Framework for **A**ccelerated **S**pa**T**ial awareness in **A**ugmented **R**eality (**FAST**-**AR**)

I. Project Overview

1. Idea and Application

The objective of this project is to design and develop a hardware accelerated assistive learning augmented reality (AR) headset for use in training or fast navigation settings. FAST-AR aims to implement a direct FPGA video processing pipeline using an AUP-ZU3 FPGA/SoC. By providing microsecond latencies, and visual filters, users will enjoy improved navigational awareness in a seamless experience while eliminating motion sickness associated with software based AR/HUDs [1].

2. Motivation

In technical environments, or during fast navigation, even small amounts of delay in an AR/HUD display can result in motion sickness [1]. Our project aims to eliminate this by providing near-zero latency video processing, enhancing user experience.

II. Technical Approach

1. Methodology/Architecture

Our architecture implements a heterogeneous architecture using the ZU3 MPSoC. Video data obtained from the on-board MIPI connector will be streamed directly to the FPGA programmable logic (PL). In our PL, One pipeline will then process low-latency, high-fidelity video, while another serves as a graphics overlay pipeline. Finally, the processed video is streamed out through the mini-DisplayPort (mDP). Meanwhile, the SoC will simultaneously collect navigation data and relay coordinates via AXI-lite to the FPGA to aid in placing “virtual” aid objects with minimal CPU/SoC overhead, reducing power draw of the SoC [2].

Low Latency Video Architecture: Video capture is managed by the FPGA over the MIPI interface. The video will then be processed in several different parallel pipelines to undergo demosaicing, contrast adjustments, sharpening and edge-detect. A final stage will be at the end of the pipeline. Here simple UI elements are overlaid based on data provided from the SoC (described below) and finally, to compensate for the biconvex lenses found in AR style headsets, barrel distortion may be added via a live distortion algorithm [3].

Graphics & Filtering: The system will also inject navigation data directly into the video stream using a dedicated Hardware Sprite Engine. The ARM SoC receives direction strings (e.g., directional arrows or distance markers) from a smartphone and writes these coordinates to set registers on the FPGA. These registers will be written over the AXI protocol. After reading these values, the Sprite Engine reads these shapes from on-chip BRAM and overlays them onto the live video feed. A module will then mix these graphics with the live frame, allowing directions to appear as if they are floating in the physical environment. This approach replaces complex software-based UI rendering with high-speed hardware logic, significantly reducing SoC utilization.

2. Justification

This project is well suited for 554 due to its heavy reliance on FPGA processing, along with the use of a heterogeneous system. This provides complexity through system interfaces such as SoC to FPGA and the need to perform low latency processing of high fidelity video frames at a high framerate. Additionally, for some UI/overlay elements, head motion will need to be monitored so some sensors such as an IMU will be needed. These sensors will require additional SPI, UART or I2C interfaces to be implemented and monitored.

Our design is novel in that the graphics are entirely managed by the FPGA with data being supplied from the SoC. This allows for very low latency and smooth operation that will improve user experience.

Additionally, FPGAs are designed exactly for such high-speed, low-latency processing and so prevents the end user from suffering nausea due to high latency and low framerate. Many filters that we will use are memory-bound on an SoC, such as kernel operations for filtering and potentially barrel distortion. Without the FPGA PL, this project would be entirely ineffective at its intended goal of improving user awareness.

III. Strategy

1. Anticipated Challenges

We anticipate several challenges with implementation. Chief among these we anticipate interfacing with the MIPI camera(s), and creating a video pipeline capable of handling high framerate full RGB video at near zero latency to be challenging. Depending on the exact filters implemented, up to several MB of BRAM may be needed to realize our image processing, so

managing memory at these bounds will be challenging.

Additionally, we expect challenges with buffering large amounts of high speed, high fidelity data. Depending on the end resolution of our camera, up to several MB of data may be needed for image processing techniques and filters. Further, implementing live barrel distortion is a documented process, but translating the complex mathematics and memory mapping into efficient, real-time Verilog logic will be a significant developmental hurdle and so is left as a reach goal [3].

2. Risk Mitigation

To mitigate risk, we first will focus on physical risk including risk to the board, for this, Henry will manufacture a polycarbonate shell for the FPGA to provide physical and ESD protection. If we encounter issues with the SoC to FPGA communication, we will shift to an all-FPGA focused approach where sensors on the headset will provide basic cardinal directions. The largest concern we have is that our Xilinx board will fail. In the event, our new MVP will shift to using the DE1-SoC. In this scenario we will follow a similar path as if the SoC→FPGA communication is unfeasible and focus only on filtered video throughput with sensor assisted directionality.

IV. Timeline & Evaluation Method

To evaluate our progress throughout the semester, we will employ two main milestones to keep the project on track

Mid-Term Criteria: Successful implementation of the MIPI→mDP video passthrough pipeline. The FPGA must be able to read a frame from the MIPI camera, perform basic image preparation (demosaicing), apply basic hardware filters (e.g., grayscale or edge detection), and output it to the mDP without visual tearing. Additionally, the SoC will be able to pass basic values to the FPGA via AXI .

Final Criteria (MVP): Full integration of the heterogeneous architecture. The FPGA will dynamically draw the overlay and filters at a minimum of 30 frames per second to ensure a tolerable AR experience and perform the requisite filtering and low latency. The SoC will provide navigational data to the FPGA which will then be rendered as part of the overlay.

Below is a rough timeline for our project detailing both PL and PS goals. Gantt will be used for actual progress tracking. For access email Henry.

Week	PL	PS
1-2	MIPI→FPGA→ mDP; begin sprite engine	OS setup; investigate setup for online data receipt
2-4	AXI→FPGA register writing; test sprite drawing	Memory mapped I/O setup; send dummy data to FPGA registers
4-6	Finalize image filtering; begin work on image integration and overlay.	Integrate Bluetooth/UART stack; parse incoming navigation strings.
6-8	Finish image overlay; ensure timing for 30fps; Barrel distortion	Finalize PYNQ-based UI driver; optimize register write

V. Members and Responsibilities

Harshul *Jalan* brings experience in HW/SW codesign, system architecture and embedded systems. Further background includes RTL development, AXI interfacing and MM I/O. He will define system architecture and PS/PL integration including video pipelining, overlay, filtering, UI, Bluetooth/UART stack and AXI register writing.

Andrew *John Cadiena* brings a skillset in digital system design and embedded and parallel systems. His interests lie in RTL module development as well as computer architecture. He will be responsible for implementing sprite overlay on the display.

Ananya *Virmani* brings with her a mixed experience of hardware and software focussing on digital design, computer architecture. Her interests lie in RTL development, OS, C programming and testbenching. She will be responsible for testbenching modules, overlay onto live images, assist with the software stack and UART stack.

Henry *Wysong-Grass* has experience in digital system design, especially in RTL development, C firmware, and testbenching. Their interests lie in low-energy computing. They will be responsible for the core video pipeline (MIPI to FPGA to mDP), filters, IMU integration and testbenching modules.

Several tasks will “float” throughout the semester such as MIPI configuration (camera dependent) creating sprites and barrel distortion.

References

[1] F. M. Alessa, M. H. Alhaag, I. M. Al-Harkan, M. Z. Ramadan, and F. M. Alqahtani, “A Neurophysiological Evaluation of Cognitive Load during Augmented Reality Interactions in Various Industrial Maintenance and Assembly Tasks.,” Sensors (Basel), vol. 23, no. 18, Sep. 2023, doi: 10.3390/s23187698.

[2] Real Digital, “AUP-ZU3 Reference Manual.” 2025. [Online]. Available: https://www.realdigital.org/downloads/3cd0cbae4d00eaf94270e5792c746d4c.pdf

[3] H. Blasinski, Wei Hai, and F. Lohier, “FPGA architecture for real-time barrel distortion correction of colour images,” in 2011 IEEE International Conference on Multimedia and Expo, Jul. 2011, pp. 1–6. doi: 10.1109/ICME.2011.6011854.