A 40 nm 144 mW VLSI Processor for Real-Time 60-kWord Continuous Speech Recognition

Guangji He, *Student Member, IEEE*, Takanobu Sugahara, Yuki Miyamoto, Tsuyoshi Fujinaga, Hiroki Noguchi, *Member, IEEE*, Shintaro Izumi, *Member, IEEE*, Hiroshi Kawaguchi, *Member, IEEE*, and Masahiko Yoshimoto, *Member, IEEE*

***Abstract*—We have developed a low-power VLSI chip for 60-kWord real-time continuous speech recognition based on a context-dependent hidden Markov model (HMM). Our implementation includes a cache architecture using locality of speech recognition, beam pruning using a dynamic threshold, two-stage language model searching, highly parallel Gaussian mixture model (GMM) computation based on the mixture level, a variable-frame look-ahead scheme, and elastic pipeline operation between the Viterbi transition and GMM processing. The accuracy degradation of the important parameters in Viterbi computation is strictly discussed. Results show that our implementation achieves 95% bandwidth reduction (70.86 MB/s) and 78% required frequency reduction (126.5 MHz) comparing to the referential Julius [1] system. The test chip, fabricated using 40 nm CMOS technology, contains 1.9 M transistors for logic and 7.8 Mbit on-chip memory. It dissipates 144 mW at 126.5 MHz and 1.1 V for 60-kWord real-time continuous speech recognition.**

***Index Terms*—40 nm VLSI, hidden Markov model (HMM), large vocabulary continuous speech recognition (LVCSR), memory bandwidth reduction.**

I. Introduction

SPEECH recognition based on a hidden Markov model (HMM) can provide high recognition accuracy, thus has been used in various applications such as automatic transcripting, audio indexing, navigation, mobile devices, ubiquitous systems, and robotics. Large vocabulary real-time continuous speech recognition (LVRCSR) with acoustic and language models is too resource-hungry and power-sensitive for software applications [1]. Hardware implementation by VLSI or FPGA is demanded especially for use in mobile equipment [2] and intelligent robots because of advantageous high processing speed and low power consumption.

Lin *et al.* investigated FPGA implementations for 5-kWord speech recognition [3], [4]. They increase the data-pin to improve the data-transmission ability of IO, but their architecture is not extendable for a larger vocabulary because it consumes too much power and is not cost-effective: it requires multiple FPGAs and SDRAMs. Yoshizawa *et al.* proposed a scalable architecture for speech recognition [5], but their chip uses 136 mW, even with a limited vocabulary of 800 words. Choi *et al.* investigated FPGA and VLSI implementations [6]–[9] for 5-kWord and 20-kWord speech recognition. They implemented a special memory interface for several parts of the recognition engine to apply optimized DRAM access, which improves the data transfer efficiency. However, the large amount of external DRAM access cause high IO frequency, which requires a high supply voltage and high power consumption in both chip side and DRAM side. Ma *et al.* reported memory-bandwidth reduction of Gaussian mixture model (GMM) processing for real-time 20-kWord speech recognition [10], but that method did not accommodate Viterbi processing. Comparison of power consumption among recently announced hardware-based speech recognizers is presented in Fig. 1. To date, the hardware approach has never achieved real-time operation with a 60-kWord language model because, with the size of vocabulary increasing, the external memory bandwidth and computation workload grows exponentially. For low-power and real-time 60-kWord processing, both the memory bandwidth and the operating clock frequency must be reduced.

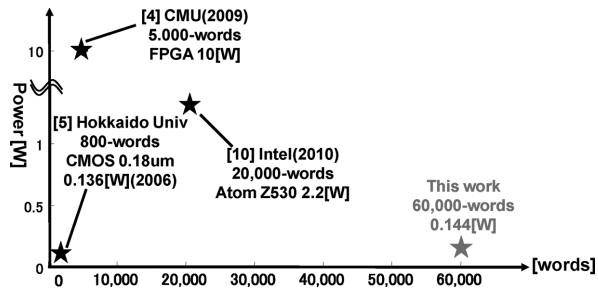


Fig. 1. Comparison of power consumption.

As described herein, we proposed a VLSI implementation for 60-k Word real-time continuous speech recognition. It employs algorithm optimization such as two-stage language model search to reduce cross-word transitions for the Viterbi search [11], beam pruning using a dynamic threshold to avoid sort processing. A variable-frame look-ahead scheme [12] is used to reduce the memory bandwidth for GMM computation. We introduced part of the External DRAM data into the internal cache memory using the locality of speech recognition and proposed a specialized cache architecture to improve the cache hit rate. Elastic pipeline operation between the Viterbi search and GMM processing is applied. We analyzed the trade-off between the accuracy and the important parameters in Viterbi computation to choose the most appropriate parameter combination. Finally, we designed and fabricated a VLSI test chip using 40-nm CMOS

Manuscript received January 15, 2012; revised April 29, 2012; accepted May 08, 2012. Date of publication July 17, 2012; date of current version July 24, 2012. This development was performed by the author for STARC as part of the Japanese Ministry of Economy, Trade and Industry sponsored "Silicon Implementation Support Program for Next Generation Semiconductor Circuit Architectures." The VLSI chip used in this study was fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), University of Tokyo. This paper was recommended by Associate Editor C. H. Chang.

The authors are with the Graduate School of System Informatics, Kobe University, Kobe, 657-0851, Japan (e-mail: achilles@cs28.cs.kobe-u.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2012.2206501

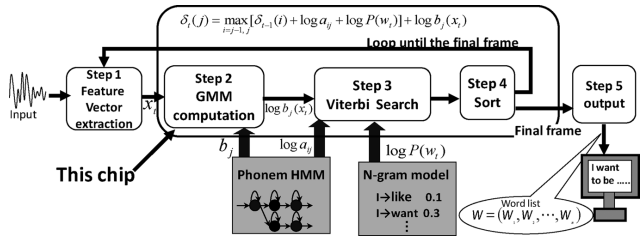


Fig. 2. Speech recognition flow with HMM algorithm.

technology and measured its performance. Measurement results show that our test chip can achieve 60-kWord continuous real-time speech recognition with 144 mW power consumption and only slight accuracy degradation.

The remainder of this paper is organized as follows. The speech recognition algorithm is described in Section II. Section III explains the proposed schemes and accuracy trade-off. Section IV presents the proposed architecture. Section V presents the VLSI implementation and its measurement results. Finally, conclusions are explained in Section VI.

II. Speech Recognition Overview

Fig. 2 presents the speech recognition flow with the HMM algorithm [13]. The following items describe concrete stages. **Step 1:** Feature vector extraction—The input speech signal is converted from the time domain to frequency domain to obtain more unique acoustic characteristics. Feature vectors are extracted from 30 ms length of speech every 10 ms. **Step 2:** GMM computation—A phonemic-model GMM is read and state output probabilities is calculated for all active state nodes. **Step 3:** Viterbi search—$\delta_t(j)$ is calculated for all active state nodes using state output probabilities, transition probabilities, and the N-gram language model. **Step 4:** Sort—according to the beam width, active state nodes having a higher score (accumulated probability) are selected. The others are dumped. **Step 5:** Output sentence—The word list with the maximum score is output as a speech recognition result after final-frame calculation and determination of the transition sequence.

A. GMM Computation

Gaussian mixture models (GMMs) are used to represent the state output probability of HMMs. A fast state-likelihood computation technique [14], [15] using approximate algorithms has been applied in [6]–[9], but it will cause around 0.49% degradation in accuracy. Because the computation work-load problem in GMM processing can be solved by utilizing highly parallel architecture, we implemented the following conventional algorithm for this chip. The GMM computation obtains acoustic likelihood $\log[b_j(x_t)]$ from a feature vector x_t and parameters of a GMM. As expressed in (1),

$$\begin{aligned}
&\log\, b_j(x_t) \\
&\quad = \log \sum_{k=1}^{mix} \lambda_i N(x_t, \mu_k, \sigma_k) \\
&\quad = \log \left[\sum_{k=1}^{mix} \lambda_k \left[\frac{1}{\prod_{i=1}^{p} \sqrt{2\pi\sigma_{ki}^2}} \exp\left\{ -\sum_{i=1}^{p} \frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right\} \right] \right] \\
&\quad = \log \left[\sum_{k=1}^{mix} \exp\left\{ C_k - \sum_{i=1}^{p} \frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right\} \right] \\
&\quad = add \log \left[C_k - \sum_{i=1}^{p} \frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right] \qquad (1)
\end{aligned}$$

$$C_k = \log \left[\frac{\lambda_k}{\prod_{i=1}^{p} \sqrt{2\pi\sigma_{ki}^2}} \right]. \qquad (2)$$

Therein, $b_j(x_t)$ stands for a GMM probability density function (PDF), N represents a Gaussian distribution PDF, P is the number of dimensions in a feature vector, mix is the number of mixtures, x_t signifies a feature vector, μ and σ denotes mean and variance parameter respectively. λ is a mixture weight coefficient and C_k is a constant number. Equation (2) shows that the computation for one mixture consists of P subtractions, 2P multiplications, P summations, and one addition. After that, the add-log operation is taken between the mixture results, which can be processed quickly based on an add-log table.

B. Time-Synchronous Viterbi Beam Search

The following formulas show a time-synchronous Viterbi beam search algorithm [16], which is divisible into two parts: internal word transition and cross-word transition. Dynamic programming (DP) recursion for the internal word transition is shown in (3).

$$\delta_t(s_j; w) = \max_{i=j-1,\ j} [\delta_{t-1}(s_i; w) + \log a_{ij}] + \log b_j(x_t) \quad (3)$$

where a_{ij} is the transition probability from state s_i to s_j, and $\delta_t(s_j; w)$ stands for the largest accumulated probability of the state sequence reaching state s_j of word w at time t. Once an internal word transition reach a word-end state, cross-word transition will be treated, a bi-gram (2-gram) model is used in this chip, where the transition probability of a word depends on the immediately preceding word. DP recursion for this part is shown in (4).

$$\delta_t(s_0; w) = \max_{v} \{\delta_{t-1}(s_f; v) + \log[p(w|v)]\}. \qquad (4)$$

Therein, $p(w|v)$ stands for the bi-gram probability from word v to word w, s_0 and s_f respectively denote the start state of word w and the last state of word v.

In actual speech recognition, the problem is too large to allow for calculation of all the likelihood values. Therefore, after all the transitions in one frame are completed, only a limited number ("beam width") of nodes with large likelihood values remains, the other nodes are terminated. This process is designated as beam pruning. Nodes that are unpruned in this stage are designated as active state nodes. In the next frame,

only the likelihood values for the active state nodes will be computed.

C. Computation Amounts and Memory Bandwidth

We profiled referential hardware of Julius 4.0 [1], a well-known Japanese speech recognition system software program using hardware description language (HDL), with 60-kWord speech recognition models, and beam width of 4000. The required memory bandwidth and the operation frequency of the prototype necessary to achieve real-time speech recognition respectively reach 3446 MB/s and 567.46 MHz. This kind of external memory bandwidth causes high IO frequency, which requires a high supply voltage and high power consumption at the chip side and causes hundreds of microwatts at DRAM side even with the state of the art lower power mobile DRAM [17]. Consequently, reducing the external memory bandwidth is the one of the most important things to implement a low-power speech recognition system.

III. Proposed Schemes

A. Variable- Frame Look-Ahead Scheme

The GMM calculations must load numerous parameters, which requires about 576.03 MB/s when processing the 60-kWord speech recognition. The memory bandwidth can be reduced by sharing the parameter for several frames. Many studies have explored the use of this look-ahead scheme and compute the same state for several frames because the state which must be computed in the present frame might need to be calculated again in the subsequent frame at high probability. However, it is apparent that the probability decreases when the number of look-ahead frames increases. When different states are required, the results will become useless. The computation for those new states will delay the Viterbi operation. For 20-kWord and 60-kWord recognition, it is necessary to maintain sufficient beam width according to the number of words to achieve highly accurate recognition, which is true for almost all states of GMM processing. Therefore, in this study, we compute all 1987 states for the maximum of 50 look-ahead frames. The number of look-ahead frames is variable to make an adjustment between the delay and the memory bandwidth. This scheme requires 5 Mbit internal memory for storing the GMM probabilities. However, the memory bandwidth for GMM processing can be reduced to 13.3 MB/s at most. Because that can be accomplished for all states of GMM computation, pipeline operation between GMM and Viterbi is readily applicable.

B. Modified Unigram Language Model

The Viterbi processing comprises word-internal transitions, cross-word transitions to isolated trees and cross-word transitions to shared trees. A unigram (1-gram) language model was used for computation of word-internal transitions. Each unigram value corresponds individually to a state of HMM trees. In the conventional scheme, when an HMM state transfers, the unigram probability of the previous state is subtracted from the temporal score before the new unigram probability of the current state is added. In terms of a unigram language model, the previous state of every state is individually identifiable. Therefore, we modified the unigram language model to hold only difference values between the probability of a new HMM state and the probability of its previous HMM state. Using our modified unigram language model, the extra memory access to the previous state and the memory to save the previous unigram probability can be reduced. Furthermore, because the unigram update process can be eliminated, word-internal transitions, cross-word transitions to the isolated trees, and cross-word transitions to shared trees can be treated using the same process module.

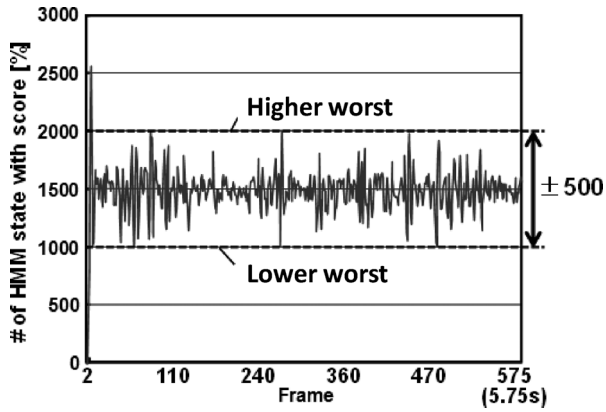


Fig. 3. Beam width variation with dynamic threshold-cut scheme.

C. Beam Pruning Using a Dynamic Threshold

In the conventional process, the sort is implemented after Viterbi processing at every frame before pruning the transitions with a lower score. This pruning necessitates a large workspace because all temporal scores generated by the Viterbi transition must be retained until the Viterbi search of the current frame is completed, although most scores will eventually be pruned by the beam-cutting process. Moreover, sort processing requires computational rates higher than 10 MIPS and demands external memory bandwidth greater than 400 MB/s for 60-kWord recognition.

A threshold-cut scheme is widely used to reduce the sort processing workspace and memory bandwidth. In this scheme, a threshold is set and all the transitions which have a lower score than the threshold are pruned immediately while processing the Viterbi search. Only the selected transitions with a higher score than the threshold are stored in workspace memory, which can cut off the superfluous workspace and processing. However, an improper threshold yields inappropriate cases in which too many nodes remain or too many nodes are cut off compared to the beam width, which degrades accuracy. Therefore, the means of deciding the threshold is important for this scheme.

In some other works [3], [4] , the max score of the current frame is used as the threshold for next frame, which is insufficient because we should also consider the number of active nodes. When there are too few or too many active nodes, the threshold must be adjusted. As described in this paper, we proposed beam pruning using a dynamic threshold. An adaptive threshold is set based on the difference between the average scores of the previous frame and the current frame and the number of active nodes between the previous frame and the current frame. Fig. 3 shows the beam width variation

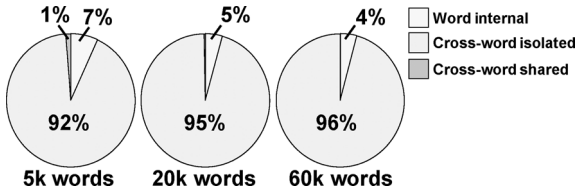


Fig. 4. Appearance ratio of three types transitions in Viterbi search.

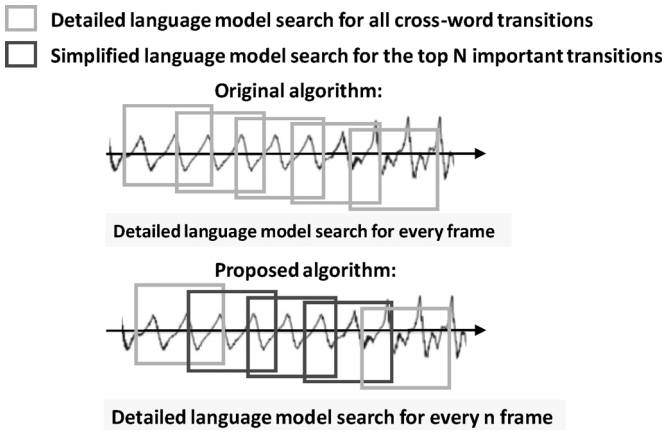


Fig. 5. Two-stage language model search.

with a dynamic threshold-cut scheme when the target width is set to 1500. The threshold-cut results fluctuate ± 500 and the speech recognition accuracy is unaffected because almost all transitions that engender the final speech recognition output word list have higher scores.

D. Two-Stage Language Model Search

Fig. 4 presents the appearance ratio of the three types of transitions in Viterbi search: the cross-word transitions to isolated trees are dominant. Consequently, we proposed a two-stage language model search scheme in this part to reduce the computational workload and memory bandwidth for cross-word transitions to isolated trees. This scheme is derived from the transition frequency difference between phonemic HMM and language HMM. The cross-word transition search is divided into two stages. The first stage is a simplified language model search for the top N important transitions of two-gram probability. The second stage is a detailed language model search for all cross-word transitions. As depicted in Fig. 5, in the traditional language model search, only our second search treated every frame. However, in our proposed language model search, the second stage is treated at every n frames. By applying this proposed search, the computational amount and memory bandwidth can be reduced to $1/n$.

With the increase of the detailed language model search cycle, we can achieve greater reduction of cross-word transitions, which is the main processing undertaken in Viterbi computation. However, the risk of losing the cross-word transition to the correct candidate word might increase, thereby affecting the recognition accuracy. Moreover, the beam width and the number of cross-word transitions during the detailed search and the simplified search strongly influence the recognition accuracy. Therefore, the trade-off of these parameters described above must be discussed carefully.

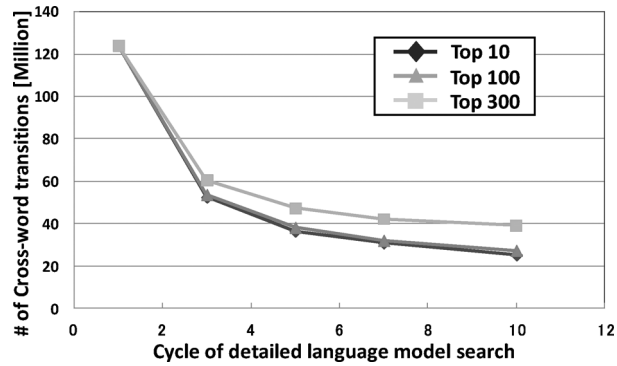


Fig. 6. Cycle of detailed language model search versus the number of cross-word transitions.

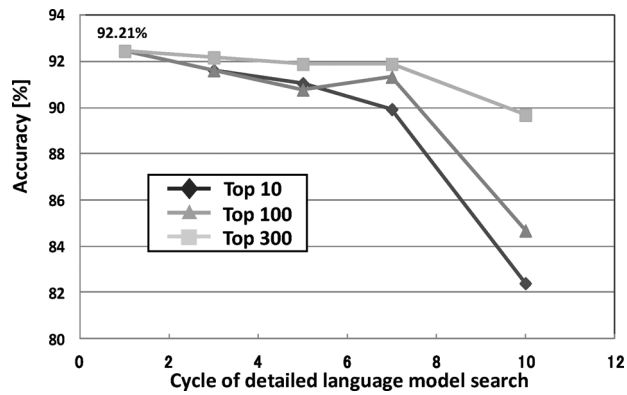


Fig. 7. Cycle of detailed language model search versus recognition accuracy.

E. Accuracy Trade-off

We measured the accuracy using a referential software prototype profiling with Julius 4.0 [1]. The test speech data consists of 48 test patterns, which totally include 172 sentences of Japanese speech spoken by different speakers. The average values of all the patterns for each parameter set are shown in the following graphs.

1) Detailed Search Cycle, the Number of Cross-Word Transitions (in Simplified Search): First, the trade-off of the detailed language model search cycle and the number of cross-word transitions during the simplified language model search are discussed. The beam width is set to 4000. The cross-word transitions during the detailed language model search are set to 2000 to maintain high recognition accuracy.

Fig. 6 presents the relation between the detailed search cycle and the number of cross-word transitions. Top 10, Top 100, and Top 300 respectively signify the numbers of cross-word transitions during the simplified language model search. As portrayed in Fig. 7, more cross-word transitions during a simplified search can suppress the decrease in recognition accuracy. However, when the detailed search cycles became greater than 7, all curves became steeper and the recognition accuracy falls below 90%. Moreover, the accuracy of "Top 10" is greater than "Top 100," with a detailed search cycle of 5 and is lower than "Top 100" with a detailed search cycle of 7. Some words that can be recognized correctly with "Top 10," but will not become the best results with "Top 100." This is because for these words, even if the correct candidates are served, they will be defeated by some other cross-word transitions served by "Top 100."

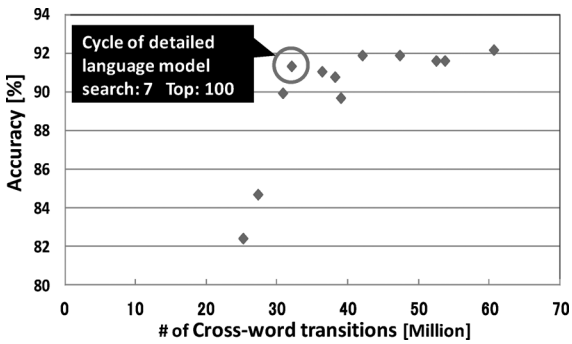


Fig. 8. Relation between accuracy and the # of cross-word transitions.

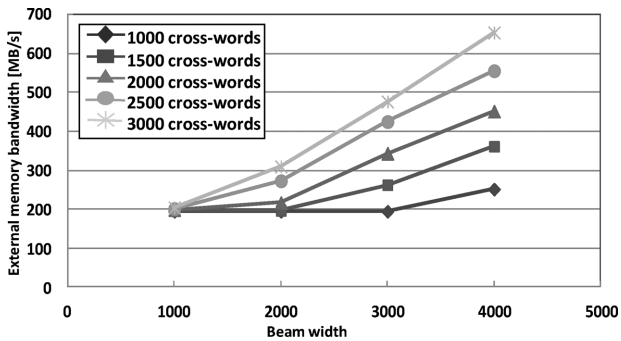


Fig. 9. External memory bandwidth versus beam width.

Fig. 8 portrays the relation between the number of cross-word transitions and recognition accuracy. A detailed search cycle of 7 with "Top 100" is chosen for the examinations described in this paper, causing 0.27% accuracy degradation. These parameters are alterable in the chip. Therefore, we can select them according to the request processing speed and recognition accuracy, but higher accuracy will cost more power because large amounts of extra cross-word transitions must be treated.

2) Beam Width, the Number of Cross-Word Transitions (in Detailed Search): In this section, the trade-off of the beam width and the number of cross-word transitions during the detailed language model search are discussed. The detailed search cycle is set to 7 and the number of cross-word transitions during simplified searching is set to 100, in light of the previous discussion.

Fig. 9 presents the external memory bandwidth for Viterbi computation according to the beam width. The numbers of cross-word transitions during detailed searching were set to 1000, 1500, 2000, 2500, and 3000. Fig. 10 presents the accuracy degradation. It is apparent that 1500 for the cross-word transition during detailed search is a good choice since it demands relatively smaller external memory bandwidth and less degradation in recognition accuracy with 3000 of beam width.

3) Default Parameters: The final choice of the parameter combination is presented in Fig. 11. Total accuracy degradation by the proposed schemes is 0.82%, thereby achieving almost 80% reduction $(1500 + 100 * 6)/(1500 * 7) = 20\%)$ of both memory access and arithmetic operations in Viterbi processing.

From these test results, we also found that larger beam width can absolutely offer higher recognition accuracy while treating more cross-transitions might not. Therefore the parameters for cross-word transition must be decided carefully, the most appropriate parameter combination may change when using different language models.

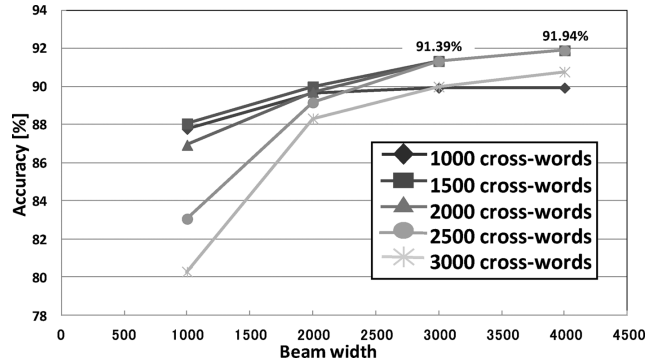


Fig. 10. Accuracy versus beam width.

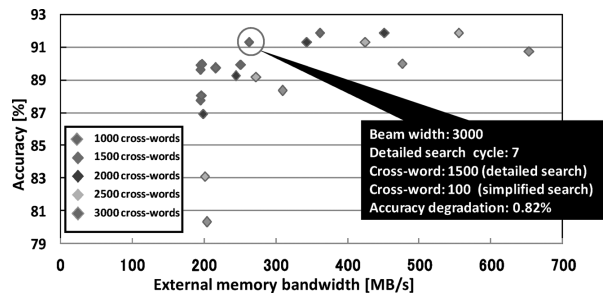


Fig. 11. Accuracy vs. External memory bandwidth.

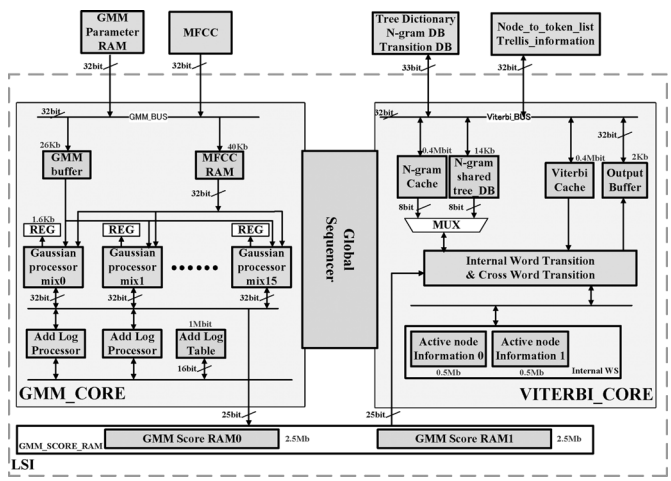


Fig. 12. Proposed speech recognition architecture.

IV. Architecture

A. Elastic Pipeline Architecture

The overall chip architecture is depicted in Fig. 12. The proposed architecture comprises a global Sequencer, GMM-core, Viterbi-core, and double GMM result buffer to support pipeline operation. Because of the all-state GMM computation and variable 50-frame look-ahead scheme, it is easy to apply elastic pipeline operation between the Viterbi transition and GMM processing by 1–50 frames. Herein, we will explain why the

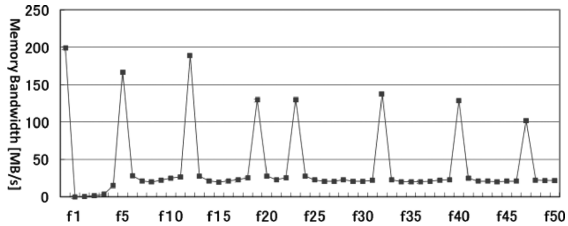


Fig. 13. External memory bandwidth variety of Viterbi transition through frames after the two-stage language model search.

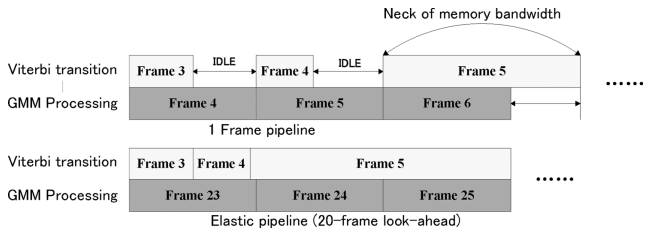


Fig. 14. Elastic pipeline.

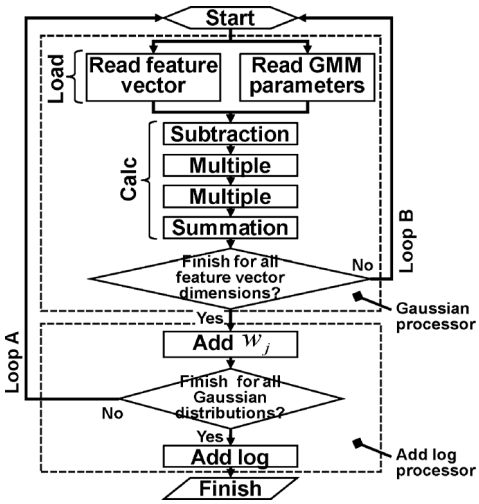


Fig. 15. GMM computation flow.

elastic pipeline architecture can reduce the memory bandwidth for Viterbi transition. Fig. 13 shows the memory bandwidth variety of Viterbi transition after applying the two-stage language model search. The frames, when detailed search are used, have the largest amount of data to be read. For conventional operation, these data must be read at 0.01 s, which yields the largest memory bandwidth (549.91 MB/s). By applying this scheme, we can process several frames together, although the frames needing the largest memory bandwidth can use the IDLE time of other frames to load data (Fig. 14), which can reduce the peak memory bandwidth by 87.2% (70.86 MB/s). By changing the number of look-ahead frames, we can readily adjust the delay and the memory bandwidth and maximize the elastic pipeline operation efficiency.

B. Highly Parallel GMM Architecture

Fig. 15 shows the operation flow of the GMM calculation, which consists of data loading, mixture computation and add-log processing. The GMM core has 16 mixture processing blocks, each of which has a 1.6 Kb register to preserve the mixture parameter. All blocks are processed simultaneously for the look-ahead frames, which are saved in the MFCC buffer.

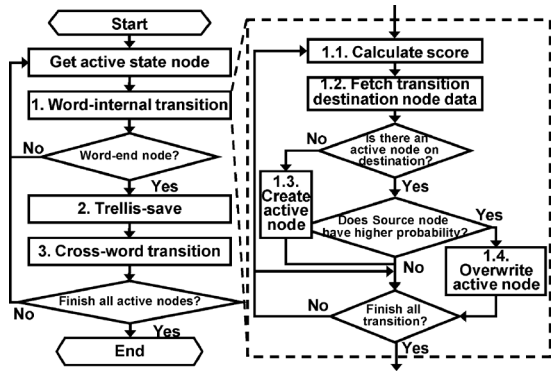


Fig. 16. Viterbi computation flow.

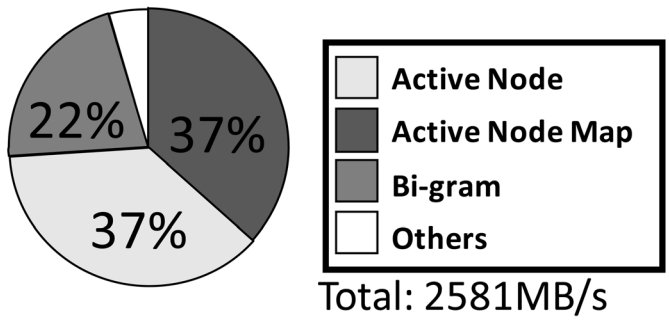


Fig. 17. External memory bandwidth of Viterbi transition.

The parameters will be reused until all look-ahead frames are processed. The mixture results will soon be calculated using the add-log processor based on a look-up table. The mixture computation, add-log calculation, and parameter reading are processed in the pipeline.

C. Viterbi Cache Architecture

Fig. 16 shows the Viterbi transition flow. The Viterbi transition in a frame is divided roughly into three steps; word-internal transition, trellis-save [1], and cross-word transition.

The Viterbi transition is performed for all active state nodes left in the previous frame. First, fetch an active state node from an active node queue. 1) *Word-internal transition*: perform word-internal transition when the transition source node and destination node belong in the same word. 2) *Trellis save*: save a trellis when the active state node is the end of a word. The trellis is a dataset, which has a word history and a score of the word-end node. It is used to determine the recognition result in the last frame. 3) *Cross-word transition*: perform cross-word transition after trellis save. In this step, the transition is performed from a word-end state node, to all word-start state nodes.

The word-internal transition and cross-word transition can be expressed with the same flow. 1.1) *Calculate score*: The transition probability from the HMM dictionary and the GMM probability from the result buffer is added to the score of the active state node. 1.2) *Fetch transition destination node data*: fetch the information of a transition destination node from the active node work space. 1.3) *Create active state node*: create an active state node when there is no active state node on the transition destination. 1.4) *Overwrite the active state node*: overwrite the active state node at a destination node when its score is lower.

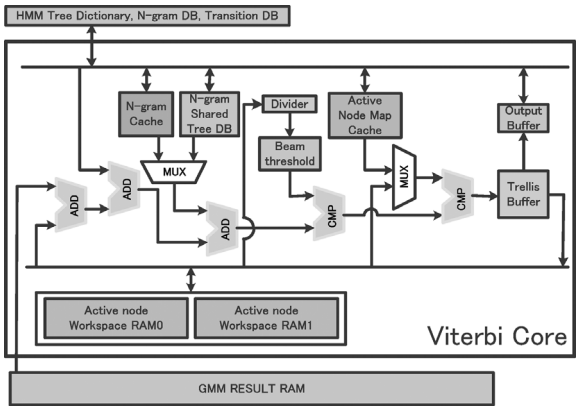


Fig. 18. Viterbi Cache Architecture.

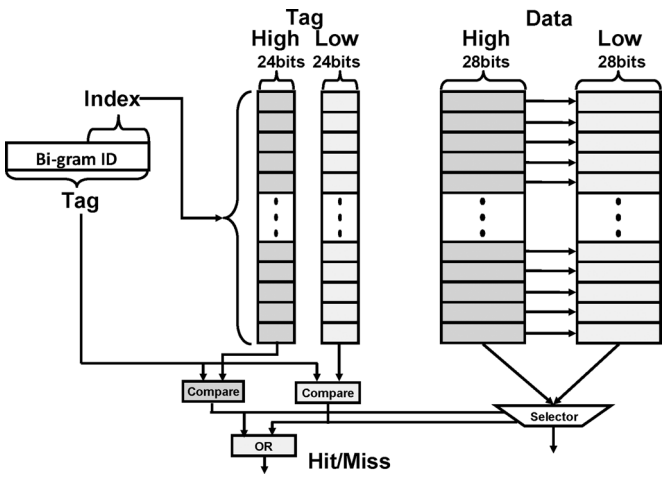


Fig. 19. Proposed two-way bi-gram cache.

Fig. 18 shows the Viterbi cache architecture. Since it is apparent that the Active Node, Active Node Map, and the Bi-gram account for 96% of the memory bandwidth for Viterbi transition, as portrayed in Fig. 17, we introduce all the active node work space and part of the bi-gram and active node map data into the cache memory using the locality of speech recognition that some data which have been used for this frame might be reused in the following frames. We employ some novel schemes for these caches to achieve a higher hit rate.

1) Bi-Gram Cache: We try to retain the bi-gram data with higher scores in the cache because they have higher probability to be reused in the subsequent frame. We set the timing of writing new data to the cache as a previous node is overwritten whereas a higher score appears in the cross-word transition. In doing so, a bi-gram probability with a higher score in one frame tends to remain in the cache. However, one index of the bi-gram cache stands for many bi-gram IDs. Therefore when a new bi-gram score is written to cache, a high-score bi-gram for other nodes might be overwritten, which affects the hit-rate.

Therefore, we proposed a specialized cache architecture as shown in Fig. 19, the cache adopts a two-way set associative scheme for both tags and data. The lower bit of bi-gram ID is used to create the index, and the whole bi-gram ID is used as the tag. The data part has 20 bit for the destination node ID and 8 bit for bi-gram score. The two-way cache has "two places" for each

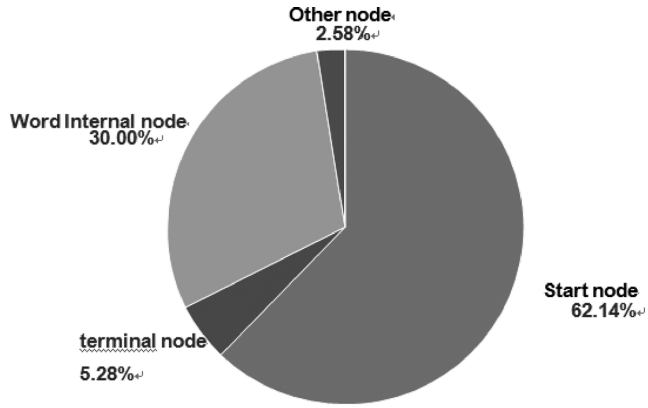


Fig. 20. Memory accesses of active node.

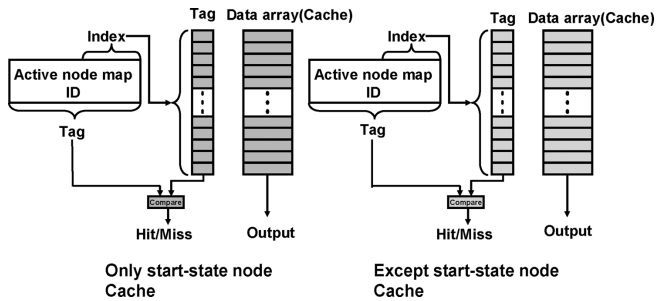


Fig. 21. Active node map cache.

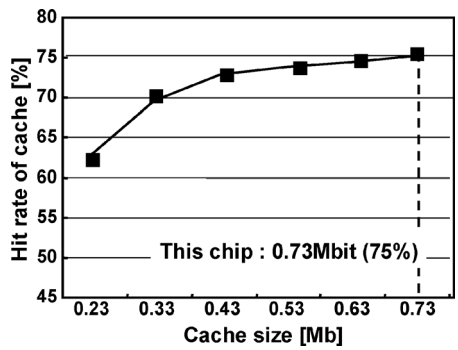


Fig. 22. Cache hit rate.

index. Therefore, when a miss-hit occurs, a new bi-gram probability is written to the first way, whereas the previous score can be stored to the second way. The hit rate is improved by 14.5% because two-way cache can retain more high-score bi-gram data than a one-way cache.

2) Active Node Map Cache: The active node map is also called a "Token list," which is used to check if an active node exists in the transition destination. It must be checked every time a transition occurs, thereby causing many memory accesses. We adopt a direct mapping cache for it.

As presented in Fig. 20, more than 60% of the node information accesses are attributable to the start-state node because cross-word transitions must read them frequently, however, one index of the cache stands for start-state node, word internal node, terminal node and other nodes. The start node may be overwritten by other nodes if they stay in the same cache. Therefore we divided the active node map cache into two parts: one for the start-state node only, and the other nodes use the other part (Fig. 21), thereby improving the hit rate by almost 20%.

TABLE I
IMPLEMENTATION FLOW

Implementation Flow	Implementation Tool
RTL Simulation	NC-verilog (Cadence)
Logic Synthesis	Design Complier (synopsys)
Placement and Routing	Astro (Synopsys)
Layout Design	Virtuoso (Cadence)
Design Rule Check	Dracula (Cadence)
DRC / LVS	Calibre (Mentor Graphics)

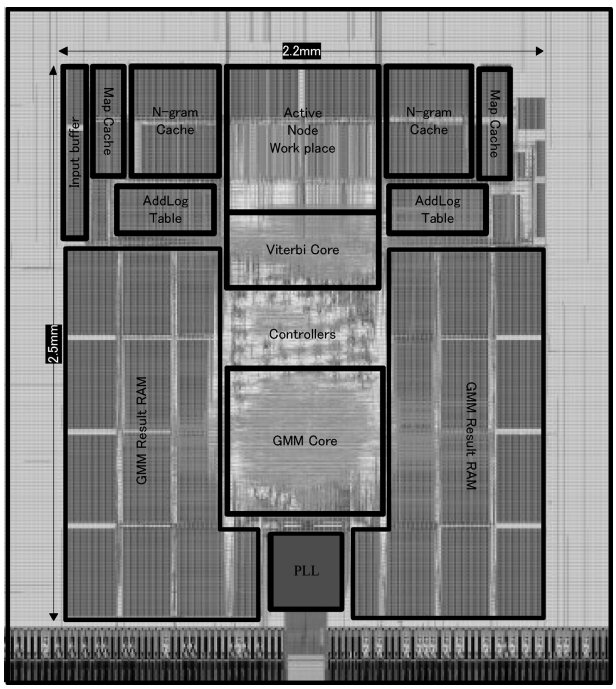


Fig. 23. Chip microphotograph.

3) Hit Rate: We maximize the cache memory size of bi-gram and active node map to 0.73 Mbit, as portrayed in Fig. 22, which can produce a hit rate of 75%.

V. IMPLEMENTATION

As described in the previous sections, we used software prototype profiling with Julius 4.0 and Microsoft Visual C++ to analyze the accuracy trade-off of the parameters and the proposed schemes. We implement the referential hardware using hardware description language (HDL) to check the required memory bandwidth and frequency for real-time operation. After that, a VLSI chip was designed and fabricated. Its performance was evaluated using a SoC logic tester. Table I shows the implementation flow and the CAD tools.

A. Chip Layout

Fig. 23 shows a chip for the proposed SoC fabricated using 40 nm CMOS technology. A summary of the chip statistics is shown in Table II. It occupies 2.2×2.5 mm^2 containing 1.9 M transistors for the Logic and 7.8 Mbit on-chip SRAM. The logic transistors are mainly used by the GMM core because of the highly parallel architecture. The breakdown of the internal memory is shown in Table III. The power was measured when performing real-time 60-kWord continuous speech recognition.

TABLE II
SUMMARY OF CHIP IMPLEMENTATION

Process Technology		40-nm CMOS
Core area		2.2 mm × 2.5 mm
Chip area		5 mm × 5 mm
Transister Count (Logic)	GMM	1.1 M
	Viterbi	0.3 M
	Other	0.5 M
	Total	1.9 M
On-Chip Memory (SRAM)		7.8 Mbit
Supply voltage		1.1V
I/O voltage		3.3V
Evaluation environment		ADVAN SOC Tester System
Operating Frequency		126.5 MHz for 60 kWord real-time processing
Measured Power (without IO)		144 mW
Leakage current		2.28 mA

TABLE III
BREAKDOWN OF INTERNAL MEMORY

Description	# of SRAM	Size of SRAM (Kb)
gmm_result_ram	26	5000
gmm_buffer	1	26
MFCC_buffer	1	8
add_log_table	3	993
active_node_work_space	6	1025
shared Tree DB	1	14
bi-gram	4	384
active_node_map	6	346
initial parameter	1	4
output buffer	1	2
Total	50	7802 (975kB)

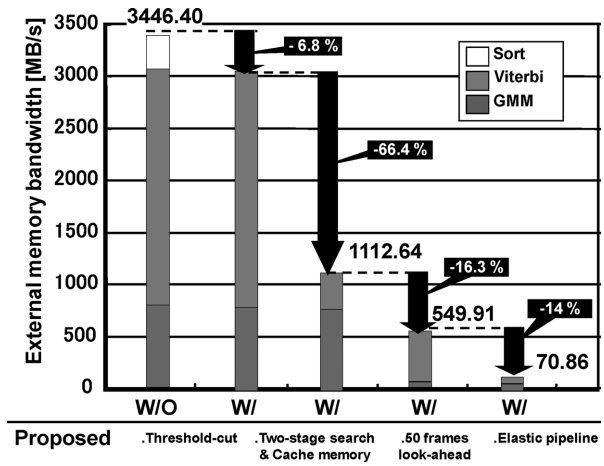


Fig. 24. Bandwidth reduction by the proposed schemes.

The clock gating is implemented in the GMM result RAM and GMM Core.

B. Required Frequency and Memory Bandwidth

Fig. 24 presents the total memory bandwidth reduction when using all the proposed schemes. The dynamic threshold-cut scheme reduces the memory bandwidth for sort processing. The variable-frame look-ahead scheme can reduce memory bandwidth by 16.3% at most, whereas the two-stage language model search and the Viterbi cache can provide reduction of 66.4%, with the default parameters and cache memory size of 0.73 Mbit. The total cycle time is reduced by 78%. This chip can process real-time 60-kWord continuous speech recognition at the frequency of 126.5 MHz.

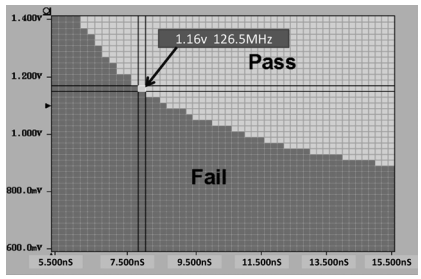


Fig. 25. Vdd versus Period Shmoo plot generated by SOC logic tester.

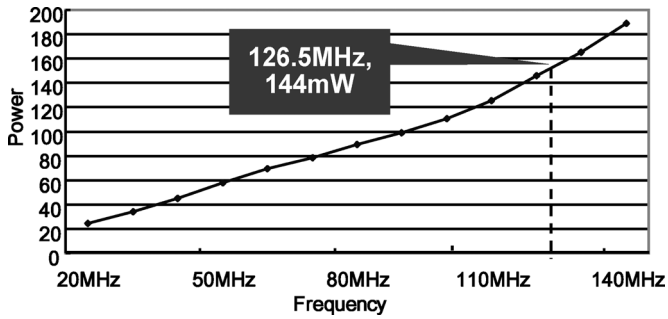


Fig. 26. Power consumption with the lowest operation voltage.

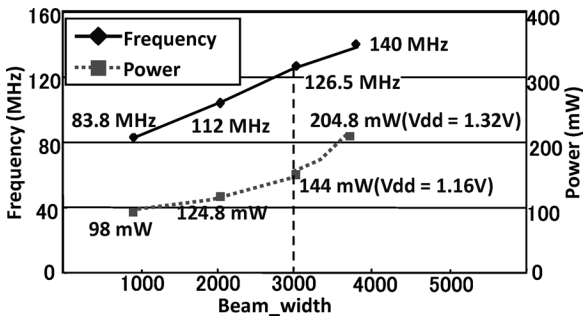


Fig. 27. Measurement results of frequency vs. power consumption vs. beam-width.

C. Power Consumption

Using the shmoo plot [18] presented in Fig. 25, we measured the power consumption with the lowest operation voltage. Figs. 26 and 27 show measured data of power consumption versus operating frequencies versus beam-width. As described in Section III, the larger beam-width can yield higher accuracy, but it will also increase the computational workload. This chip can function at 126.5 MHz for a beam-width of 3000. The power consumption is 144 mW with accuracy of 91.94%. It can function at 140 MHz for a beam-width of 3800: the power consumption is 204.8 mW with accuracy of 91.89%.

We also measured power consumption for I/O and PLL as shown in Table IV, The chip IO consumes 58.2 mW at 31.625-MHz and it grows proportionally with the increase of I/O frequency, it grows much faster than the core power because it works at higher voltage . The IO power consumption was reduced greatly by our proposed schemes because it is mainly caused by external DRAM access.

D. Comparison With Other Works

We compared the performance of our chip with those of other recently reported works (Table V) in terms of the vocabulary size, GMM model, language model, real-time factor, operation frequency, memory bandwidth and the logic element. The real-time factor represents the system speed; for example, a real-time factor of "0.5" corresponds to "$\times$ 2" faster than real-time operation, which means that the recognizer takes half the time of the input speech length to process it. The comparison reveals that our chip achieves the lowest external memory bandwidth, even with the largest vocabulary size of 60-kWord. Few reports of other studies presented the power consumption because most of them are FPGA-based systems. However, the comparison still proved that our chip is not only the first hardware-based recognizer able to process 60-kWord continuous speech recognition in real time; but also achieves the lowest power consumption due to the great reduction of power for IO.

TABLE IV
SUMMARY OF POWER CONSUMPTION

Power type	Frequency	Supply Voltage	Power consumption
Core	126.5 MHz	1.16V	144 mW
IO	31.625 MHz	3.3V	58.2 mW
PLL	126.5 MHz	1.1V	1.98 mW

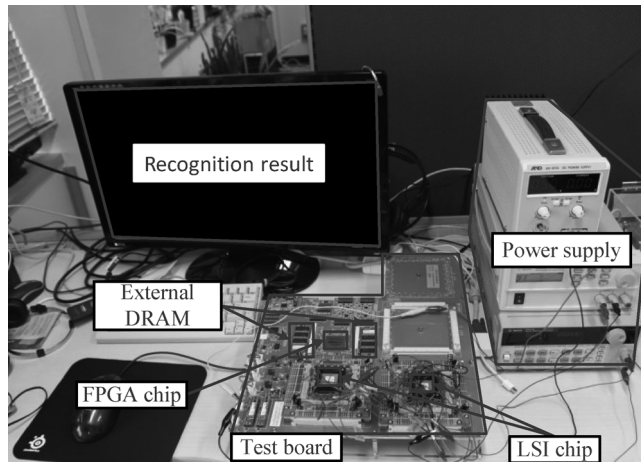


Fig. 28. Demonstration system photograph.

E. Demonstration System

We implement a demo system using a "PowerMedusa" custom test board [19], as presented in Fig. 28. The test board has two external DRAMs, two DUT interfaces for test chips, and an on-board FPGA. The clock block and power supply for the chip use the peripheral equipment. The mel frequency cepstral coefficient (MFCC) feature extraction is implemented by PC, the input speech data can either be recorded as an audio stream or with real-time speaking. First, the DB data are set up to the external DRAM and the speech feature vectors are sent to a buffer in the FPGA. The chip reads the parameters and feature vectors throw the FPGA and start processing. The vectors of look-ahead frames are sent to the chip every time the previous frames are finished.

VI. CONCLUSION

We developed a low-power VLSI chip for 60-kWord real-time continuous speech recognition. We proposed several schemes to reduce the memory bandwidth and the operating

TABLE V
COMPARISON WITH RECENTLY REPORTED WORKS.

		[3]	[4]	[6]	[7]	[8]	[9]	This work
Vocabulary (k)		1	5	5	5	5	20	60
Technology		FPGA	FPGA	FPGA	VLSI (0.18 um)	VLSI (0.18 um)	FPGA	VLSI (40 nm)
GMM Module	# of states	NA	4,147	7,862	3,001	3,001	3,001	2,000
	# of distributions	8	NA	8	16	16	16	16
	# of dimensions	39	NA	39	39	39	39	25
LM	# of unigram	1,000	4,989	NA	5000	5000	19,983	60,001
	# of bigram	2,385	1,639,687	NA	835,000	835,000	1,440,272	4,000,273
Viterbi beam width		NA	NA	NA	NA	NA	500	3000
Real-time factor		2.3	0.1	0.66	0.55	0.42	0.66	1
Internal Frequency (MHz)		50	100	100	133	100	100	126.5
External memory BW (MB/s)		100	950	NA	530	NA	800	70.86
Core area (mm^2)		NA	NA	NA	8.78	15.47	NA	5.5
Power consumption(mW)		NA	NA	NA	NA	NA	NA	144
Logic elements		13,449 slices	52,941 slices	9,576slices	NA	NA	13,835 slices	1.9 MTr. (25,108 slices) *
Internal memory (KB)		138	993	394	60.2	140.1	416	975

* It is difficult to precisely compare the logic elements because even the "slice" has different structure between platforms, however, we synthesized our design by "Quartus" using a FPGA which has similar slice with the "multi-FPGA work". Our design occupy about 25108 slices.

clock frequency. The accuracy degradation of the important parameters in Viterbi computation is discussed. Results show that our implementation achieves 95% bandwidth reduction (70.86 MB/s) and 78% of the required frequency reduction (126.5 MHz) for 60-kWord real-time continuous speech recognition. We fabricated a VLSI test chip in 40 nm CMOS technology and assessed its performance. Results show that the chip described in this paper can perform 60-kWord continuous real-time speech recognition at 126.5 MHz with power consumption of 144 mW and with little accuracy degradation. Future works will specifically examine processing speed improvement and optimization of the latency.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the Associate Editor for their valuable comments and suggestions.

REFERENCES

[1] A. Lee, T. Kawahara, and K. Shikano, "Julius—An open source, real-time, large-vocabulary recognition engine," in *Proc. Eur. Conf. Speech Commun. Tech. (EUROSPEECH)*, Aalborg, Denmark, Sep. 2001, pp. 1691–1694.

[2] K. Yu and R. A. Rutenbar, "Profiling large-vocabulary continuous speech recognition on embedded devices: A hardware resource sensitivity Analysis," in *Proc. 10th Conf. Interspeech*, Brighton, U.K., Sep. 2009, pp. 995–998.

[3] E. C. Lin, Y. Kai, R. A. Rutenbar, and T. Chen, "A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single FPGA," in *Proc. 15th ACM/SIGDA Int. Symp. FPGA*, Monterey, CA, Sep. 2007, pp. 60–68.

[4] E. C. Lin and R. A. Rutenbar, "A multi-FPGA 10 ×-real-time high-speed search engine for a 5000-word vocabulary speech recognizer," in *Proc. 17th ACM/SIGDA Int. Symp. FPGA*, Monterey, CA, Feb. 2009, pp. 83–92.

[5] S. Yoshizawa, N. Wada, N. Hayasaka, and Y. Miyanaga, "Scalable architecture for word HMM-based speech recognition and implementation in complete system," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 1, pp. 70–77, Jan. 2006.

[6] Y. Choi, K. You, and W. Sung, "FPGA-based implementation of a real-time 5000-word continuous speech recognizer," in *Proc. 16th Eur. Signal Process. Conf.*, Lausanne, Switzerland, Aug. 2008.

[7] Y. Choi, K. You, J. Choi, and W. Sung, "VLSI for 5000-word continuous speech recognition," in *Proc. IEEE ICASSP*, Taipei, Taiwan, Apr. 2009, pp. 557–560.

[8] K. You, Y. Choi, J. Choi, and W. Sung, "Memory access optimized VLSI for 5000-word speech recognition," *J. Signal Process. Syst.*, vol. 63, no. 1, pp. 95–105, Apr. 2011.

[9] Y. Choi, K. You, J. Choi, and W. Sung, "A real-time FPGA-based 20,000-word speech recognizer with optimized DRAM access," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 8, pp. 2119–2131, Aug. 2010.

[10] T. Ma and M. Deisher, "Novel ci-backoff scheme for real-time embedded speech recognition," in *Proc. IEEE ICASSP*, Dallas, TX, Mar. 2010, pp. 1614–1617.

[11] H. Noguchi, K. Miura, T. Fujinaga, T. Sugahara, H. Kawaguchi, and M. Yoshimoto, "VLSI architecture of GMM processing and Viterbi decoder for 60,000-word real-time continuous speech recognition," *IEICE Trans. Electron.*, vol. 94, no. 4, pp. 458–467, Apr. 2011.

[12] G. He, T. Sugahara, T. Fujinage, Y. Miyamoto, H. Noguchi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A 40 nm 144 mW VLSI processor for realtime 60 kword continuous speech recognition," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Sep. 2011, pp. 1–4.

[13] X. Huang, A. Acero, and H. W. Hon, *Spoken Language Processing—A Guide to Theory, Algorithm, and System Development*. Englewood Cliffs, NJ: Prentice-Hall, 2001.

[14] B. Pellom, R. Sarikaya, and J. Hansen, "Fast likelihood computation techniques in nearest-neighbor based search for continuous speech recognition," *IEEE Signal Process. Lett.*, vol. 8, no. 8, pp. 221–224, Aug. 2001.

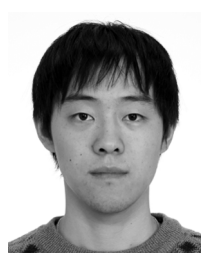
[15] F. Seide, "Fast likelihood computation for continuous-mixture densities using a tree-based nearest neighbor search," in *Proc. Eur. Conf. Speech Commun. Tech. (EUROSPEECH)*, Madrid, Spain, Sep. 1995, pp. 1079–1083.

[16] H. Ney and S. Ortmanns, "Dynamic programming search for continuous speech recognition," *IEEE Signal Process. Mag.*, vol. 16, no. 5, pp. 64–83, Sep. 1999.

[17] B. H. Jeong *et al.*, "A 1.35 V 4.3 Gb/s 1 Gb LPDDR2 DRAM with controllable repeater and on-the-fly power-cut scheme for low-power and high-speed mobile application," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2009, pp. 132–133.

[18] K. Baker, E. Philips, and T. Eindhoven, "Shmoo plotting: The black art of IC testing," in *Proc. IEEE Int. Test Conf.*, Washington, DC, Oct. 1996, pp. 932–933.

[19] PowerMedusa Custom Test Board. Amagasaki, Japan, MMS [Online]. Available: http://www.mms.co.jp/powermedusa/concept/index.html



Guangji He received the B.Eng. degree in electronic engineering from the Beijing Institute of Technology, Beijing, China, in 2007 and the M.Eng. degree in computer science and systems engineering from Kobe University, Hyogo, Japan, in 2010, where he is currently pursuing the Ph.D. degree.

His current research interests include speech recognition, speech signal processing, and low-power multimedia VLSI design.

Takanobu Sugahara received the B.Eng. degree in computer and systems engineering from Kobe University, Hyogo, Japan, in 2010 where he is currently working toward the M.E. degree.

His current research interest is speech recognition algorithms and their hardware implementation for wearable devices.

Yuki Miyamoto received the B.Eng. degree in computer and systems engineering from Kobe University, Hyogo, Japan, in 2011. Currently, he is working toward the M.S. degree at Kobe University.

Since 2009, he has been involved in the research and development of low-power speech recognition designs.

Tsuyoshi Fujinaga received the B.Eng. and M.Eng. degrees in computer and systems engineering from Kobe University, Hyogo, Japan, in 2009 and 2011, respectively.

Since 2009, he has been involved in the research and development of low-power speech recognition designs.

Hiroki Noguchi (M'11) received the B.Eng. and M.Eng. degrees in computer and systems engineering and the Ph.D. degree in engineering from Kobe University, Hyogo, Japan, in 2006, 2008, and 2011, respectively.

His research interests are low-power SRAM designs, multimedia/ ubiquitous systems, and digital signal processing architectures, which include speech-recognition for handheld devices, image-recognition for wearable computing, and mixed integer programming for real-time robotics controlling, and their low-power hardware implementation.

Shintaro Izumi (M'12) received the B.Eng. and M.Eng. degrees in computer science and systems engineering and the Ph.D. degree in engineering from Kobe University, Hyogo, Japan, in 2007, 2008, and 2011 respectively.

He was a JSPS Research Fellow at Kobe University during 2009–2011. Since 2011, he has been an Assistant Professor of the Organization of Advanced Science and Technology at Kobe University. His current research interests include biomedical signal processing, communication protocols, low-power VLSI design, and sensor networks.

Dr. Izumi is a Member of IEICE and IPSJ.

Hiroshi Kawaguchi (M'98) received the B.Eng. and M.Eng. degrees in electronic engineering from Chiba University, Chiba, Japan, respectively, in 1991 and 1993, respectively, and the Ph.D. degree in engineering from the University of Tokyo, Tokyo, Japan, in 2006.

He joined Konami Corporation, Kobe, Japan, in 1993, where he developed arcade entertainment systems. He moved to the Institute of Industrial Science, University of Tokyo, as a Technical Associate in 1996, and was appointed as a Research Associate in 2003. In 2005, he moved to Kobe University, Kobe. Since 2007, he has been an Associate Professor with the Department of Information Science at that university. He is also a Collaborative Researcher with the Institute of Industrial Science, University of Tokyo. His current research interests include low-voltage SRAM, RF circuits, and ubiquitous sensor networks.

Dr. Kawaguchi is a Member of ACM, IEICE, and IPSJ. He was a recipient of the IEEE ISSCC 2004 Takuo Sugano Outstanding Paper Award and the IEEE Kansai Section 2006 Gold Award. He has served as a Design and Implementation of Signal Processing Systems (DISPS) Technical Committee Member for IEEE Signal Processing Society, as a Program Committee Member for IEEE Custom Integrated Circuits Conference (CICC) and IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips), and as a Guest Associate Editor of IEICE *Transactions on Fundamentals of Electronics, Communications and Computer Sciences* and IPSJ *Transactions on System LSI Design Methodology* (TSLDM).

Masahiko Yoshimoto (M'05) received the B.S. degree in electronic engineering from Nagoya Institute of Technology, Nagoya, Japan, in 1975, and the M.S. degree in electronic engineering and the Ph.D. degree in Electrical engineering from Nagoya University, Nagoya, in 1977 and 1998, respectively.

He joined the LSI Laboratory, Mitsubishi Electric Corp., Itami, Japan, in April 1977. During 1978–1983 he was engaged in the design of NMOS and CMOS static RAM including a 64 K full CMOS RAM with the world's first divided-wordline structure. From 1984, he was involved in research and development of multimedia ULSI systems for digital broadcasting and digital communication systems based on MPEG2 and MPEG4 Codec LSI core technology. Since 2000, he has been a Professor of the Department of Electrical and Electronic Systems Engineering at Kanazawa University, Japan. Since 2004, he has been a Professor of the Department of Computer and Systems Engineering at Kobe University, Japan. His current activity is focused on the research and development of an ultra low power multimedia and ubiquitous media VLSI systems and a dependable SRAM circuit. He holds 70 registered patents.

Dr. Yoshimoto has served on the program committee of the IEEE International Solid State Circuit Conference from 1991 to 1993. Also he served as Guest Editor for special issues on Low-Power System LSI ,IP and Related Technologies of IEICE Transactions in 2004. He was a chair of IEEE SSCS (Solid State Circuits Society) Kansai Chapter from 2009 to 2010. He is also a chair of the IEICE Electronics Society Technical Committee on Integrated Circuits and Devices from 2011–2012. He received the R&D100 awards from the R&D magazine for the development of the DISP and the development of the realtime MPEG2 video encoder chipset in 1990 and 1996, respectively. He also received 21th TELECOM System Technology Award in 2006.