

Software/Hardware Co-Design of HMM Based Isolated Digit Recognition System

V. Amudha, B.Venkataramani, R. Vinoth kumar and S. Ravishankar
 Department of ECE, National Institute of Technology, Tiruchirappalli - 620015, India.
bvenki@nitt.edu

Abstract – In this paper, the design and implementation results of a system on a chip (SOC) based speech recognition system as software/hardware co-design is presented. The hidden markov model (HMM) is used for the speech recognition. In order to implement this in SOC, the various tasks required are optimally partitioned between hardware and software. The SOC, housed in Altera FPGA boards, has Nios II soft core processor. Custom hardware blocks are developed for computationally intensive blocks such as Viterbi decoder. The preprocessing and training of HMM are implemented in software (using C program). The Viterbi decoding is implemented in hardware as custom block for real time recognition. It is also implemented in software for verification and comparison. It is observed that the sequential hardware implementation of viterbi block is 80 times faster than the software approach using C program with UP3 kit. An over all recognition accuracy of 94.8% is achieved for speaker independent digit recognition for our own database of 6 speakers. Altera's DE2 board with cyclone II FPGA is used to implement TI46 digit recognition. Since the logical elements in DE2 board is high compared to UP3 kit the viterbi decoding is implemented in parallel for 0-9 digits. Because of this speed of recognition is '772' times faster than software implementation with cyclone II FPGA. And also it is observed that for TI-46 speech database for f1 speaker the recognition accuracy is 87% using LPC as feature extraction technique. Extension of this work for larger vocabulary size and using MFCC as feature extraction is under progress.

Index terms – Software/hardware co-design, Custom hardware, SOC, HMM

I. INTRODUCTION

Speech recognition system has been developed for many real world applications, often using low cost speech recognition software. However, high performance and robust isolated word recognition system in hardware, particularly for digits, is still useful for many applications such as recognizing telephone numbers, and for use by physically challenged persons. The most successful approach to speech recognition is based on the Hidden Markov Model (HMM) [1] [2]. It is a probabilistic process, which model spoken utterances as the outputs of finite state machines (FSMs). A number of HMM based speech recognition systems implemented in PC are discussed in the literature [3] [4]. The algorithm demands increased computing task on the host computer. Hence, as in the case of graphics, it may be advantageous to transfer

speech processing to some form of co-processor or other hardware implementation.

Efficient implementation of complete System On Chip (SoC) got an impetus with the advent of multi-million gate FPGAs. FPGAs can be efficiently used for performing digital signal processing operations and other computation intensive tasks. Microcontroller or RISC processor is efficient for performing control and decision-making operations.

Integration of block RAMs as look up tables (LUTs) in an FPGA and the availability of soft core processors such as Nios II and Micro blaze for FPGAs, enable the best of both software and hardware to be gainfully obtained for an application using SOC.

FPGA implementation of speech recognition system has been already reported in the literature [5] [6]. However, the feasibility of designing systems on programmable chips offers an attractive alternative. In this paper, we describe the implementation of the decoder part of a speech recognition system on a SOC, targeting in particular, the soft-core processor NIOS II.

The paper is organized as follows: An overview of speech recognition system is discussed in section 2. Introduction to HMM and viterbi decoding algorithm are presented in section 3. NIOS II custom instructions and SOPC builder are discussed in section 4. Section 5 gives the implementation details and section 6 presents the results and conclusions.

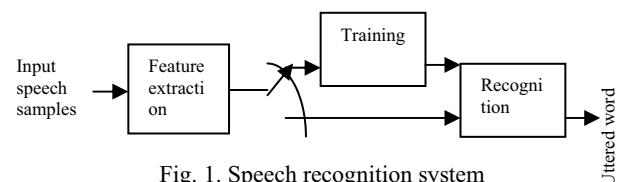


Fig. 1. Speech recognition system

II. AN OVERVIEW OF THE SPEECH RECOGNITION SYSTEM

The block diagram of the speech recognition system is shown in Fig.1. It is divided into three modules: preprocessing/feature extraction, training and recognition. The feature extraction block transforms the input speech into spectral components. The recognition block uses the spectral features and compares with the trained model features. LPC10, MFCC are generally used as feature extraction techniques. Recognition can be based on a

wide variety of techniques such as Dynamic Time Warping (DTW), Hidden Markov models (HMM) or Neural Network (ANN). Since the recognition accuracy is more for HMM, it is widely used in speech recognition. Hence, HMM is used for speech recognition system in this paper. For separating the input samples from background noise, energy and zero crossings rate based voice activity detection (VAD) is used. Linear predictive analysis (LPC-10) followed by cepstral analysis is used for feature vector extraction. Baum Welch algorithm is used for training the speech models and Viterbi decoder is used for recognition. To reduce the memory for feature vector storage, vector quantization is used.

III. INTRODUCTION TO HMM

A HMM, in simple terms is a model which is used to model a system about which we know nothing except its input and output sequences. A number of HMM models are proposed in the literature [2] such as left right, cyclic etc. We assume the left right model (Bakis model) in this paper. We train the HMM so that it produces an output which closely matches the available output sequence. A HMM is characterized by

(i) N , the number of states in the model. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or to sets of states of the model.

(ii) M , the number of distinct observation symbols per state, i.e., physical output of the system being modeled.

(iii) The state transition probability distribution $A = \{a_{ij}\}$ where $a_{ij} = P[q_{t+1} = j | q_t = i]$, $1 \leq i, j, \leq N$, a_{ij} can be greater than or equal to 0.

(iv) The observation symbol probability distribution in state j , $B = \{b_j(k)\}$, where $b_j(k) = P[V_k \text{ at } q_t = j]$ $1 \leq j \leq N$, $1 \leq k \leq N$

(v) The initial state distribution $\pi = \{\pi_i\}$ where $\pi_i = P[q_1 = i | 1 \leq i \leq N]$

The speech recognition problem is:

Given an observation sequence $O = O_0 O_1 O_2 \dots O_{T-1}$ where each O_t is data representing speech which has been sampled at fixed intervals, and a number of potential models M , each of which is a representation of a particular spoken utterance (e.g. word or sub-word unit), find the model M which best describes the observation sequence, in the sense that the probability $P(M|O)$ is maximized (i.e. the probability that M is the best model given O). Viterbi algorithm, used for recognition, is as follows:

Viterbi Algorithm:

To find the single best state sequence,

$Q = \{q_1, q_2 \dots q_T\}$, for the given observation sequence

$O = (O_1, O_2 \dots O_T)$, we need to define the quantity

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_t} P[q_1, q_2, \dots, q_t = i, O_1, O_2 \dots O_t | \lambda]$$

i.e., $\delta_t(i)$ is the best score (highest probability) along a single path, at time t , which accounts for the first t observations and ends in state i . By induction we have

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}).$$

To actually retrieve the state sequence, we need to keep track of the argument that maximized $\delta_{t+1}(j)$, for each t and j . We do this via the array $\psi_t(j)$. The complete procedure for finding the best state sequence can now be stated as follows:

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N$$

$$\psi_1(i) = 0$$

2) Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T; \quad 1 \leq j \leq N$$

$$\psi_t(j) = \operatorname{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 2 \leq t \leq T; \quad 1 \leq i \leq N; \quad 1 \leq j \leq N$$

3) Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

$$q_T^* = \operatorname{argmax}_{1 \leq i \leq N} [\delta_T(i)]$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T - 1, T - 2, \dots, 1.$$

For training the HMM for multiple speakers, the HMM parameters corresponding to each speaker is averaged. Compared to Rabiners's [2] approach, this has a number of advantages such as lower data requirement, higher detection accuracy and lesser computation complexity.

Feature extraction, training and other pre processing stages of HMM are implemented in software (C/C++) in the offline mode and the recognition process is implemented in the SOC as online process. For the recognition, viterbi decoder is implemented in hardware and the rest of the blocks such as feature extraction, computation of the elements of observation matrix are realized using the NIOS II soft core processor in the SOC.

IV. INTRODUCTION TO NIOS II AND SOPC BUILDER

The NIOS II processor system is equivalent to a microcontroller or computer on a chip that includes a CPU and a combination of peripherals and memory on a single chip. It consists of Nios II processor core, a set of on-chip peripherals, on-chip memory, and interfaces to off-chip memory, all implemented on a single Altera FPGA. Like a microcontroller family, all Nios II processor systems use a consistent instruction set and programming model. Designers can easily add Nios II processors to their systems by using the SOPC builder tool featured in Altera's industry leading Quartus II design software.

A. Custom Instructions for NiosII Embedded Processor

With the Altera NiosII embedded processor, time critical software algorithms can be accelerated by adding

custom instructions to the Nios II instruction set. The NiosII CPU configuration wizard, which is accessed via the QuartusII software's SOPC builder, provides a graphical user interface (GUI) used to add up to 256 custom instructions to the Nios II processor.

A complex sequence of standard instructions can be reduced to a single instruction implemented in hardware using custom instructions. Custom instructions consist of two essential elements: a *Custom logic block*, the hardware that performs the user-defined operation, and a *Software macro*, which allows the system designer to access the custom logic through software code. The task performed by the custom block may be defined either as single-cycle combinatorial operation or as multi-cycle sequential operation as shown in fig.2. In both cases, two 32-bit operands (dataa, datab) may be passed to custom block and a 32-bit result is returned. The CPU clock is made available to the custom block only when it is defined to be sequential.

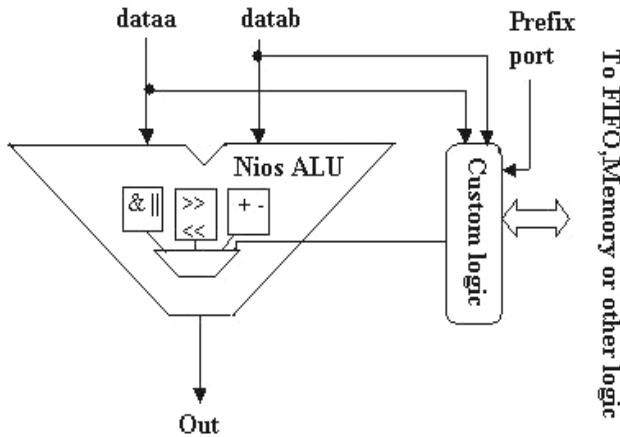


Fig. 2. Adding Custom Logic to NIOSII CPU

V. IMPLEMENTATION

In this paper, HMM is trained for five utterances of three speakers, each uttering the digits 0 - 9. For the implementation, the Altera UP3 kit with EP1C6Q240C8 is used. Implementation details of various blocks are discussed in this section.

A. Feature Extraction

The block diagram of the feature extraction system is shown in Fig. 3. Speech is sampled at the rate of 8KSPS and stored as 16 bit numbers. By applying energy and zero crossing rate algorithms, the starting and ending points of the speech are found. Preprocessing involves low pass filtering the speech samples to enhance the low frequency samples. Speech frames are formed with duration of 30msec and with an overlap of 10msec between adjacent frames. Hamming window is used to minimize the signal discontinuities at the beginning and end of frame. Auto correlation coefficients are computed from the windowed frame. Using Levinson & Durbin's algorithm, LPC analysis is carried out to convert the autocorrelation coefficients to LPC parameter set. Cepstral coefficients are obtained from the LPC coefficients by employing the conversion formula given in [2]. Cepstral weighting is done by multiplying the cepstral coefficients with the window function.

B. Vector quantization

For a HMM with discrete observation symbol density, a vector quantizer (VQ) is required to map each continuous observation vector (feature vectors) into a discrete codebook index. Once the codebook of vectors has been obtained, the mapping between continuous vectors and codebook indices becomes a simple nearest neighbour computation, i.e., the continuous vector is assigned the index of the nearest (in a spectral distance sense) codebook vector. Thus the major issue in VQ is the design of an appropriate codebook for quantization. Binary split algorithm is used for codebook generation. Codebook size of 128 was considered.

C. Training the HMM

For training the HMM model, Baum-Welch iterative algorithm [1] was used. As shown in fig. 4 indices of the codebook are fed as the input to this algorithm and the iteration limit is chosen as 20. Corresponding to each

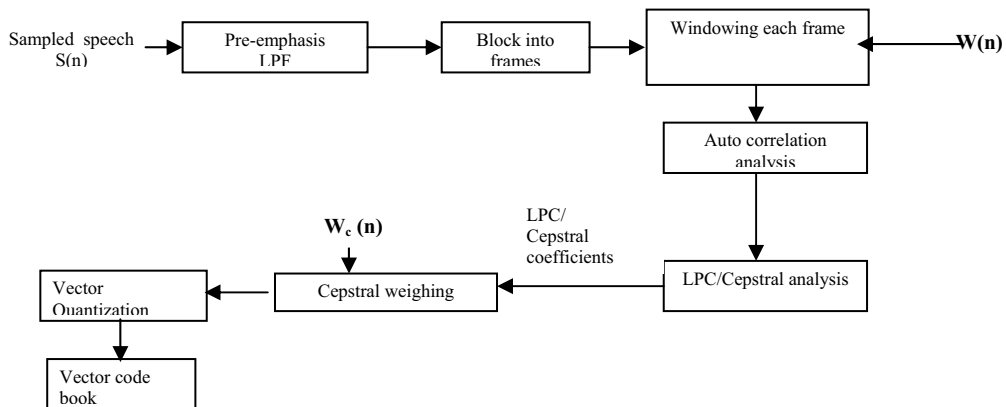


Fig.3. Block diagram of feature extraction

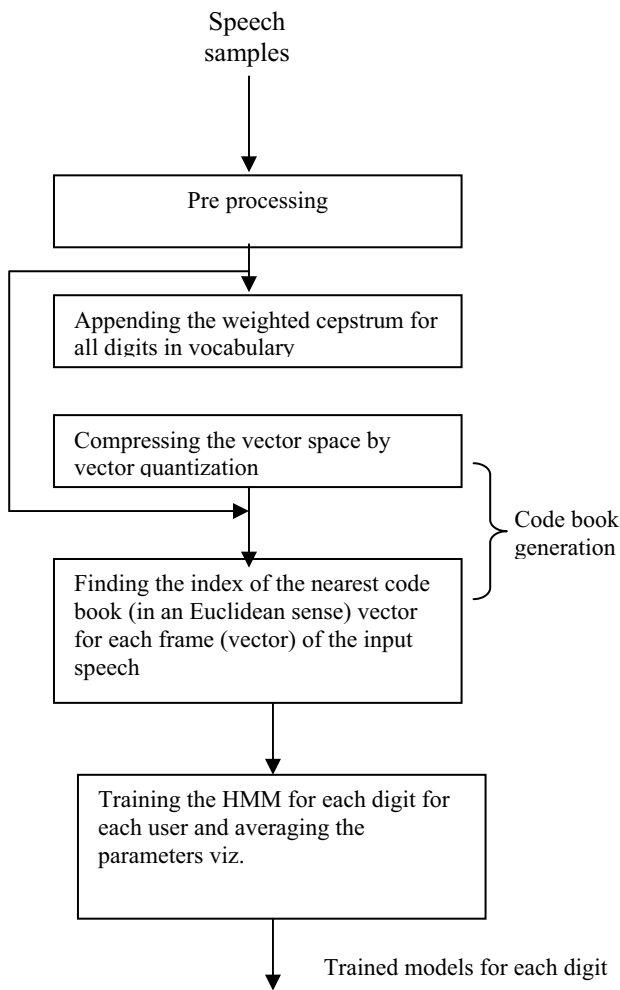


Fig.4 Block diagram of Training

digit, the A, B and π model parameters are computed for each user. The A, B and π matrices of all three users for each digit is averaged to generalize it. In order for the design to faithfully recognize the same digit uttered by another user for which the design has not been trained, the zero probabilities in B matrix have been replaced by a low value so that on recognition, it gives a non-zero value. This overcomes the problem of less training data to some extent.

D. Hardware Implementation

Preprocessing and training are done in off line in “C” programming. The viterbi decoder is implemented in hardware and it is interfaced to NIOS II as custom instruction. The elements of A and B matrices are passed through the “dataa” and “datab” ports and the prefix port is used for the control operations.

The on chip memory of EP1C6Q240C8 is only 128KB and hence only one of the A and B matrices corresponding to one particular digit is stored in the on chip memory. For computation of probability using viterbi decoder, the A, B matrices corresponding to each digit is read from the hard disk and then stored into the

on-chip memory used by the custom hardware. Use of FPGAs with higher on-chip memory can circumvent the sequential approach.

Since the number of logical elements are around 33K for cyclone II FPGAs. We used cyclone II for implementing parallel viterbi decoding . In which all the A,B and phi matrices for all 10 digits is stored in on-chip memory. Only codebook indices of the recognizing digit was passed in dataa and datab ports as quadruplets to reduce the number of cycles for passing custom instruction.

All the floating point values are converted to fixed point representation before passing to the hardware block for efficient computation. Since Bakis model is used, which is a left right model, the zero valued B matrix entries are not passed to save the time and memory. Prefix ‘0’ is used for writing all matrix entries and prefix ‘1’ is used for execution of the algorithm. To minimize the computation time for viterbi algorithm, the multiplication with B matrix is replaced with an equivalent logarithmic operation which requires only addition. Table I, gives the computation time required for multiplication and addition. From the table, it is observed that a speedup by a factor of 4 is achieved by logarithmic addition technique.

Table I : Comparative analysis for multiplication & addition

	Execution time	Number of LEs
Floating point Multiplication	5.48 sec	28,938
Addition	1.3 sec	11,875
Improvement factor	4 times	41% less consumption

E. Recognition

The input speech sample is preprocessed to extract the feature vector first. Next, the index of the nearest codebook vector for each frame of the feature vectors is found. The codebook indices are sent to all the HMM models, and the output probabilities are calculated by using viterbi algorithm from all the models. Out of these, the model giving the maximum probability is chosen as the uttered word. Fig.5. describes the recognition process.

VI. RESULTS & CONCLUSION

With UP3 kit, five utterances of each digit from 3 male speakers are used for training. Five utterance of the same speakers as well as three new speakers are used for recognition. It was found that 100% recognition is achieved for the recognition of trained samples. The recognition performance for new utterances of the same 3 speakers and the utterance of 3 untrained male speakers are given in fig.6. It is found that trained speakers recognition performance is better than the untrained speaker’s recognition. And the overall recognition rate of 94.8% is achieved with speaker independent speech

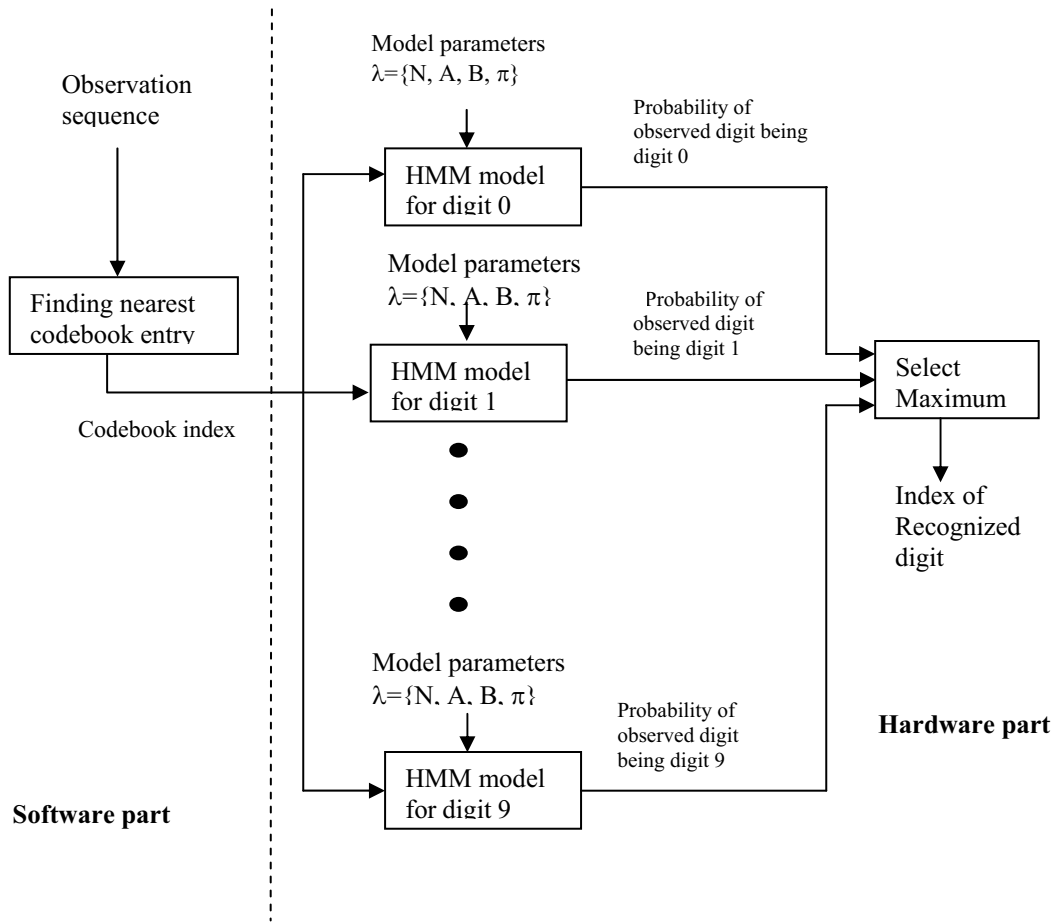


Fig.5. HMM recognition system

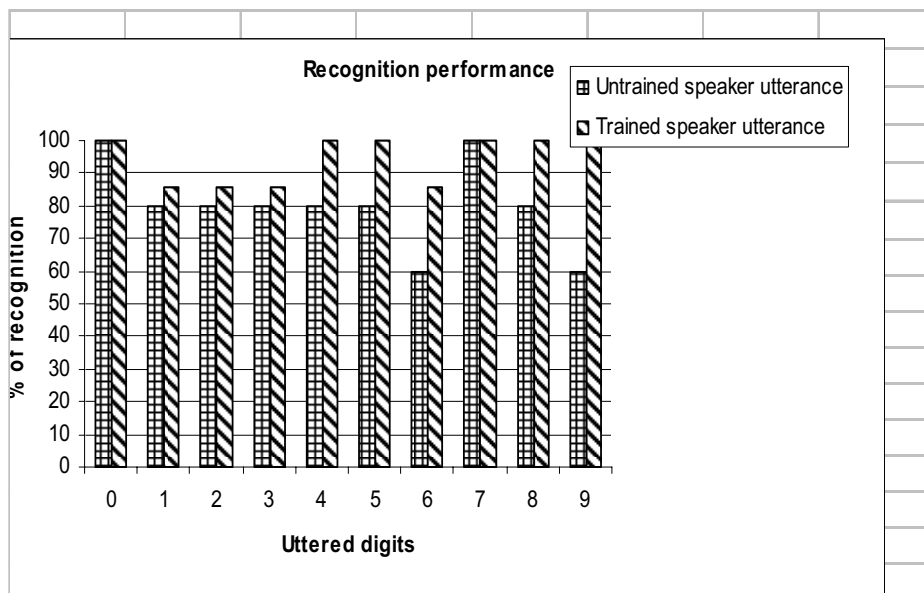


Fig.6. Recognition Performance for 6 speaker database

recognition system. Viterbi block is implemented in both hardware and software. The number of cycles taken for implementing viterbi algorithm by C programming is

80,00000 and viterbi as custom hardware is 1, 00000. From that it is found that viterbi block as custom hardware is 80 times faster than software implementation.

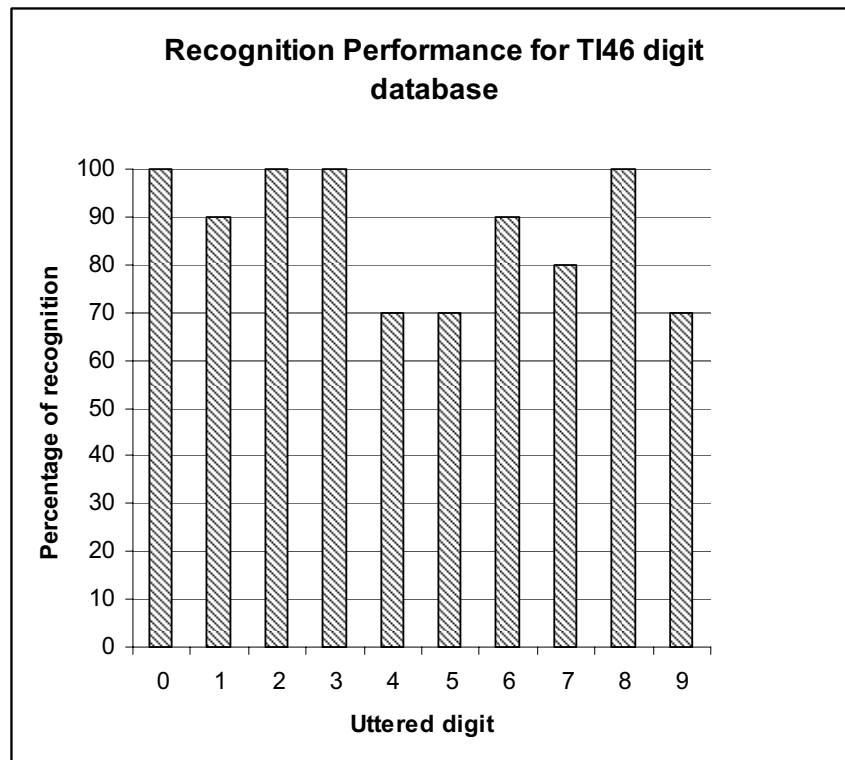


Fig. 7. Recognition Performance for TI46 digit F1 database

Since on-chip memory is less for UP3 kit sequential viterbi processing is done with UP3 kit. In order to circumvent this we used DE2 board with cycloneII FPGA. In which parallel execution of viterbi is carried over for all digits at the same time. Hence with this parallel implementation there is '772' times increase in speed is achieved with hardware implementation. TI46 digit database was used and the recognition accuracy for 'f1' female speaker is shown in fig. 7. Table 2 gives resource utilization details for implementing viterbi decoding as custom hardware. SOC implementation of ANN based speaker dependent isolated word recognition is discussed in [7] with recognition accuracy of 95%.

Study of the speech recognition system with MFCC as feature extraction scheme is under progress. Extension of the vocabulary size using phoneme based recognition scheme is also under progress.

Table 2 . Resource utilization details

	Logic elements	Memory bits
Sequential approach(UP3 kit)	3356	57,440
Parallel approach (DE2 board)	11849	3,50,392

ACKNOWLEDGEMENTS

This work is supported by the Department of Science and Technology, Government of India under the women

scientist scheme. Sanction order No. SR/WOS-A/ET-59/2004

REFERENCES

- [1] Rabiner , L.R ,” A tutorial on hidden markov models and selected applications in speech recognition” proceedings of IEEE, vol. 77, No.2 , 1989, pp 257-286.
- [2] Lawrence Rabiner & Bing-hwang juang,”Fundamentals of speech recognition”, prentice hall PTR, 1993.
- [3] www.htk.eng.cam.ac.uk
- [4] S.Neffi,O.Boefferd, “ Acoustical and topological experiments for an HMM based speech segmentation system”, in proceedings of Eurospeech pp1711-1714, 2001.
- [5] S.J. Melnikoff, S.F. Quigley , M.J. Russell “Implementing a simple continuous speech recognition system on an FPGA”, proceedings of the 10th annual IEEE symposium on Field programmable custom computing machines (FCCM'02), 2002
- [6] Vaidyanathan.A, Lakshmi Prabha.V,”Speech recognition of monosyllables using hidden markov model in VHDL”, proceedings of TENCON 2004, pp 76-79.
- [7] V. Amudha , B.Venkataramani, J. Karthick and C. Praveen , “SoC implementation of the neural network based isolated word recognition ,, proceedings of VDAT2006.pp 130 – 138
- [8] <http://www.altera.com>