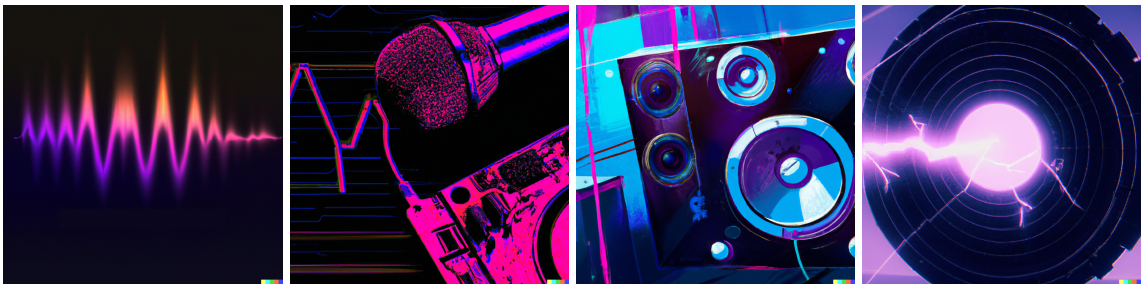
UNSW COMP3601 - Design Project A - 23T3

FPGA Microphone Audio System



Introduction:
Audio-based applications are becoming increasingly advanced as technology progresses. From music production to virtual reality, film soundscapes and live teleconferences, there is a growing need for specialized audio solutions that can handle complex processing tasks in real-time. Such demands can exceed the capabilities of general purpose hardware: this is where *custom* hardware can come into play.

Custom hardware is specifically designed to meet the unique needs of an application, providing improved performance and efficiency. For advanced audio applications, custom hardware can be used to implement specialized algorithms, such as digital signal processing (DSP) routines and/or interfaces that enable real-time processing of large amounts of audio data.

Field Programmable Gate Arrays (FPGAs) are the most popular avenue for the development and deployment of custom hardware. They are flexible and highly configurable devices that can be programmed to perform a wide range of digital processing tasks. Compared to traditional single-purpose hardware, FPGAs can reconfigure themselves to achieve application specific needs: such as those posed by modern audio engineering.

Overview:
In this project, you will be assigned into groups of 4 students and tasked to implement a custom microphone system using an FPGA-based Multiprocessor System-on-Chip (MPSoC) - the Xilinx Kria KV260. This is a ‘best of both worlds’ device that has both FPGA fabric (to provide the advantages of custom hardware) as well as a ‘hard’ ARM quad-core processor (to provide the benefits of a traditional processor).

To this platform you will connect an I2S MEMs microphone, and use this to collect audio (the ‘Basic Requirements’). Once this is achieved, you will then take ownership of the project and develop it in a unique way specific to your interests (the ‘Application’). The teaching staff have provided you with some project ideas, or you can work with them to design your own!

Setup for the Basic Requirements:

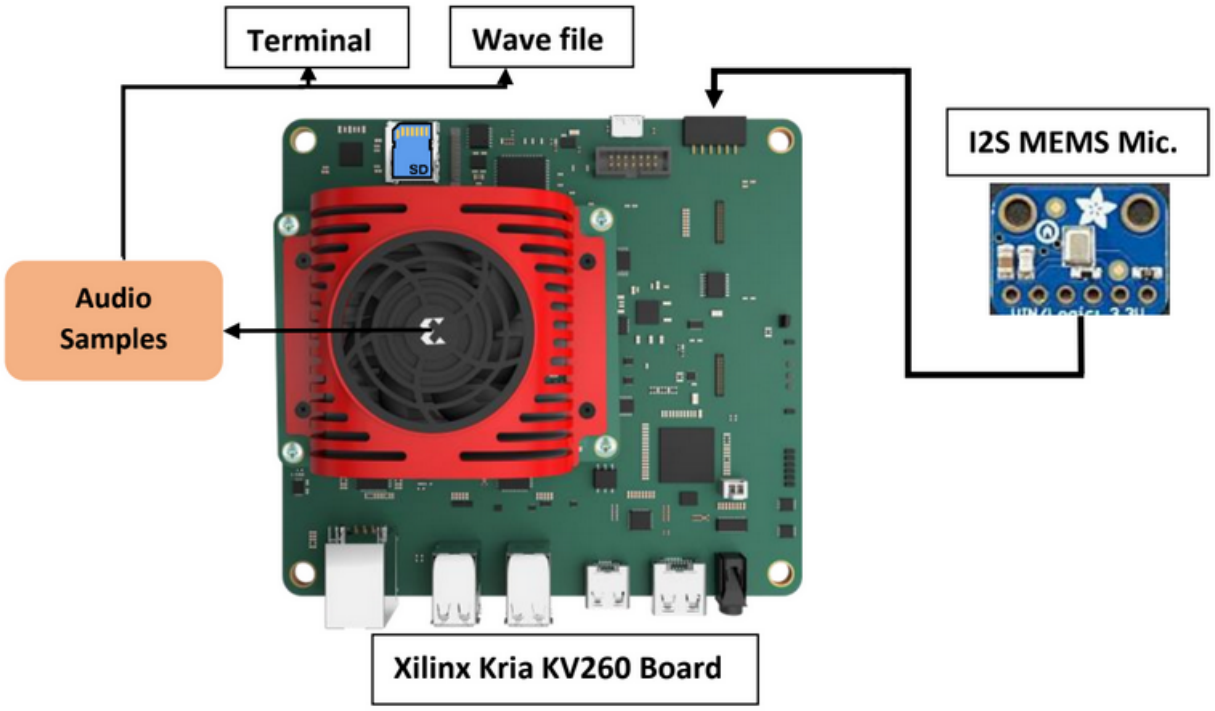


Figure 1: Basic Requirements System Diagram

In this project, you and your team will first implement the basic system diagram specified in Figure 1: That is, you will connect the I2S microphone to the Kria board and capture the audio data. This data will need to be output either via the Terminal or by saving it as a .wav audio file.

The Development Board:
The FPGA kit used in this project is the Xilinx Kria KV260 Vision AI Starter Kit, a development board including a Zynq UltraScale+ MPSoC. This platform combines a quad-core ARM Cortex A53 hard processor (the Processing System / PS) and an FPGA (the Programmable Logic / PL). The processor will run a custom Linux image called PetaLinux. In PetaLinux, device drivers and application software can be executed - you will develop these to save audio samples from the microphone and perform any other needed operations.

I2S MEMS Microphone:
The provided microphone is a MEMS device which converts sound to voltage, giving out the sampled audio as a purely digital signal. It is capable of capturing sound waves with frequencies from 50Hz to 15KHz, which is good for all general audio recording/processing applications. The microphone supports I2S (Inter-IC Sound) serial bus interface protocol, which is used for connecting digital audio devices together. This will be the protocol used in this project to transfer the sampled audio data to the FPGA-based audio processing pipeline.

Provided Components (per group):
2 x Kria KV260 Vision AI Starter Kit; 2 x Power Supply; 2 x I2S MEMS Microphone; Wires

Project Layout (Source):

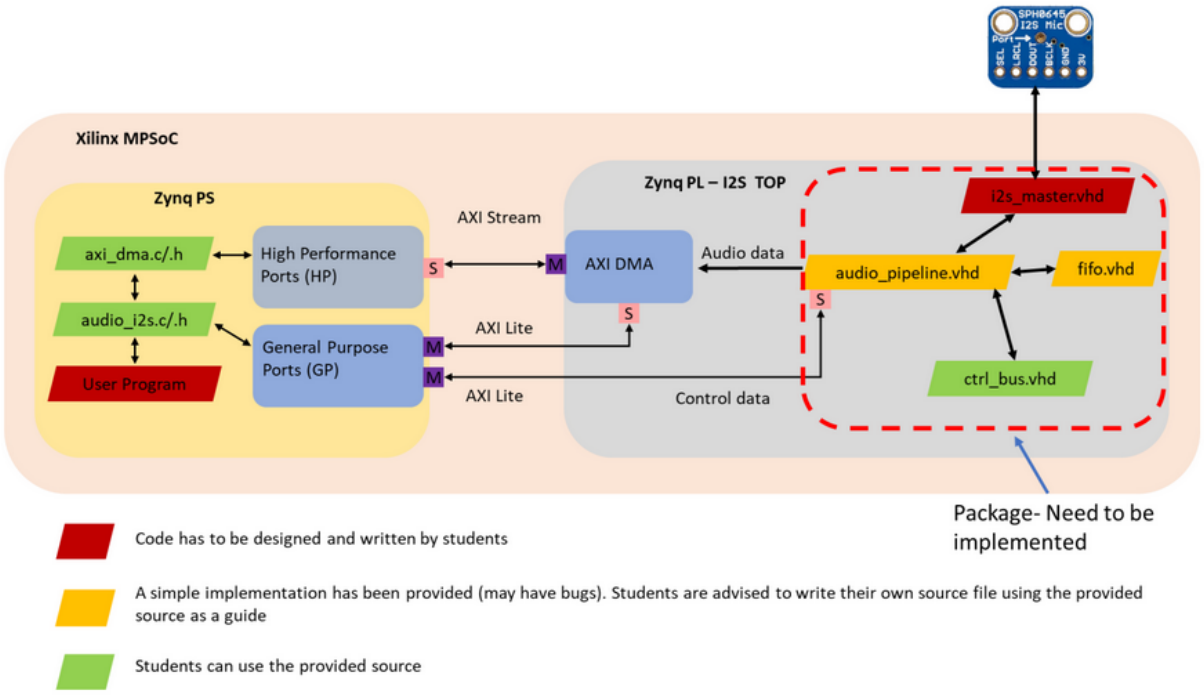


Figure 2: Basic Requirements Project Source Layout

You will be provided with a template project (Figure 2). You will be able to use some existing code, and will need to write some of your own. Once completed, you will have achieved the Basic Requirements - and you can then choose what to extend to achieve your Application! Several Application ideas are included in this specification document (Appendix 1).

Reference material:

- https://www.xilinx.com/support/download.html
- https://www.xilinx.com/products/som/kria/kv260-vision-starter-kit.html
- https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html
- https://www.adafruit.com/product/3421
- https://digilent.com/reference/pmod/pmodbb/reference-manual?redirect=1
- https://en.wikipedia.org/wiki/I%C2%B2S

Course Project Rubric / Marking Guidance:

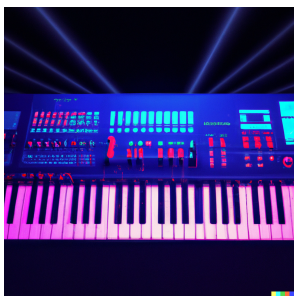
The project is divided up into a number of deliverables, divided up over a number of *milestones*. BR indicates a Basic Requirement (must be completed). Applications will be graded on an effort/difficulty scale, i.e. **they do not need to be at 100% to receive full credit**, but work must demonstrate good faith progress to the final goal. Markers should be convinced that given enough time, you *could* have finished the application. Please see Appendix 2 for more details.

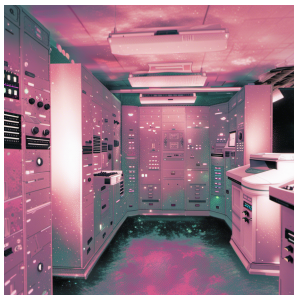
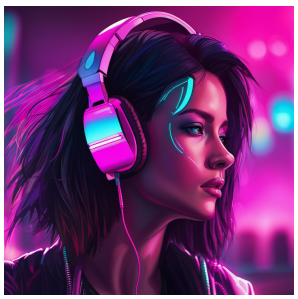
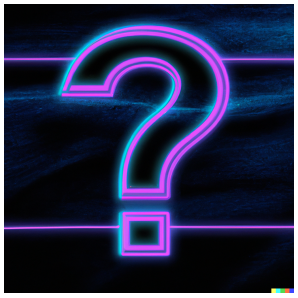
Note that two of the deliverables will be assessed individually. Good luck!

Course Objectives:

After completing this course, students will: gain experience in co-operating and delegating within a group, aid the design of a real system, gain research skills in design, analysis and optimization, gain experience in building systems, write design documentation, and learn to work with others within a limited time budget. Please see Appendix 3 for more details.

Appendix 1 - Application Ideas:

	**Hardware Audio Output:** In the basic requirements, we examine only what is required to capture audio to .wav files. Ask your TA for additional hardware for I2S output and a speaker (or pair of speakers - try for multiple channels!) In this application, you will build a similar pipeline to that already completed, but this time for the production of audio. You should make it such that it can either reproduce the audio from the mic. or play .wav files directly.
	**Sound Mixing:** Advanced audio applications need to be able to combine and adjust different audio tracks. In this application, consider adding the second microphone to your FPGA, then add support for rudimentary (or advanced!) adjustments: choosing between microphone channels, volume and frequency adjustments. You could have software or hardware control of this process (e.g. add expansion hardware with buttons and knobs).
	**Sound FX:** Some of the most distinctive villains in film history owe much of their identity to vocal / audio augmentation. Similarly to Sound Mixing above, in this application you could thus experiment with the kinds of filters and processing which are used to add effects to sound (e.g. echos, distortions, delays, tonal shifts). To implement these you will need to investigate various filters, such as FIR and IIR. Work towards goals such as real-time processing and implementations using custom hardware!
	**Musical Instrument:** As another variant of the above sound mixers/FX, music synthesizers can be created by the mixing of tones in hardware or software. In this application you would add support to your platform for the production of music by using off-chip hardware for the playable space. Consider your input options! You could use analog inputs (e.g. LDR) to make a quirky theramin, or use switches and keys for making an instrument closer to a keyboard or digital piano.
	**Compression:** Modern technology tends not to use .wav files for the transfer of audio - we instead compress the sound waves to better formats. This is most often done with custom hardware, e.g. mp3 encoder/decoder co-processors! In this application, you will explore how these are achieved, and likewise implement some kind of compression pipeline which works with your processor. For full marks, ensure that the files are playable using another device (e.g. VLC on a PC!)
	**Networking:** Modern audio setups often feature multiple devices connected via networks or the internet (single room theaters to international teleconferencing). Here, you will explore how your audio device may be made part of a greater whole. This could be done through the DEP platform provided by Audinate or via a custom stack that you define yourself. Your audio should be receivable / replayable by another machine!

 	**Security:** In World War II, the SIGSALY project was used to encrypt the highest level of Allied communications. It performed this by cryptographically scrambling the audio waves in hardware. In this project you could similarly protect your microphone data. To avoid the risks of software compromise in PetaLinux, you could add custom hardware to encrypt your audio signal before it is exposed to the processor. Of course, encrypting the data is no good if we can't decrypt it, so you should provide additional components (which could be in hardware or software) which are capable of later decoding your audio. For this purpose, you may consider investigating algorithms like AES, DES, RC5, or ASCON and design a way for them to be combined with the audio processing pipeline.
	**Voice Recognition / "Wake word":** While full speech-to-text transcription is unlikely to be able to run locally, you could combine a networking goal with an online transcription service such as OpenAI's 'Whisper'. However, what should be sent? Voice 'AI assistants' like Amazon Alexa have a 'wake word' which is processed locally before sending the audio stream to a server. In this application, you could try making one or other (or both) of these components, possibly utilizing statistical or AI models.
	**Sound Localization:** Many real-world products in the consumer and industrial space feature sound localization, where you determine the direction of an audio source (e.g. Amazon Alexa directional lights). Developing a system that can identify the direction from which a sound is coming can be challenging. You will need to combine multiple microphone inputs with signal processing and direction-of-arrival (DoA) algorithms.
	**Noise Cancellation: (warning - DIFFICULT)** Over-ear headphones are frequently equipped with noise cancellation technology, which is a system that uses inverse sound wave techniques to reduce or eliminate surrounding noise. By combining both 'hardware audio output' and 'sound effects' you can try out techniques for this purpose - by combining 'audio mixing' for a playback stream you're then well on the way to a real product!
	**Something else:** You might find that you have another idea from that provided above, and wish to try forge your own path forwards! In this course, that is completely fine: you will just need to coordinate with the teaching staff in advance so that we can verify that your application idea has a suitable scope and level of difficulty for the course.

Appendix 2 - Deliverables:

This course is divided into four milestones with the following deliverables:

Points (/100)	Individual / Group	Task/Feature	Milestone
10	Individual	BR: Custom hardware can be deployed to the MPSoC FPGA fabric *(10 pts) Demonstrated via in-lab demo to a tutor. Q&A to test understanding.*	M1 (wk3)
20	Group	BR: I2S Bus Master implemented correctly and with clean code. Some data can be received in the Zynq PS. *(10 pts) In-lab demo, showing group ability to capture audio to the Zynq PS.* *(5 pts) Preliminary project files (zip) and documentation.* *(5 pts) Preliminary project plan with desired application, timeline and assigned tasks.*	M2 (wk5)
20	Group	BR: A software program written with clean code running on the Zynq PS which saves the audio .wav files readable by other computers. *(15 pts) In-lab demo, showing all Basic Requirements and walkthrough of project architecture and files. Q&A to test understanding.* *(5 pts) BR project files (zip) and documentation.*	M3 (wk7)
10	Group	Application: Proposal and System Diagram / Specification / Design / Plan, with timeline + tasks. Provide justifications and “sell the solution”! *(10 pts) PDF interim report describing progress and motivating next steps and Application specification.*	M3 (wk7)
25	Group	Application: Implementation *(10 pts) In-lab demo of application.* *(10 pts) BR+Application project files (zip) with README and clean code, and include…* *(5 pts) …well-written documentation and software architecture diagrams. Describe what was provided and how it did/would have fit into the bigger picture of what the application intended.*	M4 (wk10)
15	Individual	Individual Report and Peer Review *(15 pts) Reflection on the project, on your individual contribution, on challenges encountered and solutions found, what went well and what could have gone better. Explain what the project intended to accomplish and your feelings on whether or not this was or could have been achieved. Describe how your efforts fitted into the broader team. Discuss how the team performed - the report should include a peer review table attachment.*	M4 (wk10)

Appendix 3 - Learning Principles

The course relates to the following learning principles:

ESY Embedded Systems	
ESY-1	History and overview
ESY-2	Relevant tools & standards
ESY-3	Characteristics of embedded Systems
ESY-6	Asynchronous and synchronous serial communications
ESY-7	Periodic interrupts, waveform generation, time measurement
ESY-8	Data acquisition, control, sensors, actuators
ESY-13	Computing platforms for embedded systems
SPE Systems and Project Engineering	
SPE-3	Project management principles
SPE-6	Hardware and software processes
SPE-7	Requirements analysis and elicitation
SPE-8	System specifications
SPE-9	System architectural design and evaluation
SPE-10	Concurrent hardware and software design
SPE-11	System integration, testing, and validation