

# Text-To-Speech (TTS) Conversion for Gujarati Language

## Major Project Report

*Submitted in partial fulfillment of the requirements*

*for the degree of*

Master of Technology

in

Electronics & Communication Engineering

(Embedded Systems)

By

Jayesh B. Tanna

(12MECE23)



Electronics & Communication Engineering Branch

Department Of Electrical Engineering

Institute Of Technology

Nirma University, Ahmedabad-382481

May-2014



# Text-To-Speech (TTS) Conversion for Gujarati Language

## Major Project Report

*Submitted in partial fulfillment of the requirements  
for the degree of*

**Master of Technology  
Electronics and Communication Engineering**

Submitted by

**JAYESH B. TANNA**  
(12MECE23)

Under the Guidance of

**Prof. Vijay Savani & Prof. Amit Degada**



**Electronics & Communication Engineering Branch  
Department Of Electrical Engineering  
Institute Of Technology  
Nirma University, Ahmedabad-382481  
May-2014**





## Certificate

This is to certify that the Major Project entitled “*Text-To-Speech Conversion for Gujarati Language*” submitted by *Jayesh B. Tanna (12MECE23)*, towards the partial fulfillment of the requirements for the degree of Master of Technology in Electronics and Communication Engineering Branch of Institute of Technology, Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven’t been submitted to any other university or institution for award of any degree or diploma.

-----  
Guide

Prof. Vijay Savani

-----  
Co-Guide

Prof. Amit Degada

-----  
Dr. N.P Gajjar

PG Co-ordinator (Embedded Systems)

-----  
Dr. P.N Tekwani

Head, EE

-----  
Dr K Kotecha

Director, IT-NU

**Date:** -----

**Place:** Ahmedabad



## Acknowledgements

With immense pleasure, I would like to present this report on the dissertation work related to “Text-To-Speech (TTS) conversion for Gujarati language”. I am very thankful to my parents, all family members and all those who helped me for the successful completion of the dissertation and for providing valuable guidance throughout the project work.

I would first of all like to offer thanks to **Prof. Vijay Savani**, Guide & **Prof. Amit Degada**, Co-Guide Institute of Technology, Nirma University, Ahmedabad, whose keen interest and excellent knowledge helped me to finalize the topic of the dissertation work. Moreover, I would like to thank **Dr. Tanish Zaveri** for his constant support and interest in the subject equipped me with a great understanding of different aspects of the required architecture and resources for the project work. They have shown keen interest in this dissertation work right from beginning and has been a great motivating factor in outlining the flow of my work.

My sincere thanks and gratitude to **Prof. N.P Gajjar**, P.G Co-ordinator (Embedded Systems), Electronics and Communication Engineering Branch, Institute of Technology, Nirma University, Ahmedabad for his continual kind words of encouragement and motivation throughout the Dissertation work.

I am thankful to Nirma University for providing all kind of required resources. I would like to thank The Almighty, Family members, Guides, Mr. Prasann Shukla, especially my colleagues Darshan Limbachiya, Bhaumik Pandya, Vipul Prajapati, Snehal Surti, Devanshi Desai, Rujul Joshi, Miral Desai, Ankur Patel, Yogesh Badole and many more for supporting and encouraging me in all possible ways. I would also like to thank all the remaining persons, who have directly or indirectly helped me in making this dissertation work successful.

- Jayesh B. Tanna

12MECE23



## Abstract

Text-To-Speech (TTS) conversion is a great topic of research nowadays. This project will be very useful for the illiterate and especially for the blind people. By using this system any person can read any article and understand it.

The overall process of Text-to-Speech (TTS) conversion can be divided into mainly three blocks: Text Normalization, Text-to-Phoneme and Phoneme-to-Sound. *Text normalization* block removes the symbols and replace it with blank space and analyze all the digits and texts from the input texts. *Linguistic analysis* block provides intonation and prosody to the graphemes. And finally, *waveform generation* block will generate the original output sound.

Text-To-Speech (TTS) conversion can be classified in four ways mainly: Concatenate synthesis, Formant synthesis, Hidden Markov Model (HMM) and Articulatory synthesis. As the name suggest, *concatenative synthesis* concatenates the different words from the database of pre-recorded words and then maps each of the matched input words with its equivalent phoneme. Due to large memory requirement of this system, this synthesis technology mainly not used in embedded systems, where memory and power is the main factors of the whole system. *Formant synthesis* works very well without any kind of pre-recorded word's database. But, the drawback of this synthesis technology is that the naturalness in the output sound (robotic or not like human). *HMM* based synthesis is a synthesis method based on hidden markov models, also called Statistical Parametric Synthesis. In this system, the frequency spectrum (Vocal tract), fundamental frequency (vocal source) and duration (prosody) of speech are modeled simultaneously by HMMS. Speech waveforms are generated from HMMs themselves based on the maximum likelihood criterion *Articulatory synthesis* is an ideal synthesis technique. In which, the whole articulatory system (mouth) of human being is modeled and the sound will generate by the program. Programming point of view, this technique is very complex as compared to above methods and output is not that much accurate. Due to its complexity, this technique is rarely used for TTS system. Thus, by considering all the feasible parameters like simplicity, complexity, power, memory etc., concatenative synthesis is superior than other ones.



## Abbreviation Notation and Nomenclature

ARM	.....	Advance Risc Machine
BSP	.....	Board Support Package
CCS	.....	Code Composer Studio
DMA	.....	Digital Memory Access
DSP	.....	Digital Signal Processor
EMIFA	.....	External Memory Interface
EVM	.....	Evaluation Module
GCC	.....	GNU Compiler Collection
GEL	.....	General Extention Language
GPIO	.....	General Purpose Input Output
GPMC	.....	General Purpose Memory Controller
I2C/I2S	.....	Inter Integrated Circuit/Sound
IPA	.....	International Phonetic Alphabet
LDO	.....	Low Drop Out
McASP	.....	Multi Channel Audio Serial Port
McBSP	.....	Multi Channel Buffered Serial Port
mDDR	.....	Mobile Double Data Rate
OMAP	.....	Open Multimedia Application Platform
OTG	.....	On The Go
PHY	.....	Physical Layer
SATA	.....	Serial Advanced Technology Attachment
SDRAM	.....	Synchronous Dynamic RAM
SD Card	.....	Secure Digial Card
SOC	.....	System On Chip
SOM	.....	System On Module
TFT	.....	Thin Film Transistor
TSC	.....	Touch Screen Controller
uPP	.....	Universal Parallel Port
USB	.....	Universal Serial Bus
VLIW	.....	Very Long Instruction Word
VPIF	.....	Video Port Interface



# Contents

<b>Certificate</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Abbreviation Notation and Nomenclature</b>	<b>vi</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Background . . . . .	3
1.3 Block Diagram . . . . .	4
1.4 Report Organization . . . . .	5
<b>2 Literature Survey</b>	<b>6</b>
2.1 TTS Synthesizer Technologies . . . . .	7
2.2 Implementation of TTS using Dictionary based approach . . . . .	10
2.3 Speech Signal Representation for Concatenative Synthesis . . . . .	11
<b>3 Hardware implementation</b>	<b>12</b>
3.1 Introduction to OMAP-L138 . . . . .	12
3.2 Block Diagram . . . . .	12
3.2.1 DSP Subsystem . . . . .	13
3.2.2 ARM Subsystem . . . . .	13
3.2.3 DMA Subsystem . . . . .	13
3.3 ARM Subsystem . . . . .	14
3.3.1 Introduction . . . . .	14
3.3.2 Operating States/Modes . . . . .	15
3.3.3 Co-Processor 15 (CP15) . . . . .	16
3.3.4 Addresses in an ARM926EJ-S System . . . . .	16
3.3.5 Memory Management Unit (MMU) . . . . .	17
3.3.6 Caches and Write Buffer . . . . .	17
3.4 DSP System . . . . .	19
3.4.1 TMS320C674x Megamodule . . . . .	19
3.4.2 Internal Memory Controllers . . . . .	20
3.4.3 Internal Peripherals . . . . .	21



3.4.4	Advanced Event Triggering (AET) . . . . .	21
3.5	Steps for connecting the OMAP-L138 eXperimenter kit to CCSv5 . . .	22
<b>4</b>	<b>Behavioral Simulation</b>	<b>27</b>
4.1	Objective . . . . .	27
4.2	Simulation results for typed input . . . . .	27
4.3	Simulation results for scanned input . . . . .	28
4.3.1	Simulation results for Histogram method . . . . .	28
4.3.2	Simulation results for Edge detection algorithm method . . . . .	29
4.4	Festival - Open source tool for TTS synthesis . . . . .	29
4.4.1	Installing Festival(Windows) . . . . .	29
4.4.2	Starting Festival . . . . .	30
<b>5</b>	<b>Proposed Method of TTS for Gujarati language</b>	<b>32</b>
5.1	Recognizing scanned Input by Histogram . . . . .	33
5.1.1	Challenges for recognizing a character for Histogram method . .	34
5.2	Recognizing scanned Input by Edge detection algorithm . . . . .	34
<b>6</b>	<b>Conclusion and Future Scope</b>	<b>36</b>
	<b>Reference</b>	<b>38</b>
	<b>List of Publications</b>	<b>39</b>
<b>A</b>	<b>MATLAB code for TTS with typed Input text</b>	<b>40</b>
A.1	Dictionry based TTS . . . . .	40
A.1.1	Main code . . . . .	40
A.1.2	Script file for recording each word . . . . .	42
<b>B</b>	<b>MATLAB code for TTS with scanned Input text</b>	<b>44</b>
B.1	Histogram method . . . . .	44
B.2	Edge detection algorithm . . . . .	45



# List of Tables

2.1	Comparison of Different Synthesis Technologies . . . . .	9
3.1	Different Address Types in ARM System . . . . .	16



# List of Figures

1.1	Block Diagram of TTS . . . . .	4
2.1	Classification of Synthesizer Technologies . . . . .	7
3.1	OMAP-L138 Applications Processor Functional Block Diagram . . . . .	13
3.2	TMS320C674x Megamodule Block Diagram . . . . .	20
3.3	Create New Target Configuration File . . . . .	22
3.4	Location and Extention of Target Configuration File . . . . .	23
3.5	Basic Configuration Parameters for Target Configuration File . . . . .	24
3.6	Advance Configuration parameters for Target Configuration File . . . . .	25
3.7	Launching of Selected Target Configuration File . . . . .	26
4.1	Simulation inputs for TTS using word based concatenative synthesis . . . . .	28
4.2	Histogram of Gujarati characters ‘અ’ and ‘ઉ’ . . . . .	28
4.3	Character recognition by Edge detection algorithm . . . . .	29
4.4	Terminal of Festival . . . . .	30
4.5	Getting waveform out of Festival . . . . .	31
5.1	Flow chart for recognizing scanned Input by Histogram . . . . .	33
5.2	Flow chart for recognizing scanned Input by Edge detection algorithms . . . . .	35



# Chapter 1

## Introduction

The fundamental thing which has to be performed by text-to speech system is that it has to generate a clear linguistic sound. This process of generating artificial sound is known as speech synthesis and the system, which can be used for this purpose is known as speech synthesizer. This can be implemented by using software also and hardware too as per our requirement and resources available. We are going to implement on software (MATLAB) first and afterwards we will implement a standalone device for this system.

A Text-to-Speech system converts the normal language text into phonetic speech. Phone is the smallest unit of sound in any language and phonetics is the branch of acoustics concerned with speech process including its production and perception. There are basically two parts in TTS system: Front end and Back end. Front end converts the input text, which contains symbols, numbers and abbreviations into equivalent pronunciation of that words. This process is known as text normalization. Back end performs basically two operations: Text-to-Phoneme conversion and Phoneme-to-Sound conversion. The process of mapping each words to their equivalent phones is known as Text-to-Phoneme conversion. The process of mapping each phoneme to its equivalent sound is known as phoneme-to-sound conversion[1].

There are two main characteristics for measuring the performance of any TTS systems: Naturalness, Intelligibility and Comprehensibility. Naturalness shows that how closely the TTS system generated sound seems like human speech while Intelligibility is, how easily the TTS understand the input text and generate the sound. Compre-



hensibility has to do with the clarity of the generated speech. Thus, the ideal speech synthesizer should have all the characteristics naturalness, intelligibility and Comprehensibility. So, our main goal is to maximize & get all the characteristics of TTS at the level of acceptance. [2]

## 1.1 Motivation

The grapheme-to-phoneme (G2P) conversion module is one of the most important modules in the TTS systems. Smallest unit in the scripts and text is known as *Grapheme* and the basic unit of speech is known as *Phoneme*. We get the pronunciation for a given word through this G2P module. The pronunciation of any words can be achieved by *Dictionary* (manually) based and *Rule-based* or statistical methods. For any language, manually developing a pronunciation dictionary is tiresome and it is almost next to impossible to create dictionary due to presence of huge number or unique words. But this method is the most convenient method for developing basic TTS system. For almost all Indian languages, there is one-to-one correspondence between letters (graphs) and sounds (phones). Hence, creating a rule-based pronunciation generator for Indian languages is easier. Unfortunately, even it poses a challenge to build a minimal set of rules for handling out-of-vocabulary words. Moreover, a rule always yields numerous exceptions due to the complexity of a language and vastness of a vocabulary[5]. But, such exceptions are rare in nature, so rule-based method is the optimal solution for handling out-of-vocabulary.

Text-to-Speech conversion system enables user to enter text in Gujarati language and as an output it generates the equivalent sound. This type of system will be greatly useful for an illiterate and vision impaired people to hear and understand the content. TTS systems are still suffering with the problem of producing emotional speech like human being. Scientist are trying to give emotions and feelings to it. This shows that, research work can be done to enhance the efficiency of TTS system.

There are TTS systems, which are under development for many languages other than Gujarati language. So, we are proposing the TTS system, which will work on Gujarati language. The input of our desired system is typed or scanned Gujarati text



and equivalent Gujarati speech with smooth flow will be generated as an output. We are trying to add a new feature through which, we can hear our own voice by superimposing our own voice frequency on the pre-recorded synthesized speech, so that we can listen any text in our own voice. This paper starts with the introduction to the fundamental concepts of TTS synthesis. So, it will be useful for the readers who are less familiar in this area of research.

## 1.2 Background

TTS systems are mainly useful for blind people who cannot read anything. So, this TTS system will be their great companion to gain some knowledge from any book or magazine or even from a news paper. Below are some applications and their significance of TTS system described :

- Text-to-Speech conversion is mainly useful for the blind people or physically handicapped people to communicate with others.
- Stephen hawking is the famous personality, who is paralyzed for decades by Lau Gehrigs disease.
- For making computer or any electronics device conversational TTS is very useful.
- Currently, braille language is available for the blind people to read something, but to convey the information in braille language is very difficult and is equivalent to 4 pages as compared to normal 1 A4 size page in English language. So, very few books are available in braille language. If they want to read the normal printed book than TTS system can be useful.
- If someone doesn't have time to read the e-mails, so they listen them directly or by recording them in .mp3 or other file formats[3].
- Continuous and repetitive announcement for the passengers at the airport, railway stations etc.



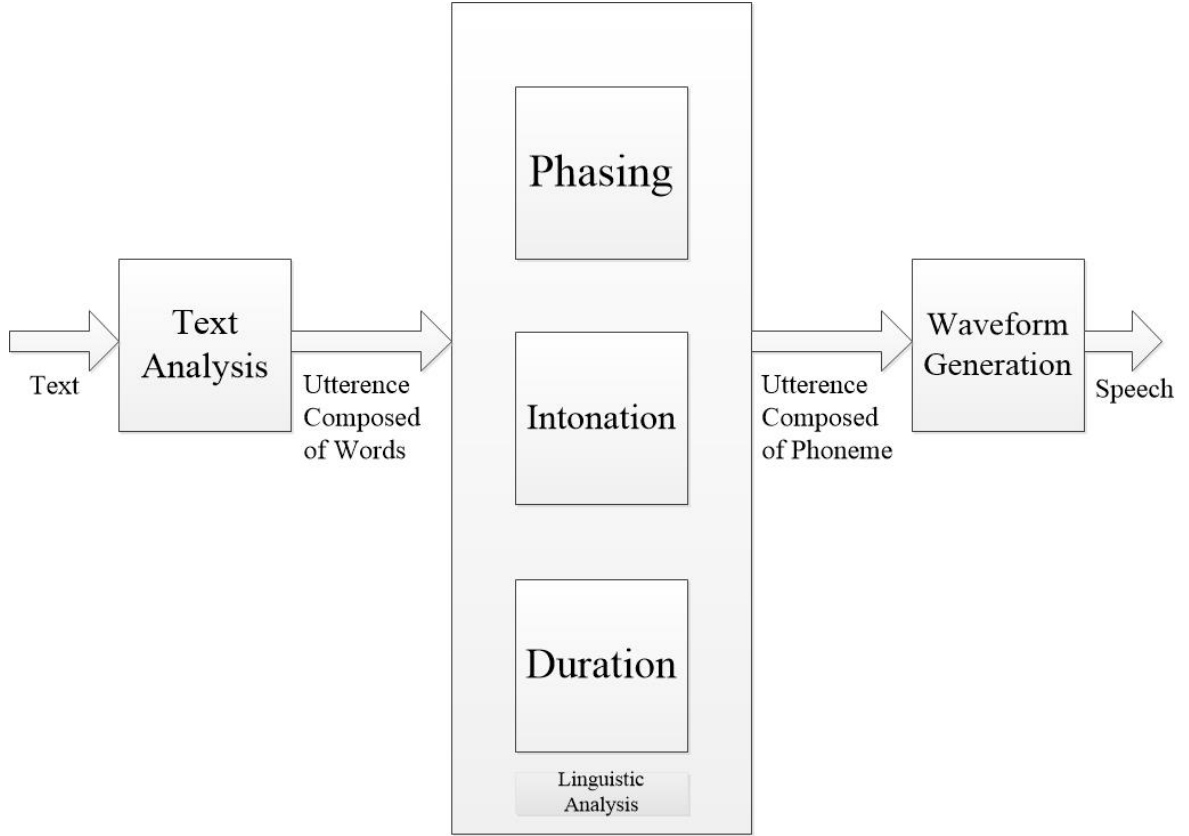


Figure 1.1: Block Diagram of TTS

### 1.3 Block Diagram

The overview of the TTS system is shown in the fig.1.1. After having glance on this block diagram, anyone can understand the working and function of TTS system. In our TTS system, scanned or typed text is applied as an input and at the output we get sound/pronunciation/speech equivalent to the related words. The heart of the system linguistic analysis. This part is purely responsible for the pitch and intonation of the output speech. To build a very good linguistic analysis module is the biggest challenge for the researchers till now, though research is going on from last many years. Because of this limitation of linguistic module accuracy, we cannot generate pure natural sound as human being from the TTS system. Researchers are trying to give emotions and feelings in the generated speech by TTS systems.

The main function of text analysis block is to perform text normalization, which deals with symbols, numbers and abbreviations. For example, if in Gujarati language



is written,

## 1.4 Report Organization

**Chapter 1**, *Introduction*, gives the brief idea about what is text to speech conversion, flow of how text is converted into speech and the main characteristics of measuring the performance of any TTS systems. In addition, block diagram shows the different modules of the TTS system and function of each and every module.

**Chapter 2**, *Literature Survey*, describes about the different types of synthesizer technologies and comparison in terms of pros and cons shows in the form of table.

**Chapter 3**, *Open Multimedia Application Platform (OMAP)- L138*, is a latest application processor by Texas Instruments, which contains both ARM and DSP processor in a single core. Thus, we can use ARM processor for general processing and system control and DSP for efficiently handle communication and audio processing tasks.

**Chapter 4**, *Behavioral simulation*, describes the different methods of TTS with its simulation results.

**Chapter 5**, *Proposed method*, describes the methods, we proposed. It includes typed & scanned input for TTS system and two sub methods for scanned input method.

Finally, in **chapter 6** concluding remarks regarding this project has mentioned.



# Chapter 2

## Literature Survey

Speech synthesis is the automatic generation of an artificial speech signal by a computer. A Text-To-Speech (TTS) system accepts an input from the user and produces the synthesized speech. The first stage of the synthesis, traditionally referred to as grapheme-to-phoneme conversion, consists of translating a written utterance into the corresponding stream of phonemes (including, for some languages, the encoding of lengthened phonemes, of lexically stressed syllables, of syllable boundaries, etc.). The second stage consists of computing a series of prosodic markers to be attached to this phonemic string. The last stage deals with the actual production of the speech waveforms. Speech synthesis can be viewed as a chain: the output quality depends on the quality of individual components. It thus makes sense to conduct a specific evaluation for each part of this chain.

Simple TTS systems accomplish this by concatenating phonemes or diphones in their database. Alternatively, a synthesizer can incorporate a speech production model specifically a model of the vocal tract to create synthetic voice output. A model of the human vocal tract considers speech to be the output of a linear all-pole filter excited by a periodic impulse train, an impulse source, white noise or their combination.

Now, I will introduce you with different types of Text-To-Speech synthesizer technologies, which are generally being used nowadays.



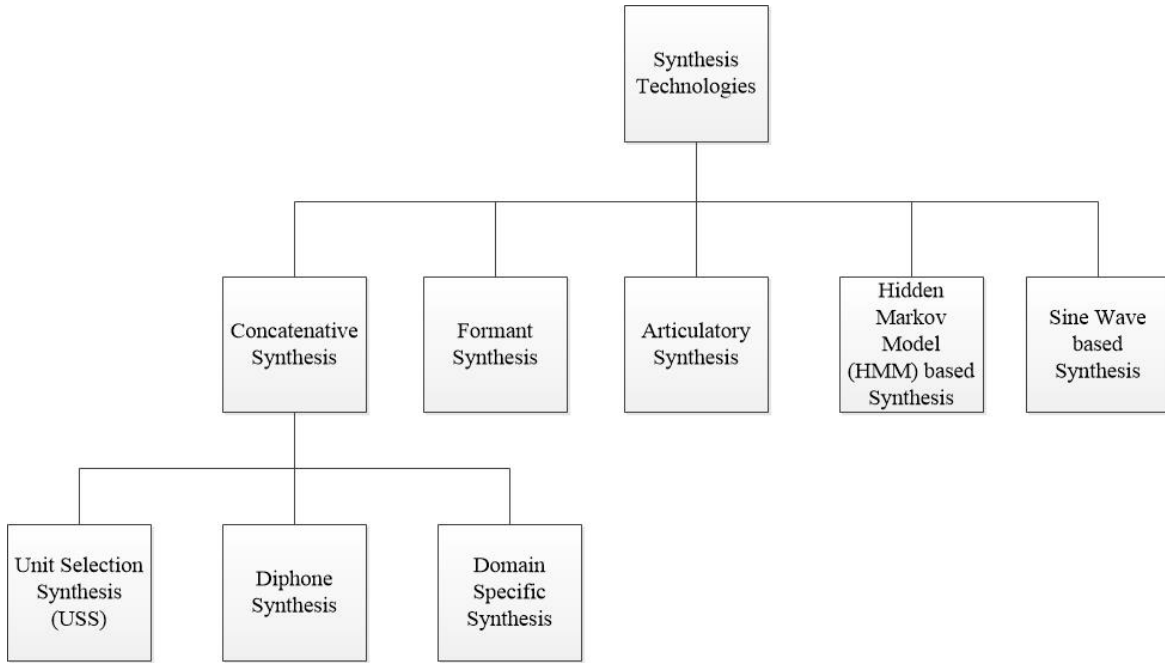


Figure 2.1: Classification of Synthesizer Technologies

## 2.1 TTS Synthesizer Technologies

Fig.2.1 shows the classification of TTS synthesizer technology. Mainly there are five types of TTS synthesizer technologies:

- *Concatenative Synthesis* is widely accepted and used to develop any language dependent TTS system. It uses the pre-recorded sounds of each and every words and maps them with the equivalent input text words. These type requires very large memory space for storing all the pre-recorded sounds of each and every words, may be in terms of giga bytes (GBs). This technique gives the most natural sound in the output of TTS system because the recorded clips are already in voice of any person.[4]

Concatenative synthesis technique is again classifie in three techniques as follows:

- ◊ *Unit Selection Synthesis(USS)* uses large database of pre-recorded speech. The smallest recorded segment in the database is known as ‘Unit’and which can be a phoneme, diphone, syllables, word or even a sentence. After applying text as an input, system search out for the best candidate unit (which is done by giving each units a unique index) and concatenates at the output to generate full natural speech.



- ◇ *Diphone Synthesis* has the database of all the diphones in a language which is much more less than that of USS. The number of diphones depends upon the language itself. This method is suffers from glitches of concatenative synthesis. But it requires less memory space than the USS.
  - ◇ *Domain Specific Synthesis* concatenates the pre-recorded words or sentences from the database to create complete utterances of the input text. This technique is used where limited number of words or sentences are used. E.g. At Airport, railway station or Bus Rapid Transport Service (BRTS), where repeatedly instructions has to be announced with limited words and sentences. The level of naturalness in this technique is very high because of limited number of pre-recorded speech.[4]
- *Formant (A main band of frequency) Synthesis* produce the robotic speech and does not use human speech samples at runtime. Thus it requires very less memory and power, so this technique can be used in embedded systems, where memory and power are big constraints. The drawback of this system is the complexity and lower level of naturalness. This technique is reliable and enough intelligent to work at higher speeds too. This type of synthesizer can be created by smaller programs than concatenative systems because the absence of the pre-recorded database of speech. [4]
- *Articulatory Synthesis* is the technique, in which human articulatory system is being modeled (jaw, tongue, teethes etc.) and simulate how the airflow passes through them and produce the sound as per our requirement. This system is not used due to its very high complexity level and low performance and efficiency level. Memory requirement is almost nothing for this type of technique[6].
- *Hidden Markov Model (HMM) based Synthesis* is also referred to as a Statistical Parametric Synthesis. In this technique, frequency spectrum (vocal tract), fundamental frequency (vocal source) and duration (prosody) of speech are modeled simultaneously by HMMs. Speech waveforms are generated from these models based on maximum probability criterion. This technique consumes large CPU resource but very little memory. In addition, this technique gives better prosody without glitches and still produces the most natural sound.[7]



- In *Sine wave based synthesis*, formants (main bands of energy) of the synthesized speech is replaced by the pure tone whistles (sine waves with specific frequency) to produce the synthesized speech. This technique is generally used as an internal module of another device because as a standalone unit, it cannot produce human understandable sound.

Table 2.1: Comparison of Different Synthesis Technologies [4, 6, 7]

Synthesis Technologies	Pros	Cons
Concatenative Synthesis	Most natural sound, Almost 100% efficiency if sound of word is stored in the database	Large speech database, Tedious method, Time consuming method
Unit Selection Synthesis (USS)	Highest accuracy and most natural sound	Large database of different units like phoneme, diphone, syllable etc.
Diphone Synthesis	Minimal speech database containing all the diphones occurring in a language	Lesser accuracy and naturalness than USS, Suffers from glitches
Domain Specific Synthesis	Fast responsive and natural, less memory requirement due to predecided words	Application is only that particular domain
Formant Synthesis	Least Memory and power requirement, Used in embedded system, Efficiency is 70-75%, Program size is smaller	Robotic sound in output, very low naturalness
Articulatory Synthesis	Memory requirement is almost nothing	Computation intensive (High complexity)

*Continued on next page*



Table 2.1 – *Continued from previous page*

Synthesis Technologies	Pros	Cons
Hidden Markov Model (HMM) based Synthesis	Very little memory, sound with better prosody, produce natural sound, Efficiency more than 90%	Consumes large CPU resource, Complex in structure
Sine-wave based Synthesis	Replace formants with pure tone whistles	Rarely used

## 2.2 Implementation of TTS using Dictionary based approach

The main goal for using the manual pronunciation dictionary in TTS systems is to achieve 100% accuracy for grapheme-to-phoneme conversion for the most common words for any language. It has been observed that the manual pronunciation dictionary may also have inflected words in it. It has been observed that the manual pronunciation dictionary may also have inflected words in it. This redundancy can be removed by the sub-word method[10].

The main goal of the sub word method in the manual approach is to reduce the dictionary size without affecting the dictionary coverage. In this approach, store only the main parts, known as sub-words in the dictionary. All possible inflections due to suffix for the word in a separate suffix dictionary. The steps for implementing the sub-word method for manual G2P conversion approach discussed in below points:

- All possible suffixes (selected sub words) have been collected from the unique words.
- A manual dictionary has been built by inserting only the unique main parts of all the stored words. Another suffix dictionary has been built with all the suffixes present in the words.
- The size for both the dictionaries has been determined and compared with the



dictionary having stored unique words.

## 2.3 Speech Signal Representation for Concatenative Synthesis

A good speech signal representation for concatenative synthesis approximates the following set of requirements:

- The speech signal can be stored in a highly compressed (i.e., coded) form so that a large voice database can be used even under tight memory limitations. Coder and decoder are of low computational complexity.
- Coding algorithms have to allow for “random access”. Since most speech coders contain some sort of autoregressive memory, all state variables of the coder have to be made available at concatenation points since the decoder will have to switch between units of speech that are very unlikely to have been recorded consecutively in time.
- An ideal speech representation must allow for natural-sounding modifications of pitch, duration, and amplitude. Unfortunately, experience shows that, for most signal processing algorithms, modifying pitch more than a few percent may destroy perceived naturalness; that is, a pitch-modified speech signal is likely to be perceptually much different from a speech signal that has been recorded without modifications from the speaker producing the desired pitch value directly. (This is the reason why “singing TTS” does not sound like an opera star.) For some advanced applications, it even might be desirable to allow for fine-tuning of the voice, for example, to add more aspiration, mellowness, or let the voice “scream” when needed. Instead of recording different voice inventories for different speaking “styles”, advanced “voice conversion” might be used to approximate an “angry” voice using a “happy” (or “neutral”) voice as a starting point. Today, algorithms for voice conversion (usually concerned with converting the speech of one speaker to sound like speech from another speaker) still do not produce consistently good enough results for sounding like the “real thing”, but might be sufficient for applications such as computer games where even the original voice does not sound “human”.



# Chapter 3

## Hardware implementation

Text-To-Speech (TTS) conversion for Gujarati language project can be implemented on *OMAP-L138* (Open Multimedia Application Platform). The main reason behind using this board is it is cost effective and low power consumption device.

### 3.1 Introduction to OMAP-L138

The OMAP-L138 Applications Processor contains two primary CPU cores: an *ARM RISC CPU* for general-purpose processing and systems control; and a powerful *DSP* to efficiently handle communication and audio processing tasks. The OMAP-L138 Applications Processor consists of the following primary components:

- ARM subsystem and associated memories
- DSP subsystem and associated memories
- A set of I/O peripherals
- A powerful DMA subsystem and SDRAM EMIF interface

### 3.2 Block Diagram

A block diagram for the OMAP-L138 Applications Processor is shown in fig.3.1.



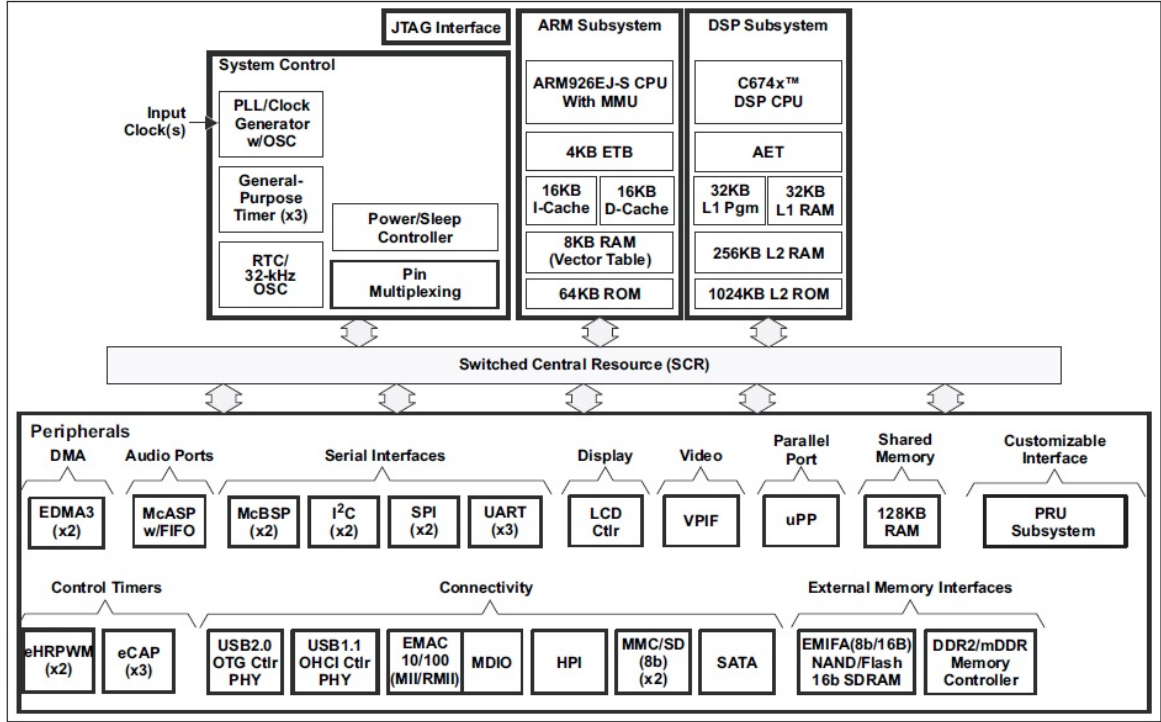


Figure 3.1: OMAP-L138 Applications Processor Functional Block Diagram [8]

### 3.2.1 DSP Subsystem

The DSP subsystem (DSPSS) includes TI's standard TMS320C674x megamodule and several blocks of internal memory (L1P, L1D, and L2). The DSP Subsystem chapter describes the DSPSS components.

### 3.2.2 ARM Subsystem

The ARM926EJ-S 32-bit RISC CPU in the ARM subsystem (ARMSS) acts as the overall system controller. The ARM CPU performs general system control tasks, such as system initialization, configuration, power management, user interface, and user command implementation. The ARM Subsystem chapter describes the ARMSS components and system control functions that the ARM core performs.

### 3.2.3 DMA Subsystem

The DMA subsystem includes two instances of the enhanced DMA controller (EDMA3).



## 3.3 ARM Subsystem

### 3.3.1 Introduction

Following section will describes about ARM subsystems and its associated memories.

The ARM subsystem consists of the following components:

- ARM926EJ-S 32-bit RISC CPU
- 16-KB Instruction cache
- 16-KB Data cache
- Memory Management Unit (MMU)
- Co-Processor 15 (CP15) to control MMU, cache, etc.
- Jazelle Java Accelerator
- ARM Internal Memory
  - ◊ 8 KB RAM
  - ◊ 64 KB built-in ROM
- Embedded Trace Module and Embedded Trace Buffer (ETM/ETB)
- Features:
  - ◊ The main write buffer has a 16-word data buffer and a 4-address buffer
  - ◊ Support for 32-bit ARM/16-bit THUMB instruction sets
  - ◊ Fixed little-endian memory format
  - ◊ Enhanced DSP instructions

The ARM926EJ-S processor is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S processor targets multi-tasking applications where full memory management, high performance, low die size, and low power are all important.

The ARM926EJ-S processor supports the 32-bit ARM and the 16-bit THUMB instruction sets, enabling you to trade off between high performance and high code density.



This includes features for efficient execution of Java byte codes and providing Java performance similar to Just in Time (JIT) Java interpreter without associated code overhead.

The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debugging. The ARM926EJ-S processor has a Harvard architecture and provides a complete high performance subsystem, including the following:

- An ARM926EJ-S integer core
- A Memory Management Unit (MMU)
- Separate instruction and data Advanced Microcontroller Bus Architecture (AMBA) Advanced High Performance Bus (AHB) bus interfaces

The ARM926EJ-S processor implements ARM architecture version 5TEJ.

The ARM926EJ-S core includes NEON signal processing extensions to enhance 16-bit fixed-point performance using a single-cycle 32 × 16 multiply-accumulate (MAC) unit. The ARM core also has 8 KB RAM (typically used for vector table) and 64 KB ROM (for boot images) associated with it. The RAM/ROM locations are not accessible by the DSP or any other master peripherals. Furthermore, the ARM has DMA and CFG bus master ports via the AHB interface.

### 3.3.2 Operating States/Modes

The ARM can operate in two states: ARM (32-bit) mode and THUMB (16-bit) mode. You can switch the ARM926EJ-S processor between ARM mode and THUMB mode using the BX instruction.

The ARM can operate in the following modes:

- User mode (USR): Non-privileged mode, usually for the execution of most application programs.
- Fast interrupt mode (FIQ): Fast interrupt processing
- Interrupt mode (IRQ): Normal interrupt processing
- Supervisor mode (SVC): Protected mode of execution for operating systems



- Abort mode (ABT): Mode of execution after a data abort or a pre-fetch abort
- System mode (SYS): Privileged mode of execution for operating systems
- Undefined mode (UND): Executing an undefined instruction causes the ARM to enter undefined mode.

You can only enter privileged modes (system or supervisor) from other privileged modes.

To enter supervisor mode from user mode, generate a software interrupt (SWI). An IRQ interrupt causes the processor to enter the IRQ mode. An FIQ interrupt causes the processor to enter the FIQ mode.

Different stacks must be set up for different modes. The stack pointer (SP) automatically changes to the SP of the mode that was entered.

### 3.3.3 Co-Processor 15 (CP15)

The system control coprocessor (CP15) is used to configure and control instruction and data caches, Tightly-Coupled Memories (TCMs), Memory Management Units (MMUs), and many system functions. The CP15 registers are only accessible with MRC and MCR instructions by the ARM in a privileged mode like supervisor mode or system mode.

### 3.3.4 Addresses in an ARM926EJ-S System

Three different types of addresses exist in an ARM926EJ-S system. They are listed in table 3.1

Table 3.1: Different Address Types in ARM System [8]

Domain	ARM9EJ-S	Caches and MMU	TCM and AMBA Bus
Address type	Virtual Address (VA)	Modified Virtual Address (MVA)	Physical Address (PA)



### 3.3.5 Memory Management Unit (MMU)

The ARM926EJ-S MMU provides virtual memory features required by operating systems such as SymbianOS, WindowsCE, and Linux. A single set of two level page tables stored in main memory controls the address translation, permission checks, and memory region attributes for both data and instruction accesses. The MMU uses a single unified Translation Lookaside Buffer (TLB) to cache the information held in the page tables.

The MMU features are as follows:

- Standard ARM architecture v4 and v5 MMU mapping sizes, domains, and access protection scheme.
- Mapping sizes are 1 MB (sections), 64 KB (large pages), 4 KB (small pages) and 1 KB (tiny pages)
- Access permissions for large pages and small pages can be specified separately for each quarter of the page (subpage permissions)
- Hardware page table walks
- Invalidate entire TLB, using CP15 register 8
- Invalidate TLB entry, selected by MVA, using CP15 register 8
- Lockdown of TLB entries, using CP15 register 10

### 3.3.6 Caches and Write Buffer

The ARM926EJ-S processor includes:

- An Instruction cache (Icache)
- A Data cache (Dcache)
- A write buffer

The size of the data cache is 16 KB, instruction cache is 16 KB, and write buffer is 17 bytes. The caches have the following features:

- Virtual index, virtual tag, addressed using the Modified Virtual Address (MVA)



- Four-way set associative, with a cache line length of eight words per line (32 bytes per line), and two dirty bits in the Dcache
- Dcache supports write-through and write-back (or copy back) cache operation, selected by memory region using the C and B bits in the MMU translation tables
- Perform critical-word first cache refilling
- Cache lockdown registers enable control over which cache ways are used for allocation on a line fill, providing a mechanism for both lockdown and controlling cache pollution.
- Dcache stores the Physical Address TAG (PA TAG) corresponding to each Dcache entry in the TAGRAM for use during the cache line write-backs, in addition to the Virtual Address TAG stored in the TAG RAM. This means that the MMU is not involved in Dcache write-back operations, removing the possibility of TLB misses related to the write-back address.
- Cache maintenance operations to provide efficient invalidation of the following:
  - ◊ The entire Dcache or Icache
  - ◊ Regions of the Dcache or Icache
  - ◊ The entire Dcache
  - ◊ Regions of virtual memory
- They also provide operations for efficient cleaning and invalidation of the following:
  - ◊ The entire Dcache
  - ◊ Regions of the Dcache
  - ◊ Regions of virtual memory

The write buffer is used for all writes to a non-cachable bufferable region, write-through region, and write misses to a write-back region. A separate buffer is incorporated in the Dcache for holding write-back for cache line evictions or cleaning of dirty cache lines.

The main write buffer has a 16-word data buffer and a four-address buffer.



The Dcache write-back has eight data word entries and a single address entry.

The MCR drain write buffer enables both write buffers to be drained under software control.

The MCR wait for interrupt causes both write buffers to be drained and the ARM926EJ-S processor to be put into a low power state until an interrupt occurs.

## 3.4 DSP System

The DSP subsystem shown in fig.3.2 includes TI's standard TMS320C674x megamodule and several blocks of internal memory (L1P, L1D, and L2). This chapter provides an overview of the DSP subsystem and the following considerations associated with it:

- Memory mapping
- Interrupts
- Power management

### 3.4.1 TMS320C674x Megamodule

The C674x megamodule shown in fig.3.2 consists of the following components:

- TMS320C674x CPU
- Internal memory controllers:
  - ◊ Level 1 program memory controller (PMC)
  - ◊ Level 1 data memory controller (DMC)
  - ◊ Level 2 unified memory controller (UMC)
  - ◊ Extended memory controller (EMC)
  - ◊ Internal direct memory access (IDMA) controller
- Internal peripherals:
  - ◊ Interrupt controller (INTC)
  - ◊ Power-down controller (PDC)
  - ◊ Bandwidth manager (BWM)
- Advanced event triggering (AET)



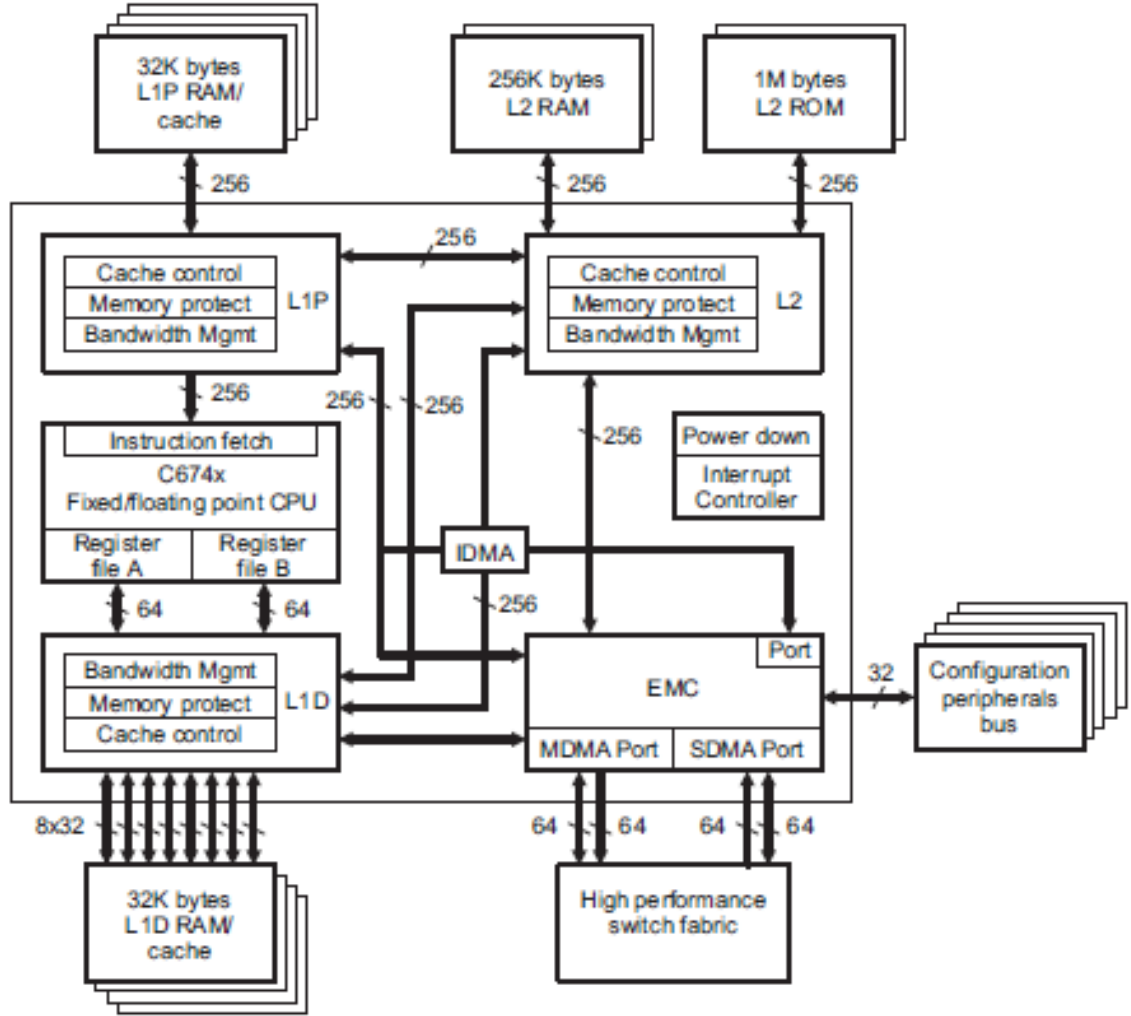


Figure 3.2: TMS320C674x Megamodule Block Diagram  
[8]

### 3.4.2 Internal Memory Controllers

The C674x megamodule implements a two-level internal cache-based memory architecture with external memory support. Level 1 memory (L1) is split into separate program memory (L1P memory) and data memory (L1D memory). L1 memory is accessible to the CPU without stalls. Level 2 memory (L2) can also be split into L2 RAM (normal addressable on-chip memory) and L2 cache for caching external memory locations. The internal direct memory access controller (IDMA) manages DMA among the L1P, L1D, and L2 memories.



### 3.4.3 Internal Peripherals

The C674x megamodule includes the following internal peripherals:

- DSP interrupt controller (INTC)
- DSP power-down controller (PDC)
- Bandwidth manager (BWM)
- Internal DMA (IDMA) controller

### 3.4.4 Advanced Event Triggering (AET)

The C674x megamodule supports advanced event triggering (AET). This capability can be used to debug complex problems as well as understand performance characteristics of user applications. AET provides the following capabilities:

- Hardware Program Breakpoints: specify addresses or address ranges that can generate events such as halting the processor or triggering the trace capture.
- Data Watchpoints: specify data variable addresses, address ranges, or data values that can generate events such as halting the processor or triggering the trace capture.
- Counters: count the occurrence of an event or cycles for performance monitoring.
- State Sequencing: allows combinations of hardware program breakpoints and data watch points to precisely generate events for complex sequences.



## 3.5 Steps for connecting the OMAP-L138 eXpérimental kit to CCSv5

First, Install CCSv5 and activate the licence key. If you are using it for personal use than, ask for free licence from TI's website. After that, for using OMAP-L138 kit with the CCS for the first time, you must perform the chain of steps. These steps are as following:

1. Goto 'View -> Target Configurations' for createing new target configuration file for our OMAP-L138 board.

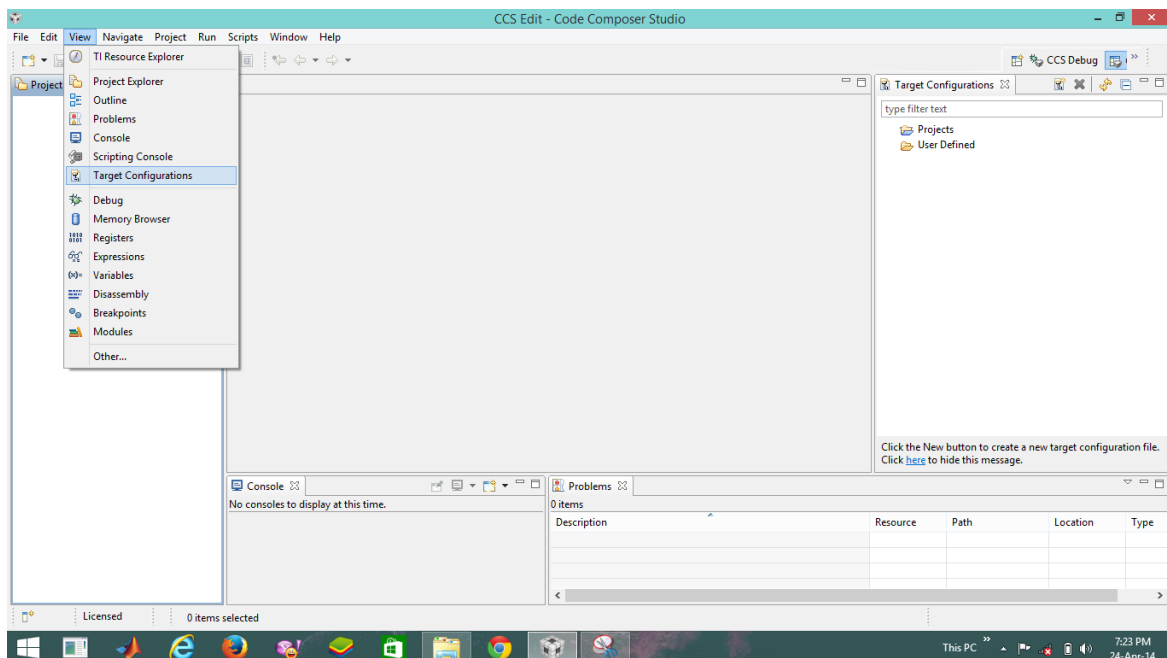


Figure 3.3: Create New Target Configuration File



2. A dialogue box will open, give appropriate name with extension 'ccxml' and press 'finish' button.

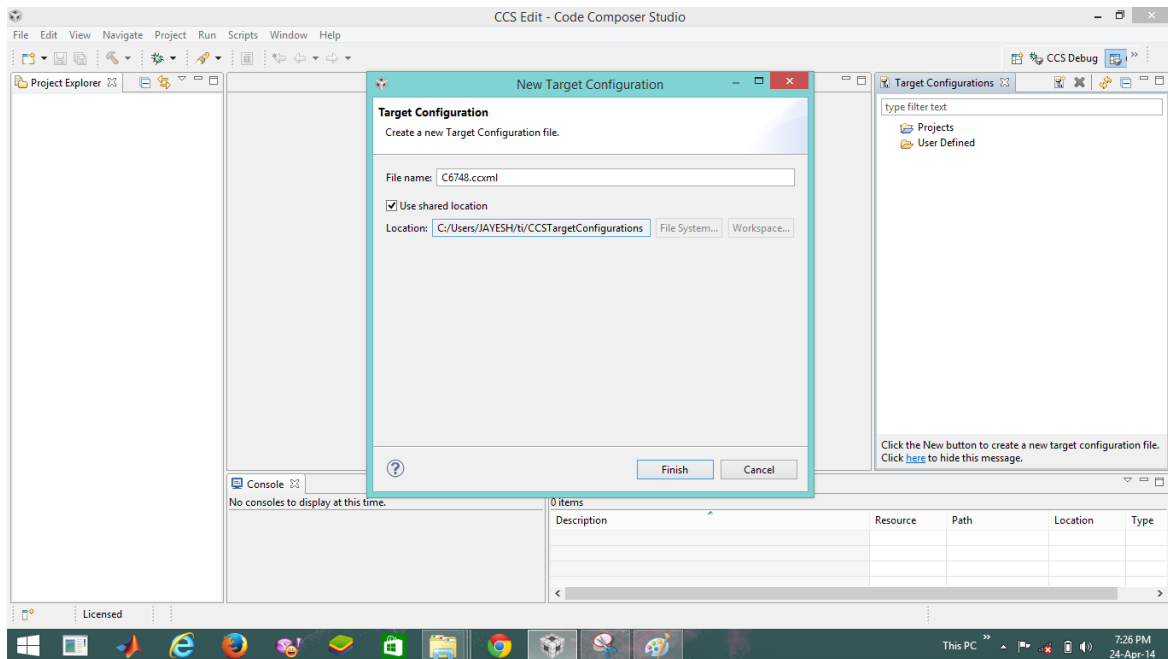


Figure 3.4: Location and Extention of Target Configuration File



- Now, from the ‘Target Configuration’ box, select the Target configuration file with ccxml extension which, we have just created. Select “Texas Instruments XDS100v1 USB Emulator” from the ‘connection’ option and “TMS320C6748” from ‘Board or Device’ option from the ‘Basic’ tab.

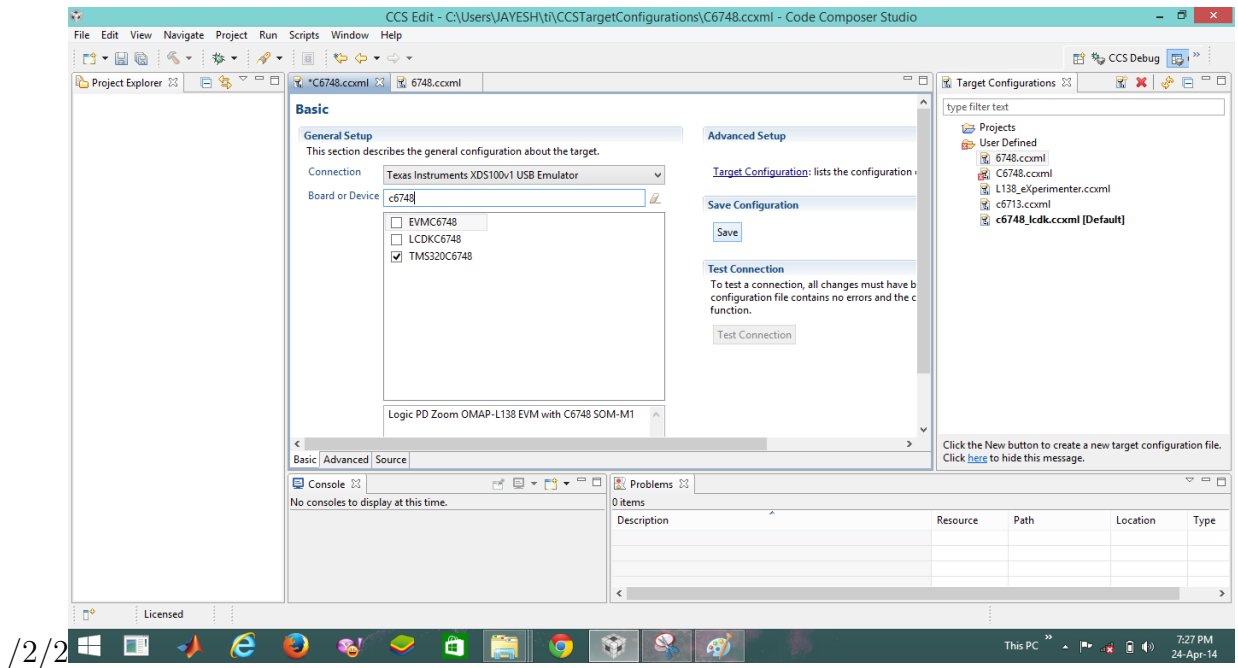


Figure 3.5: Basic Configuration Parameters for Target Configuration File



- Now, from the 'Advanced'tab, select C674X\_0 and browse 'C6748.gel'file from your system.

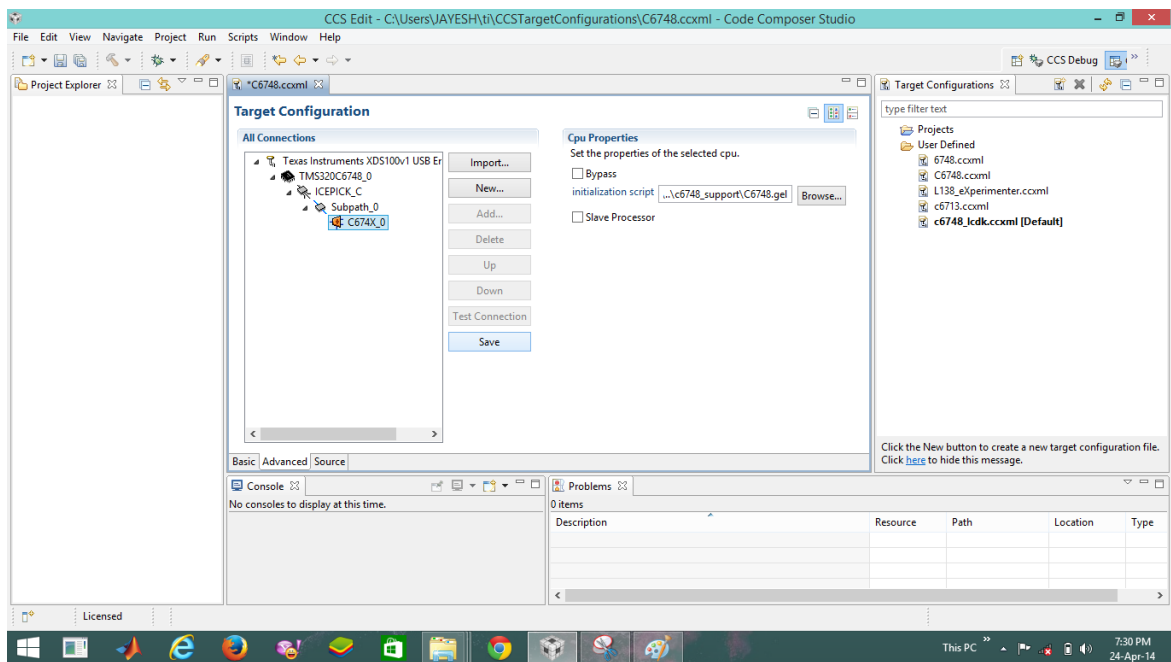


Figure 3.6: Advance Configuration parameters for Target Configuration File



5. At last, right click on the created ccxml file and select 'Launch selected configuration' and it will perform the initialization process for connecting the OMAP-L138 kit with CCSv5.

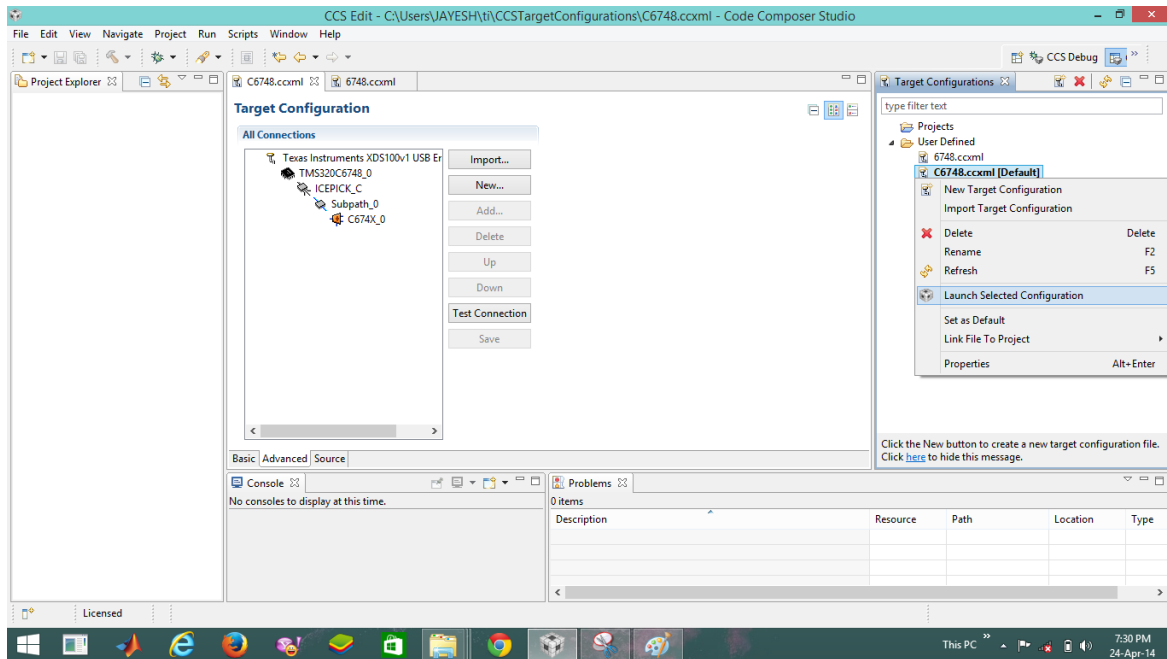


Figure 3.7: Launching of Selected Target Configuration File



# Chapter 4

## Behavioral Simulation

Behavioral simulation is a simulation process that is performed by interpreting the equations that define the design. The equations do not need to be converted to the logic that represents them.

### 4.1 Objective

Feasibility of any project can be proposed after having some simulation results. For the same purpose, the simulation results of TTS system at some abstract level is been done and shown here in the form of behavioral simulation. The main objective of Text-To-Speech(TTS) system is to convert the typed or scanned text to its equivalent pronunciation. Thus, there are two methods of input for this system : Typed Input and Scanned Input. Here, in this chapter, results for both the input has been shown.

### 4.2 Simulation results for typed input

Simple TTS system can be implemented by using Dictionary based concept. All general words which are used in the Gujarati language can be recorded and stored in the form of dictionary. For this project, input is to be taken in Gujarati language in English from keyboard. After that, the program will first ask the number of words in your string. Then, it will map each and every words with the recorded/stored words. If the entered word is not matched with the recorded words, then an error message will be played as an output. If the entered word is match with the recorded words, then it will directly play that word as an output. At last, program will concatenate all



the words and play a smooth sentence as an output. Fig.4.1(a) shows the output in which program asks user to give no. of words and a string as an input. Thus, program will give the equivalent phonemes or voiced words as an output. It will also give us the graph of every words respectively in time domain as shown in fig.4.1(b).

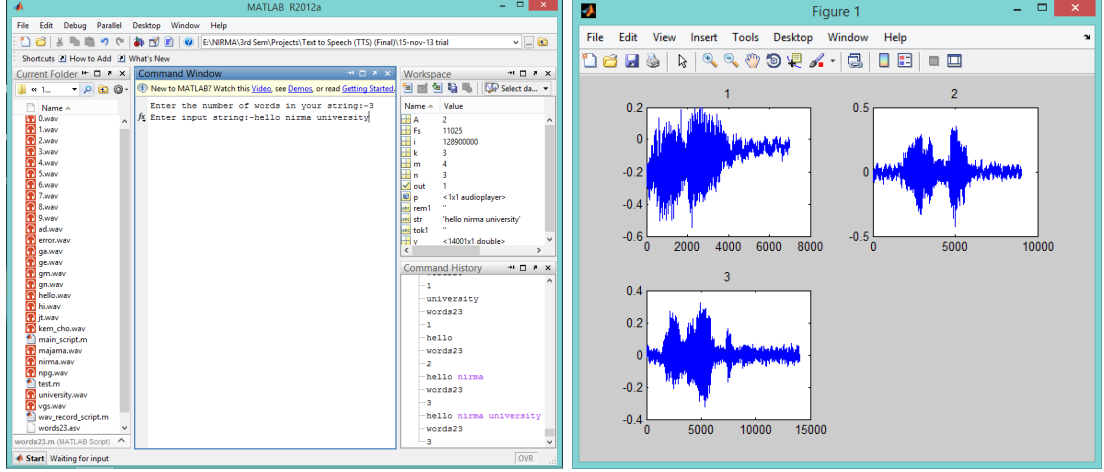


Figure 4.1: Simulation inputs for TTS using word based concatenative synthesis

### 4.3 Simulation results for scanned input

For this system, first of all the Gujarati characters in scanned image should be recognize by the processor. After that, one to one mapping of each letter with its equivalent pronunciation and concatenation has been performed as to generate desired output speech. Thus, for working with scanned input, TTS have two methods. *Histogram method* and *Edge detection algorithm method*

#### 4.3.1 Simulation results for Histogram method

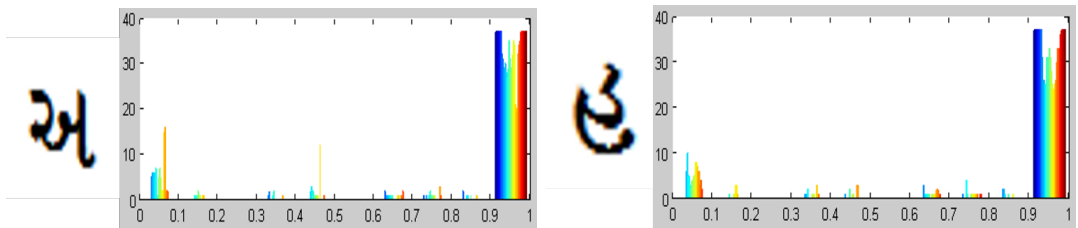


Figure 4.2: Histogram of Gujarati characters ‘અ’ and ‘ઉ’



This method is explained in detail in section 5.1. After referring the flow chart for this method, it can be easily understand by everyone, what this method wants to say. But, as mentioned in the subsection 5.1.1, the biggest challenge in this method is to compare the histogram of stored image and histogram of recognized character. The histogram produce by the code given in the section B.1 is shown in the fig.4.2.

### 4.3.2 Simulation results for Edge detection algorithm method

This method is explained in detail in section 5.2. After referring the flow chart for this method, it can be easily understand by everyone, what this method wants to say. By using this method, one can easily recognize the text from the scanned documents but, again the problem is to compare two images. The code for detecting text from the scanned document is shown in section B.2 and the simulation result produce by that code is shown in fig.4.5.

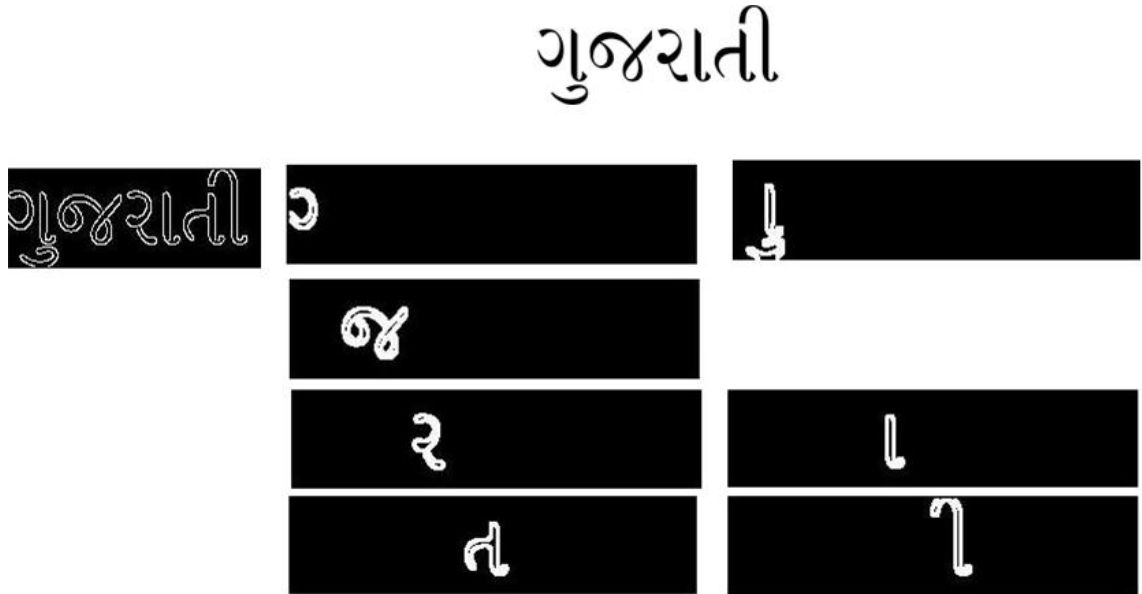


Figure 4.3: Character recognition by Edge detection algorithm

## 4.4 Festival - Open source tool for TTS synthesis

### 4.4.1 Installing Festival(Windows)

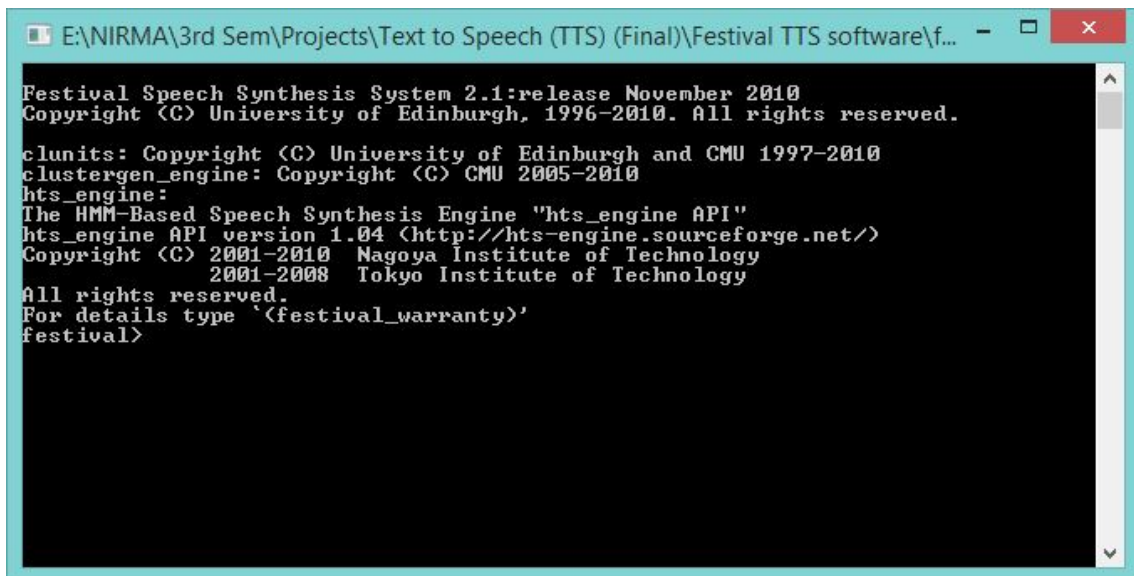
To install Festival you need to type the following commands:



1. Download the following file  
<http://homepages.inf.ed.ac.uk/jyamagis/misc/demo-material-win.zip>
2. Unzip the file
3. Copy “festival” directory in the unzipped directory to the top directory of C:\
4. Please click festival.exe

### 4.4.2 Starting Festival

- (1) Click ‘festival.exe’ file in unzipped directory.
- (2) Making Festival speak, you need to type the following commands:  
 Festival> (SayText "Jayesh Tanna is a Smart boy.")



```

Festival Speech Synthesis System 2.1:release November 2010
Copyright (C) University of Edinburgh, 1996-2010. All rights reserved.

clunits: Copyright (C) University of Edinburgh and CMU 1997-2010
cluster engine: Copyright (C) CMU 2005-2010
hts_engine:
The HMM-Based Speech Synthesis Engine "hts_engine API"
hts_engine API version 1.04 (http://hts-engine.sourceforge.net/)
Copyright (C) 2001-2010 Nagoya Institute of Technology
                2001-2008 Tokyo Institute of Technology
All rights reserved.
For details type '<festival_warranty>'
festival>

```

Figure 4.4: Terminal of Festival

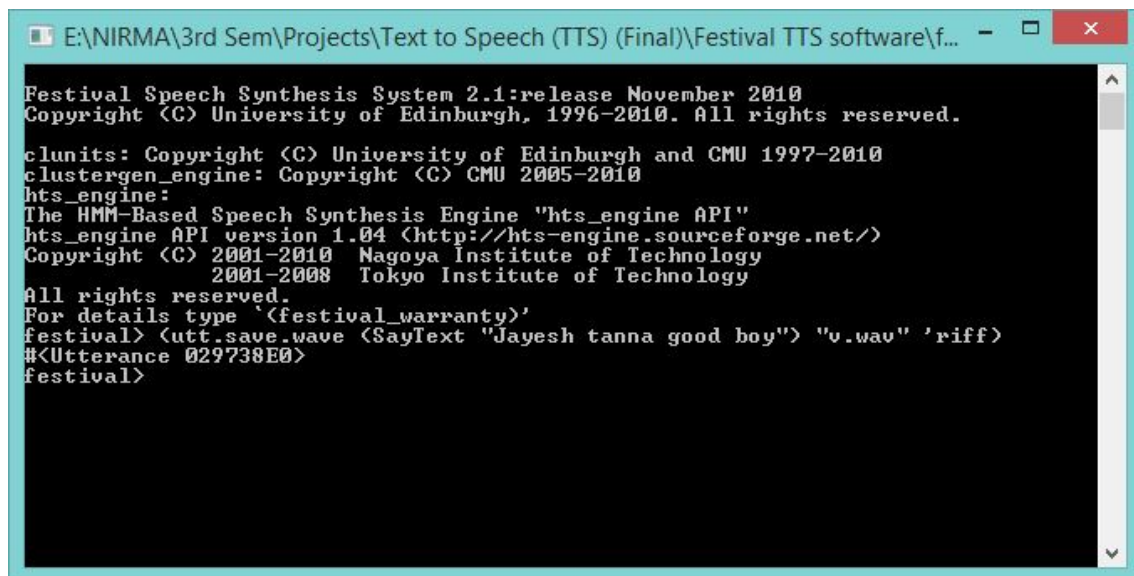
5. Getting waveforms out of festival tool: Once you have a fully synthesised utterance object in festival, it is possible to extract the waveform to a (.wav) file as follows:

```

festival>(utt.save.wave (SayText "Jayesh Tanna is a Smart boy.") "name.wav"
'riff)

```





```
E:\NIRMA\3rd Sem\Projects\Text to Speech (TTS) (Final)\Festival TTS software\f...  
Festival Speech Synthesis System 2.1:release November 2010  
Copyright (C) University of Edinburgh, 1996-2010. All rights reserved.  
clunits: Copyright (C) University of Edinburgh and CMU 1997-2010  
clusterengen: Copyright (C) CMU 2005-2010  
hts_engine:  
The HMM-Based Speech Synthesis Engine "hts_engine API"  
hts_engine API version 1.04 (<http://hts-engine.sourceforge.net/>  
Copyright (C) 2001-2010 Nagoya Institute of Technology  
2001-2008 Tokyo Institute of Technology  
All rights reserved.  
For details type '<festival_warranty>'  
festival> (utt.save.wave (SayText "Jayesh tanna good boy") "v.wav" 'riff)  
#<Utterance 029738E0>  
festival>
```

Figure 4.5: Getting waveform out of Festival



## Chapter 5

# Proposed Method of TTS for Gujarati language

There are always some fundamental rules for any language's grammar for pronouncing a word. Thus, for working on any specific language, first one must know almost everything about that specific language as to avoid mispronunciation. However, there may be some words or characters having different pronunciation at different places, where they are being used.

All the characters of Gujarati language can be classified in the following four groups:

- Groups of characters having Vertical line at the end:-  
'અ', 'ખ', 'ગ', 'ઘ', 'ચ', 'ણ', 'ત', 'થ', 'દ', 'ધ', 'ન', 'ભ', 'ભ',  
'મ', 'ય', 'લ', 'વ', 'શ', 'ષ', 'સ', 'હ', 'ણ', 'ણ'
- Group of Characters with no vertical line:-  
'ક', 'ઘ', 'જ', 'ઝ', 'ઙ', 'ઠ', 'ડ', 'ઢ', 'ફ', 'ર', 'લ'
- Group of heads:- 'ં', 'ૃ'
- Group of tails:- 'ં', 'ૃ', 'ૃ'

As shown in the fig.1.1, the main function of text analysis or text normalization block is to convert all the pixel values to '0' and '1'. Pixel having gray value above threshold is converted to '1' and below threshold is converted to '0'. For perfect result of TTS system, first, we need to separate out lines from the scanned document, then words from lines and at last characters from the words. Again, may be these characters have heads and tails, so again we need to separate out them[3].



## 5.1 Recognizing scanned Input by Histogram

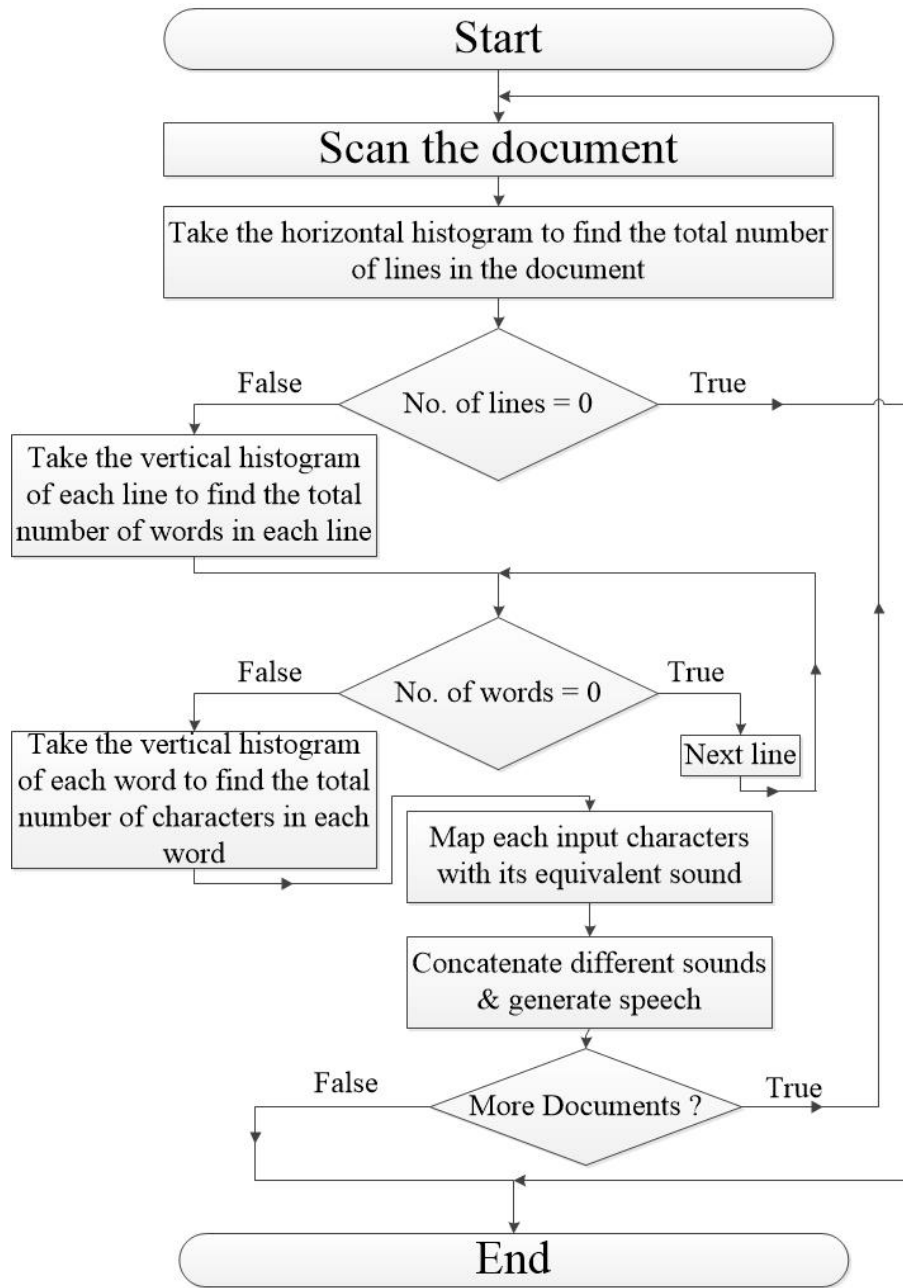


Figure 5.1: Flow chart for recognizing scanned Input by Histogram

The first method is recognizing an input image by comparing its histogram with stored histograms of each and every characters. Histogram is the distribution of intensity of the pixels of image. Thus, for separating lines in a document, horizontal histogram of document is useful. The blank space between the two lines indicates the separation of the lines. To separate down the words from the lines, we need to take vertical histogram. The whole process of how to recognize scanned Input image



by histogram is shown the fig.5.1 and histogram of Gujarati character ‘અ’ and ‘ઉ’ shown in fig.4.2(a) and 4.2(b) respectively.

### 5.1.1 Challenges for recognizing a character for Histogram method

- Challenging task in this histogram method is the comparison of the two histograms (histogram of pre-stored characters and histogram of scanned characters).
- Even comparison of two different histogram is almost impossible due to histogram is just a distribution of intensity of any image.

After separating the characters from lines, first it is to be decided that from which group it belongs to. After identifying the proper group, we have to identify the exact character by using different methods. When TTS system come to know the exact character, it will be assigned a unique number[3], which is already been stored in the database, and by using that number our TTS system will do mapping to its equivalent phoneme.

## 5.2 Recognizing scanned Input by Edge detection algorithm

Another method for recognizing scanned input image is by using edge detection algorithms. Edge detection methods aims at identifying points in digital image at which image brightness changes sharply. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges[9]. Applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. There are many edge detection algorithms like *canny*, *sobel*, *prewitt*, *roberts* etc. are already available. The process of recognizing scanned Input by edge detection algorithm explained in the form of flow chart in the fig.5.2 For detecting the edges from any image, first convert them into grayscale and then filter them for removing the noise. Now,



apply edge detection algorithm to the filtered image and get the edged image as output. Next is to mask that image with proper masking matrix and segment all the characters individually as shown in the fig.4.5.

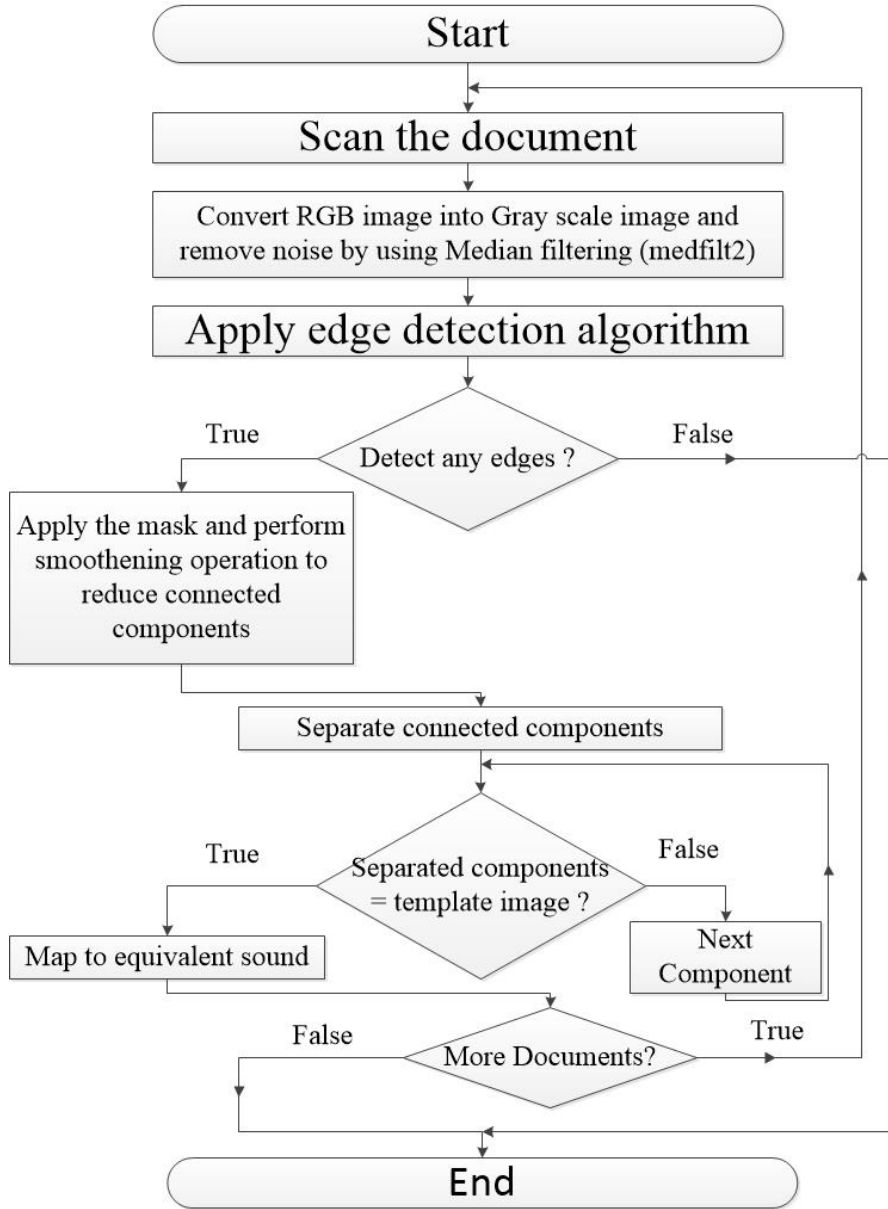


Figure 5.2: Flow chart for recognizing scanned Input by Edge detection algorithms



# Chapter 6

## Conclusion and Future Scope

Among all the different synthesis technologies, *concatenative synthesis* is superior than other synthesis technologies and used widely. Concatenative synthesis can be used universally for any language of any part of the word. General characteristics of different synthesis technologies are:

1. *Concatenative synthesis* requires amount of memory according to the words recorded and stored in the dictionary. Memory requirement will be high if this system supports all the words of any language. As memory is the drawback of this system, concatenative synthesis technology is not generally used in the embedded system and it provides highest naturalness in the output speech.
2. *Formant synthesis*, requires high programming for generating each words sound automatically, thus requires very less memory as compared to others, does not provide naturalness in the output, it will generate robotic sound.
3. *Articulatory synthesis*, is the most complex system amongs all, requires smart programming to model the whole articulatory system of human being.

Text-To-Speech (TTS) system can be used to make any device conversational. By using this system, any scanned document can be read out loud. Furthermore, imposing of one particular frequency on the recorded sound for generating the output speech in the imposed frequency. This system can be made font and font size independent.



# References

- [1] John F. Pitrelli, “The IBM Text-to-Speech synthesis system for American English”, in *IEEE transactions on audio, speech and language processing*, Vol.14, No.4, July 2006.
- [2] Applied speech and audio processing with MATLAB examples by , “Ian McLoughlin”, *Cambridge publications*.
- [3] Prajakta S. Rathod, “Script to speech conversion for Hindi language by using Artificial Neural Network”, in *NUiCONE-2011*.
- [4] Aimilios Chalamandaris, Sotiris Karabetsos, “A Unit Selection Text-to-Speech Synthesis System Optimized for Use with Screen Readers”, in *IEEE Transactions on Consumer Electronics*, Vol. 56, No. 3, August 2010.
- [5] A.B. Mosaddeque, N. UzZaman and M. Khan, “Rule based Automated Pronunciation Generator”, *Proceedings of 9th International Conference on Computer and Information Technology*, Dhaka, Bangladesh, 2006.
- [6] Marian Macchi, “Issues in Text-to-Speech Synthesis”.
- [7] Chien-Liang Liu, Wen-Hoar Hsaio, Chia-Hoang Lee, and Hsiao-Cheng Chi, “An HMM-Based Algorithm for Content Ranking and Coherence-Feature Extraction”, in *IEEE transactions on systems, man and cybernetics: systems*, Vol. 43, No.2, March 2013.
- [8] OMAP-L138 DSP+ARM Processor *Technical Reference Manual (Texas Instruments)* Literature Number : SPRUH77A December 2011
- [9] [http://en.wikipedia.org/wiki/Edge\\_detection](http://en.wikipedia.org/wiki/Edge_detection)



- [10] Krishnendu Ghosh, K.Sreenivasa Rao “Subword Based Approach for Grapheme-To-Phoneme Conversion in Bengali Text-To-Speech Synthesis System” 978-1-4673-0816-8/12 \$31.00 ©2012 IEEE.
- [11] Er. Sheilly Padda, Ms. Rupinderdeep kaur, Er. Nidhi, “Architecture and Implementation of Punjabi Text to Speech System Using Transcriptions Concept”, in *International Journal of Engineering Research and Development*, ISSN: 2278-067X, Volume 1, Issue 5 (June 2012), PP.08-11.



# List of Publications

- [1] Jayesh B. Tanna, Vijay Savani, Amit Degada “Gujarati Text-To-Speech (TTS) conversion system using Histogram and Edge detection method”, in *International Conference IAET-2014 at Jaipur*, Volume.3, No.1, January-June, 2014, pp- 204-207. Print-ISSN: 2277-4904, Online-ISSN: 2277-4912.
  
- [2] Jayesh B. Tanna, Vijay Savani, Amit Degada, “Text-To-Speech Conversion for Gujarati language”, in *Journal of Electronics Design Technology (JoEDT), STM Journals*, Volume 4, Issue 3, December-2013. ISSN: 2229-6980.



# Appendix A

## MATLAB code for TTS with typed Input text

### A.1 Dictionry based TTS

#### A.1.1 Main code

```
% Author : Jayesh Tanna

clc;
clear all;
n=input('Enter the number of words in your string:-');
if(rem(n,2)==0)
    m=n;          % Make the number of words even
else
    m=n+1;
end
str=input('Enter input string:-','s');
[tok1,rem1]=strtok(str);
% Find token in string; Separate out each words from the sentence
for k=1:n

    for i = 1:128900000          % Provides delay
        A=1;
```



```

        A = A+1;
    end

    Fs=11025;          % Set sample frequency

    out=strcmpi('hello',tok1); % Compares each words regardless of their case
    if(out==1)
        y=wavread('hello.wav'); % Reads wav file

        p=audioplayer(y,Fs); % Creates audioplayer object for signal y with Fs freq
        play(p);             % Plays audio samples in audioplayer object

    elseif(out==0)
        out=strcmpi('nirma',tok1);
        if(out==1)
            y=wavread('nirma.wav');
            p=audioplayer(y,Fs);
            play(p);

        end

    if(out==0)
        out=strcmpi('university',tok1);
        if(out==1)
            y=wavread('university.wav');
            p=audioplayer(y,Fs);
            play(p);

        end

    if(out==0)
        out=strcmpi('9',str);
        if(out==1)

```



```
        y=wavread('9.wav');
        p=audioplayer(y,Fs);
        play(p);

    else
        y=wavread('error.wav');
        p=audioplayer(y,Fs);
        play(p);

    end

end

end

end

end

if(m>2)
    subplot((m/2),(m/2),k);
    plot(y);

    title(k);
else
    subplot((m),(m),k);
    plot(y);
    title(k);
end

[tok1,rem1]=strtok(rem1);

end
```

### A.1.2 Script file for recording each word

```
% Author : Jayesh Tanna
```

```
clc;
```



```
clear all;
Fs=11025;
y=wavrecord(5*Fs,Fs,'int16'); % Record sound using Windows audio input device
wavplay(y,Fs);                % Play sound using Windows audio output device
yain=y(20000:1:42000);
wavplay(yain,Fs);
wavwrite(yain,Fs,'hi.wav');    % Play sound using Windows audio output device
```



# Appendix B

## MATLAB code for TTS with scanned Input text

### B.1 Histogram method

```
% Author : Jayesh Tanna

x=imread('big.bmp');
y=rgb2gray(x);
z=mat2gray(y);
figure(1);
hist(z);
title('histogram of big a');
u=size(z);
disp(u);

a=imread('small.bmp');
b=rgb2gray(a);
c=mat2gray(b);
v=size(c);
disp(v);

figure(2);
```



```
hist(c);
title('histogram of small a');
```

## B.2 Edge detection algorithm

```
% Author : Jayesh Tanna
```

```
clc;
clear all;
%close all;
k=input('Enter the file name','s'); % input image; color image
im=imread(k);
im1=rgb2gray(im);
im1=medfilt2(im1,[3 3]); % Median filtering the image to remove noise
BW = edge(im1,'canny',0.10); % finding edges
[imx,imy]=size(BW);
msk=[0 0 0 0 0;
      0 1 1 1 0;
      0 1 1 1 0;
      0 1 1 1 0;
      0 0 0 0 0;];
B=conv2(double(BW),double(msk));
% Smoothing image to reduce the number of connected components
L = bwlabel(B,8); % Calculating connected components
mx=max(max(L))

figure,imshow(im);
figure,imshow(im1);
figure,imshow(B);

for n=1:12
[r,c] = find(L==n);
```



```
rc = [r c];  
[sx sy]=size(rc);  
n1=zeros(imx,imy);  
for i=1:sx  
x1=rc(i,1);  
y1=rc(i,2);  
n1(x1,y1)=255;  
end % Storing the extracted image in an array  
  
figure,imshow(n1,[]);  
end
```