Real Time Speech Recognizer Implementation using WFST and GMM based on FPGA

T. Balasharmitha
Assistant Professor,
Dept.of ECE,
Mount Zion College of Engineering and College,
Pudukkottai.

S. Jerald
UG student, Dept.of ECE,
Mount Zion College of Engineering and College,
Pudukkottai

***Abstract*-The speech recognition technique evolved from dynamic time warping to secreted markov model has many applications in energy controlled devices that are not well served by existing software or hardware decoders. Modern speech recognition techniques describe an integrated circuit that provides a local speech recognition capability for a variety of electronic devices. It starts with a generic speech decoder architecture that is programmable with industry-standard weighted finite state transducer (WFST) and Gaussian mixture model (GMM) speech models. Algorithm and architectural enhancements are incorporated. In order to achieve real time performance amid system level constraints on internal memory size and external memory bandwidth. The band pass filter banks employ two multipliers in the conventional methods. The proposed method is implemented using the modified block of multiplier. This method performs a word recognition task in real-time with reduced word error rate, core area and power consumption, and improved search efficiency per hypothesis. The performance parameters are analyzed and compared with the previous multiplier and concluded to be the fast and best outcome in the evaluation results.**

Index terms- Speech recognition, weighted finite state transducer (WFST), Gaussian mixture model (GMM).

I. INTRODUCTION

Speech recognition has applications in energy- constrained devices that are not well served by existing software or hardware decoders. This paper shows how modern speech recognition techniques can be mapped to digital circuits, and presents a test chip with design techniques that reduce power consumption and memory bandwidth to levels appropriate for embedded systems. Much of current large-vocabulary speech recognition is based on models such as hidden Markov models (HMMs), lexicons, or n-gram statistical language models that can be represented by *weighted finite state transducers*. Even when richer models are used, for instance context-free grammars for spoken-dialog applications, they are often restricted, for efficiency reasons, to regular subsets, either by design or by approximation. The envision cooperation and interchangeability between cloud-based (software) and circuit-based (hardware) implementations of speech recognition. For Internet-connected devices, cloud- based decoders can be provisioned to run in real-time with decoding adding little to the latency of the network. Hardware speech recognition becomes useful when the Internet connection is slow or unreliable, when the overhead of using a cloud-based decoder is prohibitive (e.g., wearable electronics with limited battery capacity),or for local operations that do not require Internet capabilities (e.g., appliances and industrial equipment). As hardware speech decoders improve, client devices will be able to seamlessly transition between cloud-based and local decoding.

Speech recognition is an area of active research. While usefully accurate algorithms and models for domain-constrained tasks are well known, practical limitations have prevented the widespread adoption of hardware recognizers:

- Fixed-function digital circuits cannot easily be reprogrammed to keep up with developments in algorithms and statistical modeling techniques.
- Circuits developed so far require complex external components and interfaces to realize complete speech- to-text decoding.

- The memory size and bandwidth requirements of speech decoding (often similar to those of a general-purpose computer) are excessive.

II. RELATED WORK

Speech recognition technology evolved from dynamic time warping (DTW) in the 1980s to hidden Markov models (HMM) that are still used today. The HMM is a graphical model expressing a relationship between hidden variables (representing the underlying word sequence) and observed variables (representing the acoustic observations). The regular structure of connections in this graph (Markov property) is exploited to allow approximate inference in linear time (Viterbi algorithm), making statistical speech recognition tractable.

A. *Dynamic Time Warping (DTW) Method*

Dynamic time warping is an algorithm for measuring similarity between two sequences that may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another he or she were walking more quickly, or even if there were accelerations and deceleration during the course of one observation. DTW has been applied to video, audio, and graphics – indeed, any data that can be turned into a linear representation can be analyzed with DTW.

B. Hidden Markov Models

Hidden Markov models (HMMs) are widely used in many systems. Language modeling is also used in many other natural language processing applications such as document classification or translation. Modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models that output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time scale (e.g., 10 milliseconds).

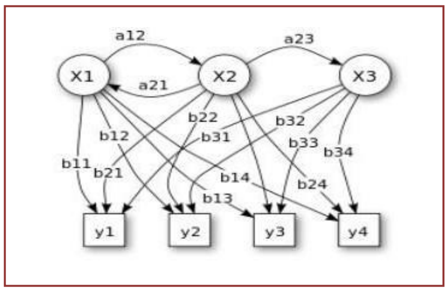


Fig 1. Probabilistic parameters of a hidden Markov model

Speech can be approximated as a stationary process. Speech can be thought of as a Markov model for various stochastic purposes.
X —states
y —possible observations
a —state transition probabilities
b — output probabilities

In its discrete form, a hidden Markov process can be visualized as a generalization of the Urn problem with replacement (where each item from the urn is returned to the original urn before the next step). Consider this example: in a room that is not visible to an observer there is a genie. The room contains urns X1, X2, X3,…each of which contains a known mix of balls, each ball labeled y1, y2, y3, … . The genie chooses an urn in that room and randomly draws a ball from that urn. It then puts the ball onto a conveyor belt, where the observer can observe the sequence of the balls but not the sequence of urns from which they were drawn. The genie has some procedure to choose urns; the choice of the urn for the *n*-th ball depends only upon a random number and the choice of the urn for the ($n - 1$)-th ball.

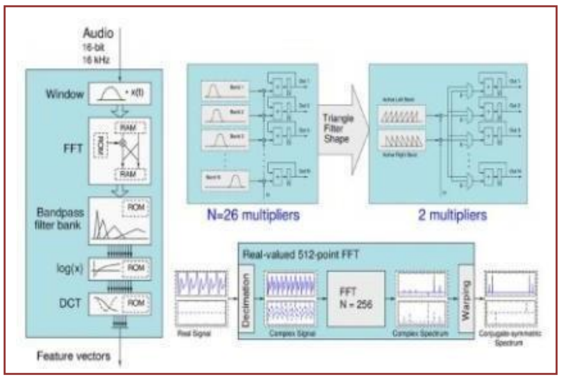


Fig 2. Three types of transitions in HMM

The HMM framework requires modeling the dependencies between variables, specifically the

III. PROPOSED METHODOLOGY

Fig. 3 shows the high-level architecture of a speech recognition system. Training components are implemented in software and decoding components are implemented on an IC. The decoder includes a front-end which transforms audio (typically 16 bitdataat16kHz) into the feature presentation used by the models, and a search component which generates hypotheses about the underlying speech production process.

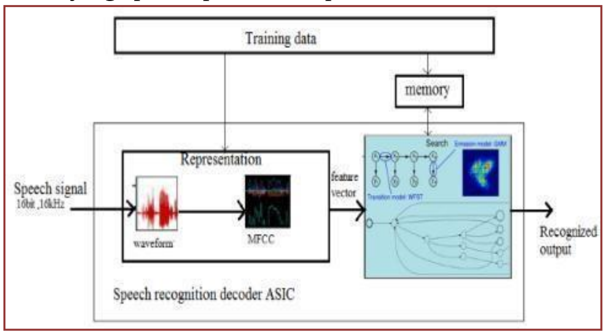


Fig 3. Architecture of speech recognition system

Speech audio signals are sparse in the sense that time-domain sampling captures redundant information. To use Mel-frequency cepstral coefficients (MFCCs) to concisely represent relevant characteristics of the signal[10]. The MFCCs' low dimensionality simplifies classification, and channel effects (convolution) are additive in the cepstral domain. Several standalone feature extraction devices have been reported in the literature, including DSPs, FPGAs, fixed-function digital circuits, and low-power analog front-ends. The front-end could be used to provide features to an external (e.g., cloud-based) decoder or to the on-chip decoder.

A. MFCC

MFCCs are derived from an audio time series via a chain of conventional signal processing blocks including an FFT, melscale bandpassfilter bank, and DCT. The audio is split into a series of overlapping frames 25 ms long with a 10 ms pitch. In addition to computing 12 cepstral coefficients and the log power for each frame, it extract first and second-order time differences and concatenate them into a 39-dimensional feature vector at 16 bit resolution. Cepstral mean normalization is approximated by subtracting a 10 second moving average from the feature vectors.

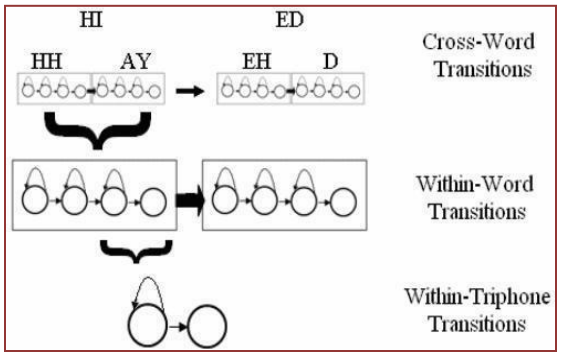


Fig 4. Block diagram of MFCC

The FFT exploits the real-valued nature of the input signal to operate on half as many points, and the bandpassfilter bank employs two multipliers which are reused across the 26 bands. The circuit has an area of 51.8 k gates (plus 107 kb of SRAM and 66 kb SROM) and requires a clock speed of at least 625 kHz to process a 16 kHz waveform in real-time.

B. *Viterbi Search*

Viterbi search begins with an empty hypothesis for the utterance text and incorporates a stream of information from the feature vectors to develop a set of active hypotheses, represented by states in the HMM. Each of these hypotheses is modeled using the WFST and GMM; if its likelihood is sufficiently high, it will be saved. Fig shows the architecture of this subsystem. The forward pass of Viterbi search propagates a set of hypotheses forward in time, from the active state list of frame t to that of frame t+1.

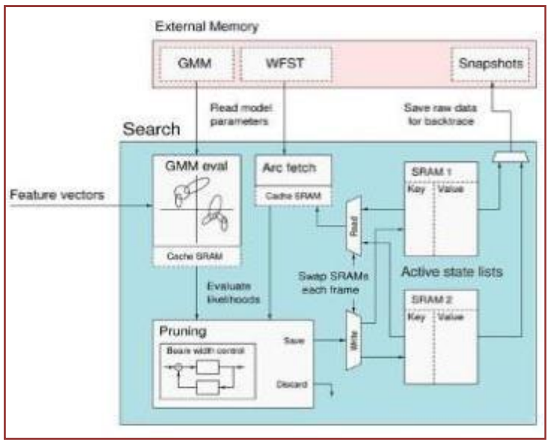


Fig 5. Block diagram of Viterbi search module

1) Hypothesis Fetch-Each hypothesis is read from the active state list for the current frame (frame t). The active state list is implemented as a hash table in SRAM with open addressing and collisions resolved by linear probing. This store accepted states for the next frame in a separate SRAM and swap the two SRAMs using multiplexers at the end of each frame. In these hash tables, the key is a unique state ID (the memory address of the state in the WFST model) and the value is a structure describing the state. Hash table operations become slower as more states are stored (increasing the number of collisions), but this latency is masked by the much longer time required by WFST and GMM operations that access external memory. To reduce the need for memory accesses, arc labels and other metadata are carried through the pipeline along with state information.

2) Arc Fetch-Each state ID is an index into the WFST model, from which we can retrieve the parameters of all outgoing arcs. These include all of the information (except a final score) that will be stored in the active state list if the hypothesis is accepted. While this operation requires far less memory bandwidth than fetching GMM parameters, the arcs retrieved during a typical frame are distributed sparsely across a large memory space: 193 MB for the 5,000 word WSJ model, versus 1 GB or larger for state-of-the-art models. The memory access pattern is

3) sparse and depends on the hypotheses being searched.

4) Control- A control module accepts commands via a byte wise host interface (UART or USB) to actuate the front-end, search, and modeling components. These commands can program models into external memory, send audio data to the front-end, send feature data directly to the search system, and adjust configuration parameters. The chip is also instrumented with registers that collect a variety of statistics at the per-frame and per-utterance level. The statistics include the number of active states, number of WFST states expanded, and amounts of memory access required for WFST and GMM parameters. The host can be a portable or embedded device since its computational demands are trivial.

5) Pruning and Storage-The state space of a Viterbi search grows exponentially unless a beam or histogram method is used to prune unlikely hypotheses from being stored and considered. A beam pruning stage tests each hypothesis against a threshold (relative to the highest likelihood encountered on the previous frame). Only hypotheses that pass the test are stored into the active state list for frame t+1. Once all hypotheses for a given frame have been processed, a snapshot of the active state list is saved to the external memory.

C. Weighted Finite State Transducer

Weighted Finite-State transducers (WFST) have proven quite successful in many fields of written and spoken language processing[2]. This includes in particular machine translation, large vocabulary continuous speech recognition and speech synthesis. An interesting feature of FSMs is that they can be automatically built or "learned" from training data using corpus based techniques. Compared to more traditional knowledge based approaches, these techniques are attractive for their potential of much lower development costs.
between states. Each transition has a source state, a destination state, a label and a weight. Such automata are called weighted finite state acceptors (WFSA), since they accept or recognize each string that can be read along a path from the start state to a final state. Each accepted string is assigned a weight, namely the accumulated

weights along accepting paths for that string, including final weights. An acceptor as a whole represents a set of strings, namely those that it accepts. As a weighted acceptor, it also associates to each accepted string the accumulated weights of their accepting paths.

2) Weighted Transducer-A weighted finite-state transducer (WFST) is quite similar to a weighted acceptor except that it has an input label, an output label and a weight on each of its transitions. This adds no new information, but is a convenient way we use often to treat acceptors and transducers uniformly. The word corresponding to a pronunciation is out-put by the transition that consumes the first phone for that pronunciation. The transitions that consume the remaining phones output no further symbols, indicated by the null symbol as the transition's output label. In general, an input label marks a transition that consumes no input, and an output label marks a transition that produces no output.

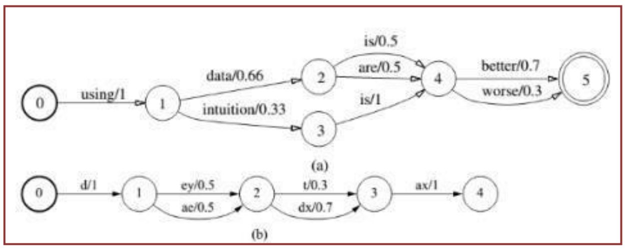


Fig 6. Example of weighted transducer

3) Composition-Composition is the transducer operation for combining different levels of representation. For in-stance, a pronunciation lexicon can be composed with a word-level grammar to produce a phone-to-word transducer whose word strings are restricted to the grammar. A variety of ASR transducer combination techniques, both context-independent and context-dependent, are conveniently and efficiently implemented with composition.

1) Weighted Acceptors-Weighted finite automata (or weighted acceptors) are used widely in automatic speech recognition (ASR). The automaton is a toy finite-state language model. The legal word strings are specified by the words along each complete path, and their probabilities by the product of the corresponding transition probabilities. These automata consist of a set of states, an initial state, a set of final states (with final weights), and a set of transitions

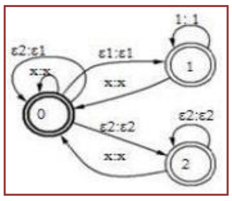


Fig 7. Example of composition

4) Determinization : In a deterministic automaton, each state has at most one transition with any given input label and there are no input labels. The non-deterministic weighted acceptor: at state 0, for instance, there are two transitions with the same label a. The automaton on the other hand, is deterministic. Deterministic automaton over equivalent nondeterministic ones is its irredundancy it contains at most one path matching any given input string, thereby reducing the time and space needed to process the string. This is particularly important in ASR due to pronunciation lexicon redundancy in large vocabulary tasks. The familiar tree lexicon in ASR is a deterministic pronunciation representation.

5) Minimization-The size reduced by minimization, which can save both space and time. Any deterministic unweighted automaton can be minimized using classical algorithms. In the same way, any deterministic weighted automaton can be minimized using the minimization algorithm, which ex-tends the classical algorithm, this algorithm works in two steps: the first step pushes weight among transitions, and the second applies the classical minimization algorithm.

D. Gaussian Mixture Model

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

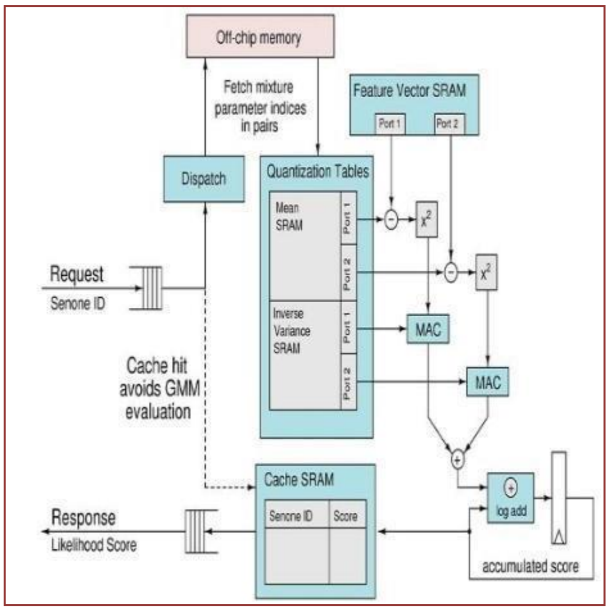


Fig 8. Block diagram of GMM evaluator

1) GMM Score Caching-While the GMM acoustic model occupies less memory than the WFST, reads of GMM parameters are highly sequential and make up the bulk of data volume in any speech decoding task. We now present two architectural

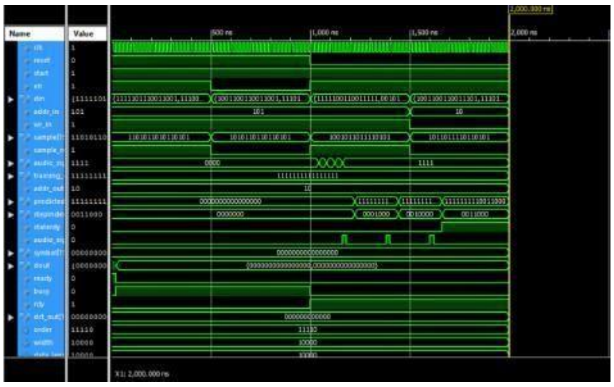
enhancements which together reduce the bandwidth needed for GMM parameters by a factor of 55.A typical decoding workload in tests required 2,900 GMM evaluations per frame, each with 32 mixture components. In a 39-dimensional feature space, using single-precision parameters, this would require 2.9 GB/s of memory bandwidth. However, the same GMM density is often evaluated multiple times per frame (for different arcs with the same input label). By caching each distinct GMM log-likelihood and adding the weight and previous likelihood of each candidate arc, we avoid repeated calculations (and parameter reads) with a hit rate of 86%.

2) GMM Parameter Quantization-Compressing the GMM model itself allows a further reduction in memory bandwidth into the sub-100 MB/s range that is feasible with flash memory. With an appropriate choice of quantizer one can eliminate many bits of resolution from the mean and variance values while incurring just 1–2% higher WER. We use 32 quantization levels (5 bits) for the mean and 8 levels (3 bits) for the inverse variance; these sizes were identified by as a good trade- off between accuracy and bandwidth. The feature representation is not normalized or whitened, so the parameters have very different empirical distributions in different dimensions; thus, we need a different quantizer for each of the 39 dimensions. Figure demonstrates how caching and quantization are incorporated into the GMM evaluator.

A dual-port SRAM totaling 41 kb is used to store the quantization tables at 16 bit resolution; these are loaded from the external memory at startup. The two ports of the SRAM feed two arithmetic units in parallel, such that evaluating one 39-dimensional mixture component requires 20 clock cycles. Parallel evaluation would increase throughput but require corresponding increases in the width of the external memory bus.

IV. SIMULATION RESULT

The modified speech recognizer using GMM uses the modified multiplier block, so the area is reduced compared with previous architecture. Using pipelining throughout the design increases the throughput and allow for fast clock frequency. One example is GMM computation, where GMM propabilities in six different dimensions are computed at same time. Buffering model is used to improve the decoding speed of frame and to prevent the droping frames. Therefore buffer model used after normalizing the MFCC values.
The 2n-bit product register (A) is initialized to
0. Since the basic algorithm shifts the multiplicand register (B) left one position each step to align the multiplicand with the sum being accumulated in the product register, use a 2n-bit multiplicand register with the multiplicand placed in the right half of the register and with 0 in the left half.



The performance comparisons of the conventional and a new model of proposed system is tabulated as below. The area occupied by Look Up Tables (LUTs) of these methods is 17% and 9% respectively. The speed of the method is also increased and power consumption is reduced.

PARAMETERS	EXISTING METHOD	PROPOSED METHOD
Area(LUTs)	17%	9%
Power(MUL)	1.358%	1.274%
Speed	1.437ns	0.797ns

V. CONCLUSION

The proposed method is analyzed and compared to show the modified architecture is better in reduced computational complexity. Computational problems such as speech recognition admit a spectrum of system design approaches. While the proposed method has focused on the low-power results that can be obtained using ASICs, commercial systems may choose to use other approaches including FPGA, GPU and DSP architectures (or some combination, such as SoC hardware accelerators). The architectural changes described above can also be applied to other types of systems, (including embedded software) if memory bandwidth reductions are desired. The proposed method presented an ASIC speech decoder that uses industry- standard WFST models and performs end-to-end recognition (audio in, text out). Compared to state-of- the-art hardware speech decoders, it achieves order of magnitude lower core power consumption. In future work, will modify the speech decoder to be better than the previous work in order to achieve low area and power consumption.

REFERENCES

[1] R. Lawrence, Fundamentals of Speech Recognition.NewYork,NY, USA: Pearson Education, 2008.
[2] M. Mohri, "Weightedfinite-state transducer algorithms. An overview," in Formal Languages and Applications.NewYork,NY,USA: Springer, 2004, pp. 551–563.
[3] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers,"Springer Handbook on Speech Processing and Speech Communication, 2008.
[4] C. Allauzen, M. Mohri, M. Riley,and B. Roark, "A generalized construction of integrated speech recognition transducers," inProc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'04), 2004, vol. 1, pp. I- 761–I-764.
[5] S. J. Young,The HTK Hidden Markov Model Toolkit: Design and Philosophy, Cambridge Univ., Cambridge, U.K., Tech. Rep., 1994.
[6] M.Riley,A.Ljolje,D.Hindle,andF.Pereira,"The AT&T60,000word speech-to-text system," in4th Eur. Conf. Speech Communication and Technology, 1995.
[7] A.Stolzle,S.Narayanaswamy,H.Murveit,J.Raba ey,andR. Brodersen, "Integrated circuits for a real-time large-vocabulary continuous speech recognition system,"IEEE J. Solid-State Circuits, vol. 26, no. 1, pp. 2–11, Jan. 1991.
[8] A.Burstein, "Speech recognition for portable multimedia terminals," Ph.D. dissertation, Univ. California, Berkeley, CA, USA, 1996.
[9] E. C. Lin, K. Yu, R. A. Rutenbar, and T. Chen, "Moving speech recognition from software to silicon: The in silico vox project," presented at the 9th Int. Conf. Spoken Language Processing (INTERSPEECH-2006, ICSLP), Pittsburgh, PA, USA, 2006.
[10]E. C. Lin, K. Yu, R. A. Rutenbar, and T. Chen, "A 1000-word vocabulary, speaker- independent, continuous live-mode speech recognizer implemented in a single FPGA," inProc. 2007 ACM/SIGDA 15th Int. Symp. Field Programmable Gate Arrays, FPGA'07,NewYork,NY, USA, 2007, pp. 60–
68
[11]E. C. Lin and R. A. Rutenbar, "A multi-FPGA 10x-real-time highspeed search engine for a 5000-word vocabulary speech recognizer," in Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays, FPGA'09, New York, NY, USA, 2009, pp. 83–92.