UNIVERSITY OF SOUTHAMPTON

Speech Recognition on Embedded Hardware

by

Ricardo da Silva

Technical Report

Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

December 11, 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

by Ricardo da Silva

This work is all about . . . TODO: ABSTRACT

Contents

Chapter 1

Introduction

- Outline the motivation for this project - Brief overview of project goals

Chapter 2

Project Goals

At the highest level, the goal of this project is to develop a speech recognition system that will run on embedded hardware. In pursuing this goal, the aim is to achieve several other goals that will be beneficial to the author and to the University of Southampton.

2.1 Speech Recognition

The speech recognition system proposed aims to be run using an ARM chip doing pre-processing and an FPGA doing statistical decoding. Ideally, it will be capable of performing HMM based speaker-independent speech recognition. This, however, is very ambitious, and so initially the project will focus on getting the most basic forms of the relevant algorithms running in software. Once this is complete, the system will be built into hardware, and extended to be more complete.

2.2 The Micro Arcana

In terms of hardware, one of the project goals is to further development of the Micro Arcana family of boards, currently under development by Dr Steve Gunn. The speech recognition system proposed will make use of two of the boards – the "L'Imperatrice" ARM based mini-computer, and the "La Papessa" FPGA board. Part of the project is setting up and configuring these two boards, and having a finished system will help demonstrate their capabilities.

2.3 Theoretical understanding

A major goal of the project is to develop a higher level of understanding of the algorithms used in speech recognition, and to get experience designing a large-scale embedded application. This complements the intrests of the author and the subjects being studied, in particular, Intelligent Algorithms and Digital Systems Design.

Chapter 3

Background and Investigations

3.1 Speech Recognition Systems

In general, 'Speech Recognition' refers to the process of translating spoken words or phrases into a form that can be understood by an electronic system, which usually means using mathematical models and methods to process and then decode the sound signal. Translating a speech waveform into this form typically requires three main steps [8]. The raw waveform must be converted into an 'observation vector', which is a set of data that is compatible with the chosen speech model. This data is then sent through a decoder, which attempts to recognise which words or sub-word units were spoken. These are then sent through a language model, which imposes rules on what combinations of words of syntax are allowed. This project aims to focus on the first and second tasks, as they are the more interesting from an electronic engineering point of view.

There are a variety of different methods and models that have been used to perform speech recognition. An overview of the most popular will be described here, along with the chosen approach.

3.1.1 Tor's Algorithm

The author first became interested in speech recognition when reading about "Tor's Algorithm", which is a very simple small dictionary speech recognition system [4]. This algorithm is capable of very accurate speaker dependent speech recognition for a dictionary of less than ten words. It is based on a fingerprinting model where each word in the dictionary must be trained to form an acoustic 'fingerprint'. This fingerprint is based on the time variations of the speech signal after being filtered appropriately. Then recognition is reduced to finding the Euclidean distance squared between the input vector and each of the stored fingerprints. The 'closest' match is the word with the smallest distance from the input. Although this system is very simplistic, it outlines two major

components of any speech recognition system – pre-processing and decoding (recognition). More complex systems just use more complex speech models and pre-processing methods.

3.1.2 Dynamic Time Warping

Speech, by nature, is not constrained to be at a certain speed – the duration of words will vary between utterance, and a speech recognition system should be able to handle this. Dynamic Time Warping (DTW) is essentially the process of expanding and contracting the time axis, so that waveforms may be compared, independent of talking speed. Combined with a dynamic programming technique for finding the optimal 'warp' amount, it became a widely used approach to solving the problem of speech duration modelling [6]. One useful property of DTW is that it may offer good performance even with little training, as it only needs one word as a template [8]. Conversly, the performance of DTW based systems cannot be increased much with more training, unlike Hidden Markov Models.

3.1.3 HMMs

By far the most prevalent and successful approach to modern speech recognition uses Hidden Markov Models (HMMs) for the statistical modelling and decoding of speech [5]. The flexibility inherent in HMMs is key to their success, as a system can be made more and more accurate by simply improving the HMM models or training the models further. The classic tutorial paper by Rabiner ([10]) is one of the best references for HMMs in speech recognition, and provides a very good overview of modern systems. The following three sections are based heavily on [10] and [13].

3.1.3.1 Single Word HMMs

The simplest HMM based systems use a single HMM for every word in the recognition dictionary. Given a set of observations, each HMM can be scored based on the probability that it would output the observations. The HMM with the highest score is taken as the recognised word. The most apparent limitation of this system is that a very large amount of training would be required if a dictionary of any size was to be used. At the very least, one sample of each word would need to be recorded to train the full system, which would be a very time consuming process. However, for simple applications (voice dialling, for example) this is manageable.

3.1.3.2 Sub-word HMMs

The next step up in complexity from single word HMMs is models that consider sub-word utterances (phonemes). This allows a smaller set of HMMs to be used for much larger dictionary recognition, as words are recognised based on sequences of sub-word HMMs. Thus instead of searching through a single HMM to recognise a word, the recognition process becomes a search through a trellis of multiple HMMs in order to find the best path through them. The most simple HMM system of this form is based on monophones, of which there are about 50 in English. This may be improved by using bi- or tri-phone HMMs, which model transitions between two or three monophones. Using this form of HMM will increase the acoustic model size greatly however, as there are many possible combinations of monophones in the English language.

3.1.3.3 Viterbi Decoding

For most HMM models there are three problems:

- Training the model
- Finding the probability that a model produced a given observation sequence
- Finding the 'best' path through a model to produce a given observation sequence

The 'best' path is generally taken to be the path with highest probability, and it is this problem that is central to the project. The first problem is solved by using Voxforge (3.3), and the second problem is more important for word based recognisers.

In all literature encountered, the Viterbi algorithm is the primary method for solving problem 3. It is an iterative approach to solving the optimisation problem, and has the added bonus that not much data needs to be stored during the calculation [11]. A full explanation of the Viterbi decoding process is available from [10],[8],[12].

3.2 Speech Pre-Processing

Speech signals are complex waveforms and cannot be processed without some form of feature extraction which reduces the complexity while retaining the important features. In modern speech recognition systems the two most common methods of analysing and representing speech are: [7]

- Linear Predictive Coding (LPC)
- Mel-Frequency Cepstrum Coefficients (MFCC)

Both these methods attempt to model the movement and dynamics of the human vocal tract and auditory perception. LPC is more suited to speaker recognition (the process of identifying voices, rather than speech), while MFCCs are more useful for speech recognition [2].

The Mel-Frequency Cepstrum is based on a filterbank analysis with a cepstral transformation, which is required due to the high correlation between filterbank amplitudes. The human ear perceives sound on a non-linear frequency scale, and one way of improving recognition performance is by using a similar scale for analysis of speech. A filterbank analysis can be used to perform this discrimination between different frequencies, and the frequency bins are usually spaced using the Mel frequency scale. However, the filterbank amplitudes are highly correlated, which greatly increases the computational complexity of the HMM based recogniser as the covariance matrix will not be diagonal. In order to correct this, a discrete cosine transform is taken on the log filterbank amplitudes, finally resulting in a set of Mel Frequency Cepstrum Coefficients. The HTK (3.3) defaults to using twelve MFCC filterbank bins. [13] [8]

In order to attain MFCCs, a sampling rate must be chosen such that enough data is gathered while allowing sufficient processing time. In addition, to perform Fourier transforms on the speech, the incoming signal must be windowed appropriately. The HTK has a set of default values for these parameters, which are assumed to be appropriate.

An improvement to both LPC and MFCCs is to compute time derivatives in the feature extraction process, which gives a better idea of how the signal changes over time. In addition, the log energy of each sample may also be computed to also boost recognition ability.

3.3 The HTK and VoxForge

The Hidden Markov Model Toolkit (HTK) is a set of tools and libraries for developing and testing Hidden Markov Models (HMMs), primarily for speech processing and recognition tools [13]. Given a model structure and a set of transcribed speech recordings (a 'speech corpus'), a set of HMMs may be trained using the HTK. This includes performing all pre-processing in a number of formats, and testing recognition capabilities of a model.

Voxforge is an open source speech corpus which is aimed at facilitating speech recognition development. It provides pre-compiled acoustic models – essentially large sets of HMMs – in the format created by HTK, licensed under the GPL (GNU General Public License) [3]. The alternative would be to use another speech corpus (such as TIMIT), and then use the HTK to design and train the acoustic model. This is potentially a very time consuming process, so Voxforge is useful because it essentially cuts this step out. In

addition, the Voxforge models may be easily adapted to a specific person's voice using only a few minutes of transcribed speech. However, the drawback is that the Voxforge model is very complex (8000 tri-phone HMMs with Gaussian output probabilities, 25 coefficient observation vectors). Implementing a recogniser system based on this model will require a lot more work than if a simpler model was used, such as one based on discrete (output probability) HMMs.

3.4 Embedded hardware and Speech Silicon

A wide range of speech recognition software (commercial and open source) exists for desktop PCs or laptops. However, speech recognition for embedded systems is less widespread. Recently there has been increased research into the use of DSPs and FPGAs for speech recognition [8], [11], [9]. Of particular interest is Stephen Melnikoff's PhD Thesis, and the Speech Silicon architecture. The former investigates a variety of HMM based recognisers on an FPGA, using a PC to perform pre and post processing. The latter details a flexible FPGA based system capable of performing recognition on medium-sized vocabularies.

Chapter 4

Technical Progress

4.1 System Design

The main progress so far has been made in the area of speech recognition research and overall system structure. At this stage there is a fairly solid idea of the final system, and only implementation remains.

Initially the aim was to build a small dictionary recogniser, based on methods similar to those used by Tor [4]. However, further research showed that a phoneme based system would be far more useful and interesting to build. The primary steps to such a system are:

- Pre-processing (convert waveform to MFCCs)
- Observation probability calculations (based on Gaussian output distributions for each state)
- Decoding to find the best chain of states and HMMs (Viterbi decoder preferable)
- Results compilation (backtrack HMMs and compile into words)

There are a huge number of design choices that must be made for a speech recognition system. In designing the project's system, it was important that it is implementable in the time given, will challenge the author's abilities, and has potential for extendability or further work. The existence of the Voxforge acoustic models was also a strong influence. Not only do they remove the need to spend time training models, but also they may be used with other speech recognition software in order to test the project's performance.

The main focus has been on developing solutions for the second and third steps in the above list. The system proposed to run on the FPGA will contain two main blocks – the probability calculator and the viterbi decoder.

4.2 Model training

The acoustic models from Voxforge (3.3) have proven to be very useful. Using the HTK, they have been adapted to the author's voice, and a complete set of HMMs is now ready to be used. The model uses:

- 7094 States
- 8309 HMM definitions (each with 3 outputting states)
- 51 monophones

The term 'outputting states' refers to states that produce an observation. The HMMs have 5 states in total, but the first and last are non-emitting. The transition probability from the fourth to last state is the probability of exiting that particular HMM. Similar to the approach taken by [8], the first state of all the HMMs is ignored for implementation, as it always transitions to the second state with a probability of one.

4.3 Pre-Processing Tools

The chosen method of pre-processing speech is MFCCs, as they are a compact and simple representation, but are good at modelling speech acoustics. For training and testing, the HTK is capable of producing MFCCs with a variety of options (time derivatives, absolute energy, etc) for a given waveform.

For the final product, the goal is to build a dedicated pre-processing module which will run on the L'Imperatrice board and send data to the FPGA. A library exists for computing MFCCs [1], and there are many Fast Fourier Transform libraries available. A combination of FFTW [2] and LibMFCC will be used to perform this task, and custom code will need to be written to calculate the MFCC time derivatives.

4.4 Code Prototypes

The first goal of the project is to produce software that performs the statistical calculations that will ultimately be built into hardware. As detailed in the System Design section (4.1), this is comprised of two main parts – the output probability calculations, and the Viterbi decoding. So far, the output probability calculations have been implemented in C, which is the first block of the system. In addition, a script to extract the requried data from HTK files has been created. This includes the HMM definitions, along with associated gaussian distribution parameters, and MFCC file data.

[1] libmfcc is a C library for computing MFCCs (http://code.google.com/p/libmfcc/)
[2] FFTW: Open source FFT library (http://www.fftw.org/)

4.5 Hardware Development Environment

As the Micro Arcana is still under active development, part of the project involves setting up and testing the two boards that the project aims to use.

4.5.1 La Papessa

In order to facilitate the development of code on the La Papessa board, several combinations of software environments were explored. The board is based on a Xilinx Spartan 3AN FPGA, which is compatible with the Xilinx ISE Webpack design software package [1]. However, one drawback to the ISE Webpack is its lack of support for synthesis in SystemVerilog. Synplify Pro/Premier is an alternative HDL synthesis tool, which is compatible with the Xilinx software toolchain and also supports SystemVerilog. For this reason, and because the author is more familiar with the Synplify design flow, it was decided that Synplify Premier would be used for synthesis for the project.

However, there is limited documentation on getting synplify to work with the ISE tools. This has been accomplished and tested with very simple code (a binary counter in SystemVerilog), and works correctly. However, with larger designs it may be that this design flow may need to be tweaked or adapted. The fallback is to use the ISE Webpack for everything, so the code will be written such that it is very close to Verilog, and thus easy to adapt for the ISE tools.

4.5.2 L'Imperatrice

The ARM based L'Imperatrice board is still under active development, and several important features are not yet working on it. To be used for the project, the following items are critical:

- GPIO functionality
- Native or cross compiler set-up
- Microphone input access

An LTIB (Linux Target Image Builder) environment has been installed on an Ubuntu virtual machine, which is to be used for setting up the board support package (BSP). This has been used to build and test various kernel configurations, TODO: check cross compiler etc In addition to LTIB, the ArchLinux build system (ABS) has been investigated as a potential alternative to LTIB. The primary advantage of the ABS is that only a small number of files need to be distributed, which, when run, will download and compile all dependencies of the build. An ABS configuration exists for the Olinuxino,

a linux board also based on the Freescale iMX23, which may be tweaked to suit the L'Imperatrice. However, due to the (perceived) relative ease of the LTIB set-up, this has not been done yet.

Chapter 5

Plan of Remaining Work

5.1 Overview

Since the start of the project there has been a fair amount of decision changes, as the scope and challenges of the project became more clear. The initial plan (Figure 5.1) was unrealistic in some senses, and a revised Gantt chart [1] has been created for the remaining work (Figure 5.2).

A notable difference between the first and second Gantt charts is that the first made far fewer attempts to break the implementation work up. In addition, it underestimated the time that would be required to set up the L'Imperatrice board. Finally, more time has been allowed to do the project report, as more time should have been given to writing the interim report.

The second plan is designed to split the remaining work in such a way that there is less dependence between tasks. If at all possible the work is modularised, so that if one section becomes unfeasible, it can be dropped without affecting the outcome of the final product greatly.

5.2 Details of tasks

The implementation of the full system requires several smaller parts,

[1] Both Gantt charts were created and modified using Gantter (http://app.gantter.com)

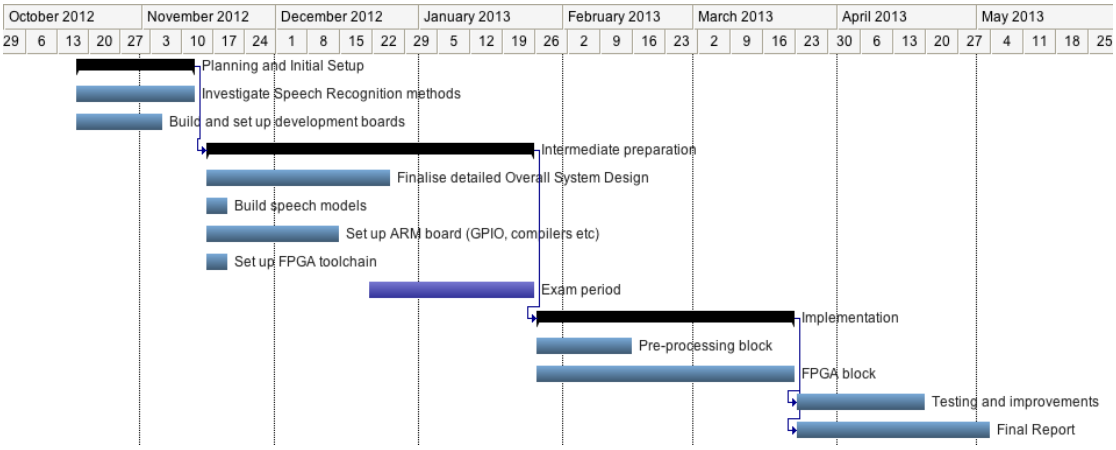

FIGURE 5.1: First Gantt chart with high expectations

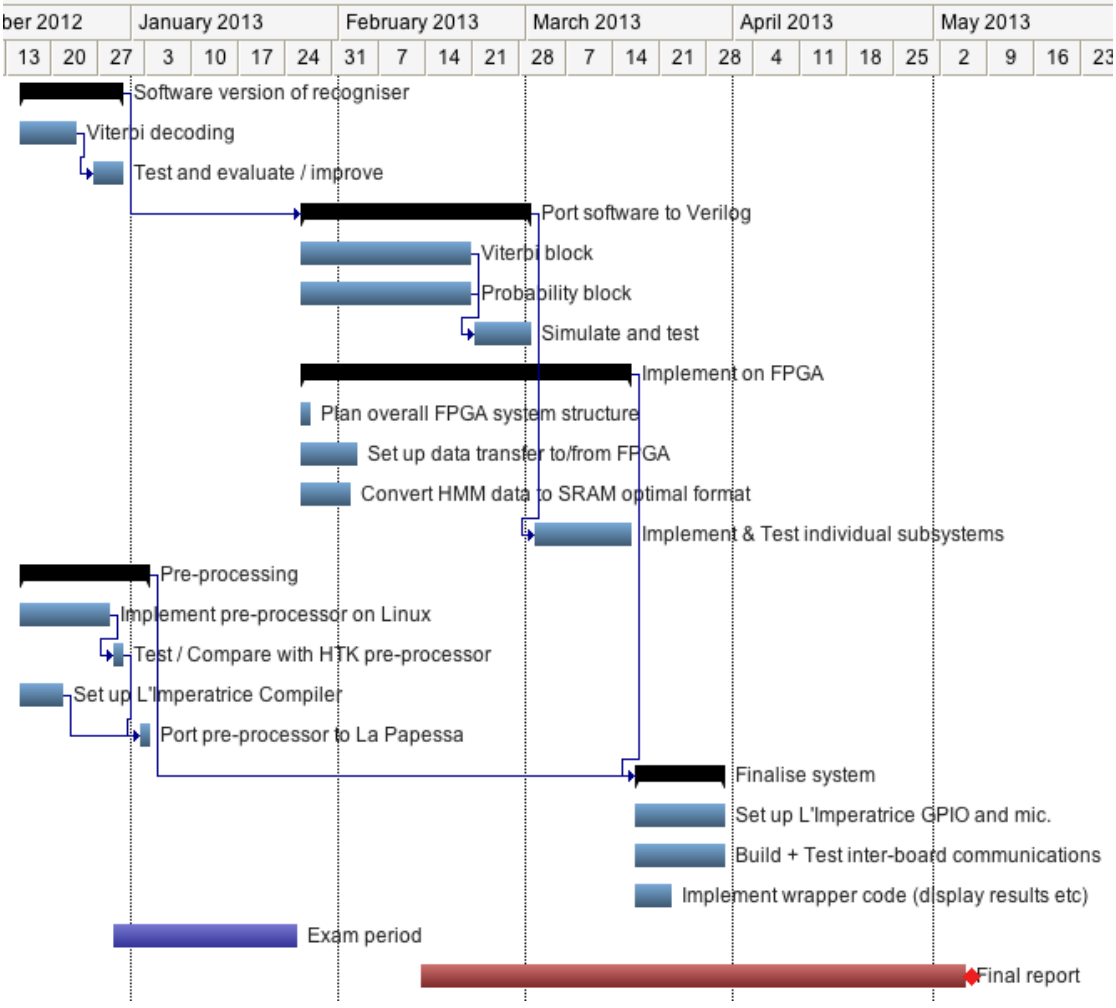

FIGURE 5.2: Interim Gantt chart for remaining work

Bibliography

[1] Xilinx ise.

[2] Personal interview with srinandan dasmahapatra, 11 2012.

[3] Voxforge, 2012.

[4] Tor Aamodt. A simple speech recognition algorithm for ece341, 04 2003.

[5] SJ Cox and G. Britain. *Hidden Markov models for automatic speech recognition: theory and application*. Royal Signals & Radar Establishment, 1988.

[6] Sadaoki Furui. *Digital Speech Processing, Synthesis, and Recognition*. Marcel Dekker, 1989.

[7] S.K. Gaikwad, B.W. Gawali, and P. Yannawar. A review on speech recognition technique. *International Journal of Computer Applications IJCA*, 10(3):24–28, 2010.

[8] S.J. Melnikoff. *Speech recognition in programmable logic*. PhD thesis, University of Birmingham, 2003.

[9] S. Nedevschi, R.K. Patra, and E.A. Brewer. Hardware speech recognition for user interfaces in low cost, low power devices. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 684–689. IEEE, 2005.

[10] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[11] J. Schuster, K. Gupta, R. Hoare, and A.K. Jones. Speech silicon: an fpga architecture for real-time hidden markov-model-based speech recognition. *EURASIP Journal on Embedded Systems*, 2006(1):10–10, 2006.

[12] Saeed V. Vaseghi. *Advanced Digital Signal Processing*. John Wiley and Sons, fourth edition, 2008.

[13] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK Book*. Cambridge University Engineering Department, 2009.