

https://dspace.org/

FPGA phonetic speech synthesiser

Mamombe, Allen

2011-05-13
http://hdl.handle.net/10539/9746

Downloaded from a DSpace repository. https://dspace.org/

FPGA BASED PHONETIC SPEECH SYNTHESISER

Allen Mamombe

A Dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, in fulfilment of the requirements of the degree of Master of Science in Engineering

Johannesburg 2010

Copyright © 2010 Wits University

All Rights Reserved

Declaration of authorship

I declare that this thesis is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science Electrical Engineering in the University of the Witwatersrand, Johannesburg South Africa. It has not been submitted before for any degree or examination in any other university.

Signed this ____ day of _____ 20____

_____

Allen Mamombe

Preface

This dissertation reports the design of a Field Programmable Gate Array (FPGA) based speech synthesiser using autoregressive techniques namely; Linear Predictive Coding (LPC) and Very High Speed Integrated Circuits (VHSIC) algorithms. The dissertation presents extensive insight into current speech synthesis techniques particularly LPC and the Harmonic plus Noise Model (HNM). There were cases on which the work presented conflicted with literature, in such cases extensive tests were performed and discussed. Some chapters do present trivial work or review of work already covered in literature, this was done in order to give the reader insight into the approach taken. The work presented is not the last word on the embedded speech synthesiser but should trigger further research.

To family, friends and mentors

Abstract

Considerable advancements have been made in the field of speech synthesis and speech analysis. Despite these advancements little progress has been made in the field of embedded speech synthesisers. This can be attributed to the slow pace in the development of Application-Specific Integrated Circuits (ASIC) and the affordability of personal computers in developed countries. The same cannot be said however for Sub-Saharan Africa and developing countries. It is therefore imperative to design low cost, memory and processor efficient devices.

This dissertation discusses the design of such a real time embedded speech synthesiser based on a 400000 system gate FPGA. An extensive literature review is documented on various speech synthesis models used in the FPGA based synthesiser. Significant attention is given to the LPC model, commonly known in the telecommunications circles as the principle behind the Global System for Mobile Communications (GSM) codec.

The challenge posed in designing the embedded speech synthesiser was to optimise the memory requirements of the LPC model to suite the suggested FPGA architecture, whilst maintaining the integrity and the quality of the speech. This challenge was solved by using a speech modelling technique combining LPC source signal modelling with the Harmonic plus Noise Model HNM. The LPC-HNM model was used to synthesise phonemes and words of the English language as required by the objectives of the FPGA based phonetic speech synthesiser. Quality of Service (QOS) and Mean Opinion Score (MOS) based listening tests were conducted on MATLAB^{TM}, VHSIC Hardware Description Language (VHDL) and on an FPGA, by a group of 20 native English speakers. Listening test results showed that the designed model performed better than renowned LPC models obtaining scores of 99% and 4.5 out of 5 on the MOS and QOS scores respectively. All speech used in this dissertation was sampled at 8 kHz.

An FPGA was chosen as the development platform because of its huge multiprocessing structure. Particular attention was given to simplifying LPC algorithms to suite the FPGA structure. This was acheived through the use of popular mathematical models such as the Taylor and the McLaurin's series. The designed system used less than 200000 FPGA system gates.

Results and the work carried out in this dissertation significantly illustrate the contribution made by this work in the field of embedded speech syndissertation.

Acknowledgments

The author would like to thank first and foremost the research supervisor Professor Beatrys Lacquet for her suggestions, guidance and supervision. I have gained substantial knowledge from her to fulfill this research work as well as future dreams and aspirations.

A big thank you goes to the entire School of Electrical and Information Engineering at the University of the Witwatersrand, Johannesburg. Particularly, the electronic engineering research group and all the students who helped in conducting experiments published in this dissertation. A special mention goes to Mr Cuthbert Nyamupangedengu and Dr Shuma-Iwisi the co supervisor in this research for the countless suggestions, unwavering assistance and scrutiny of every work submitted.

Work presented by Gideon Klompie formerly of the speech and language-processing group at the University of Stellenbosch is greatly acknowledged, as well as his permission to include an extension of his Master of Science degree as part of this dissertation. A sincere gratitude goes to my family in Zimbabwe for their unwavering support through difficult times. The project could not have been fulfilled without their heartfelt support and warmth.

Publications

A. Mamombe, B. Lacquet, "Optimised source signal modelling for Linear predictive speech synthesis," *In proceedings of the 18th international symposium of the Pattern Recognition Association of South Africa PRASA 2007*, pp.93-98, Pietermarizburg, South Africa, Nov 2007.

B. Lacquet, M. Shuma-Iwisi, A. Mamombe, "Advancements in assistive speech technology for sub Saharan Africa," *Conference on Collaborative Research for Technological Development*, pp. 131-136, Kampala Uganda, 17th - 21st December 2007.

B. Lacquet, M. Shuma-Iwisi, A. Mamombe, "An optimised parametric speech synthesis model based on linear prediction (LP) and the Harmonic plus Noise Model (HNM)," *In proceedings of the 19th international symposium of the Pattern Recognition Association of South Africa PRASA 2008*, pp. 176-177, Cape Town South Africa, Nov 2008.

Abbreviations

VHSIC	Very High Speed Integrated Circuits
VHDL	VHSIC Hardware Description Language
FPGA	Field Programmable Gate Array
LPC	Linear Predictive Coding
ASIC	Application-specific integrated circuits
GSM	Global System for Mobile communications
HNM	Harmonic plus Noise Model
QOS	Quality Of Service
MOS	Mean Opinion Score
PDA	Personal Digital Assistant
LP	Linear Prediction
TTS	Text-to-Speech
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LMA	Log Magnitude Approximate
ARX	Auto-Regressive with Exogenous input filter
R-K	Rosenburg-Klatt
DFT	Discrete Fourier Transform
LFSR	Linear Feedback Shift Registers
GUI	Graphic User Interface
FBLS	Forward Backward Least Squares
AR	Auto-Regressive
ANSI C	American National Standards Institute C
CORDIC	Coordinate Rotation Digital Computer
LUT	Look Up Tables
JEDEC	Joint Electron Device Engineering Council

Contents

Table of Contents **xii**

List of Figures **xvii**

List of Tables **xx**

1 Introduction **1**

1.1 Importance of embedded speech synthesis 1

1.2 Problem statement 1

1.3 Objective of the research 2

1.4 Background information 2

1.4.1 The human speech production system 3

1.4.2 Linguistic analysis 5

1.4.3 Co-articulation and prosody 5

1.5 Speech synthesis models 7

1.5.1 Rule based models 7

1.5.2 Concatenate based models 8

1.5.3 Basic constituents of a speech synthesis system 8

1.6 Present speech synthesis models 9

1.6.1 Slovenian speech synthesiser 9

1.6.2 MicroDress system 9

10subsection.1.6.3

1.6.4 Speak and spell toy by texas instruments 11

1.7 Discussion of systems 11

2 Literature review **13**

2.1 Rule based speech synthesis . . . 13

2.2 Linear prediction . . . 13

2.3 Harmonic plus noise model . . . 15

2.4 Log magnitude approximate filter . . . 16

2.5 Auto-regressive with exogenous input filter . . . 17

2.6 Forward-backward least squares spectral estimate . . . 17

2.7 Discussion of rule based models . . . 18

3 Benchmarking tests **20**

3.1 Introduction . . . 20

3.2 LP source signal modelling . . . 20

3.2.1 Traditional source signal modelling techniques . . . 21

3.2.2 Rosenburg-Klatt modified model . . . 23

3.2.3 HNM based source signal modelling . . . 24

3.2.4 Discussion . . . 28

4 Parametric optimisation **30**

4.1 Introduction . . . 30

4.2 Optimising the number of LP parameters . . . 30

4.3 Optimising the window length . . . 33

4.4 Chapter discussion . . . 34

5 Speech synthesis design **35**

5.1 Introduction . . . 35

5.2 Speech recordings . . . 35

5.3 Speech analysis . . . 36

5.4 Inverse LP analysis . . . 38

5.5 Phoneme analysis . . . 38

5.5.1 Phoneme parametric corpus . . . 39

5.5.2 Word parametric corpus . . . 41

5.6 Chapter discussion . . . 41

6 Implementation of the design method 42

6.1 Speech generation . . . 42

6.2 Speech generation algorithm . . . 43

6.3 Analysis of the speech output . . . 44

6.4 Spectrogram analysis . . . 45

6.5 Listening tests . . . 46

6.5.1 Mean opinion score tests . . . 47

6.5.2 Transcription tests . . . 48

6.6 Discussion of results . . . 49

7 Embedded development 50

7.1 Introduction . . . 50

7.2 The VHDL platform . . . 50

7.3 VHDL code development . . . 51

7.4 Modelling the signal frequency clock . . . 53

7.4.1 Algorithm development . . . 53

7.4.2 Simulation and testing . . . 53

7.5 Modelling the noise component . . . 54

7.5.1 Algorithm development . . . 54

7.5.2 Simulation and testing . . . 55

7.6 Modelling the exponent . . . 56

7.6.1 Algorithm development . . . 56

7.6.2 Simulation and testing . . . 58

7.7 Modelling the key-in component . . . 58

7.7.1 Algorithm development . . . 58

7.7.2 Simulation and testing . . . 59

7.8 Modelling the residual adder component . . . 60

7.8.1 Algorithm development . . . 60

7.8.2 Simulation and testing . . . 60

7.9 Modelling the cosine generator . . . 61

7.9.1 Algorithm development . . . 61

7.9.2 Simulation and testing . . . 62

7.10 IIR filter modelling . . . 63
7.10.1 Algorithm development . . . 63
7.10.2 Simulation and testing . . . 63
7.11 Modelling the hamming window component . . . 64
7.11.1 Algorithm development . . . 64
7.11.2 Simulation and testing . . . 65
7.12 Interfacing module components . . . 66
7.12.1 Algorithm development . . . 66
7.12.2 Memory utilisation . . . 66
7.12.3 Simulation and testing . . . 67
7.12.4 Output analysis . . . 68
7.12.5 Spectrogram analysis . . . 69
7.13 VHDL based listening tests . . . 70
7.13.1 Mean Opinion Score tests . . . 70
7.13.2 Transcription tests . . . 71
7.13.3 Discussion of results . . . 72

8 Hardware development **74**
8.1 Hardware implementation . . . 74
8.2 External hardware . . . 74
8.3 Hardware tests . . . 75
8.4 Discussion of results . . . 78

9 Conclusion and future work **79**
9.1 Conclusion . . . 79
9.2 Improvements and future work . . . 80
9.3 Contributions of the research . . . 81

References **82**

A Parametric Corpus **86**

B Development Code **111**

C Publications from the thesis **128**

List of Figures

1.1 The human vocal system 2

1.2 Typical human excitation signal 3

1.3 Typical speech signal in the frequency domain 4

1.4 A typical spectrogram of the speech signal 4

1.5 Model of the human vocal system 7

1.6 The life cycle of a speech synthesis system 8

2.1 The vowel /a/ in the frequency domain 15

2.2 Speech synthesis development methodology 19

3.1 The unit impulse source signal in the time domain 21

3.2 The triangular source signal in the frequency domain 22

3.3 The Rossenburg Klatt source signal 23

3.4 The modified Rossenburg Klatt source signal 24

3.5 The vowel /a/ residual signal in the frequency domain 24

3.6 Resultant HNM residual signal 26

3.7 Scatter plot of harmonic components for the signal /a/ 27

3.8 Modelled HNM residual signal for the vowel /a/ 28

4.1 Residual signal derived from using 2 LP parameters 31

4.2 Residual signal derived from using 10 LP parameters 31

4.3 Residual signal scatter plot derived from inverse LP analysis 32

4.4 Residual signal derived from using 20 LP parameters 32

5.1 Vowel /a/ at 44kHz in the frequency domain 36

5.2 Vowel /a/ at 8kHz in the frequency domain 36

5.3 Different trunctation window spectral leakage 37
5.4 A hamming filter output of the vowel /a/ speech segment 37
5.5 Chained hamming signal vowel /a/ 38

6.1 The speech synthesis block diagram 42
6.2 The speech synthesis algorithm 43
6.3 A comparison of the systhesised and original signal 45
6.4 A spectrogram analysis of the original vowel /a/ 46
6.5 A spectrogram analysis of the synthesised vowel /a/ 46

7.1 The speech synthesis circuit 52
7.2 Schematic of the 8 kHz block component 53
7.3 Simulation in Xilinx of the 8 KHz block component 54
7.4 The LFSR random noise generator 55
7.5 Simulation in Xilinx of the random noise generator 56
7.6 The exponential gradient component 57
7.7 Simulation in Xilinx of the exponent component 58
7.8 The keyin component 59
7.9 Simulation in Xilinx of the Keyin component 59
7.10 The residual adder component 60
7.11 Simulation in Xilinx of the residual adder component 61
7.12 The cosine component 62
7.13 Simulation in Xilinx of the cosine component 62
7.14 Reconfigurable filter block component 63
7.15 Simulation in Xilinx of the filter component 64
7.16 Hamming window component 65
7.17 Simulation in Xilinx of the hamming window component 65
7.18 Recorded wave analysis of the phoneme /a/ 68
7.19 Time domain comparison of the VHDL systhesised waveform vs the original waveform 68
7.20 VHDL synthesised vowel /a/ signal in the frequency domain 69
7.21 A spectrogram analysis of the VHDL synthesised vowel /a/ 69
7.22 A spectrogram analysis of the MATLAB^{TM}synthesised vowel /a/ 70

8.1 A hardware schematic of the speech synthesiser 75

List of Tables

1.1 The English phonemes table 6

3.1 Goodness of fit results on scatter plots 27

4.1 Goodness of fit results on variable LP parameters 33
4.2 Goodness of fit results on variable window length 33

5.1 Parametric speech corpus for HNM LP model vowel /i/ 39
5.2 Words included as part of the corpus 41

6.1 MATLAB^{TM}based mean opinion scores (words) 47
6.2 MATLAB^{TM}based mean opinion scores (phonemes) 48
6.3 MATLAB^{TM}based transcription scores (words) 48
6.4 MATLAB^{TM}based transcription scores (phonemes) 49

7.1 Characteristics of the Xilinx XC3S1600E FPGA device 51
7.2 Effects of varying the harmonic gradient on speech output 57
7.3 Stage interfacing of circuit components 66
7.4 Logic utilisation on the FPGA chip 67
7.5 VHDL based mean opinion scores (words) 71
7.6 VHDL based mean opinion scores (phonemes) 71
7.7 VHDL based transcription scores (words) 72
7.8 VHDL based transcription scores (phonemes) 72

8.1 Phoneme listening test results for the built speech synthesiser 75
8.2 Word listening test results for the built speech synthesiser 77

A.1 Parametric speech corpus for HNM and LP model vowel /a/ 86
A.2 Parametric speech corpus for HNM and LP model vowel /e/ 88
A.3 Parametric speech corpus for HNM and LP model vowel /i/ 89
A.4 Parametric speech corpus for HNM and LP model vowel /o/ 91
A.5 Parametric speech corpus for HNM and LP model plossive /d/ 92
A.6 Parametric speech corpus for HNM and LP model plossive /p/ 93
A.7 Parametric speech corpus for HNM and LP model fricative /s/ 94
A.8 Parametric speech corpus for HNM and LP model fricative /h/ 96
A.9 Parametric speech corpus for HNM and LP model vowel /hello/ 98
A.10 Parametric speech corpus for HNM and LP model vowel /hat/ 100
A.11 Parametric speech corpus for HNM and LP model vowel /too/ 101
A.12 Parametric speech corpus for HNM and LP model vowel /door/ 103
A.13 Parametric speech corpus for HNM and LP model fricative /shop/ 105
A.14 Parametric speech corpus for HNM and LP model fricative /that/ 107
A.15 Parametric speech corpus for HNM and LP model plossive /dig/ 108
A.16 Parametric speech corpus for HNM and LP model plossive /pit/ 110

Chapter 1

Introduction

1.1 Importance of embedded speech synthesis

Embedded speech synthesis is the artificial generation of speech on application specific integrated circuits (ASIC) [1]. The most important role of embedded speech synthesis is in the development of assistive speech technology for vocally impaired people e.g. the artificial larynx [2, 3]. A typical artificial larynx is composed of biometric sensors that are brought into contact with the person's larynx and sounds are uttered based on the larynx movement. Embedded speech synthesis is also important in developing language tools i.e. the text-to-speech synthesiser. The text-to-speech synthesiser utters the pronunciation of the input text usually entered through a keyboard.

1.2 Problem statement

Although much developmental work and resources have been committed to the field of embedded speech synthesis, there are still teething problems in the field.

- Modern speech synthesisers require large amounts of processing power and memory [1, 4].
- The unavailability of electricity and the Personal Computer (PC) in most sub-Saharan countries i.e. Zimbabwe my home country, means that most of the modern day speech synthesisers cannot be used.
- The variety in speech synthesis methodology and approaches has left a lot of room for developmental study.
- Modern day speech synthesisers are expensive and mostly targeted at the developed market.

1.3 Objective of the research

Recognising the listed problems, the objective of this dissertation was to design a resource efficient embedded speech synthesiser. In order to meet this objective a real-time speech synthesiser at 8 kHz embedded on an FPGA chip with 8064 logic cells or 400000 system gates was required. A significant literature review had to be done on speech synthesis models including LPC, HNM, GSM, Concatenate Synthesis and Power Spectrum Estimation. Comparisons were to be done on present speech models to find the optimal model for the design. In cases where the present models were in conflict with the design methodology, reasons for the conflict were provided backed by experimental procedure.

1.4 Background information

The challenge posed in designing an embedded speech synthesiser is the fact that most speech synthesis models are aimed at high end fast computational systems [1]. Taking this into consideration it becomes important to accurately redesign the speech synthesis model. When designing a speech synthesiser it is essential that the designer fully understands the physical aspects of speech production in humans as shown in Fig. 1.1 [5].

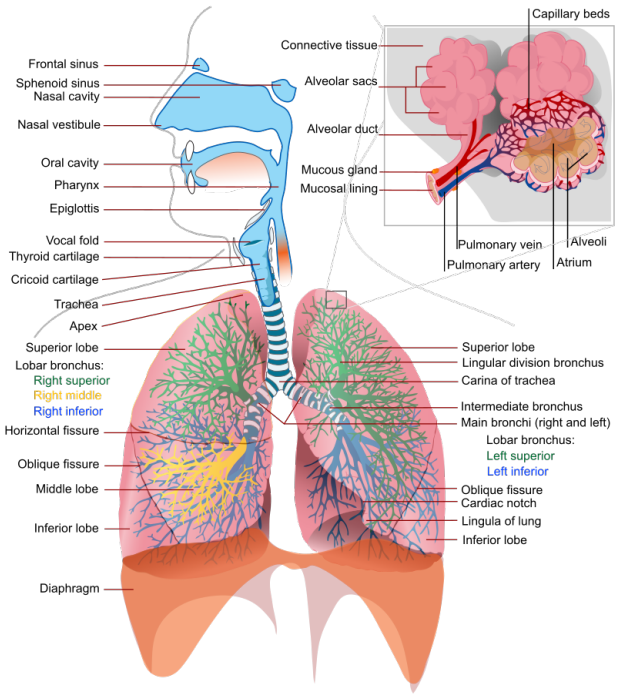


Figure 1.1 The anatomy of the human vocal system. Source: LadyofHats, A complete, schematic view of the human respiratory system, Public domain listing, Wikimedia commons 2007 [5].

1.4.1 The human speech production system

During the speaking process lungs are filled with air through an expansion of the rib cage. As the rib cage contracts air is forced through the trachea and glottis to produce either a periodic, quasi-periodic or random waveform known as the excitation signal [3, 4]. The excitation signal can be controlled in various ways inside the vocal tract to produce different excitation modes for the vocal system. Control of the glottis produces three broad classes of sounds which are voiced sounds, unvoiced sounds and nasals analysed in detail at linguistic level in [4]. The frequency at which the glottis is excited is known as the fundamental frequency F_o usually 120 Hz for male and 140 Hz for female. The fundamental frequency determines the pitch of the sound produced. Fig. 1.2 [6] shows a typical excitation signal U_m produced by the glottis movement U_g.

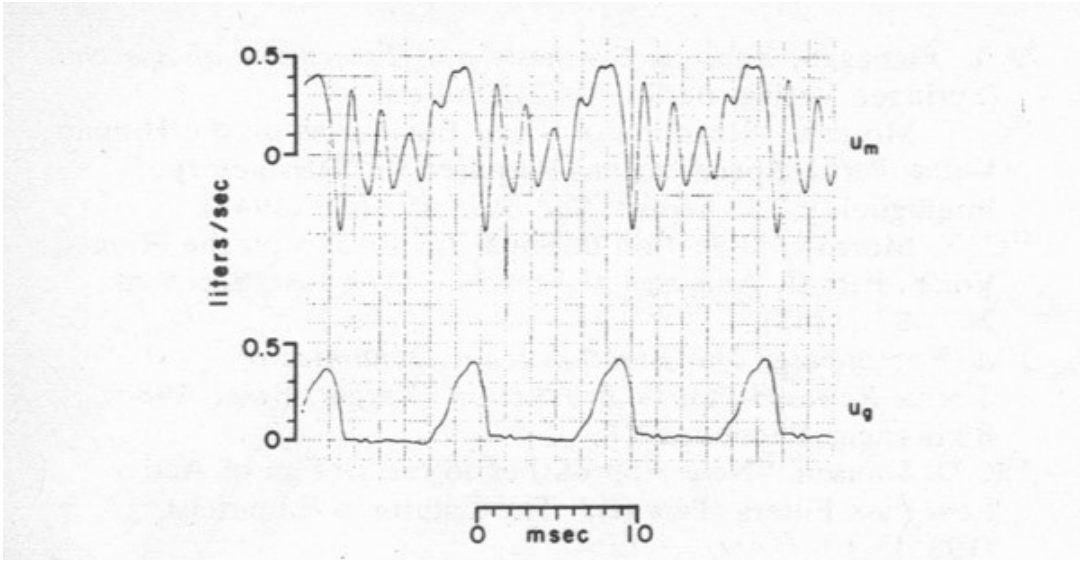


Figure 1.2 A typical excitation signal U_m produced by the glottis movement U_g. Source: M. Rothenberg, A New Inverse-Filtering Technique for Deriving the Glottal Airflow Waveform During Voicing, ©J Acoust Soc Am 53, pp.1632-1645 (1973) [6].

According to [4] a typical human vocal system produces speech that is not quasi-periodic. The analysis of non quasi-periodic speech signals is performed at two levels namely qualitative and linguistic. Qualitative speech analysis entails working with the speech signal in both the time and frequency domains as shown in Fig. 1.3 and Fig. 1.4 respectively. The data presented in the figures was obtained from analysing a speech sample of the vowel /a/ in both the time and frequency domains.

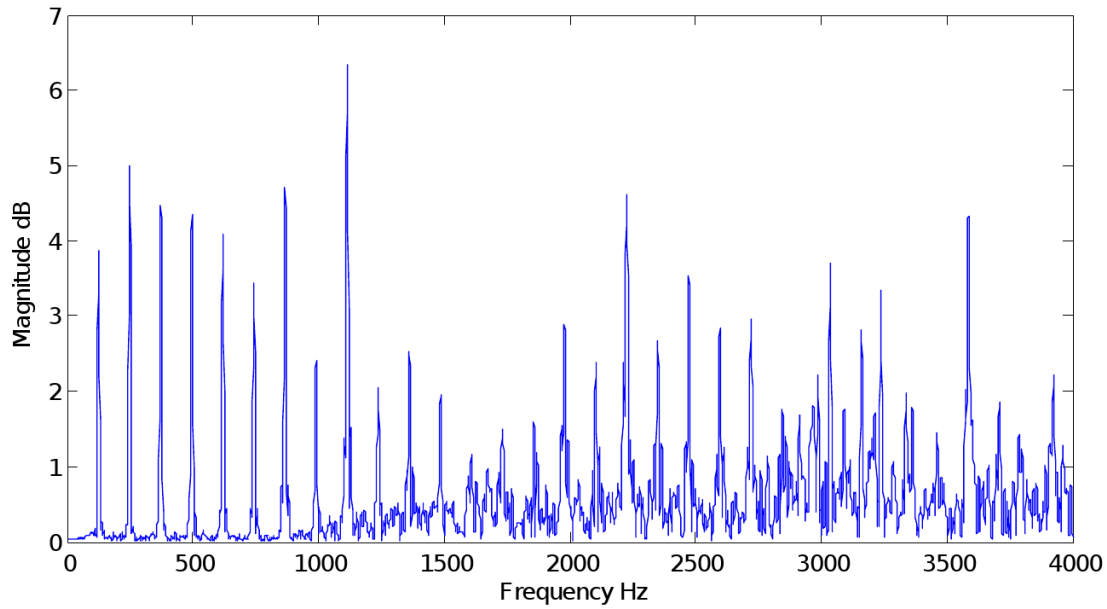


Figure 1.3 A typical speech signal in the frequency domain.

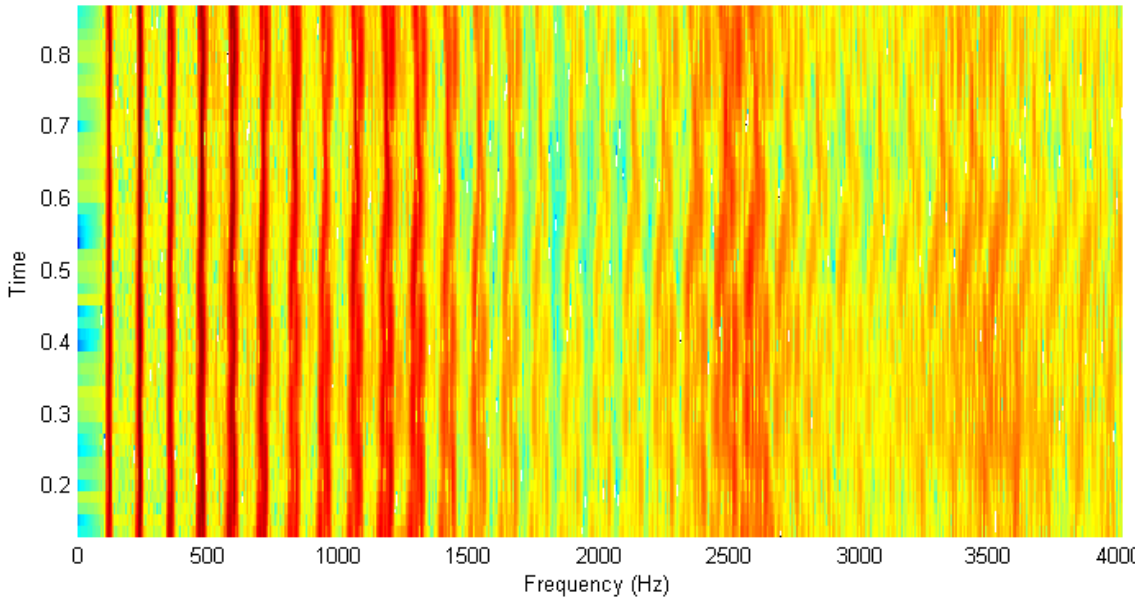


Figure 1.4 A spectrogram analysis of the speech signal.

1.4.2 Linguistic analysis

At linguistic level the three broad classes of sounds produced can be further classified into a sequence of basic sound units called phonemes. Phonemes convey the same message yet sound different because of different dialects. The quality or intelligibility of modelled speech is heavily dependent on the listener and his or her dialect background [7]. Because of this it becomes more important to model the speech at phoneme level than at sound class. Phonemes correspond directly to articulatory positions and movement called articulatory gestures. Speech sounds can thus be classified according to the gestures as voiced sounds e.g. plosives, vowels and semivowels, or unvoiced sounds such as nasals, liquids and diphthongs [4]. Table. 1.1 shows the classification of all 44 English phonemes to their respective phoneme types [8].

1.4.3 Co-articulation and prosody

In normal speech the targeted articulatory positions for most gestures may never be reached as the other articulatory gesture is already taking place [7]. In order for the speech not to sound monotonous there is a variation in utterance of the phonemes hence communicating is more than just the message but the feeling of the speaker. This variation in the intensity of the sound is known as prosody [7]. The research will however not emphasise the prosody effect as this is difficult to model [4]. Instead, the research will concentrate on analysis based speech synthesis models available today, namely, rule based models and concatenative dictionary based models.

Table 1.1 The English phoneme database [8].

Vowel	Sound	Common Spelling	Consonant	Sound	Common Spelling
1	/I/	pit	21	/p/	pit
2	/e/	pet	22	/b/	bit
3	/æ/	pat	23	/t/	time
4	/ɒ/	pot	24	/d/	door
5	/ʌ/	luck	25	/k/	cat
6	/ʊ/	good	26	/g/	get
7	/ə/	ago	27	/f/	fan
8	/i:/	meat	28	/v/	van
9	/a:/	car	29	/θ/	think
10	/ɔ/	door	30	/s/	send
11	/ɜ/	girl	31	/z/	zip
12	/u:/	too	32	/m/	man
13	/ei/	day	33	/n/	nice
14	/ai/	sky	34	/ŋ/	ring
15	/oi/	boy	35	/l/	leg
16	/iə/	beer	36	/r/	rat
17	/eə/	bear	37	/w/	wet
18	/ʊə/	tour	38	/h/	hat
19	/əʊ/	go	39	/j/	yet
20	/aʊ/	cow	40	/ʃ/	shop
			41	/ð/	that
			42	/ʒ/	leizure
			43	/tʃ/	chop
			44	/dʒ/	jump

1.5 Speech synthesis models

1.5.1 Rule based models

The most common approach taken in modelling the human vocal system is the source filter model [4,7]. This approach aims to model speech based on the architecture of the actual vocal articulatory parameters, hence the name rule based models. In Rule Based Speech Synthesis speech is generated based on the formant and anti-formant parameters [3], examples include formant synthesisers and to a greater extend, auto-regressive models. Auto-regressive speech models are composed of two main components namely the source and the filter component. The source component is composed of a continuous signal waveform at the fundamental frequency [4] whilst the filter component is defined by the poles and zeros that make up the filter parameters. A source filter model corresponds to the articulatory gestures of the human vocal system as shown in Fig. 1.5. Experiments over time have shown that the quality and intelligibility of speech produced from such models still falls short of many people's expectations [4].

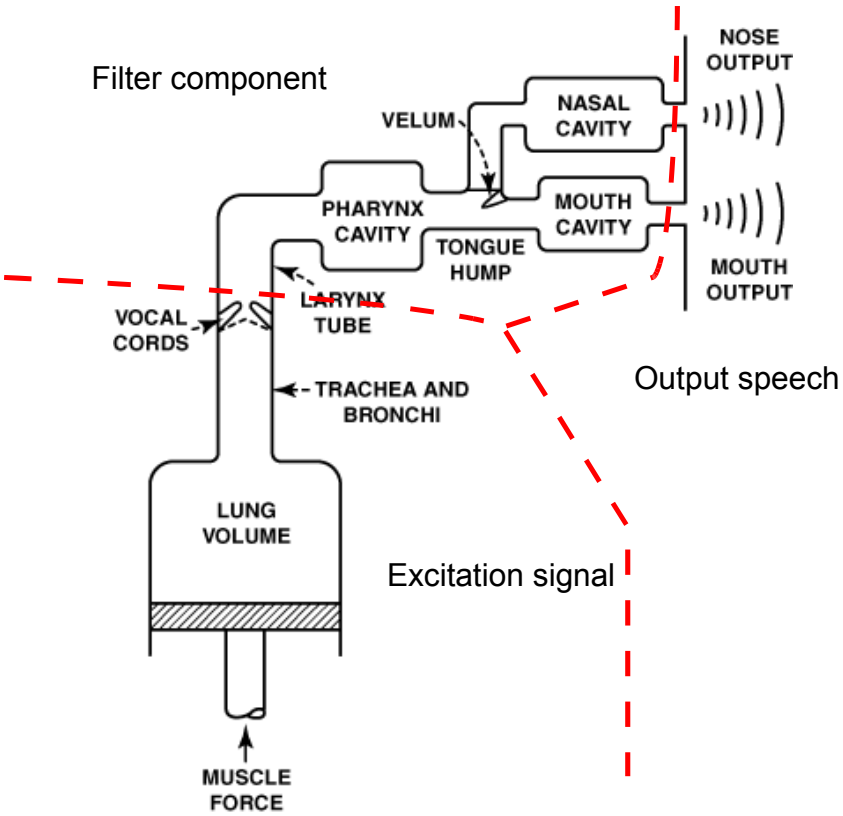


Figure 1.5 Model of the human vocal system. Adopted From: J. L. Flanagan, Speech Analysis and Perception, Springer-Verlag, Berlin, 2nd edition, 1965 [9].

1.5.2 Concatenate based models

In Concatenative or Dictionary Based Synthesis examples of recorded phonetic transitions and co articulation are stored into a speech database as either diphones or triphones speech segments [4]. These speech segments are chained up to produce the required output speech. The quality and intelligibility of speech produced from such speech synthesisers has been widely accepted across community [10]. However this is greatly dependent on the quality of the corpus or database. Concatenative speech synthesisers also require huge amounts of memory because complete speech segments are stored in the model instead of parameters.

1.5.3 Basic constituents of a speech synthesis system

The design of a speech synthesis system is best described as a cycle. Fig. 1.6 shows a basic block diagram of the speech synthesis design cycle . The cycle has four main constituents namely: Speech Input, Speech Analysis, Speech Synthesis and the Speech Output.

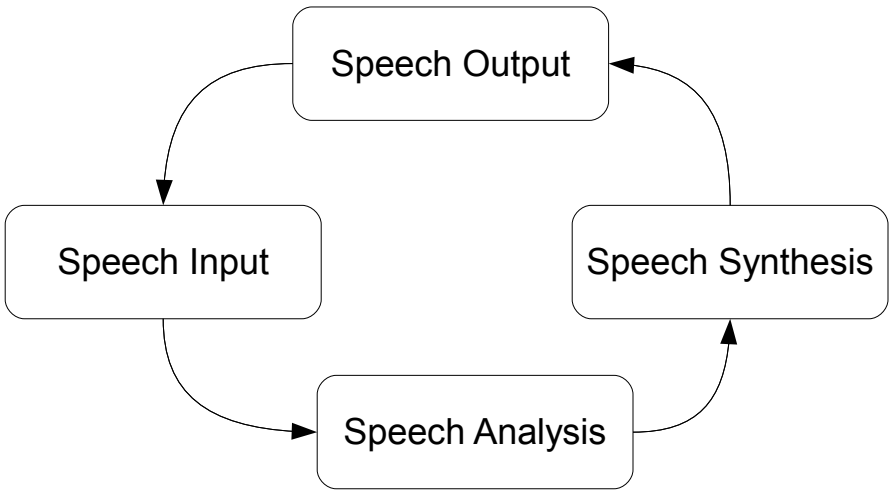


Figure 1.6 The cycle of a speech synthesis system.

The speech input is a trigger to generate the sound, this can be in the form of a text key pad in the case of Text-To-Speech (TTS) systems or sensors monitoring the movement of the vocal tract in the case of vocoders. Speech analysis entails the extraction of features to form a speech corpus in rule based synthesis or the collection of speech segments in the case of dictionary based methods. The speech synthesiser is the actual system that generates the speech using the speech corpus and algorithm to produce the output sound.

1.6 Present speech synthesis models

Examples of modern day speech synthesis systems built around the four step speech cycle include the Slovenian TTS system [12], the speak and Spell system [4], the microDress system [1] and the Papageno system [13]. The section briefly describes each of the systems with particular attention to the architecture of synthesis engine. The pros and cons of each system are discussed in order to build an argument around the synthesis approach taken.

1.6.1 Slovenian speech synthesiser

The Slovenian TTS system converts Slovenian text feed through a keyboard into uttered speech. The system utilizes both rule based and dictionary based speech models. Rule based prediction methods are used to determine the pitch, duration and prosodic parameters. A dictionary based unit selection algorithm is used to select speech recordings from a database consisting of diphones, triphones and sentence recordings. Finally a time varying warping acoustic alignment procedure is used between the synthetic voice and recordings [12] .

Advantages of the system

- According to [12] the Slovenian Speech System utilises a small corpus of about 297 diphones and triphones of the possible 1024 diphones and triphones in the Slovenian language.
- The total memory footprint of the speech system has small memory footprint of about 2 MB, to accommodate both the rule based parameters and dictionary based corpus [12].

Disadvantages of the system

- Most embedded chips have a memory capacity of less than 2 MB.
- Design specific application algorithms are used. Therefore, the system cannot be directly used for other languages.

1.6.2 MicroDress system

One of the main disadvantages of the Slovenian TTS system was the size of the parametric corpus. The microDress TTS addresses this problem by making use of a compressed speech corpus consisting of segments recorded natural speech. A dictionary based approach with a diphone inventory and a

reduced code database is used to concatenate the speech segments to form sentences. In order to reduce the size of the corpus the recorded speech quality is reduced to telephone quality. Optimised algorithms are used for prosodic parameter manipulation and smoothing of the formant contour. Code and data are strictly separate thus the system is adaptable to many language databases [1].

Advantages of the system

- The total memory footprint of the microDress system if about 1 MB almost half that utilised by the Slovenian system.
- The system utilises efficient algorithms making real time processing possible [1].

Disadvantages of the system

- Only a microDress specific diphone inventory can be used for the speech corpus.
- The bandwidth of the speech inventory is reduced to telephone quality.

1.6.3 Papageno TTS system

The Papageno TTS system was designed for use on mobile phones and PDAs. The system fits on the ARM 500 kB, 50 MHz platform. A neural network on the front-end of the TTS system greatly reduces the size of the inventory. The speech generator utilises both diphone and triphone based inventories. Prosodic parameter generation is based on a bigger neural network. A time domain manipulation and concatenation method is used. Interpolation errors are greatly reduced in the system due to the use of the triphone and diphone inventories [13].

Advantages

- The sound quality produced from the Papageno system is high [13], this is because the system utilises both diphone and triphone inventories.
- The entire Papageno system can fit on most embedded devices because the entire speech corpus is about 500 kB [13].

Disadvantages

- There is inadequate database segmentation and annotation which results in bad segment bounds and poor speech quality.

- According to [13] the error rate on the system is high due to a wrong database tagging method.

1.6.4 Speak and spell toy by texas instruments

This system was invented in the early 1980s using a rule based linear predictive method. Two 128 kB memory chips hold data for words and phrases [4]. An embedded microprocessor and external user interfaces like keyboards and displays were incorporated in the system. A lattice filtering method was utilised for good stability properties. A total of twelve parameters were utilised for every 35 ms of speech including 10 reflection coefficients [4].

Advantages

- The system is cheap at a total cost of about US50 dollars [4].
- An efficient data rate of 1.2 kbit/s is used [4].
- Reasonably intelligible speech is produced [4].

Disadvantages

- Real-time processing of speech is not possible [4].
- Fricatives and nasals are pronounced poorly [4].

1.7 Discussion of systems

This section discussed the various speech synthesis systems available today. Each of the individual systems discussed in the prior sections has short-falls in either memory requirements or real-time speech processing capability. This is mainly attributed to the type of models used in the systems. As a solution, I proposed the use of rule based models discussed further from the next section. The rule based approach shall form the core of the work presented in this dissertation.

The rest of the dissertation will discuss the work carried out by me in designing the proposed speech synthesis system. The work will be presented in chapters as detailed below:

Chapter 2: Will discuss in detail rule based approaches and the various models available including LP, HNM, LMA and Power Spectrum Estimation. It is proposed at the end of this chapter to use the LP and HNM based models.

Chapter 3: Presents ways of adjusting the LP and HNM based models by presenting simpler mathematical approaches. This chapter is the first to introduce the work and experiments carried out by me.

Chapter 4: Will discuss ways of fine tuning the LP and HNM model through parametric optimisations.

Chapter 5: Presents the steps taken in building the speech synthesis model with particular attention to feature extraction and analysis.

Chapter 6: Presents the building of the speech synthesis model in MATLAB^{TM}. This chapter introduces the first fully fledged experiments performed by me on the designed model in MATLAB^{TM}. Results of listening tests conducted here are also presented.

Chapter 7: Discusses the building of the speech synthesiser in VHDL for the targeted FPGA platform. Here, detailed simulations of the model are done and presented graphically before building the hardware. Listening tests are also performed and compared to the MATLAB^{TM}results.

Chapter 8: The penultimate chapter discusses building the FPGA hardware and downloading the VHDL program to the hardware. This chapter discusses listen tests done on the hardware output and comparisons to the VHDL / MATLAB^{TM}results are presented.

Chapter 9: Discusses and concludes the document by giving recommendations and presenting a measure of how far the objectives have been reached.

Chapter 2

Literature review

2.1 Rule based speech synthesis

Rule based speech synthesis aims to model speech based on the architecture of the human vocal system. As discussed in the previous chapter the human vocal system can be modeled as a source filter system. Speech is produced using mathematical parameters that define the source filter system. The pitfall with this approach is that a speech signal is non quasi periodic and does not contain a definite set of parameters. This results in low quality speech being produced whenever rule based models are applied. Modern day science has turned to dictionary based approaches when it comes to speech synthesis. The only pitfall with this method is that it requires a vast amount of memory and would not be ideal for the intended FPGA based speech synthesiser. It was therefore worth while to investigate the advancements that have been made in the rule based synthesis. This section describes the various rule based approaches available namely: Linear Prediction (LP), Harmonic Plus Noise Model (HNM), Log Magnitude Approximate (LMA), Auto-Regressive with Exogenous input filter (ARX) and the Forward Backward Least Squares (FBLS).

2.2 Linear prediction

Linear prediction is based on an autoregressive model that calculates future samples of a quasi periodic signal based on past predicted samples [4]. Sample values of speech, $x[n]$ are approximated as a linear combination of the past p speech samples as shown by [14];

$$\widetilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k]. \tag{2.1}$$

$\widetilde{x}[n]$ is the predicted sample at instant n and a_1,a_2,....,a_k are predictor coefficients. When the predicted sample is not the same as the actual sample it results in a prediction error $e[n]$ given as [4];

$$e[n] = x[n] - \widetilde{x}[n]. \tag{2.2}$$

In its simplicity the Linear Prediction (LP) model constitutes of a source signal $e[n]$ passing through an all pole filter defined by LP coefficients a_k shown in [7];

$$x[n] = e[n] + \sum_{k=1}^{p} a_k x[n-k]. \tag{2.3}$$

It can be proved through an inverse filtering process that if the original speech signal is known and the residual signal $e[n]$ minimised to almost zero, then the filter coefficients can be established through a method of autocorrelation and lattice filtering [7].

If the error $e(n)$ and linear prediction coefficients a_k are known, then by mathematical substitution the original speech can be reconstructed by applying the error signal to an all pole digital filter with the transfer function given as [7];

$$H(z) = \frac{1}{\sum_{k=1}^{p} a_k z^{-k}}. \tag{2.4}$$

The error signal models the excitation signal and is usually represented by an impulse signal with a harmonic frequency of F_o equivalent to the speaker's fundamental frequency.

Advantages

- Few parameters as little as 12 per 25 ms of speech can effectively synthesise LP speech [4].
- LP still remains the technology of choice, for example GSM is based on LP methods [37].
- The LP filter as shown in equation 2.3 is a simple Infinite Impulse Response (IIR) filter easily modelled in embedded algorithms.
- The residual signal can be modelled effectively as simple unit impulse.

Disadvantages

- Experiments have shown that the audibility of speech produced from LP falls short of many people's expectations [4].

- The number of LP parameters are directly proportional to the quality as a listening test will prove later in this dissertation. Therefore more memory is needed to improve quality output.

2.3 Harmonic plus noise model

The harmonic plus noise model is based on the fact that speech is composed of two spectra namely a quasi-periodic and the non periodic white noise spectra [15]. Fig. 2.1 shows the frequency spectra of the vowel /a/. The distinction of the two signal components the noise and the harmonics is quite evident. The two components are distinctly separated by a time varying quantity F_{max} the maxed voiced frequency. F_{max} is the frequency at which harmonics in the signal can be distinctively classified either as periodic or non periodic [15].

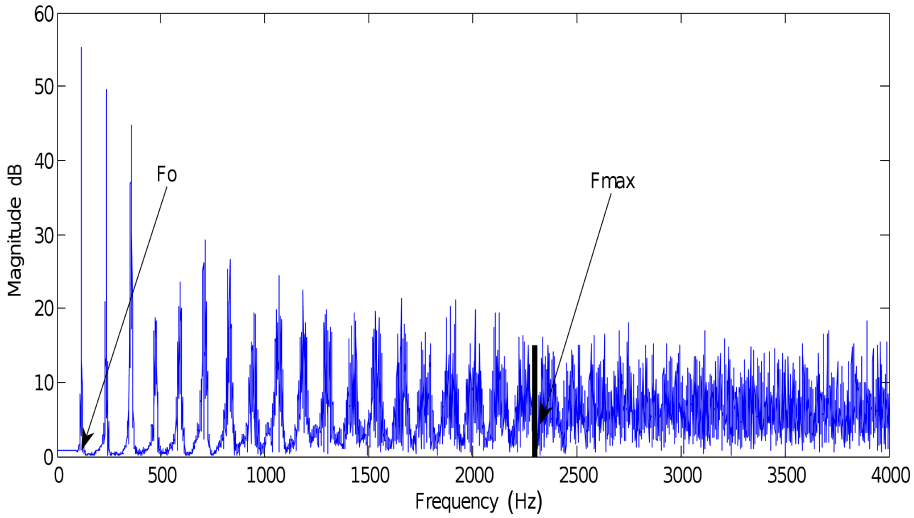


Figure 2.1 The vowel /a/ in the frequency domain.

The harmonic plus noise model proposes a way of modelling the speech signal as a periodic signal $h(t)$ and the noise component $n(t)$ as shown in [16];

$$y(t) = \sum_{k=1}^{K} A_k(t)\cos(k\theta(t) + \theta_k(t)) + n(t). \tag{2.5}$$

The periodic component is composed of harmonics whilst the non periodic component is composed of Gaussian white noise.

$\theta(t)$: the phase component

K : number of harmonics

k : k^{th} harmonic

A_k : harmonic magnitude

$n(t)$: noise component

The difficulty with the HNM model is in finding the correct equation parameters as illustrated in [17]. This dissertation highlights the research and advancements made in-order to simplify the process of finding the HNM parameters.

Advantages

The entire speech signal is parameterised hence there is no need for an excitation signal and the quality of speech signal produced from HNM is excellent [16].

Disadvantage

Obtaining the exact HNM parameters i.e the harmonic phase and the maximum frequency F_{max} is very difficult [17].

2.4 Log magnitude approximate filter

The log magnitude approximate filter was successfully adapted for a novel Chinese text to speech synthesiser [18]. The model is similar to the linear prediction discussed in section 2.2. It is composed of the spectral coefficients C_m that make up the LMA filter as shown in;

$$H(z) = \exp\left(\sum_{m=0}^{M} C_m z^{-m}\right). \tag{2.6}$$

where C_m are the Cepstrum coefficients of the undertaken speech signal and M the number of coefficients [18]. If M is large enough then the logarithmic amplitude spectrum $H(z)$ can optimally approximate the log envelope of the analysed speech signal using the least mean square method [19]. The excitation model of the voiced signal is a quasi-triangular glottal waveform and the unvoiced excitation signal is represented as white Gaussian noise [18].

Advantage

Modelling speech with cepstrum coefficients is comparable to linear predictive coding but with reduced corpus parameters.

Disadvantage

There is little literature describing the methodology and tests performed on the LMA model making it difficult to evaluate the successes of the system.

2.5 Auto-regressive with exogenous input filter

The auto-regressive with exogenous input filter model consists of a cascade of formant and anti-formant filters driven by a voicing source and an unvoiced turbulent source [20]. A Windows compatible software ARX-xml is available which allows easy extraction and modification of speech parameters such as fundamental frequency, glottal quotient tense and breath for the ARX model [20]. A Kalman filter [21] is utilised to give the formant and anti-formant parameters of the ARX model [20]. The ARX speech production model is represented by a linear differential equation [20];

$$x(n) + \sum_{k=1}^{p} a_k x[n-k] = \sum_{k=0}^{q} b_k u[n-k] + e(n). \tag{2.7}$$

where $e(n)$ is assumed to be white noise, $u(n)$ is the periodic voicing source and $s(n)$ the speech signal. The z-transform of the system is presented in [20];

$$S(z) = \frac{B(z)}{A(z)} U(z) + \frac{1}{A(z)} E(z) \tag{2.8}$$

where $B(z)$ and $1/A(z)$ represent the voiced and unvoiced sound vocal tract filter transfer function respectively.

Advantage

The model is based on the vocal tract model and therefore tries to emulate exactly the human vocal system.

Disadvantage

The speech produced tends to be monotonous or robotic.

2.6 Forward-backward least squares spectral estimate

The forward-backward least squares spectral estimate method adaptively computes the least squares estimate of the signal power spectrum. This is achieved by modelling the input as an m^{th} order

Auto-Regressive (AR) signal [19] and computing the sum of the forward and backward prediction error energies [22]. If $x(M), x(M+1), ...x(N)$ are the actual data samples of the power spectrum to be estimated then the forward and backward error energies can be approximated as $\epsilon^f(n)$ and $\epsilon^b(n)$ shown in;

$$\epsilon^f(n) = x(n) + \sum_{k=1}^{m} c_k x[n-k]. \tag{2.9}$$

$$M + m \leq n \leq N.$$

$$\epsilon^b(n) = x(n-m) + \sum_{k=1}^{m} c_k x[n-m+k]. \tag{2.10}$$
$$M + m \leq n \leq N.$$

where c_k are the auto-regressive coefficients. m and n are the maximum and minimum number of coefficients computed. $x[n]$ is the speech signal. Minimising the sum of the squares of the forward and backward vector norms $\epsilon^f(M+n, M)$ and $\epsilon^b(M+n, M)$ with respect to the AR coefficient vector c_m results in [22];

$$S_m(M, N)c_m(M, N) = -s_m(M, N). \tag{2.11}$$

Advantage

The model uses an adaptive algorithm which makes it possible to compute non periodic signals like speech.

Disadvantage

There are limited cases were the Forward-Backward Least Squares Spectral estimate has been used to model speech.

2.7 Discussion of rule based models

This chapter identified, analysed and compared the different rule based speech synthesis models available. Of all the models discussed the linear prediction model was mathematically simpler than the Log Magnitude Approximate, the Forward-Backward Least Squares Spectral estimate and the

Harmonic plus Noise Model. The main pitfall identified with using the Linear Prediction model is the poor quality of speech produced. This was attributed to a greater extent to the inaccuracy of the source signal model. The next section discusses how i was able to improve the quality of speech produced by the Linear Prediction model. The subsequent section also marks the beginning of my own work. The work is summarised as a speech synthesis development methodology shown in Fig. 2.2.

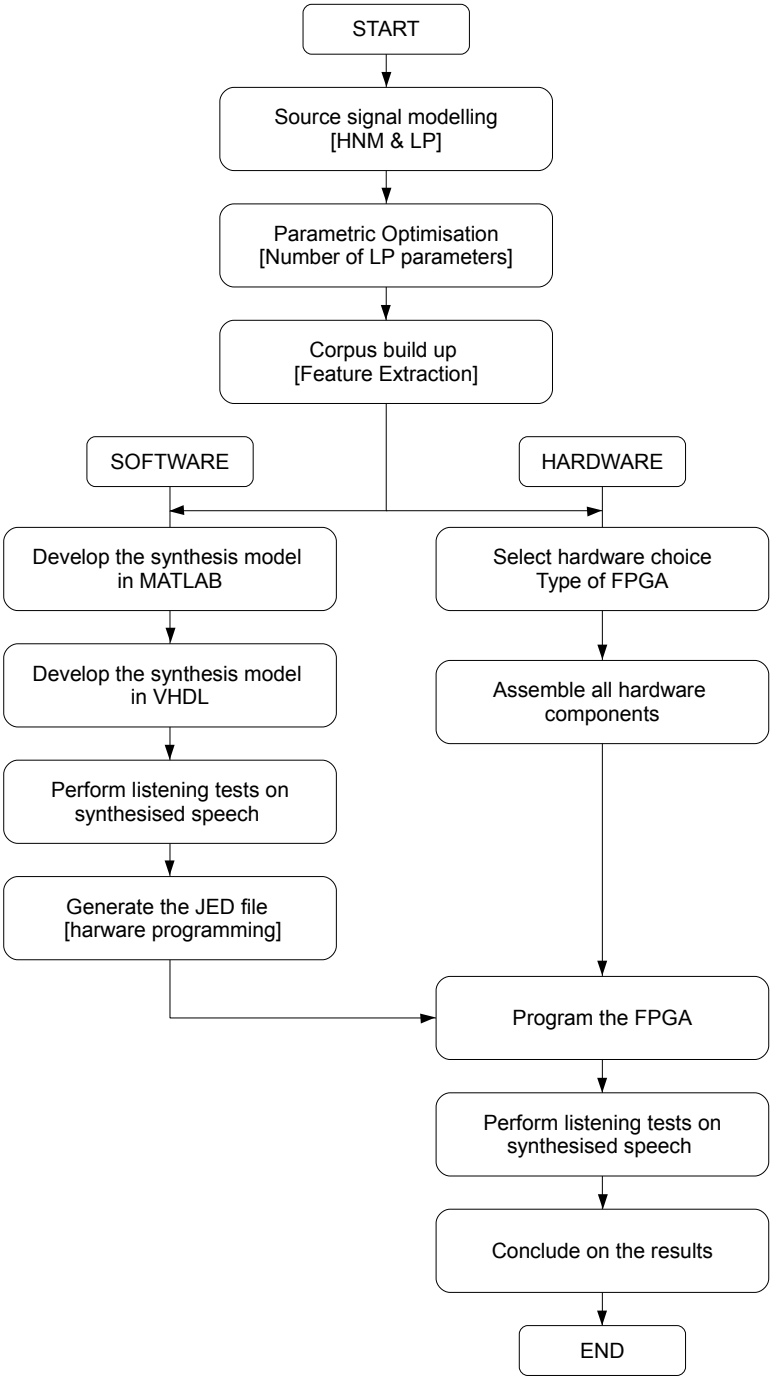


Figure 2.2 Speech synthesis development methodology.

Chapter 3

Benchmarking tests

3.1 Introduction

The standard linear prediction model as described in section 2.2 utilises the autoregressive concept [19] to model speech. In LPC speech is viewed as a stationary signal over a period of $20 - 30$ ms [7]. Thus the model consists of the LP parameters that define the filter as well as the residual signal that depicts the excitation signal. In standard LPC the residual signal is modelled as either triangular, Rosenburg Klatt or the unit impulse wave at a fundamental frequency F_o of between $120 - 140$ Hz [4]. In this section we discuss two new source signal modelling techniques used to improve the quality of Linear Predictive synthesised speech .

3.2 LP source signal modelling

Popular models of the LP source signal include the Rosenburg-Klatt (R-K), the triangular pulse, codebooks and the unit impulse [4]. Tests in [7] have proved that the R-K model is the most favourable compared to the triangular and unit impulse. The problem with the R-K model is that the voiced sounds are assumed to have no noise component hence the sound becomes robotic and monotonous. In this dissertation we investigated two fairly recent source signal modelling techniques that solve this problem namely:

- A linear modification of the R-K signal.
- A modification of the Harmonic Plus Noise (HNM) speech processing technique to model the source signal [7, 16].

3.2.1 Traditional source signal modelling techniques

Impulse train

The impulse train Fig. 3.1 is one of the traditional models used in modelling the source signal. The unit impulse source signal was used to synthesise the vowel /a/ with thirteen LP filter coefficients in a MATLABsimulated environment. The result of the synthesis was recorded in MATLABas a wave file. In order to test the quality of the synthesised speech, standard listening tests namely the Mean Opinion Score MOS and Quality Of Services QOS [23] were conducted by the author in [25]. The resultant synthesised speech performed fairly on the MOS and QOS test with scores of 4.5 and 82% respectively [25]. The perfect MOS and QOS scores would have been a 5 and 100%, respectively.

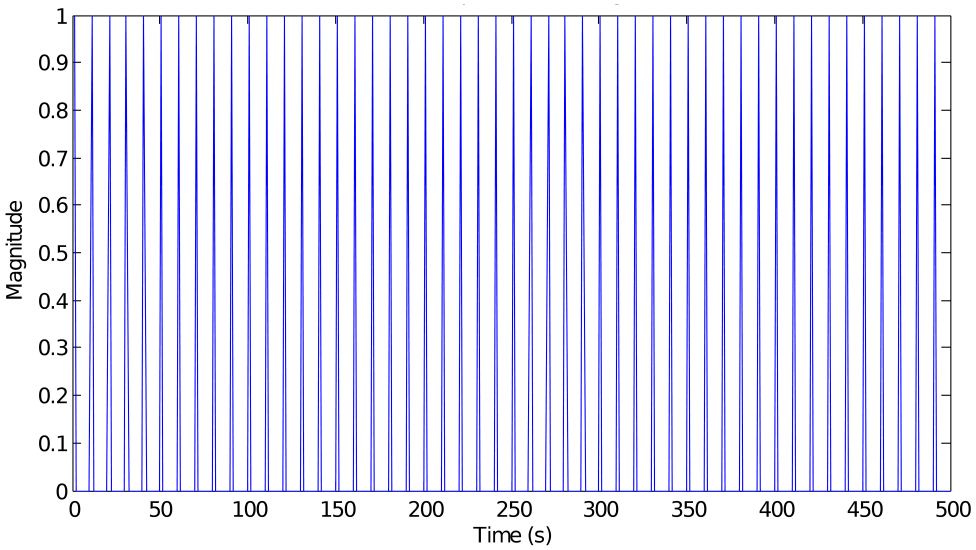


Figure 3.1 The unit impulse source signal in the time domain.

Triangular pulse

The triangular pulse Fig. 3.2 is another method of modelling the residual signal for LPC synthesis. An experiment similar to that carried out using the unit impulse source signal was conducted. The triangular pulse was used to synthesise the vowel /a/ with thirteen filter coefficients. Standard listening tests namely MOS and QOS were conducted by me in [25]. Poor results were obtained for both listening tests with scores of 2.6 and 72% on the MOS and QOS, respectively. The sample space of listeners used in [25] were native South African English speakers and most of them attributed the poor scores to the monotocity and inaudibility of the sound produced.

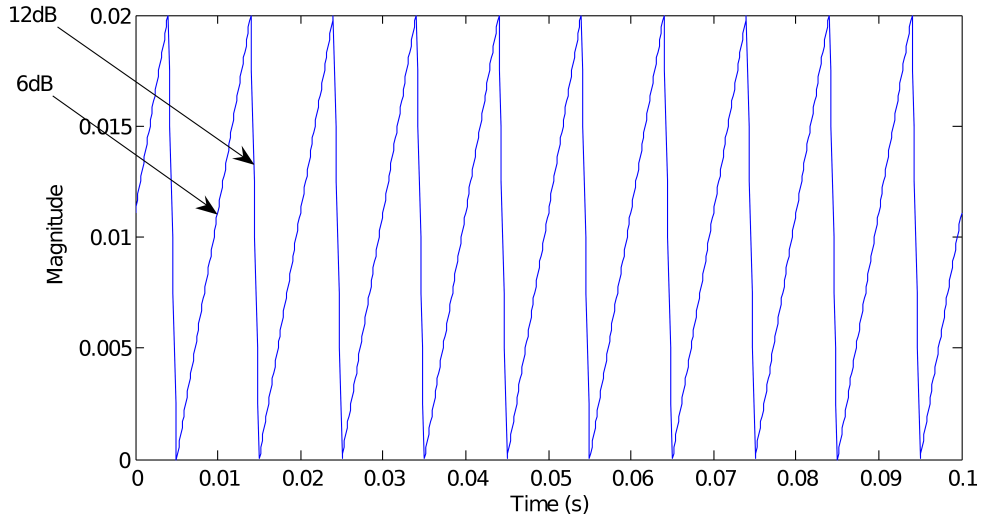


Figure 3.2 The triangular source signal in the frequency domain.

Rosenburg-Klatt model

The R-K signal shown in Fig. 3.3 is the most widely used source signal modelling technique for LPC [4]. The idea behind Rosenburg's model was to emulate the exact time domain characteristics of the human glottal excitation signal shown in Fig. 1.2 using the polynomial equation 3.1 [4]. To date the R-K polynomial has been widely modified into equation 3.2 in order to reduce computational complexity. Tests perfomed in [25] have shown that the modified model performs as well as the original R-K model on the MOS test with a score of 3.9.

$$g(t) = \begin{cases} 0 & for\ 0 \leq t \leq t_1, \\ A(\frac{(t-t_1)}{(t_2-t_1)})^2(3-2\frac{(t-t_1)}{(t_2-t_1)}) & for\ t_1 \leq t \leq t_2, \\ A(1-\frac{(t-t_2)}{(b-t_2)}) & for\ t_2 \leq t \leq b, \end{cases} \tag{3.1}$$

A : scaling factor.

t_1 : point at which the signal rises.

t_2 : point at which the signal returns to zero.

b : period of the signal.

$$g(t) = A\frac{t}{T_0}\exp(1 - \frac{t}{T_0}). \tag{3.2}$$

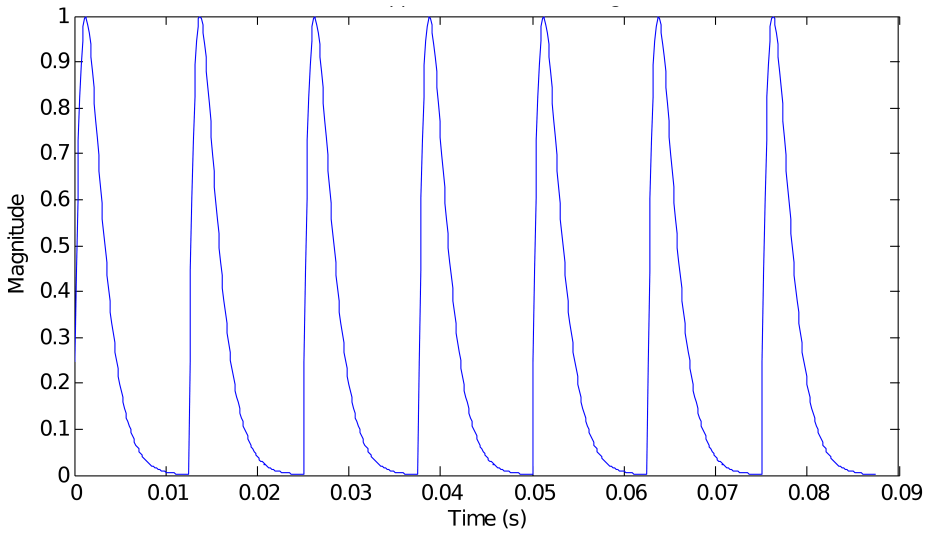


Figure 3.3 The Rossenburg-Klatt source signal.

3.2.2 Rosenburg-Klatt modified model

One of the limitations of the Rosenburg-Klatt model is the complexity of the polynomial [4]. In this dissertation a new technique of modelling the Rosenburg-Klatt signal is proposed. The new technique is a linear modification of the R-K source signal equation in [4]. A set of linear ratios is used to simplify the signal equation 3.1 by relating the values t_1, t_2 and the pitch period T_0. The resultant of the modification is shown in equation 3.4 and the time domain signal in Fig. 3.4. The modified Rosenburg-Klatt source signal was used to synthesise speech in conjunction with thirteen LP coefficients. Standard listening tests were conducted on the synthesised speech. Impressive scores of 4.1 and 96% were obtained on the MOS and QOS test, respectively;

$$b = T_0 \quad t_1 = 0.111b = aT_0 \quad t_2 = 0.667T_0 = cT_0. \tag{3.3}$$

$$g(t) = \begin{cases} 0 & for\ 0 \le t \le aT_0, \\ A(\frac{(t-aT_0)}{(cT_0-aT_0)})^2(3 - 2\frac{(t-aT_0)}{(cT_0-aT_0)}) & for\ aT_0 \le t \le cT_0, \\ A(1 - \frac{(t-cT_0)}{(T_0-cT_0)}) & for\ cT_0 \le t \le T_0, \end{cases} \tag{3.4}$$

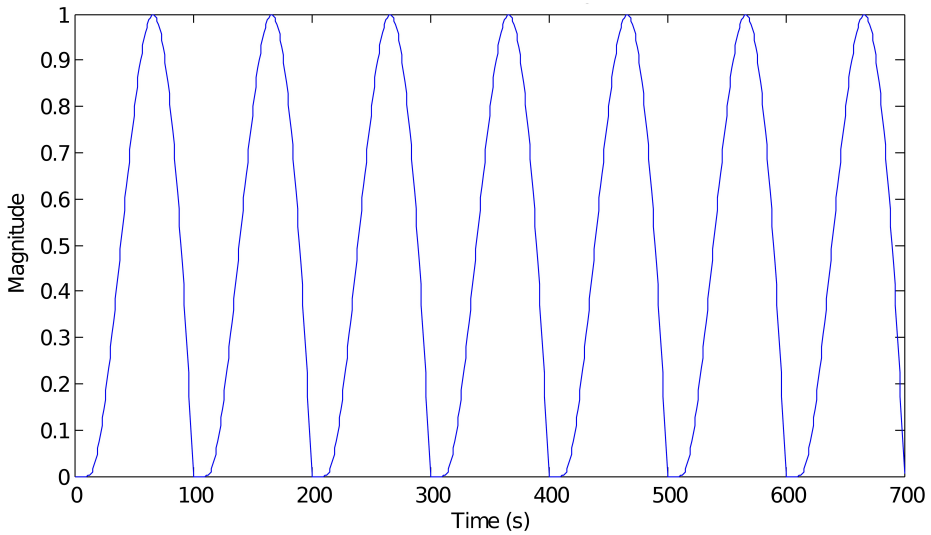


Figure 3.4 The modified Rossenburg Klatt source signal.

3.2.3 HNM based source signal modelling

The source signal for the vowel sound /a/ in Fig. 3.5 exhibits characteristics equivalent to those of the actual speech signal in Fig. 2.1. The source signal can thus be described in a similar manner to the actual speech signal as a sum of the harmonic and noise component of the residual. This meant that the HNM could be used as the source signal model in this dissertation.

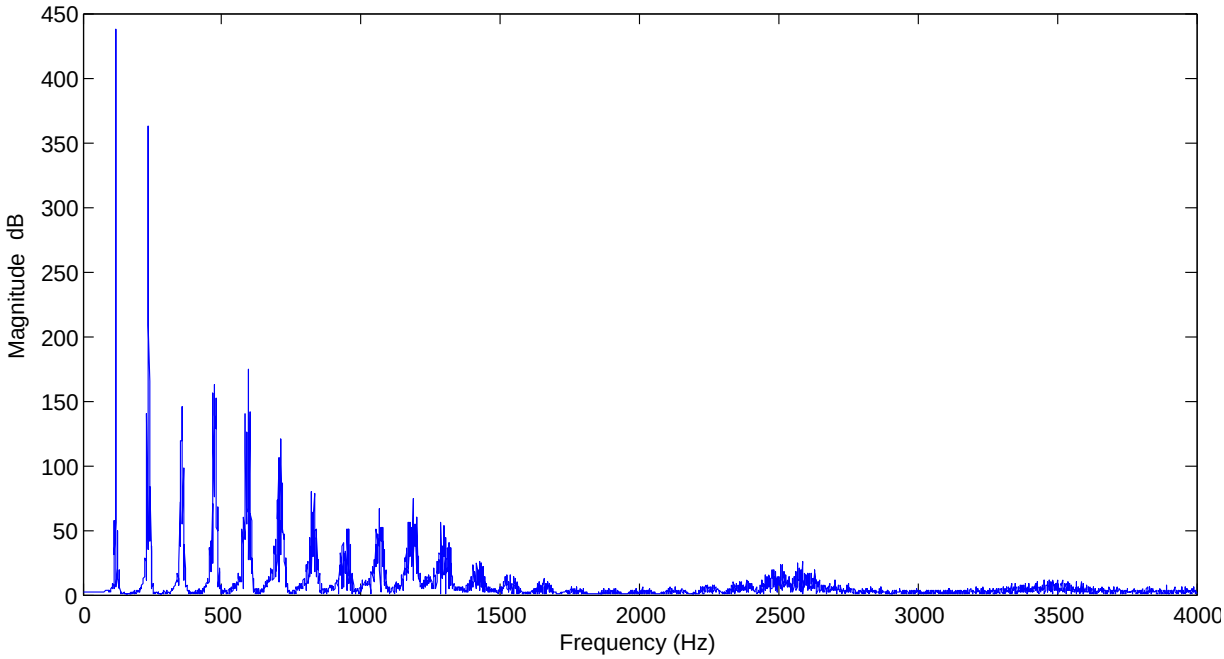


Figure 3.5 The vowel /a/ residual signal in the frequency domain.

The main difficulty with using HNM is in deriving the model parameters F_o, F_{max} and A_k shown in [16, 17];

$$y(t) = \sum_{k=1}^{K} A_k(t)\cos(k\theta(t) + \theta_k(t)) + n(t). \tag{3.5}$$

The techniques applied in solving the HNM parameters are illustrated in the sections below.

F_o and F_{max} Estimation

F_o is defined as the pitch frequency or the fundamental frequency i.e. the frequency of the first harmonic [16]. The fundamental frequency F_o was obtained using a pitch estimation method defined in [24]. The maximum voiced frequency F_{max} was obtained using [25];

$$Fmax = KFo. \tag{3.6}$$

Where F_{max} is a multiple of F_o, K is the total number of harmonics, whilst KF_o is the harmonic at which the peak amplitude drops to $13dB$.

Phase modelling

One of the main complexities faced when using the HNM model was obtaining an accurate phase component $\theta(t)$ from the speech signal [17]. Although the phase in any speech is indistinguishable by the human ear, in HNM it plays a significant role in modelling the harmonic magnitude. Because in this dissertation HNM was used only for the residual signal, a linear phase shift across all the harmonics was proposed and tested [25]. The linear phase shift relation used is shown in [26].

$$\theta_k = (3 - 2\frac{2\pi}{K})(k-1). \tag{3.7}$$

Modelling the harmonic and noise interaction

The advantage of modelling the source signal using HNM was that both voiced and unvoiced sounds could be modelled effectively. The HNM model was achieved by multiplying the harmonic components of the source signal with a noise window of equal length but having frequency characteristics determined by F_{max}. The noise window model was based on white Gaussian noise passing through a band pass filter bounded by $0.75F_{max}$ and $0.85F_{max}$. The resulting residual is shown in Fig. 3.6.

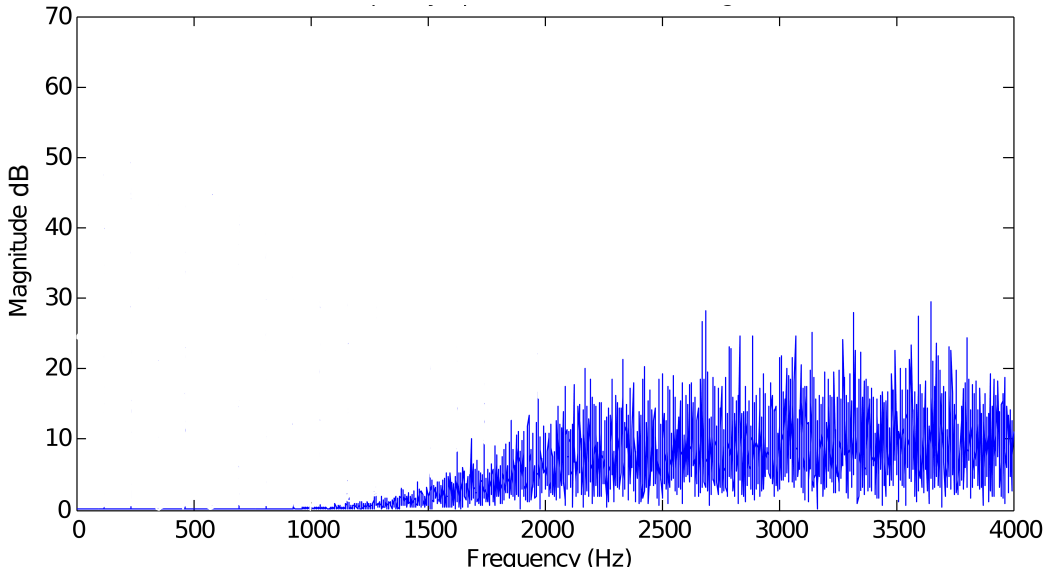


Figure 3.6 Resultant HNM residual signal.

Modelling the harmonic magnitude A(t)

The magnitudes of the residual harmonic components followed a consistent pattern when observations were made from different waveforms using the same number of LP parameters [26]. It was hence forth proposed that the magnitudes of the frequency components $A(t)$ in the HNM equation could be simplified into a time dependent function characterised by the fundamental harmonic F_o. Fig. 3.7 shows a derived scatter plot of the harmonic components of vowel /a/ speech signal. Goodness of fit tests [27], were then performed on the scatter plot with linear, quadratic and exponential functions . Results of the goodness of fit test are tabulated in Table. 3.1. The results show that the harmonic magnitudes are related to the fundamental magnitude $A(t_o)$ through an exponential equation 3.8, with an approximate 70% confidence interval.

$$A_k(t) = A_k(t_o)\exp(a_e k). \tag{3.8}$$

$a_e k$ is a constant between -0.04 to -0.08 based on the goodness of fit tests with SSE and R-square scores of around 32.9527 and 0.6016 respectively.

Table 3.1 Goodness of fit scatter plots.

Phoneme	Function	SSE	R-Square
/a/	linear	45.9052	0.4451
	exponential	34.4023	0.5841
	quadratic	32.9527	0.6016
/v/	linear	47.0052	0.4002
	exponential	35.2082	0.5744
	quadratic	33.1245	0.5912
/o/	linear	48.0100	0.4102
	exponential	34.1032	0.5740
	quadratic	32.1200	0.5702
/i/	linear	45.0001	0.4200
	exponential	34.0000	0.5900
	quadratic	32.4505	0.5875
/e/	linear	44.2050	0.4400
	exponential	34.0000	0.5890
	quadratic	30.1450	0.6210

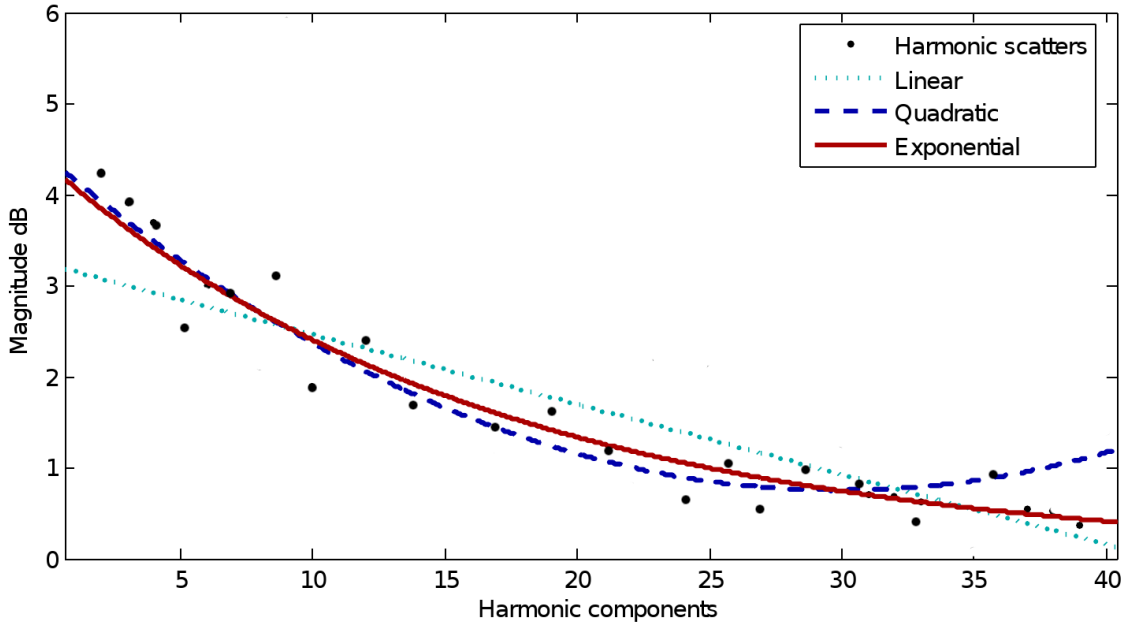


Figure 3.7 Scatter plot of harmonic components [26].

HNM source signal model

The derivations in this section enabled me to formulate an HNM based residual signal for LP speech synthesis. The mathematical representation of the residual signal is described in equation 3.9 and diagrammatically shown in Fig 3.8. With the residual and LP filter coefficients known, the LP speech equation 3.10, could thus be transformed into a speech model equation 3.12.

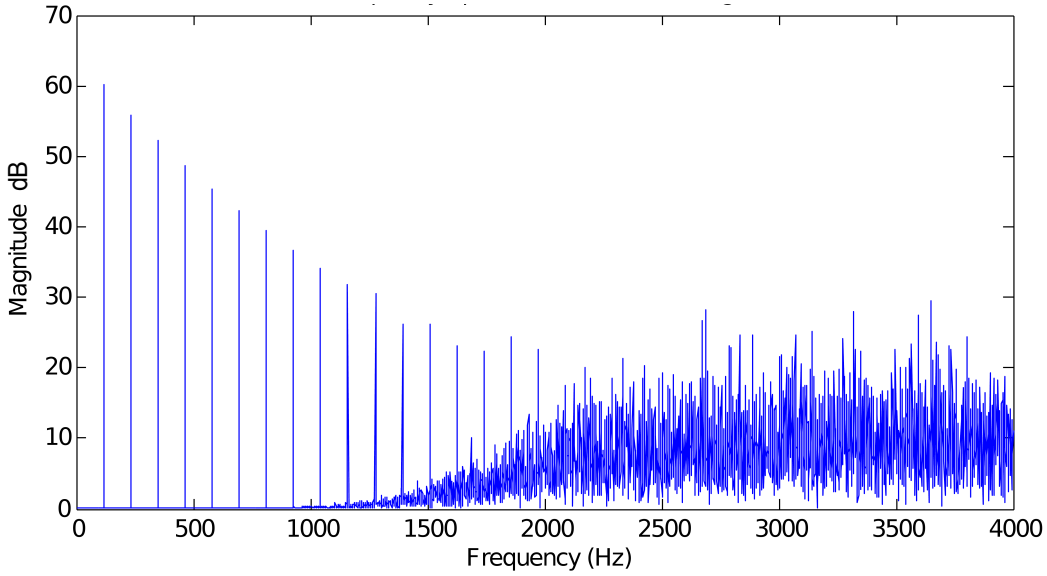


Figure 3.8 Modelled HNM residual signal for the vowel /a/ [26].

$$e(n) = \sum_{k=1}^{K} A_k(t_o) \exp(a_e k) \cos(k\theta(t) + (3 - 2\frac{2\pi}{K})(k-1)) + n(t). \tag{3.9}$$

Recalling that

$$x[n] = \widetilde{x}[n] + e[n]. \tag{3.10}$$

$$\widetilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k]. \tag{3.11}$$

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] + \sum_{k=1}^{K} A_k(t_o) \exp(a_e k) \cos(k\theta(t) + (3 - 2\frac{2\pi}{K})(k-1)) + n(t). \tag{3.12}$$

3.2.4 Discussion

In this section we discussed the various source signal techniques available for LP speech synthesis. The section also illustrated two new source signal modelling techniques using the Harmonic Plus Noise Model and the modified R-K model. Experiments were conducted on the quality of speech

produced using the various source signal models. The results illustrated that the best quality speech was produced using the HNM source signal model. A speech model was then built based on the HNM and LP. In order to outline the advantages of the derived model in equation 3.12, an investigation into ways of improving the parametric corpus was carried out.

Chapter 4

Parametric optimisation

4.1 Introduction

The parametric corpus is defined by the number of LP and residual signal parameters. The improvements carried out on the parametric corpus involved optimising both the number of LP parameters and the window lengths.

4.2 Optimising the number of LP parameters

An experiment was conducted to identify the effects of varying the number of LP parameters on the characteristics of the residual signal [26]. The process involved conducting a goodness of fit test on the residual harmonics to the proposed exponential harmonic model. It was discovered that the behaviour of the residual harmonic amplitudes $A_k(t)$ was consistent with the number of LP parameters. The optimal number of parameters was supposed to produce harmonic amplitudes that exhibit a perfect goodness of fit test with the exponential function. Fig. 4.1 - Fig. 4.4 below illustrate the effects of varying the number of LP parameters on the residual vowel /a/ in the frequency domain. In order to perform the goodness of fit test, scatter plots such as in Fig. 4.3 were developed for each residual signal obtained from varying LP parameters.

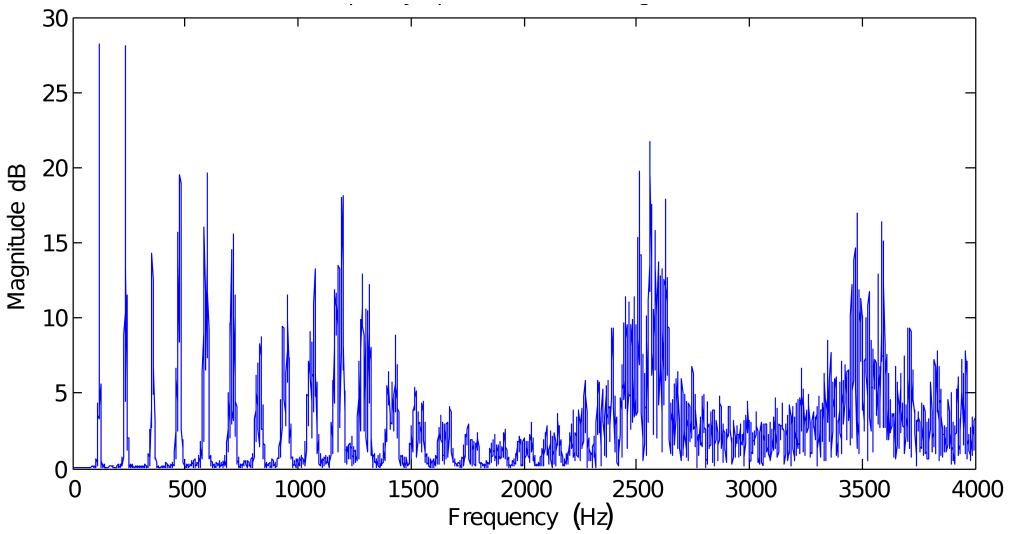


Figure 4.1 Residual signal derived from using 2 LP parameters.

The residual signal derived from using 2 LP parameters exhibits random harmonic magnitude characteristics as shown in Fig. 4.1. From this observation it was concluded that two LP parameters were inadequate in producing harmonic characteristics that match the exponential function proposed in the speech model.

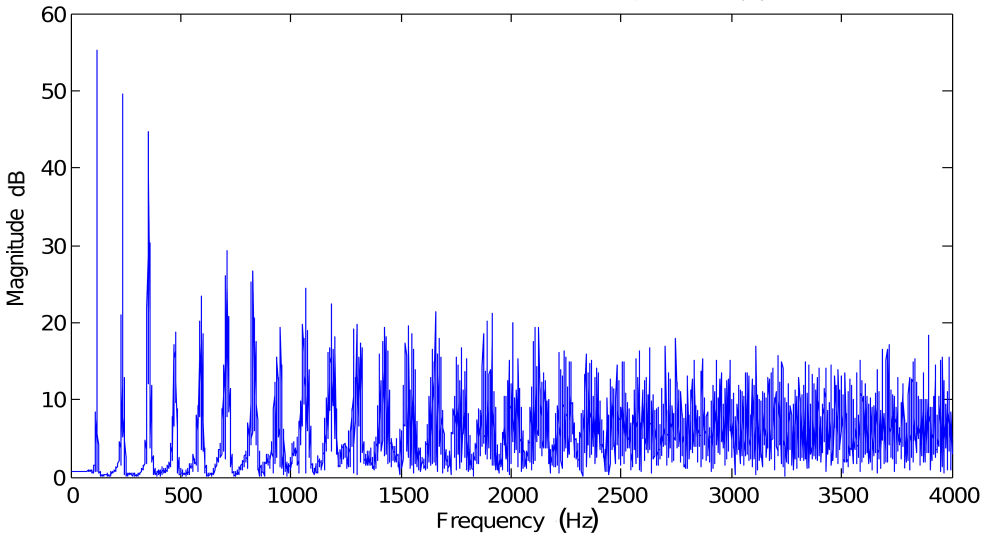


Figure 4.2 Residual signal derived from using 10 LP parameters.

The residual signal derived from using 10 LP parameters exhibits exponential harmonic characteristics as shown in Fig. 4.2. From this observation it was concluded that a minimum of 10 LP parameters were suitable for modelling the LP and HNM based speech synthesiser.

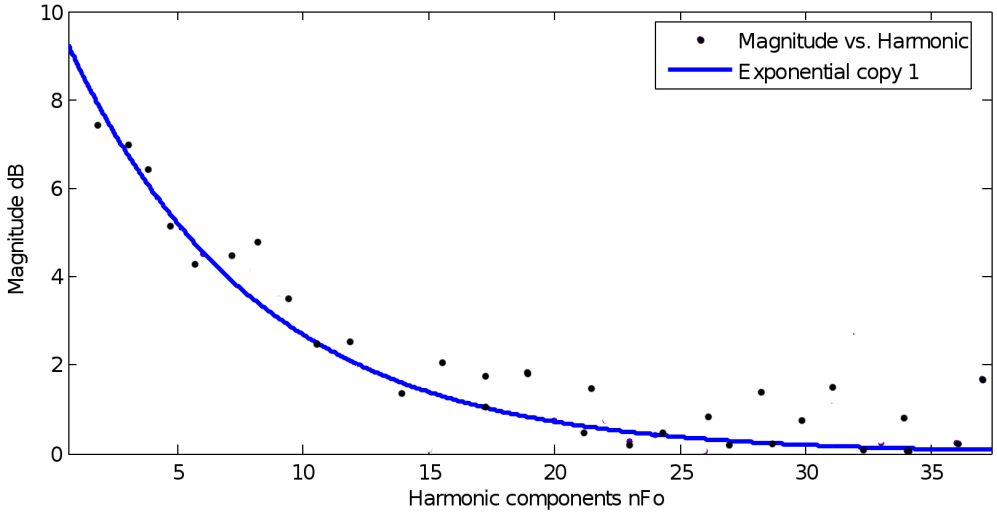


Figure 4.3 Residual signal scatter plot derived from inverse LP analysis.

When using more than ten LP parameters it was observed that the residual signal exhibits exponential harmonic characteristics as shown in Fig. 4.4. This meant that a saturation point had been reached on the significance of using a greater number of LP parameters.

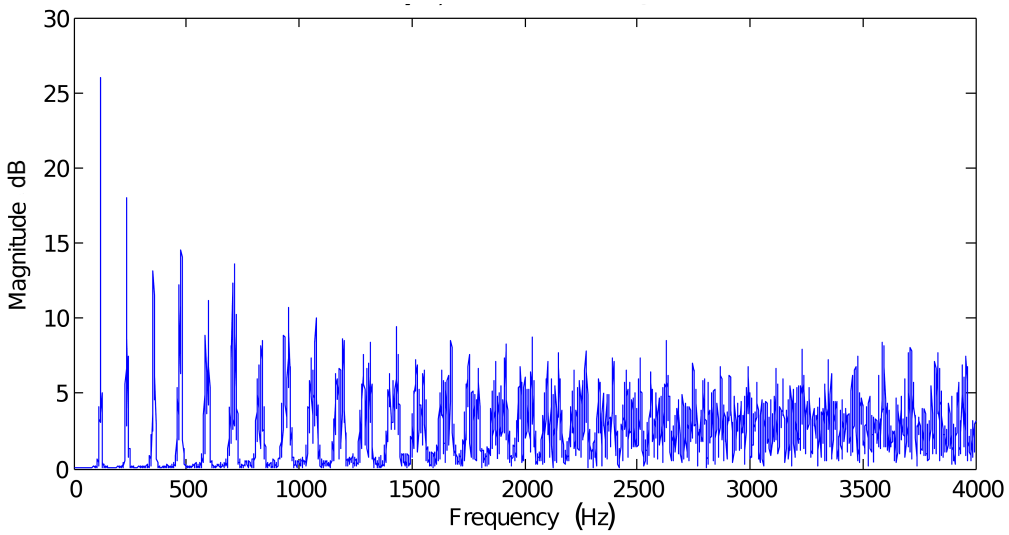


Figure 4.4 Residual signal derived from using 20 LP parameters.

Table. 4.1 shows the results of the goodness of fit test on a vowel /a/ residual using a different number of LP parameters. These tests confirmed observations made by the author in *section 4.2*. In Table. 4.1 it is observed that at less than 10 LP parameters the residual signal does not fit accurately the proposed harmonic function. However, at a higher number of LP parameters the residual fits suitably the exponential model with a confidence interval of approximately 64% 0.6412 on R-Square.

Table 4.1 Goodness of fit results on variable LP parameters.

Number of LP	Function Fit	SSE	R-Square
2	exponential	94.1021	0.1341
5	exponential	74.2347	0.3711
10	exponential	40.0040	0.5564
12	exponential	44.0000	0.5600
14	exponential	30.1034	0.6412
16	exponential	28.4000	0.6764
18	exponential	27.5259	0.7022
20	exponential	27.5200	0.7000

4.3 Optimising the window length

The main restriction of LP based synthesis is that signal analysis can only be carried out at specific window lengths or segments usually $30 - 50$ms long [4]. Suppose the same parameters were to be used at larger window lengths then the entire parametric corpus in our phonetic speech synthesiser is reduced. An experiment was performed [26] by the author to find the optimal window length. As with optimising the number of LP parameters the experiment involved using a goodness of fit test on the harmonic amplitudes produced at variable window lengths. Table. 4.2 derived from [26] shows the results of the goodness of fit test conducted to find the optimal window length.

Table 4.2 Goodness of fit results on variable window length.

Window length	Function Fit	SSE	R-Square
6.25ms	exponential	90.3022	0.1941
62.5ms	exponential	40.4446	0.5665
125ms	exponential	30.0040	0.6865
250ms	exponential	33.1034	0.6012
500ms	exponential	35.2082	0.5504

From Table. 4.2 it can be deduced that from $125 - 150$ms the residual signal optimally fits the proposed speech model.

4.4 Chapter discussion

An optimal number of 10 LP parameters for every 150 ms of speech segment was arrived at based on experiments conducted in [25,26] and described briefly in this section. Once the optimal number of LP parameters had been established, the "FPGA based phonetic speech synthesiser" could be designed. The next section discusses the implementation of the speech synthesis model. The chapter explains the speech synthesis process adopted in this dissertation from analysis through to synthesis including standard speech analysis and synthesis procedures.

Chapter 5

Speech synthesis design

5.1 Introduction

A design methodology of a speech synthesiser involves three processes namely speech gathering, speech analysis and speech synthesis. This chapter presents the design methodology of the speech synthesiser in detail. The design methodology was built around the literature and experimentation discussed in the previous chapters.

5.2 Speech recordings

The first step in the methodology was obtaining recorded speech segments of the English language. Recordings were done using a PC sound card and a wave editor NERO^{TM}. Phonetic sounds were uttered by me for all the sounds in the British English phonetic database. It was however soon discovered that because the recording environment was noisy the recordings were not clear and in most instances the amount of Gaussian white noise was quite large. In order to solve this problem the recordings used in this thesis were done in a professional studio [28]. The sourced recordings were all at a frequency of 44 kHz, a standard for all music and audio signals as this accomodates all the speech frequencies [28]. An experiment using NERO^{TM} wave editor was performed to downsample the speech recordings to 8 kHz. According to the down sampling formulae in [29], down sampling reduces the amplitude of the harmonics by a down sampling factor M. This is illustrated in Fig. 5.1 and Fig. 5.2 showing frequency domain segments for the vowel /a/ at different sampling rates.

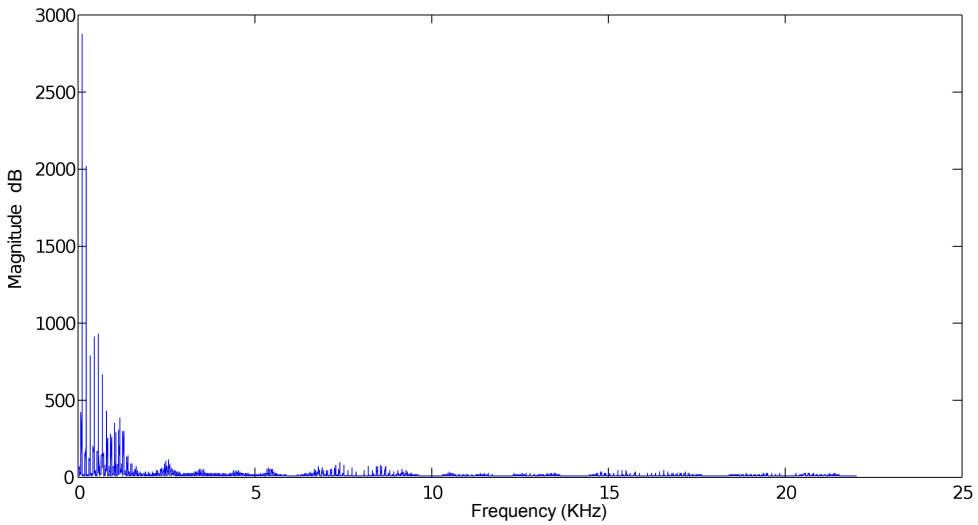


Figure 5.1 Vowel /a/ at 44kHz in the frequency domain.

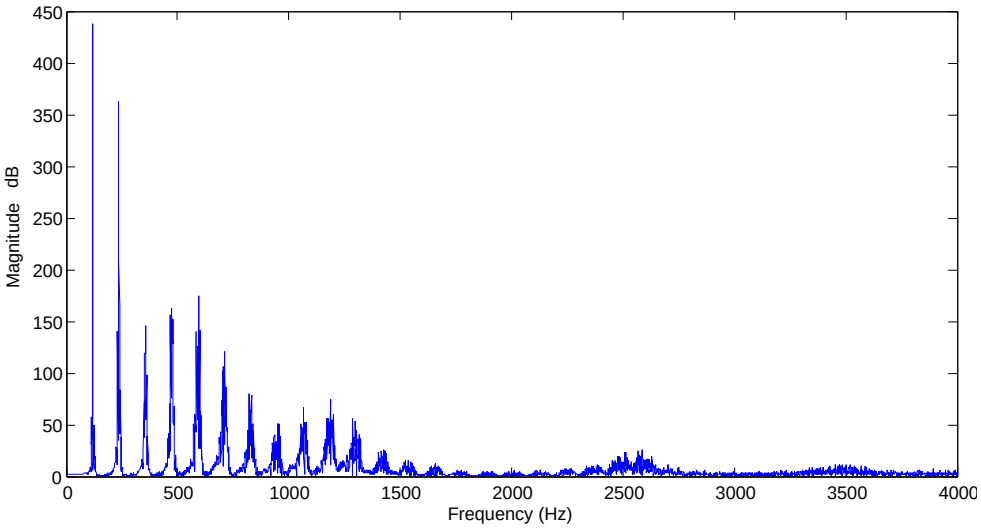


Figure 5.2 Vowel /a/ at 8kHz in the frequency domain.

5.3 Speech analysis

Most of the sourced speech recordings were of length greater than optimal window length of 150 ms. In order to accommodate the extra lengths, speech segments were trimmed into finite time samples. If we take the finite time segment of a sampled signal and evaluate the Discrete Fourier Transform (DFT) we suffer spectral leakage [11]. The spectral leakage phenomena is caused by the frequency response of the rectangular filter, which corresponds to the truncation of the signal. Fig. 5.3 below shows how the spectral leakage can be reduced by using different truncation techniques namely Hamming, Blackman, Gaussian and Hanning.

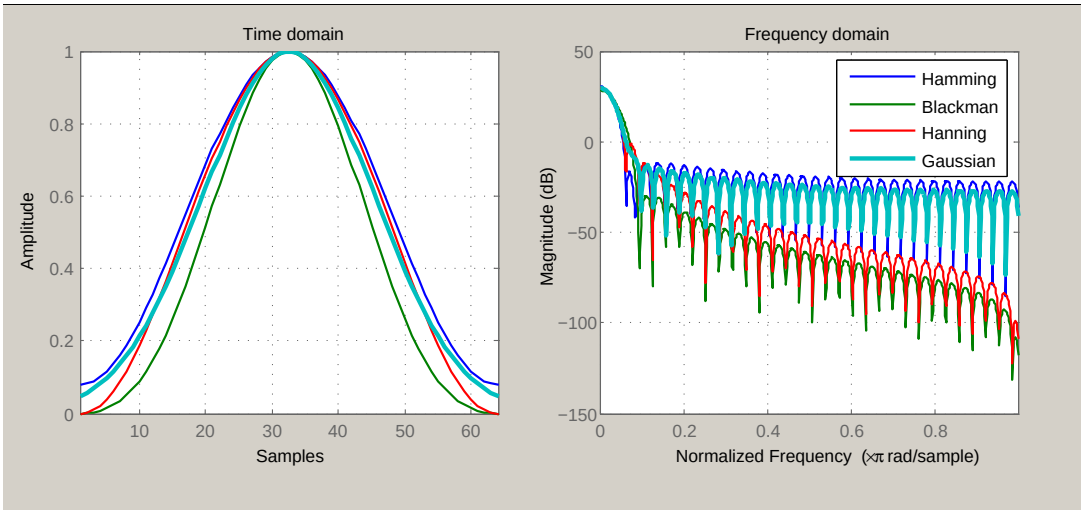


Figure 5.3 Different trunctation window spectral leakage.

The Hamming window has been shown from experimentation in [11] to be the best window for speech analysis purposes. Fig. 5.4 shows a time domain signal for a speech signal passing through a Hamming filter. It is important to note that in order to preserve the amplitudes at the truncated ends of the signal, the next segment is evaluated from half the window segment. Fig. 5.5 shows the typical layout of multiple speech segments passed through Hamming filters as a chained signal.

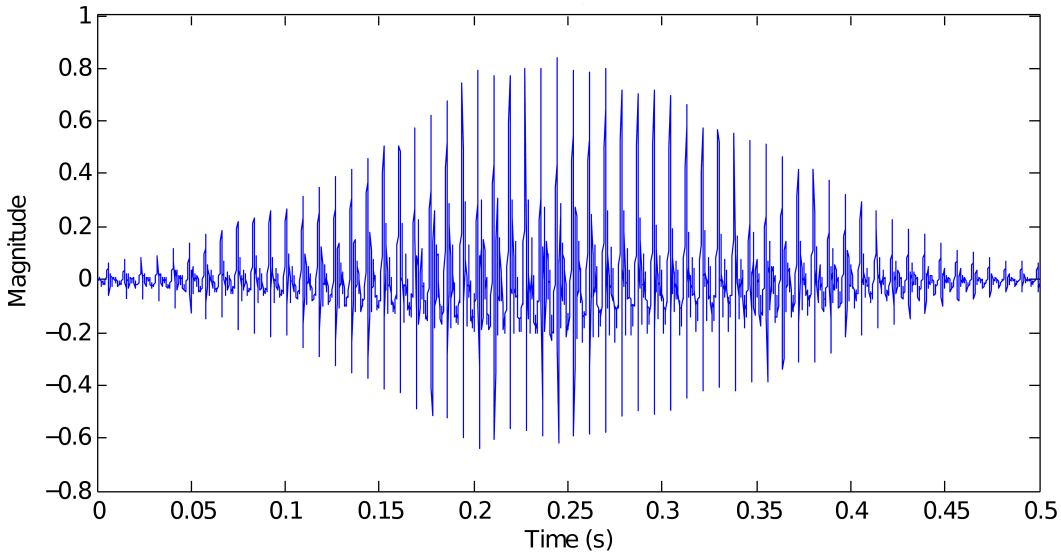


Figure 5.4 A hamming filter output of the vowel /a/ speech segment.

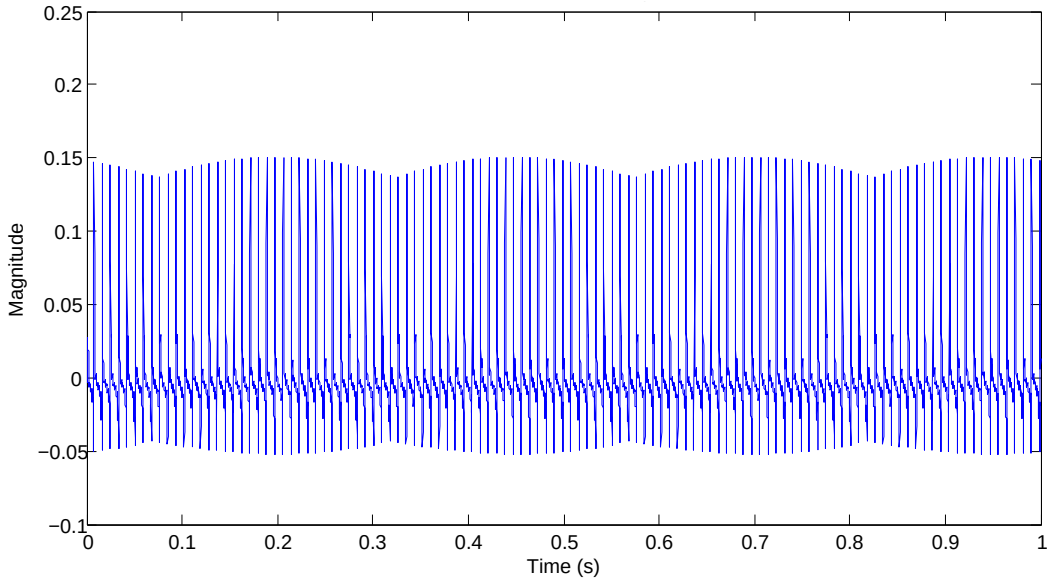


Figure 5.5 Chained hamming signal vowel /a/.

5.4 Inverse LP analysis

A process of inverse LP analysis was used to obtain the LP parameters from the recorded speech segments. The inverse LP analysis principle works on the auto regressive model shown in equation 2.1 [7]. The idea behind linear prediction analysis is to obtain the linear prediction coefficients a_k such that the mean square error i.e. (the difference between the predicted and the original signal) is zero [4]. This yields equation 5.1. Substituting equation 2.1 in equation 5.1 results in equation 5.2, when recalling $e[n]$ is ideally 0.

$$0 = x[n] - \widetilde{x}[n]. \tag{5.1}$$

$$0 = x[n] - \sum_{k=1}^{p} a_k x[n-k]. \tag{5.2}$$

If the original speech signal $x[n]$ is known, then two mathematical methods namely Cepstral coefficients and the lattice matrix can be used to obtain the LP parameters a_k [4]. In this thesis a MATLAB^{TM} Signal Processing Toolbox was used to compute the LP parameters.

5.5 Phoneme analysis

The entire memory footprint for the parametric corpus contained 44 English phonemes with 10-15 LP parameters per each 150 ms recorded speech. It was noted that for nasal sounds and plosives

the entire recording was at times less than 150 ms. In such cases the speech segment was analysed for the phoneme duration. In most cases the recorded speech segment was greater than 150 ms therefore the phoneme was divided into 150 ms speech segments. This was mostly the case for vowel sounds. This variation allowed for a variable number of LP parameters to be tagged per phoneme.

5.5.1 Phoneme parametric corpus

The corpus of all 44 English phonemes contained a total of approximately 2000 parameters of the suggested model, totalling to footprint of approximate 2 kB based on the assumption that 1 Byte could sufficiently carry 1 parameter. On average each phoneme was divided into $2-4$ windows with 10-15 LP parameters and 5 residual model parameters, namely:

F_o : the fundamental frequency,

K : number of harmonics,

A_n : magnitude of the noise,

A_k : harmonic magnitude and

a_e : exponent factor harmonic magnitude.

Table. 5.1 shows the detailed set of residual and LP parameters obtained from analysing the vowel /i/. Appendix A shows the entire parametric library for the suggested HNM model for all English phonemes. Vowel words were also included for testing methods using the same parametric model experiments as demonstrated in [26].

Table 5.1 Parametric speech corpus for HNM LP model vowel /i/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/i/	1.0000	1280	121	0.0016	0.04	0.0070	$2\pi/10$
	-0.2257						
	-0.4786						
	-0.5460						
	-0.3810						
	1.0218						

Continued on next page

Table 5.1 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.4970						
	-0.0025						
	-0.0041						
	-0.4587						
	0.1553						
/i1/	1.0000	1300	124	0.0017	0.05	0.0075	$2\pi/11$
	-0.2001						
	-0.3710						
	-0.5901						
	-0.4213						
	0.9701						
	0.3000						
	0.0001						
	-0.1008						
	-0.3201						
	0.0700						
/i2/	1.0000	1340	122	0.0018	0.04	0.0075	$2\pi/10$
	-0.1680						
	-0.5300						
	-0.6705						
	-0.3100						
	0.9400						
	0.3800						
	0.0040						
	-0.1320						
	-0.2860						
	0.0600						

5.5.2 Word parametric corpus

The use of phonemes in speech synthesis presents interpretation problems when performing listening tests [25, 26]. In this dissertation words instead of phonemes were used when conducting listening tests. These words amounted to 8 English words in the form of 2 plossives, 2 fricatives and 4 vowels shown in Table. 5.2. The words were added to the parametric corpus of the designed speech synthesiser as shown in Appendix A.

Table 5.2 Words included as part of the corpus.

Word No	Classs	Word
1	vowel	hello
2	vowel	hat
3	vowel	too
4	vowel	door
5	fricative	shop
6	fricative	that
7	plosive	dig
8	plosive	pit

5.6 Chapter discussion

Once the optimal number of LP parameters had been established, a model for the FPGA based phonetic speech synthesiser had to be designed. The next chapter discusses the implementation of the speech synthesis model, by explaining the methodology and design techniques used from analysis through to synthesis.

Chapter 6

Implementation of the design method

The process of speech generation involves producing software algorithms that model the speech synthesiser. This process is more commonly known as speech generation and involves building algorithms in MATLAB^{TM}, VHDL or any other high level language.

6.1 Speech generation

The speech generation process comprises of three main constituents namely: the residual signal, the filter and the output signal as illustrated in Fig. 6.1,

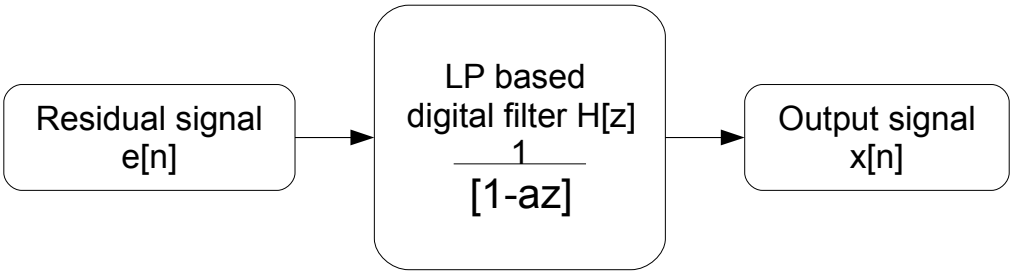


Figure 6.1 The speech synthesis block diagram.

In this dissertation the three main constituents of the speech generation process were designed namely:

- The residual signal $e[n]$: The residual signal was calculated as the difference between the predicted and the original signal. The residual signal was constructed using HNM model parameters to produce an excitation signal.

- The signal filter $H[z]$: The filter transfer function was calculated using poles derived from

the LP coefficients.

- The output signal $x[n]$: The signal was generated from filtering the excitation signal with a filter based on the transfer function $H[z]$.

6.2 Speech generation algorithm

The first synthesis algorithm was developed in MATLAB^{TM} for simulation and testing. The algorithm was developed based on the speech model presented in equation 3.12. The methodology of developing the MATLAB algorithm is best illustrated in the flowchart shown in Fig. 6.2 and the MATLAB^{TM} code listing shown below. A detailed presentation of the algorithm is illustrated in Appendix B.

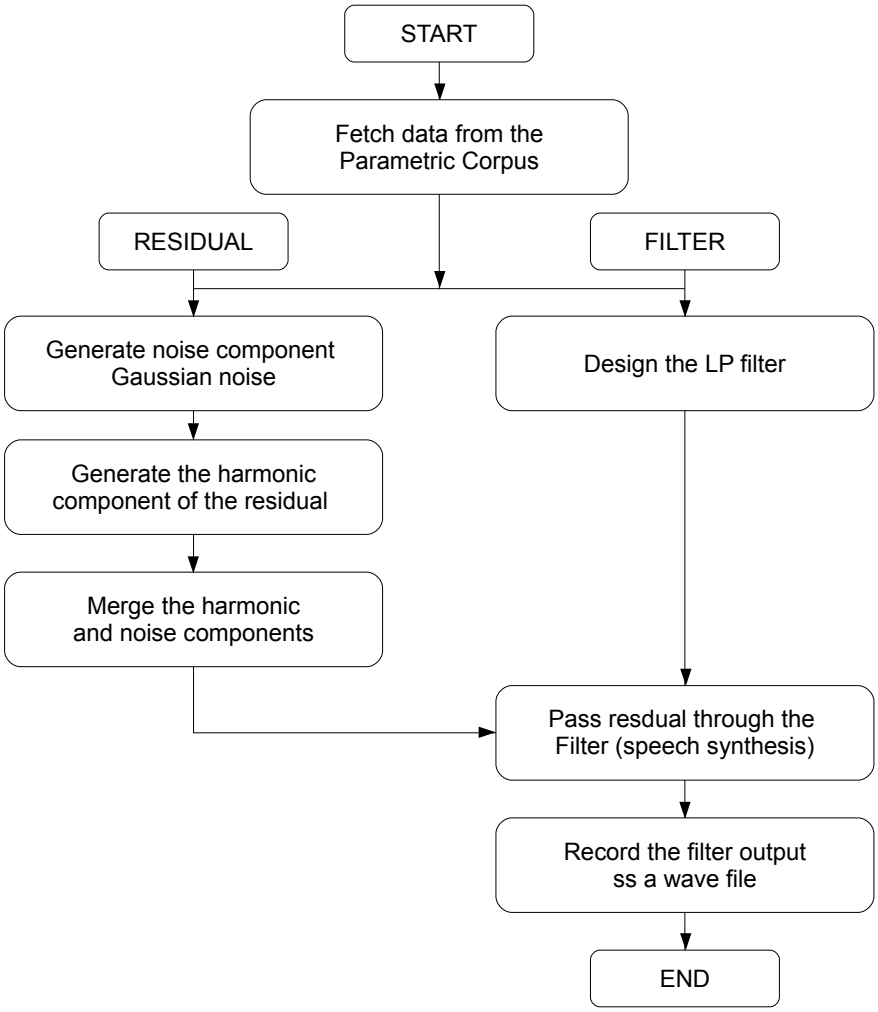


Figure 6.2 The speech synthesis algorithm.

Algorithm 1 Speech synthesiser MATLAB

Require: windowlength = 150

```
wn = wgn(8000,1,2)
b = remez(20,[0 0.05 0.88 1],[0 0 1 1])
a = [1]
```

Ensure: wn = filter(b,a,ws);

```
for (t = 1:8000) do
    yz(1) = 0
    for (i = [2:20] do
        yz(i) = yz(i-1)+(.015*exp(0.07*(1-(i-1))))*cos((2*pi*(116/8000)*(i-1)*t)+0.72*(i-2))
    end for
    yz(t)=yz(20)
end for
yout = yz + 0.09*ws1
yb1 = fft(yz1)
plot(l,abs(yb1(1:4000))); title('Frequency Spectrum HNM Source Signal /a/')
reconstructed = filter(1,ak,yout)
plot(i,predyb(1:4000)); title('Frequency Spectrun Residual Signal Vowel /a/')
wavwrite(reconstructed,aout)
```

Once this algorithm had been developed speech could be generated by feeding the model parameters to the algorithm. The output signal of this process was then recorded and stored as a wave file that could be played using most media player software or fed directly to the digital audio output.

6.3 Analysis of the speech output

Output waveforms were compared to the original speech segment in both frequency and time domains. Fig. 6.3 shows the predicted vowel /a/ signal against the original analysed waveform in

the time domain. The predicted waveform is shifted by 2 ms from the original waveform in order to accurately distinguish the two signals. It is evident from Fig. 6.3 that the synthesised and the original signals exhibit similar characteristics.

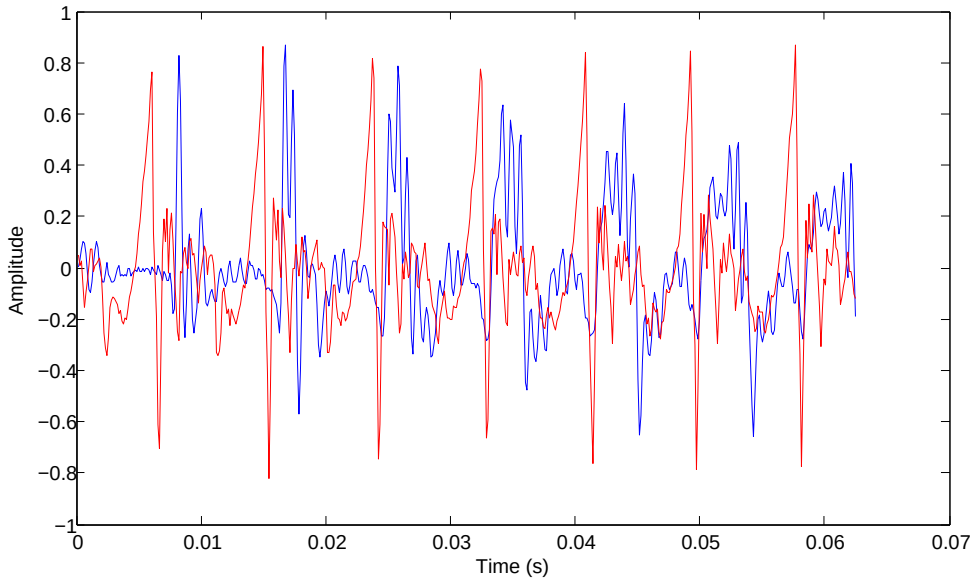


Figure 6.3 A comparison of the synthesised (red) and original (blue) signal.

6.4 Spectrogram analysis

One of the best ways to analyse speech signals is through a spectrogram [11]. The spectrogram looks at the speech signal in both the time and frequency domains. A spectrogram distinguishes clearly between the voiced and unvoiced sounds in the speech recording by presenting the signal in the form of energy formants. High energy formants, usually harmonics, are depicted in a darker colour whilst the low energy formants are in a lighter colour. Fig. 6.5 shows the spectrogram of the synthesised vowel /a/ signal whilst Fig. 6.4 shows the spectrogram of the original signal. From the two spectrograms it was observed that the original and the synthesised signal exhibit similar voicing formants and characteristics, indicated by the dark lines of the spectrum.

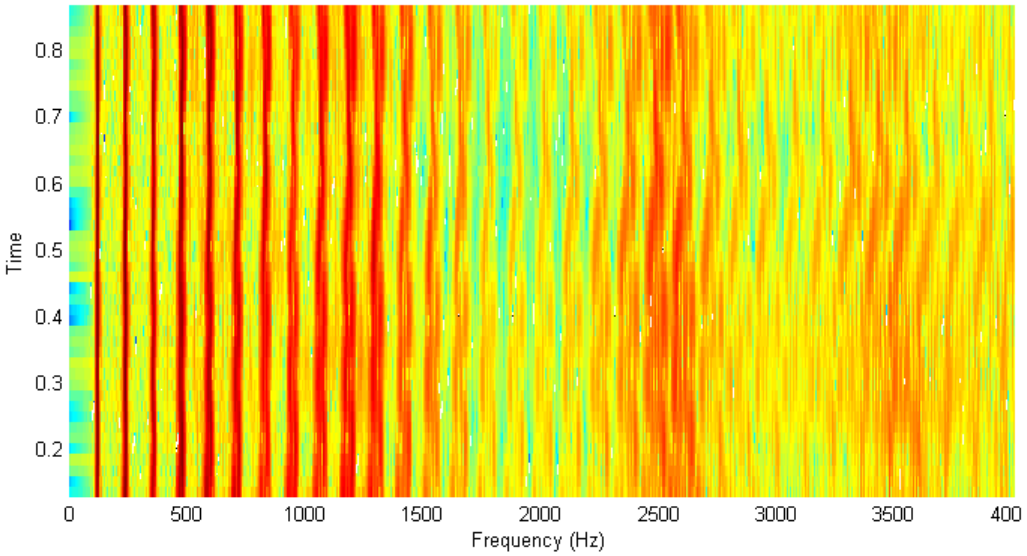


Figure 6.4 A spectrogram analysis of the original vowel /a/.

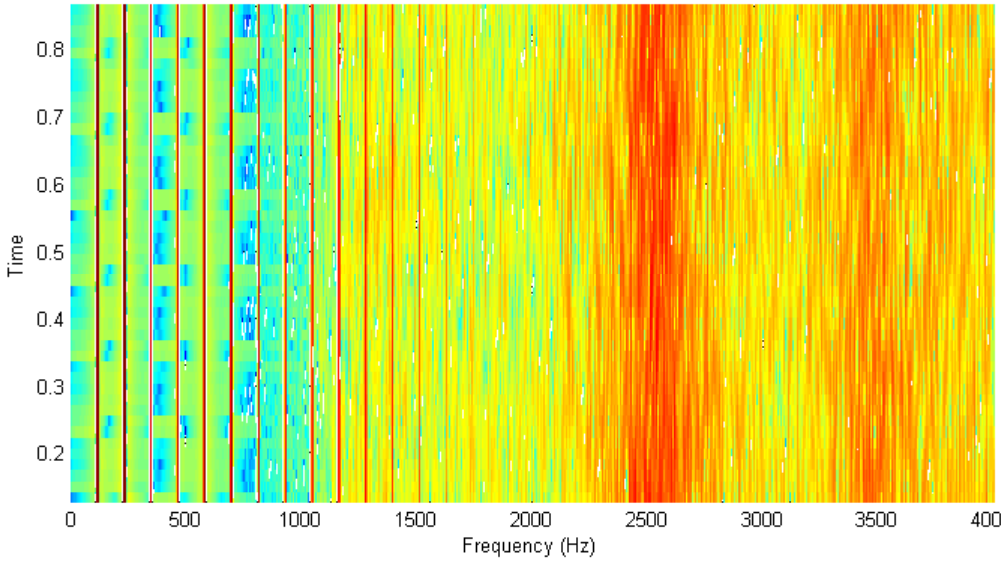


Figure 6.5 A spectrogram analysis of the synthesised vowel /a/.

6.5 Listening tests

Listening tests are a standard procedure used in testing the quality of the speech. The most common listen tests include the MOS and the transcription test also known as QOS [23]. In this dissertation both MOS and QOS tests were conducted by a group of 20 native South African english speakers from the University of Witwatersrand, Johannesburg.South African english speaker were chosen because the parametric corpus had been built on a South African english database. In order to have a good comparison of the speech output, the original speech segment was played first followed

by the synthesised speech segment. The listener was not made aware of which sound was being played at the time of scoring. All the users were briefed on the experiment and the testing criteria used on MOS and QOS tests as described below.

6.5.1 Mean opinion score tests

Each listener from the test sample was asked to give a score from $1-5$ on the quality of the uttered speech. Table. 6.1 shows some results of this mean opinion score MOS. The score is an average from all the listeners on their MOS interpretation of the original speech segment and the synthesised segment.

Table 6.1 MATLAB^{TM}based mean opinion scores (words).

Word	Class	Original recording (MOS)	Synthesised recording (MOS)
hello	vowel	4.3	3.8
hat	vowel	4.0	3.8
too	vowel	4.0	3.5
door	vowel	4.2	4.0
shop	fricative	4.1	3.7
that	fricative	4.5	4.1
dig	plossive	4.4	4.0
pit	plossive	3.9	3.7

It was stated in the previous section that the reason why tests were performed on uttered words rather than the phonemes was the difficulty in interpretation by the listeners. To illustrate the problems in interpreting phonemes MOS tests were performed on synthesised phonemes using the same test sample of 20 University students. The results of the MOS tests conducted on the uttered phonemes are tabulated in Table. 6.2 below.

Table 6.2 MATLAB^{TM}based mean opinion scores (phonemes).

Word	Class	Original recording (MOS)	Synthesised recording (MOS)
/a/	vowel	4.5	3.0
/e/	vowel	4.0	4.2
/i/	vowel	3.9	3.5
/o/	vowel	4.1	2.7
/s/	fricative	4.9	3.4
/h/	fricative	4.6	4.1
/d/	plossive	4.5	4.5
/p/	plossive	3.9	1.7

6.5.2 Transcription tests

Transcription tests entail the sample space to utter back the words that have been played, the scoring is based on the users ability to interpret correctly the uttered words [23]. The same synthesised words as with the MOS test were played to listeners, each listener was asked to re-pronounce the word he/she had just heard. Table. 6.3 displays the results of the transcription test.

Table 6.3 MATLAB^{TM}based transcription scores (words).

Word	Class	Original recording (QOS)	Synthesised recording (QOS)
hello	vowel	100%	99%
hat	vowel	100%	98%
too	vowel	99%	99%
door	vowel	100%	94%
shop	fricative	100%	98%
that	fricative	100%	94%
dig	plossive	100%	99%
pit	plossive	99%	98%

Similar to the MOS test, the same experiment was conducted for synthesised phonemes. As with the MOS test the listeners struggled to interpret plain phonemes as shown in Table. 6.4.

Table 6.4 MATLAB^{TM}based transcription scores (phonemes).

Word	Class	Original recording (QOS)	Synthesised recording (QOS)
/a/	vowel	90%	82%
/e/	vowel	99%	98%
/i/	vowel	84%	86%
/o/	vowel	99%	99%
/s/	fricative	98%	97%
/h/	fricative	80%	75%
/d/	plossive	94%	90%
/p/	plossive	79%	67%

6.6 Discussion of results

From Table. 6.1 it can be noted that the synthesised speech performed exceptionally well on the MOS score. One of the main reasons that could be attributed to the significantly high score is that words and not sentences were used in conducting the tests. By using short words the listener was able to concentrate and interpret the words easily as shown in Table. 6.3.

Listening tests conducted on phonemes proved to be trivial as there were lots of discrepancies on the results Table. 6.2. This was attributed to the fact that some of the phonemes like /a/ /e/ /d/ and /o/ sound very similar. It was also observed that in most cases the users could not distinguish between the original and synthesised utterances. The similarity in both the QOS and MOS test result was attributed to the fact that the same test sample was used for both listening tests.

Chapter 7

Embedded development

7.1 Introduction

The MATLAB^{TM}algorithm described in the previous chapter had to be converted to an embedded algorithm using the American National Standards Institute C (ANSI C) programming language. ANSI C is the most common programming language used in embedded development. MATLAB^{TM}has a similar syntax to ANSI C but is embedded with mathematical functions such as *sine*, *cosine* and *exponent* that are not available in ANSI C. The problem arises with the fact that ANSI C programs are mostly procedural and work best with microprocessors. The architecture of the speech synthesis model requires real time processing difficult with modern microprocessors but possible with the FPGA. The FPGA, unfortunately, is not programmed in ANSI C but rather in VHDL. Converting the MATLAB^{TM}code to VHDL posed more complications because of the absence of floating point numbers and simple mathematical functions like division and multiplication. The chapter explains how some of these complications were solved.

7.2 The VHDL platform

Choosing the right FPGA platform is important when developing VHDL based algorithms as this determines how much work will be put into code development. Some FPGA devices have built in multipliers and others have toolkits to make developing *sine* and *cosine* functions easier. Taking this into consideration the Xilinx^{TM}family of devices was chosen. The actual device sourced was the XC3S1600E though the design was built around a smaller system the XC3S400E with only 400000 system gates. The specifications for the device are listed in Table. 7.1 [30]. The

Xilinx9.2i^{TM}software [30] provides a platform for developing code for the Xilinx^{TM}range of devices. The software includes a simulator for code testing purposes. All code was designed on an 18 bit signed number network.

Table 7.1 Characteristics of the Xilinx XC3S1600E FPGA device.

No	Property	Quantity
1	System Gates	1.600K
2	Logic Cells	33,192
3	Block RAM Bits	648K
4	Distributed RAM Bits	231K
5	DCMs	8
6	Multipliers	36
7	I/O Standards	18
7	Max Single Ended I/O	376
8	Max Differential I/O Pairs	156

7.3 VHDL code development

Due to the absence of complex mathematical functions in VHDL, a Maclaurin series expansion was adopted for modelling all complex mathematical functions embedded in;

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] + \sum_{k=1}^{K} A_k(t_o)\exp(a_e k)\cos(k\theta(t) + (3 - 2\frac{2\pi}{K})(k-1)) + n(t). \tag{7.1}$$

The Maclaurin series expansion [31] defines complex mathematical functions as an infinite sum of polynomial terms as shown in equation 7.2 and equation 7.3 [31] for the cosine and exponent functions, respectively.

$$cos[x] = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}. \tag{7.2}$$

$$\exp[x] = \sum_{n=0}^{\infty} \frac{(x)^n}{n!} for|x| \leq 1, x \neq 1. \tag{7.3}$$

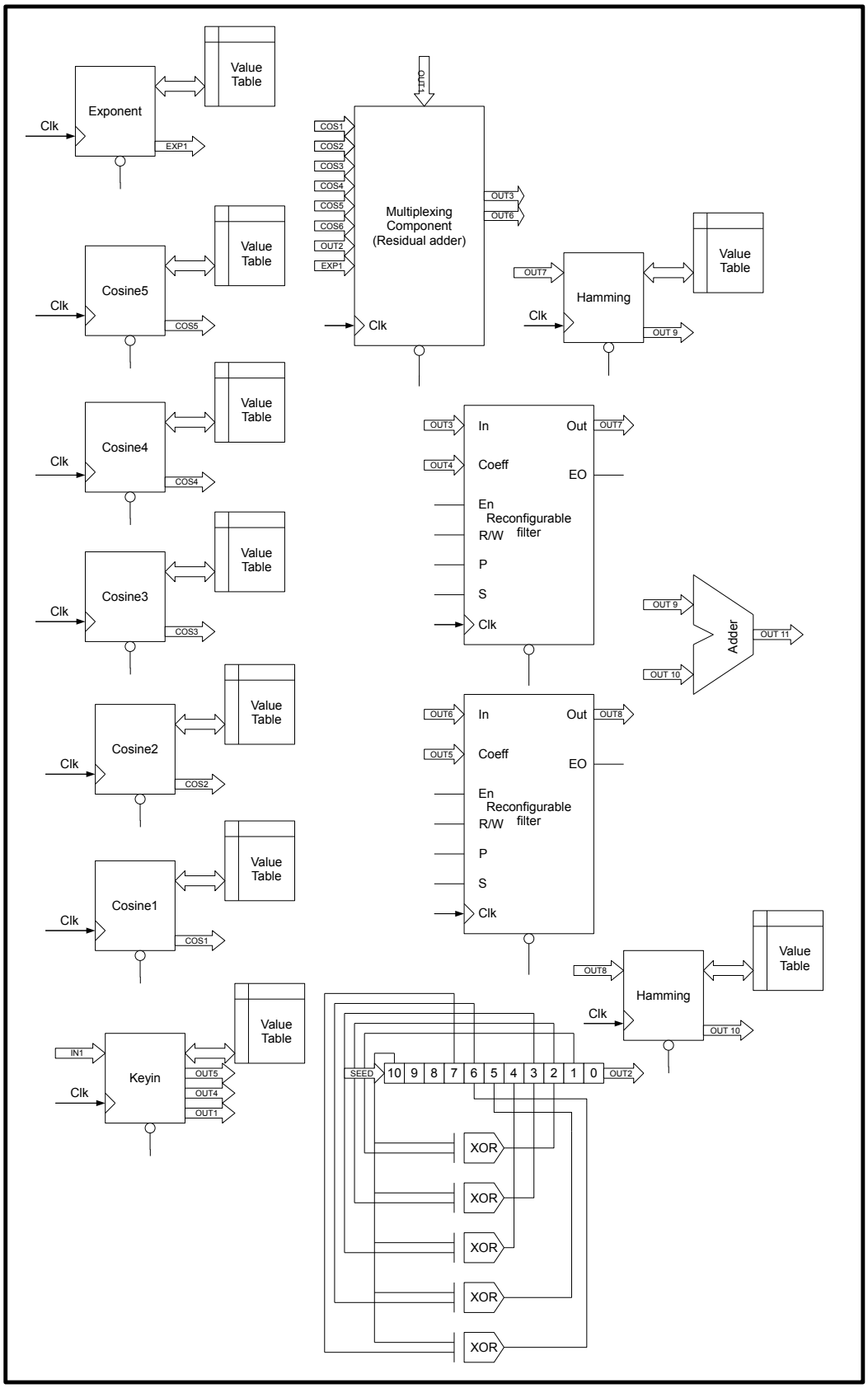


Figure 7.1 The speech synthesis circuit.

Using Maclaurin series expansions meant that the speech algorithm could be subdivided into subcomponents. Dividing the algorithm into subcomponents meant that VHDL code could be developed in a sequential manner, with individual unit testing at various stages. The various unit testing stages established for the speech synthesis model are illustrated as a schematic in Fig. 7.1.

7.4 Modelling the signal frequency clock

7.4.1 Algorithm development

In order to produce a real time system the entire VHDL model had to be synchronised at 8 kHz. A clocking component was designed using a digital counter, comparator and the crystal frequency of 250 MHz. The block diagram in Fig. 7.2 shows the detailed schematic of the frequency/clocking component.

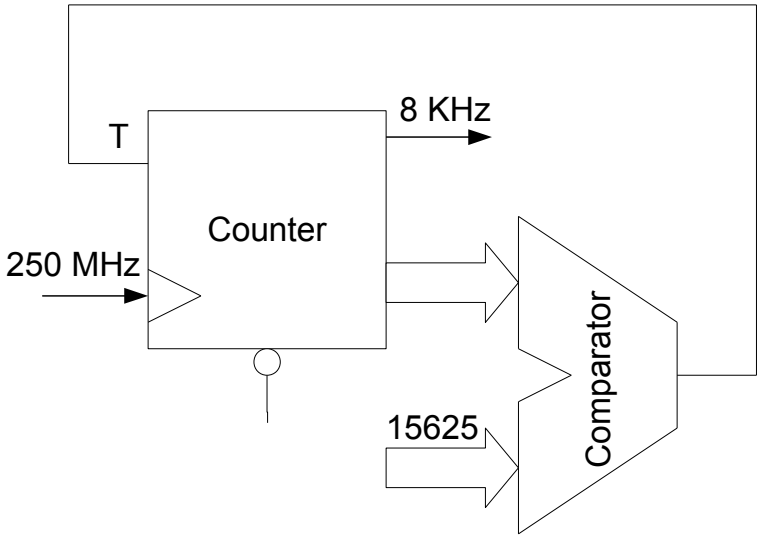


Figure 7.2 Schematic of the 8 kHz block component.

7.4.2 Simulation and testing

Once the VHDL code for the component had been developed as shown in Appendix B, a test bench was designed in order to simulate the output of the developed code. A 250 MHz clock trigger was used in the simulation for a period of 10ms. Fig. 7.3 shows the simulated results of the clocking component on the Xilinx9.2i^{TM}platform.

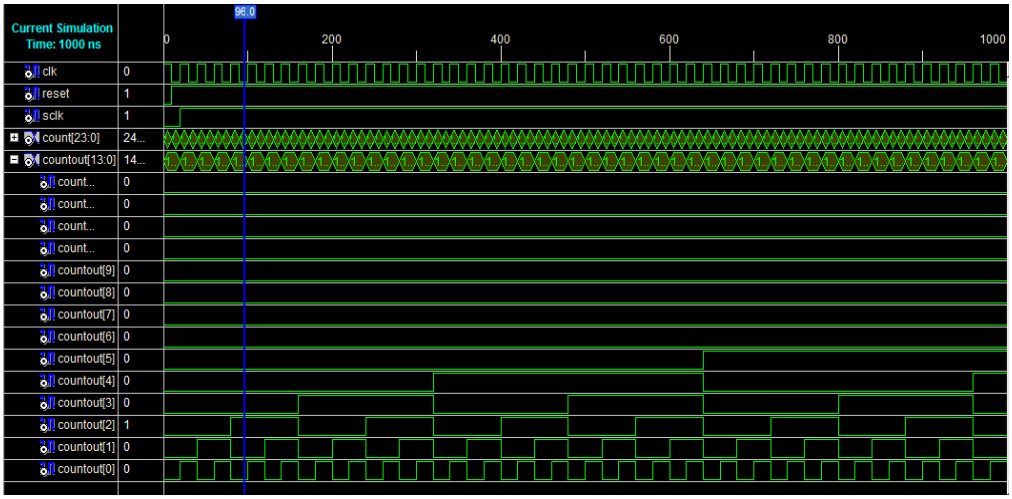


Figure 7.3 A simulated view of the 8 kHz block component in Xilinx.

The following signals can be identified from the simulation:

- Clk: A crystal input of 200 MHz.
- Countout: An internal counting sequence signal.
- Count: An internal counting sequence signal.
- Sclk: An output signal of 8 kHz.

7.5 Modelling the noise component

7.5.1 Algorithm development

White noise in speech synthesis is modelled using a Box-Muller method [32]. A Box-Muller method generates pairs of standard normally distributed random numbers, given a source of uniformly distributed random numbers [32]. The basic form of the Box-Muller method is defined in [32];

$$X = \sqrt{-2\log(U_1)}\cos(2\pi U_2). \tag{7.4}$$

$$Y = \sqrt{-2\log(U_1)}\sin(2\pi U_2). \tag{7.5}$$

Where U_1 and U_2 are two uniformly distributed random numbers, X and Y are standard normally distributed randoms. A Box-Muller equation requires complex mathematical manipulations to develop on embedded platforms. Given that the accuracy of the noise distribution was not important, a simpler approach using random number generators was adopted. Constructing the

random number generator in ANSI C or other embedded languages entails the user call upon a pre-built random number generator. In VHDL the random noise generator can be constructed using a pseudo binary random generator. The architecture of the pseudo random generator is based on Linear Feedback Shift Registers (LFSR) constructed with XOR gates. Fig. 7.4 below shows the pseudo random generator with 10 bit input(seed) and output registers that was constructed for this work.

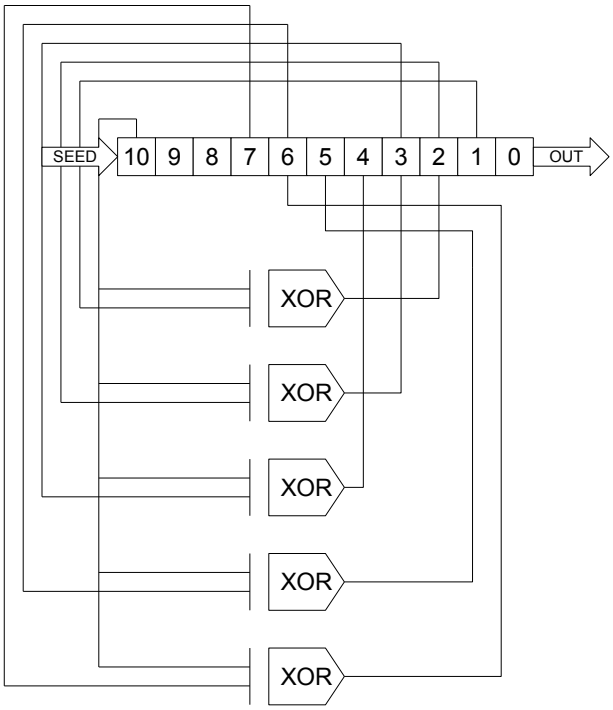


Figure 7.4 The LFSR random noise generator.

7.5.2 Simulation and testing

Once the VHDL code for the pseudo random generator had been written as seen in Appendix B, a test bench was then designed in order to simulate the output of the developed code. A 10 bit seed as well as the 8 kHz clock trigger was used in the simulation. At each clock trigger the output of the pseudo random generator was observed. Fig. 7.5 shows a simulation of the noise component performed on the Xilinx9.2i^{TM}platform.

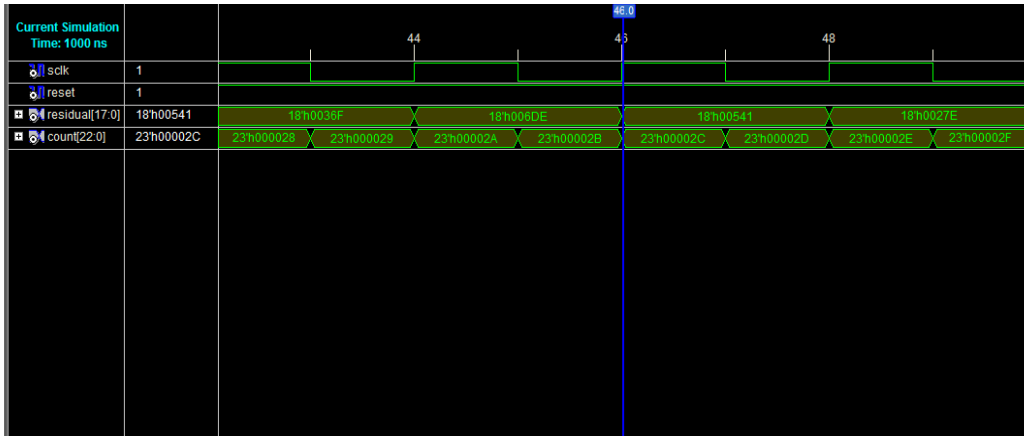


Figure 7.5 A simulated view of the random noise generator in Xilinx.

The following signals can be identified from the simulation:

- Clk: A clock input of 8 kHz.
- Countout: An internal counting sequence signal.
- Residual: The generated residual signal in hexadecimal.

7.6 Modelling the exponent

7.6.1 Algorithm development

The magnitude of the harmonic components in the speech model were built around a constant gradient exponent function. MATLAB^{TM}experiments conducted on the speech model revealed that the gradient of the harmonic, shown as the exponential power $(a_e k)$ in the speech model, did not play a significant role on the quality of the speech produced. This is shown in Table. 7.2, when the exponential gradient is varied the MOS and QOS scores do not change significantly. A lot of computational power could thus be saved by defining $(a_e k)$ as a constant. The experiment involved using the QOS and MOS tests. As illustrated in the table the harmonic gradient only becomes of significance to the quality of the speech speech produced at less than 0.001.

Table 7.2 Effects of varying the harmonic gradient on speech output.

Utterance	Harmonic gradient $\exp(a_e k)$	MOS result	QOS result
hello	0.0005	1.5	70%
hello	0.001	3.0	90%
hello	0.005	4.0	98%
hello	0.010	4.2	98%
hello	0.015	4.1	99%
hello	0.020	4.2	99%
hello	0.030	4.4	100%
hello	0.040	4.5	100%
hello	0.050	4.4	100%
hello	0.060	4.7	99%
hello	0.070	4.3	99%
hello	0.080	4.4	100%

Using a constant gradient in the exponent component allowed for a model design based on value tables. The use of value tables allowed for a reduction on the number of multipliers that would have been used when using the Maclaurin series. Fig. 7.6 illustrates the structure of the designed exponent component. The detailed VHDL code for the designed exponent component is shown in Appendix B.

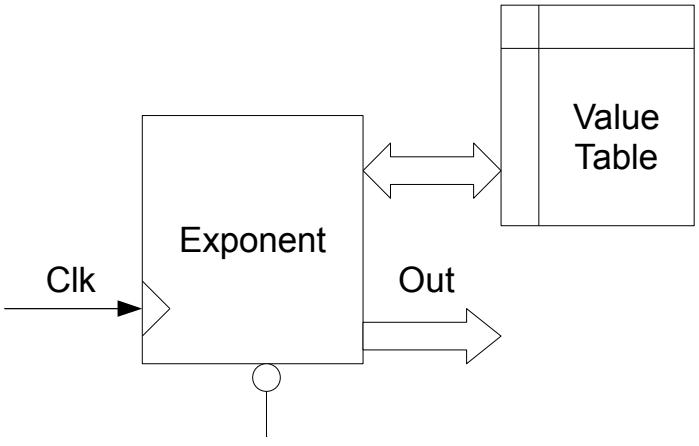


Figure 7.6 The exponential gradient component.

7.6.2 Simulation and testing

To test the exponent component a test bench was built in Xilinx9.2i^{TM}. A unit impulse signal was fed through the testbench as well as a clock signal at 8 kHz. The output of the component was observed at periodic intervals equivalent to phoneme lengths of 150 ms. Results of the simulation tests are shown in Fig. 7.7.

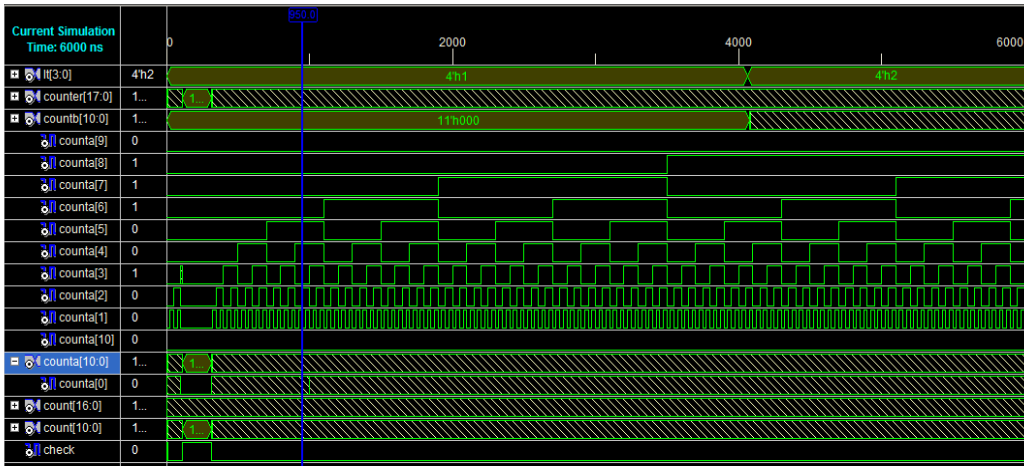


Figure 7.7 A simulated view of the exponent component in Xilinx.

The following signals can be identified from the simulation:

- Lt: A hexadecimal reference to the exponent table.
- Check: The unit impulse trigger input.
- Counta: An internal counting sequence signal.
- Countb: A delayed internal counting sequence signal.
- Counter: The generated exponent signal in hexadecimal.

7.7 Modelling the key-in component

7.7.1 Algorithm development

The Keyin component shown Fig. 7.8 was developed to fetch and decode the model parameters from the parametric corpus. The parametric corpus is in the form of a value table containing all the parameters extracted from the MATLAB^{TM}analysis in the prior section. The input to the Keyin component is an 8 bit input to select each of the phonemes and words from the parametric corpus.

Once the word or phoneme had been selected the data was fed onto the output bus for generation of the residual component and the filter structure. The data transfer on to the output buses was timed in such a manner that parameters would be fed every 150 ms, which is the length of the analysis speech segments.

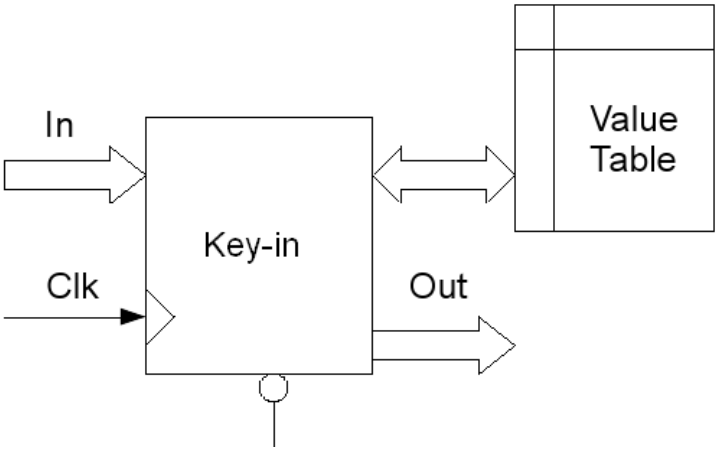


Figure 7.8 The key-in component.

7.7.2 Simulation and testing

A test bench was developed in VHDL to simulate the Keyin component selection on the Xilinx9.2i^{TM}platform. Variable 8 bit inputs were presented at the input of the Keyin component in simulation. For each Keyin input the output from the parametric corpus was observed. Fig. 7.9 shows simulation results.

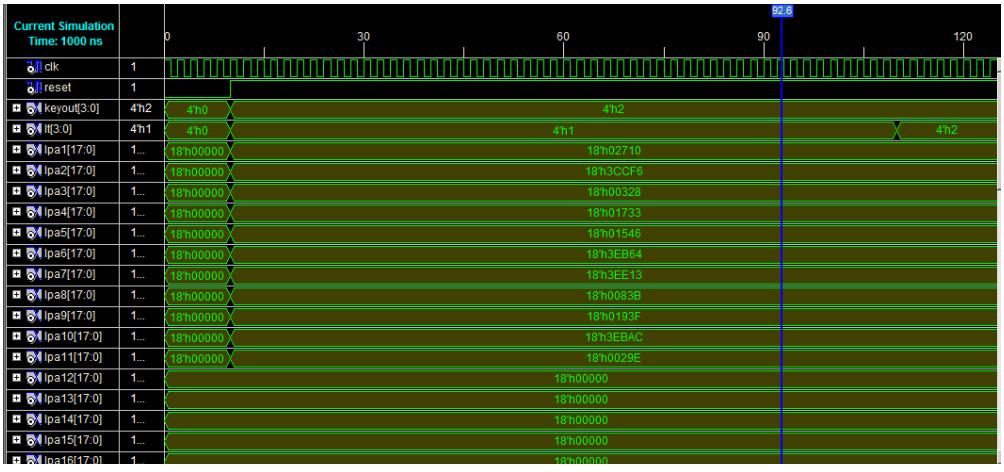


Figure 7.9 A simulated view of the Keyin component in Xilinx.

The following signals can be identified from the simulation:

- Keyout: Variable 8 bit inputs.
- Lpa: Output coeficient parameters from the table.

- Lt: A 150 ms latching counter.

7.8 Modelling the residual adder component

7.8.1 Algorithm development

The residual adder component shown in Fig. 7.10 is a multiplexer and summing circuit that takes up various inputs to produce the residual signal. The inputs of the residual adder are the cosine, exponent and the noise signals. The cosine signal is then summed together with the noise signal and the harmonic exponent signal to produce the residual outputs. To accommodate for the signal windowing done to the residual signal, adjacent speech segments are processed at the same time. This is made possible by the fact that the Keyin component deposits sequentially the corpus parameters.

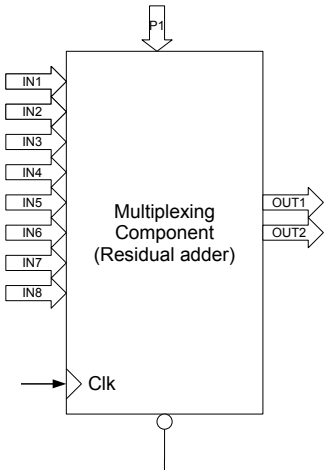


Figure 7.10 The residual adder component.

7.8.2 Simulation and testing

A test bench was developed in VHDL to simulate the residual adder component. The cosine, exponent and noise signals were provided as inputs to the residual adder test bench. The output from the test bench was observed and plotted in the Xilinx9.2i^{TM}platform. Fig. 7.11 shows the signal output plot from the simulated component.

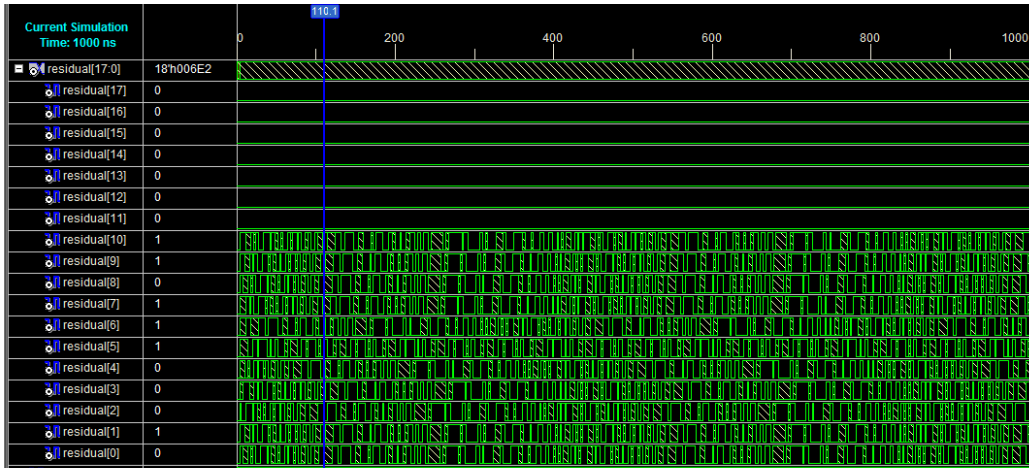


Figure 7.11 A simulated view of the residual adder component in Xilinx.

The following signals can be identified from the simulation:

- Residual: The random noise output from the residual adder.
- Residual[17:0]: The totalised output of the noise component.

7.9 Modelling the cosine generator

7.9.1 Algorithm development

Similar to the exponent component, the cosine component was initially developed using the Maclaurin series. The Maclaurin series expansion for a cosine requires a lot of arithmetic logic units. This presented a problem with the limitation in the number of multipliers available on the FPGA platform. To solve this problem a mathematical table similar to that of the exponent component was used. The mathematical table contained the cosine signal values at a sampling rate of 8 kHz. A total of nine cosine components were constructed representing each of the harmonics in the speech model. Fig. 7.12 shows the block layout of the cosine generator component.

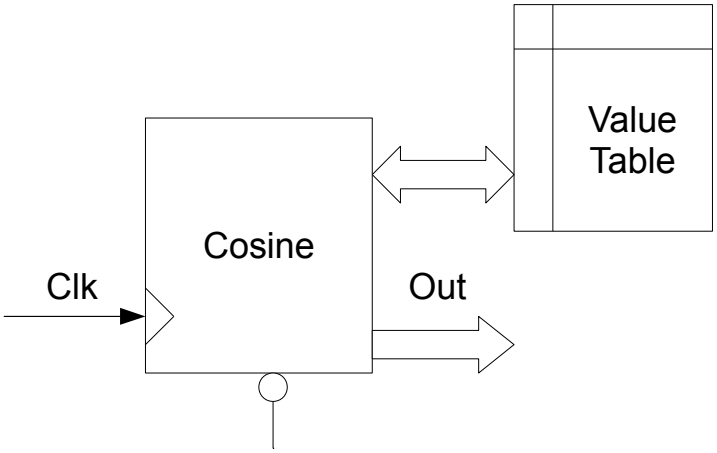


Figure 7.12 The cosine component.

Alternatively the Coordinate Rotation Digital Computer (CORDIC) algorithm [34] could have been used to model the cosine component. The CORDIC algorithm uses addition, subtraction, a look up table and bit shifting to compute trigonometric functions. The CORDIC algorithm was not used in this dissertation because the FPGA provided enough registers to store the entire cosine table.

7.9.2 Simulation and testing

A test bench was developed in VHDL to simulate the cosine component. An external 8 kHz trigger signal from the clock generator component was used to trigger the cosine output based on input values. The output of the simulated cosine component is shown in Fig. 7.13.

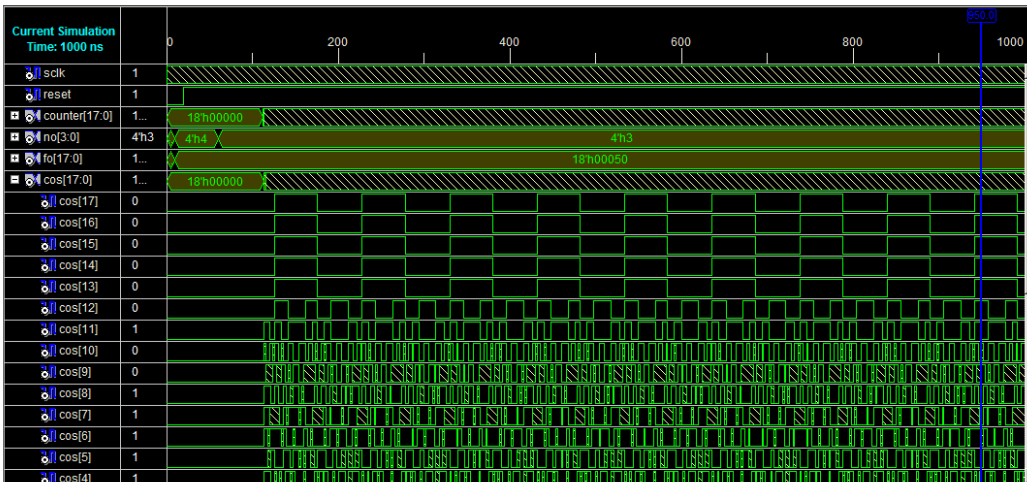


Figure 7.13 A simulated view of the cosine component in Xilinx.

The following signals can be identified from the simulation:

- Sclk: A clock input of 8 kHz.

- No: A reference tag for the cosine table.
- Counter: An internal counting sequence signal.
- Cos: The generated ouput cosine signal.

7.10 IIR filter modelling

7.10.1 Algorithm development

Modelling the IIR filter requires a lot of multipliers and dividers, this proved to be difficult considering the architectural limitations of the FPGA. The process was however made simpler because of the presence of a Xilinx IIR compiler toolbox [30]. The IIR compiler toolbox is similar to the standard MATLAB^{TM}toolbox in that it provides the user with a Graphic User Interface (GUI) to generate automatically the VHDL code for a reconfigurable IIR filter. Fig. 7.14 illustrates the block diagram for one such reconfigurable filter component. The filter component contains an automatically reconfigurable lattice structure based on current filter coefficients. The reconfiguration process takes approximately 25 clock cycles. In order to synchronise the filter with the rest of the circuit the filter clock was triggered at $8X25 = 200$ kHz. This allowed the filter to reconfigure its structure on the fly and produce real-time signal outputs at 8 kHz.

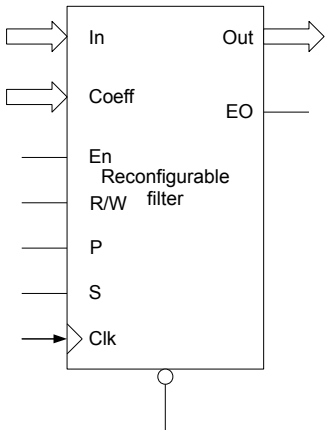


Figure 7.14 Reconfigurable filter block component.

7.10.2 Simulation and testing

A test bench was developed in VHDL to simulate the IIR filter component. The test bench was composed of a 200 kHz signal to simulate the input clock, a unit impulse input signal and the initial filter coefficients. The reconfiguration of the filter's lattice structure was tested after every

150 ms. Each time the IIR filter was reconfigured the unit impulse response was recorded on the Xilinx9.2i^{TM}platform. The results of the simulated test are shown in Fig. 7.15, whilst the detailed VHDL code is shown in Appendix B.

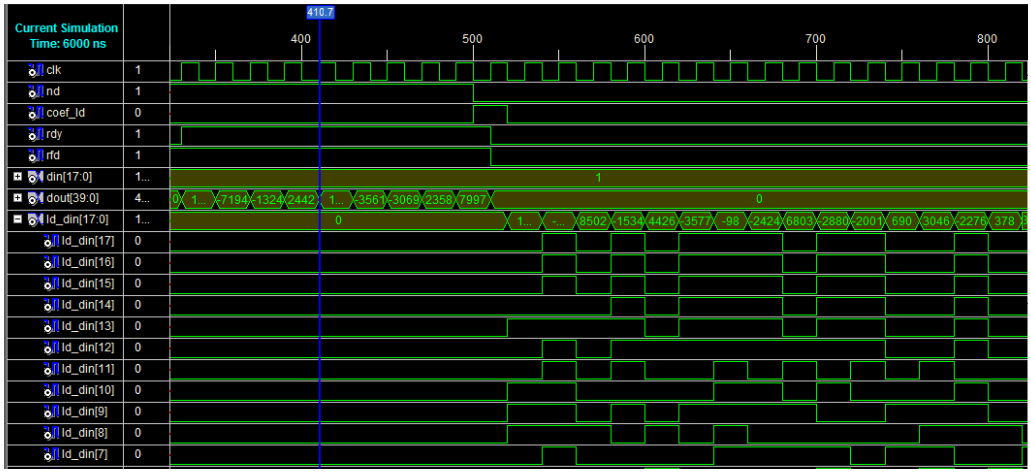


Figure 7.15 A simulated view of the filter component in Xilinx.

The following signals can be identified from the simulation:

- Clk: Input clock signal of 8 kHz.
- Lddin: Are filter parameters.
- Din: Is the input excitation signal.
- Dout: The filtered output signal.

7.11 Modelling the hamming window component

7.11.1 Algorithm development

The Hamming window component was modelled in VHDL using corpus tables similar to the cosine and exponent component. The corpus table contained the magnitude spectra of the Hamming window over a period of 150 ms. An input signal was passed through the Hamming component and multiplexed with the spectra magnitudes to give a Hamming window output. Fig. 7.16 shows a schematic of the Hamming window component.

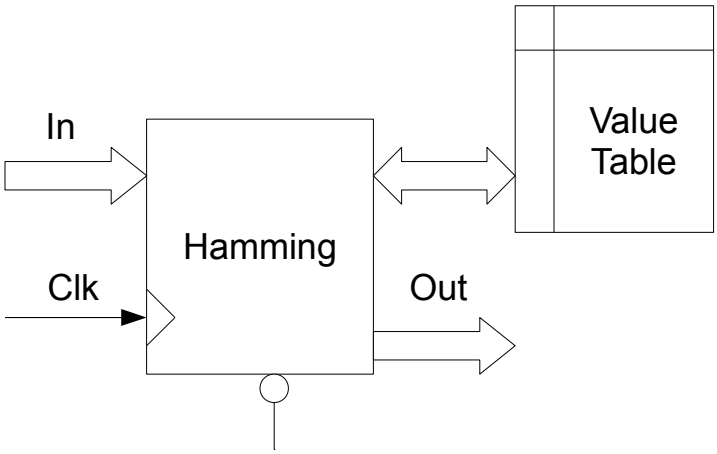


Figure 7.16 Hamming window component.

7.11.2 Simulation and testing

A test bench was developed in VHDL to test the Hamming window component. In order to get a Hamming window output a unit impulse signal was fed at the input. Fig. 7.17 shows the results of the component simulation in the Xilinx9.2i^{TM}simulator.

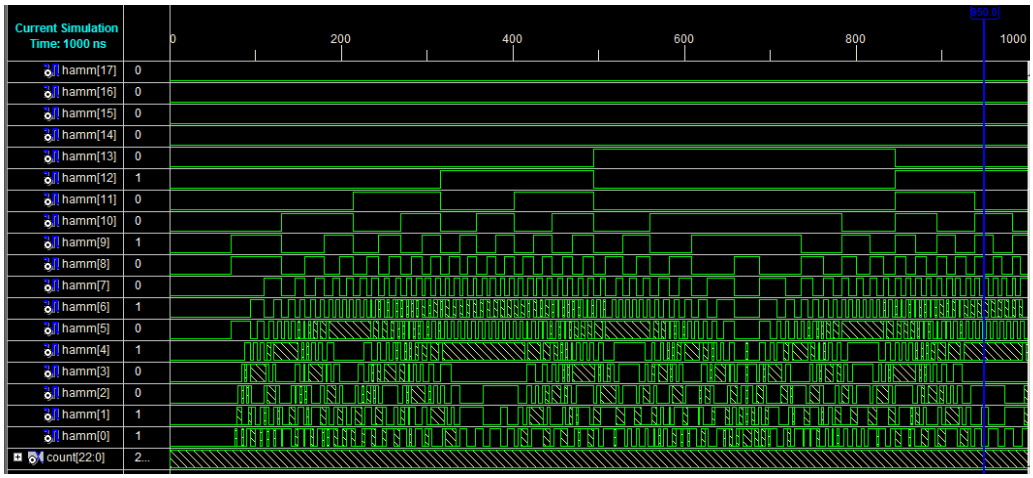


Figure 7.17 A Simulated view of the hamming window component in Xilinx.

The following signals can be identified from the simulation:

- Count: Internal counting sequence.
- Hamm: Generated Hamming window signal.

7.12 Interfacing module components

7.12.1 Algorithm development

The functionality of the speech synthesiser designed is best explained through Table. 7.3. The table illustrates how all the individual components function inside the speech synthesiser.

Table 7.3 Stage interfacing of circuit components.

Interface stage	Active components	Output
1	Frequency generator	8 kHz and 200 kHz signal
1	Keyin component	Circuit initialisation
2	Cosine generator	Cosine signal
2	Random noise generator	Noise signal
2	Exponent generator	Harmonic gradient
3	Residual adder	Residual signal
3	IIR filter	Filter configuration
4	IIR filter	Filtered signal
4	Hamming component	Hamming filter output signal
5	Final adder	Output signal

Firstly the speech synthesiser accepts 5 bit inputs through the Key-in component and initialises all digital subcomponents including the frequency generator. The second cycle begins generation of the excitation signal through the cosine, exponent and random noise components. In the third cycle the output excitation signal is then summed up through the residual adder whilst the reconfiguration of the IIR filter takes place. In the forth cycle the filtering of the excitation signal and the hamming window generation takes place. Finally the hamming signal output is super imposed onto the filter output to produce the digital speech.

7.12.2 Memory utilisation

The objective of the dissertation was to develop a real-time speech synthesis system utilising a small memory footprint. The designed speech synthesiser was compiled in the Xilinx9.2i^{TM}platform with a target memory of 200000 system gates. A summary of the system gates and Look Up Tables (LUT) used by each of the components is presented in Table. 7.4.

Table 7.4 Logic utilisation on the FPGA chip.

Component	Number of slices	Number of system gates	Equivalent LUTs
Frequency generator	34	712	68
Keyin component	512	12080	1018
Cosine generator	2000	41500	4000
Random noise generator	24	480	48
Exponent generator	1028	24600	2408
Residual adder	7	150	14
IIR filter	521	11040	1047
Hamming component	1400	30000	2800
Final adder	64	1640	128
Interfacing	105	2400	212
Total	5695	124602	11743

7.12.3 Simulation and testing

To test the functionality of the whole system, a test bench was developed for all simulated user inputs. The output wave generated from circuit output was then recorded as a wav file. The recorded wave file was played and analysed in Audacity™ [35]. Fig. 7.18 shows a typical time domain signal of the recorded wave file in Audacity™. Listening tests similar to those carried out in section 6.5 were conducted on the recorded speech segments, the results of these tests are elaborated in section 7.13.1.

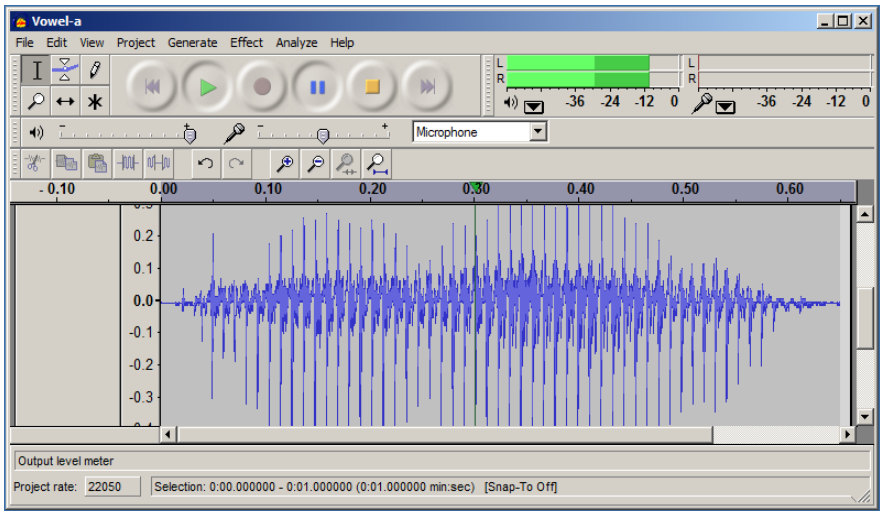


Figure 7.18 Recorded wave analysis of the phoneme /a/.

7.12.4 Output analysis

The encoded output wave file was also analysed in MATLAB^{TM}and compared against the original speech segment in both frequency and time domains. Fig. 7.19 shows the encoded wave recording for the vowel /a/ signal against the original waveform in the time domain. The synthesised signal in red is set 2 ms out of phase with the original signal in blue and shown in red to enhance visibility. Fig. 7.20 shows the VHDL synthesised recording in the frequency domain.

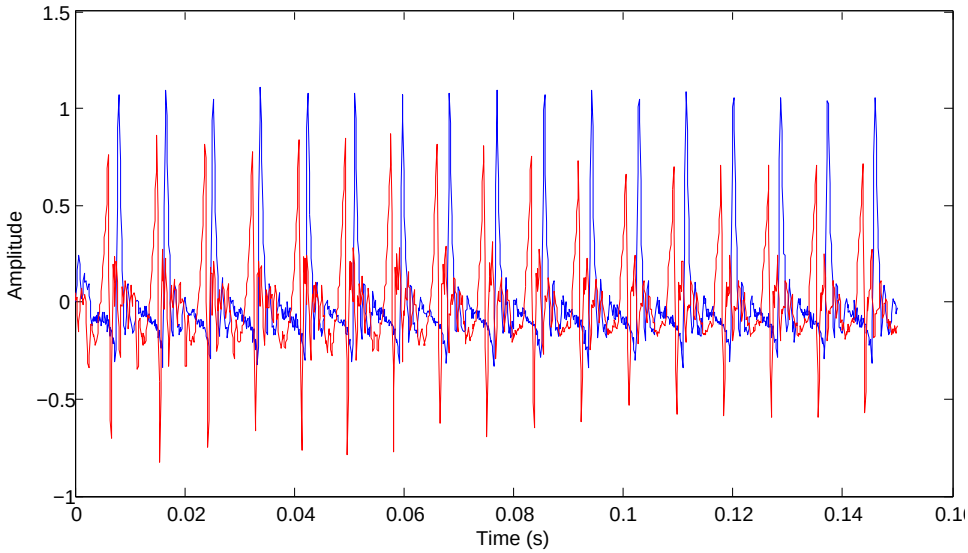


Figure 7.19 A time domain comparison of the VHDL systhesised waveform (red) vs the original waveform (blue). The synthesised signal was shifted 2 ms out of phase to enhance visibility.

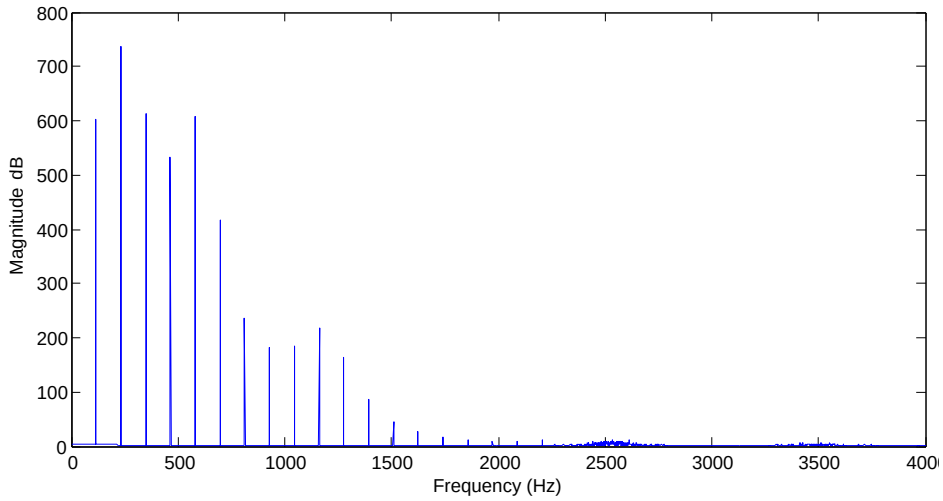


Figure 7.20 The VHDL synthesised vowel /a/ signal in the frequency domain.

7.12.5 Spectrogram analysis

The characteristics of the synthesised speech were determined by analysing the speech signal in both the time and frequency domains. It was discovered during the experimentation process, that the time and frequency domain signals do not clearly depict the quality and audibility of the uttered speech. In order to visualise the quality of speech produced a third domain utilising both frequency and time was used i.e. spectrogram. A spectrogram was used to distinguish the speech energy losses incurred in developing the speech synthesiser in both MATLAB^{TM}and VHDL. Fig. 7.21 shows the spectrogram for the VHDL synthesised vowel /a/ signal in comparison to Fig. 7.22 the spectrogram for the MATLAB^{TM}synthesised vowel /a/ signal.

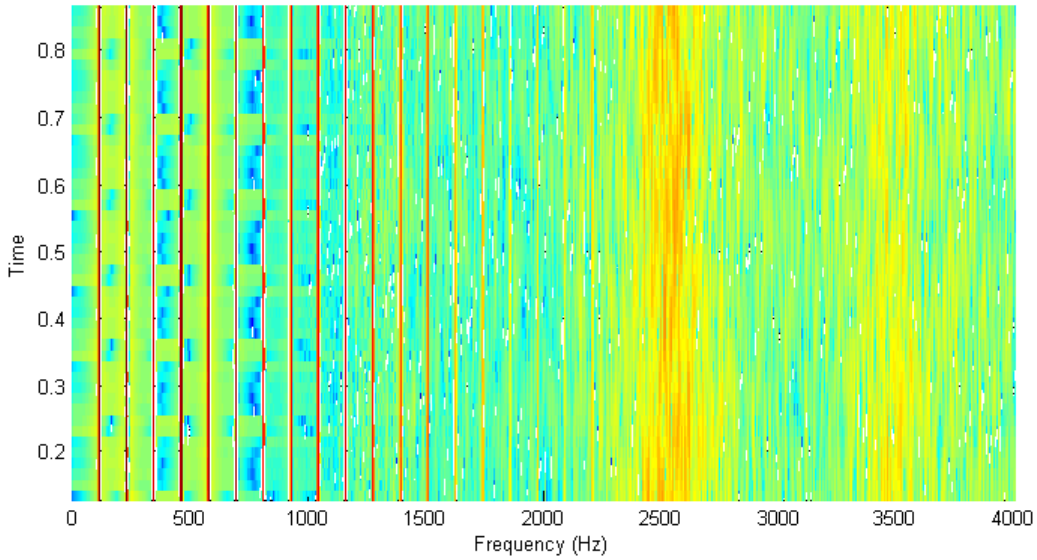


Figure 7.21 A spectrogram analysis of the VHDL synthesised vowel /a/.

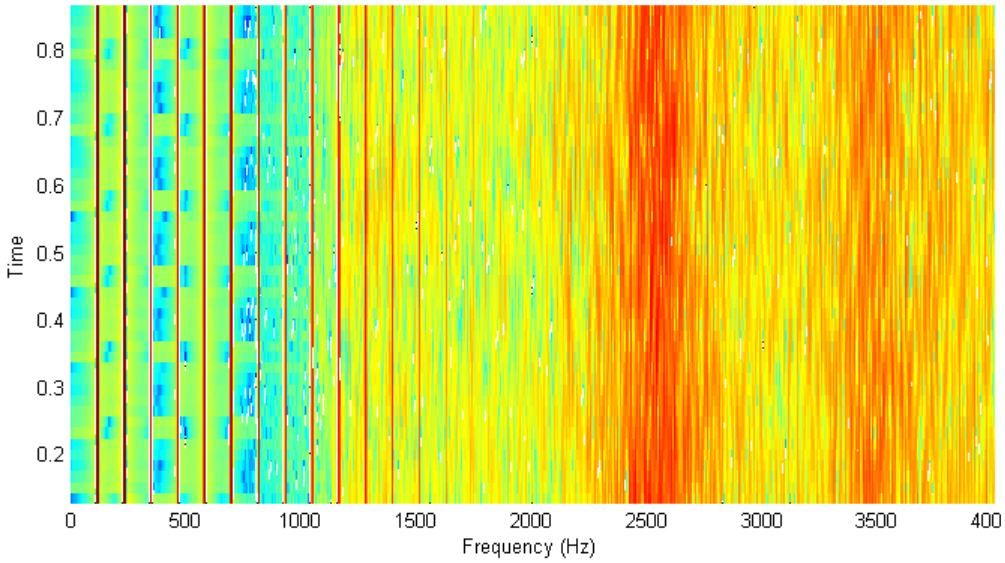


Figure 7.22 A spectrogram analysis of the MATLAB^{TM}synthesised vowel /a/.

The figures show that the VHDL synthesised signal has more voicing in the lower frequency and a lower noise component in the higher frequency range compared to the MATLAB^{TM}synthesised signal. This is depicted by the dark lines in the lower frequencies and the lighter lines in the high frequency region. This phenomenon was attributed to the rounding effect of using integer numbers in modelling the system in VHDL. The yellow lines depict higher frequencies mostly composed of white noise.

7.13 VHDL based listening tests

Listening tests were performed on the VHDL synthesised waveforms using words. A test sample of 20 native South African English speakers from the University of Witwatersrand was used. All 20 speakers were eloquent in the English language. MOS and transcription tests described in section. 6.5.1 and 6.5.2 were performed on the VHDL synthesised speech. In a similar manner to the MATLAB^{TM}tests, all listeners were briefed on the criteria used on MOS and QOS tests.

7.13.1 Mean Opinion Score tests

Each listener from the test sample was asked to give a score from 1 to 5 on the quality of the speech recordings played to him or her. Table. 7.5 shows the mean opinion score MOS score for VHDL synthesised words. The score is an average from the listeners interpretation of the VHDL sysnthesied speech in comparison to the original speech segment.

Table 7.5 VHDL based mean opinion scores (words).

Word	Class	Original recording (MOS)	Synthesised recording (MOS)
hello	vowel	4.4	3.6
hat	vowel	4.6	3.8
too	vowel	4.4	3.5
door	vowel	4.8	4.0
shop	fricative	4.2	3.7
that	fricative	4.5	4.1
dig	plossive	4.4	4.1
pit	plossive	4.2	3.7

The reason for using synthesised words rather than the phonemes as earlier stated was the difficulty listeners faced interpreting phonemes. The same experiment was, however, performed with the VHDL synthesised phonemes. The results of this experiment are shown in the Table. 7.6 below.

Table 7.6 VHDL based mean opinion scores (phonemes).

Word	Class	Original recording (MOS)	Synthesised recording (MOS)
/a/	vowel	4.5	2.8
/e/	vowel	4.1	3.5
/i/	vowel	3.9	3.2
/o/	vowel	3.7	2.7
/s/	fricative	4.3	3.4
/h/	fricative	4.2	4.0
/d/	plossive	4.4	4.1
/p/	plossive	3.9	2.5

7.13.2 Transcription tests

Transcription tests QOS were carried out on VHDL synthesised speech segments using the same test sample used for the MOS test. The results of the transcription tests are shown in Table. 7.7

Table 7.7 VHDL based transcription scores (words).

Word	Class	Original recording (QOS)	Synthesised recording (QOS)
hello	vowel	100%	99%
hat	vowel	100%	99%
too	vowel	99%	97%
door	vowel	99%	93%
shop	fricative	99%	97%
that	fricative	100%	99%
dig	plossive	99%	97%
pit	plossive	99%	99%

The same transcription tests were carried out in MATLAB^{TM}as well as VHDL using synthesised phonemes. The results of the transcription tests are shown in Table. 7.8.

Table 7.8 VHDL based transcription scores (phonemes).

Word	Class	Original recording (QOS)	Synthesised recording (QOS)
/a/	vowel	92%	92%
/e/	vowel	90%	96%
/i/	vowel	84%	82%
/o/	vowel	96%	91%
/s/	fricative	90%	97%
/h/	fricative	98%	85%
/d/	plossive	100%	91%
/p/	plossive	99%	70%

7.13.3 Discussion of results

The results in Table. 7.5 and Table. 7.7 show that VHDL synthesised speech achieved high scores on the MOS and Transcription tests respectively. The slight discrepancy in the results can be attributed to the fact that the initial listening tests were already performed on MATLAB^{TM}synthesised speech, therefore the person hearing the sound again would be able to interpret it easily. This fact

is verified by the improved scores on both the MOS and QOS tests for the original speech segment. Results of the spectrogram analysis revealed that the synthesised speech posessed high energy formants in the lower frequency band than the original speech segment. The quality of the speech produced was clearly audible as depicted by the MOS and QOS scores. The VHDL simulations enabled the development of a FPGA based hardware platform discussed in the next chapter. The next chapter answers the objective of the dissertation by presenting the hardware based embedded speech synthesiser.

Chapter 8

Hardware development

8.1 Hardware implementation

In order to seamlessly implement the speech synthesiser in hardware a demonstration board based on the Xilinx XC3S1600E FPGA was used. The development board provided a plug and play option for external interfaces and digital input/output pins. Before the VHDL code was downloaded onto the hardware a selection of the input/output pins was performed in the compiler based on the demonstration board schematic. Once the input/output pins had been defined, the VHDL code was compiled and synthesised to produce a Joint Electron Device Engineering Council (JEDEC) file that could be programmed on the FPGA.

8.2 External hardware

To fully implement the functional speech synthesiser external hardware had to be added on the demonstration for user interface. The external hardware was composed of the following items:

- An $8X8$ matrix keyboard connected to a matrix decoder to provide 8 bit key-in inputs to the FPGA: HITACHI;
- A digital audio amplifier connected to the output of the FPGA: LM49370;
- A loud speaker connected to the output of the audio amplifier: LM4931;

Fig. 8.1 shows a schematic of the speech synthesiser hardware.

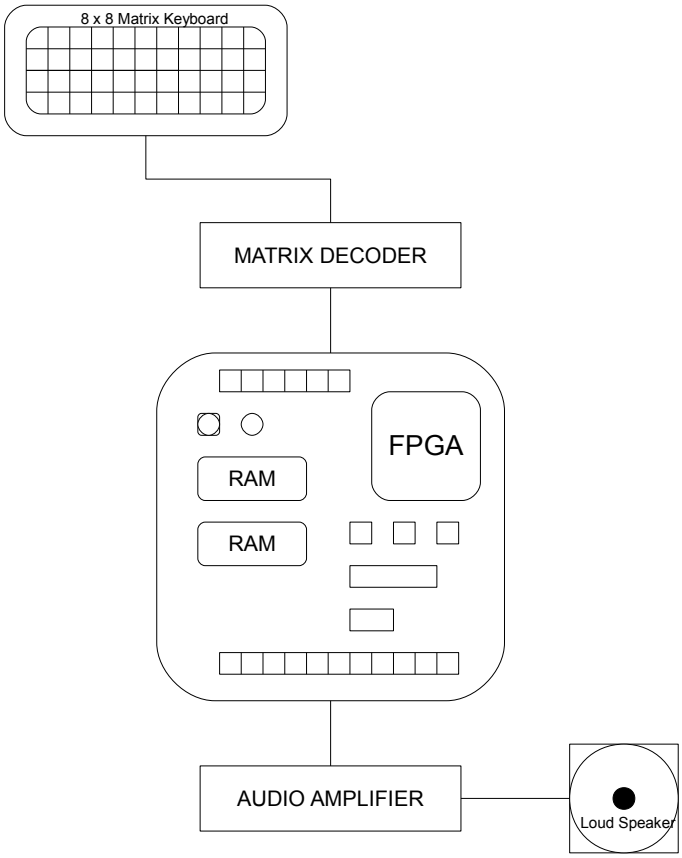


Figure 8.1 A hardware schematic of the speech synthesiser.

8.3 Hardware tests

Once the hardware had been assembled listening tests were conducted on the uttered speech output. The speech was triggered by pressing a key on the matrix keyboard. Listening tests, namely MOS and QOS were performed on the speech output. The test sample used earlier was unavailable at the time the hardware platform was built. A test sample composed of 20 students from the University of the Witwatersrand, Johannesburg all native English speakers was used. In order to get a good comparison of the scores listening tests were also carried out on the original speech segments. The results of the listening tests are shown in Table. 8.2 and Table. 8.1, for both phonemes and words respectively.

Table 8.1 Phoneme listening test results for the built speech synthesiser.

Key	Sound	Original QOS	Synthesised QOS	Original MOS	Synthesised MOS
1	/I/	84%	82%	3.2	3.0
2	/e/	98%	98%	4.4	4.2

Continued on next page

Table 8.1 – continued from previous page

Key	Sound	Original QOS	Synthesised QOS	Original MOS	Synthesised MOS
3	/æ/	86%	84%	3.1	3.2
4	/ɒ/	80%	80%	2.8	2.6
5	/ʌ/	85%	83%	3.5	3.5
6	/ʊ/	84%	84%	3.6	3.2
7	/ə/	75%	74%	2.1	2.2
8	/i:/	92%	90%	4.1	3.8
9	/a:/	99%	99%	4.6	4.5
10	/o/	100%	99%	4.7	4.5
11	/ɜ:/	90%	90%	3.5	3.4
12	/u:/	100%	99%	4.0	4.0
13	/ei/	92%	91%	4.0	3.9
14	/ai/	95%	94%	4.2	4.2
15	/oi/	100%	100%	4.4	4.4
16	/iə/	93%	92%	4.0	4.0
17	/eə/	96%	96%	4.4	4.2
18	/ʊə/	86%	88%	3.8	4.0
19	/əʊ/	90%	89%	3.8	3.6
20	/aʊ/	99%	100%	4.4	4.4
21	/p/	98%	97%	3.8	3.8
22	/b/	98%	98%	4.1	4.0
23	/t/	99%	95%	4.0	3.9
24	/d/	92%	90%	3.8	3.6
25	/k/	94%	94%	3.9	3.5
26	/g/	99%	94%	4.2	4.0
27	/f/	80%	76%	3.1	2.8
28	/v/	88%	88%	3.6	3.66
29	/θ/	90%	90%	3.8	3.5
30	/s/	98%	99%	4.2	4.1

Continued on next page

Table 8.1 – continued from previous page

Key	Sound	Original QOS	Synthesised QOS	Original MOS	Synthesised MOS
31	/z/	90%	90%	3.5	3.4
32	/m/	99%	99%	4.4	4.4
33	/n/	100%	100%	4.6	4.6
34	/ŋ/	84%	82%	3.2	3.2
35	/l/	86%	82%	3.4	3.2
36	/r/	99%	99%	4.4	4.4
37	/w/	82%	82%	3.4	3.4
38	/h/	80%	82%	3.2	3.2
39	/j/	94%	90%	3.8	3.8
40	/ʃ/	92%	92%	4.0	4.0
41	/ð/	74%	74%	2.4	2.4
42	/ʒ/	76%	78%	2.4	2.6
43	/tʃ/	80%	78%	3.0	3.0
44	/dʒ/	82%	84%	3.4	3.4

Table 8.2 Word listening test results for the built speech synthesiser.

Key	Sound	Original QOS	Synthesised QOS	Original MOS	Synthesised MOS
45	hello	100%	100%	4.6	4.6
46	hat	100%	99%	4.6	4.4
47	too	100%	98%	4.8	4.2
48	door	100%	100%	4.8	4.8
49	shop	100%	96%	4.8	4.4
50	that	100%	100%	4.8	4.8
51	dig	100%	100%	4.8	4.8
52	pit	100%	99%	4.6	4.6

8.4 Discussion of results

The results of the listening tests, show that speech produced from the hardware performed well on the QOS and MOS scales with scores of 98% and 4.8% respectively. These scores were significantly higher compared to the MOS and QOS scores obtained from the VHDL and MATLAB^{TM}platforms. This can be attributed to the fact that hardware platform does not contain external filters, which are present on most computer simulated platforms as sound card drivers. A perfect transcription score of 100% was recorded for words like hello, hat, that and door. Equally impressive results were recorded on the MOS test, with scores of 4.6 recorded for the word hello. The results of the phoneme listening tests were perfect as expected, although descrepencies were recorded in some instances as shown in Table. 8.1. This covered the scope of work defined in this dissertation. In the last chapter of this document suggestions for future work including tests with complete senctences is suggested.

Chapter 9

Conclusion and future work

9.1 Conclusion

This dissertation described and explained the development of an FPGA based phonetic speech synthesiser. The work analysed the problems with modern day embedded speech synthesisers. These problems were portability, memory usage, quality of uttered speech and the processing power requirements. The objective of this work therefore, was to develop a portable high quality speech synthesis device utilising a small memory footprint of at least 200000 system gates.

An extensive literature review of the models used in speech synthesisers was performed. It was then proposed to use rule based synthesis models in developing the embedded speech synthesiser. These models included linear prediction LP methods, the harmonic plus noise models HNM, the log magnitude filter LMA, forward backward least square spectral estimate FBLS and the auto regressive with exogenous input filter ARX. The final solution proposed was to use the linear prediction model in conjunction with the HNM model.

Tests and simulations were performed in MATLAB^{TM}and VHDL, respectively, for speech segments that were synthesised using the suggested model. The tests performed included MOS and QOS listening tests using a sample of 20 native South African English speaking students. It was discovered that there were inconsistencies with results obtained when performing listening test on synthesised phonemes. In order to counter the inconsistencies 8 words were also synthesised using the model. Good scores of 98% and 4.1 were achieved on the QOS and MOS test, respectively, for both VHDL and MATLAB simulations.

The algorithm used in the VHDL tests was compiled and written to a Xilinx XC3S1600E FPGA device. To make a fully functional speech synthesiser external devices were added to the FPGA. These included a matrix keyboard, the matrix decoder, the audio amplifier and the loud speaker. Once the hardware setup was completed listening tests were performed on speech uttered from the device. The results of the listening tests were significantly higher than those of the VHDL and MATLAB^{TM}simulations including scores as high as 99% and 4.5 on the QOS and MOS, respectively.

The system developed performed real time speech synthesis with a memory utilisation of 125000 system gates on the FPGA. In this dissertation a high quality embedded speech synthesiser with a small memory footprint and real-time speech processing was designed and developed. This met the objective of the dissertation adequately though more work would need to be carried out as explained in the coming section.

9.2 Improvements and future work

The results of the mean opinion score MOS and transcription tests QOS were not perfect in both simulation and hardware. The imperfection emanated from the fact that phonemes were used as the bases of the speech synthesiser. On the other hand sentences would have presented a better method of obtaining accurate MOS and QOS scores on the synthesis model. Using sentences would entail developing a complete TTS system. This was not covered in the scope of work and is suggested here as future work.

The FPGA based speech synthesiser did not address the concept of phonetic interpolation. Phonetic interpolation addresses the aspect of joining speech segments smoothly which enables the production of intelligible speech sentences. This aspect was not included in the initial scope of work. It is suggested that as future work the concept of interpolation must be added to the speech synthesis model.

In conclusion this research provided a platform for further research and future work in the field of embedded speech synthesis.

9.3 Contributions of the research

Appendix C presents the contributions from this dissertation in the form of paper publications in the field of speech synthesis. The paper entitled *Optimised source signal modelling for linear prediction speech synthesis* was published at the Pattern Recognition Association conference of South Africa in 2007. The paper reviewed ways of modelling the residual signal in linear prediction. These methods included the Rosenberg Klatt, unit impulse, triangular pulse and the Harmonic plus noise model.

The second contribution was a paper published at the $21st$ Conference on Collaborative Research for Technological Development held in Kampala, Uganda. The publication talked about the advancement in assistive speech technology in sub-Saharan Africa. These advancements included using embedded speech tools. The publication also gave a platform to address the practicality of using the FPGA in embedded speech synthesis.

The final publication in the list of contributions was another paper published as Work in Progress at the Pattern Recognition Association of South Africa conference held in Cape Town 2008. The publication was entitled an *Optimised parametric speech synthesis model based on Linear Prediction and the Harmonic plus Noise Model*. The paper illustrated a new speech synthesis technique utilising Linear Prediction and the Harmonic plus noise model. The model was compared to traditional speech synthesis models and performed well on the MOS and QOS tests.

References

[1] R. Hoffmann, O. Jokisch, G. Strecha, D. Hirschfeld, "Advances in Speech Technology for Embedded Systems," *Conference and Workshop on Assistive Technologies for Vision and Hearing Impairment CVHI*, Granada Spain, 28 June - 2 July 2004.

[2] B. Lacquet, M. Shuma-Iwisi, A. Mamombe, "Advancements in assistive speech technology for sub Saharan Africa," *Conference on Collaborative Research for Technological Development*, pp. 131-136, Kampala Uganda, 17th - 21st December 2007.

[3] F.A. Everest, "Master Handbook of Acoustics", Fourth Edition, McGraw-Hill, 2001.

[4] F.J. Owens, "Signal Processing of Speech", The Macmillan Press Ltd, 1993.

[5] M.S. LadyofHats, "A complete, schematic view of the human respiratory system", Public domain listing, Wikimedia commons, July 2007.

[6] M. Rothenberg, "A New Inverse-Filtering Technique for Deriving the Glottal Airflow Waveform During Voicing", Journal of the Acoustical Society of America 53, pp. 1632-1645, 1973.

[7] I.H. Witten, "Principles of Computer Speech", Academic Press, 1982.

[8] A. Davies, "The Phoneme Test: Should All Teachers Pass It?," *The Journal of the Dyslexia Institute Guild*, Volume 11, Number 4, pp. 9-12, Summer 2000.

[9] J.L. Flanagan, "Speech Analysis and Perception", Springer-Verlag, Berlin, 2nd edition, 1965.

[10] T. Dutoit, *A Short introduction to text-to-speech synthesis*, Published electronically, 1999: *http://tcts.fpms.ac.be/synthesis/introtts _old.html* [last accessed 2008-05-10].

[11] J. Schroeter, "Text-to-Speech (TTS) Synthesis", *Chapter 16: Circuits, Signals, Speech and Image Processing*, CRC Press, 2006.

[12] J. Gros, A. Mihelic, N. Paveic, M. Ganec, S. Gruden, "Slovenian Text-to-Speech Synthesis for Speech User Interfaces," *In Proceedings of the Third World Enformatika Conference, WEC'05*, pp. 216-220, Istanbul Turkey, 27-29 April 2005.

[13] H. Hain, J. Racky, T. Volk, "The Papageno TTS System," *In Proceedings of the TC-STAR Workshop on Speech-to-Speech Translation*, pp. 193-198 Barcelona Spain, 19-21 June 2006.

[14] J. Makhoul, "Linear prediction: A tutorial review," *In Proceedings of the IEEE, Vol 63 Issue 4*, pp. 561-580, April 1975.

[15] Y. Stylianou, "On the implementation of the harmonic plus noise model for concatenative speech synthesis," *In Proceedings. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP Volume 2, Issue 2000*, pp. 11957 - 11960 , Istanbul Turkey, 9 June 2000.

[16] Y. Stylianou, "Applying the harmonic plus noise model in concatenative speech synthesis," *IEEE Transactions on speech and audio processing, Volume 9, Issue 1*, pp. 21 - 29, January 2001.

[17] G. Klompje, T.R. Niesler, "A parametric monophone speech synthesis system", *In proceedings of the seventeenth annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, Parys South Africa, November 2006.

[18] R. Wang, Q. Liu, D. Tang, "A new Chinese text-to-speech system with high naturalness," *In Proceedings of the 4th International Conference on Spoken Language Processing ICSLP 96, Volume 3*, pp. 1441-1444, Philadelphia USA, 3-6 Oct 1996.

[19] G.E.P. Box, G.M. Jenkins, "Time Series Analysis: Forecasting and Control", San Francisco:Holden Day, 1976.

[20] W. Zhu, H. Kasuya, "A New Speech Synthesis System Based On The Arx Speech Production Model," *In Proceedings of the 4th International Conference on Spoken Language Processing ICSLP 96, Volume 3*, pp. 1413-1416, Philadelphia USA, 3-6 Oct 1996.

[21] S.M. Bozic, *Digital and Kalman Filtering*, Edward Arnald Publications, 1979.

[22] N. Kalouptsidis, J. Theodorides, "Fast adaptive least-squares algorithms for power spectral estimation," *IEEE Transactions on Acoustics, Speech and Signal Processing, Volume 35*, pp. 661-670, 1987.

[23] L. Tomokiyo, K. Peterson, A. Black, K. Lenzo, Intelligibility of Machine Translation Output in Speech Synthesis, In proceedings of the Interspeech ICSLP (2006), pp. 2434-2437, Pittsburgh, PA, September 2006.

[24] S. Roa, M. Bennewitz, S. Behnke, "Fundamental frequency estimation based on pitch-scaled harmonic filtering," *In Proceedings. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 4*, pp. 397-400, Honolulu Hawaii, 15-20 April 2007.

[25] A. Mamombe, B. Lacquet, "Optimised source signal modelling for Linear predictive speech synthesis," *In proceedings of the 18th international symposium of the Pattern Recognition Association of South Africa PRASA 2007*, pp. 93-98, Pietermarizburg, South Africa, Nov 2007.

[26] B. Lacquet, M. Shuma-Iwisi, A. Mamombe, "An optimised parametric speech synthesis model based on linear prediction (LP) and the Harmonic plus Noise Model (HNM)," *In proceedings of the 19th international symposium of the Pattern Recognition Association of South Africa PRASA 2008*, pp. 176-177, Cape Town South Africa, Nov 2008.

[27] K. Levenberg, "A Method for the Solution of Certain Problems in Least Squares," Quart Appl Math, Volume 2, pp. 164-168, 1944.

[28] Handbook of the International Phonetic Association, "A Guide to the Use of the International Phonetic Alphabet," Cambridge University Press, 1999.

[29] M.C. Ndinechi, N. Onwuchekwa, G.A. Chukwudebe"Algorithm for Applying Decimator Structures in Digital Signal Processing Systems for Energy Conservation," *International Journal of Soft Computing Year, Vol 4, Issue 6*, pp. 236-242, 2009.

[30] Xilinx. Inc, "CPLD and FPGA soultions from Xilinx Inc", *http://www.xilinx.com* [last accessed 2009-08-09].

[31] T. George, T. Finney, L. Ross, "Calculus and Analytic Geometry", Ninth Edition, Addison Wesley, 1996.

[32] G.E. Box, M.E. Muller, "A Note on the Generation of Random Normal Deviates," *The Annals of Mathematical Statistics, Volume 29, Issue 2*, pp 610-611, 1958.

[33] E. Weisstein, "Gaussian Function : MathWorld A Wolfram Web Resource", *http://mathworld.wolfram.com/GaussianFunction.html* [last accessed 2009-08-09].

[34] J.E. Volder, "The Birth of CORDIC", J. VLSI Signal Processing 25, pp. 101-102, 2000.

[35] Audacity. Inc, "Audacity a digital audio editor and recording application", *http://www.audacity.com* [last accessed 2009-08-09].

[36] T. Lindeberg, "Scale-space for discrete signals", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 3, pp. 234-254, March 1990.

[37] S. Redl, M. Weber, M. Oliphant, "An Introduction to GSM", Artech House, March 1995.

Appendix A

Parametric Corpus

Table A.1 Parametric speech corpus for HNM and LP model vowel /a/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/a/	1.0000	1200	124	0.0016	0.04	0.0075	$2\pi/10$
	-0.3247						
	-0.2776						
	-0.4670						
	-0.3810						
	1.0218						
	0.3900						
	-0.1100						
	-0.0441						
	-0.4387						
	0.1040						
/a1/	1.0000	1280	126	0.0017	0.05	0.0075	$2\pi/11$
	-0.2000						
	-0.3602						
	-0.6012						
	-0.3720						
	0.9900						
	0.3350						

Continued on next page

Table A.1 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.0200						
	-0.0850						
	-0.3207						
	0.0806						
/a2/	1.0000	1280	124	0.0017	0.04	0.0070	$2\pi/10$
	-0.1770						
	-0.4300						
	-0.6105						
	-0.3116						
	0.9400						
	0.3900						
	0.0001						
	-0.1022						
	-0.2602						
	0.0600						
/a3/	1.0000	1280	122	0.0017	0.04	0.0070	$2\pi/10$
	-0.1779						
	-0.4327						
	-0.6200						
	-0.3203						
	0.9500						
	0.3900						
	0.0043						
	-0.1320						
	-0.2830						
	0.0610						

Table A.2 Parametric speech corpus for HNM and LP model vowel /e/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/e/	1.0000	1280	127	0.0016	0.04	0.0070	$2\pi/10$
	-0.2247						
	-0.4776						
	-0.5450						
	-0.3800						
	1.0238						
	0.3970						
	-0.1025						
	-0.0641						
	-0.3787						
	0.1443						
/e1/	1.0000	1390	126	0.0017	0.05	0.0075	$2\pi/11$
	-0.2318						
	-0.3811						
	-0.5883						
	-0.3896						
	0.9807						
	0.3451						
	0.0219						
	-0.0908						
	-0.3307						
	0.0906						
/e2/	1.0000	1280	125	0.0018	0.04	0.0075	$2\pi/10$
	-0.1889						
	-0.4368						
	-0.6285						
	-0.3286						

Continued on next page

Table A.2 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.9680						
	0.3942						
	0.0039						
	-0.1422						
	-0.2864						
	0.0687						
/e3/	1.0000	1240	125	0.0018	0.04	0.0075	$2\pi/10$
	-0.1889						
	-0.4368						
	-0.6285						
	-0.3286						
	0.9680						
	0.3942						
	0.0039						
	-0.1422						
	-0.2864						
	0.0687						

Table A.3 Parametric speech corpus for HNM and LP model vowel /i/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/i/	1.0000	1280	121	0.0016	0.04	0.0070	$2\pi/10$
	-0.2257						
	-0.4786						
	-0.5460						
	-0.3810						
	1.0218						

Continued on next page

Table A.3 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.4970						
	-0.0025						
	-0.0041						
	-0.4587						
	0.1553						
/i1/	1.0000	1300	124	0.0017	0.05	0.0075	$2\pi/11$
	-0.2001						
	-0.3710						
	-0.5901						
	-0.4213						
	0.9701						
	0.3000						
	0.0001						
	-0.1008						
	-0.3201						
	0.0700						
/i2/	1.0000	1340	122	0.0018	0.04	0.0075	$2\pi/10$
	-0.1680						
	-0.5300						
	-0.6705						
	-0.3100						
	0.9400						
	0.3800						
	0.0040						
	-0.1320						
	-0.2860						
	0.0600						

Table A.4 Parametric speech corpus for HNM and LP model vowel /o/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/o/	1.0000	910	132	0.0015	0.04	0.0040	$2\pi/8$
	-0.7954						
	-0.7118						
	-0.0049						
	1.1348						
	0.3880						
	-0.6468						
	-0.3763						
	0.0766						
	0.3567						
	-0.0263						
/o1/	1.0000	1200	132	0.0016	0.04	0.0040	$2\pi/9$
	-0.8299						
	-0.6627						
	0.0440						
	1.0191						
	0.3988						
	-0.5774						
	-0.3736						
	0.0401						
	0.3446						
	-0.0050						
/o2/	1.0000	1195	132	0.0016	0.04	0.0040	$2\pi/10$
	-0.8601						
	-0.7301						
	0.1266						
	1.1385						

Continued on next page

Table A.4 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.3355						
	-0.7407						
	-0.3848						
	0.1880						
	0.3590						
	-0.0762						
/o3/	1.0000	1200	125	0.0018	0.04	0.0075	$2\pi/10$
	-0.8312						
	-0.7765						
	0.1592						
	1.0883						
	0.3799						
	-0.7085						
	-0.4107						
	0.1813						
	0.3274						
	-0.0388						

Table A.5 Parametric speech corpus for HNM and LP model plossive /d/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/d/	1.0000	1100	105	0.0010	0.04	0.0040	$2\pi/10$
	-0.3634						
	-0.0474						
	-0.0640						
	-0.1714						
	-0.1615						

Continued on next page

Table A.5 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/d1/	0.2224 0.0610 0.5889 -0.1661 -0.1573 1.0000 -0.3738 -0.1647 0.0957 -0.1154 -0.2817 0.2986 0.1084 0.4362 -0.0578 -0.1777	510	103	0.0010	0.07	0.0040	$2\pi/5$

Table A.6 Parametric speech corpus for HNM and LP model plossive /p/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/p/	1.0000 -0.6115 -0.1035 0.3185 -0.3596 -0.0363 0.2248	400	120	0.0003	0.03	0.0020	$2\pi/3$

Continued on next page

Table A.6 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/p1/	-0.3012 0.0970 0.6985 -0.3628 1.0000 -0.6750 -0.0679 0.1146 -0.1578 -0.0208 -0.0524 -0.0224 0.1078 0.0570 0.0240	560	130	0.0001	0.07	0.0009	$2\pi/4$

Table A.7 Parametric speech corpus for HNM and LP model fricative /s/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/s/	1.0000 -0.2756 0.3915 -0.1076 -0.0774 0.1934 -0.1841 -0.0547	600	105	0.0005	0.04	0.0003	$2\pi/6$

Continued on next page

Table A.7 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	-0.0209						
	0.0818						
	0.0348						
/s1/	1.0000	610	85	0.0001	0.04	0.0008	$2\pi/8$
	-0.0793						
	0.4513						
	0.0360						
	0.1201						
	0.0815						
	0.0881						
	-0.0976						
	-0.0473						
	0.0414						
	-0.0938						
/s2/	1.0000	610	100	0.0003	0.07	0.0003	$2\pi/6$
	-0.1930						
	0.3565						
	-0.0956						
	0.0213						
	0.1149						
	-0.0535						
	0.0333						
	-0.0089						
	-0.0098						
	-0.0286						
/s3/	1.0000	500	105	0.0005	0.04	0.0003	$2\pi/5$
	-0.2707						

Continued on next page

Table A.7 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.1956						
	-0.0199						
	-0.0721						
	0.2092						
	0.0626						
	0.0209						
	-0.0184						
	-0.1044						
	0.0032						

Table A.8 Parametric speech corpus for HNM and LP model fricative /h/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/h/	1.0000	1100	110	0.0003	0.07	0.0009	$2\pi/10$
	-1.4123						
	1.3827						
	-1.0168						
	0.3198						
	-0.0046						
	-0.0141						
	0.1760						
	0.0232						
	0.1712						
	-0.2206						
/h1/	1.0000	560	130	0.0001	0.07	0.0009	$2\pi/4$
	-1.2118						
	0.7793						

Continued on next page

Table A.8 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	-0.4490						
	-0.2315						
	0.3145						
	-0.0430						
	-0.0027						
	0.3064						
	0.0478						
	-0.2896						
/h2/	1.0000	1800	125	0.0003	0.07	0.0009	$2\pi/15$
	-1.2181						
	1.0200						
	-0.8743						
	0.4074						
	-0.2461						
	0.2979						
	-0.2531						
	0.5236						
	-0.1990						
	-0.0834						
/h3/	1.0000	900	120	0.0008	0.07	0.0009	$2\pi/8$
	-1.1279						
	1.0159						
	-0.5788						
	-0.1367						
	0.4393						
	-0.4330						
	0.3426						
	0.1500						

Continued on next page

Table A.8 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.0857						
	-0.0788						

Table A.9 Parametric speech corpus for HNM and LP model vowel /hello/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/hello/	1.0000	1100	133	0.0010	0.04	0.0040	$2\pi/8$
	-1.2937						
	0.4184						
	-0.3750						
	0.5077						
	-0.1034						
	0.2236						
	-0.1227						
	-0.1063						
	-0.5058						
	0.6124						
/hello1/	1.0000	1100	134	0.0010	0.07	0.0040	$2\pi/10$
	-1.3282						
	0.4293						
	-0.3296						
	0.5652						
	-0.1969						
	0.2014						
	-0.1256						
	-0.0805						
	-0.4353						

Continued on next page

Table A.9 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.5541						
/hello2/	1.0000	970	133	0.0011	0.04	0.0050	$2\pi/10$
	-1.3402						
	0.4451						
	-0.3059						
	0.4970						
	-0.1417						
	0.1788						
	-0.1092						
	-0.0875						
	-0.4444						
	-0.5595						
/hello3/	1.0000	970	133	0.0010	0.04	0.0050	$2\pi/10$
	-1.2623						
	0.2771						
	-0.1402						
	0.3500						
	-0.0332						
	0.1708						
	-0.1309						
	-0.0992						
	-0.4546						
	0.5830						

Table A.10 Parametric speech corpus for HNM and LP model vowel /hat/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/hat/	1.0000	1000	100	0.0010	0.04	0.0004	$2\pi/10$
	-0.8613						
	0.0599						
	0.1291						
	-0.0530						
	-0.2695						
	0.1561						
	-0.2297						
	0.1591						
	0.3313						
	-0.1295						
/hat1/	1.0000	800	100	0.003	0.04	0.0004	$2\pi/8$
	-1.3282						
	0.4293						
	-0.3296						
	0.5652						
	-0.1969						
	0.2014						
	-0.1256						
	-0.0805						
	-0.4353						
	0.5541						
/hat2/	1.0000	600	100	0.005	0.05	0.0003	$2\pi/6$
	-0.8222						
	0.0859						
	0.0600						
	0.0360						

Continued on next page

Table A.10 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	-0.3784						
	0.1653						
	-0.2673						
	0.2226						
	0.2281						
	-0.0035						
/hat3/	1.0000	1100	100	0.00045	0.04	0.0005	$2\pi/11$
	-0.9036						
	0.1123						
	0.1650						
	-0.1336						
	-0.2604						
	0.1520						
	-0.3122						
	0.3270						
	0.1443						
	-0.0091						

Table A.11 Parametric speech corpus for HNM and LP model vowel /too/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/too/	1.0000	800	136	0.0030	0.04	0.0030	$2\pi/6$
	-0.2511						
	-0.2468						
	0.1081						
	-0.6380						
	-0.0676						

Continued on next page

Table A.11 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.4222						
	-0.2624						
	0.3571						
	0.0878						
	-0.0427						
/too1/	1.0000	820	130	0.003	0.04	0.0030	$2\pi/6$
	-0.1797						
	-0.2493						
	0.0496						
	-0.6494						
	-0.0924						
	0.4275						
	-0.2161						
	0.3686						
	0.0884						
	-0.0730						
/too2/	1.0000	800	137	0.0032	0.04	0.0030	$2\pi/6$
	-0.1701						
	-0.2513						
	0.0524						
	-0.6521						
	-0.0839						
	0.4218						
	-0.2155						
	0.3663						
	0.0940						
	-0.0781						

Continued on next page

Table A.11 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/too3/	1.0000	700	135	0.0032	0.04	0.0040	$2\pi/11$
	-0.2507						
	-0.2491						
	0.0943						
	-0.6307						
	-0.0727						
	0.4461						
	-0.2561						
	0.3521						
	0.0896						
	-0.0422						

Table A.12 Parametric speech corpus for HNM and LP model vowel /door/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/door/	1.0000	1100	136	0.0025	0.035	0.0020	$2\pi/8$
	-0.4441						
	0.1128						
	-0.2556						
	-0.4838						
	0.1591						
	-0.0205						
	-0.0505						
	0.5348						
	-0.3928						
	0.3349						
/door1/	1.0000	1200	136	0.002	0.04	0.0040	$2\pi/9$

Continued on next page

Table A.12 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	-0.4959						
	0.1515						
	-0.3091						
	-0.4127						
	0.1314						
	0.0374						
	-0.0910						
	0.5560						
	-0.4361						
	0.3336						
/door2/	1.0000	1210	136	0.0018	0.05	0.0070	$2\pi/9$
	-0.6011						
	0.1722						
	-0.3119						
	-0.3639						
	0.1977						
	0.0519						
	-0.1038						
	0.5330						
	-0.4952						
	0.3456						
/door3/	1.0000	1100	137	0.0018	0.04	0.0040	$2\pi/9$
	-0.6173						
	0.1929						
	-0.2271						
	-0.4141						
	0.2441						
	-0.0546						

Continued on next page

Table A.12 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	-0.0965						
	0.4999						
	-0.4760						
	0.4053						

Table A.13 Parametric speech corpus for HNM and LP model fricative /shop/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/shop/	1.0000	1800	133	0.0008	0.04	0.0035	$2\pi/13$
	-1.2734						
	-0.1726						
	0.7166						
	0.1090						
	-0.5472						
	0.4116						
	-0.0061						
	0.0582						
	-0.5945						
	0.5002						
/shop1/	1.0000	1600	133	0.0008	0.04	0.0020	$2\pi/12$
	-1.2811						
	-0.1138						
	0.6427						
	0.0965						
	-0.5056						
	0.4658						
	-0.1047						

Continued on next page

Table A.13 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.0756						
	-0.5482						
	0.4764						
/shop2/	1.0000	9000	133	0.0007	0.04	0.0020	$2\pi/6$
	-1.2723						
	-0.0775						
	0.5374						
	0.1748						
	-0.5050						
	0.4301						
	-0.0766						
	0.1014						
	-0.6153						
	0.5126						
/shop3/	1.0000	1600	133	0.0010	0.04	0.0020	$2\pi/12$
	-1.2357						
	-0.1449						
	0.5569						
	0.1861						
	-0.4831						
	0.3997						
	-0.0329						
	0.0507						
	-0.6103						
	0.5272						

Table A.14 Parametric speech corpus for HNM and LP model fricative /that/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/that/	1.0000	600	131	0.0012	0.04	0.0040	$2\pi/5$
	0.0779						
	-0.9209						
	-0.7976						
	-0.2531						
	0.9704						
	0.4656						
	-0.2041						
	-0.0948						
	0.0103						
	-0.0141						
/that1/	1.0000	600	131	0.0012	0.04	0.0040	$2\pi/5$
	0.0949						
	-0.9203						
	-0.8379						
	-0.2470						
	0.9447						
	0.4962						
	-0.1776						
	-0.0948						
	0.0243						
	-0.0501						
/that2/	1.0000	610	131	0.0012	0.07	0.0035	$2\pi/5$
	0.1501						
	-0.8990						
	-0.8840						
	-0.3325						

Continued on next page

Table A.14 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	0.9485						
	0.5522						
	-0.1272						
	-0.1268						
	-0.0174						
	-0.0069						
/that3/	1.0000	600	132	0.0012	0.04	0.0045	$2\pi/5$
	0.1590						
	-0.9057						
	-0.9170						
	-0.3014						
	1.0106						
	0.5589						
	-0.1690						
	-0.2348						
	0.0054						
	0.0688						

Table A.15 Parametric speech corpus for HNM and LP model plossive /dig/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/dig/	1.0000	600	105	0.0005	0.04	0.0003	$2\pi/6$
	-0.2532						
	-0.3653						
	-0.0449						
	0.0874						
	-0.1705						
Continued on next page							

Table A.15 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
	-0.1628						
	-0.0221						
	0.2251						
	0.2167						
	-0.0192						
/dig1/	1.0000	800	100	0.0005	0.04	0.0003	$2\pi/8$
	-0.1264						
	-0.2507						
	-0.1883						
	-0.0391						
	-0.0967						
	-0.1582						
	-0.1267						
	0.1965						
	0.3743						
	0.0024						
/dig2/	1.0000	610	100	0.0003	0.07	0.0003	$2\pi/6$
	-0.0033						
	-0.4043						
	-0.2490						
	0.2188						
	-0.2144						
	-0.2767						
	-0.1684						
	0.5146						
	0.2970						
	-0.0987						

Continued on next page

Table A.15 – continued from previous page

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/dig3/	1.0000	500	105	0.0005	0.04	0.0003	$2\pi/5$
	-0.2303						
	-0.3123						
	-0.1288						
	0.1538						
	-0.2952						
	-0.1519						
	-0.0587						
	0.4683						
	0.0354						
	0.0629						

Table A.16 Parametric speech corpus for HNM and LP model plossive /pit/.

Phoneme	Coefficients	$Fmax$ (Hz)	F_o (Hz)	$A_o(k)$	$a_e(k)$	$n_o(t)$	$\theta(t)$
/pit/	1.0000	145	140	0.0006	0.09	0.0060	$2\pi/1$
	0.6773						
	0.2686						
	0.1281						
	0.3212						
	0.1140						
	0.0080						
	0.0698						
	0.0947						
	0.0919						
	0.0466						

Appendix B

Development Code

```
----------------------------------------------------------------------

-- Created by   :       Allen Mamombe
-- Create Date : 10:56:17 03/22/2008
-- Module Name : Global counter - Behavioral
-- Project Name :       Gcounter
-- Description : This module is used to generate the window timing
-- Revision 0.01 : For the Msc Electrical Engineering Degree
-- Additional Comments : This is also used to generate

----------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity Gcounter is
  Port ( sclk,reset             :       in      std_logic;
         start :       in              std_logic;
         check :       inout std_logic;
               counter :       inout   std_logic_vector(17 downto 0);
               counta  :       inout   std_logic_vector(10 downto 0);
               countb  :       inout   std_logic_vector(10 downto 0);
         Lt :  inout   std_logic_vector(3 downto 0)
          );
end Gcounter;

architecture Behavioral of Gcounter is

signal count :  std_logic_vector(10 downto 0):=(others=>'0');

        begin

        wcounter        : process (sclk,reset)

        begin

        if (reset = '0')then
```

```
        counter <= (others => '0');
        counta <= (others => '0');
        countb <= (others => '0');
              count <= (others => '0');
              Lt <= "0001";
              check <= '1';

        elsif( sclk'event and sclk = '1')then

              if (start = '1')then

               counter <= counter + '1';
               count <= count + '1';
               check <= '0';
               counta <=      counta + '1';

                      -- starts after 300
                      if (counter > "0000000001001011 00")then
                       countb <= countb + '1';
                       if (countb = "01001011001")then
                              countb <= (others=>'0');
                              else
                              end if;
                              else
                               countb <= countb;
                              end if;
                              -- reset after 600
                              if (counta = "01001011001")then
                                      counta <= (others=>'0');
                              else
                                      -- do nothing
                              end if;

                              if (count = "00100101100")then -- one less than the
                               Lt     <= Lt + '1';
                               count <= (others=>'0');
                              else
                                      -- do nothing
                              end if;
                      else
                              counta <= (others=>'0');
                              countb <= (others=>'0');
                              count   <= (others=>'0');
                              counter <= (others=>'0');
                              check   <= '1';
                              Lt <= "0001";
                              end if;
                      else
                              count   <= count;
                              counter <= counter;
                              check   <= check;
                              Lt              <= Lt;
                      end if;
              end process;

end Behavioral;
```

```
----------------------------------------------------------------------------

-- Created by    : Allen Mamombe
-- Create Date   :       10:56:17 03/22/2008
-- Module Name   : Global clock - Behavioral
-- Project Name  :       Gclock
-- Description   : This module is used to generate the speech rate clock
-- Revision 0.01 :       For the Msc Electrical Engineering Degree
-- Additional Comments : 8Khz is about 6250 pulses at 50Mhz clock

----------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity Gclock is
 Port ( clk,reset :     in      std_logic;
        sclk :  out     std_logic
                  );
end Gclock;

architecture Behavioral of Gclock is

signal  countout : std_logic_vector(13 downto 0):=(others=>'0');

begin

Khzclock        : process ( clk,reset )

        begin
                if (reset = '0')then
                        countout <= (others => '0');
                        sclk            <= '0';
                elsif( clk'event and clk = '1')then
                        countout <= countout + '1';
                        if (countout < "00110000110101")then
                                sclk <= '1';
                        elsif (countout < "01100001101010")then
                                sclk <= '0';
                        else
                                countout <= (others => '0');
                        end if;
                else
                        countout <= countout;
                end if;
        end process;

end Behavioral;
```

```
----------------------------------------------------------------------------
-- Company:
-- Engineer:              Allen Mamombe
--
-- Create Date:    10:56:17 03/22/2008
-- Design Name:
-- Module Name:    Interface - Behavioral
-- Project Name:          0912812638 ed 0765888969 rc 0763186021
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity adder is
Port (
        -- Input key and basic clock and reset instructions --
                cos1 : in std_logic_vector (17 downto 0);
                cos2 : in std_logic_vector (17 downto 0);
                cos3 : in std_logic_vector (17 downto 0);
                cos4 : in std_logic_vector (17 downto 0);
                cos5 : in std_logic_vector (17 downto 0);
                cos6 : in std_logic_vector (17 downto 0);
                cos7 : in std_logic_vector (17 downto 0);
                cos8 : in std_logic_vector (17 downto 0);
                cos9 : in std_logic_vector (17 downto 0);
        -- Outout cosine signals

                cosout : out std_logic_vector (17 downto 0)
                  );
end adder;

architecture Behavioral of adder is
begin

cosout <= (cos1 + cos2) + (cos3 + cos4) + (cos5 + cos6) +
  (cos7 + cos8) + cos9;

end Behavioral;
```

```
----------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    10:56:17 03/22/2008
-- Design Name:
-- Module Name:    lens - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity multiplier is
Port( hamm : in  std_logic_vector (17 downto 0);
        addedresidual   : in    std_logic_vector (17 downto 0);
        excitation : out std_logic_vector (17 downto 0)
        );
end multiplier;

architecture Behavioral of multiplier is

-- Multiplier for the hamming window

component MULT18X18
port(
A : in std_logic_vector (17 downto 0);
B       : in std_logic_vector (17 downto 0);
P       : out std_logic_vector (35 downto 0)
);
end component;


-- Begin declaration of signals --

signal  residualsig     : std_logic_vector (35 downto 0);

-- End declaration of signals --

begin


-- Define port maps for the multiplier

MULTI18 : MULT18X18
```

```
port map (
A => hamm,                                -- insert input signal
B => addedresidual,      -- insert input signal
P => residualsig                 -- insert output signal
);

-- Down cast the residual bits 0-26 thus

excitation(17) <=       residualsig(35); -- Maintain the sign
excitation(16 downto 0) <= residualsig(26 downto 10);

end architecture Behavioral;
```

```
----------------------------------------------------------------------------------

-- Created by : Allen Mamombe
-- Create Date : 10:56:17 03/22/2008
-- Module Name : Keyin - Behavioral
-- Project Name :       Keyin
-- Description : This module is used to latch the iput from the keyboard and
-- trigger the speech sythesis computation
-- Revision 0.01 : For the Msc Electrical Engineering Degree
-- Additional Comments : The ec will always be triggered to restart until
-- button is released then speech processing can resume not

----------------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity Keyinput is
Port( clk,reset,check   :       in      std_logic;
        keyin : in std_logic_vector     (3 downto 0);
        counter :       in std_logic_vector     (17 downto 0);
        start : inout   std_logic;
         keyout :       inout   std_logic_vector        (3 downto 0)
        );
end Keyinput;

architecture Behavioral of Keyinput is

begin

latchkey        : process ( clk,reset,keyin )

        begin
                if (reset = '0')then
                                keyout <= (others => '0');
                                start    <= '0';
                elsif( clk'event and clk = '1')then
                        if (counter = "00000111111010000000")then
                                start <= '0';
                        else
                                case keyin is
                                        when    "0001"  =>
                                        if (check = '0') then
                                                -- do nothing
                                        else
                                                keyout <= "0001";
                                                start    <= '1';
                                        end if;
                                        when    "0010" =>
                                        if (check = '0') then
                                                -- do nothing
                                        else
                                                keyout <= "0010";
                                                start    <= '1';
```

```
                                        end if;
                                    when    "0011"  =>
                                        if (check = '0') then
                                            -- do nothing
                                        else
                                            keyout <= "0011";
                                            start    <= '1';
                                        end if;
                                    when    "0100" =>
                                       if (check = '0') then
                                            -- do nothing
                                        else
                                            keyout <= "0100";
                                            start    <= '1';
                                        end if;
                                    when    "0101"  =>
                                        if (check = '0') then
                                            -- do nothing
                                        else
                                            keyout <= "0101";
                                            start    <= '1';
                                        end if;
                                    when    "0110" =>
                                        if (check = '0') then
                                            -- do nothing
                                        else
                                            keyout <= "0110";
                                            start    <= '1';
                                        end if;
                                    when    "0111"  =>
                                       if (check = '0') then
                                            -- do nothing
                                        else
                                            keyout <= "0111";
                                            start    <= '1';
                                        end if;
                                    when    "1000"  =>
                                        if (check = '0') then
                                            --- do nothing
                                        else
                                            keyout <= "1000";
                                            start    <= '1';
                                        end if;
                                    when    others =>
                                        keyout <= keyout;
                                        start    <= start;
                                    end case;
                                end if;
                            else
                                        keyout <= keyout;
                                        start    <= start;
                            end if;

        end process;

end Behavioral;
```

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date : 10:56:17 03/22/2008
-- Design Name :
-- Module Name : lens - Behavioral
-- Project Name :
-- Target Devices :
-- Tool versions :
-- Description :
--
-- Dependencies :
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity addnoise is
 Port ( sclk,reset : in         std_logic;
 residual : out  std_logic_vector (17 downto 0));
end addnoise;

architecture Behavioral of addnoise is

-- -- Begin declaration of signals -- --

signal random   : std_logic_vector (17 downto 0):= (others => '0');
signal q : std_logic_vector(10 downto  0);
constant seed : std_logic_vector(10 downto  0):= (others => '1');

-- End declaration of signals --

begin

        noiseadd : process(sclk,reset)

        begin

        if(reset='0') then
                q <= seed;  -- set seed value on reset

        elsif (sclk'event and  sclk='1') then -- clock with rising edge

                q(0) <= q(7); -- feedback to LS bit
                q(1) <= q(0);
                q(2) <= q(1) xor q(10); -- tap at stage 1
                q(3) <= q(2) xor q(10); -- tap at stage 2
                q(4) <= q(3) xor q(10); -- tap at stage 3
                q(5) <= q(4) xor q(10); -- tap at stage 4
```

```
                q(6) <= q(5) xor q(10); -- tap at stage 5
                q(7) <= q(6) xor q(10);  -- tap at stage 6
                q(10 downto 8) <= q(9 downto 7); -- others bits shifted

                random(10 downto 0) <= q(10 downto 0);

        end if;

end process noiseadd;

residual <= random;

end Behavioral;
```

```
----------------------------------------------------------------------------
-- Company:
-- Engineer: Allen Mamombe
--
-- Create Date: 10:56:17 03/22/2008
-- Design Name:
-- Module Name: cosine - Behavioral
-- Project Name: 0912812638 ed 0765888969 rc 0763186021
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity residualadd is
 Port (
 -- Input key and basic clock and reset instructions --
 cosout : in std_logic_vector (17 downto 0);
 residual : in std_logic_vector (17 downto 0);
 addedresidual : out std_logic_vector (17 downto 0)
  );
end residualadd;

architecture Behavioral of residualadd is
begin

addedresidual <= cosout + residual;

end Behavioral;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This scripts reads in a wave file of a neutral voiced phoneme, high  %
%  pass filters to detrend the data and remove room noise, computes     %
%  linear predictive coefficients (LPCs) for the segment and plots the  %
%  prediction error and reconstructed signal with FIR and IIR           %
%  implementations of the filter. The pitch is estimated from the error %
%  signal and a simple synthesised signal is created to imitate real    %
%  speech at a different pitch. A pole-zero diagram is create for the   %
%  IIR reconstruction of the signal                                     %
%  the formants are identified and the PSD and Spectrum from the LPCs   %
%  is plotted.                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


winlens = 50;  %PSD window length in milliseconds
[y,fs] = wavread('aaa3.wav');  %  Read in wavefile
winlen = winlens*fs/1000;
[cb,ca] = butter(5,2*100/fs,'high');  %  Filter to remove LF noise
yf = filtfilt(cb,ca,y);
[a,er] = lpc(yf,45);  %  Compute LPC coefficent with model order 10
i = [1:4000];
kd=1;   %  Starting figure number
figure(kd) ; plot((i/8000),y(1:4000)); title('Time␣Signal␣Vowel␣/a/');
xlabel('Time␣(s)'); ylabel('Magnitude')

predy = filter(a,1,yf);  %  Compute prediction error  all zero filter
figure(kd+1) ; plot((i/8000),predy(1:4000)); title('Residual␣Signal␣Vowel␣/a/');
xlabel('Time␣(s)'); ylabel('Magnitude')
predyb = abs(fft(predy));
predyphase = unwrap(angle(predy));

%figure(kd+2)
%plot(predyphase*180/pi);

recon = filter(1,a,predy);  %  Compute reconstructed signal all-pole filter
figure(kd+2)
plot(i,predyb(1:4000)); title('Frequency␣Spectrun␣Residual␣Signal␣Vowel␣/a/');
 xlabel('Frequency␣KHz'); ylabel('Magnitude␣dB')

g = [];
for k=1:50
    g = [g, 1, zeros(1,9)];
end
figure(kd+3)
plot(g); title('Unit␣Impulse␣Source␣Signal␣/a/'); xlabel('Time␣(s)');
ylabel('Magnitude')

T = .100;
t = 0:1/8000:T;
d = 0:.01:T;
width=.01;
y1 = pulstran(t,d,'tripuls',width,0.8);
figure(kd+4)
plot(t,(y1/50)); title('Triangular␣Pulse␣Source␣Signal␣/a/');
xlabel('Time␣(s)'); ylabel('Magnitude')

for ( k = 1:100)
```

```
    sxi(k) = (k/10)*exp(1-(k/10));
end
for ( k = 1:100)
    sxi(k+100) = sxi(k);
end
for ( k = 1:100)
    sxi(k+200) = sxi(k);
end
for ( k = 1:100)
    sxi(k+300) = sxi(k);
end
for ( k = 1:100)
    sxi(k+400) = sxi(k);
end
for ( k = 1:100)
    sxi(k+500) = sxi(k);
end
for ( k = 1:100)
    sxi(k+600) = sxi(k);
end
zi = 1:700;
figure(kd+5)
plot((zi/8000),sxi); title('R_K␣Approximation␣Source␣Signal␣/a/');
xlabel('Time␣(s)'); ylabel('Magnitude')

i = 1
for (t = [0:.01:1])
    gx(i) = ((t-.1078)/(.6576-.1078))*((t-.1078)/(.6576-.1078))*
      (3-2*((t-.1078)/(.6576-.1078)));
    i = i + 1;
end
i = 66
for (t = [.66:.01:1])
    gx(i) = (1-(((t-.6576)/(1-.657))*((t-.6576)/(1-.657))));
    i = i + 1;
end
gx(101)=0;
for (kx = [1:10])
    gx(kx)=0;
end
length(gx)
figure(kd+6)
plot(gx); title('Modified␣Model␣R_K␣Source␣Signal');
xlabel('Time␣(s)'); ylabel('Magnitude')

for ( k = 1:100)
    gx1(k) = gx(k);
end
for ( k = 1:100)
    gx1(k+100) = gx(k);
end
for ( k = 1:100)
    gx1(k+200) = gx(k);
end
for ( k = 1:100)
    gx1(k+300) = gx(k);
end
for ( k = 1:100)
```

```
    gx1(k+400) = gx(k);
end
for ( k = 1:100)
    gx1(k+500) = gx(k);
end
for ( k = 1:100)
    gx1(k+600) = gx(k);
end

figure(kd+7)
plot(gx1); title('Modified␣R_K␣Source␣Signal␣/a/');
xlabel('Time␣(s)'); ylabel('Magnitude')

for (t = 1:8000)
   yz(1) = 0;
    for (i = [2:20])
        yz(i) = yz(i-1)+(.015*exp(0.07*(1-(i-1))))*
          cos((2*pi*(116/8000)*(i-1)*t)+0.72*(i-2));
    end
    yz(t)=yz(20);
end
yzy = [1:8000];

figure(kd+8)
plot(yzy/8000,yz); title('Modified␣HNM␣Source␣Signal␣/a/');
xlabel('Time␣(s)'); ylabel('Magnitude')

yb = fft(yz);

figure(kd+9)
plot(abs(yb)); title('Frequency␣Spectrum␣HNM␣Source␣Signal␣/a/');
xlabel('Time␣(s)'); ylabel('Magnitude')


[CM,PM] = max(predyb)
tm = length(y)

for i = 1:8000
    predt(i) = 0.05*(0.5*sin(2*pi*116*i/8000) +
      0.48*sin(2*pi*2*116*i/8000) + 0.46*sin(2*pi*3*116*i/8000) +
      0.44*sin(2*pi*4*116*i/8000) + 0.42*sin(2*pi*5*116*i/8000) +
      0.40*sin(2*pi*6*116*i/8000) + 0.38*sin(2*pi*7*116*i/8000) +
      0.36*sin(2*pi*8*116*i/8000) + 0.34*sin(2*pi*9*116*i/8000) +
      0.32*sin(2*pi*10*116*i/8000) + 0.30*sin(2*pi*11*116*i/8000) +
      0.28*sin(2*pi*12*116*i/8000) + 0.26*sin(2*pi*13*116*i/8000) +
      0.24*sin(2*pi*14*116*i/8000) + 0.22*sin(2*pi*15*116*i/8000) +
      0.20*sin(2*pi*16*116*i/8000) + 0.18*sin(2*pi*17*116*i/8000) +
      0.16*sin(2*pi*18*116*i/8000));
end

tester = predt + 0.03*randn;

pirty1 = abs(fft(tester));
figure(kd+9)
plot(pirty1);

pirty = abs(fft(predt));
figure(kd+9)
```

```
plot(pirty);

recon3 = filter(1,a,tester);  %  Compute reconstructed signal
figure(kd+9)
%  Plot reconstructed signal
plot(recon3,'k')
wavwrite(recon3,'ale2');

recon = filter(1,a,predy);  %  Compute reconstructed signal
figure(kd+9)
%  Plot reconstructed signal
plot(recon,'b')
wavwrite(recon,'ale');

recon2 = filter(1,a,predt);  %  Compute reconstructed signal
figure(kd+9)
%  Plot reconstructed signal
plot(recon2,'g')
wavwrite(recon2,'ale1');
hold on
%  Plot with orginal delayed by a unit so it does not entirely
%  the perfectly reconstructed signal
plot(yf(2:end),'r')
hold off
xlabel('Samples'); ylabel('Amplitude')
title('Reconstructed␣Signal␣(blue)␣and␣Original␣(red)')

% Estimating the fundamental frequency
ms20 = fs/100;
res = xcorr(yf,ms20,'coeff');

% plot the autocorretion of the signal

des = (-ms20:ms20)/fs;

figure(kd+10)

plot(des,res)

% Evaluate the fundamental frequecy

ms2 = fs/1000;
rr = res(ms20+1:2*ms20+1);
[rmax,tx] = max(rr(ms2:ms20))
fprintf('rmax=%g␣fmax=%gHz\n',rmax,fs/(ms2+tx-1));
%  By examining a the error sequence,
%  generate a simple impulse sequence to simulate its period
%  (about 103 sample period)
g = [];
for k=1:150
    g = [g, 1, zeros(1,103)];
end
%  Run simulated error sequence through all pole filter
sim = filter(1,a,g);
%soundsc([(sim')/std(sim); zeros(fix(fs)*1,1); yf/std(yf)],fs)
%  Compute reconstructed signal from error and all-pole filter
figure(kd+9)
%  Plot reconstructed signal
```

```
plot(sim,'b')
wavwrite(sim,'ale3');
%Estimation of the maximum voiced frequency Fmax less 13db
% Rosenburg Klatt analysis
%  Plot pole zero diagram
figure(kd+9)
r  = (roots(a))
w = [0:.001:2*pi];
plot(real(r),imag(r),'xr',real(exp(j*w)),imag(exp(j*w)),'b')
title('Pole diagram of vocal tract filter')
xlabel('Real'); ylabel('Imaginary')

%  Find resonant frequencies corresponding to poles
froots = (fs/2)*angle(r)/pi;
nf = find(froots > 0 & froots < fs/2);  %  Find those corresponding
figure(kd+9)
%  Examine average specturm with formant frequencies
[pd,f] = pwelch(yf,hamming(winlen),fix(winlen/2),2*winlen,fs);
dbspec = 20*log10(pd);
mxp = max(dbspec);  %  Find max and min points for graphing verticle lines
mnp = min(dbspec);
plot(f,dbspec,'b')  %  Plot PSD
hold
% Over lines on plot where formant frequencies were estimated from LPCs
for k=1:length(nf)
plot([froots(nf(k)), froots(nf(k))], [mnp(1), mxp(1)], 'k--')
end
hold off
title('PSD plot with formant frequencies (Black broken lines)')
xlabel('Hertz')
ylabel('dB')
%  Get spectrum from the AR (LPC) parameters
[hz,fz] = freqz(1, a, 1024, fs);
figure(kd+9)
plot(fz,abs(hz))
title('Spectrum Generated by LPCs')
xlabel('Hertz')
ylabel('Amplitude')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Written by Allen Mamombe, October 2007                                     %
%  This script generates a wav file based on the input                        %
%  filter and residual parameters ak and lp                                   %
%  The script utilises LP and HNM models to generate the speech               %
%  Absolute window length                                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
windowlength = 150;  %PSD window length in milliseconds

%  Compute the random white noise
wn = wgn(8000,1,2)';

%  Filter the noise
b = remez(20,[0 0.05 0.88 1],[0 0 1 1]);
a = [1];
wn = filter(b,a,ws);

%  Compute the residual signal
for (t = 1:8000)
   yz(1) = 0;
    for (i = [2:20])
        yz(i) = yz(i-1)+(.015*exp(0.07*(1-(i-1))))
          *cos((2*pi*(116/8000)*(i-1)*t)+0.72*(i-2));
    end
    yz(t)=yz(20);
end

%  Merge the residual signal
yout = yz + 0.09*ws1;
yb1 = fft(yz1);

%  Plot the residual signal
figure(kd+10)
plot(l,abs(yb1(1:4000))); title('Frequency␣Spectrum␣HNM␣Source␣Signal␣/a/');
xlabel('Frequency␣(Hz)'); ylabel('Magnitude')

%  Generate the speech signal
reconstructed = filter(1,ak,yout);  %  Compute reconstructed signal
figure(kd+2)
plot(i,predyb(1:4000)); title('Frequency␣Spectrun␣Residual␣Signal␣Vowel␣/a/');
xlabel('Frequency␣KHz'); ylabel('Magnitude␣dB')

%  Write the output wav file
wavwrite(reconstructed,aout);
```

Appendix C

Publications from the thesis

Optimised Source Signal Modelling for Linear Predictive Speech Synthesis

A Mamombe and Bea Lacquet

Department of Electrical and Information Engineering
University of the Witwatersrand, Johannesburg, South Africa.
a.mamombe@ee.wits.ac.za

Abstract

Linear predictive (LP) speech synthesisers still play an important role in linguistic analysis and speech processing. However, the quality of speech produced from such synthesisers still falls short of many people's expectations. This paper discusses ways of improving the quality of speech-produced by LP synthesisers through unique source signal models. Popular models of the source signal include the Rosenburg Klatt (R-K), the triangular pulse, codebooks and the unit impulse [1]. Tests have proved that the R-K model is the most favourable [2], though it has limitations related to the processing difficulties and accounting for fricative noise. Two fairly new source signal modelling techniques that solve this problem are discussed in this paper namely 1) A linear modification of the R-K signal and 2) A modification of the Harmonic plus noise (HNM) speech processing technique to model the source signal [2],[6]. Favourable results were obtained when using the HNM technique for vowel sounds.

Keywords: Linear Prediction, Source Signal Modelling, Harmonic plus noise.

1. Introduction

Linear predictive synthesis, is a technique based on the auto-regressive model as shown in equation 1,2 [3]. The two main parameters of LP synthesis are the predictive coefficients a_k (*vocal tract filter characteristics*) and the source signal $e(n)$ (*the glottal pulse source signal*).

$$\widetilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k] \tag{1}$$

$$e[n] = x[n] - \widetilde{x}[n] \tag{2}$$

$x[n]$ is the actual speech signal. $\widetilde{x}[n]$ is the predicted sample at instant n and a_1,a_2,....,a_k are predictor coefficients.

There are various methods of obtaining the filter parameters a_k and the residual signal $e(n)$ as discussed in [4]. Once the filter parameters and the residual signal *(source signal)* is known, speech can be synthesised by passing the residual signal $e(n)$ through an all pole filter with transfer characteristics shown in equation 3[4]. The filter parameters are stored in a codebook and residual signal (source signal) is either stored or modelled using the unit impulse, triangular or R-K methods [1]. Modelling the residual signal greatly reduces the need for a bigger memory but compromises quality. This paper presents a brief critical overview of the existing source signal modelling techniques. Proposed techniques for improving the quality of the source signal models are presented and discussed.

2. Source Signal Modelling

The following sections will discuss various ways of modelling the source signal accurately whilst maintaining highly natural and intelligible speech. In order to achive this an algorithm was developed in MATLAB to obtain the residual signal and LPC parameters, for the vowel /a/ shown in Fig 1.0 sampled at 8 KHz. The residual signal obtained from the algorithm is shown in Fig 2.0 next section.

$$H(z) = \frac{1}{\sum_{k=1}^{p} a_k z^{-k}} \tag{3}$$

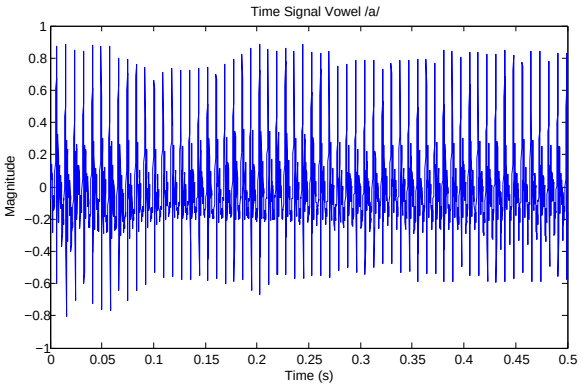


Fig 1.0 Time domain signal for vowel /a/

Most linear predictive (LP) synthesisers tend to simplify matters once the residual signal is obtained, by using an impulse train, R-K or a triangular pulse signal in the modelling *(source signal)* [1].

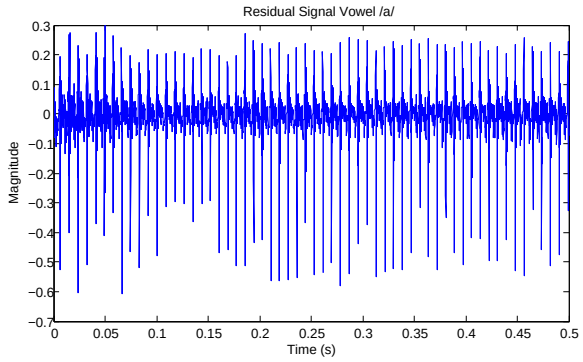


Fig 2.0 Time domain residual signal for vowel /a/

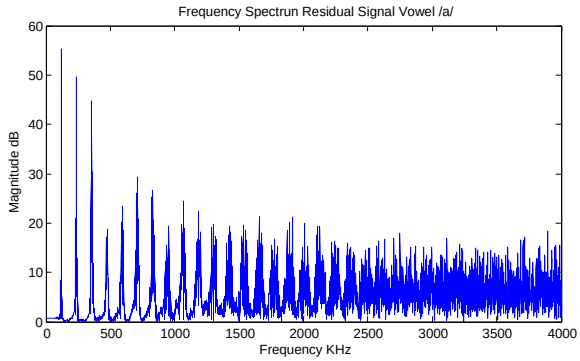


Fig 3.0 Frequency Domain Residual Signal for Vowel /a/

Two fairly new techniques that employ the *modified R-K and the HNM synthesis* to model the source signal are also discussed [6]. The criterion used to quantify the quality of the source signal models discussed is that ideally the model should exhibit characteristics similar to those of the actual residual signal in Fig 2.0 , 3.0 and produce intelligible speech.

2.1. Current Source Signal Modelling Techniques

In this section, we give discriptions of the current residual/source signal modelling techniques namely the triangular pulse, the unit impulse and Rosenburg Klatt (R-K). By applying them in synthesis to the vowel /a/.

2.1.1. Impulse Train

The impulse train Fig 4.0 was used to model the source signal for a vowel /a/ shown in Fig 2.0 The method produced reasonable speech quality for the vowel /a/; however, comparing the frequency and magnitude components of the signal in Fig 2.0 it is evident that the impulse train Fig 4.0 is far from the ideal residual signal.

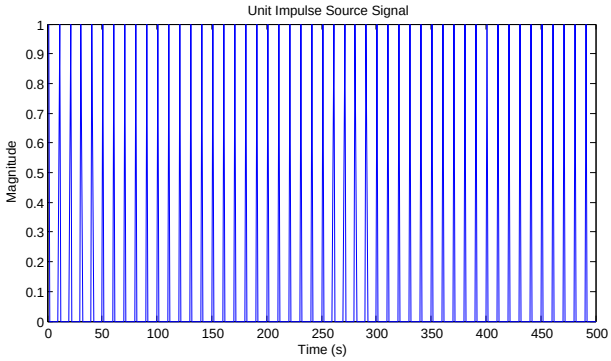


Fig 4.0 Impulse Train Source Signal model

The pitch period To of the impulse train is derived from the pitch frequency Fo, that is the frequency of the largest harmonic in the source signal [5]. Such that $To = 1/Fo$.

2.1.2. Triagular Pulse Approximation

Most LPC based speech synthesisers use the triangular pulse Fig 5.0 as the source signal [1]. The triangular pulse is a good estimate of the source signal *(actual glottal pulse)* and is easier to generate unlike the R-K signal. The triangular signal in Fig 5.0 was applied as the source signal to synthesise the vowel /a/ using Linear prediction. The resulting synthetic speech produced was fairly intelligible and is further discussed in the results section.

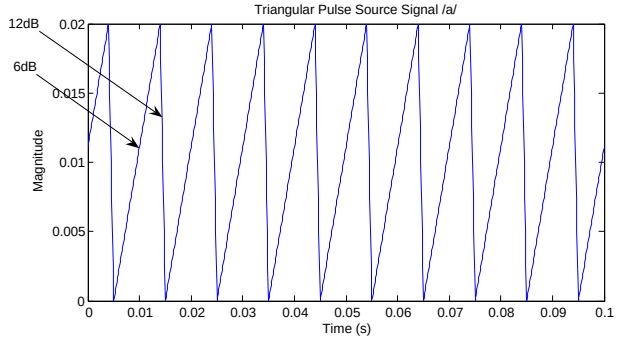


Fig 5.0 Triangular Pulse Source Signal model

2.1.3. The R-K Source Signal model

Literature suggests that a better way of modelling the source-signal is the use of the R-K model [1]. Rosenburg reported that the source signal produced a more natural speech when modelled similar to the glottal excitation signal Fig 6.0. He derived a polynomial that closely modelled the glottal pulse shown in equation 4 [1]. Morden research has simplified this polynomial as a unit impulse driven through a filter or simply modelled the signal as in equation 5 [1]. The R-K signal was modified for the experiment in order to reduce the computational requirements as shown in the next section.

$$g(t) = \begin{cases} 0 & for\ 0 \le t \le t_1 \\ A(\frac{(t-t_1)}{(t_2-t_1)})^2(3-2\frac{(t-t_1)}{(t_2-t_1)}) & for\ t_1 \le t \le t_2, \\ A(1-\frac{(t-t_2)}{(b-t_2)}) & for\ t_2 \le t \le b \end{cases} \quad (4)$$

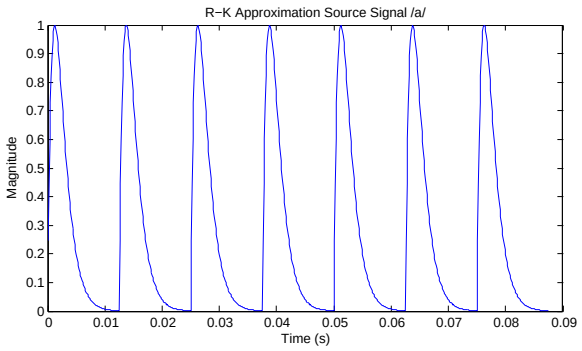


Fig 6.0 R-K Approximate Source Signal model

The approximate R-K equation

$$g(t) = A\frac{t}{T_0}\exp(1-\frac{t}{T_0}) \quad (5)$$

Were T_0 is the period of the pitch frequency and $g(t)$ the source signal.

2.2. Optimised Source Signal Modelling

The following sections of the paper describe two fairly new source signal models that the authors used for LPC speech synthesis. The first is the linear modification of the R-K signal and the second is the use of the HNM synthesis to model the source signal.

2.2.1. Modification of the R-K Source Signal

A new technique discussed in this paper is a linear modification of the R-K source signal. A set of linear ratios were used to simplify the computation of the signal by relating the values t_1, t_2, b *from equation 4* to the pitch period To. The ratios used in relating the variables t_1, t_2 , b and T_0 are presented in equation 6. By specifying the variable ratios, the R-K polynomial was reduced to Equation 7. The derived model from this modification is shown in Fig 7.0. The resulting source signal was used to synthesise the vowel /a/ and produced equally intelligible speech as the R-K polynomial.

$$b = T_0 \quad t_1 = 0.111b = aT_0 \quad t_2 = 0.667T_0 = cT_0 \quad (6)$$

$$g(t) = \begin{cases} 0 & 0 \le t \le aT_0 \\ A(\frac{(t-aT_0)}{(cT_0-aT_0)})^2(3-2\frac{(t-aT_0)}{(cT_0-aT_0)}) & aT_0 \le t \le cT_0 \\ A(1-\frac{(t-cT_0)}{(T_0-cT_0)}) & cT_0 \le t \le T_0 \end{cases} \quad (7)$$

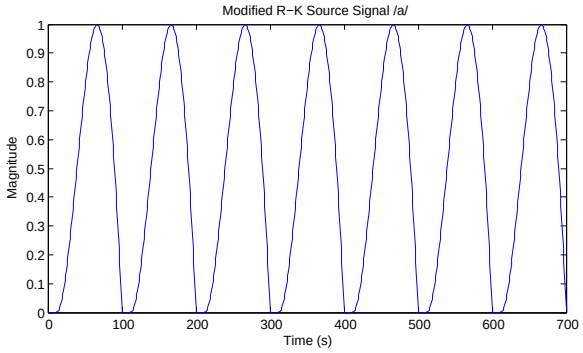


Fig 7.0 Modified R-K Source Signal model

2.3. HNM Synthesis

A fairly new technique discussed in this paper is modelling the source signal using HNM [6]. HNM is a speech synthesis and modelling technique on its own [2]. Research has generally shied away from this technique because of the complication in finding the HNM model parameters [7].

The harmonic plus noise model (HNM) is based on the fact that speech can be viewed as two components, namely the harmonic part $h(t)$ a quasi periodic signal and the non periodic component noise $n(t)$. These two components are distinctly separated by a time varying quantity $Fmax$ (maximum voiced frequency). The lower component is solely composed of harmonics and the upper band noise as shown in Fig 8.0 and Equation 8 [2].

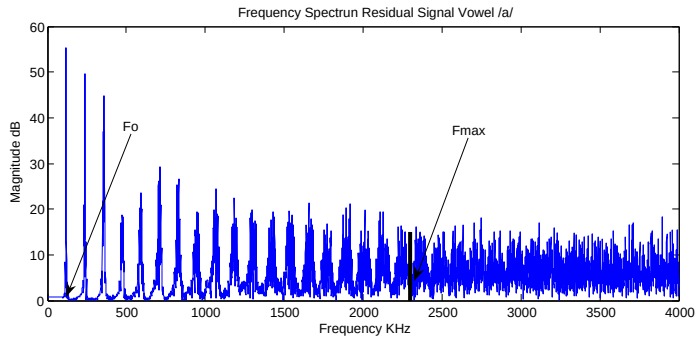


Fig 8.0 Characteristics HNM Signal model

$$h(t) = \sum_{k=1}^{K} A_k(t)cos(k\theta(t) + \theta_k(t) \quad (8)$$

$n(t)$ noise component is derived from filtered white Gaussian noise.

The number of harmonics K is given by $Fmax/Fo$ were Fo is the pitch frequency [7].

2.3.1. HNM Source Signal Modelling

This section describes how the HNM technique was used to model the source signal, as well as deriving the parameters for HNM equation 8. From Fig 8.0 it is clear the residual signal exhibits characteristics equivalent to those of the actual speech signal. Therefore, the source signal can be described as a sum

of the harmonic and noise of the residual. The major complecation as stated earlier is the derivation of HNM parameters Fo, $Fmax$, θ and A_k. The techniques we applied in solving the HNM parameters are explained below.

2.3.2. F_0 and $Fmax$ Estimation

F_0 is defined as the pitch frequency and is given as the frequency of the first harmonic [5]. Once the F_0 was obtained then $Fmax$ and the number of harmonics K were calculated based on the relationship in equation 9 [7].

$$maxA_i - A_n \geq 13db \tag{9}$$

A_n is the average magnitude of the noise spectrum
Were A_k is the peak amplitude in the range specified in equation 10

$$[F_k - \frac{F_0}{2}, F_k + \frac{F_0}{2}] \tag{10}$$

F_k is a multiple of F_0 the fundamental Frequency such that $F_k = kF_0$

*The first instant that equation 10 is not satisfied defines the number of harmonics in the signal spectrum as K and the maximum voiced frequency as Fmax.

2.3.3. Phase modelling

One complexity of HNM is computing the phase from the frequency domain waveform [8]. A method of linearity was used to model the phase relationships between HNM harmonics [2]. Tests were performed by observing the quality produced for vowel sounds /a/, /e/, /i/, /o/, /u/ when the phase of all the harmonics was varied linearly over a 360, 180, 270, 90 degree intervals equation 11. Positive results were obtained for all vowels when the phase was varied on the 360 degree interval.

$$\theta_k = (\frac{2\pi}{K})(k-1) \tag{11}$$

2.3.4. Modelling the harmonic and noise interaction

The source signal models discussed thus far fail to model effectively the noise interaction between the harmonics *(voiced source)* and the noise *(unvoiced source)* [2]. This is because the R-K, triangular and impulse signal models assume the source signal to be purely harmonic or noise [1]. As a solution to this problem the HNM synthesis model developed, allows the modelling of the noise interaction by multiplying the developed source signal with a noise window at the interaction of the two components. The noise window is equivalent to passing a white Gaussian noise through a band pass filter bounded by $0.75Fmax \quad and \quad 0.85Fmax$. The resulting residual is shown in Fig 9; clearly this is a better approximate of the residual signal. The vowel /a/ was synthesised using this source signal model and satisfactory results were obtained when comparing the intelligibility with the other source signal models.

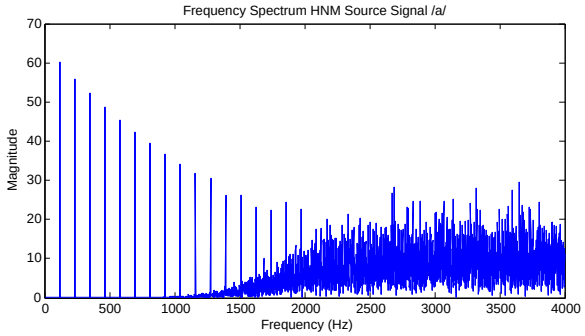


Fig 9.0 HNM Source Signal model

3. Discussion

The models discussed in this paper were not only tested for their intelligibility in synthesising the vowel /a/ but also for vowels /e/, /i/, /o/, /u/. The results of the listening tests for the source signal models are shown in Table 1.0. A scale of 1-5 was used to classify the quality of the synthetic speech produced from all the five source signal models when applied to LPC synthesis. Where 1 is poor inaudible quality and 5 best audible quality.

SS Model	**/a/**	**/e/**	**/i/**	**/o/**	**/u/**
Impulse Train	2.5	3	3	3	2.5
Triangular Pulse	3	3.2	3.2	3	3
R-K Signal	3.5	3.8	3.8	3.5	3.5
Modified R-K	3.4	3.8	3.8	3.4	3.5
HNM Source Signal	3.6	4.0	4.0	3.3	3.3

Table 1.0 Performance of the source signal models for vowel LP synthesis

From the results it is evident that the HNM model produced better synthetic speech. It is also evident that the modified R-K and the original R-K source signal models were comparable.

4. Conclusions

The paper has described two fairly new approaches to source signal modelling for LPC synthesis based on HNM and a linearization of the R-K model. Other well documented source signal modelling methods for LPC synthesis were briefly described in this paper. The two modified models produced better quality synthetic speech when compared to previously renowned simplified models such as Impulse train for the vowels /a/, /e/, /i/, /o/, /u/. Further testing still has to be done for fricative and nasal sounds using these described models.

5. Acknowledgements

The authors would like to thank the Electronic Engineering research group at the University of Witwatersrand Johannesburg and the department of trade and industry in South Africa for providing funding through the THRIP. Finally yet importantly, the authors would also like to thank Gedion Klompje previously of the language-processing group at the University of Stellenbosch in South Africa for sharing ideas in the field of speech synthesis.

6. References

[1] I.H. Witten, *Principles of Computer Speech*, Academic Press, 1982.

[2] Y Stylianou “On the implementation of the harmonic plus noise model for concatenative speech synthesis,” *In Proceedings. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP Volume 2, Issue 2000*. pp II957 - II960 , Istanbul Turkey, 9 June 2000.

[3] J. Makhoul. ”Linear prediction: A tutorial review,” *In Proceedings of the IEEE, vol 63*. pp 561-580, April 1975.

[4] F.J. Owens, *Signal Processing of Speech*, The Macmillan Press Ltd, 1993.

[5] S Roa, M Bennewitz, S Behnke “Fundamental frequency estimation based on pitch-scaled harmonic filtering,” *In Proceedings. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP Volume: 4*. pp IV-397-IV-400, Honolulu Hawaii, 15-20 April 2007.

[6] G Klompje, T.R Niesler, ”A parametric monophone speech synthesis system”, *In proceedings of the seventeenth annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, Parys South Africa, November 2006.

[7] Y Stylianou “Applying the harmonic plus noise model in concatenative speech synthesis,” *IEEE Transactions on speech and audio processing, Volume 9, Issue 1*. pp 21 - 29 , January 2001.

[8] Y Stylianou “Concatenative speech synthesis using the harmonic plus noise model,” *Third ESCA Speech Syntheis Workshop*. pp 261 - 266 , November 1998.

An optimised parametric speech synthesis model based on Linear prediction (LP) and the Harmonic plus noise model (HNM)

Allen Mamombe[1], *Beatrys Lacquet*[1]

[1]Electrical and Information Engineering, University of Witwatersrand, Johannesburg, South Africa

a.mamombe@ee.wits.ac.za, b.lacquet@ee.wits.ac.za

Abstract

Linear predictive speech synthesis plays an important role in acoustic verification and analysis. This is because system parameters can be tuned to account for prosody and intonation. The quality and intelligence of speech produced from such parametric synthesisers however falls short of many people expectations. In this paper we discuss a parametric speech model based on Linear Prediction (LP) and Harmonic plus Noise Model (HNM). We investigate ways of optimising our LP parameters and window lengths. We describe a mathematical model for LP and HNM speech synthesis. Mean opinion score (MOS) and transcription tests were then carried out on English phonemes and words synthesised using our model and renowned LP models i.e Rosenburg-Klatt (R-K) and Unit impulse. The test sample was composed of 20 native South African English listeners. The results of both tests favoured speech synthesised with our LP/HNM model when compared with renowned LP models based on the R-K and Unit impulse.

Index Terms: harmonic plus noise, linear prediction, parametric synthesis, transcription tests and subjective quality tests.

1. Introduction

Linear prediction (LP) is based on an autoregressive model that calculates future samples of a signal based on past predicted samples [1]. In its simplicity the LP model constitutes of a source signal $e(n)$ passing through an all pole filter defined by LP coefficients a_k. It can be proved mathematically that if the source signal is the exact replica of an inverse filtering process using LP coefficients, then the speech produced from such a model is indistinguishable from the actual speech equation 1[1]. To model as accurately as possible the residual signal we proposed the use of the HNM. The complexity with the HNM method is in finding the model parameters. We discuss in this paper a simplified mathematical HNM model specifically for modelling the residual [2]. We proceed to give a brief background on proposed models and discuss mathematical formulas to determine the optimal number of LP parameters and the window lengths.

2. Background

The source filter model discussed above emulates the human auditory system by modelling the acoustic process as an excitation signal through a digital filter [1]. The excitation signal is a product of the inverse LP filtering process of the analysed speech. In rule based synthesis (LP) this excitation signal is usually modelled as unit impulse or Rosenburg-Klatt signal [2]. Such a model is not an accurate representation of the residual signal as shown in Fig 1.0 and 1.1. We propose to use the HNM model equation 1[3] to model this residual signal, The HNM model is a better approximate of the residual signal because it accounts for both the noise and the harmonic component of the residual signal. The main advantage of such a model is that it does not confine one into analysing or modelling LP speech over a finite window length 20-40ms [3].

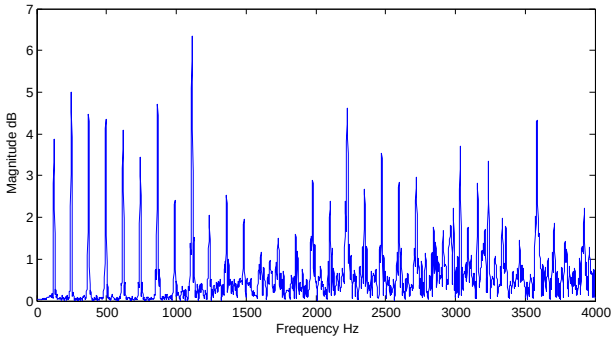


Fig 1.0 Frequency domain residual signal vowel /a/

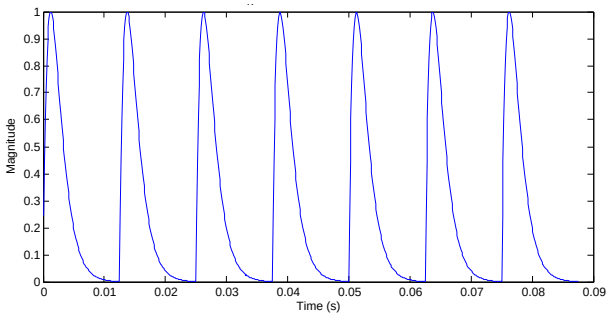


Fig 1.1 Rosenburg-Klatt residual signal waveform

The HNM equation

$$y(t) = \sum_{k=1}^{K} A_k(t) cos(k\theta(t) + \theta_k(t) + n(t) \qquad (1)$$

3. Speech synthesis a mathematical model

To reduce the complexity of the HNM model a number of mathematical formulas are proposed. The formulation is based on the frequency domain residual signal Fig 2.0.

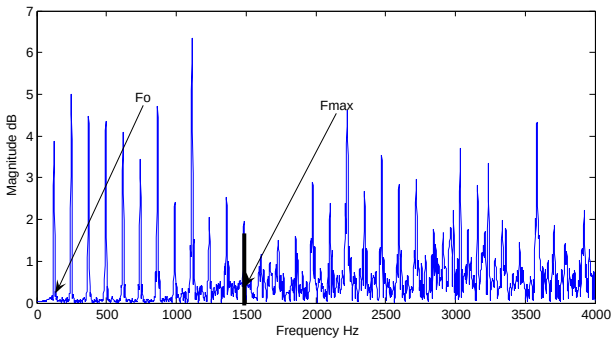


Fig 2.0 Frequency domain residual signal vowel /a/

- F_o or $\theta(t)$ is the fundamental frequency is observed as the first peak harmonic in the residual signal [4].
- $Fmax$ is the maximum voiced frequency[5].
- k is the number of harmonics [4].
- $\theta_k(t)$ is the harmonic phase. The phase in any speech is not distinguishable by the human ear therefore the formulation of this parameter is not very critical to the design process [6]. A linear phase shift across all the harmonics is therefore proposed equation 2.

$$\theta_k = (\frac{2\pi}{K})(k-1) \tag{2}$$

- $n(t)$ is the noise component.
- A(t) is the magnitude of the harmonic component at time intervals. Discovering the true value of A(t) is a complex process. A simplified mathematical formula is suggested in the next section.

3.1. Simplifying the harmonic magnitude A(t)

By observing the magnitude of the harmonic components Fig 2.0 for different sounds we discovered a similar trend on all waveforms. We therefore suggest a new way of simplifying the function $A(t)$ from a time dependent function to a Harmonic depended function $A(k)$. From Fig 2.0 we derive a scatter plot Fig 3.0 of the harmonic components in the frequency domain. We then perform a goodness of fit test [7] on the scatter plots with linear, quadratic and exponential functions. The results of these tests are tabulated in Table 1.0 and the curve fits are shown in Fig 3.0.

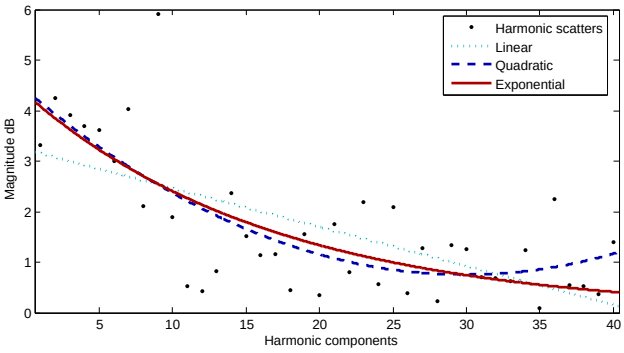


Fig 2.0 Scatter plot residual signal vowel /a/

Phoneme	Function	SSE	R-Square
/a/	linear	45.9052	0.4451
	exponential	34.4023	0.5841
	quadratic	32.9527	0.6016
/v/	linear	47.0052	0.4002
	exponential	35.2082	0.5744
	quadratic	33.1245	0.5912

Table 1.0 Goodness of fit results on phoneme residuals

From the test we formulated a function of $A(k)$ equation 3 with an approximate 60% confidence interval equation 4.

$$A[k] = A_k \exp(a_e k) \tag{3}$$

3.2. Modelling the noise component n(t)

The noise component in HNM is generally modelled as white noise. The frequency at which the noise becomes distinguishable is known as Fmax. We observed from Fig 4 that the harmonic component also contained some small noise components. We thus modelled our noise component as random signal high pass butterworth filter [8] with a passband transition region around Fmax.

3.3. Speech synthesis model

Using the derivations above, a mathematical model of our speech synthesiser can be descibed with equation 4 and 5 [1].

$$\widetilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k] + e(n) \tag{4}$$

e(n) the residual signal. Substituting e(n) with the HNM model the speech synthesis model becomes equation 5.

$$\widetilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k] + \sum_{k=1}^{K} A_k \exp(a_e k) cos(k\theta(t) + \theta_k(t) + n(t) \tag{5}$$

Fig 4 shows a typical LP residual signal for the vowel /a/ formulated with the model described above.

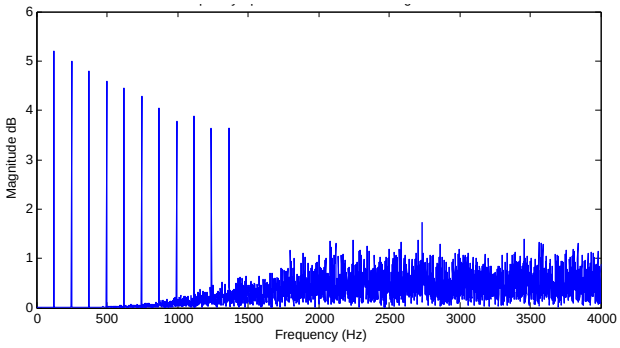


Fig 4.0 HNM based residual signal vowel /a/

4. Parametric Optimisation

Once our mathematical model had been defined we investigated ways of improving the parametric corpus by optimising the number of LP parameters and the analysis window length.

4.1. Optimising the number of LP parameters

The effects of varying the number of LP parameters on the characteristics of the residual signal for phonemes /a/, /e/ and /s/ are observed at different LP parameters. Fig 5.0 shows a typical residual signal with 10 lp parameters and the harmonic scatter plot in Fig 5.1. We performed a goodness of fit test on the residual signal plots obtained at different LP parameters. The test was used as a criteria to define the region of LP parameters for which the model proposed in section 3can be used. Table 2.0 shows the results of such a fussy test.

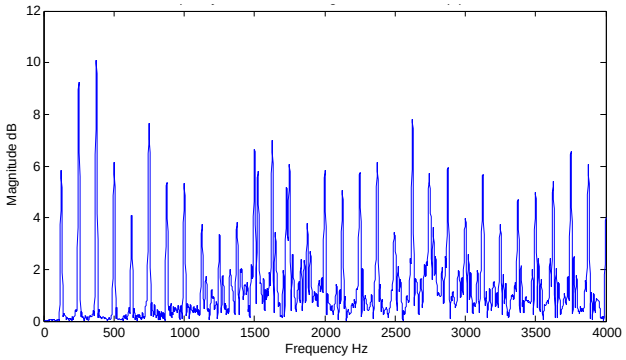


Fig 5.0 Frequency domain residual signal vowel /e/ at 10lp parameters

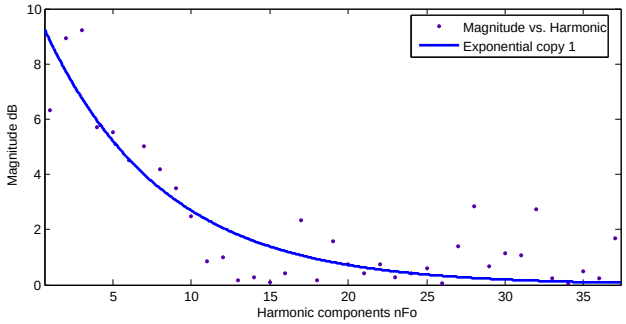


Fig 5.1 Performing a functional fit on a scatter plot for the vowel /e/

Number of LP	Function	SSE	R-Square
2	exponential	94.1021	0.1341
5	exponential	74.2347	0.3711
10	exponential	40.0040	0.5564
15	exponential	30.1034	0.6412
20	exponential	27.5259	0.7022

Table 2.0 Goodness of fit test results at different LP parameters

We observed from table 2.0 that at less than 10 LP parameters the residual signal does not fit accurately our proposed function. However at a higher number of LP parameters the residual suitably fits our model with a confidence interval of approximately 60%.

4.2. Optimising the window length

One of the main restrictions of LP based synthesis is that the analysis can only be carried out at window lengths or segments usually 30-50ms long [1]. To find the optimal window length for our modified HNM model, we conducted a goodness of fit test similar to the one in section 4.1 on residual signals for phonemes /a/, /e/ and /s/ as in Fig 6.0 with a constant number of LP parameters but variable window lengths. The results of these tests for the vowel /a/ are shown in Table 3.0.

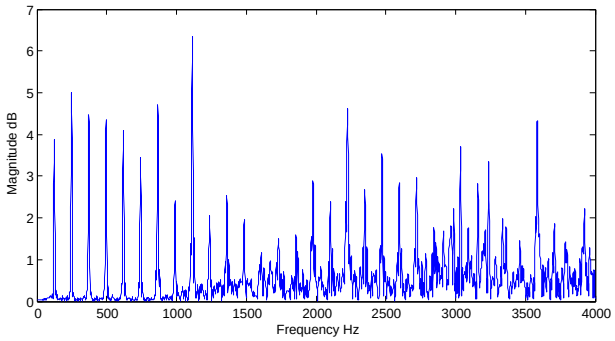


Fig 6.0 Vowel /a/ residual at 125ms

Window length	Function	SSE	R-Square
6.25ms	exponential	90.3022	0.1941
62.5ms	exponential	40.4446	0.5665
125ms	exponential	30.0040	0.6865
250ms	exponential	33.1034	0.6012
500ms	exponential	35.2082	0.5504

Table 3.0 Goodness of fit results at different window lengths

From table 3.0 we determine a region of optimal window lengths for our model around 150ms. This region is far greater than the 20-50ms used in normal LP. Fig 7.0 shows a graphical representation of the optimal window lengths defined for our model based on the tests and results above.

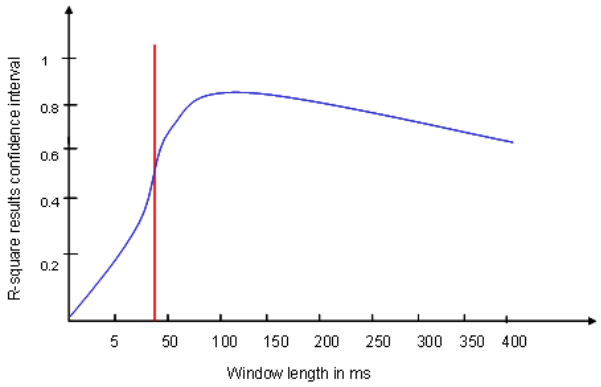


Fig 7.0 Optimisation region for variable window lengths

The optimal window length can be chosen at lengths greater than the red line.

5. Tests and Methodology

For testing purposes we constructed a parametric speech corpus based on the discussed speech model. The corpus of all 50 English phonemes contained a total of approximately 2000 parameters a footprint of approximate 2KB. Each phoneme was divided into two windows of length 250ms modelled with 15 LP parameters and 5 residual model parameters namely.

- $F_o\ \theta(t)$ the fundamental frequency.
- k the number of harmonics.
- a_n the magnitude of the noise.
- A_k Harmonic magnitude.
- a_e The exponent factor harmonic magnitude.

For testing purposes a further 14 phonetic words in the form of 4 nasals, 4 fricatives and 6 vowels words were analysed. The window lengths and the number of LP parameters were not changed. Listening tests with a sample of 20 native South African English speakers were conducted on synthesised words the accent of these people was the same with the speech used to construct the parametric corpus. Two test methods were used namely MOS and transcription tests.

5.1. Subjective quality tests / Mean Opinion Score (MOS):

Each listener from our sample was asked to give a score from 1-5 on the quality of the uttered speech generated with LP models based on the R-K, unit impulse and HNM residual signal. Table 4 shows the results of the MOS test [9].

Phonetic word	R-K based LPC	HNM based LPC
hello	3.5	4.3
world	3.8	4.0
father	3.5	4.0
act	3.3	4.3
stop	3.9	4.4

Table 4.0 MOS score results for synthesised words

5.2. Transcription tests

The same synthesised words were played to listeners, each listener was asked to re-pronounce the word he/she had just heard. Table 5 describes the transcription scores [9] from these tests.

Phonetic word	R-K based LPC	HNM based LPC
hello	96%	99%
world	88%	98%
father	86%	96%
act	90%	99%
stop	90%	98%

Table 5.0 Transcription score results for synthesised words

6. Discussion

Our model performed significantly better on MOS and transcription test scores when compared to traditional LP models. This can be attributed to the fact that the Harmonic plus niose model was used to model the residual signal. Modelling the residual with HNM reduced synthesis errors that arise with traditional R-K and unit impulse models. These errors are usually caused by the fact that pulse models assume the residual speech has one harmonic fundamental frequency.

7. Conclusion

In this paper we have discussed a speech synthesis model based on LP and HNM. We have mathematically modelled the residual signal with HNM and reduced the complexity of the model by formulating mathematical functions. The number of parameters and window lengths were optimised to reduce the speech corpus. The entire parametric speech corpus for English phonetics was less than 2KB, this is significantly small considering the quality of speech produced. The results from MOS and transcription tests showed that our model performed significantly well when compared with renowned parametric speech models. A thorough review of our mathematical model and tests with other languages is suggested for future work.

8. Acknowledgments

The authors would like to extend their gratitude to students at the University of Witwatersrand Johannesburg South Africa for volunteering to perform listening tests. Other acknowledgments go to the Electronic research group at the school of electrical and information engineering University of Witwatersrand for the project funding. Last but not least the authors would like to thank fellows from the Pattern Recognition Association of South Africa for sharing ideas in the field of speech synthesis.

9. References

[1] I.H. Witten, *Principles of Computer Speech*, Academic Press, 1982.

[2] F.J. Owens, *Signal Processing of Speech*, The Macmillan Press Ltd, 1993.

[3] Y Stylianou "On the implementation of the harmonic plus noise model for concatenative speech synthesis," *In Proceedings. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP Volume 2, Issue 2000*. pp II957 - II960 , Istanbul Turkey, 9 June 2000.

[4] S Roa, M Bennewitz, S Behnke "Fundamental frequency estimation based on pitch-scaled harmonic filtering," *In Proceedings. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP Volume: 4*. pp IV-397-IV-400, Honolulu Hawaii, 15-20 April 2007.

[5] Y Stylianou "Applying the harmonic plus noise model in concatenative speech synthesis," *IEEE Transactions on speech and audio processing, Volume 9, Issue 1*. pp 21 - 29 , January 2001.

[6] G Klompje, T.R Niesler, "A parametric monophone speech synthesis system", *In proceedings of the seventeenth annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, Parys South Africa, November 2006.

[7] Levenberg, K., "A Method for the Solution of Certain Problems in Least Squares," Quart. Appl. Math, Vol. 2, pp. 164-168, 1944.

[8] Lutovac, Miroslav D., Tosic, Dejan V., Evans, Brian L, "Filter Design for Signal Processing using MATLAB and Mathematica (in English)". New Jersey, USA: Prentice Hall (2001).

[9] Tomokiyo, L., Peterson, K., Black, A., and Lenzo, K. "Intelligibility of Machine Translation Output in Speech Synthesis", *In proceedings of the Interspeech ICSLP (2006)*. pp 2434-2437, Pittsburgh, PA, September 2006.

ADVANCEMENTS IN ASSISTIVE SPEECH TECHNOLOGY (SPEECH SYTHESISERS) FOR SUB SAHARAN AFRICA

A. Mamombe[1], B. Lacquet[2] and M. Shuma-Iwisi[3],

ABSTRACT

Speech synthesisers play an important role in assisting communication. An example is that of vocally impaired people that can use a speech synthesiser to utter words comprehendible to an ordinary person. The problem in Sub Saharan Africa is that most speech synthesisers are commercialised and are applied to renowned international languages examples include the speak and spell tool from Texas Instruments and the Microsoft speech tools. The biggest problem faced by Africa in adapting these tools is the finance, electricity to power the gadgets and literacy level requirements. The objective of the work reported in this paper was the development of an embedded speech synthesiser capable of uttering African speech; that is cheap portable and battery powered. The technique used to come up with the African based speech synthesiser was to compare the current technological trends in speech synthesis and then devise an optimal method of speech synthesis. The work discussed covers the development of a speech synthesis model/technique using a modified combination of linear prediction. The approach was taken in order to fit the whole speech synthesiser on an embedded device thereby reducing the cost and power requirements. The results obtained thus far through simulations of the model in the synthesis of *Shona* (an African language in southern Africa) vowel sounds have been encouraging. Limited resources were used thus allowing the synthesis model to fit on an embedded device. The outstanding issues n this work includes speech quality improvement. The model discussed in the paper is comparable to European based speech synthesisers when tested in terms of quality, application and the cost of constructing such a device.

[1] School of Electrical & information Engineering, University of the Witwatersrand, P Bag 3, Wits 2050, Johannesburg, South Africa. Email: a.mamombe@ee.wits.ac.za

[2] Professor and Dean, Faculty of Engineering and the Built Environment, University of the Witwatersrand, P Bag 3, Wits 2050, Johannesburg, South Africa. Email: b.lacquet@ee.wits.ac.za

[3] Lecturer, School of Electrical & information Engineering, University of the Witwatersrand, P Bag Wits 2050, Johannesburg, South Africa. Email: m.shumaiwisi@ee.wits.ac.za

Keywords: Speech synthesis, assistive speech technology, linear prediction, harmonic plus noise model

According to the United Nations "*72 percent of families with children who use sign language do not use sign language with their children (Gallaudet Research Institute, 2002); for these children, the interpreter may be the only person with whom they can communicate effectively.*"

INTRODUCTION

The paper discusses recent trends and developments in assistive speech technology with particular attention to the design of a generic speech synthesiser for sub Saharan Africa. The speech synthesiser usually forms the front-end of a text-to-speech conversion machine or an assistive speech device. Text-to-speech synthesisers are important in grammatical teaching as well as language learning. Assistive speech devices also play an important role in facilitating communication for the vocally impaired. Figure 1 is a block diagram of a typical speech synthesis system.

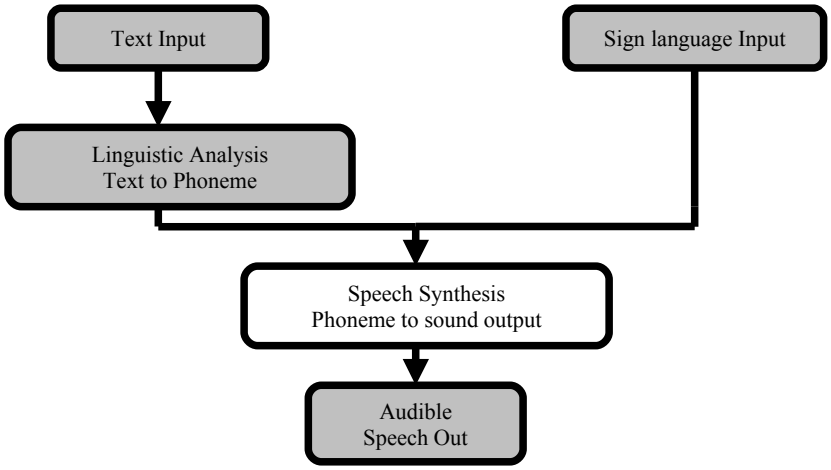


Fig.1: The block diagram of a typical speech synthesis system.

Examples of text-to-speech systems include the Talking notepad, Microsoft speech tools, the Speak n Spell toy and the Slovenian TTS system. The problems with such systems include;

- Such speech systems were built for well renowned international languages

- A relatively vast amount of computational processing power and memory is required when implementing such systems.
- The platforms for such modern speech systems include cellular phones and computers that are beyond the affordability of the rural African populous.
- Such systems are usually implemented on platforms powered by electricity; the unavailability of electricity in rural Africa creates the need for battery powered efficient and portable systems.

The objective of this paper is to discuss a novel approach to the design of a speech synthesis system. The aim of our discussion is to find a model **speech-synthesis** system that solves the problems stated above and hence can be incorporated on an assistive speech device suitable for African environments.

BACKGROUND INFORMATION

Speech Synthesis

Speech synthesis is the generation of synthetic speech defined in (F.J. Owens 1993). The generation of synthetic speech is similar to the emulation of the human auditory system. The human auditory system consists of two main components; the glottal pulse and the vocal tract. Natural speech is produced by air from the lungs passed through the glottis to produce a pulse that is filtered by the vocal tract (mouth). Synthetic speech is produced by two main methods namely *concatenative speech synthesis* and *rule based speech synthesis*.
Concatenative synthesis involves the use of previously recorded speech segments stored in a corpus joined to produce speech. Rule based synthesis methods aim to model the human production system with a source-filter method as in (T. Dutoit 1999). The later method uses less memory, is adaptable to other languages and is the basis for the development of our speech system.

Current speech synthesis systems

We describe some of the speech systems on the market. For each speech system the advantages and disadvantages are listed. The two examples discussed include the Slovenian TTS system and the Speak n Spell toy from Texas instruments.

(1) The Slovenian TTS System

The Slovenian TTS system utilises a unit selection (dictionary based) synthesis method for the Slovenian language with a reduced speech corpus database as in (J. Gros, A Mihelic, N Paveic, M ganec & S Gruden 2005).

Advantages of the system

- The system utilises a small speech database.
- The system has small memory footprint of about 2MB adaptable to most embedded systems.

Disadvantages of the system

- Most embedded chips have a memory capacity of less than 2MB.
- The system cannot be directly used for other languages.

(2) The speak and spell toy by Texas Instruments

The Speak and Spell system designed in the early 1980s used a rule based linear predictive method. The system uses an embedded microprocessor and other external user interfaces like keyboards and displays as in (I.H. Witten 1982).

Advantages

- The system is cheap at a total cost of about US50 dollars.
- An efficient data rate of 1.2kb/s is used.
- The system produces reasonably intelligible speech.

Disadvantages

- Real-time processing of speech is not possible.
- Fricatives and nasals are pronounced poorly.

OUR SPEECH SYNTHESIS SYSTEM

Our optimised African based speech synthesis system model uses optimal speech modelling techniques and testing. A generalised block diagram of the system is shown in Figure 2.

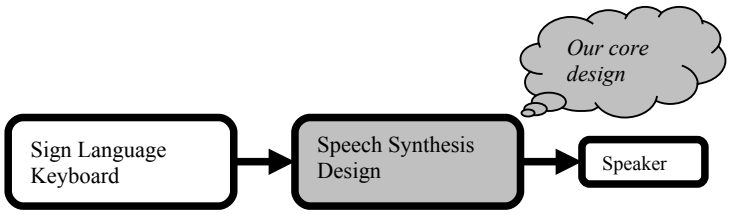


Fig. 2: General block diagram of our speech synthesis system

In the model design in order to solve some of the shortfalls stated earlier the following conditions and criterion were set during the design process.

- The system should fit on an embedded device for portability specification <500KB.
- The system was to synthesise at least one African language in this example Shona.
- Limited resources were to be used whilst maintaining high quality speech.

A system that would meet such a criterion could easily fit on embedded battery powered ARM, PIC and Motorola microprocessors.

The Speech Synthesis Model

Our speech synthesiser used a rule based speech synthesis method Linear Prediction (LP) defined in (I.H. Witten 1982). The main variation used in developing our African language based synthesiser was the use of optimised methods to improve the quality of synthetic speech produced by LP synthesiser.

Linear prediction

Linear prediction is a powerful speech processing technique used in speech synthesis, recognition and coding. Sample values of speech, x[n] are approximated as a linear combination of the past speech samples as in the equation 1 in (F.J. Owens 1993).

$$\tilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k] \qquad (1)$$

$\tilde{x}[n]$ is the predicted sample at instant n and $a_1, a_2 \ldots . a_p$ are predictor coefficients. The predicted sample is not the same as the actual sample x[n] this results in a prediction error e[n] given in equation 2 in (F.J. Owens 1993).

$$e[n] = x[n] - \tilde{x}[n] \qquad (2)$$

The problem with linear predictive methods is in the determination of the coefficients a_k, which will minimise the mean square error *e*. If the error *e* and linear prediction coefficients a_k are known, then the original speech can be reconstructed by applying the error signal to an all pole digital filter with the transfer function given in equation 3 in (F.J. Owens 1993).

$$H(z) = \frac{1}{\sum_{k=1}^{p} a_k z^{-k}} \tag{3}$$

Implementing the LP process

The first step in linear predictive speech synthesis is speech segment analysis this is performed in order to obtain LP filter parameters a_k. For our speech synthesis model, phonetic speech segments of the Shona language were analysed to obtain the LP filter parameters. For each of the 50 Shona phonemes 32 LP parameters were used. Once the filter parameters were obtained, a process of inverse filtering was used to obtain the residual signal *e(n)* or our (source signal). Figure 3 show the actual speech signal and the residual signal for the *Shona* vowel /a/ using 32 LP parameters.

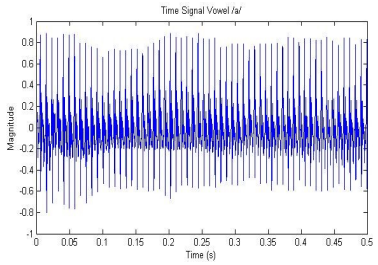


Fig. 3.1: Actual speech signal for vowel /a/

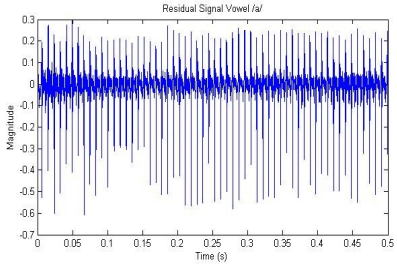


Fig. 3.2: Residual signal for vowel /a/

Once the LP parameters were obtained the residual signal (source) was modelled using the Harmonic plus Noise Model (HNM) defined in (Y Stylianou 2001). Artificial construction of the residual signal allowed significant reduction in the memory unlike the use of codebooks that store the residual signal. HNM is a speech synthesis technique based on the fact that the speech signal constitutes of two components namely the harmonic *h(t)* and noise *n(t)* component modelled by the equation 4 as in (Y Stylianou 2001).

$$h(t) = \sum_{k}^{K} A_k(t)\cos(k\theta t) + \theta_k t \tag{4}$$

n(t) can be modelled as white Gaussian noise.

Simulation procedure tests

We tested our speech synthesis through MATLAB algorithms. The simulations were performed on the three blocks in Fig. 2 namely phonetic inputs, speech synthesis and speech output.

Phonetic Inputs: The input from the keyboard was simulated in MATLAB as a different character entry into the program to perform modelling of the speech.

Speech Processing: For each phonetic input, a set of filter and excitation signal parameters obtained from LP analysis was stored in the program. Each key input selection excited a particular set of filter and excitation signal parameters from our small database. The output speech signal was produced from passing the model source signal through a filter defined by the LP filter parameters.

Speech Output: The speech signal output was stored as a wav file. The file was played using windows media and listening tests on quality audibility were conducted. The total memory and complexity of the synthesis code used in our speech synthesiser was compared to renowned models results obtained from listening tests and analysis are discussed in the next section.

PERFORMANCE RESULTS AND DISCUSSIONS

Our speech synthesis model was efficient as it used only 32 parameters for each phonetic sound together with eight source signal parameters. The processing code and parameter corpus was less than 500KB for the entire Shona speech. A memory map of less than 500KB easily fits on most embedded integrated circuits like the ARM processor and PIC microprocessor.

The most important aspect of the whole speech synthesis system was the use of optimised HNM method to synthesise speech. The resultant model produced better quality of speech than previously renowned models.

Listening and analytical tests were conducted using a sample of 10 students at the University of Witwatersrand in South Africa. Firstly the original phonetic sound used in deriving the filter parameters was played, the synthetic sound was played next and each student was asked to give his or her score on a range from 1 – 5, 1 being the poorest quality and 5 the best.

The same students were asked to give a score on a simulated model using conventional LP methods similar to those used in the Speak and Spell toy. The average score recorded from the 10 students on the quality and audibility for each phonetic sound is tabulated in Table1.

Table 1: Results of the three synthesis models

	Speak and Spell Toy		Our Method		Conatenative Synthesis	
Phoneme	Quality	Audibility	Quality	Audibility	Quality	Audibility
/a/	3.2	3.0	3.7	3.5	4.9	4.9
/e/	3.4	3.7	3.7	3.6	4.9	4.8
/i/	3.0	3.0	4.1	4.2	4.8	4.9
/o/	3.4	3.0	4.0	4.3	4.8	4.9
/u/	3.6	3.5	3.8	3.8	4.9	4.9
/r/	3.4	3.5	4.0	4.0	4.8	4.9

The results show that our system of quality comparable to commercialised systems even though it requires less memory and no electric power. Figure 4 shows an analytic comparison of the actual speech signal (blue) and the synthetic speech signal (red).

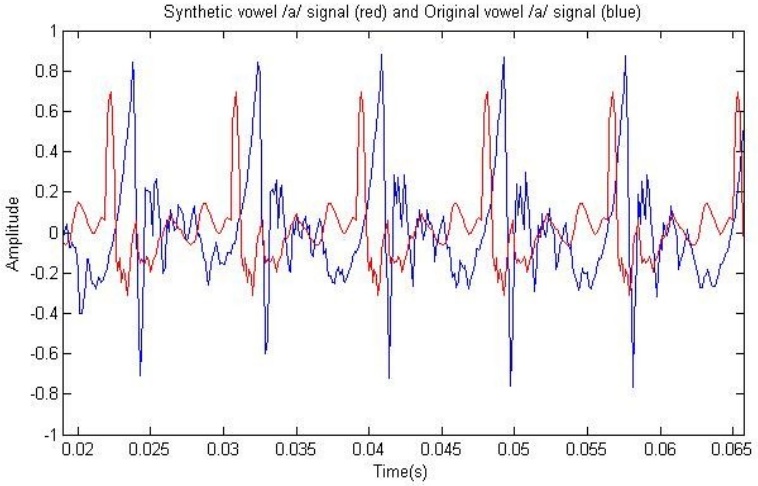


Fig. 4: An analytic comparison of the synthetic and original signal.

CONCLUSION

In the paper, we have discussed a speech synthesis model applicable for use in sub Saharan Africa. The system easily fits on most embedded processors like the PIC hence making the model affordable, battery powered and portable. The use of the rule-based approach meant that the memory requirements of the speech system were reduced. Our design rated fairly in quality in comparison

with previously renowned rule based synthesis models. We have discussed an optimised speech synthesis system that could be used in rural sub Saharan Africa for the Shona language. The model makes use of the efficiency of rule based synthesis methods whilst resulting in quality similar to that of concatenative speech synthesis.

REFERENCES

F.J. Owens 1993, *Signal Processing of Speech*, The Macmillan Press Ltd.

T. Dutoit 1999, *A Short introduction to text-to-speech synthesis,* http://tcts.fpms.ac.be/synthesis/introtts old.html [last accessed 2007-05-10]

J. Gros, A Mihelic, N Paveic, M ganec, S Gruden 2005, *Slovenian Text-to-Speech Synthesis for Speech User Interfaces,* In Proceedings of the Third World Enformatika Conference, WEC'05. pp 216-220, Istanbul Turkey.

I.H. Witten 1982, *Principles of Computer Speech,* Academic Press.

Y Stylianou 2001, *Applying the harmonic plus noise model n concatenative speech synthesis,* IEEE Transactions on speech and audio processing, Volume 9, Issue 1. pp 21 – 29.

Index

Index
audibility, 69
audicity, 67
auto-regressive, 7
blackman, 36
british, 35
cepstrum, 16
concatenative, 8
cosine, 62
diphones, 8
dipthongs, 5
embedded, 50
fbls, 17
footprint, 38
formant, 7
fpga, 50
fricatives, 41
fundamental frequency, 25
gaussian, 25
glottis, 3
goodness fit, 26
hamming, 36, 64
hanning, 36
harmonic plus noise, 15
impulse train, 21
inaudibility, 21
interpolation, 80
inventory, 10
klatt, 20
linear prediction, 13
linguistic, 5
lma, 16
lpc, 20
maclaurins, 51
matlab, 43
matrix, 75
mean opinion score, 49
methodology, 35
microprocessor, 11
monotonous, 20
parametric, 29
phase, 25
plossives, 41
prosody, 5
qos, 48
residual, 20
rosenburg, 20
spectrogram, 45
synthesis, 8
taylor, 51
transcription, 49
triphones, 8
unvoiced, 25
vhdl, 60, 62
voiced, 25
warping, 9
witwatersrand, 70
xilinx, 51