Hardware for Hidden Markov-Model-Based, Large-Vocabulary Real-Time Speech Recognition

M. Weintraub, G. Chen, J. Mankoski, H. Murveit
SRI International
A. Stölzle, S. Narayanaswamy, P. Schrupp, B. Richards, J. Rabaey, R. Brodersen
University of California, Berkeley, CA

Abstract

SRI and U.C. Berkeley have begun a cooperative effort to develop a new architecture for real-time implementation of spoken language systems (SLS). Our goal is to develop fast speech recognition algorithms, and supporting hardware capable of recognizing continuous speech from a bigram- or trigram-based 20,000-word vocabulary or a 1,000- to 5,000-word SLS.

1 Introduction

In order to implement a hidden Markov-model-based, real-time, large-vocabulary speech recognition system it is necessary to support the most time-critical parts of the algorithm in custom VLSI hardware. For systems with 20,000 words and more it is also necessary to have algorithmic support to reduce the amount of computations that have to be done.

Our first version of real-time speech recognition hardware [1] was targeted for systems with up to 3,000 words. It continuously processed all the states of the Markov model that describe the vocabulary words, even states that are not likely to fit to the speech that has to be recognized. However, for larger systems (20,000 words and more) it is necessary to perform a pruning algorithm such that only states with a high probability are processed. Nevertheless, it is desirable to process as many states as possible to reduce the risk that meaningful states are pruned. Our first hardware version used dynamic memories with an access cycle time of 200 ns, and since accessing memory is the crucial bottleneck of the system, not more than 5,000 states per frame could be processed. Other shortcomings were that the board design turned out to be very complicated: at the end of each frame most of the memories had to be flipped between processors. Since these memories also had to be accessed by the host processor, about 50% of the board area was used just for multiplexing the buses. In addition, the grammar processing system was implemented with custom hardware such that only systems with a statistical bigram grammar [1] could be supported.

This paper describes a new system that implements pruning and is capable of processing up to 20 million active states per second. Assuming that a frame has a duration of 10 ms and a phoneme is modeled with three states, a vocabulary word is modeled with an average of six phonemes, 50% of the vocabulary words are active at any given frame and that one third of the phones

in these words are active, the new system can perform in real time for vocabulary sizes of up to 60,000 words. It also contains custom hardware to support grammar processing routines that are common to a whole variety of grammar models. Thus, grammar processing or natural language processing can be implemented on a general-purpose processor that uses the custom hardware for the time-critical part. To ease the system design and to reduce the memory requirements, we use a switching processor architecture and a caching scheme to preload only a small subset of the model parameters at a given frame. The new hardware contains 14 custom VLSI processors that access fast static memories with cycle times of 20 MHz.

2 Architecture

Fig. 1 shows the overall architecture of the recognition hardware. The *phone processing system* updates the state probabilities using the Viterbi algorithm, while the *grammar processing system* takes care of the transition between phones. The communication between these subsystems is done using "grammar nodes". Associated with a grammar node is a probability that gives the probability function that a phone starts (source grammar node) or that a phone ends (destination grammar node). These nodes are not states in the hidden Markov model, which means, a transition into a grammar node does not consume a frame delay, and they do not output a speech segment. Their purpose is solely to formalize the communication between the subsystems.

The grammar subsystem multiplies the destination grammar node probabilities (DGNP) with transition probabilities to source grammar nodes (see Fig. 1). The source grammar node probability (SGNP) of a certain phone is the maximum probability of all the incoming transitions.

The recognition hardware is partitioned according to Fig. 2: The phone processing system and the part of the grammar system that computes the best SGNP is implemented on a custom board using application-specific integrated circuits. The computation of the product of the DGNP with the transition probability is performed on general-purpose hardware. Thus, different algorithms to dynamically derive the transition probabilities between phones can be implemented on the general-purpose hardware while the computationally most intensive part of the grammar system, finding the best SGNP, can be done with custom VLSI hardware.

Fig. 3 shows the overall architecture of the custom board. At any given frame two processes, each implemented with three custom VLSI processors, are operating in parallel. One process computes the state probabilities of active phones that are listed in the Activeword Memory (Viterbi process) while the other process generates a list of active phones for the next frame (ToActiveWord process).

2.1 Viterbi process

The Viterbi process sequentially reads active phones from the ActiveWord Memory and computes their state probabilities. Based on a pruning threshold derived from the best state probability of that current frame, the Viterbi process decides whether the phoneme should stay active in the next frame and/or whether it has a high probability to end so that succeeding phonemes can be activated. Based on this decision, information associated with this phone is sent to the ToActiveWord processor and/or to the general-purpose grammar processor. To prevent arithmetic overflow, the Viterbi process also normalizes probabilities based on the best state probability of the previous frame.

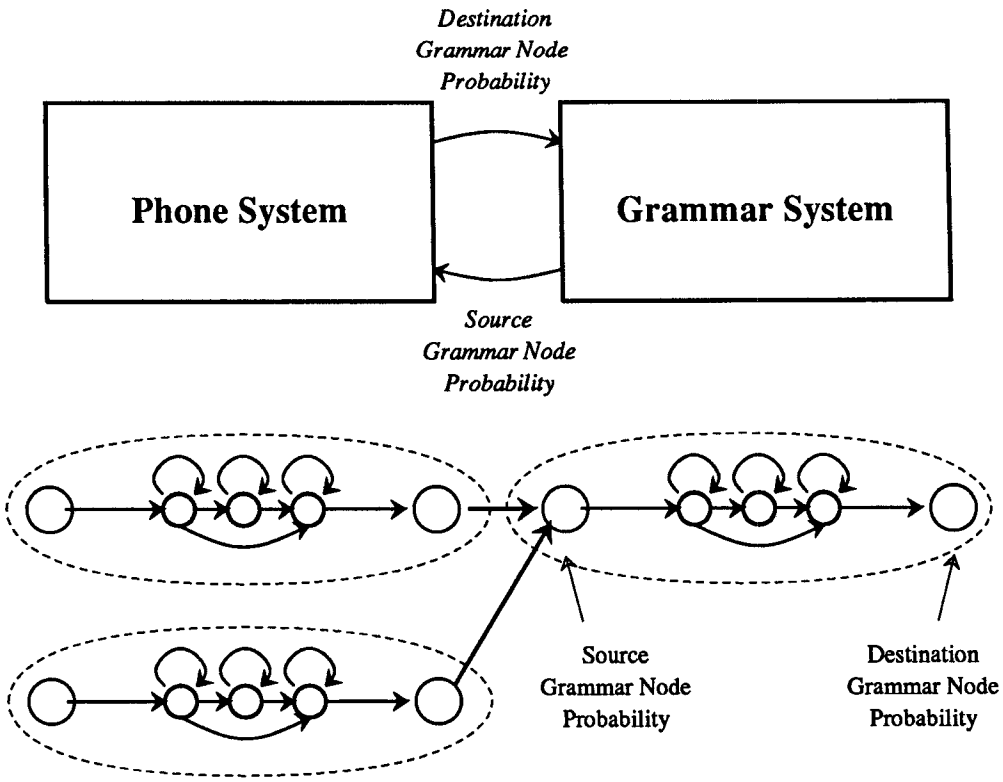

Figure 1: Basic Architecture of the Speech Recognition Hardware

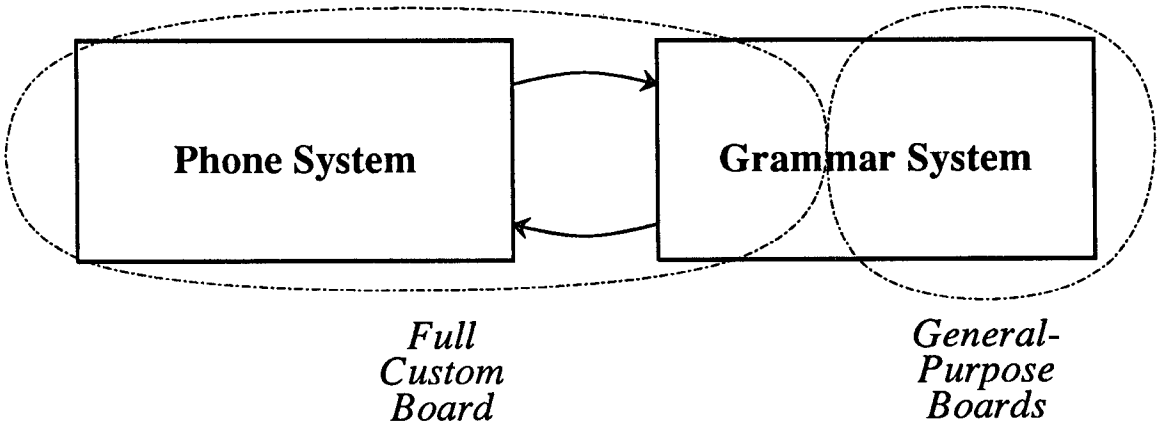

Figure 2: Hardware Partition

The model parameters that describe the topology of phonemes are partitioned into two memories. One memory is located on the prob and the back processor (see Fig. 3) and describes the graph of the hidden Markov chain for certain prototype phonemes. This description can span up to 128 states, partitioned into up to 32 prototype phonemes. The other memory is an off-chip static memory that contains the transition probabilities of up to 64,000 unique phonemes. Thus, the topology of a phoneme is defined with a 5-bit value to indicate the graph and a 16-bit address that specifies the transition probabilities.

To reduce the memory bandwidth the processors contain a dual-ported register file to cache the state probabilities of the previous frame (see [1]).

2.2 ToActiveWord process

The ToActiveWord process has two inputs: it gets information from the Viterbi process associated with phones that were active in the current frame and should stay active in the next frame. The other input is from the grammar processor that gives information about phonemes that are newly activated because their predecessor phonemes had a high probability to end. Given these inputs, the ToActiveWord process generates a list of active phonemes for the next frame. A certain phoneme can be activated several times because it might be activated by the grammar processor as well as by the Viterbi process. Also, the grammar processor could activate the phoneme several times, especially if it is the first phoneme of a word with several predecessor words that have a high probability to end. To avoid replication in the ActiveWord Memory, the ToActiveWord process merges all these different instances of an active phoneme into one, based on the best probability that this phone starts.

2.3 Caching model parameters

To decrease the amount of memory on the system board, we use a caching scheme for the output probabilities, the parameters with the biggest storage requirements: only a small subset of these parameters–the subset that corresponds to the output probabilities for a given speech segment–are loaded onto the board. This loading operation is overlapped with the processing of the frame whose output probabilities had been downloaded in the previous frame. With this approach it is possible to use different modeling techniques for computing the output probability distributions. The current approach is to use as many as four independent discrete probability distributions that are stored and combined on the "Output Distribution Board." Other modeling approaches such as continuous distributions and tied mixtures are also possible, as long as the probabilities can be computed and loaded in real time.

2.4 Switching processors

A frame is processed if the Viterbi process finished the computation of the state probabilities of the active phones in the ActivePhone Memory and if the ToActiveWord process finished the generation of the list of active phones for the next frame. Conceptually, the ActiveList Memories as well as the memories containing the state probabilities have to be swapped before the next frame can be processed. However, instead of swapping the memories, we activate a second set of processing elements that are connected to the memories in the right way. Fig. 4 sketches this principle. During frame A the ToActiveWord process A is active and builds up the ActiveWordMemoryA. Simultaneously, ViterbiB is active and processes the active phonemes listed in the ActiveWordMemoryB. In the next frame, ViterbiB

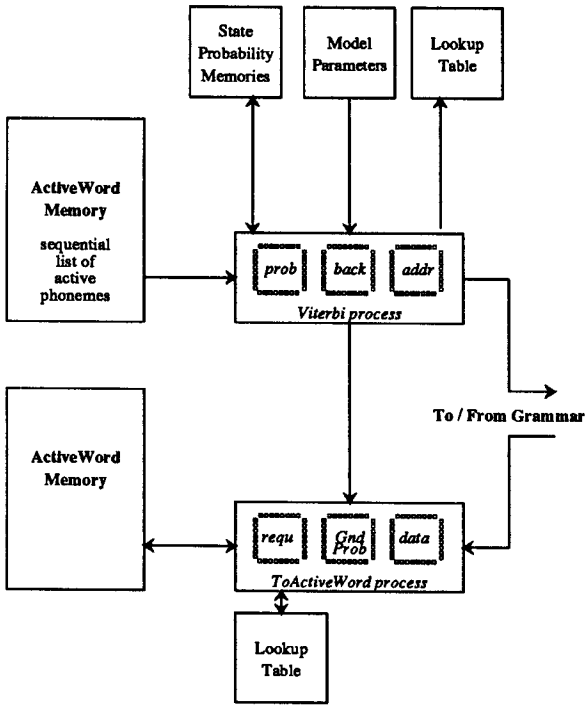

Figure 3: Basic Architecture of the Phone Processing System

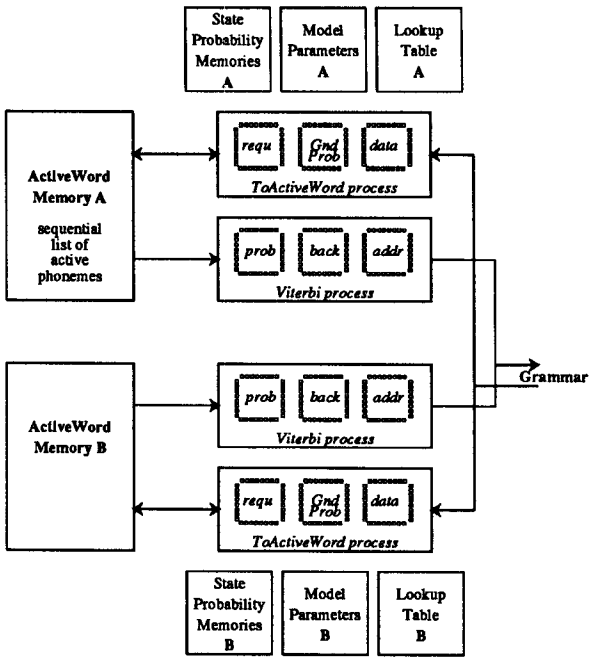

Figure 4: Switching Processors

and ToActiveWordA are inactive and ViterbiA and ToActiveWordB are active. This way, no multiplexors are needed to swap memories. All that is required is to activate the right set of processors. This approach also has the advantage that the complete system is symmetric: the subsystem that has the elements A is identical to the subsystem with elements B.

3 Implementation

All the memories on the system are accessible by the host CPU via VME bus. To reduce the number of discrete components on the system, the host CPU communicates only to the custom VLSI processors. These processors have a small instruction set to read and write memories and internal status registers. Using this approach, no address or data bus has to be multiplexed.

The testing strategy for the custom processors is scanpath testing. Individual chips can be tested by using a generic scantest setup, or they can be tested on the board by using the existing VME interface. A dedicated on-chip test controller supervises this VME test mode so that even the VME interface controller can be tested. This way, every state on the complete board (except the test controller itself) is observable and controllable without a change of hardware.

The board has two copies of six generic VLSI processors that implement the ToActiveWord and Viterbi processes. The chips were designed with the Berkeley LagerIV silicon assembly system [2] and are currently under fabrication using a 2-μm CMOS technology. The table below summarizes the statistics for the processors.

VLSI Custom processors			
name	transist.	size [mm^2]	Pads
	Viterbi chipset:		
Prob	57,000	12.1x12.2	204
Back	38,000	11.2x11.8	204
Add	12,000	10.5x10.5	204
	ToActiveList chipset:		
Request	27,000	10.5x11.2	204
GndProb	5,530	4.9 x 5.6	83
Data	15,700	11.2x10.5	204
HS_Int	19,800	6.7 x 5.8	108
SH_Int	31,900	6.9 x 7.4	130

4 Status and future work

All the chips listed have been designed and verified. They are currently being fabricated through MOSIS using a 2-μm CMOS process. We have received some of the chips back, and are currently testing them and building the two custom boards. After completing the construction of the current hardware design, we will be developing software tools to support this architecture, and to run recognition experiments and real-time systems.

Once we have completed the construction of the first system, we will evaluate the current architecture to determine the computational and algorithmic bottlenecks. To fully use the capabilities of this design we will be developing a large vocabulary recognizer to run on this board. A major area of research will be the design and implementation of algorithms for real-time grammar processing computation, since these parts of the system will be running on general-purpose CPUs (TMS320C30s communicating with SUNs).

5 Conclusion

We have presented a novel architecture that uses full custom integrated circuits to perform hidden Markov-moddel-based speech recognition for large vocabularies in real time. The system will be at least by a factor of 50 more powerful than existing solutions.

References

[1] J. Rabaey, R. Brodersen, A. Stölzle, S. Narayanaswamy, D. Chen, R. Yu, P. Schrupp, H. Murveit, and A. Santos, *VLSI Signal Processing III*, chapter A Large Vocabulary Real Time Continuous Speech Recognition System, pages 61–74, IEEE Press, 1988.

[2] C. S. Shung, R. Jain, K. Rimey, E. Wang, M. B. Srivastava, E. Lettang, S. K. Azim, P. N. Hilfinger, J. Rabaey, and R. W. Brodersen, An Integrated CAD System for Algorithm-Specific IC Design. In *22nd Hawaii Int. Conf. System Science (HICSS-22)*, January 1989.