2 weeks of debugging. Despite HDMI having a more complex block design, integrating the HDMI into the working sound and ACL2 project worked best. This was due to the organization of the sound and ACL2 project, particularly in respect to how the processor system resets and AXI interconnects were organized compared to the organization of the IP blocks in the HDMI project. Thus, using the sound and ACL2 project as a base and importing the HDMI related IP blocks into the project proved easier.

4.0 Description of the Blocks

Tables 2, 3, 4, and 5 below describe the IP blocks associated with the accelerometer drum hit detection, sound output, pixel detection, and HDMI camera and display subsystems, respectively.

Table 2: Accelerometer IP's

Accelerometer	
IP	Description and Source
PmodACL2_v1_0	IP block for interfacing with ACL2 (12 pin peripheral sensor that connects to PMOD port on either Nexys Video or Nexys DDR board). Uses SPI communication and requires a 50MHz SPI clock. Testing by serial outputting the collected data. Imported IP block from Vivado master IP library

Table 3: Sound generation IP's

Sound Generation	
IP	Description and Source
d_axi_i2s_audio_v2_0	IP testing was done through trial and error through the demo project, and used the demo project as a base to build the functionality of our Air Drum project. Requires 100Mhz clock. A DMA was attached to the AXI_S2MM output so that sound files within memory can be retrieved more efficiently. Imported IP block from Digilent Nexys Video DMA Audio Demo

Table 4: Pixel Detection IP's

Pixel Detection

IP	Description and Source
XCoord DMA	Custom IP. The IP burst-reads the pixels captured by the camera from the frame buffer, extracts the *x*-coordinates of the first red and green pixels encountered, and stores the coordinates in a status register. As of Apr. 7, 2025, the organizations of the registers is as follow: ● 0x0: Frame Buffer Start Address Register ● [31:0]: Start address of 1920x1080x3-bit frame buffer. ● 0x4: Reserved ● 0x8: X Coordinates Register ● [10:0]: *x*-coordinate of first red pixel ● [20:12]: *x*-coordinate of first green pixel ● 0xC: Control Register ● [0:0]: Start *x*-coordinate extraction. Self-clearing. ● [4:4]: Stop *x*-coordinate extraction. Self-clearing. ● [9:9]: *x*-coordinate extraction done. ● [12:12]: Red pixel found. ● [13:13]: Green pixel found. ● [26:16]: Number of read bursts completed.

Table 5: HDMI IP's

HDMI	
IP	Description and Source
Dynamic Clock Generator v1.0 (axi_dynclk_0)	Generates a pixel clock for a given resolution for the output signal. Imported Digilent IP block from Digilent Nexys Video HDMI Demo
AXI GPIO v2.0 (Rev. 20) (axi_gpio_video)	GPIO controller used to determine whether HDMI is plugged in and signal is stable.

	Within the project, the webcam from a laptop was used as dummy input as modifying the block diagram to function without HDMI input was difficult. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 19 to Rev. 20.
AXI Timer v2.0 (Rev. 20) (axi_timer_0)	Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 19 to Rev. 20.
AXI Uartlite v2.0 (Rev. 21) (axi_uartlite_0)	Used for UART communication to a PC for functionality like debugging, printing menus / peripheral data, or receiving user data. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 20 to Rev. 21
AXI Video Direct Memory Access v6.3 (Rev. 5) (axi_vdma_0)	Used to put frames from the incoming AXI4-Stream video stream into memory. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 5 to Rev. 6
DVI to RGB Video Decoder (Sink) v2.0 (dvi2rgb_0)	Takes in HDMI input and decodes it into a 24-bit RGB datastream. Imported Digilent IP block from Digilent Nexys Video HDMI Demo, child IPs ila_pixclk and ila_refclk automatically upgraded from Rev. 7 to Rev. 8
MicroBlaze Debug Module v3.2 (Rev. 15) (mdm_1)	Provides hardware debugging capabilities for the MicroBlaze. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 14 to Rev. 15.
MicroBlaze v11.0 (microblaze_0)	Executes C code.

	Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from v10.0 (Rev. 7) to v11.0.
Block Memory Generator v8.4 (Rev. 2) (microblaze_0_local_memory/lmb_bram)	Provides local memory for the MicroBlaze processor to use. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 1 to Rev. 2.
AXI Interrupt Controller v4.1 (Rev. 12) (microblaze_0_axi_intc)	Used to consolidate interrupts for MicroBlaze. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 11to Rev. 12.
AXI Interconnect v2.1 (Rev. 19) (microblaze_0_axi_periph, axi_mem_intercon)	Connects the AXI master devices to AXI slave peripherals. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 18 to Rev. 19.
Concat v2.1 (microblaze_0_xlconcat)	Used to concatenate multiple interrupt signals. In the project, this block had a total of 10 input signals. Interrupt signals are from the following blocks: ● Video Timing controller (x2) ● AXI Video Direct Memory Access (x2) ● AXI Direct Memory Access for audio (x2) ● AXI GPIO (x1) ● AXI IIC (x1) ● PmodACL2 (x1) ● AXI Timer (x1) Imported Xilinx IP block from Digilent Nexys Video HDMI Demo
Memory Interface Generator (MIG 7 Series) v4.2 (mig_7series_0)	Interfaces between the AXI memory bus on the FPGA and DDR memory on the board. Used for moving video frame buffers to and from the AXI VDMA and MicroBlaze.

	Within the project, each frame buffer was used to store a frame of our GUI, which allows for fast frame updates in real time by switching the frame buffer that was currently being shown. This approach was taken because drawing a new image/animation onto the screen in real time was very slow. Thus at the beginning of the project, each frame that is needed is pre-loaded. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from v4.1 to v4.2
RGB to DVI Video Encoder (Source) v1.4 (rgb2dvi_0)	Takes in a 24-bit RGB datastream and encodes it into an HDMI signal. Imported Digilent IP block from Digilent Nexys Video HDMI Demo
Processor System Reset v5.0 (Rev. 13) (rst_mig_7series_0_100M, rst_mig_7series_0_pxl)	Generates reset signals for different clock domains across the design. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 12 to Rev. 13
AXI4-Stream to Video Out v4.0 (Rev. 9) (v_axi4s_vid_out_0)	Converts an AXI4-Stream video stream into a 24-bit RGB datastream. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 8 to Rev. 9
Video Timing Controller v6.1 (Rev. 13) (v_tc_0, v_tc_1)	Generates or analyzes timing and synchronization signals for output or from input video. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 12 to Rev. 13
Video In to AXI4-Stream v4.0 (Rev. 9) (v_vid_in_axi4s_0)	Converts a 24-bit RGB datastream into an AXI4-Stream video stream.

	Imported Xilinx IP block from Digilent Nexys Video HDMI Demo, automatically upgraded from Rev. 8 to Rev. 9
Constant v1.1 (Rev. 5) (xlconstant_0)	Used for turning on or off the HDMI signal. Imported Xilinx IP block from Digilent Nexys Video HDMI Demo

Constraint Files from Digilent Nexys Video DMA Audio Demo and Digilent Nexys Video HDMI Demo were imported and used in the final project. The SDK files and code from these demo projects were repurposed and edited to fit the use of our project.

5.0 Description of Your Design Tree

The project's GitHub repository is located here with the following organization:

- **doc:** Contains the final report and presentation slides.
- **ip_repo**: Contains the custom *x*-coordinate detection IP, and the pack of Vivado IPs used in our project.
- **Final_drum**: Contains the final version code where after a hit is detected, drum sound will be generated from the audio output port and HDMI switches buffer to indicate the hit. Note that the *x*-coordination detection is integrated in the block design, but the software side is not yet fully integrated.
    - The software code is under: *Final_drum.sdk/drum_v6/src*
- **Two_sounds_with_accelerometer**: Contains the version of the project where we generate a sound after triggering a hit from the accelerometer. The push buttons left and right can be used to switch between bass and cymbal sound, and the up and down button can be used to adjust the sound output volume.
    - The software code is under: *Two_sounds_with_accelerometer.sdk/acl_audio_128/*

6.0 Tips and Tricks

Try to start integration as early as possible! Integration is the step that takes the longest time. Furthermore, if using HDMI, upgrading IP blocks from 2018.2 to 2018.3 was very difficult. So, if you can find a working project that uses HDMI already, it would speed up the process significantly. Always start simple and aim for the minimum functionality before looking into adding more complex logic. Test often, and work with your teammates to debug - more eyes looking at a problem, the faster it will be solved, especially if you are stuck. Lastly, consider the available board resources when designing the project, for example the number of PMOD ports.

7.0 Video

A video introducing the project can be found here.

8.0 References

[1] A. Chen, D. Singh, and K. Xu, “ECE 5730 Final Project: Air Drums,” Dec. 11, 2023. https://ece4760.github.io/Projects/Fall2023/ds2392_kx74_ac2839/Air_Drum.html (accessed Jan. 30, 2025).

[2] R. Naidu and L. Fernandez, “Virtual Drum Set 6.111 Final Project Report,” Dec. 12, 2012. https://web.mit.edu/6.111/www/f2012/projects/luisf_Project_Final_Report.pdf (accessed Jan. 30, 2025).