Silicon Technologies for Speaker Independent Speech Processing and Recognition Systems in Noisy Environments

Karthikeyan Natarajan[1], Dr.Mala John, Arun Selvaraj[2]
Madras Institute of Technology, Anna University
India

1. Introduction

As the speaker independent speech recognition problem itself is highly computation intensive, the external environment adds to recognition complexity. As per Moore's law, doubling of number of transistors in a chip per year lead to the integration of various architectures in high density chips which lead to the implementation of high complex mixed signal speech systems in FPGA and ASIC technologies. Though several software based speech recognition systems are developed over the years, speech system implementations are yet to unleash the capabilities of silicon technologies. Direct mapped, completely hardware based systems will be highly energy efficient and less flexible but processor based implementation will be less energy efficient and flexible. Software based recognition systems fail to meet the latency requirements of the real time conditions whereas a completely hardware based recognition systems are power intensive. Hence in this case study, a hardware software based co-design is considered for the speech recognition implementation. Sequential algorithms which have been developed need to be modified to suit the parallel hardware systems. Hardware and software based co-design of the isolated word recognition problem will be applicable for low power systems like an AI based robotic system which could use a fixed point arithmetic and hence algorithmic optimizations needed to be considered to suit the actual hardware. Isolated word recognition problem can be split into three stages namely speech analysis, robust processing and final recognition stage. This hardware based speech recognition system is characterized for power and computation efficiency with the following parameters namely vocabulary size, robust speech recognition, speech variability, power and fixed point inefficiencies. This hardware system uses 50Mbps (Max 100Mhz) / 50Mhz NIOS 2 processor with WM8731 audio codec, DRAM controller, I2C controller, Avalon Bus bridge controller, ASIP matrix processor and parallel log Viterbi based hardware module implemented in ALTERA FPGA.

This chapter provides an Introduction to Hidden Markov model based speech Recognition. Relative merits and demerits of conventional Filter bank based feature extraction algorithm via windowed Fourier transform method is compared with a parallel linear predictive coding based CMOS implementation. Detailed description of the HMM based speech

[1] Author is currently working in IBM India Systems and Technology Engineering Labs
[2] Author is currently working in Wipro Technologies, Chennai.

Source: Speech Recognition, Technologies and Applications, Book edited by: France Mihelič and Janez Žibert,
ISBN 978-953-7619-29-9, pp. 550, November 2008, I-Tech, Vienna, Austria

recognition SOC chip is explained in this section. The robust processing step which involves the removal of external unintended noise component from speech signal and the novel Application specific matrix processor for noise removal based on signal subspace based Frobenious norm constrained algorithm are further discussed. This ASIP matrix solver consists of Singular value decomposition unit, QR decomposition unit, matrix bi-diagonilization unit, Levinison-Durbin Toeplitz matrix solver, fast matrix transposition unit based on efficient address generation module. Discussion on word recognition implementation as a parallel 32 bit fixed point 32 state univariate Hidden Markov model based system in ALTERA FPGA is carried out in the final section of this chapter.

1.1 Introduction to HMM based speech recognition system

Speech recognition can be classified into three categories namely Isolated, Connected and Continuous speech recognition systems. In an isolated word recognition system, each word is assumed to be surrounded by silence or background noise. This means that both sides of a word must have no speech input, making definite word boundaries easy to construct. This kind of recognition is mainly used in applications where only a specific digit or a word needs to be identified. Implementation of Isolated word recognition doesn't require any language information and it uses the minimum information about the source speech and has the low recognition accuracy for very large vocabulary. Connected speech (or more correctly 'connected utterances') recognition is similar to isolated word Recognition, but it allows several words/digits to be spoken together with minimal silence period between them. Longer phrases or utterances are therefore possible to be recognized. Continuous speech recognition is method for recognizing spontaneous speech. The system is able to recognize a sequence of connected words, which are not separated by pauses, in a sentence. This mode requires much more computation time and memory, and it is more difficult to operate when compared to isolated word recognition. A speaker-dependent system is a system that recognizes a specific speaker's speech while speaker-independent systems can be used to detect speech by any unspecified speaker. Currently speaker independent systems are modeled using Gaussian Mixture based quantizers which have high recognition accuracy. For speaker independent speech recognition system the training data must be exhaustive, which should incorporate all kinds of speaker variations. It is clear that the smaller the vocabulary size, the higher the recognition accuracy. In an isolated digit recognition system we can achieve higher accuracy by storing finer models of the digits. Further if the vocabulary size is increased there is significant reduction in the computational performance of the system. The training data needs to be generated from the field or the environment where we are planning to implement it.

Isolated Word Recognition problem can be divided into two parts, namely - Front End Processing and Pattern recognition. Typically, the front-end building block includes two modules, data acquisition and feature extraction. In our system we have also implemented the end-point detection and speech enhancement module to make the speech signal more adaptive and robust to the noise. The first stage in any Speech Recognition system is modeling the input speech signal based on certain objective parameters also called the Front End Parameters. Modeling of the input speech signal involves three basic operations spectral modeling, Feature extraction, and parametric transformation (Figure 1). Spectral shaping is the process of converting the speech signal from analog to digital and emphasizing important frequency components in the signal. Noise suppression and speech enhancement module can be added to the Front end processing module which will improve the recognition accuracy.

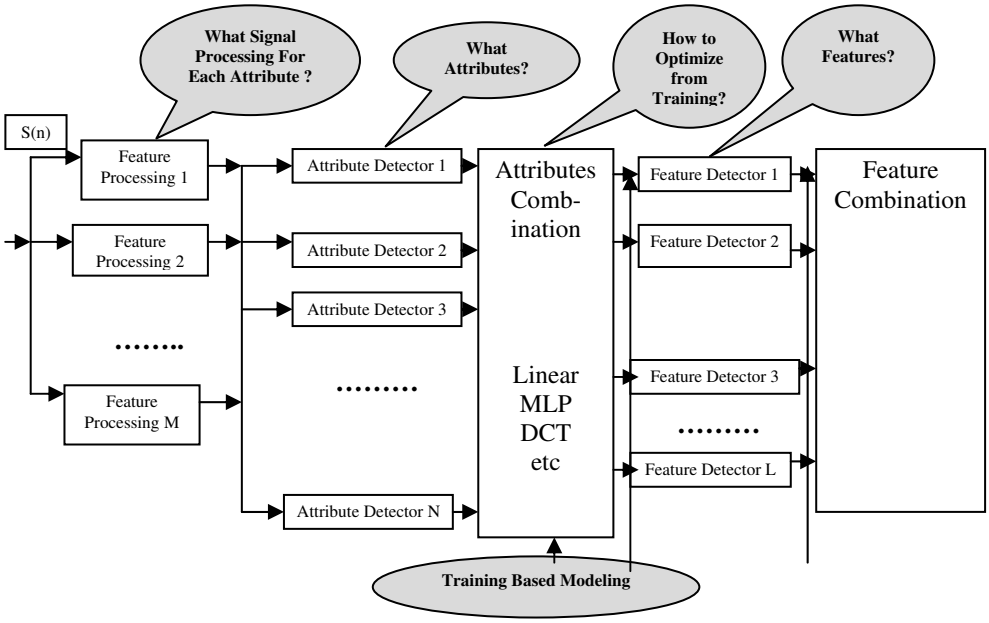

Fig. 1. Components of a speech recognition system

Two major kinds of front end Processing methods are Linear Predicative Coding and Mel Frequency Cepstral Co-efficient. The basic idea behind the linear predictive coding (LPC) analysis is that a speech sample can be approximated as a linear combination of past speech samples. By minimizing the sum of the squared differences (over a finite interval) between the actual speech samples and the linearly predicted ones, a unique set of predictor coefficients is determined. Speech is modeled as the output of linear, time-varying system excited by either quasi-periodic pulses (during voiced speech), or random noise (during unvoiced speech). The linear prediction method provides a robust, reliable, and accurate method for estimating the parameters that characterize the linear time-varying system representing vocal tract.In linear prediction (LP) the signal s (n) is modeled as a linear combination of the previous samples:

$$s(n) = \overleftrightarrow{S}(n + e(n) = \sum_{i=1}^{NlP} a_{LP}(i)s(n-i) + e(n) \tag{1}$$

a_{LP} (i) are the coefficients that need to be decided, N_{LP} is the order of the predictor, i.e. the number of coefficients in the model, and e(n) is the model error, the residual. There exists several methods for calculating the coefficients. The coefficients of the model that approximates the signal within the analysis window (the frame) may be used as features, but usually further processing is applied. Higher the order of the LP Filters used, better will be the model prediction of the signal. A lower order model, on the other hand, captures the trend of the signal, ideally the formants. This gives a smoothened spectrum. The LP coefficients give uniform weighting to the whole spectrum, which is not consistent with the

human auditory system. For voiced regions of speech all pole model of LPC provides a good approximation to the vocal tract spectral envelope. During unvoiced and nasalized regions of speech the LPC model is less effective than voiced region. The computation involved in LPC processing is considerably less than cepstral analysis. Thus the importance of method lies in ability to provide accurate estimates of speech parameters, and in its relative speed. The features derived using cepstral analysis outperforms those that do not use it and that filter bank methods outperform LP methods. Best performance was achieved using MFCCs with Filter bank processing. Even though the CPU computations and memory accesses for MFCC are more, they are less speaker dependent and more speaker Independent. In our implementation we are using Short Time Fourier Transform based MFCC Feature Extraction Method for Front End Processing(Figure 2).

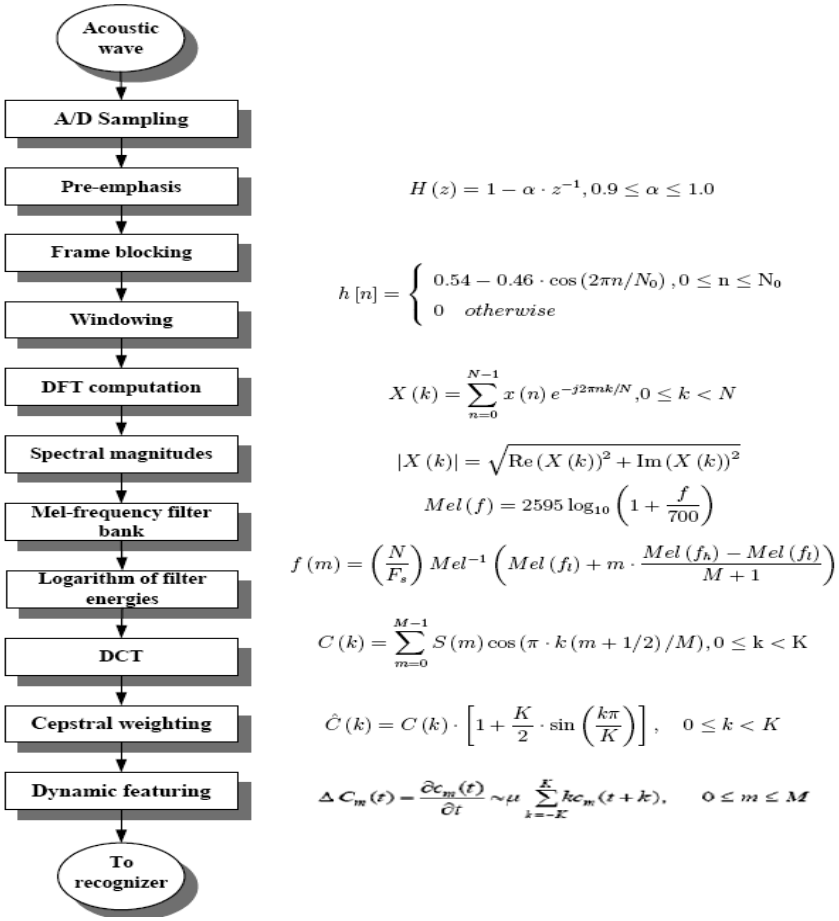

Fig. 2. Flow for Front End processing with feature extraction

We have found that with the hamming window of length 256 the signal can be represented efficiently with consideration to the hardware computational requirements of implementing

a FFT routine. After windowing the speech signal, Discrete Fourier Transform (DFT) is used to transfer these time-domain samples into frequency-domain ones. Direct computation of the DFT requires N^2operations, assuming that the trigonometric functions have been pre-computed. Meanwhile, the FFT algorithm only requires on the order of $N \log_2 N$ operations, so it is widely used for speech processing to transfer speech data from time domain to frequency domain.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \; 0 <= k < N \tag{2}$$

The Spectral magnitudes were obtained by computing the absolute values of the FFT real and imaginary outputs. The square root is a monotonically increasing function and can be ignored if only the relative sizes of the magnitudes are of interest (ignoring the increased dynamic range).

$$|X(k)| = \sqrt{\mathrm{Re}(X(k))^2 + \mathrm{Im}(X(k))^2} \tag{3}$$

The computation still requires two real multiplications and consumes a lot of latency. A well-known approximation to the absolute value function is given as.

$$\left|A_{\mathrm{re}} + jA_{\mathrm{im}}\right| \approx \left|A_{\mathrm{re}}\right| + \left|A_{\mathrm{im}}\right| \tag{4}$$

A less frequently used approximation is only slightly more complex to implement but offers far better performance (refer table 1).

$$\left|A_{\mathrm{re}} + jA_{\mathrm{im}}\right| \approx \max\left(\left|A_{\mathrm{re}}\right|, \left|A_{\mathrm{im}}\right|\right) + \tfrac{1}{2}\min\left(\left|A_{\mathrm{re}}\right|, \left|A_{\mathrm{im}}\right|\right) \tag{5}$$

The above approximation was considered for the computation of spectral magnitude of the FFT outputs and their spectral magnitudes are taken. Human auditory system is nonlinear in amplitude as well as in frequency. We have taken logarithm to emulate amplitude nonlinearity and Mel filter banks to incorporate frequency nonlinearity. We have used 27 Mel triangular filter banks with 102 coefficients evenly spaced in Mel domain and the cepstral vectors are extracted based on the following equation 6 (refer Figure 3).

$$f(k) = (N/Fs) * Mel^{-1}(Mel(F_{low}) + k * \frac{(Mel(F_{high}) - Mel(F_{low})}{M+1}) \tag{6}$$

$$Mel(f) = 2595 * \log_{10}(1 + \frac{f}{700}) \tag{7}$$

$$Mel^{-1}(f) = 700 * (10^{(\frac{f}{2595})} - 1) \tag{8}$$

$$H_m(k) = \begin{cases} 0 \cdots\cdots\cdots if \cdots\cdots\cdots k < f(m-1) \\ \dfrac{(k - f(m-1))}{f(m) - f(m-1)} \cdots\cdots\cdots if \cdots\cdots\cdots f(m-1) \le k \le f(m) \\ \dfrac{(f(m+1) - k)}{f(m+1) - f(m)} \cdots\cdots\cdots if \cdots\cdots\cdots f(m) \le k \le f(m+1) \\ 0 \cdots\cdots\cdots if \cdots\cdots\cdots k > f(m+1) \end{cases} \quad (9)$$

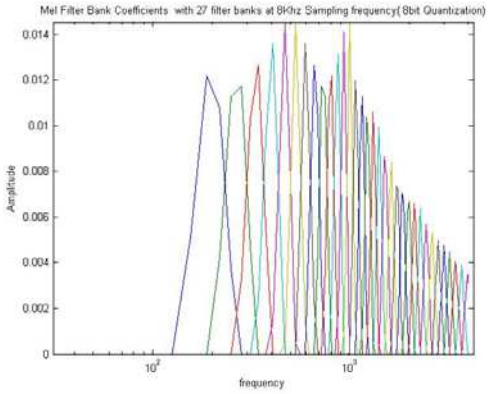

Fig. 3. Mel Filter Bank

The inverse DFT is performed on the output of the filter bank. Since the log power spectrum is symmetric and real, the inverse DFT is reduced to discrete cosine transformation (DCT). This transformation decorrelates features, which leads to using diagonal covariance matrices instead of full covariance matrices while modeling the feature coefficients by linear combinations of Gaussian functions. Therefore complexity and computational cost can be reduced. This is especially useful for speech recognition systems. Since DCT gathers most of the information in the signal to its lower order coefficients, by discarding the higher order coefficients, significant reduction in computational cost can be achieved. Typically the number of coefficients, *K*, for recognition ranges between 8 and 13. The equation is as followingSensitivity of the lower order cepstral coefficients to overall slope and a higher order coefficient to noise has necessitated weighing of the cepstral coefficients by a tapered window to minimize these sensitivities. We have used weighing by a band pass filter of the form. Temporal cepstral derivatives are an improved feature vector for forming the speech frames. They can be used with the cepstral derivative in case the cepstral Coefficients do not give acceptable recognition accuracy. Cepstral representations provide good approximations to the local spectral Properties. Derivatives of cepstral coefficients can be used to describe the dynamic movement of spectrum.In practical applications, the following approximation is used,

$$\Delta C_m(t) = \frac{\partial\, C_m(t)}{\partial t} \approx \left\{ \mu * \sum_{k=-K}^{K} k * C_m(t+k) \right\} 0 \le m \le M \quad (10)$$

Where μ is a normalization factor.

Typical feature vector: (Figure 4):
[E(t) c1(t) c2(t)… cM(t), E(t) , ΔE(t) , Δc1(t) Δc2(t)… ΔΔcM (t-1) ΔΔc1(t) ΔΔc2(t).. ΔΔcM (t-1)]^T
Feature vector consists of both static part and the Dynamic part of the speech signal.

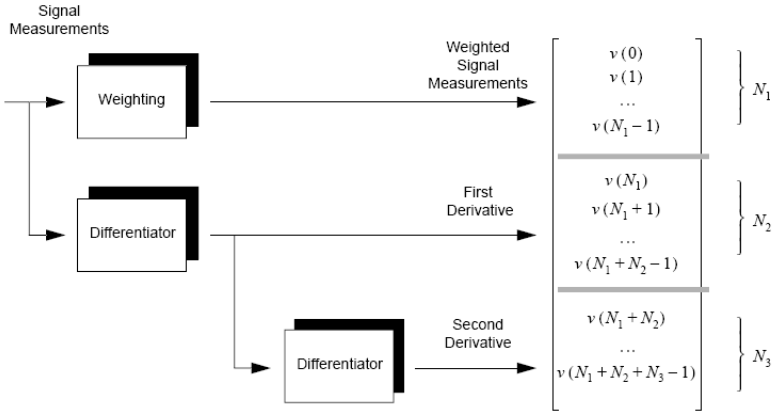

Fig. 4. Representation of Delta and Delta- Delta parameters

2. The hidden Markov models

2.1 The three basic problems of HMM

1. Given the observation sequence O = (o_1 o_2 …o_T), and a model λ=(A, B, π), how do we efficiently compute p(o | λ), the probability f the observation sequence, given the model. This is the "evaluation problem". Using the forward and backward procedure provides solution.
2. Given the observation sequence O = (o_1 o_2 …o_T), and the model λ, how do we choose a corresponding state sequence q=(q_1,q_2…q_T) that is optimal in some sense(i.e. best explains the observation).The Viterbi algorithm provides a solution to find the optimal path.
3. How do we adjust the model parameters λ= (A, B, π) to maximize p(O | λ).This is by far the most difficult problem of HMM. We choose λ= (A, B, π) in such a way that its likelihood, p(O | λ), is locally maximized using an iterative procedure like Baum-Welch method (L. Rabiner 1993).

The base speech recognizer works only with noiseless HMM states and the matrix processor is used as pre conditioning block to generate the noiseless HMM models from the noisy speech vectors. There are three kinds of Hidden Markov models described in the literature namely Discrete HMM, Continuous HMM and Semi-continuous HMM (Vaseghi) in which a Continuous HMM model is used to model the HMM states.A HMM model is characterized by the no of states N, no of distinct observation symbols M, the transition probability matrix A, the initial probability matrix Πi,the output observation probability for a feature x1 in state I , $b_i(x_1)$.

2.2 The log-viterbi algorithm:

1) Initialization:

$$\delta^{(\log)}{}_1(i) = \log a_{1i} + \log b_i(x_1) \tag{11}$$

$$\psi_1(i) = 0 \tag{12}$$

2) Recursion:

$$\delta^{(\log)}{}_{t+1}(j) = \max_{i=2}^{N-1}(\delta^{(\log)}{}_t(i) + l\log a_{ij}) + \log b_j(x_{t+1}) \tag{13}$$

$$\psi_{t+1}(i) = \arg\max_{i=2}^{N-1}(\delta^{(\log)}{}_t(i) + l\log a_{ij}) \tag{14}$$

3) Termination:

$$\log(P(O/\lambda)) = \max_{i=2}^{N-1}(\delta^{(\log)}{}_T(i) + l\log a_{iN}) \tag{15}$$

$$q^T = \arg\max_{i=2}^{N-1}(\delta^{(\log)}{}_T(i) + l\log a_{iN}) \tag{16}$$

4)Backtracking:

$$q_t = \psi_{t+1}(q_{t+1})_{for t=T-1 to 1} \tag{17}$$

The probability of observation vectors , p(O| λ) has to be maximized for different model parameter values which corresponds to HMM models for different words. The implementation of the log likely computation can be done in an efficient way using the Forward and backward procedures as described in (Karthikeyan –ASICON 2007). Since the direct implementation of Viterbi algorithm results in underflow due to very low probability values are multiplied recursively over the speech frame window, logarithmic Viterbi algorithm is implemented which is different from methods given in (Karthikeyan - ASICON 2007). Since the direct implementation of Forward, Backward as well as the Viterbi algorithm results in underflow, we took logarithm on both sides and we have implemented logarithmic versions of the above algorithm. Since the Forward algorithm uses summation which is being replaced by the following conversion in the modified forward algorithm. We have used the modified forward algorithm, backward algorithm as well as viterbi algorithm which is different from the methods given in [6].

2.3 The Baum Welch re-estimation procedure

The third, and by far the most difficult, problem of HMMs is to determine a method to adjust the model parameters (A, B, π) to maximize the probability of the observation sequence given the model. There is no known way to analytically solve for the model, which maximizes the probability of the observation sequence. In fact, given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. We can, however, choose λ = (A, B, π) such that P(O | λ) is locally maximized using an iterative procedure such as the Baum-Welch method. To describe the procedure for re-estimation

(iterative update and improvement) of HMM parameters, we first define ξt(i,j), the probability of being in state Si at time t, and state Sj, at time t+1, given the model and the observation sequence.

In order to use either ML or MAP classification rules, we need to create a model of the probability p(oj) for each of the different possible classes. The PDF can be modeled using a Gaussian distribution. We can create a Gaussian model by just finding the sample Mean, and the sample covariance matrix Ui.

$$\begin{aligned} \mu_i &= \frac{1}{N}\sum_{n=1}^{N} o_n \\ U_i &= \frac{1}{N-1}\sum_{n=1}^{N}(o_n-\mu_i)'(o_n-\mu_i) \end{aligned} \tag{18}$$

$$\mathcal{N}(o;\mu,U) = \frac{1}{\sqrt{(2\pi)^p|U|}}\exp\left(-\frac{1}{2}(o-\mu)U^{-1}(o-\mu)'\right) \tag{19}$$

Probability of being in state S_i at time t, and state S_j at time t+1, given the model and the observation sequence, i.e.

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda). \tag{20}$$

2.4 Covariance selection in speech recognition

The covariance Matrix used in model based speech Recognition problem which uses N state Univariate Gaussian HMM modeling with M dimensional features can be considered in the following ways. The following 39 dimensional feature vectors are considered for designing the continuous HMM based speech recognizer.

[c1(t) c2(t)… cM(t), Δc1(t) Δc2(t)… ΔΔcM (t-1) ΔΔc1(t) ΔΔc2(t).. ΔΔcM (t-1), E(t) , ΔE(t)]^T

Where $\Delta C_m(t)$, $\Delta\Delta C_m(t)$ can be represented as below.

$$\Delta C_m(t) = \frac{\partial\, C_m(t)}{\partial t} \approx \left\{\mu * \sum_{k=-K}^{K} k * C_m(t+k)\right\} 0 \le m \le M \tag{21}$$

$$\Delta\Delta C_m(t) = \frac{\partial\, \Delta C_m(t)}{\partial t} \approx \left\{\mu * \sum_{k=-K}^{K} k * \Delta C_m(t+k)\right\} 0 \le m \le M \tag{22}$$

i. Complete covariance matrix (distance measure: Mahalanobis distance measure) Complete covariance matrix when considered results in very high implementation complexity and cannot be easily achieved with the existing hardware resources (S.Yozhizawa – 2006).

ii. The second method is to do covariance parameter tying (Pihl – 1996). In this a method a common parameter is considered for all the states and other statistical characteristics are considered different. For instance, use of common covariance matrix for all the

clusters obtained during GMM block quantization and considering mean, no of observation output different for each state.

iii. A still modified version of the above mentioned method having block covariance matrices instead of having complete covariance matrix can be considered for implementation which has complex implementation in hardware. Our assumption that the covariance are block diagonal is valid since the use of an orthogonal transform like DCT decorrelates the cepstral vectors. The correlation only exists between the time difference cepstral vectors, delta cepstral vectors, and the delta – delta cepstral vectors. So we can construct the covariance matrix as three element block diagonal matrix for which the inverse matrix can be easily found using Singular value decomposition.

iv. The last method is to consider the covariance matrix to be diagonal which yields the simplest hardware architecture. The inverse diagonal values are stored in memory locations and only multiply operations are performed and this method is computationally less intensive. Present hardware based recognizers implement this method due to its low complexity. But this method will produce significant errors and degrade the recognition performance of the system as it is doesn't efficiently represent the correlation introduced by the Vector quantizer. Earlier proposed implementations were based on this method only (Karthikeyan – ASICON 2007).Where E(.,.) represents the statistical Expectation operation on the cepstral vectors.

$$R=\begin{pmatrix} \begin{pmatrix} E(c_1,c_1) & E(c_1,\Delta c_1) & E(c_1,\Delta\Delta c_1) \\ E(\Delta c_1,c_1) & E(\Delta c_1,\Delta c_1) & E(\Delta c_1,\Delta\Delta c_1) \\ E(\Delta\Delta c_1,c_1) & E(\Delta\Delta c_1,\Delta c_1) & E(\Delta\Delta c_1,\Delta\Delta c_1) \end{pmatrix} & 0 & 0 \\ 0 & \begin{pmatrix} E(c_2,c_2) & E(c_2,\Delta c_2) & E(c_2,\Delta\Delta c_2) \\ E(\Delta c_2,c_2) & E(\Delta c_2,\Delta c_2) & E(\Delta c_2,\Delta\Delta c_2) \\ E(\Delta\Delta c_2,c_2) & E(\Delta\Delta c_2,\Delta c_2) & E(\Delta\Delta c_2,\Delta\Delta c_2) \end{pmatrix} & 0 \\ 0 & 0 & \begin{pmatrix} E(c_3,c_3) & E(c_3,\Delta c_3) & E(c_3,\Delta\Delta c_3) \\ E(\Delta c_3,c_3) & E(\Delta c_3,\Delta c_3) & E(\Delta c_3,\Delta\Delta c_3) \\ E(\Delta\Delta c_3,c_3) & E(\Delta\Delta c_3,\Delta c_3) & E(\Delta\Delta c_3,\Delta\Delta c_3) \end{pmatrix} \end{pmatrix} \quad (23)$$

Earlier we have discussed the influence of the covariance selection on the performance of the recognition and it directly influences the word error rate. Earlier implementation considers completely diagonal co variances which cause drastic errors as we have introduced correlation into the feature vectors through vector quantization as well as dynamic feature vector set. Hence we can consider the feature vectors to be correlated only among the two dynamic features set delta and delta delta feature vectors the static features. Hence we can assume the correlation matrix to be block diagonal and hence the inverse of such a matrix can be easily obtained by linear equation solvers. Computation of the Singular Value Decomposition of a matrix A can be accelerated by the parallel two sided jacobi method with some pre-processing steps which would concentrate the Frobenius norm near the diagonal. Such approach would help noise reduction is great way as the noise sub-space is computed with Frobenius norm constraints. Such a concentration should hopefully lead to fewer outer parallel iteration steps needed for the convergence of the entire algorithm.However the gain in speed as measured by total parallel execution time depends decisively on how efficient is the implementation of the distributed QR and LQ factorizations on a given parallel architecture.

3. Speech recognition architecture

3.1 NIOS embedded processor based system design

NIOS 2 is a soft Processor which can be realized in any of the Altera's FPGA Development kits. It is based on a 32-bit RISC architecture and is a natural choice in projects where CPU performance is essential.The NIOS processor can be run at different frequencies based on which the Computational capability of the processor can be choosen. Nios Processor is available in three different speed grades and can be extended with additional coprocessors, instruction sets, and so forth. By doing so, it is possible to develop a large part of the system on an ordinary PC running Windows or any variant of UNIX. By simulating IP cores (firmware modules) as software objects, a system can be developed to an advanced state before it needs to be tested on the actual target. Another benefit of this approach is that it allows concurrent development of multiple projects on a single target. The NIOS processor is a 32-bit Harvard Reduced Instruction Set Computer (RISC)architecture optimized for implementation in Altera FPGAs with separate 32-bit instruction and data buses running at full speed to execute programs and access data from both on-chip and external memory at the same time. Nios Processor has got 32 32bit general purpose registers and 16 32bit control registers, an Arithmetic Logic Unit (ALU),Exception Unit, Instruction cache and Data Cache, Hardware multiply and Hardware divide, a barrel shifter unit and 32 software interrupts. This flexibility allows the user to balance the required performance of the target application against the logic area cost of the soft processor. NIOS processor does not separate between data accesses to I/O and memory (i.e. it uses memory mapped I/O). All the system peripherals of Altera are connected through a system bus called Avalon. for sophisticated SOPC environment or the basic environment. The stack convention used in Nios processor starts from a higher memory location and grows downward to lower memory locations when items are pushed onto a stack with a function call. Items are popped off the stack the reverse order they were put on; item at the lowest memory location of the stack goes first and etc (NIOS 2006). Nios processor also supports reset, interrupt, user exception, and break and hardware exceptions. The processor will only react to interrupts if the Interrupt Enable (IE) bit in the Machine Status Register (MSR) is set to 1. On an interrupt the instruction in the execution stage will complete, one has to manually enable the interrupt enable bit for that particular device.

Writing software to control the NIOS processor must be done in C/C++ language. The NIOS tool has got gnu based have built in C/C++ compilers and debugger to generate the necessary machine code for the NIOS processor (Agarwal 2001). NIOS Processor supports word (32 bits), half-word (16 bits), and byte accesses to data memory. Data accesses must be aligned (i.e. word accesses must be on word boundaries, half-word on half-word boundaries), unless the processor is configured to support unaligned exceptions. All instruction accesses must be word aligned.

Avalon Bus system is a simple yet extremely powerful bus system which allows any no of Bus masters to be added simultaneously and offers excellent arbitration capabilities with wait cycles. It also supports a unique kind of hardware software interface called custom instruction which acts as a hardware mapped instruction to the NIOS processor (Avalon 2006). We can also accelerate the software function in NIOS processor through a technology called Custom instruction which is unique to NIOS based system. Nearly 256 custom instructions different cycle times can be integrated into the design to accelerate the underlying software. Compared to complete software performance, a system with hardware

acceleration improves 20X performance improvement. Our design utilizes this custom instruction feature (Avalon 2006). NIOS processor supports many software IP-cores such as Timer, Programmable counters, Ethernet controller, DRAM controller, Flash controller, User logic components,PLLs,Hardware Mutex, LCD controller etc. NIOS Timers can be used to compute the execution time of a software routine or used to produce trigger at regular intervals so as to signal some of the hardware peripherals. Hardware IP cores can be connected to the system in two different ways. The hardware component can be configured as Avalon Custom instruction component and the processor can access the hardware as though it being an instruction. NIOS processor supports four different kinds of custom instruction technology namely combinational; Multi cycle, Extended and Internal Register file based custom instruction. Custom instruction module can also be connected to the Avalon Bus so that one can connect some of the custom instruction signals to external signals not related to the processor signals. Hardware IP core can also be interfaced to the NIOS system through the Avalon slave or Master Interface. Avalon Slave devices can have interrupts and they request the service of the processor through the interrupts. These interrupts can be prioritized manually.

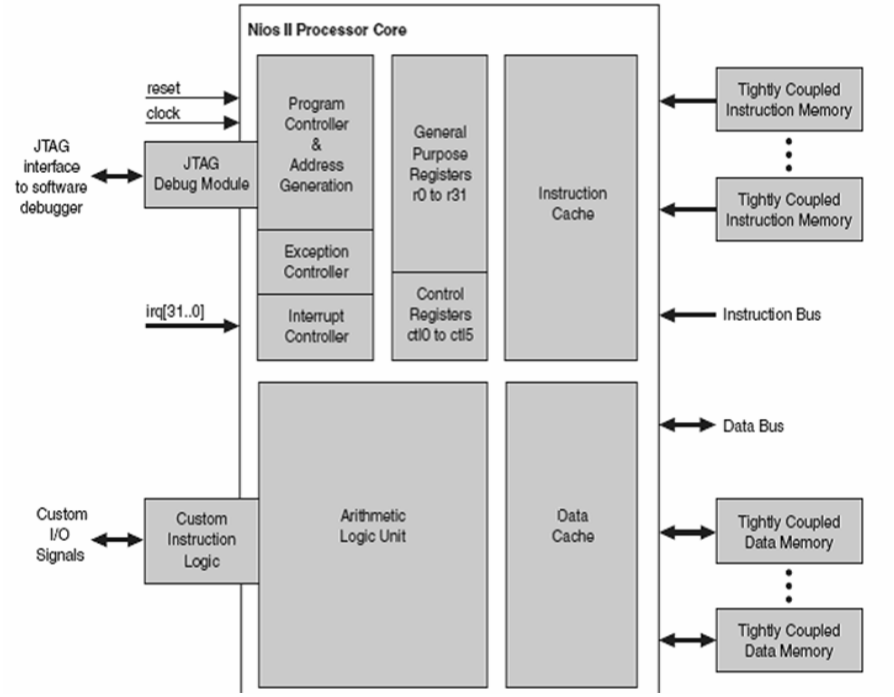

Fig. 5. NIOS Architecture

4. Design implementation using fixed point architecture

4.1 Finite word length effects

All DSP based designs strongly depend on the floating point to fixed point conversion stage as the DSP algorithm may not be implementable in floating point form. Fixed pint analysis

of the system is extremely important to understand the nonlinear nature of the quantization characteristics. This leads to certain constraints and assumptions on quantization errors: for example that the word-length of all signals is the same, that quantization is performed after multiplication, and that the word-length before quantization is much greater than that following Quantization (Meng 2004). Error signals, assumed to be uniformly distributed, with a white spectrum and uncorrelated, are added whenever a truncation occurs. This approximate model has served very well, since quantization error power is dramatically affected by word-length in a uniform word-length structure, decreasing at approximately 6dB per bit. This means that it is not necessary to have highly accurate models of quantization error power in order to predict the required signal width. In a multiple word-length system realization, the implementation error power may be adjusted much more finely, and so the resulting implementation tends to be more sensitive to errors in estimation. Signal-to-noise ratio (SNR), sometimes referred to as signal-to-quantization noise ratio (SQNR), is The ratio of the output power resulting from an infinite precision implementation to the fixed-point error power of a specific implementation defines the signal-to-noise ratio In order to predict the quantization effect of a particular word-length and scaling annotation, it is necessary to propagate the word-length values and scaling from the inputs of each atomic operation to the operation output (Haykin 1992). The precision of the output not only depends on the binary precision of the inputs, it also depends on the algorithm to be implemented. For example the fixed point implementation of complex FFT algorithm decreases 0.5 bit precision for each stage of computation (Baese 2005). So for large FFT lengths more bits of precision are lost. The Feature extraction stage was implemented in Nios Processor with fixed precession inputs. The following plots describe the fixed point characteristics of the algorithm.

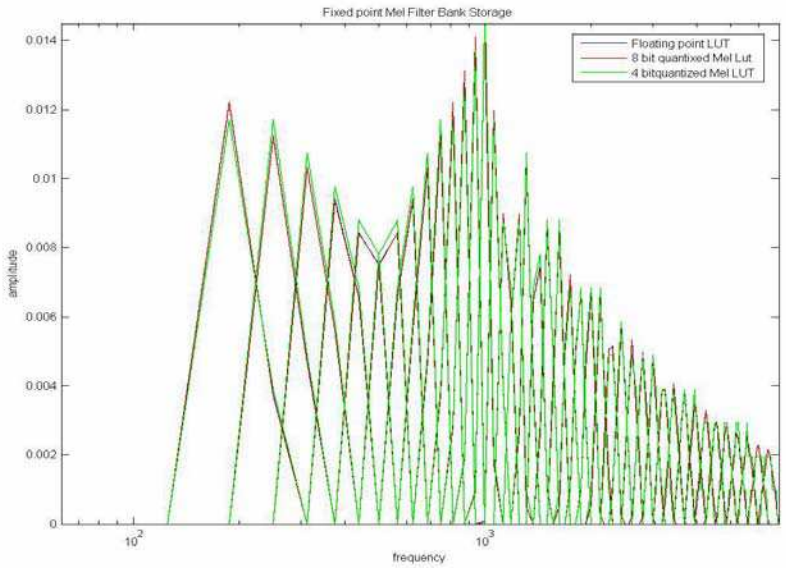

Fig. 6. Fixed point MFCC implementation

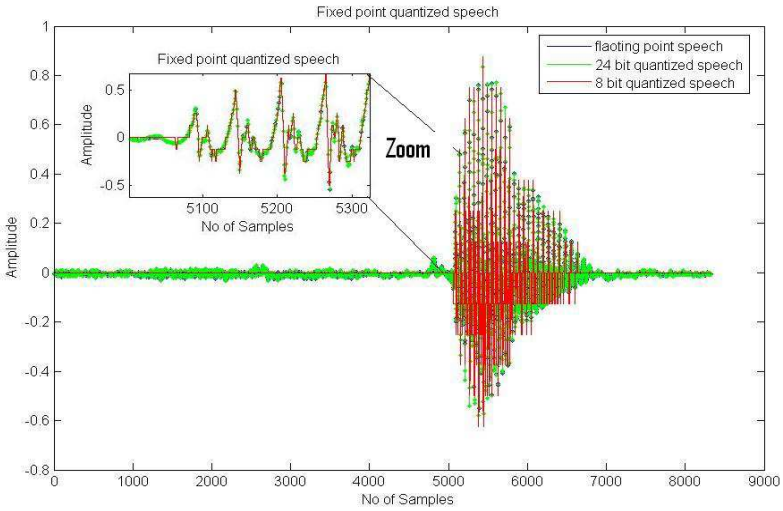

Fig. 8. Fixed point Speech input charactersistics

4.2 Flexibility

The recognition system must be able to operate under a variety of conditions (Vaseghi 2004). The signal to noise ratio may vary significantly, the word may be stretched too long or too short, some of the states may be skipped, noise content may be high and we are forced to model the noise HMM and subtract it from the actual speech HMM (Hermus 2007).The receiver must incorporate enough programmable parameters to be reconfigurable to take best advantage of each situation (Press 1992). We applied signal subspace based noise reduction algorithm based on Singular Value decomposition to reduce the noise characteristics of the speech signal (Hemkumar 1991).

4.3 FCTSVD algorithm (Abut 2005):

1. Estimate the noise vectors W from silence periods in the observed speech signal.
2. Form the Hankel matrix Hy from the observed speech signal.
3. Compute SVD of Hy.
4. Initialize the oreder of retained singular values of Hx−
5. Let S=S+1 and reconstruct the estimated matrix of Hx, Hx− using the first s eigen values.
6. Compute Frobenius Constrained norm metric and error is less than 0.0098 else goto 5.

4.4 Scaling:

As the number of frames in speech increases, the values of the variable the formal algorithm saturates whereas the log-Viterbi algorithm used in this hardware does not suffer this problem as the implementation involves only additions rather than multiplication.

4.5 Initial estimates of HMM parameters:

Uniform initial estimates were used for A and Pi Matrices. However B matrix cannot be initialized with random values as it has more influence in convergence of the Baum-Welsch

algorithm. Since continuous hidden markov models are used, the initial estimates of B, Mean, and Variance are obtained using segmental K-means algorithm..

5. Project modules:

5.1 Main modules:

1. First module is concerned with signal analysis and feature extraction (FRONT END PROCESSING → SOFTWARE EXECUTED IN NIOS 2 PROCESSOR).
2. The next module generates the values of parameters required for comparing the test speech signal with the reference signal values .Training phase and this step should be robust since it directly determines the accuracy and the application where the system is to be deployed. (TRAINING – OFFLINE DONE IN MATLAB-refer Figure 13).
3. Maximum Likelihood based word recognition (PARALLEL HARDWARE).

5.2 Supporting hardware modules:

1. Audio Codec Configuration unit based on I2C controller (MPU 2 AUDIO CODEC CONFIGURATION).
2. Custom Avalon Master Module for Audio Codec data Retrieval with integrated SRAM memory controller.
3. Custom Speech controller for hardware recognition part with efficient mode management unit based on FSMs.
4. The speech Controller has got the following modules built in:
5. Viterbi based Speech Recognition unit with memory controllers for Model parameter RAMS.
   a. Input Frame buffers for feature storage with memory controller for feature storage RAM.
   b. Output Frame buffer for Model output storage
   c. Efficient mode management unit to switch between various modes of operations using FSMs.
   d. LED Display unit to finally display the results.
6. Custom Singular Value Decomposition unit.

Software Modules	Hardware Modules interfaced to Avalon	Custom Instructions
✓ Voice Activity Detection ✓ FFT based feature extraction ✓ Mel Filter Banks ✓ Cepstral Weighing ✓ Software Back-substitution	✓ Audio Serial 2 Parallel Module(Avalon Master) ❖ Input Frame buffers ❖ Output Frame Buffers ❖ Speech Recognition Mode controller top ❖ for SVD Speech Recognition module with Viterbi.	• Max comparator for magnitude computation in FFT

Table 2. Isolated Word Recognition System Hardware/ Software Partition

- **Operating frequency of green = 18.432MHz**
- **Operating frequency of red= 12.5MHz**

✓ **Operating frequency of 50 MHz**

Audio codec is configured via I2C interface. Two signals I2C_clock and I2C_data are used to configure the internal registers of the WM8371. The WM8731 is a WRITE-ONLY device; any requests to read are ignored. Device is configured by writing data to internal registers. The internal registers are configured by transferring data and address of the internal registers serially through I2C_data pin. Clock signal is applied to the I2C_clk pin. Clock signal can be generated in two modes of operation namely USB/Normal mode master clock (AUD_XCLK, from which AUD_BCLK is generated).USB mode must have a FIXED AUD_XCLK of 12MHz, which can be easily obtained from a PLL in the SOPC system. Normal mode requires AUD_XCLK clocks of either 12.288MHz (8kHz, 32kHz, 48kHz, 96kHz) or 11.2896MHz (44.1kHz, 88.2kHz).This implementation utilizes Normal mode of clock generation at 18.432MHz. Transfer is initiated by pulling MPU_DATA low while MPU_CLK is high. The data format of the configuration of a particular internal register has got 3-bytes.

- Byte 1: {ADDR[6..0],0} →ADDR[6..0] is DEVICE ADDRESS, which is ALWAYS 0x34
- Last bit is R/W bit, which is always 0 (write,) since WM8731 is write-only
- Byte 2: {REG[6..0],DATA[8]} →REG[6..0] is 7-bit register address, DATA[8] is MSB of 9-bit DATA
- Byte 3: DATA[7..0] → Lower 8 bits of 9-bit DATA
- MPU_DATA is driven low by the CODEC between bytes as confirmation

The following operations needed to be done to make the device operate in the intended mode of operation:

- Reset device: Write 0x00 to AUDIO_RESET
- Power up device: Write '0' to WM8731_POWER_DOWN_CTL.7 bit
- Disable Line IN -> Line OUT bypass, select MIC_IN (AUDIO_ANALOG_PATH_CTL Reg)
- Turn on MASTER mode: AUDIO_INTERFACE_FMT

5.3 How this hardware system works:

The steps involved in implementing this system consist of the following steps given below:

Step 1: Audio Codec is configured via CPU 2 I2C interface with the following specifications.

- ✓ **WM8731_POWER_DOWN_CTL** is used to power up the device**.**
- ✓ **WM8731_ANALOG_PATH_CTL** Register is set to 16'h08FD to enable the MIC in facility.
- ✓ **WM8731_SAMPLING_CTL** Register is set to 16'h100E to fix the audio codec in NORMAL MODE with ADC sampling frequency of 8 KHz. Codec operating frequency is 18.432MHz

Step2: The serial input bit stream is converted in parallel data using a custom Avalon Master interface and is stored in SRAM module. The storage of audio will be interrupted by a external user controlled switch to start the processing step.

Step3: This switch will induce an interrupt signal present in the speech recognition module (AVALON SLAVE CONFIGURED) to start the feature processing of NIOS processor.

Step4: In software the speech start and end points are detected, we perform windowing (Hamming Window).

Step5: We use short time Fourier analysis on the speech signal, since speech is a Quasi-periodic signal we need to use STFT. We have used a window of duration 30ms with an overlap of 10ms.

Step6: Evaluate the distance between the speech signals and do clustering using the Gaussian Mixture based Block quantizer based on Mahalanobis distance and clustering is performed.
Step7: The features are extracted and stored in the **INPUT FRAME BUFFER** of the Speech Recognition module.
Step8: Steps 1 to 6 will continue until the end of frame is detected by the hardware module.
Step9: Starting of speech recognition in hardware and finally the results are populated and displayed in LED. Each stage output is stored in **OUTPUT FRAME BUFFER** and final recognition is done.

5.5 Implementation of continuous hidden Markov model:

Our architecture concentrates on the three major issues Power, Memory access (Throughput) and vocabulary size. There is always a trade off existing between the operating frequency and the recognition vocabulary, word accuracy, noise suppression etc. This is a word HMM based architecture which uses continuous HMM for the implementation.
Two essential steps in the recognition algorithm are:
1. Output probability calculation.
2. Log VITERBI implementation.

→Output Probability calculation is the computationally intensive process as we need to do lots of multiplies and Add operations.
→Viterbi Algorithm is also implemented as a parallel processing block for faster recognition.

5.6 Hardware design:

Our architecture (Fig 11) concentrates on the three major issues Power, Memory access (Throughput) and vocabulary size. There is always a trade off existing between the operating frequency and the recognition vocabulary, word accuracy, noise suppression etc (Pihl 1996). This is a word HMM based architecture which uses continuous HMM for the implementation (Cho 2002).
Two essential steps in the recognition algorithm are:
1. Output probability calculation.
2. Log Viterbi implementation(as in fig 12).

5.7 Modes of operation:

We can operate the system in two modes:
1. Small vocabulary mode
2. Large vocabulary mode

5.8 Small vocabulary mode:

Startup Sequence:
1. *Reset* = 0, sw0= 0, sw1=0
2. Audio is stored in SRAM by custom master Avalon interface.
3. When the user presses switch0 the Avalon master stops storing samples.
4. Speech controller interrupts processor for features after sw0 is pressed.
5. Processor starts processing the samples to extract features and once is complete activates done signal of Speech controller.