

Fig. 11. Simulated internal delay time components and operating waveforms.

put, and data output are shown. In this measurement, there was no loading of the data output and the input and output voltages are measured on bonding pads in the chip, resulting in a little faster access time than in a standard condition measurement. Simulated internal delay time components of a critical path are summarized in Fig. 11. An address input and data output waveforms of a critical access path are also shown in Fig. 11. Address access time is 6 ns at $T_a = 25^\circ\text{C}$, and $V_{CC} = 5$ V with a 30-pF load connected to each of the four common input/output pins. The fabricated SRAM's have been tested using N^2 test patterns. All four outputs' functional ability has been measured for a power supply voltage of 4 to 7 V.

V. CONCLUSION

A 256-kb BiCMOS TTL SRAM has been demonstrated with an address access time of 6 ns. The high performance of the SRAM is due to the new BiCMOS circuit technologies. These include: 1) a low-input-capacitance BiCMOS gate which reduced the gate loads in a decoder, 2) a reduced-load multiplexer-line sense amplifier, and 3) the two-level-presetting architecture of the TTL output buffer which reduced the output-drive-current change rate to 20 mA/ns for a $\times 8$ -b configured chip with a propagation delay time of 1.5 ns. The current change rate is about half of the conventional-type output buffer.

ACKNOWLEDGMENT

The support and direction provided by M. Yoshimura, T. Itou, Y. Sakai, K. Motohashi, M. Nishihara, M. Okamura, and N. Momma are appreciated.

REFERENCES

- [1] M. Takada *et al.*, "A 5 ns 1 Mb BiCMOS SRAM," in *ISSCC Dig. Tech. Papers*, Feb. 1990, pp. 138-139.
- [2] Y. Maki *et al.*, "A 6.5 ns 1 Mb BiCMOS ECL SRAM," in *ISSCC Dig. Tech. Papers*, Feb. 1990, pp. 136-137.
- [3] H. Okuyama *et al.*, "A 7.5-ns 32K \times 8 CMOS SRAM," *IEEE J. Solid-State Circuits*, vol. 23, pp. 1054-1059, Oct. 1988.
- [4] Y. Urakawa *et al.*, "11.5 ns 1M \times 1/256K \times 4 TTL BiCMOS SRAM's with voltage- and temperature-compensated interfaces," in *Symp. VLSI Circuits Dig. Tech. Papers*, May 1989, pp. 69-70.
- [5] K. Sasaki *et al.*, "A 9-ns 1-Mbit CMOS SRAM," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1219-1225, Oct. 1989.
- [6] N. Tamba *et al.*, "An 8-ns 256K BiCMOS RAM," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1021-1027, Aug. 1989.

A VLSI Grammar Processing Subsystem for a Real-Time Large-Vocabulary Continuous Speech Recognition System

D. C. Chen, R. Yu, J. Rabaey, and R. W. Brodersen

Abstract—This paper summarizes the architecture and implementation of a grammar processing subsystem which is part of a large-vocabulary continuous speech recognition system. This subsystem contains two custom VLSI chips that perform the evaluation of starting word probabilities associated with the across-word transitions in the hidden-Markov-model (HMM-) based speech recognition system. This system has a maximum computation rate of 200 MOPS and an I/O bandwidth of 265 megabytes/s.

Manuscript received August 2, 1990; revised November 26, 1990. This work was supported by DARPA under Grant 25815.

The authors are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.
IEEE Log Number 9042226.

I. INTRODUCTION

THE hidden-Markov-model (HMM-) based algorithms are currently the most effective techniques used in speech recognition systems [1], [2] and can achieve high word accuracies of 95.5% up to 98.5% (June 1990 DARPA benchmark test). These techniques have superseded the template-based dynamic-time-warp algorithms [3] for high-accuracy speech recognition. Since these algorithms are too complex to be run in real time on general-purpose computers, it is necessary to develop special-purpose hardware.

Our system [4] uses HMM algorithms along with language models to recognize, in real time, continuous speech composed of a 3000-word vocabulary. It will be able to run a

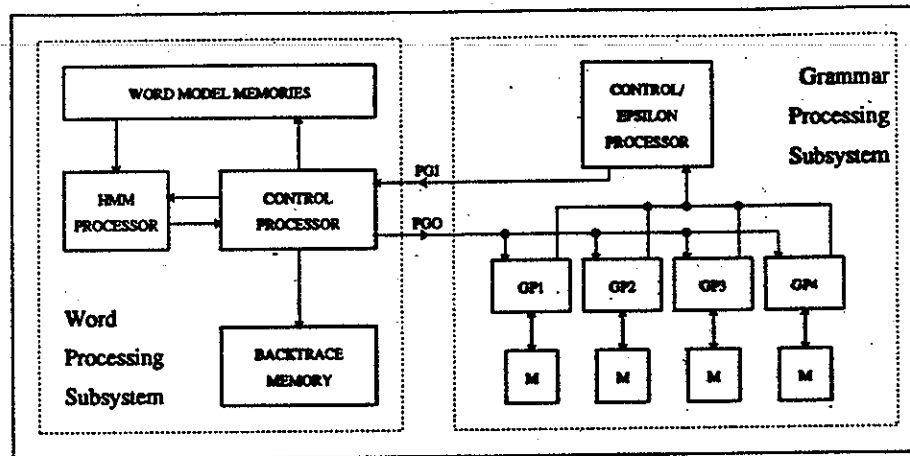


Fig. 1. Block diagram of the word processing and grammar processing subsystems.

class of HMM-based systems being developed at SRI, BBN, and CMU [5], [6].

The speech-recognition portion of our system communicates with a front-end board which does the A/D conversion and acoustic extraction, and with a 68020-based microprocessor control board. Fig. 1 shows the key sections of the speech-recognition portion. It is logically and computationally partitioned into two subsystems. The *word processing subsystem*, which performs the Viterbi algorithm to find the probabilities of states within individual words and also the probability that a word ends, was presented in [7]. Speech recognition accuracy is enhanced when syntactic constraints are imposed on the concatenation of individual words in the vocabulary. This task is performed in the *grammar processing subsystem*, which searches for the most probable word sequence given transition probabilities in this end-of-word to start-of-word, or bigram, model. An N -gram model approximates the conditional probability of one word, given all the previous words, as the probability of one word given the past $N-1$ words:

$$P(W_i|W_{i-1}, W_{i-2}, \dots, W_1) \approx P(W_i|W_{i-1}, W_{i-2}, \dots, W_{i-N}). \quad (1)$$

A simple model, the bigram model ($N=2$), represents

$$P(W_i|W_{i-1}, W_{i-2}, \dots, W_1) \approx P(W_i|W_{i-1}). \quad (2)$$

The grammar processing subsystem computes the probabilities that words start by using a set of transition probabilities between words and the probability that a word ends.

The grammar processing subsystem combines the acoustic word sequence probabilities, generated by the word processing subsystem, with precompiled word sequence probabilities generated by a grammar model of the English language. The model currently supported is a statistical grammar which allows transitions between all words. The recognized sentence is the sentence with the highest combined acoustic and language model probabilities.

This paper elaborates upon the algorithm, architecture, and implementation of the grammar subsystem that was presented in [8].

II. OTHER APPROACHES

A 20000-word speaker-dependent isolated-word recognition system has been constructed at IBM [9]. It runs in real time using programmable digital signal processors. This sys-

tem uses a sophisticated trigram grammar to improve recognition results by emphasizing likely three-word sequences. The trigram model ($N=3$) represents

$$P(W_i|W_{i-1}, W_{i-2}, \dots, W_1) \approx P(W_i|W_{i-1}, W_{i-2}). \quad (3)$$

The reason why the IBM system can recognize such a large vocabulary using a sophisticated grammar is that it is restricted to isolated-word speech, allowing grammar processing to be done only once per word (about once a second) instead of once per spectral sample (about 100 times per second). Grammar processing for trigram grammars is obviously much greater for continuous speech than for isolated speech. Isolated speech also significantly reduces the number of word candidates to choose among in a given position. Also, because words cannot overlap each other, pruning is more effective. Because of these differences between isolated and continuous speech, the IBM hardware approach is not applicable to continuous speech recognition.

Other researchers [10] have constructed a system (BEAM) for 1000-word continuous speech recognition using several parallel microprocessors and digital signal processors. BEAM approaches real-time performance when the system uses a constrained bigram grammar. This is also referred to as a finite state grammar where only certain word sequences are allowed. BEAM can process approximately 8000 candidates (in this case, states in an HMM speech-recognition system) in real time per 10-ms frame. This may be sufficient for a 1000-word vocabulary using a finite state grammar, but with a larger vocabulary and a more complex grammar, as many as 100000 to 300000, such candidates will have to be processed per frame. Thus, a technology based on general-purpose processors is expected to require considerable hardware replication to run this task.

We have found that the largest bottleneck in the system which we have designed for large-vocabulary real-time continuous speech recognition is located in the access of the memories [4]. The architecture exploits several techniques including task and memory partitioning, hardware replication, wide I/O, and single-cycle memory reads and writes to overcome this bottleneck. The throughput requirements are achieved using extensive pipelining in our special-purpose data paths (13 levels in the word processors and five levels in the grammar processors). To our best knowledge, these

techniques will enable a level of speech processing not achievable by existing means.

III. ALGORITHM

Within each frame of 10 ms, the grammar subsystem evaluates the probabilities that words in the vocabulary start and keeps pointers to the most probable paths leading to those words. These pointers are used to reconstruct, or backtrace, the most probable word sequence upon sentence completion. The algorithms used to evaluate the word probabilities are based on two models: the statistical grammar model and the ϵ model. The statistical model is used to handle word arcs (transitions) with high probabilities, and the ϵ model is for word arcs with low probabilities.

A. Statistical Model

The statistical grammar model allows any word to follow any other word. Associated with the i th word produced by the word processing subsystem is a probability PGO_i , the probability that the word i ends at a particular point in time. The grammar subsystem calculates a probability PGI_j , the probability that word j starts in the next frame. This j th successor word probability is then sent back to the word processing subsystem. The probability $PGI_j^G(t+1)$ under the statistical grammar model is found by using

$$PGI_j^G(t+1) = \max_{i \in \text{all words}} [PGO_i(t) \times c_{ij}] \quad (4)$$

where c_{ij} is the transition probability from word i to word j . The evaluation in (4) is terminated when the probability falls below a programmable threshold.

B. ϵ Model

For a vocabulary of 3000 words, a fully connected graph would require 9000000 transition evaluations and storage locations. To alleviate the excessive computational and storage requirements associated with modeling a fully connected graph, we have adopted a simplified model, called the ϵ model, to handle word arcs with low probabilities. We shall see later how storage requirements are reduced from N^2 to $2N$ for a vocabulary size N by using the ϵ model.

In this model, transitions with high probabilities are still treated as before as shown in modified equation (5):

$$PGI_j^P(t+1) = \max_{i \in \text{all high probability words}} [PGO_i(t) \times c_{ij}]. \quad (5)$$

c_{ij} 's with low probabilities are approximated by probability ϵ_{ij} , where $\epsilon_{ij} = \epsilon_i \times \epsilon_j$. ϵ_i represents a probability out of the i th word, and ϵ_j represents a probability into the j th word. The probability $PGI_j^\epsilon(t+1)$ calculated under the ϵ model is found similarly:

$$PGI_j^\epsilon(t+1) = \max_{i \in \text{all words}} [PGO_i(t) \times \epsilon_i] \times \epsilon_j. \quad (6)$$

The successor probability sent to the word processing subsystem is the larger of the word probabilities calculated under (5) and (6):

$$PGI_j(t+1) = \max [PGI_j^P(t+1), PGI_j^\epsilon(t+1)]. \quad (7)$$

Associated with the successor word probability PGI_j is a backtrace pointer to the most probable predecessor word.

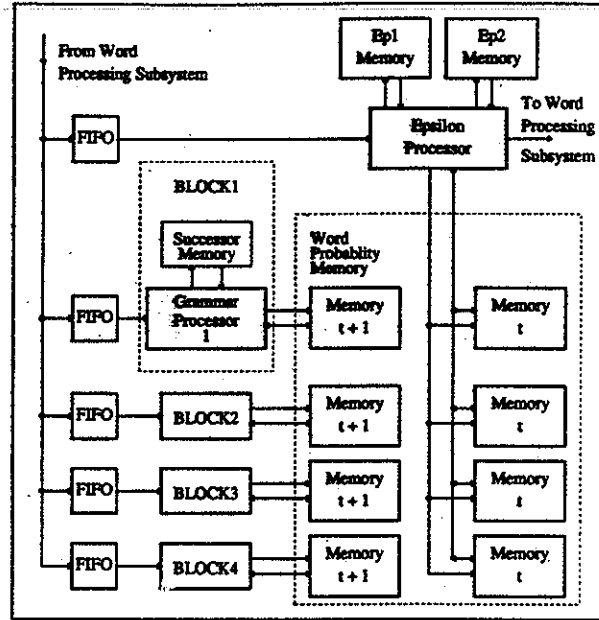


Fig. 2. Block diagram of grammar processing subsystem.

IV. SUBSYSTEM ARCHITECTURE

Fig. 2 shows a high-level block diagram of the grammar processing subsystem. The grammar processor implements (5) and the epsilon processor implements (6) and (7). The grammar processor handles the subset of word arcs with high probabilities and the epsilon processor handles the low probabilities.

A. Task Partitioning

To meet the computational requirements in the statistical grammar model, we partitioned the task into several grammar processors. With a 5-MHz system clock, a single grammar processor can update 50000 starting word probabilities in the 10-ms time frame. This corresponds to an average of 17 successors per word, given a 3000-word vocabulary. A 5-MHz clock was selected since this corresponds to the normal I/O rate of the dynamic random access memories used to store the model.

The initial configuration will have four grammar processors working in parallel to update an equivalent 200000 starting word probabilities, or an average of 70 successors per word. Each grammar processor updates a subset of the total word probability memory. With this partitioning, memory contention among grammar processors is not present and duplication of memory is not required.

With this architecture more grammar processors can be added to handle systems with larger grammars and/or to increase overall throughput.

B. Memories

The word and grammar processing subsystems communicate through several FIFO's, which buffer the word probabilities (PGO) and their associated backtrace pointers. Each processor monitors and accesses its own FIFO, allowing for independent and concurrent operation.

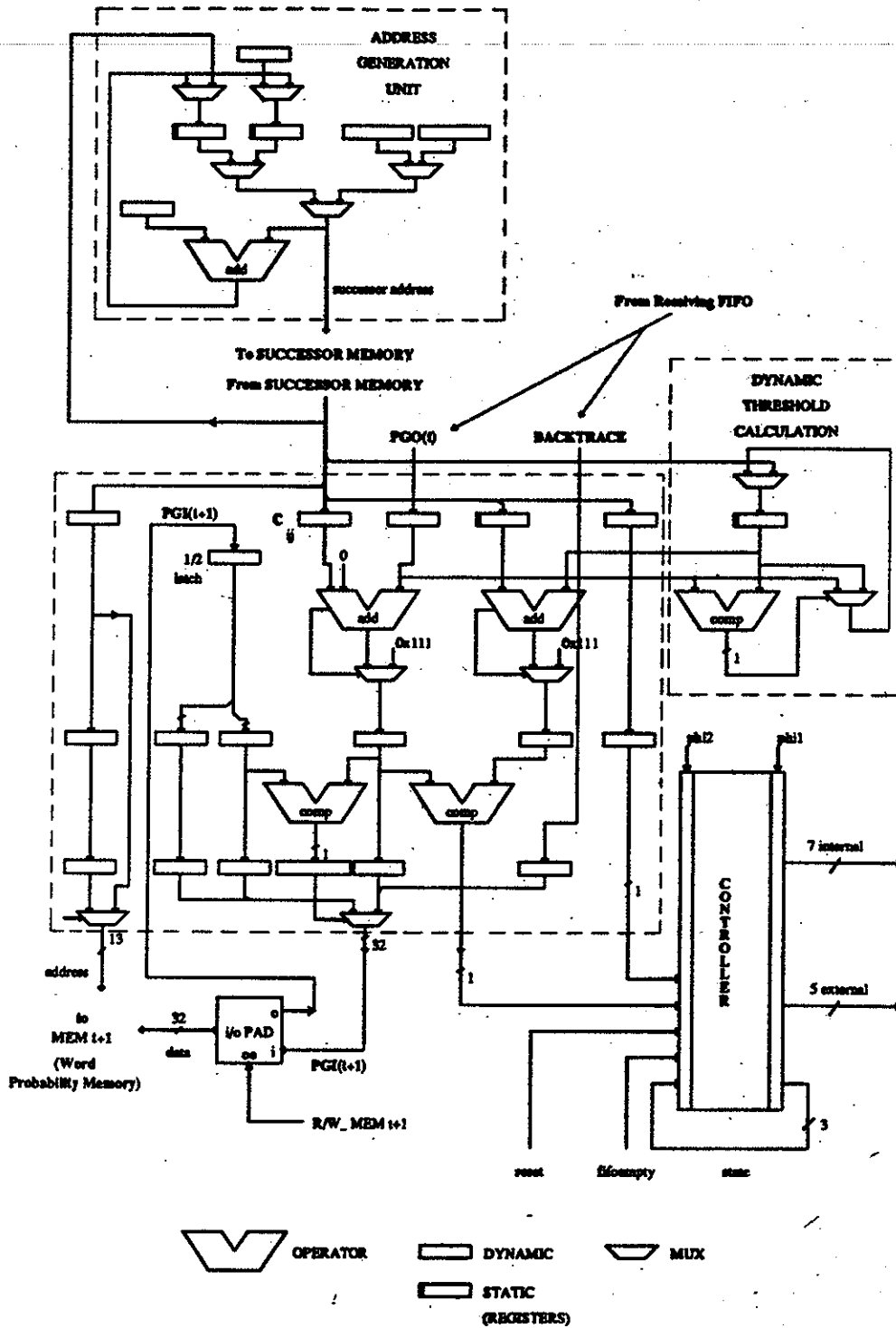


Fig. 3. Detailed architecture of grammar processor.

The probabilities ϵ_i and ϵ_j associated with each word are stored in the Ep1 and Ep2 memories, respectively, and are accessed by the epsilon processor.

Each successor memory is accessed by a separate grammar processor and contains the lists of successor words handled by that grammar processor. An entry in the successor memory contains three fields: the first is an address of the successor in the word probability memory; the second is a

transition probability c_{ij} to that successor; and the third is a flag indicating the end of the current successor list.

The word probability memory consists of two banks corresponding to the starting word probabilities (PGI) at the current and next time frames. Each bank is further subdivided into four groups, representing the successor word partitioning. During a given time frame, the "current" bank is accessed by the epsilon processor, and the "next" bank is

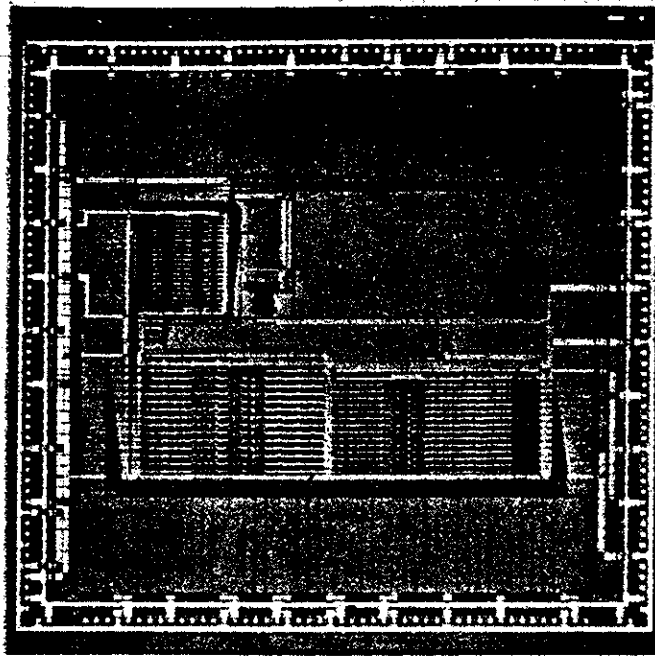


Fig. 4. Die photo of the grammar processor.

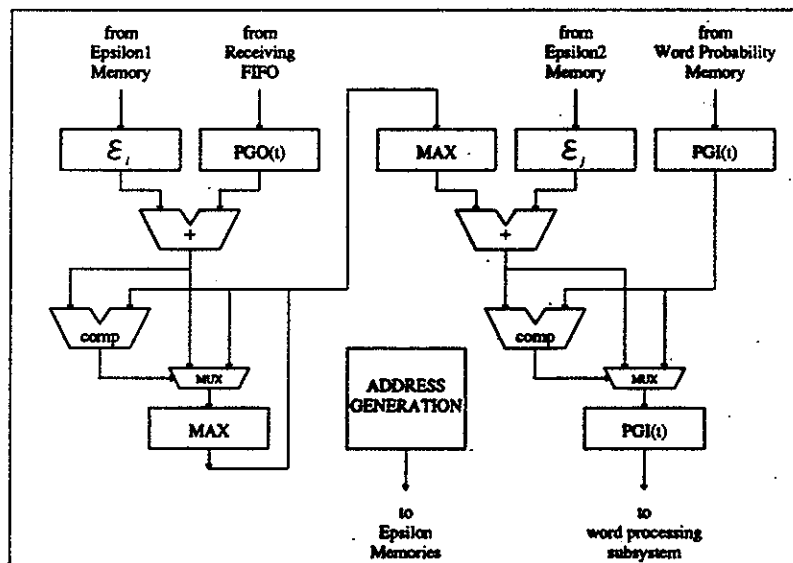


Fig. 5. Simplified block diagram of epsilon processor.

accessed separately by the four grammar processors. The access of these banks is swapped with each successive time frame.

An entry in the word probability memory contains two fields: the word probability itself and the associated back-trace pointer to the predecessor word.

C. Operation

The detailed operation of the subsystem is described in [8].

V. PROCESSOR IMPLEMENTATION

To implement the architecture summarized above, we designed two custom VLSI chips using the Lager IV CAD system [11] and manufactured them in 2- μm CMOS technol-

ogy through the MOSIS facility. This section presents some features of these custom chips.

In both of these chips, arithmetic operations are done in the log domain so that adders can be used instead of multipliers, thereby conserving chip area.

A. Grammar Processor

Fig. 3 shows a detailed block diagram of the architecture of the grammar processor. The main architectural features are: 1) the algorithm is hardwired into the data paths; 2) high performance is achieved through extensive pipelining and parallelism; 3) operators are very simple (add and compare); and 4) many I/O ports are necessary to support the wide communication bandwidth requirements.

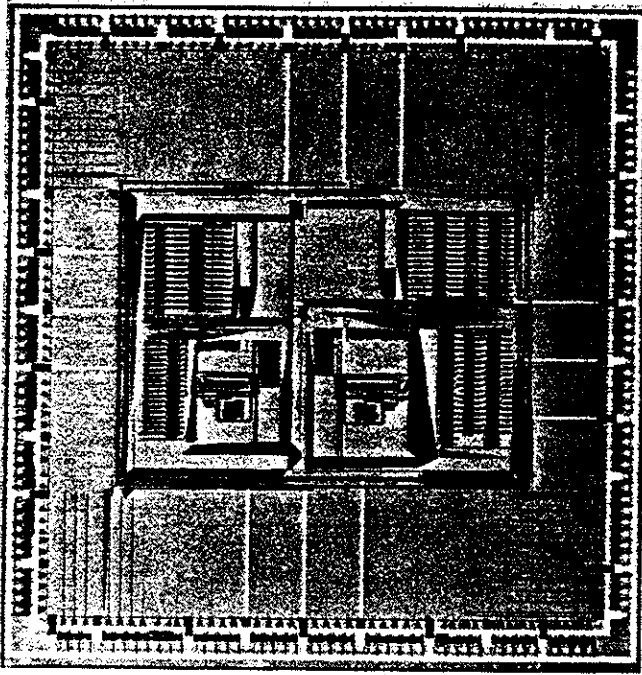


Fig. 6. Die photo of the epsilon processor.

For the i th incoming word, a grammar processor looks into the successor memory to find the precompiled list of successor words and their transition probabilities c_{ij} 's. The probability $PGI_j^P(t+1)$ of the j th successor word is next fetched from the word probability memory, and compared to the product of $PGO_i(t)$ and c_{ij} . The larger of the two is written back into the word probability memory as $PGI_j^P(t+1)$. These operations are pipelined, which meant that the grammar processor had to perform a read and write into different locations in the word probability memory within a 200-ns cycle. This was a critical part of its design and this necessitated the use of fast external static RAM parts, and special timing precautions to accommodate those parts. Because of this feature, a single grammar processor can process approximately 50000 word arcs in the 10-ms frame.

A dynamic threshold mechanism allows the grammar processor to stop processing the remaining successors of a given word once the probability of the successors falls below the threshold. The transition probabilities c_{ij} in the successor memories are stored in decreasing order, so that we can easily detect when this threshold is crossed.

The grammar processor has 142 signal pins, 11385 transistors, and a die size of $9.7 \times 10.5 \text{ mm}^2$. Fig. 4 shows the die photo of the processor.

B. Epsilon Processor

The epsilon processor contains two independent sections, each with its own controller and data paths. Fig. 5 shows a simplified block diagram of the epsilon processor. One section calculates the running maximum over the product of each word probability and its associated ϵ_i value. The other section calculates the product of this maximum value and the associated ϵ_j value and compares it to the word probabilities. The epsilon processor sends the maximum of these two

values to the word processing subsystem according to (7). It also signals completion after all the words in the vocabulary have been sent to the word processing subsystem and all the grammar processors are finished with updating successor probabilities.

The epsilon processor has 157 signal pins, 10775 transistors, and a die size of $9.8 \times 9.7 \text{ mm}^2$. Fig. 6 shows the die photo of the processor.

VI. CONCLUSION

This paper has summarized the architecture and custom implementation of a grammar processing subsystem for a real-time large-vocabulary continuous speech recognition system using hidden Markov models. Our prototype has 3000 words, and larger vocabularies and higher throughput can be obtained by adding more grammar processors. All circuits have been silicon compiled using the Lager IV system [11] and were working on first silicon. The system has been completely simulated in software and construction of the boards is in progress. Architectural enhancements are also being considered to allow for larger vocabularies and more complex grammars.

ACKNOWLEDGMENT

The authors wish to thank H. Murveit, R. Schwartz, A. Santos, and others at SRI and the University of California at Berkeley who have contributed to this project.

REFERENCES

- [1] H. Murveit and M. Weintraub, "1000 word speaker independent continuous speech recognition system using hidden Markov models," in *Proc. ICASSP 87: 1987 Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1987, pp. 115-118.
- [2] K. Lee, H. W. Hon, and R. Reddy, "An overview of the SPHINX speech recognition system," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 38, pp. 35-45, Jan. 1990.
- [3] R. A. Kavaler, M. Lowy, H. Murveit, and R. W. Brodersen, "A dynamic time warp integrated circuit for a 1000 word speech recognition system," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 3-14, Feb. 1987.
- [4] H. Murveit *et al.*, "A large-vocabulary real-time continuous-speech recognition system," in *Proc. ICASSP 89: 1989 Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1989.
- [5] Y. L. Chow *et al.*, "Byblos: The BBN continuous speech recognition system," in *Proc. ICASSP 87: 1987 Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1987, pp. 89-92.
- [6] R. M. Schwartz *et al.*, "The BBN byblos continuous speech recognition system," in *Proc. Speech and Natural Language Workshop*, Feb. 1989, pp. 94-99.
- [7] A. Stölzle, S. Narayanaswamy, K. Kornegay, R. W. Brodersen, and J. Rabaey, "A VLSI wordprocessing subsystem for a real time large vocabulary speech recognition system," in *Proc. CICC'89: 1989 Custom Integrated Circuits Conf.*, May 1989.
- [8] D. C. Chen, R. Yu, J. Rabaey, and R. W. Brodersen, "A VLSI grammar processing subsystem for a real time large vocabulary continuous speech recognition system," in *Proc. CICC'90: 1990 Custom Integrated Circuits Conf.*, May 1990.
- [9] A. Averbuch *et al.*, "An IBM-PC based large-vocabulary isolated-utterance speech recognizer," in *Proc. ICASSP 86: 1986 Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1986, pp. 53-56.
- [10] R. Bisiani, T. Anantharaman, and L. Butcher, "Beam: An accelerator for speech recognition," in *Proc. ICASSP 89: 1989 Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1989, pp. 782-784.
- [11] J. Rabaey, C. S. Shung, R. Jain, and R. Brodersen, "An integrated CAD system for algorithmic specific IC design," to be published in *IEEE Trans. Computer-Aided Design*.