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

April 8, 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

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

by Ricardo da Silva

This report presents a proof of concept system that is aimed at investigating methods of performing speech recognition in embedded systems. It makes use of two new electronic boards that are currently under development at the University of Southampton, and implements one part of a speech recognition system using a Spartan 3 FPGA and a ARMv5 Linux applications processor.

Contents

Acknowledgements

Thanks to Steve Gunn, Srinandan Dasmahapatra

Chapter 1

Introduction

1.1 Goals

At the highest level, the primary goal of this project is to implement part of a modern HMM based speech recognition system, with the constraint that it must be done 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.

1.1.1 Speech Recognition

Most modern HMM based speech recognition systems are extremely complex, and can take years to design and optimise for a particular implementation. The goal of this project is not to implement such a system, but rather to explore the possibilities of what may be achieved with a low power applications processor and a relatively small FPGA. In particular, the goal is to use the applications processor (running Linux) to perform pre-processing of speech data, and use the FPGA to perform mathematical manipulations of the observations. It is hoped that the system developed here may be later used as a basis for future research into the subject.

1.1.2 The Micro Arcana

The Micro Arcana is a new hardware platform aimed at undergraduate students, currently under development by Dr Steve Gunn. In terms of hardware, one of the project goals is to further development of the Micro Arcana family of boards, and provide a valuable example of how they may be usefully combined. The aim is to build the entire speech recognition system on two of the Micro Arcana boards, making it a self-contained embedded design. In addition, part of the project is setting up and configuring these two boards, so that they may be easily picked up by undergraduates.

1.1.3 Theoretical understanding

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

1.2 Motivation

Speech recognition is an interesting computational problem, for which there is no fool-proof solution at this time. Recently the industry for embedded devices and small-scale digital systems has expanded greatly, but in general these devices do not have the power or speed to run speech recognition. Field Programmable Gate Arrays (FPGAs) may present a way of increasing the capability of such systems, as they are able to perform calculations much faster than traditional microprocessors. The author's personal interest in embedded systems, combined with the challenges of a complex system such as speech recognition, makes this an appealing area to explore.

As hardware platforms go, most of the Micro Arcana is still very new and untested, as it is still under development. In addition, there are not many examples of how they may be used, and very little documentation. In order to improve their reception by students, it would greatly help to have proven use cases and examples of how these boards may be used individually and together. Using a larger FPGA (such as an Altera DE board) was considered during the planning stage of this project, but it was decided that it would be more beneficial and interesting to develop and use the Micro Arcana.

1.3 Results and Personal Contribution

The project implements one part of a modern speech recognition system, using two development boards from the Micro Arcana family. It is designed to be a proof of concept exercise, in order to explore the capabilities of the boards, and expand the author's knowledge of the relevant systems. Specifically, the project required substantial research into HMM based speech recognition systems, embedded Linux, and digital design. The resulting system, described in detail in Chapter 4, uses an FPGA to perform the most computationally expensive part of HMM based recognisers – scoring the states of each HMM model for a given input vector. Essentially, the ARM based "L'Imperatrice" is used as the application controller, and is connected to the FPGA based "La Papessa" board which performs the CPU intensive mathematical calculations. The embedded Linux processor reads WAV format speech files, performs the necessary pre-processing and sends observation vectors to the FPGA. Given an observation, the FPGA will process it and send back scores for each state in the speech

model, which represent the probability of that state creating the observation. Given these scores, the next step for a speech recogniser would be to perform Viterbi decoding (possibly using a token-passing or similar algorithm) in order to find the most probable sequence of HMMs, and thus eventually find the most probable word or phoneme sequence spoken.

Chapter 2

Background

2.1 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 representative 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 focusses on implementing pre-processing, and the first stage of the decoder, as these are interesting tasks 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, and then the chosen technique (HMMs) is described in Section 2.2.

2.1.1 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 [3]. This algorithm is capable of very accurate speaker dependent speech recognition for a dictionary of about 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 'best' 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 essentially just use more complex speech models and pre-processing methods.

2.1.2 Dynamic Time Warping

Speech, by nature, is not constrained to be at a certain speed – the duration of words will vary between utterances, 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.

2.2 Hidden Markov Models

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. However, a brief summary of the fundamentals of HMMs is given here. The following sections are based heavily on [10] and [14].

An N-state Markov Chain can be described as a finite state machine of N nodes with an $N \times N$ matrix of probabilities which define the transitions between each state. According to the notation in [10], the elements of this matrix are defined as $a_{ij} = P(\text{state at time } t = j | \text{state at time } t - 1 = i)$. To make this a 'Hidden' Markov Model, each state is assigned an emission probability for every possible observation, which defines how likely that state will emit that observation. In this case, the actual position in the state machine is unknown – only the state emissions (thus 'Hidden Markov Model'). The probability that a state j will emit observation O is defined as $b_j(O)$, and may be either a discrete value or a continuous distribution depending on the nature of the observations. Thus, an HMM is defined entirely by the matrices a and b, and a set of initial probabilities for each state, π, collectively denoted as $\lambda(\pi, a, b)$.

For speech recognition, the performance is substantially improved by using continuous HMMs, as it removes the need to quantise the speech data which is, by nature, continuous. !!TODOcite!!

A common distribution used for continuous probabilities is the multivariate Gaussian Mixture, which is essentially a weighted summation of several different Normal distributions [4]. However, for use in HMMs, the computational complexity is greatly reduced if the covariance matrix is diagonal (ie the components of each gaussian are uncorrelated). This requirement can lead to requiring extra pre-processing of observation data in order to remove correlation between the components.

2.2.1 Levels of Complexity

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 substantial 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.

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 mono-phones, of which there are about 50 in the English language.

Even more complexity (and, potentially, recognition accuracy) can be introduced by using bi- or tri-phone HMMs, which model transitions between two or three mono-phones. Using this form of HMM will increase the acoustic model size greatly however, as there are many possible combinations of mono-phones in the English language. However, it allows context dependent scoring of phonemes, including HMMs that model word endings and starts, or silences. In the Sphinx 3 recognition engine, the internal states of these HMMs are referred to as 'Senones', and the term has been adopted and used extensively in this project [12].

2.2.2 The HMM tasks

For an HMM model, λ, there are usually three important problems:

- Training the model to accurately represent real data
- Finding the probability that an HMM produced a given observation sequence, $P(\lambda|O)$

- Finding the 'best' path through a trellis of HMMs to produce a given observation sequence !!TODOequation!!

For this project, the first problem is solved by using Voxforge (2.4). The second problem is potentially very computationally expensive, as the speech model may be complex or large. In particular, this step requires scoring the senones of each HMM for every new observation frame, which is particularly time consuming if the HMMs have continuous output distributions. !!TODO: Speech silicon has refs to confirm this is most expensive!!

In all literature encountered, the Viterbi algorithm is the preferred 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]. This problem is beyond the scope of the current project, but a full explanation of the Viterbi decoding process is available from [10],[8],[13].

2.3 Speech Pre-Processing

Speech signals are complex waveforms and cannot be processed without some form of feature extraction which reduces the complexity whilst 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), whilst MFCCs are more useful for speech recognition [1].

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 (2.4) defaults to using twelve MFCC filterbank bins. [14] [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.

2.4 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 [14]. 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) [2]. 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. The Voxforge model is very complex, with 8000 tri-phone context-dependent HMMs with multivariate Gaussian output probabilities. Thus, implementing a recogniser system based on this model requires a lot more work than if a simpler model was used, such as one based on discrete (output probability) HMMs. However, modern speech recognisers are likely to use a model that is as complex, if not more so.

2.5 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.

Both Melnikoff and Speech Silicon perform an in-depth analysis of an entire speech recognition system based on programmable logic. Both of them require relatively large FPGAs, and

Chapter 3

Design Theory and Approach

3.1 The HMM based model

Due to the flexibility of HMMs, and the complexity of speech, there have been several different approaches to building speech models (the sheer size of the HTK book indicates how much flexibility exists). However, at this stage the author is more interested in the implementation of the algorithms, rather than devising the best way of modelling speech. Therefore, it was decided to use the pre-designed models from Voxforge for this project, and build the hardware to work with these models. Thus, various parameters were fixed from the start, including:

- Sampling rate of audio: 8kHz (Low Pass Filter with $\omega_0 = 4kHz$ required).
- Window size: 25ms (duration of observation frames).
- Frame period: 10ms (time between observation frames).
- Pre-processing output: 12 MFCCs, 12 MFCC derivatives, 1 Energy.
- Output probabilities: Single Gaussian distribution, 25-element mean and variance vectors.
- Number of monophones: 51 (Includes a monophone for silence. This is also the number of transition matrices).
- Number of senones: 7000.
- Number of HMMs: 8300[1].

The only modification made to the Voxforge models was that they were adapted for the author's voice, primarily to gain confidence with using the HTK and HMMs. Please see

[1]There are more HMMs than senones because some senones are used in more than one HMM

Appendix !!TODO!! for the scripts and HTK configuration files used to generate these models.

The senones are context dependent, that is, there are many different senones for each monophone, each with different predecessor and successor monophones. Most of the HMMs have 3 emitting states, and 2 states used to model entering and leaving a certain sequence. The number of transition matrices is equal to the number of monophones because...

3.2 Detailed Goal Specification

As described in previous sections, the FPGA is intended to be used for scoring every senone in the model, for every observation vector. In this system the new vectors arrive once every 10ms, and there are about 7000 senones that must be evaluated. The mathematical operations required to do this are now outlined.

Each senone j has an N-element vector of means, μ_j, and a $N \times N$ matrix of covariances, σ_j. However, since MFCCs are uncorrelated, the covariance matrix is diagonal, and σ_j is taken as an N-element vector. If the observation vector at time t is denoted as $\mathbf{O}_t = O_{t1}, O_{t2}, ..., O_{tN}$, then the score of senone j is given in Equation 3.1. However, the hardware complexity may be greatly reduced by taking the logs of both sides, removing the requirement to evaluate N exponentials for each senone. In addition, several parts of the equation may be precomputed, thus reducing the necessary sequence of operations to subtract, square, multiply, and accumulate. This derivation is shown in Equations 3.2–3.3, with the final result being the one most suited to hardware implementation.

$$b_j(O_t) = N(O_t; \mu_j, \sigma_j) = product... \tag{3.1}$$

$$lnofbothsides \tag{3.2}$$

$$finalequationwithsubs(TODO) \tag{3.3}$$

The hardware related goals outlined in Chapters 1 and 2 can be summarised as:

- Design a system in programmable logic that can efficiently evaluate Equation !!!.
- Design a C program to pre-process speech data according to the required form described in Section 3.1.

3.3 FPGAs

3.4 System Overview

The overall system layout, shown in Figure 3.1 is comprised of two primary blocks – L'Imperatrice and La Papessa (referred to from now on as "The processor" and "The FPGA" respectively). The entire system is powered from a single supply connected to the processor board's battery connector, and the FPGA is powered through ribbon cable connecting the two boards. This was done in order to minimise the amount of external circuitry needed, and to show that the two devices are able to work together fairly easily.

The two boards communicate over UART at approximately !!!11.5kbps!!! (115200 baud rate).

3.5 Analysis of Solution

3.5.1 The FPGA

The FPGA used was very small, and the full required design could not fit on it. In particular, the size of the model had to be reduced, so that each Senone had fewer than 25 components, and not all 7000 senones were processed. This is mainly due to the small amount of onboard RAM on the FPGA, so the design would greatly benefit from having more RAM available.

3.5.2 The processor

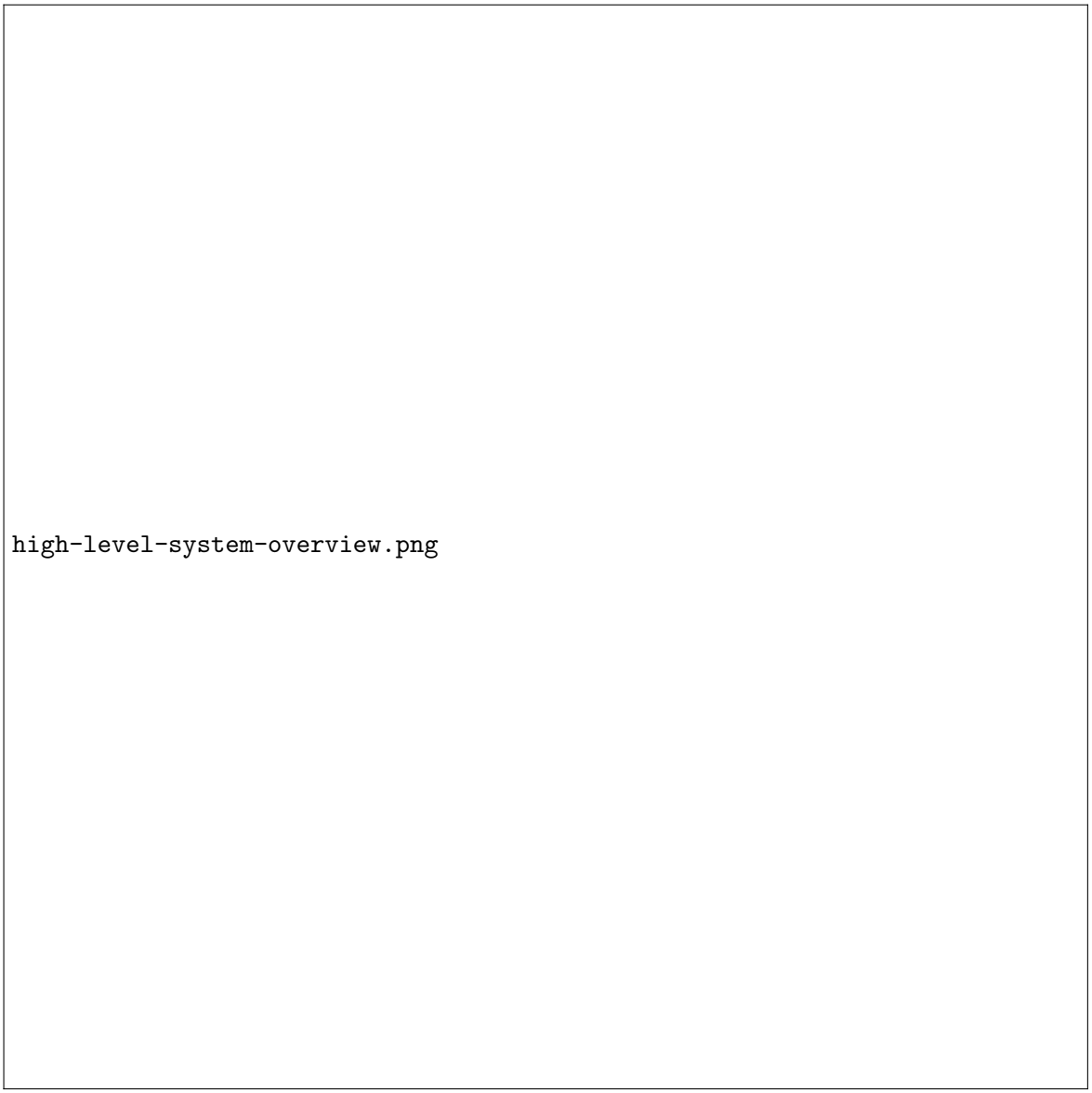

FIGURE 3.1: High level system layout overview

Chapter 4

Design Detail

A detailed description of the implemented design is given in this chapter, expanding on the outline given in the previous chapter.

4.1 La Papessa (The FPGA)

The FPGA on La Papessa is a Xilinx Spartan XC3S50AN, which is the smallest of its family, and is designed to be used with the Xilinx ISE Webpack software. Except for simulation, this software handles the entire design cycle, the most important operations being IO port mapping, synthesis, programming file generation (in .bit format). Unfortunately, the software only supports synthesis of Verilog or VHDL (!!??!!), and not the author's preferred hardware description language, SystemVerilog. Due to the advantages gained from using SystemVerilog, it was decided to use Synopsys Synplify to perform the synthesis. Besides being syntatically more powerful, SystemVerilog is the HDL that is currently taught to all new undergraduates at the University of Southampton. Having some documentation of a proven way to use SystemVerilog with this board would improve its reception and usage. Finally, SystemVerilog has advantages over Verilog for verification and simulation, which will be used to improve the design. The other design tasks (port mapping, programming file generation) are accomplished with ISE Webpack (See Appendix !!! for detailed description of this process).

4.2 Number Format

4.2.1 Top Level Module

A simplified hierarchical diagram of the top level module is given in Figure **??**, showing the main components of the system. This module included the main controller logic, which

FIGURE 4.1: Hierarchical diagram of Top Level Module

essentially waited for a new observation vector, then cycled through the necessary states. Figure **??** shows an ASM of this logic, and also outlines the main areas of the design that need explanation.

The top level module is also responsible for handling access to the on-board SRAM chip, which several modules need to write or read from. It essentially multiplexes the required signals, and leaves them floating (high impedance) when they are not needed. The 'Debug signals' shown in Figure **??** are a number of internal signals that are routed to output ports in order to facilitate hardware debugging.

4.2.2 Gaussian Distance Pipeline

The Gaussian Distance Pipeline (GDP) is the core component of the system, and computes Equation 3.3. It is a relatively simple 4-stage pipeline, with one stage for every step in the equation (subtract, square, scale, accumulate). Although the gains from using a pipeline in this case are fairly small, it would be very useful if more complex models were used. The Speech Silicon [11] project had a substantially more complex GDP, as their senones have several Gaussian distributions that must be mixed to produce the final output distribution.

The actual pipeline is fed by a controller which cycles through the senones and their components, sequentially feeding values into the pipe. This is a very simple state machine of only two states (IDLE and LOADGDP), which begins feeding the pipe when a 'new vector available' input flag is asserted. A 'last senone' output flag is asserted when the GDP has finished processing the last senone. This signal instructs the Normaliser module to begin its operation.

4.2.3 Normaliser

The Speech Silicon architecture included a module which normalised the senones before they were used for decoding. A very similar module is implemented here to perform a normalisation of senone scores. The highest score is found while scores are being evaluated, and then this score is subtracted from all the final scores. This causes the senone with the highest score to have a score of 0, which corresponds to a probablity of 1 ($e^0 = 1$).

4.3 L'Imperatrice (The processor)

4.4 LTIB

Chapter 5

System Testing

Chapter 6

Project Management

Chapter 7

Conclusions and Future Work

Bibliography

[1] Personal interview, srinandan dasmahapatra, Nov 2012.

[2] Voxforge, 2012. URL http://www.voxforge.org/.

[3] Tor Aamodt. A simple speech recognition algorithm for ece341, 04 2003. URL http://www.eecg.toronto.edu/~aamodt/ece341/speech-recognition/.

[4] J.A. Bilmes. What hmms can do. *IEICE TRANSACTIONS on Information and Systems*, 89(3):869–891, 2006.

[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] CMU Sphinx. *Sphinx 3 System Design Documentation*. CMU Sphinx.

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

[14] 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.