Poster #: 115

Advancement of Audio Processing on PYNQ-Z2 Through the Implementation of Real-Time Speech-to-Text

StrathSDR

Student: Eva Peter Supervisor: Dr. Louise Crockett

University of Strathclyde Engineering

INTRODUCTION

Motivation was to create a low-power, offline transcription device which can be used anywhere. The speech-to-text (S2T) device designed to provide live time transcription of the spoken word. Objective of project was:

- Become familiar with ZYNQ, design hardware and software system;
- Evaluate S2T APIs and implement most appropriate on board;
- Utilise AMD's video and composable pipelines within design.

Targeted UN sustainable development goals are given in Figure 1.

- Few elements to the device and will not require WiFi or charging electricity, cheap to operate, accessible;
- Subtitles increase concentration and aide comprehension; proposed used in lectures
- Private Design: as all communication is internal, vital requirement of the defence sector.
- 87,000 people deaf in UK, only 25,000 able to use BSL: rely on lip-reading, written word or an interpreter.

Figure 1: Targeted UN Sustainable Development Goals and Why

SPEECH-TO-TEXT

The chosen S2T API was Rev.ai, highly accurate and transcribes local files offline. Offline transcription works through accessing a Recurrent Neural Network, then classifying using Connectionist Temporal Classification.

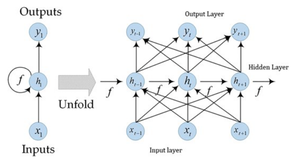

Figure 2: Layers of RNN

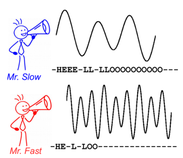

Figure 3: Variation in Input Speech

- *Figure 2* shows a RNN take previous outputs to improve learning
- *Figure 3* is a diagram of CTC reference data, CTC compares output text to training data: audio of varying speed, and elocution.

PYNQ

Python Productivity for ZYNQ (PYNQ) is an open source software framework implemented on the processing system of selected ZYNQ boards, PYNQ-Z2 was used within project. Figure 4 shows PYNQ interaction, Figure 5 is the framework.

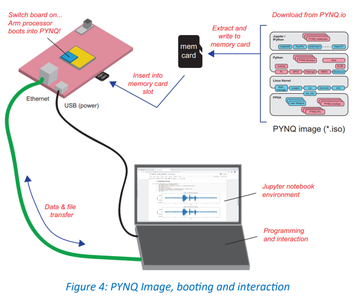

Figure 4: PYNQ Image, booting and interaction

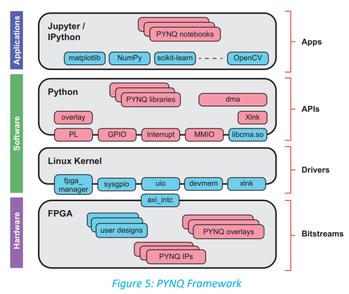

Figure 5: PYNQ Framework

Shown in Figure 6 are the utilised PYNQ-Z2 ports within the project design.

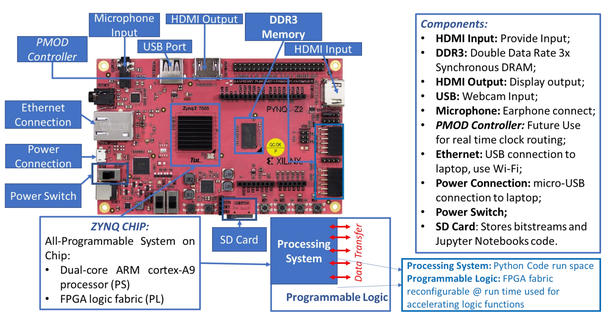

Figure 6: Utilised PYNQ Peripherals within Project

IMPLEMENTATION ONTO PYNQ-Z2

Internal System Overview Diagram, specific to design on Z2, in Figure 7.

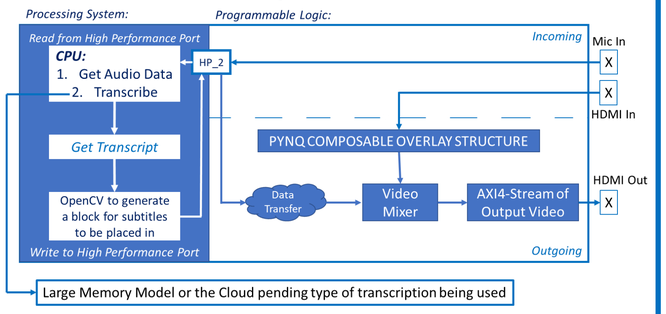

Figure 7: System Overview Diagram of Board

Audio input then is called within S2T function within processing system, a subtitled overlay is then created. Figure 8 shows the creation process.

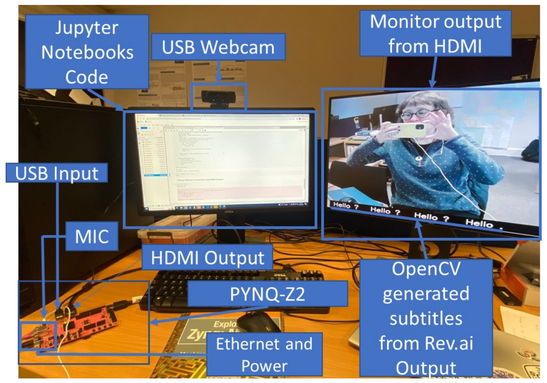

Figure 8: OpenCV Manipulation to Create Subtitled Overlay

Rev.ai is installed onto PYNQ-Z2 as a Python Software Development Kit, via Linux terminal in Jupyter Environment. Raw audio data is stored as a buffer, called within S2T functions. Audio buffer input shown in Figure 9.

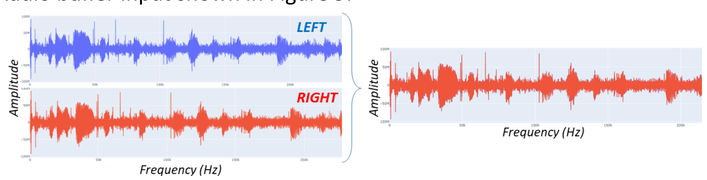

Figure 9: Raw audio read onto board.

Subtitled output is written to memory and streamed from Video Direct Memory Access to ***Video Mixer***, frame is overlaid on webcam. Block design in Figure 10.

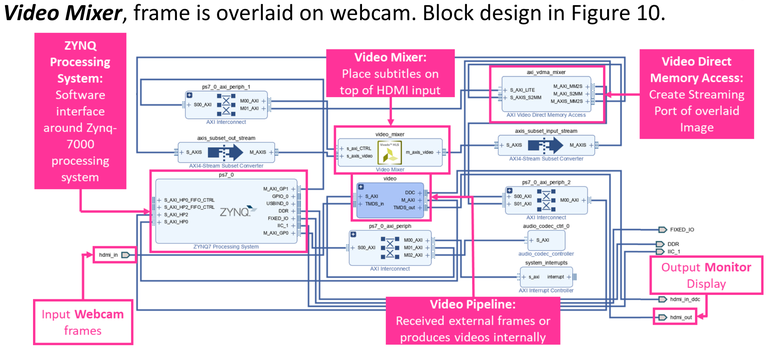

Figure 10: Hardware Design utilising Video Pipeline

RESULTS

FPGA RESOURCE FILES:

Resource table for the final design, Table 1. FPGAs have PL, routing, I/O and memory resources available for the compiler to implement HDL code.

Table 1: Resource Table for Final Hardware Design

Resource Type	Resource	Utilisation % of Available
Programmable Logic	LUT	36.4
	FF	28.9
	DSP	16.4
Memory Resources	LUTRAM	5.9
	BRAM	31.8
I/O Ports	IO	23.2

A large proportion of the resources were used as the design carries **video data** which is essentially multiple arrays of pixels contained within memory as frame buffers.

Achieved target clock ***142 MHz***: HD Video default, 1080 pixels @ 60 fps.

PYNQ DESIGN SPECIFIC:

Frames per second (fps) for hardware and software compared, Table 2. The human eye can detect fps between 30 and 60 fps; below 30, noticeable buffering.

Table 2: fps for each design

Design	fps
Python Software Lone	23.54
PYNQ Framework	25.1
PYNQ-Z2	60

Figure 11: Design Demonstration

PYNQ-Z2 produced a perfect frame rate, however it ran out of memory when secondary subtitled overlay layer was implemented.

SPEECH-TO-TEXT:

The Word Error Rate (WER) of Rev.ai was determined experimentally (shown in Table 3) to understand the impact different speakers have on the result. Other speech-to-text APIs are not tested, used to qualify final Rev.ai design.

Table 3: Determination of Word Error Rate for Numerous Speakers

Speaker Key Features		Number of Incorrectly Interpreted Words			WER
Gender	Nationality	Insertion	Deletion	Substitution	
Male	Scottish	0	0	7	0.09
Female	English	0	0	9	0.12
Female	Luxembourgish	0	2	9	0.14
Male	Irish	0	0	12	0.16
Male	Spanish	1	0	8	0.12

FURTHER WORK AND CONCLUSIONS

Learned within project how to design an embedded system using PYNQ PS and ZYNQ PL also understand the operation of S2T algorithms. An efficiently operating design was produced however hardware limitations prevented its utilisation, future work includes:

- Embed language model onto FPGA logic fabric to perform S2T;
- Use an MPSoC board (i.e., Ultra96 or ZUC105) which has more available memory for performing hardware acceleration;
- Look at downsizing physical components of design to become more portable.