Improved Accuracy of Automatic Speech Recognition System using Software and Hardware Co design

P Jayaseelan[1], B Padmasri [2], K Ganeshkumar[3]
[1]*Department.of Electronics and communication Engg, PRIST University, Trichy, Tamildau, India*
[2]&[3]*Dept. of ECE, PRIST University, Trichy.*

Abstract: *We Present Automatic Speech Recognition System For Human Machine Interaction By Using Hardware and Software Co design, The Entire Speech Signal Is Segmented Into a sequence Shorter Signal Is done By Future Extraction, This System Improves the Timing Performance by using Hardware Accelerator GMM (Gaussian mixture model), This hardware accelerator is performed by emission probability Calculation, In order to avoid large memory requirement and this accelerator adopts a double buffering scheme for accessing the acoustic parameters, The weighted finite state transducer (WFST) in Viterbi search model gives Word Sequence from Emission probability calculation. The word accuracy rate is preserved at 93.33%, as a part of recognizer is very suitable for Embedded Real time Application By Using New Adaptive Beam Pruning algorithm. Which is about four times faster than pure Software based system.*

Key words: *Automatic Speech Recognition system (ASR), Hardware software co design, Feature Extraction, Gaussian Mixture model (GMM), real time embedded application, Adaptive beam pruning algorithm.*

I. Introduction

RECENTLY, automatic speech recognition (ASR) on embedded platforms has been gaining its popularity. ASR has been widely used in human-machine interaction [1], [2]; Speech recognition applications are becoming more useful nowadays. Various interactive speech applications are available in the market. With growth in the needs for embedded computing and the demand for emerging embedded platforms, it is required that the Automatic speech recognition systems (ASR) are available on them too. A hardware–software codesign approach seems to be attractive and typically hardware–software co processing system consists of a general purpose processor and hardware accelerator unit operation is to achieve required timing performance. Computationally intensive parts of the algorithm can be handled by the hardware accelerator(s) [4], while sequential and control-oriented parts can be run by the processor core.

The additional advantages of the hardware–software approach include the following.

1) Rapid prototyping of applications. Developers can build their applications in software without knowing every detail of the underlying hardware architecture.
2) Flexibility in design modification. The parts of the algorithm which require future modification can be implemented initially in software.
3) Universality in system architecture. The use of the general purpose processor core enables developers to integrate ASR easily with other applications.

The system includes an optimized hardware accelerator that deals with the critical part of the ASR algorithm. The final system achieves real-time Performance with a combination of software- and hardware implemented functionality and can be easily integrated into applications with voice (speech) control [1]-[4]

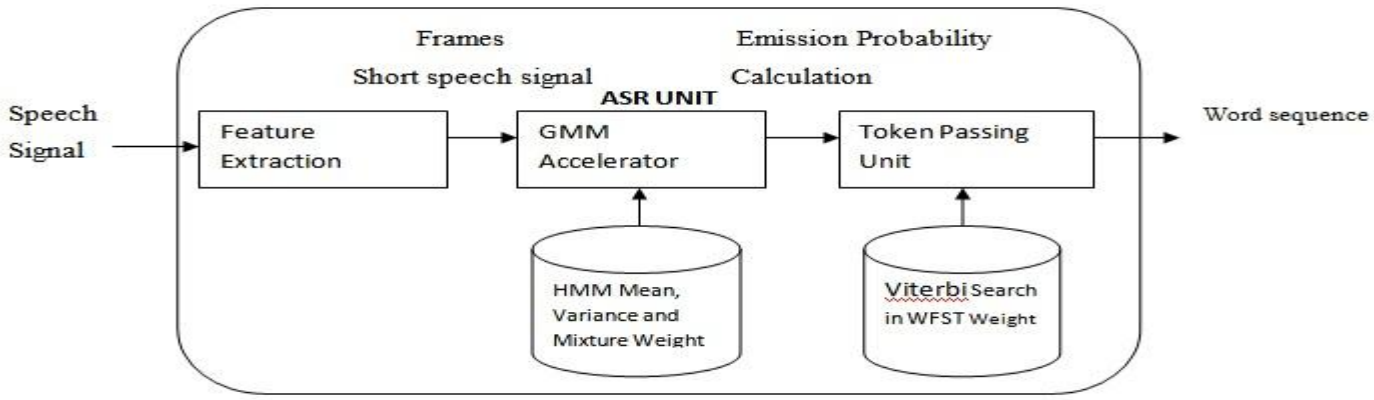

Fig. 1 Data flow diagram of ASR System

II. Automatic Speech Recognition System

In a typical hidden Markov model (HMM)-based ASR system [5], three main stages are involved. Fig. 1 shows the data flow within the ASR algorithm. The first stage is feature extraction. Its main purpose is to convert a speech signal into a sequence of acoustic feature vectors,

oT1 = *{***o**1, **o**2, . . . , **o***T }*, Where *T* is the number of feature vectors in the sequence. The entire speech signal is segmented into a sequence of shorter speech signals known as frames. The time duration of each frame is typically 25 ms with 15 ms of overlapping between two consecutive frames. Each frame is characterized by an acoustic feature vector consisting of *D* coefficients. One of the widely used acoustic features is called Mel frequency cepstral coefficient (MFCC) [3]. Feature extraction continues until the end of the speech signal is reached. The next stage is the calculation of the emission probability which is the likelihood of observing an acoustic feature vector. The emission probability densities are often modeled by Gaussian mixture models (GMMs) [7]. The last stage is Viterbi search which involves searching for the most probable word transcription based on the emission probabilities and the search space. The use of weighted finite state transducers (WFSTs) offers a tractable way for representing the search space. The advantage is that the search space represented by a WFST is compact and optimal .a WFST is a finite state machine with a number of states and transitions. Each WFST transition has an input symbol, an output symbol, and a weight.

In ASR, a word is considered as a sequence of sub word units called phones. Two or three phones are concatenated to form biphones or triphones. Each triphone or biphone label is modeled by an HMM. In other words, each WFST transition is substituted by an HMM. The entire WFST is essentially a network of HMM states.

III. Feature Extraction

The MFCC (Mel-frequency Cepstrum Coefficient) computation employed for the feature of our recognition system consists of 512- point FFT. Filter bank analysis, log operation and IDCT so that it requires intensive arithmetic computations. However, the overhead of the feature extraction is not changed with varying vocabulary size of the recognition task because it only depends on the input speech sample. All the arithmetic operations of the feature extraction are implemented in fixed-point operations to reduce the additional overhead of handling the floating-point operations. The Micro Blaze needs more processing power to achieve the same level of performance with the conventional ARM7 based implementation. The most computation intensive part for MFCC is the 512-point FFT which requires many add and multiply operations. Because the base Micro Blaze processor is not equipped with hardware multiplier, the data path is not adequate to perform FFT. The log arithmetic is implemented by table lookups, and the cache can reduce the overhead of accessing the log table in the external DRAM. Because the Micro Blaze is a soft core that can easily adopt these features in its data path, we adopted the hardware multiplier and the data cache. Figure 2 shows the execution cycles for the feature extraction with various data path configurations. We can see that the performance of Micro Blaze is similar to that of ARM7 architecture by configuring data path. Note that the ARM7 is operating at 60MHz whereas Micro Blaze is at 100MHz.

IV. GMM Accelerator

Feature extraction continues until the end of the speech signal is reached. The next stage is the calculation of the emission probability which is the likelihood of observing an acoustic feature vector. The emission probability densities are often modeled by Gaussian mixture models (GMMs).

$$\Psi(t, sj)=\max\{\Psi(t-1,si)+\log(aij)\}+ \log(b(Ot ; sj)) \quad (1)$$

aij indicates the transition probability from state si to sj and b(Ot; sj) represents the emission probability. For the word-level transcription which will be the output of the speech recognition system, we have to keep the best predecessor word for each word hypothesis.

$$H(w,t)=\max\{\log(p(w/v))+\Psi(t,Sw)\} \quad (2)$$

P (w/v) shows the bigram probability from word v to word w and S,,, indicates the last states of the word w. Since exhaustive search in Viterbi search is prohibited, pruning is employed at both the state level and the language model level. Therefore the paths which have low accumulated likelihood are pruned at the early stage to reduce the search space size .For the DP recursion shown in Eq. 1, the output probability of the current observation for each HMM state should be computed. The log likelihood for the Gaussian mixture m of HMM state s is computed. Eq 2 is a pre-computed Gaussian constant that is independent of the observation Ot = {X1X2 X39}. p and a indicate the mean and the variance of the Gaussian mixture respectively. Because all the model parameters are stored in 16 bit format, if we assume the worst case of loading all the mixture model from memory to compute the emission probability, it requires up to 64MB/s memory bandwidth. In real time implementation, however, we do not compute the mixture models which are decided to be excluded in the Viterbi beam search by beam pruning. The memory bandwidth for the emission probability required is thus reduced to about 7MB/s as shown in Fig. 3

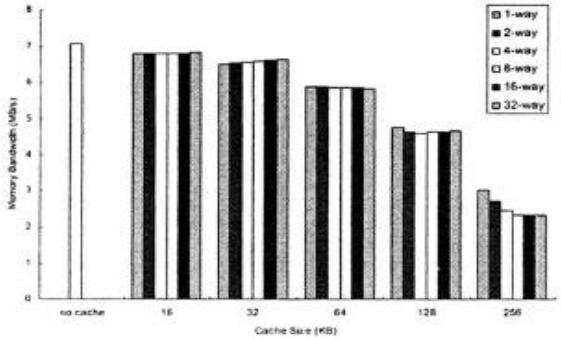

Fig 2. Memory bandwidth of various caches

4.1 Architecture Design

Figure 3 shows the architecture of hardware accelerator for the emission probability. The feature vector which is computed in the Micro Blaze processor is stored in the dedicated feature storage so that the feature vector can be reused for the single frame. The model data to be filled in the internal memory of the hardware accelerator is transferred via FSL. The cache is also employed in the micro blaze

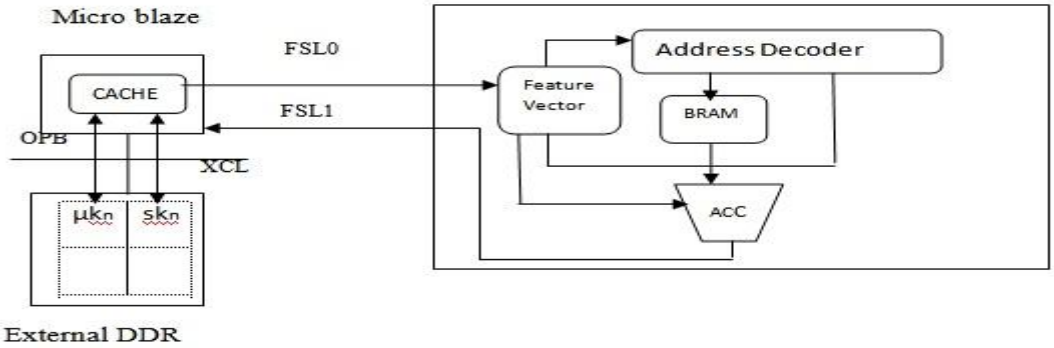

Fig 3. Hardware Accelerator for Emission Propability Calculation

To accommodate the faster bus XCL. As we can see from the architecture, the memory bandwidth is the main factor that decides the total execution time. It requires only 5 cycles to generate the output after loading the model parameter. A naive solution for the extreme memory access is storing all the acoustic model data into the internal memory which can be implemented using BRAM blocks. The result showed about 20 times reduction in the execution cycle for the emission probability. However, this approach is not adequate for a large vocabulary speech recognition system simply because it requires an excessively large BRAM size and the solution is no longer feasible.

4.2 Cache Analysis

The second option is adopting cache in the accelerator. Figure 3 shows the memory bandwidth requirement with various cache configurations. In this experiment, the cache hit ratio looks relatively high considering that the system loads large amount of data. The reason is that the Gaussian computation for one mixture loads 80 consecutive 16 bit data generating only 5 cache misses assuming the 32 bytes line size. Thus we employed memory bandwidth as the performance measure. Figure 3 shows that relatively small size of cache from 16KB to 32KB does not improve the performance much. The Gaussian mixture model should be reused as many times as possible to reduce the memory bandwidth between cache inside the hardware accelerator and memory. However, the large size of the mixture models makes the data easily flushed out of the cache. Even 32KB dedicated cache can hold up to only 200 mixtures at once whereas the number of mixtures that are accessed in one frame can reach over 1000 in usual cases. Note that caching actually start to take place when the memory size is increased to 256KB. Cache configurations below 256KB would produce only capacity misses therefore cache associatively would not yield any effect.

4.3 Data Partition

The behavior of speech recognizer varies with the input speech, especially when we adopted adaptive techniques such as beam pruning to achieve high performance. However, the underlying language model of the system also plays an important role in the recognizer. The mixture model to be computed is decided by the active states in the Viterbi beam search network.[3] .There are two components that are responsible for making the decision on which states are to remain active. The first one is the accumulated likelihood of the state. The beam pruning compares the accumulated likelihood of a state to the best accumulated likelihood so that the states with low values are pruned. The other is the language model. Figure 1 shows the back off bigram language model which is usually employed for a large vocabulary speech recognition system.

In this model, all the words in the vocabulary are composed of the word-loop grammar and the corresponding bigram suppresses the unlikely paths. The next state of an active state within a word should be active in the next iteration. The first state of a word which comes from the active final state of a word should also be active. The former is mostly dependent on the input speech where there is a little room for us' to exploit. However, the latter is highly dependent on the language model so that we can use the feature regardless of the input speech. Developing a speech recognition system requires very sensitive tuning because the performance heavily depends on the configuration parameters such as acoustic beam pruning threshold, language model scale factor, word insertion penalty, etc [1]. The parameter setting is done based on the test set that is independent of the training set. Similarly, we extracted the feature of the language model exploiting the training set. Access pattern of the emission probability computation for all the speech in the test set is recorded and the N most probable states are chosen. The memory bandwidth can be reduced by placing these states in the internal memory of the hardware accelerator. Figure 5 shows the required memory bandwidth when the internal memory is employed. The experiment of placing random states in the internal memory is also performed to emphasize the importance of efficient selection of the states. The experiment where we store the states which are known to be the most probable by profiling showed about 15% to 27% performance improvement as total internal memory size varies from 32KB to 256KB. The experiment where the random states are stored shows the performance worse than when the cache is adopted. This result shows that the carefully partitioned internal BRAM [6-8] can outperform when the same size of the cache is employed. However, if we randomly partition the model data, the performance is worse than that of the cache. Note that the profiling is performed by 300 various test speeches with all of them having different speakers and transcriptions and only one of them is matched to the speech used in this experiment. This means that the profiling results contain language model information rather than acoustic model information.

V. Adaptive Pruning Algorithm

The hardware–software co processing system demonstrates a significant improvement on the decoding speed. Out of the 1200 utterances in the RM1 corpus, about 94% of them have a real-time factor of less than one. In this section, we aim at further increasing this percentage. One method for lowering the number of active tokens is to adopt a tighter pruning beam width. However, it will introduce search errors which often decrease the recognition accuracy. Our goal is to reduce the decoding time of those utterances which have a relatively greater real-time factor, while keeping the recognition accuracy of the other utterances. In order to fulfill this goal, an adaptive pruning scheme is proposed, where the pruning beam width is adaptive according to the number of active tokens.

5.1. Importance of Algorithm

The proposed pruning scheme is more flexible than the narrow and fixed pruning scheme. The number of active tokens is often time varying in the duration of an utterance. The fixed pruning scheme applies a tight beam width throughout the entire utterance regardless of the number of active tokens. On the other hand, the adaptive scheme allows relaxation of the beam width in parts of the utterance where the workload is less heavy.

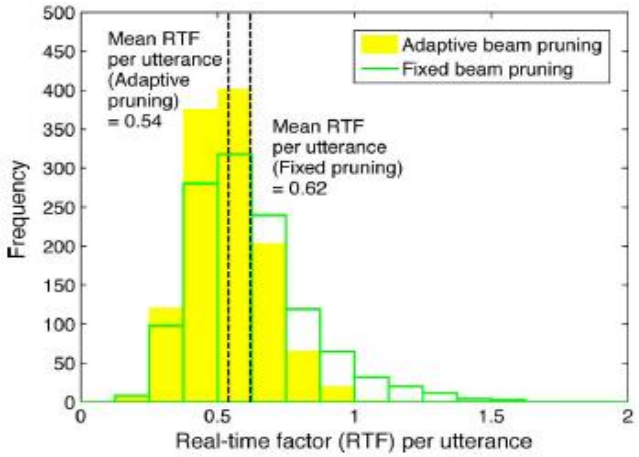

Fig 4. Real time factor utterance in HW-SW co design system, Adaptive beam pruning versus fixed beam pruning algorithm

VI. Token Passing Unit

The last stage is Viterbi search which involves searching for the most probable word transcription based on the emission probabilities and the search space. The use of weighted finite state transducers (WFSTs) offers a tractable way for representing the search space [7]. The advantage is that the search space represented by a WFST is compact and optimal [8], Fig. 1 shows an example of a search space. Each WFST transition has an input symbol, an output symbol, and a weight. The input symbols are the triphone or biphone labels. The output symbols are the word labels. In ASR, a word is considered as a sequence of sub word units called phones. Two or three phones are concatenated to form biphones or triphones. Each triphone or biphone label is modeled by an HMM. In other words, each WFST transition in Fig. 3 is substituted by an HMM. The entire WFST is essentially a network of HMM states. The WFST weights are the language model probabilities which

model the probabilistic relationship among the words in a word sequence. Usually, a word is grouped with its preceding (n − 1) words. The n-word sequence called n-gram is considered as a probabilistic event. The WFST weights estimate the probabilities of such events. Typical n-grams used in ASR are unigram (one word), bigram (two word, also known as word pair grammar), and trigram (three word). For implementation purposes, each HMM state has a bookkeeping entity called *token* which records the probability (*score*) of the best HMM state sequence ended at that state. Each token is propagated to its succeeding HMM states according to the topology of the search space.

VII. Conclusion

The proposed ASR system shows much better real-time factors than the other approaches without decreasing the word accuracy rate. Other advantages of the proposed approach include rapid prototyping, flexibility in design modifications, and ease of integrating ASR with other applications. These advantages, both quantitative and qualitative, suggest that the proposed co processing architecture is an attractive approach for embedded ASR. The proposed GMM accelerator shows three major improvements in comparison with another co processing system [5], [6]. First, the proposed accelerator is about four times faster by further exploiting parallelism. Second, the proposed accelerator uses a double-buffering scheme with a smaller memory footprint, thus being more suitable for larger vocabulary tasks. Third, no assumption is made on the access pattern of the acoustic parameters, whereas the accelerator in [6] and [7] has a predetermined set of parameters. Finally, we have presented a novel adaptive pruning algorithm which further improves the real-time factor. Compared with other conventional pruning techniques, the proposed algorithm is more flexible to deal with the time-varying number of active tokens in an utterance. The performance of the proposed system is sufficient for a wide range of speech-controlled applications. For more complex applications which involve multiple tasks working with ASR, further improvement of timing performance, for example, by accelerating the Viterbi search algorithm, might be required. The proposed co processing architecture can easily accommodate additional hardware accelerators. Aside from better word accuracy and timing performance, power consumption is also another important issue for embedded applications. The proposed architecture is not tied to any specific target technology. One of the future development paths is to transfer the current implementation from FPGA to very large scale integration which consumes less power. Our future work includes some further refinements of the speech recognition algorithm, exploration of design space to look for improvements of the hardware–software co processing system, and encapsulation of the speech recognizer into an intellectual property block that can be easily used in other applications.

References

[1] A.Green and K.Eklundh.”Designing for learn ability in human-robot communication,” IEEE trans, ind.Electron, vol.50, no.4.pp.644-650, Aug, 2003.
[2] M.Imai.T. Ono. And H.ishiguro.”Physical relation and expression: joint attention for human-robot nitration.”IEEE Trans.Ind.Electron. vol.50, no.4, pp.636-643, Aug.2003.
[3] N.Hataoka, Y.Obuchi, T.Mitamura, and E.Nyberg.”Robust Speech dialog interface for car elematics service,” in proc.IEEE consumer commun, netw.conf. 2004, pp.331-335.
[4] K.Saeed and M.nammous,” A Speech and Speaker identification, and classification of Speech- signal image,” IEEE Trans.Ind.Electron. vol.54, no.2, pp.887-897, Apr.2007
[5] K.You, H.Lim, and w. Sung, “Architectural design and implementation of an FPGA softcore based speech recognition system.” In Proc, 6th int. Workshop syst. Chip real time appl,.2006.pp.50-55.
[6] L.Raniner.”A tutorial on hidden markov models and selected applications in speech recognition,” Proc.IEEE, vol 77, no. 2, pp257-286.
[7] H.Ney , D.Mergel , A,Noll, and A. Paeselar,” A data-driven organition of the dynamic programming beam search for continous speech recognition,” in Proc, ICASSP.1987, pp.833-836.
[8] H.Ney. R.Haeb-Umbach, B.Tran, and M.Oerder,”Improvements in beam search for 10000-word Continous speech recognition,” in Proc ICASP. 1992. DD.9-12

Authors

P.Jayaseelan, Received B.E. in Electronics and communication Engg (with first-class honors) and Doing M.Tech Embedded systems in PRIST University, Trichy, India. He is Currently Working Lecturer in Ponnaiyah Ramajaym Polytechnic College, kumbakonam. His Research interests include embedded system, Speech recognition and machine learning.

B .Padmasri, received her Master of Technology in Communication systems from PRIST University, India. She is currently working as Assistant professor in PRIST University, India. Her Research Interest in Real time embedded application.

K.Ganeshkumar, doing M.tech in PRIST University, He currently working in ponnaiyah ramajayam polytechnic College, His research interest in Real time embedded security system.