Architecture for Low Power Large Vocabulary Speech Recognition

Dhruba Chandra, Ullas Pazhayaveetil, Paul D. Franzon
Department of Electrical and Computer Engineering
North Carolina State University, Raleigh, NC 27695
{dchandr, ucpazhay, paulf}@ncsu.edu

ABSTRACT

This paper proposes an architecture for real-time large vocabulary speech recognition on a mobile embedded device. The speech recognition system is based on Hidden Markov Model (HMM), which involves complex mathematical operations such as probability estimation and Viterbi decoding. This computational nature makes it power hungry and real-time recognition is not achieved by porting software solutions on embedded device. Our system architecture has a low power embedded processor and dedicated ASIC units for complex computations. These units operate at a low frequency of 50MHz thus consuming low power. The system uses RAM for the intermediate values and flash memory to store acoustic and language models for speech recognition.

I. INTRODUCTION

There is a growing demand for large vocabulary continuous automatic speech recognition from automated customer service to 'speech to text' for emails on cellphones. Real-time speech recognition is used to control unmanned vehicle from a distance, interactive video games and for a universal translator [1] to communicate in foreign languages. Speech can be a useful interface for mobile embedded environment. Present day speech recognition fails to perform real-time recognition in low power budget of mobile devices.

Speech recognition systems are based on Hidden Markov Model (HMM). It involves floating point computations for multivariate Gaussian distribution probability calculation. Majority of the research in speech recognition provides software solutions (CMU-Sphinx, Cambridge HTK etc.) and are based on HMM and designed for desktop or larger platforms. Earlier research shows [3] that Sphinx barely shows real-time performance using present day computers. Moreover, there is resource contention with other application running along with speech recognition software, hurting performance of both applications. To achieve real-time performance, threshold values are introduced to reduce the amount of computation which in-turn reduces the accuracy of recognition.

The software solutions, running on general purpose processors, are not particularly designed to be power efficient and it is not feasible to run them on a battery powered mobile device. These solutions are not aware of the underlying architecture.

[1] universal translator is coined from the famous TV series - Star Trek

Our proposed architecture has a general purpose processor which executes the less computation intensive part of speech recognition implemented in software and dedicated hardware units for intensive floating point computations in Gaussian probability calculation and Viterbi decoding. It interfaces with memory where the acoustic and language models are stored.

The rest of the paper is organized as follows. Section II gives an overview of HMM based speech recognition theory. Section III details the architecture of our speech recognizer. Section IV presents the results. Section V briefly describes related work. Finally, Section VI summarizes the findings.

II. HMM BASED SPEECH RECOGNITION

Speech recognition is most successfully done using HMM. In HMM based speech recognition theory, each word is represented as a sequence of basic sounds called *phones*. For Example, there are 51 phones in English language. The contextual effects cause variation in the way the basic sounds are produced. Each of the phones along with its neighboring phones (left and right) are called *triphones*. For each phone and triphone, there is a corresponding statistical model called hidden Markov model. HMMs are sequence of states called Markov states, connected by probabilistic transitions (called *transition probability*). Each triphone has one or more exit states. The exit state of one triphone is merged with the entry state of another to form composite HMM, which allows to represent a word and the words joined together to form a complete sentence. The states in triphones are best represented by multivariate mixture Gaussian distribution. In absence of enough training data, the states of different triphones are represented by the same distribution [2], these are called *senones*. Therefore, combination of senones forms triphones, which put together form words and words put together form a sentence or utterance.

The speech is first sampled and processed by the *Frontend*, where spectral analysis is done to extract the acoustic feature vectors. If the observation sequence of feature vector is $O = O_1, O_2, O_3, ..., O_T$, where O_i represents a feature vector sampled at interval i, the speech recognition decoder (the decoder is the part which does the actual recognition) finds the sequence of HMM states which is most likely to produce the observation sequence O. The probability that a HMM in state j emits observation O_t is called *observation probability* (also called *senone score*) and is represented by $b_j(O_t)$. The

0-7803-9782-7/06/$20.00 ©2006 IEEE

joint probability of observation of vector sequence O and state sequence $X = x_o, x_1, x_2...x_{T-1}$, given some model M (collection of HMMs) is calculated as product of the transition probabilities (a_{ij}'s) and the observation probabilities.

$$P(O, X/M) = b_0(O_0)\Pi_{t=1}^{T-1} b_{x_t}(O_t) a_{x_{t-1}x_t} \quad (1)$$

assuming the HMM is in state 0 at time $t = 0$. In practice, only observation sequence O is known and underlying state sequence is hidden. The probability $P(O/M)$ is determined by the probability associated with the state sequence which maximizes $P(O, X/M)$. This is calculated using a *Viterbi decoder*. We define $\delta_t(j)$ as the maximum probability that the HMM is in state j at time t. The value $\delta_t(j)$ is computed, as follows

$$\delta_t(j) = max_{0 \le i \le T-1}[\delta_{t-1}(i)a_{ij}].b_j(O_t) \quad (2)$$

where i is the previous state (at time $t-1$).

The observation probability [7] is measured with respect to the mixture Gaussian distribution (triphone states) and is given by

$$b_j(O_t) = \sum_{m=1}^{M} c_j^m N(O_t; \overline{\mu}_j^m, \overline{\sigma}_j^m) \quad (3)$$

where c_j^m is the weight of the mixture component m in state j and $N(O_t; \overline{\mu}_j, \overline{\sigma}_j)$ is the multivariate Gaussian with mean vector $\overline{\mu}_j$ and covariance vector $\overline{\sigma}_j$ and is given by

$$N(O_t; \overline{\mu}_j, \overline{\sigma}_j) = \frac{1}{(\sqrt{2\pi})^L} \frac{1}{\sqrt{\Pi\sigma_{ji}}} . e^{-\sum \frac{(O_{ji} - \mu_{ji})^2}{2\sigma_{ji}^2}} \quad (4)$$

L is the dimension of the feature vector considered (summation and multiplication is done from 1 through L). Calculation of observation probability is computationally very intensive and is usually calculated in the logarithm domain to reduce computation. Moreover, it is calculated against large number of senones per frame. The calculation of observation probability remains a bottleneck for real-time speech recognition even on high performance computing platform. Real-time recognition is achieved with a trade-off in accuracy by setting thresholds at various stages. In the next section, we propose an architecture of a dedicated unit for calculation of observation probability which enables us to achieve real-time performance with low power consumption. Our architecture enables us to incorporate latest algorithmic changes in speech and linguistics research thus making it flexible.

III. System Architecture

The system consist of a low power general purpose processor, dedicated hardware units and RAM memories. The processor is a low-power processor available for embedded enviroment coupled with a floating point co-processor (Ex - ARM processor ARM946EX-S processor and floating point co-processor VP9-S). The dedicated hardware units are designed for 32-bit floating-point (IEEE-754 standards) operations. This is different from the software solutions which uses fixed point arithmetic. The calculation involves exponents, therefore, all the calculation are done in logarithm domain.

The speech recognition system has components as shown in Figure 1. The input spoken speech goes through the *Frontend*, where spectral analysis is done to extract the acoustic feature vectors.

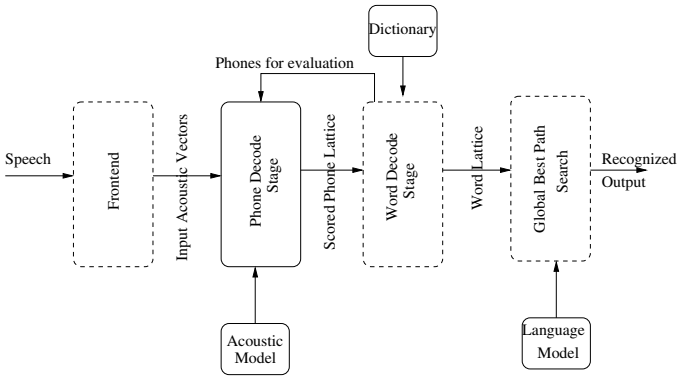

Fig. 1. Speech Recognition System (Dotted box signifies software implementation)

These acoustic vectors then goes through the *phone decode stage*, where the observation probability is evaluated and senone scores are obtained and thereby lattice of phones/triphones are generated depending on the feasible senone permutation. The *word decode stage* then puts phones/triphones together to form word lattice. The word decode also provides the phone decode stage with possible phones (sequence of senones) to be looked up for next set of input acoustic vectors. This is shown by 'Phones for evaluation' in the figure. The *global best path search* outputs the meaningful utterance from the word lattice.

A. Frontend

The assumption made by the speech recognizer is that the speech signal can be regarded as stationary (spectral characteristics are relatively constant) over an interval of a few milliseconds. The prime function of the Frontend is to divide the input speech into blocks (time intervals) and from each block, derive a smoothened spectral estimate. The intervals are typically spaced 10 msecs. Blocks are overlapped to give a longer analysis window, typically 25 msecs. The frontend is implemented in software and runs on the processor core. It is a lightweight process and the present day mobile devices (with embedded processors) have this feature.

B. Phone decode

Observation Probability (OP) Unit

This unit evaluates the observation probability. The calculations are done in logarithmic domain. Applying logarithm operation on both sides of equation (3), we get the following form.

$$log(b_j(O_t)) = log(A_{1j} + A_{2j}.....A_{Mj}) \quad (5)$$

where $A_{mj} = c_j^m N(O_t, \overline{\mu}_j^m, \overline{\sigma}_j^m)$ and applying logarithm and

using equation 4, we have

$$log(A_{kj}) = C_{jk} \sum_{i=1}^{L} (O_{ji} - \mu_{ji})^2 \times \delta_{ji} \quad (6)$$

The above two equations are the core of observation probability calculation and Figure 2 gives the block diagram of this unit. The data path consists of an $(X-Y)^2 \times Z$ floating point unit followed by an adder that completes the loop of equation (6). A fused multiply-add unit then performs the scale and weight adjustment. The 'logadd' unit evaluates the logarithm of additon and hence completes equation (5). The logadd unit uses the following mathematical simplification - $log(A+B) = log(A(1+\frac{B}{A})) = log(A) + log(1+\frac{B}{A})$. Now assuming $B \leq A$,we have $0 \leq log(1+\frac{B}{A}) \leq 0.693$. The logarithm lookup table for $log(1+\frac{B}{A})$ is stored (16bits binary value after the decimal) on a SRAM and is indexed by few least significant bits of $log(\frac{B}{A})$ (which is same as $log(B) - log(A)$). The SRAM size is kept very small (512 bytes). The values can be stored when the system is started.

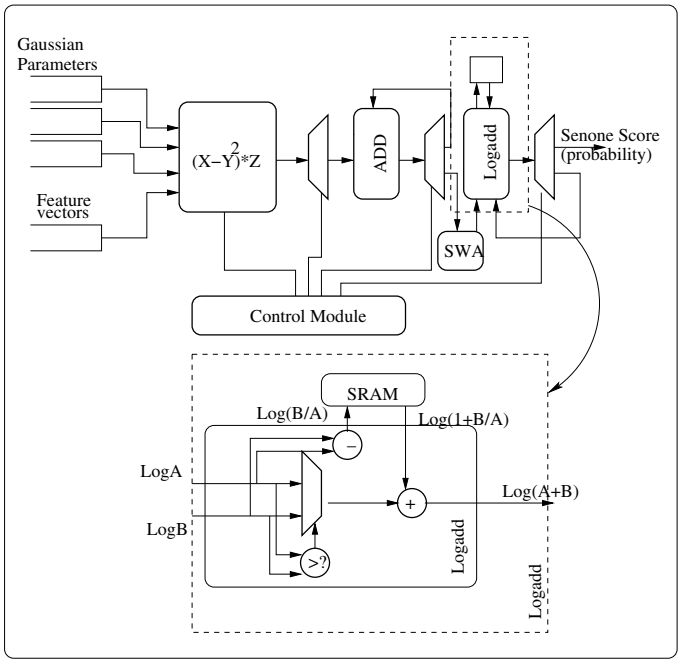

Fig. 2. Observation Probability (OP) Unit

The design is pipelined. The control unit has course grain control over most of the arithmetic units, and multiplexers. The different mode settings provide course-grain control over different stages of the pipeline.

During the normal observation probability estimation process, the input feature vector is first stored in the internal buffer ('Feature vectors' in Figure 2). The Gaussian parameter mean - μ_{ji} and functions of variance - δ_{ji} and C_{jk} are fed to Gaussian parameter buffer. The output of each of the completed internal loop (over the entire vector length) is the evaluation of one Gaussian. This output is scaled and weighed and passed onto the logadd unit, which performs the outer loop calculations in the logarithmic domain (mixture of Gaussians).

Viterbi Decoder

The Viterbi decoder finds most likely path by solving equation (2). This equation is solved in logarithm domain. Applying logarithm, the equation becomes

$$log(\delta_t(j)) = max_{0 \leq i \leq N-1}[log\delta_{t-1}(i) + log(a_{ij})] + log(b_j(O_t)) \quad (7)$$

Therefore, the Viterbi decoder in our system is set of 32-bit adder(s) and comparator(s). The design of adder and the comparator, for finding the maximum, are pipelined.

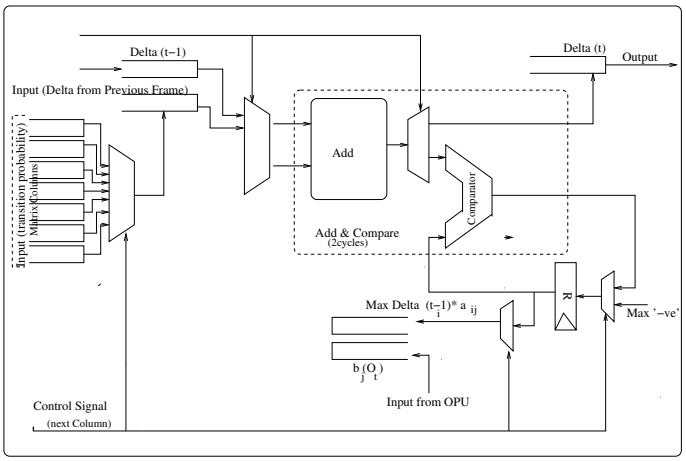

Fig. 3. Viterbi decoder. Delta = δ

The decoder is able to handle multiple state (3, 5, 7) HMMs and therefore has can handle different acoustic models. Figure 3 shows the block diagram of the Viterbi decoder.

C. Word decode and Global best path search

The word decode stage combines the triphones based on high probability values and valid triphone combination according to the words in the dictionary. This requires high bandwidth for dictionary lookup of words. The word decode is implemented in software and it accesses the dictionary (stored in flash memory) through a DMA interface. The word decode also decides which senones are to be evaluated by the *phone decode* based on the phone combinations of the active words in the dictionary. The word decode generates a lattice of probable words spoken. The global best path search iterates over the word lattice and combines the language model to produce the utterance. This is implemeneted in software. The dictionary, acoustic model and the language model are stored in Flash memory.

IV. Result and Discussion

A. Results

We extract the feature vectors from input speech using Sphinx-3 frontend. We implement the observation probability

unit and the Viterbi decoder in Verilog HDL. Synopsys design compiler is used for synthesis of these units using 0.18μ technology. We achieve an operational frequency of 50MHz. Power usage we get is 200mW and the area is 2.2 mm^2. The correctness is checked by floating point implementation of observation probability calculation.

B. Discussion

To save power, our dedicated units use clock gating. In speech recognition, evaluation of all 6000 senone are not generally required in every frame. The Sphinx 3 recognition system indicates that all senones are not evaluated in each frame. Only active senones are evaluated (number of the active senones is much less than 50% of actual senones. Two such dedicated structures (observation probability unit and the Viterbi decoder combined) can support real time speech recognition. Our architecture adapts to the four layer scheme integrated by A.Chan et al. [1]. The Conditional Down Sampling (CDS) [6] is one of the four layer and has the potential to cut the power usage by a considerable margin.

The memory requirement for the dictionary of 20,000 words (Wall street Journal, with average of 9 triphones per word) with 3 state HMM is around 11Mb (9Mb for dictionary and 2Mb of word ID to ASCII mapping). The Acoustic model with 6000 senones needs 15.16MB of memory. The worst case bandwidth requirement is therefore 1.516GBps (assuming all 6000 senones are evaluated in a frame of 10ms) All the values in our system are 32-bit floating point number with 23-bit mantissa. If the length of the mantissa is reduced, then we have smaller storage required for acoustic model and the bandwidth requirment is less. The following table shows the memory and bandwith requirement.

Mantissa	23-bits	15-bit	12-bit
Memory (MB)	15.16	11.37	9.95
Bandwidth (GB/s)	1.516	1.137	0.995

The length of mantissa can be reduced by couple of bits without compromising the accuracy of speech recognition. The word error rate for the Wall street Journal 5000 (WSJ5K) is less than 10% for mantissa of 12-bits and 23-bits. The observation probablities are calculated in logarithmic domain so the values can vary from zero to very large negative value, which may cause a problem for the systems using fixed point computation.

V. Related Work

The software solution for speech recognition is well developed in industry and academia. These solutions are not good for real-time speech recognition. They run on a desktop platform (Pentium Series) consuming all its resources. These recognition systems are not particularly designed keeping the underlying architecture in mind.

A dedicated hardware accelerator has been proposed to speed up the software implementation by Mathew et. al. [3]. This implementation meets real-time performance requirement and reduces bandwidth. Though the power requirement is low for Gaussian calculation, our design has much less power consumption. The speech recognition application is memory intensive and has huge working set and the acoustic models are not accessed through a DMA, therefore, performance may be poor because of resource contention in the underlying system. The FPGA implementation [4] speeds up the probability computation but does not target low power.

The low power device proposed by Sergui et. al. [5] uses SRAM and Flash memory for processing and acoustic and language model. The vocabulary is limited to only couple of hundred words. Therefore, large vocabulary recognition is not possible. The recognition is not triphone based and has less than 30 phones, which implies possibility of high error rate in speech recognition.

VI. Conclusions

An architecture for speech recognition is presented in this paper. Our design does speech recognition in real-time and consumes low power. In our design, we have a simple general purpose processor and dedicated structures for computationally intensive parts in speech recognition algorithm. The total area of dedicated structures (2 similar structures) for real time speech recognition is about $4.4mm^2(2X2.2mm^2)$ and the power is about $400mW(2X200mW)$. We use senones for Gaussian evaluation, thus reducing memory and bandwidth requirement compared to similar proposed systems (Section V) , and can incorporate recent changes in the speech research. The feedback from word decode stage reduces number of senone scores evaluated per frame, thus making it possible to operate at a low frequency and hence save power.

Acknowledgement

The authors would like to thank the contributors of CMU-SPHINX open source code, Cadence Design System and Synopsys for providing tools.

References

[1] A. Chan, R. Mosur, A. Rudnicky, and J. Sherwani. Four-layer categorization scheme of fast gmm computation techniques in large vocabulary continuous speech recognition systems. In *Intl. Conf. on Spoken Language Processing*, pages 689–692, 2004.

[2] M. H. Hwang and X. Huang. Subphonetic Modeling with Markov States - Senone. In *IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages I 33– 36, 1992.

[3] B. Mathew, A. Davis, and Z. Fang. A Low-Power Accelerator for the SPHINX 3 Speech Recognition System. In *Intl. Conf. on Compilers, Architecture and Synthesis for Embedded Systems (CASES '03)*, pages 210–219, 2003.

[4] S. Melnikof, S.F.Quigley, and M.J.Russell. Speech recognition in fpga using discrete and continuous hidden markov models. In *Proc. 12th Intl. Conf. on Field Programmable Logic and Applications (FPL 2002)*, 2002.

[5] S. Nedevschi, R. K. Patra, and E. A. Brewer. Hardware speech recognition for user interfaces in low cost, low power devices. In *DAC '05: 42nd annual conference on Design automation*, pages 684–689, New York, NY, USA, 2005. ACM Press.

[6] M. Woszczyan. Fast Speaker Independent Large Vocabulary Continuous Speech Recognition. In *Universitat Karlsruhe; Institut fur Logik, Komplexitat and Deduktionssysteme*, 1998.

[7] S. Young. Large vocabulary continuous speech recognition: A review. In *In IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 3–28, Snowbird, Utah, December 1995. IEEE., 1995.