A Study on VLSI Design for Real-Time Large Vocabulary Continuous Speech Recognition

He, Guangji

(Degree)
博士（工学）

(Date of Degree)
2014-03-25

(Date of Publication)
2015-03-01

(Resource Type)
doctoral thesis

(Report Number)
甲第6102号

(URL)
https://hdl.handle.net/20.500.14094/D1006102

Doctoral Dissertation

A Study on VLSI Design for Real-Time Large Vocabulary Continuous Speech Recognition

（実時間大語彙連続音声認識プロセッサ
VLSI 設計技術に関する研究）

January 2014

Graduate School of System Informatics
Kobe University

He Guangji

Abstract

Speech recognition has been used recently in various applications such as automatic transcript, website indexing, navigation, mobile systems, ubiquitous systems, and robotics as a human interface. Large vocabulary continuous speech recognition (LVCSR) with acoustic and language models is too resource-hungry and power-sensitive for software applications. Hardware implementation by very large scale integrated circuit (VLSI) or field programmable gate array (FPGA) is demanded especially for use in mobile equipment and intelligent robots because of advantageous high processing speed and low power consumption.

This dissertation reports VLSI designs for large vocabulary real-time continuous speech recognition based on context-dependent Hidden Markov Model (HMM). As the background of this research area, the objective of this study and an overview of this dissertation are presented in Chapter 1. Then issues related to the VLSI implementation for real-time continuous speech recognition are noted in Chapter 2. The main issues are explained as four parts: 1) large-vocabulary are needed to support a higher word-cover rate. 2) low-power is required for longer working time and lower temperature, 3) high-speed is demanded for a more comfortable human interface, and 4) small area which means less recourse would be cost. Algorithm optimization and specialized architectures are proposed to solve these problems.

In Chapter 3, some algorithm optimizations are presented: beam pruning using a dynamic threshold to cut the workspace and memory bandwidth for sort processing; the modified unigram language model which eliminated the updata process for word-internal transitions; two-stage language model searching to reduce cross-word transitions which reduce the memory access greatly. The accuracy degradation of the important parameters in Viterbi computation is strictly discussed. These optimizations are utilized in all the three chips introduced in the following chapters.

Chapter 4 describes the first chip (HMM1) we implemented for 60-kWord real-time continuous speech recognition. It includes a cache architecture using locality of speech recognition, as some of the data that has been used in the current frame may be reused in the following frames, on chip caches are introduced for keeping some of them to reduce the external memory access, the specialized cache architecture achieve a hit-rate

of 75%; a highly parallel Gaussian Mixture Model (GMM) architecture based on the mixture level; a variable-50-frame look-ahead scheme; and elastic pipeline operation between the Viterbi transition and GMM processing. 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.

Chapter 5 describes the second chip (HMM2) we designed for 60-kWord real-time continuous speech recognition. It features a compression–decoding scheme to reduce the external memory bandwidth for Gaussian Mixture Model (GMM) computation and a 4-path Viterbi transition units. We optimize the internal SRAM size using the max-approximation GMM calculation and adjusting the number of look-ahead frames. The test chip, fabricated in 40 nm CMOS technology, occupies 1.77 mm × 2.18 mm containing 2.52 M transistors for logic and 4.29 Mbit on-chip memory. The measured results show that our implementation achieves 34.2% required frequency reduction (83.3 MHz), 48.5% power consumption reduction (74.14 mW) for 60 k-Word real-time continuous speech recognition compared to the previous work while 30% of the area is saved with recognition accuracy of 90.9%. It can maximally process 2.4× faster than real-time at 200 MHz and 1.1 V with power consumption of 168 mW.

Chapter 6 describes the third chip (HMM3) we designed for 60-kWord real-time continuous speech recognition. It includes a 8-path Viterbi transition architecture to maximize the processing speed by cutting the on-chip latency and adopts tri-gram language model to improve the recognition accuracy. A two-level cache architecture is implemented for the demo system to cut the off-chip latency. Measured results show that our implementation achieves 25% required frequency reduction (62.5 MHz) and 26% power consumption reduction (54.8 mW) for 60 k-Word real-time continuous speech recognition compared to the previous work. The chip can maximally process 3.02× and 2.25× times faster than real-time at 200 MHz using the bigram and trigram language models, respectively.

Finally, the conclusion of this study is presented in Chapter 7. This thesis presents the VLSI designs for low-power speak-independent large vocabulary real-time continuous

speech recognition based on context-dependent Hidden Markov Model (HMM). The work contributes to achieve a comfortable human interface for mobile systems and robotics.

Keywords: large vocabulary continuous speech recognition (LVCSR), Hidden Markov Model (HMM), speak-independent, 40nm VLSI, Low power, high speed.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background of Research Area

Speech recognition based on Hidden Markov Model (HMM) can provide high recognition accuracy, thus has been used recently in various applications (Fig. 1.1) such as automatic transcript, website indexing, navigation, especially in mobile systems, ubiquitous systems, and robotics as a human interface. High-end personal computers can accommodate speech recognition task well even with large acoustic and language models [1]. However, such methods are not applicable for mobile systems while considering the physical size and power consumption [2]. Additionally, they are unsuitable for next-generation applications such as “audio mining”, which request the recognizer to deliver results at rates that are 10×, 100×, or 1000× faster than real-time [3, 4]. Hardware implementation by very large scale integrated circuit (VLSI) or an field programmable gate array (FPGA) is a good approach to satisfy these demands because of its good processing speed and power consumption.

There have been several hardware-based speech recognizers. Some of them only adopted hardware accelerator or a coprocessor for the GMM computation part [5, 6, 7], however, as the vocabulary size increases, the portion of the GMM probability computation becomes smaller while the Viterbi processing becomes larger which need to be covered. Some complete hardware implementation for both Viterbi and GMM processing are also presented. Lin et al. reported a Multi-FPGA implementation for 5 k-word continuous speech recognition [8] that achieves 10× faster than real time, but the system is not extendable for larger vocabularies because it is not cost-effective. It needs two FPGAs and two DDR2 DRAMs each with a 64-bit wide data-path. Yoshizawa et al. proposed a scalable architecture for speech recognition [9]. Their chip can have an adjustment between vocabulary size and processing speed, but the system only offers real-time performance with a limited vocabulary of 800 words. Choi et al. developed FPGA and VLSI implementations for 20 k-word speech recognition [10, 11, 12]. They implemented a special memory interface for several parts of the recognition engine to apply optimized DRAM access, which improves the data transfer efficiency, but the numerous external DRAM accesses cause high IO frequency, which requires a

high supply voltage and high power consumption in both the FPGA side and DRAM side. In Image and Vidio processing system, the DRAM only acts as a buffer between camera and chip, the pixels are read from DRAM orderly and saved to the on-chip memory. However, in large vocabulary continuous speech recognition (LVCSR) system, the DRAM functions as a data-base which saves the dictionary parameters and language models. These data will be accessed randomly during the processing. According to the characteristics of DRAM , there are several cycles of latency caused by pre-charge every time before we read from the DRAM, therefore if the required data are not saved sequentially, the access-efficiency is bad. As a result, speech recognition needs much higher IO frequency than video processing to get the same amount of data from DRAM. Especially, with the number of vocabulary increase, the external memory bandwidth become enormous which causes two problems, firstly, real-time processing is impossible to be achieved because of the I/O frequency limitation. Secondly, large amount of power is consumed by I/O because of the high supply voltage (3.3V). Consequently, reducing external memory access and optimizing the latency is the most important things to implement a low-power large-vocabulary real-time continuous speech recognition system.

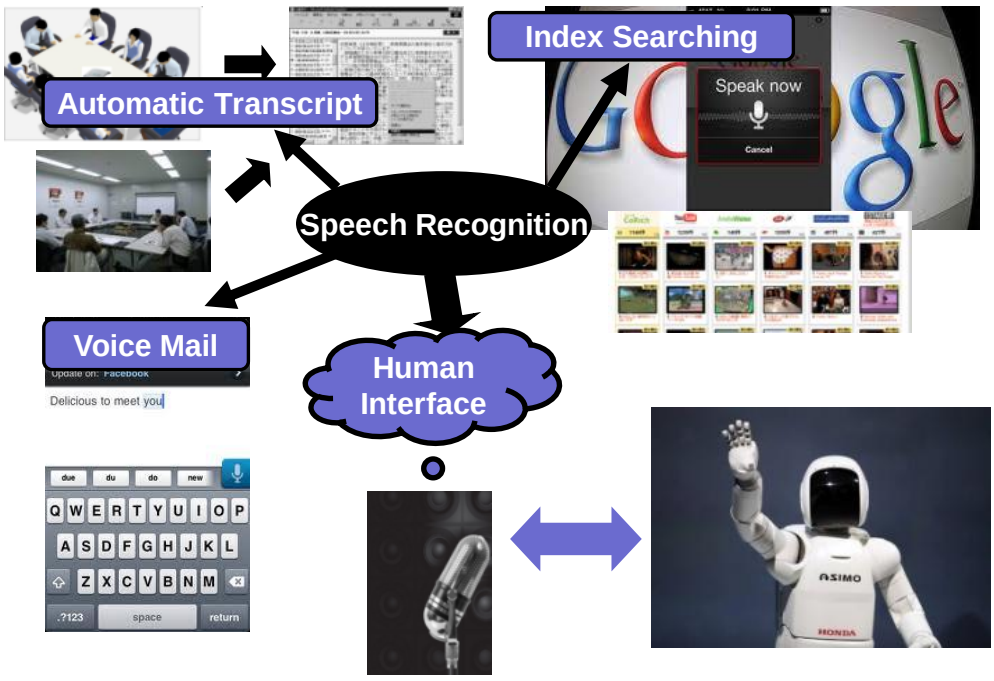

Fig. 1.1 Multifarious applications using speech recognition.

1.2 Objective of This Study

This dissertation focuses on VLSI design for low-power, large-vocabulary, real-time, speak-independent continuous speech recognition based on context-dependent Hidden Markov Model (HMM). The proposed algorithm optimizations and specialized architectures are described in this dissertation.

1.3 Overview of This Dissertation

An overview of this dissertation is presented in Fig. 1.2 with clear correlation between the issues and solutions. First, the background and the objective of this study are decribed in Chapter 1. Then issues related to the VLSI implementation for real-time continuous speech recognition are noted in Chapter 2. The main issues are explained as four parts: 1) large-vocabulary are needed to support a higher word-cover rate. 2) low-power is required for longer working time and lower temperature, 3) high-speed is demanded for a more comfortable human interface, and 4) small area is also expected which means less recourse would be cost. Algorithm optimization and specialized architectures are proposed to solve these problems.

In Chapter 3, some algorithm optimizations are presented: beam pruning using a dynamic threshold to cut the workspace and memory bandwidth for sort processing; the modified unigram language model which eliminated the updata process for word-internal transitions; two-stage language model searching to reduce cross-word transitions which reduce the memory access greatly. The accuracy degradation of the important parameters in Viterbi computation is strictly discussed. These optimizations are utilized in all the three chips introduced in the following chapters.

Chapter 4 describes the first chip (HMM1) we implemented for 60-kWord real-time continuous speech recognition. It includes a cache architecture using locality of speech recognition, as some of the data that has been used in the current frame may be reused in the following frames, on chip caches are introduced for keeping some of them to reduce the external memory access, the specialized cache architecture achieve a hit-rate of 75%; a highly parallel Gaussian Mixture Model (GMM) architecture based on the mixture level; a variable-50-frame look-ahead scheme; and elastic pipeline operation between the Viterbi transition and GMM processing. 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.

Chapter 5 describes the second chip (HMM2) we designed for 60-kWord real-time continuous speech recognition. It features a compression–decoding scheme to reduce the external memory bandwidth for Gaussian Mixture Model (GMM) computation and a 4-path Viterbi transition units. We optimize the internal SRAM size using the max-approximation GMM calculation and adjusting the number of look-ahead frames. The test chip, fabricated in 40 nm CMOS technology, occupies 1.77 mm × 2.18 mm containing 2.52 M transistors for logic and 4.29 Mbit on-chip memory. The measured results show that our implementation achieves 34.2% required frequency reduction (83.3 MHz), 48.5% power consumption reduction (74.14 mW) for 60 k-Word real-time continuous speech recognition compared to the previous work while 30% of the area is saved with recognition accuracy of 90.9%. It can maximally process 2.4× faster than real-time at 200 MHz and 1.1 V with power consumption of 168 mW.

Chapter 6 describes the third chip (HMM3) we designed for 60-kWord real-time continuous speech recognition. It includes a 8-path Viterbi transition architecture to maximize the processing speed by cutting the on-chip latency and adopts tri-gram language model to improve the recognition accuracy. A two-level cache architecture is implemented for the demo system to cut the off-chip latency. Measured results show that our implementation achieves 25% required frequency reduction (62.5 MHz) and 26% power consumption reduction (54.8 mW) for 60 k-Word real-time continuous speech recognition compared to the previous work. The chip can maximally process 3.02× and 2.25× times faster than real-time at 200 MHz using the bigram and trigram language models, respectively.

The conclusions of this study are presented in Chapter 7. The overall contributions are summarized briefly.

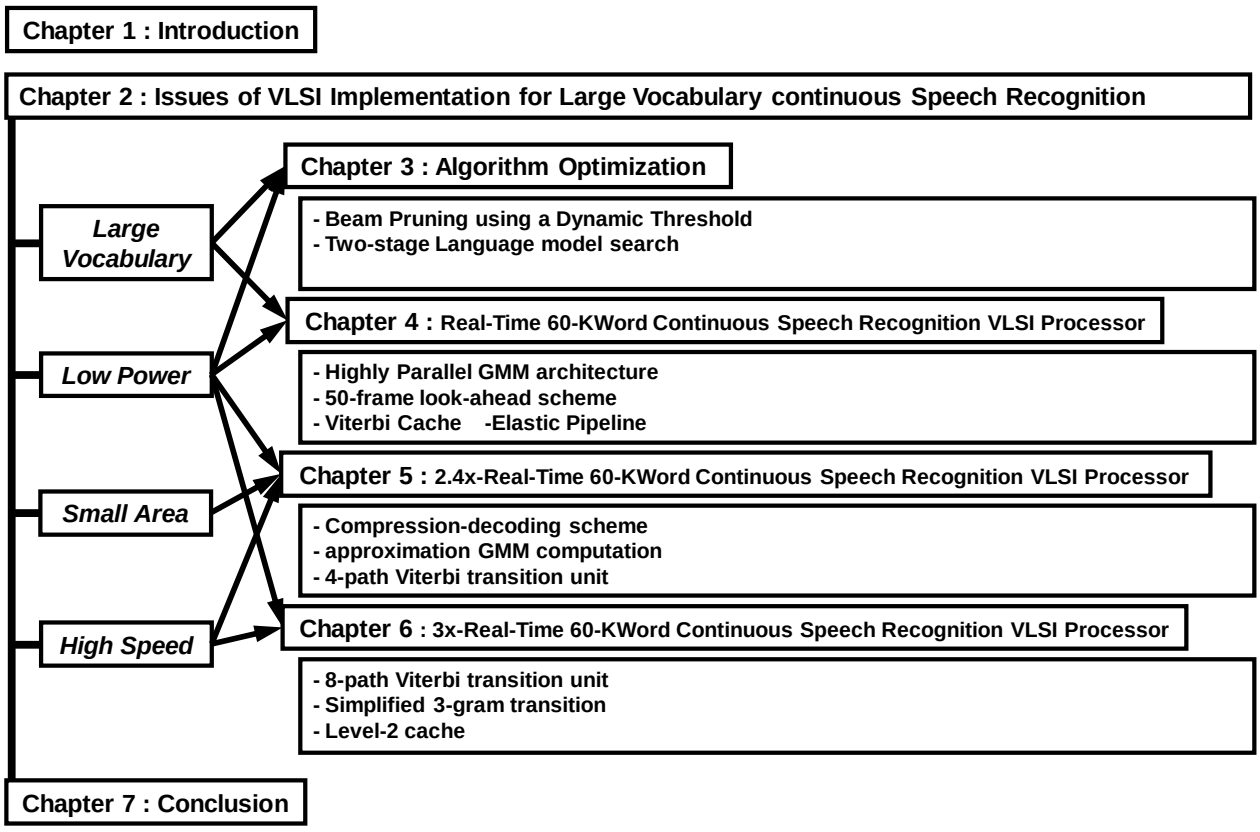

Fig. 1.2 Overview of this dissertation.

Chapter 2 Issues of VLSI Implementation for Large Vocabulary Continuous Speech Recognition

In this chapter, issues of VLSI Implementation for large Vocabulary Continuous Speech Recognition are summarized in this chapter.

The four issues are explained in detail: 1) large-vocabulary are needed to support a higher word-cover rate. 2) low-power is required for longer working time and lower temperature, 3) high-speed is demanded for a more comfortable human interface, and 4) small area is also expected which means less recourse would be cost.

2.1 Issue of Large Vocabulary

As a Recognizer can't recognize words that are not included in the database, large-vocabulary are needed for many applications to support a sufficient word-cover rate. Figure 2.1 shows the # of vocabulary versus word-cover rate reported by Japanese Language researcher center . More than 20,000 words are needed for TV speech and TV script while more than 40,000 words are needed for daily life conversation to supply a nearly 100% of word-cover rate. However, The exist hardware recognizers [4, 5, 6, 7, 8, 9, 10, 11, 12] are only available for small or medium vocabulary of 1,000-20,000 words. This is because with the increase of vocabulary size, the required frequency and external memory bandwidth for real-time processing grow exponentially as shown in Fig. 2.2 and Fig. 2.3. Consequently, we must reduce both of them to be able to handle a large vocabulary of 60,000 words.

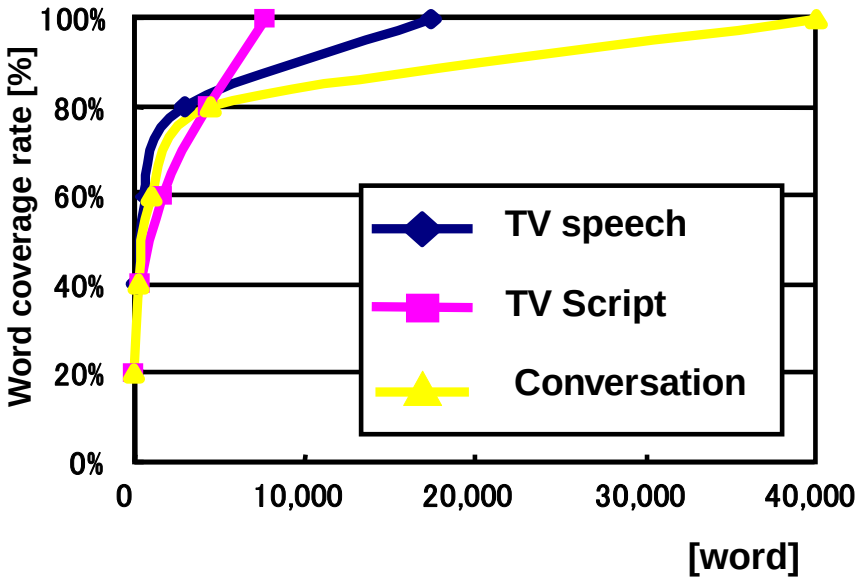

Fig. 2.1 # of vocabulary versus word-cover rate for TV speech, TV Script, and Daily Conversation.

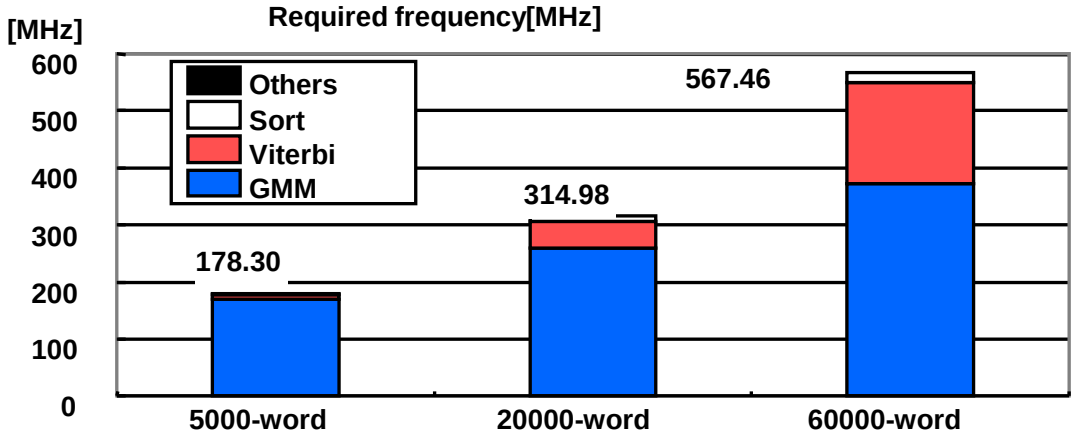

Fig. 2.2 Required frequency for real-time continuous speech recognition.

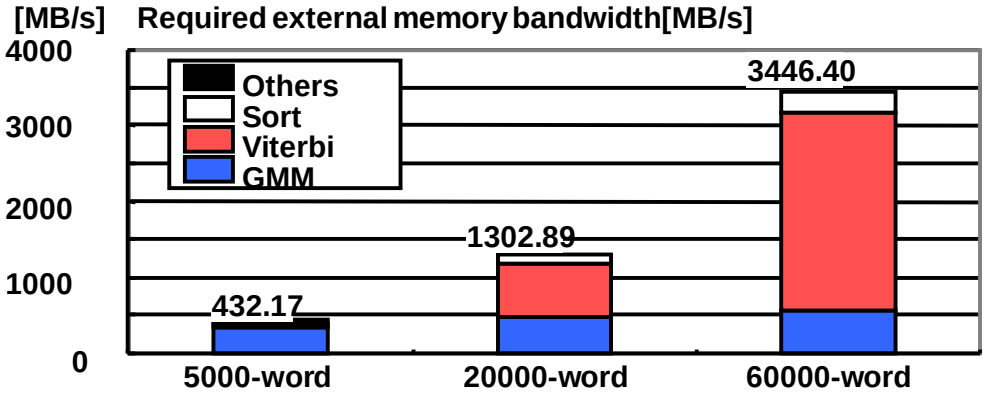

Fig. 2.3 Required external memory bandwidth for real-time continuous speech recognition.

2.2 Issue of Low Power

Power consumption is extremely important when we considering the work-time and temperature of the device for mobile application. It won't be comfortable if the device goes out of power or gets very hot in a short period. Consequently, we need to achieve the speech recognizer with as lower power consumption as possible. However, as shown in Fig. 2.2 and Fig. 2.3, with a large vocabulary of 60,000, the required memory bandwidth and the operation frequency of the hardware referential Julius [1] prototype 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]. Also, the high frequency inside the chip causes much core power. Therefore, we must reduce both the I/O power and core power to achieve a low-power speech recognizer.

2.3 Issue of High Speed

Processing-speed is also a key issue for speech recognition. Real-time processing is the least requirement which means the recognizer can deliver results at the same speed as we speak. In fact, times of real-time processing are also needed when we considering the preprocessing time for adopting noise-cancelling or speaker adaptation, and some extra time for dealing with the recognizer results like translation for example. Some next generation applications like “Audio mining” which need to recognize hours of speech in a short time. However, the “on-chip” latency (stop until the data come to the compute unit) and the “off-chip” latency (loading latency from DRAM to IO) which will be explained later delay the processing speed. Therefore, we need to reduce both of them to achieve a high-speed recognizer.

2.4 Issue of Small Area

Small-area is also expected which means less recourse would be cost. Some methods are available for high-performance speech recognizer but cost too much resource including logic element and on-chip memory [8]. For example, if we introduce all the language model data into the internal memory, the external memory bandwidth for Viterbi transition would be reduced greatly; also if we use a highly parallel architecture for all parts of computation, good performance can be achieved, however, a huge area would be cost which is not applicable and acceptable. Therefore, we must consider the efficiency of the proposed schemes to implement the VLSI chip with small-area.

2.5 Summary

This section summarizes the four issues of VLSI Implementation for large Vocabulary Continuous Speech Recognition.

- Large-Vocabulary
  the required frequency and external memory bandwidth for real-time processing must be reduced to be able to handle a vocabulary of 60,000 words.
- Low-Power
  we must reduce both the I/O power and core power to achieve a low-power

speech recognizer.

- High-Speed

  we need to reduce both the “on-chip latency” and “off-chip latency” to achieve a high-speed recognizer.

- Small-Area

  we must consider the efficiency of the proposed schemes to implement the VLSI chip with small-area.

In this dissertation, novel techniques are presented in Chapters 3 to 6 to address the issues.

Chapter 3 Algorithm Optimization

This chapter describes the algorithm optimizations to support a large-vocabulary recognizer. First, the speech recognition algorithm, as embodied in the Julius [1] software recognizer, which we adopt as our reference model is described in Section 3.1. then the beam pruning using a dynamic threshold is introduced in Section 3.2. Modified unigram language model are explained in Section 3.3. Section 3.4 presents the two-stage language model search scheme and accuracy trade-off of the important parameters are described in Section 3.5. Finally, the concluding remarks are made in Section 3.6.

3.1 Speech Recognition Overiew

Fig. 2 presents the speech recognition flow with the HMM algorithm [19]. 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 10ms. **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.

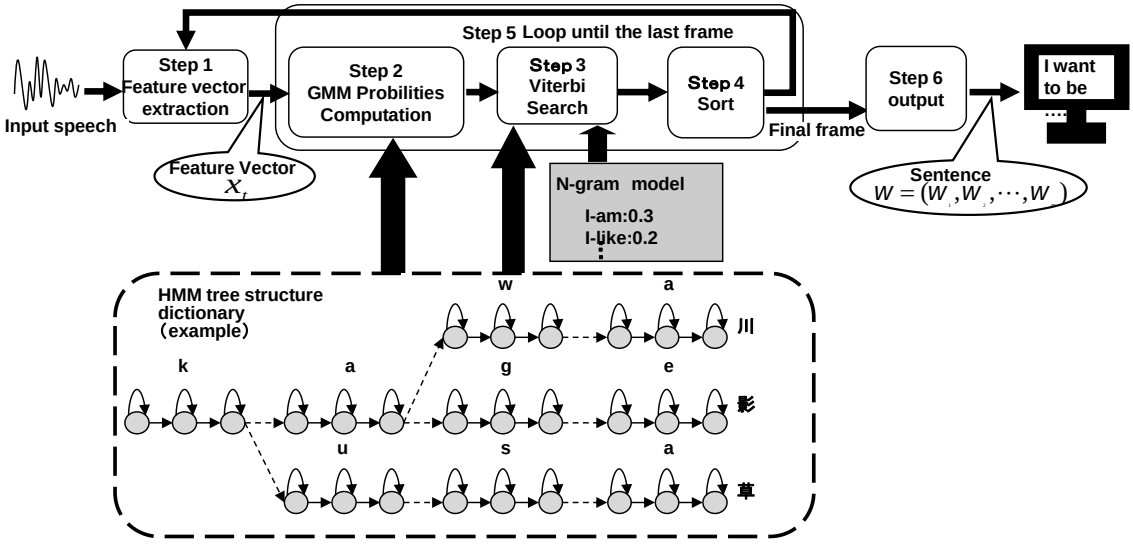

Fig. 3.1 Speech recognition flow with HMM algorithm.

3.1.1 MFCC Feature Extraction

The feature extraction converts a frame of speech input into a vector that represents frequency-domain characteristics. While there are several methods to extract the acoustic data, the most popular form, mel-frequency cepstral coefficients (MFCC), will be explained here. Its operations are based on the physiology of the human ear and relatively straightforward to implement as shown in Fig. 3.2. To generate the mfcc values per frame, a pre-emphasis filter is first used to boost the energy in the high frequencies which contain important acoustic information. A Hamming window and 512-point FFT are then applied, and the power content of the frequencies, or spectrum, is computed. A triangular filter bank designed to extract mel-frequencies is applied to simulate the spectral resolution of the ear, and an inverse discrete cosine transform converts the data back to the time domain and into the mfcc values. To generate the acoustic feature vector, the mfcc values undergo cepstral mean normalization, which average the mfcc values over a certain time period. This helps reduce the effect of background noise on the mfcc values, and the time period varies from the entire speech sample (batch mode) to a subset of the preceding frames (live mode). Since human hearing is also related to the change velocity and acceleration of the mfcc values, the first and second derivative of the normalized mfcc values are taken to generate the acoustic feature vector.

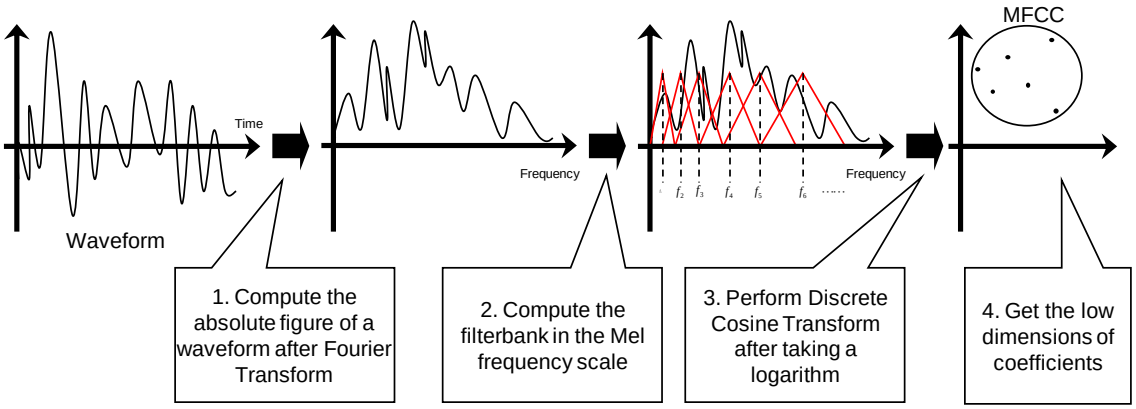

Fig. 3.2 Calculation flow to obtain MFCC from speech waveform

3.1.2 GMM Computation

The left-right HMM, which is a kind of the HMM algorithm, is depicted in Fig. 3.3. To achieve speaker-independent decoding, variations in pronunciation (different voices, accents, genders, etc.) must be accommodated, therefore the weighted sum of Gaussian probability density functions called Gaussian mixture models (GMMs) are used to represent the state output probability of HMMs. 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 (3.1),

$$\log b_j(x_t) = \log \sum_{k=1}^{mix} \lambda_i N(x_t, \mu_k, \sigma_k) \tag{3.1}$$

$$= \log \left[\sum_{k=1}^{mix} \lambda_k \left[\frac{1}{\prod_{i=1}^{p} \sqrt{2\pi\sigma_{ki}^2}} \exp\left\{ -\sum_{i=1}^{p} \frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right\} \right] \right]$$

$$= \log \left[\sum_{k=1}^{mix} \exp\left\{ C_k - \sum_{i=1}^{p} \frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right\} \right]$$

$$= add \log\left[C_k - \sum_{i=1}^{p} \frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right] \tag{3.2}$$

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

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. (3.2) 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.

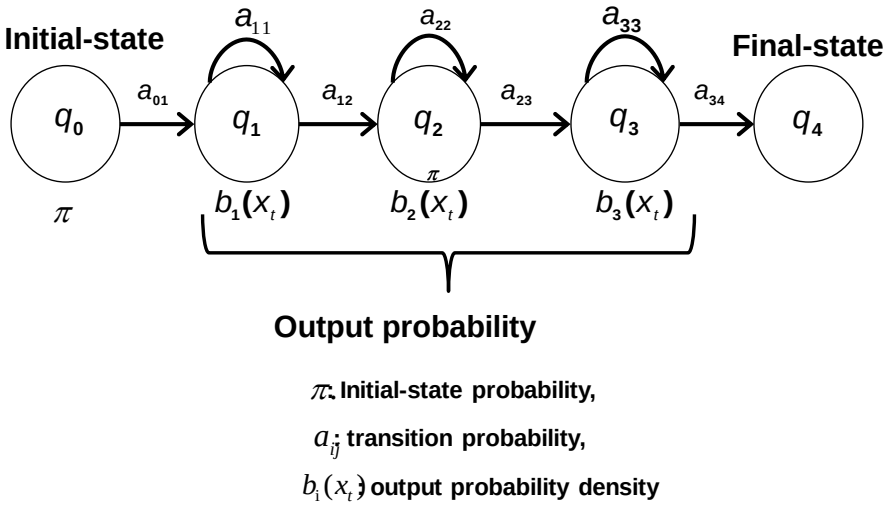

Fig. 3.3 Left-right HMM.

3.1.3 Time-Synchronous Viterbi Beam Search

The Viterbi computation is depicted in Fig. 3.4. 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 (3.3).

$$\delta_t(s_j; w) = \max_{i=j-1,\ j}[\delta_{t-1}(s_i; w) + \log a_{ij}] + \log b_j(x_t) \tag{3.3}$$

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 (3.4).

$$\delta_t(s_0; w) = \max_{v}\{\delta_{t-1}(s_f; v) + \log[\, p(w \mid v)]\} \tag{3.4}$$

Therein, $p(w \mid v)$ 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.

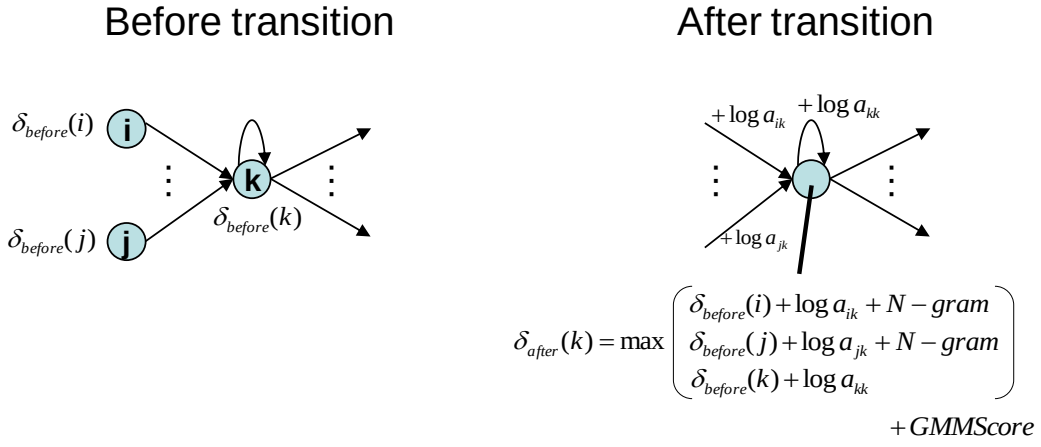

Fig. 3.4 Viterbi computation.

3.2 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 [4] [8], 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. 3.4 shows the beam width variation with a dynamic threshold-cut scheme when the target width is set to 1,500. 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.

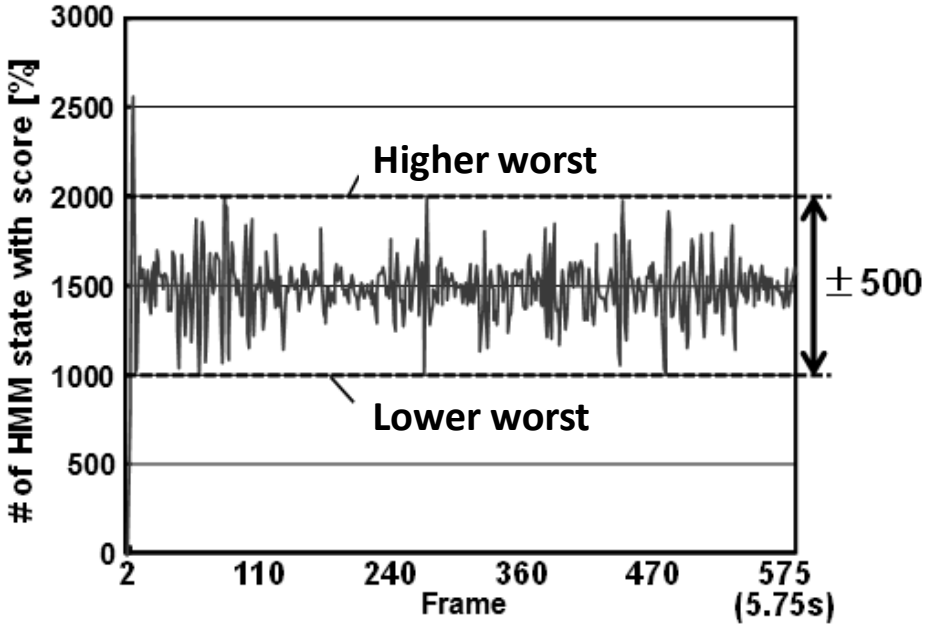

Fig. 3.5 Beam width variation with dynamic threshold-cut scheme.

3.3 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.

3.4 Two-stage Language Model Search

Figure 3.6 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 work-load 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. 3.7, 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 tradeoff of these parameters described above must be discussed carefully.

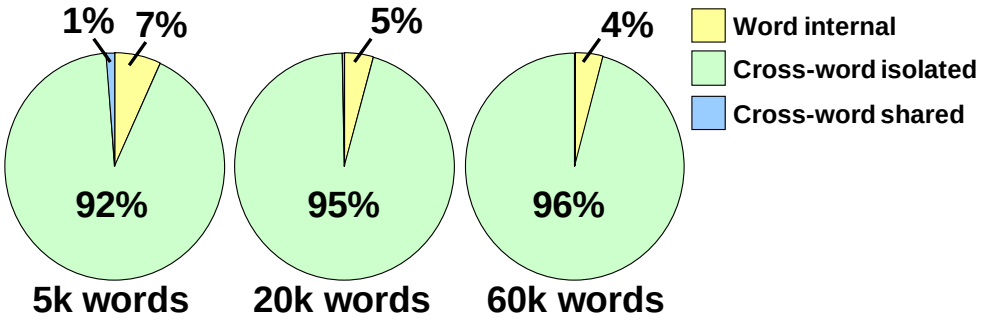

Fig. 3.6 Appearance ratio of three types transitions in Viterbi search.

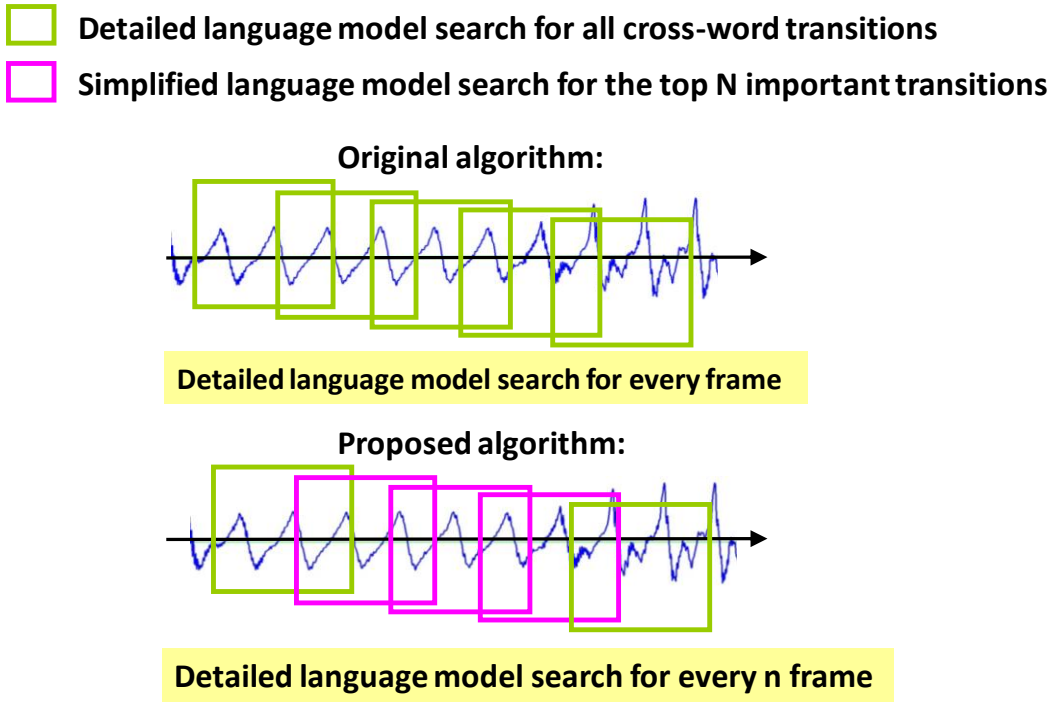

Fig. 3.7 Two-stage language model search.

3.5 Accuracy Tradeoff

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.

3.5.1 Detailed search cycle, the number of cross-word transitions (in simplified search)

First, the tradeoff 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.

Figure 3.8 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. 3.9, 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."

Figure 3.10 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.

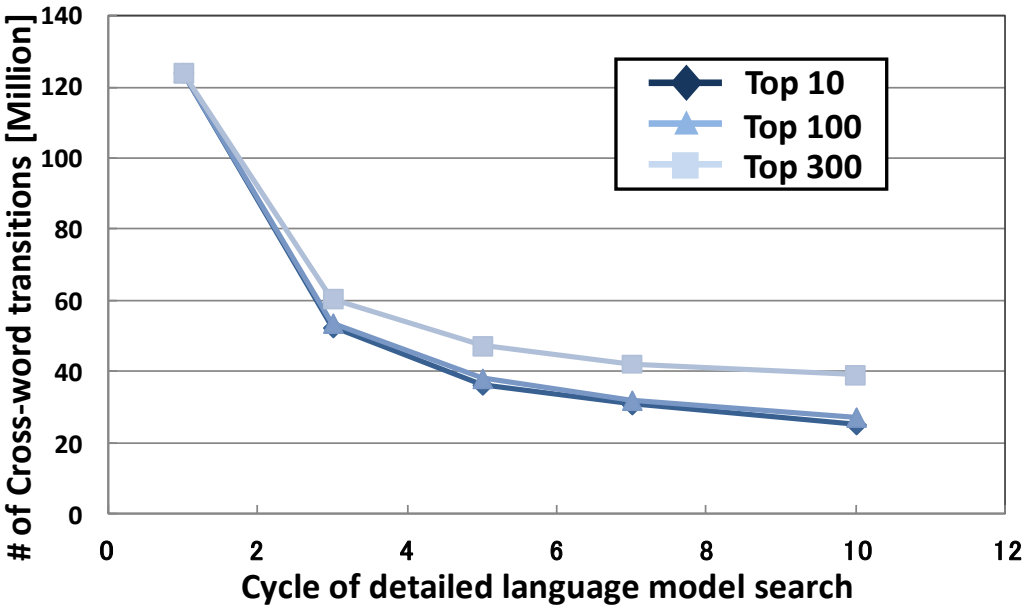

Fig. 3.8 Cycle of detailed language model search versus the number of cross-word transitions.

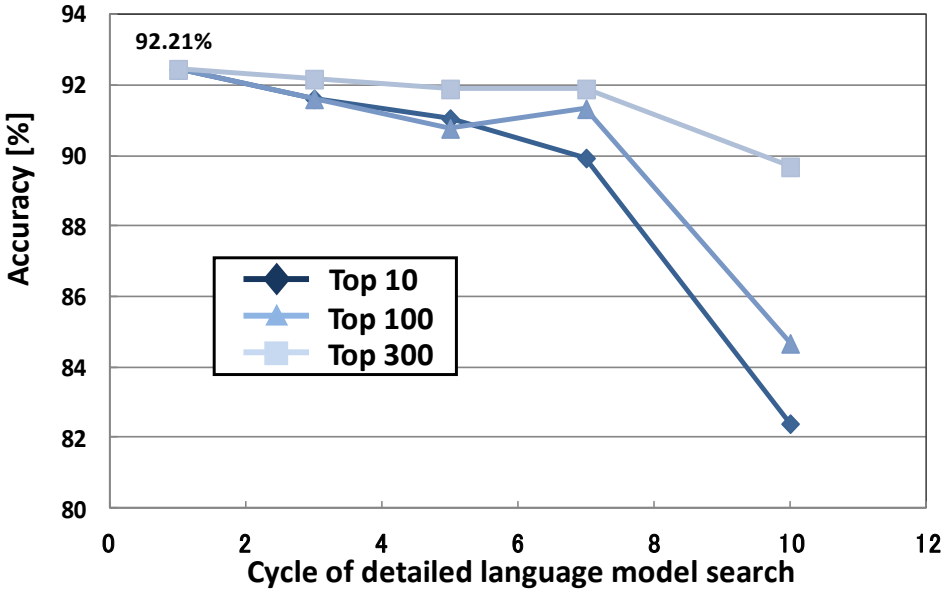

Fig. 3.9 Cycle of detailed language model search versus recognition accuracy.

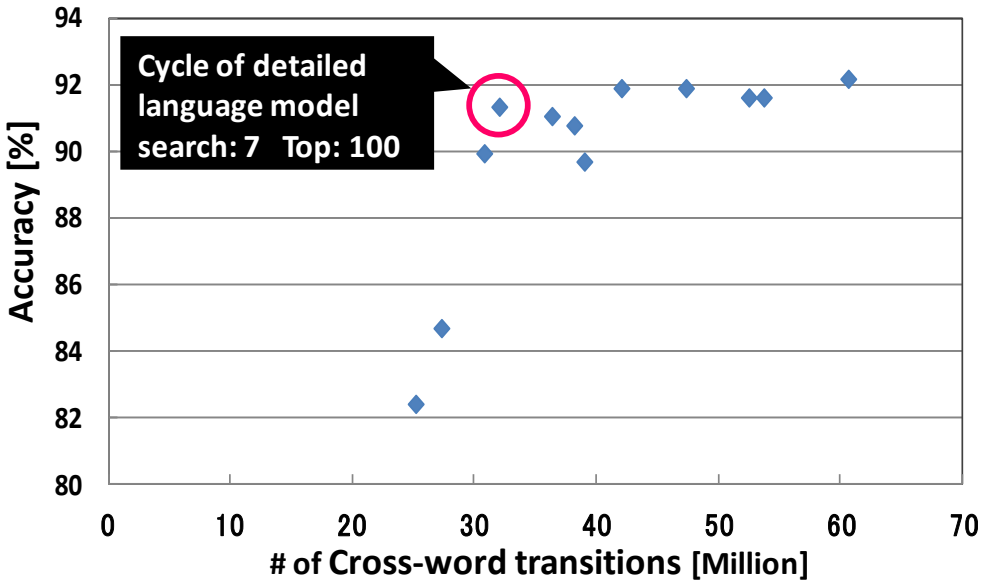

Fig. 3.10 Relation between accuracy and the # of cross-word transitions.

3.5.2 Beam width, the number of cross-word transitions (in detailed search)

In this section, the tradeoff 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.

Figure 3.11 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. Figure 3.12 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.

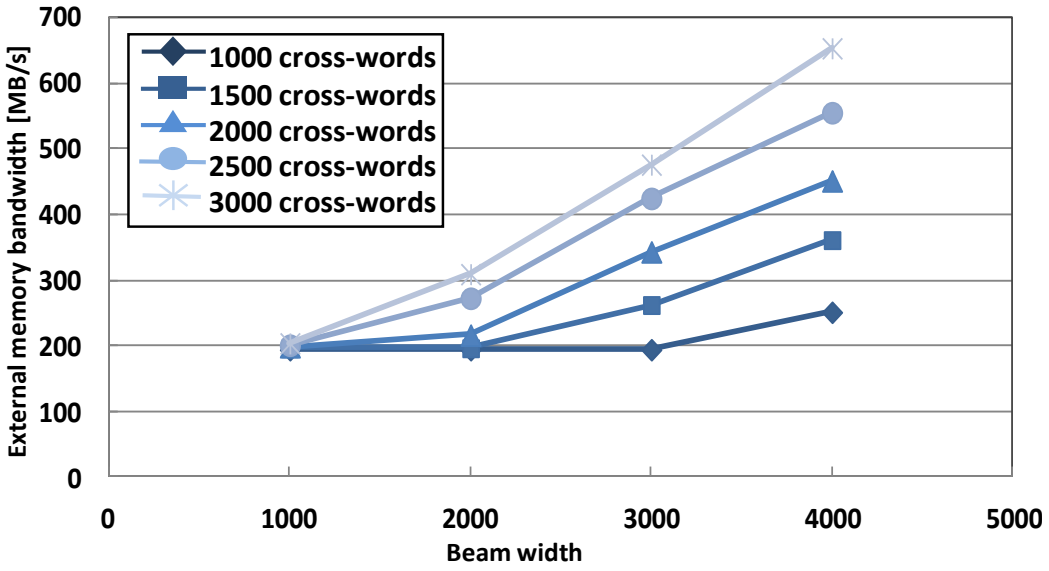

Fig. 3.11 External memory bandwidth versus beam width.

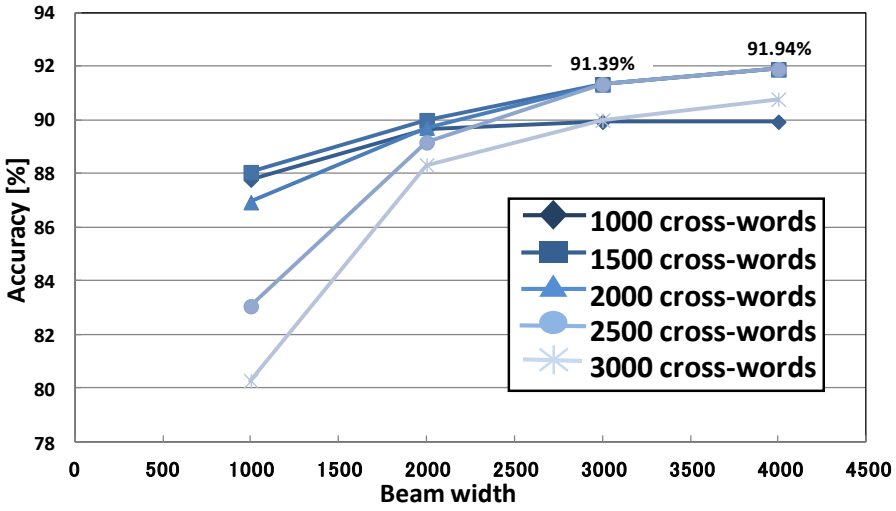

Fig. 3.12 Accuracy versus beam width.

3.5.3 Default parameters

The final choice of the parameter combination is presented in Fig. 3.13. 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.

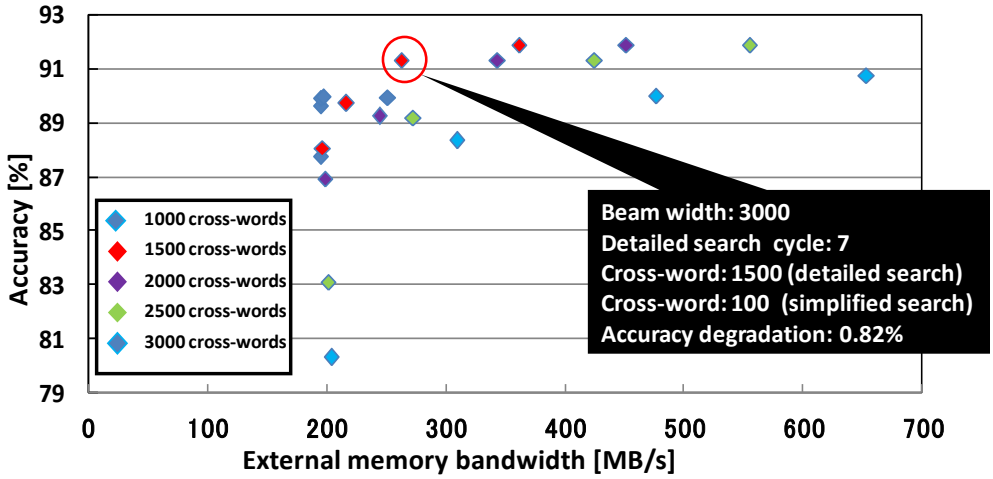

Fig. 3.13 Accuracy versus External memory bandwidth.

3.6 Summary

In this section, we describe the algorithm optimizations to support a hardware-based large-vocabulary recognizer. First, the speech recognition algorithm, as embodied in the Julius software recognizer, which we adopt as our reference model is described. Then the beam pruning using a dynamic threshold to cut the workspace and memory bandwidth for sort processing is introduced. After that, modified unigram language model, which eliminated the updata process for word-internal transitions are explained. Finally, the two-stage language model search scheme which reduce the memory access greatly, and accuracy trade-off of the important parameters are presented. These optimizations are utilized in all the three chips introduced in the following chapters.

Chapter 4 Real-Time 60-KWord Continuous Speech Recognition VLSI Processor

This chapter describes the first chip (HMM1) [13, 14] we implemented for 60-kWord real-time continuous speech recognition. The following techniques are proposed to reduce the cycle count and external memory bandwidth for a large-vocabulary of 60,000 words.

1) Variable-50-frame look-ahead scheme
2) Elastic Pipeline
3) highly parallel GMM architecture
4) Viterbi Cache

4.1 Variable-frame look-ahead scheme

The GMM calculations must load numerous parameters, which requires about 576.03 MB/s when processing the 60-k Word 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.

4.2 Elastic Pipeline Architecture

The overall chip architecture is depicted in Fig. 4.1. 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 elastic pipeline architecture can reduce the memory bandwidth for Viterbi transition. Figure 4.2 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. 4.3), 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.

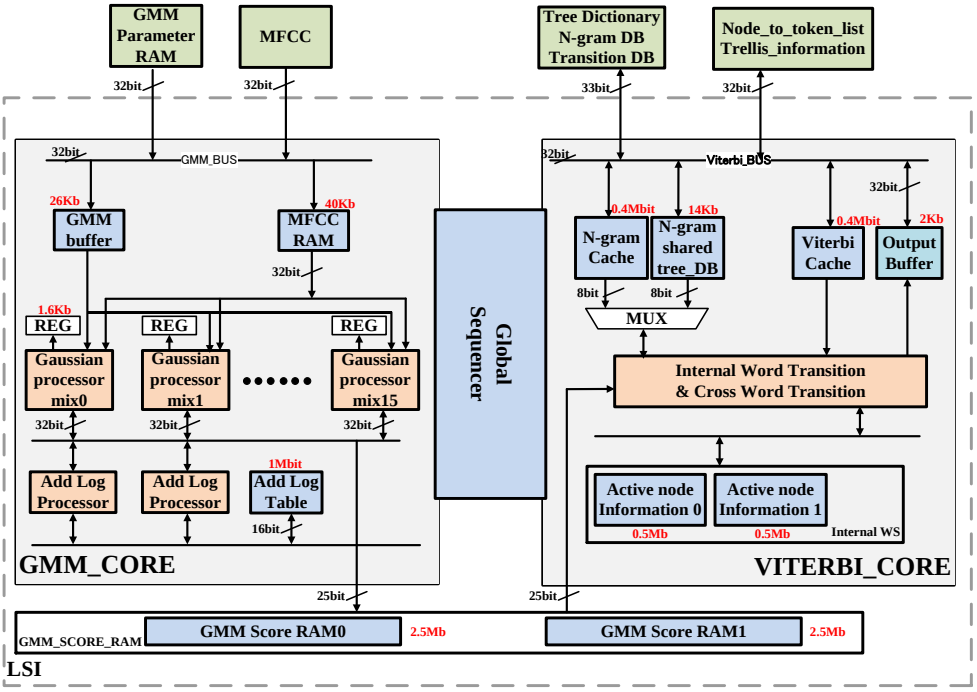

Fig. 4.1 The overall speech recognition architecture.

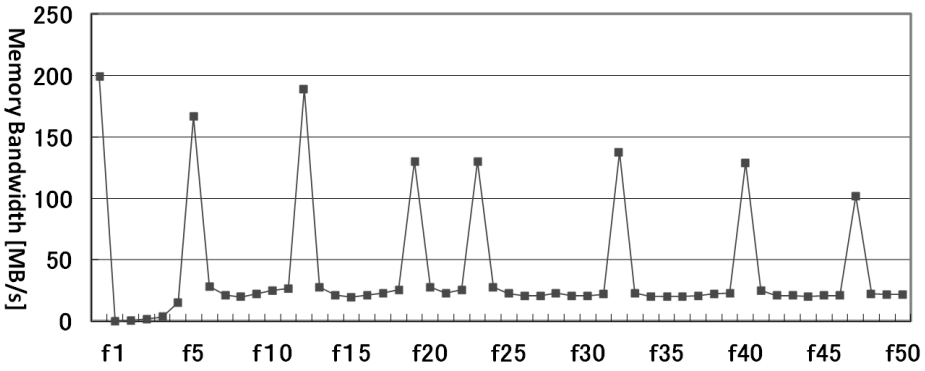

Fig. 4.2 External memory bandwidth variety of Viterbi transition through frames after the two-stage language model search.

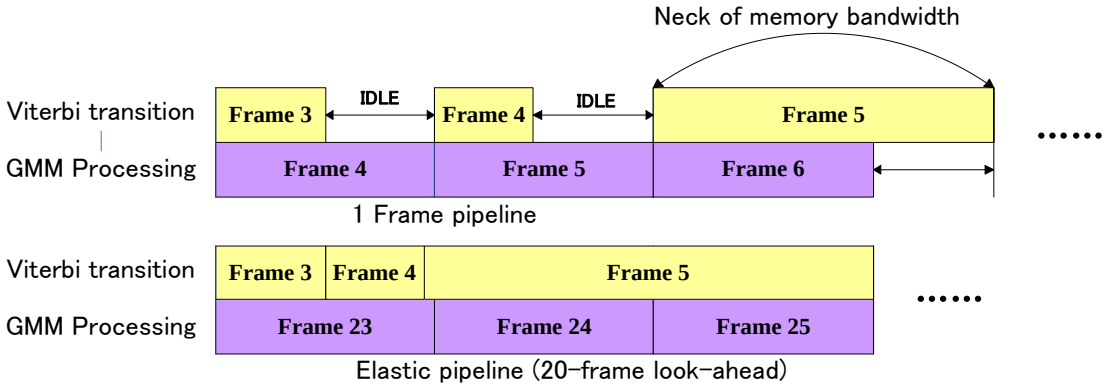

Fig. 4.3 Elastic pipeline.

4.3 Highly Parallel GMM Architecture

Figure 4.4 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 (Fig. 4.1). All blocks are processed simultaneously for the look-ahead frames, which are saved in the MFCC buffer. 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.

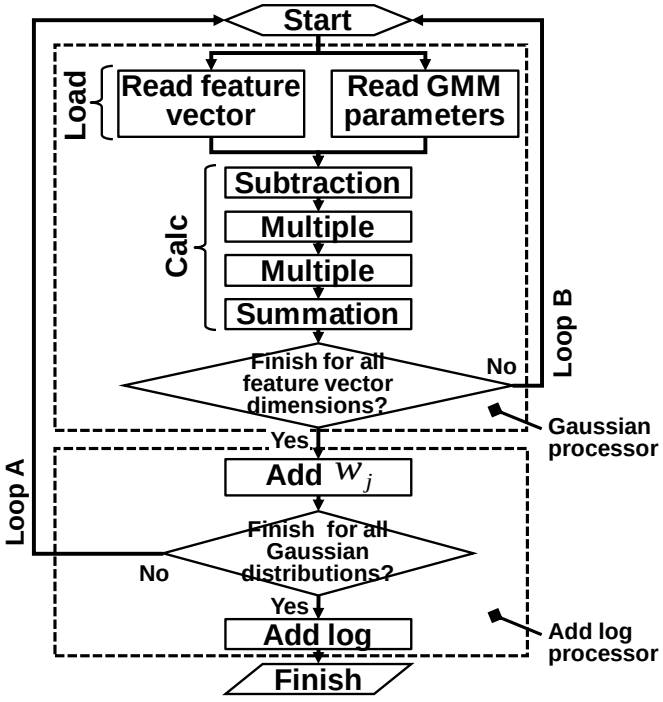

Fig. 4.4 GMM computation flow.

4.4 Viterbi Cache Architecture

Fig. 4.5 shows the Viterbi transition flow. The Viterbi transition in a frame is divided roughly into three steps; word-internal transition, trellis-word save, 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.

Figure 4.7 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. 4.6, 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.

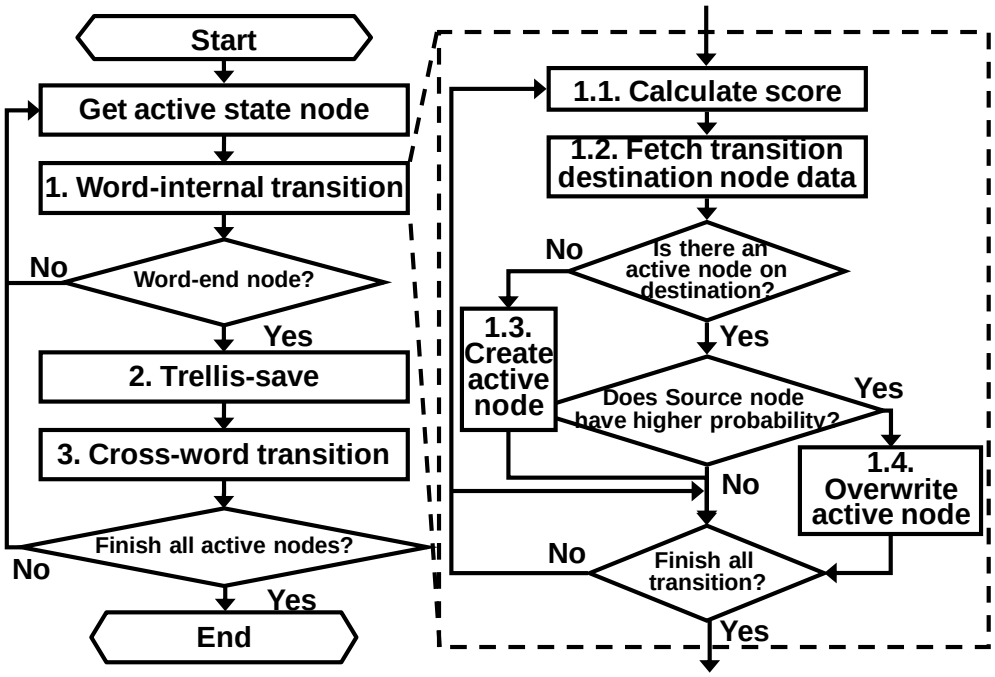

Fig. 4.5 Viterbi computation flow.

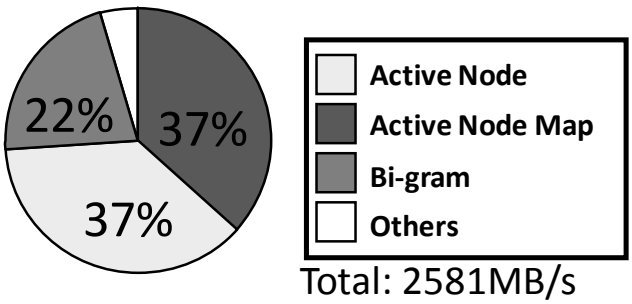

Fig. 4.6 External memory bandwidth of Viterbi transition.

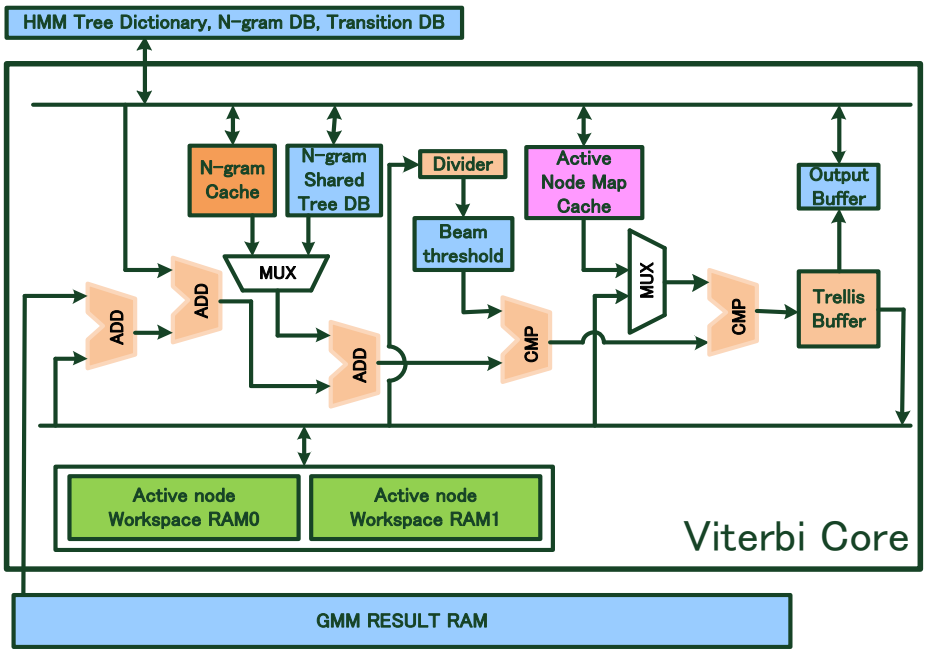

Fig. 4.7 Viterbi cache architecture.

4.4.1 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. 4.8, 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 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.

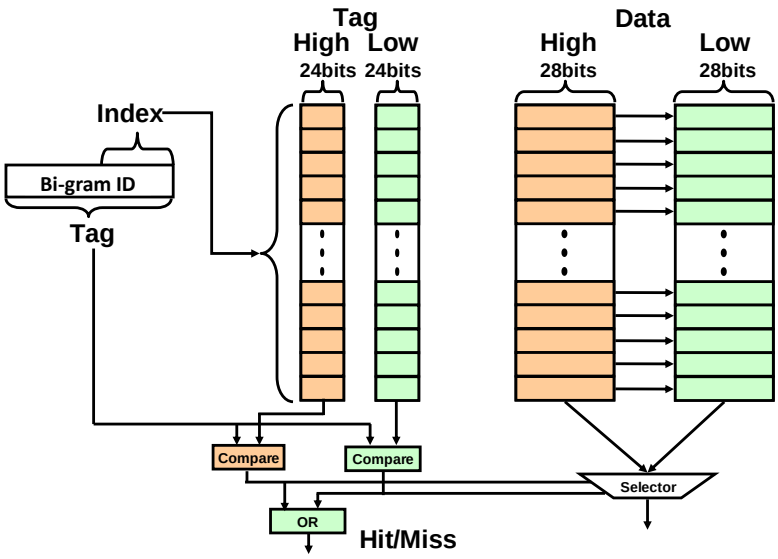

Fig. 4.8 Two-way bi-gram cache.

4.4.2 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. 4.9, 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. 4.10), thereby improving the hit rate by almost 20%.

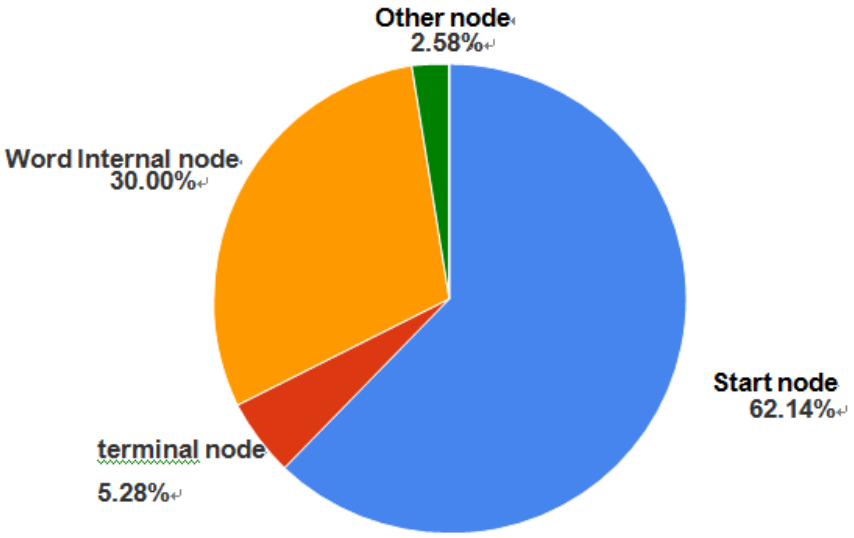

Fig. 4.9 Memory accesses of active node.

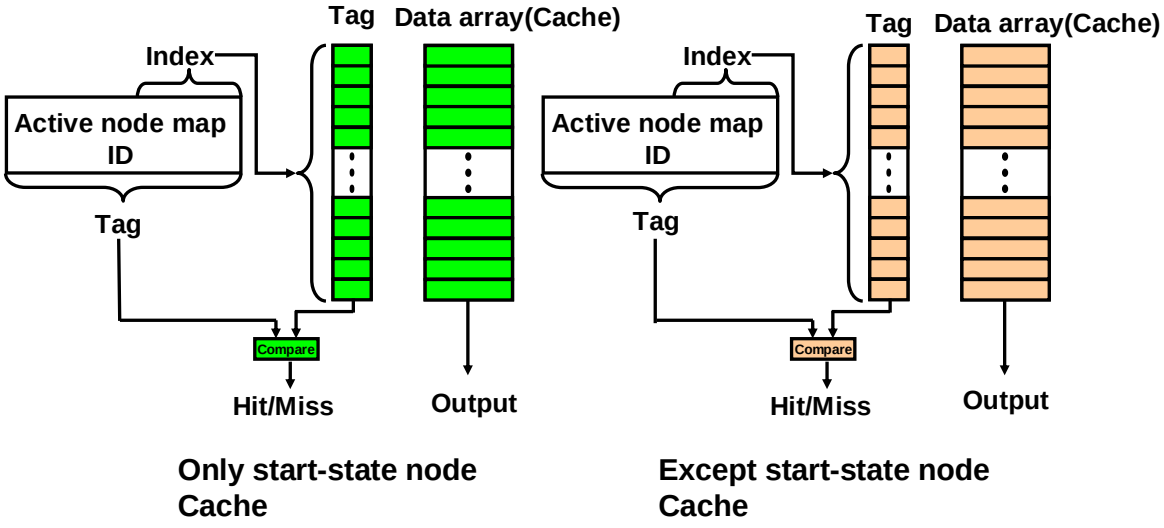

Fig. 4.10 Active node map cache.

4.4.3 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%. Further increase of cache memory size can hardly get a higher hit-rate.

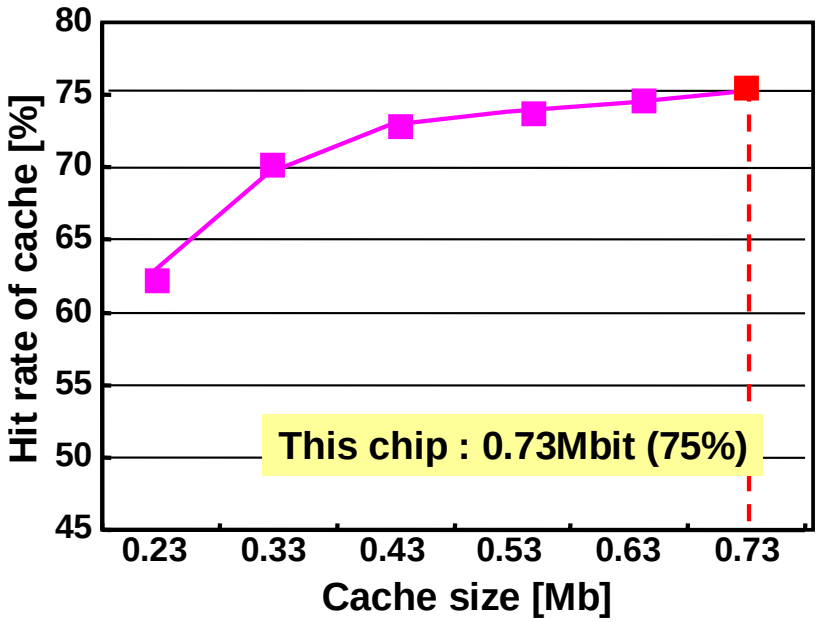

Fig. 4.11 Cache hit rate.

4.5 Implementation

As described in the previous sections, we used software prototype profiling with Julius 4.0 and Microsoft Visual C++ to analyze the accuracy tradeoff 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 4.1 shows the implementation flow and the CAD tools.

Table 4.1 Implementation.

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)

4.5.1 Chip layout

Fig. 4.12 shows a chip for the proposed SoC fabricated using 40 nm CMOS technology. A summary of the chip statistics is shown in Table 4.2. 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. The clock gating is implemented in the GMM result RAM and GMM Core.

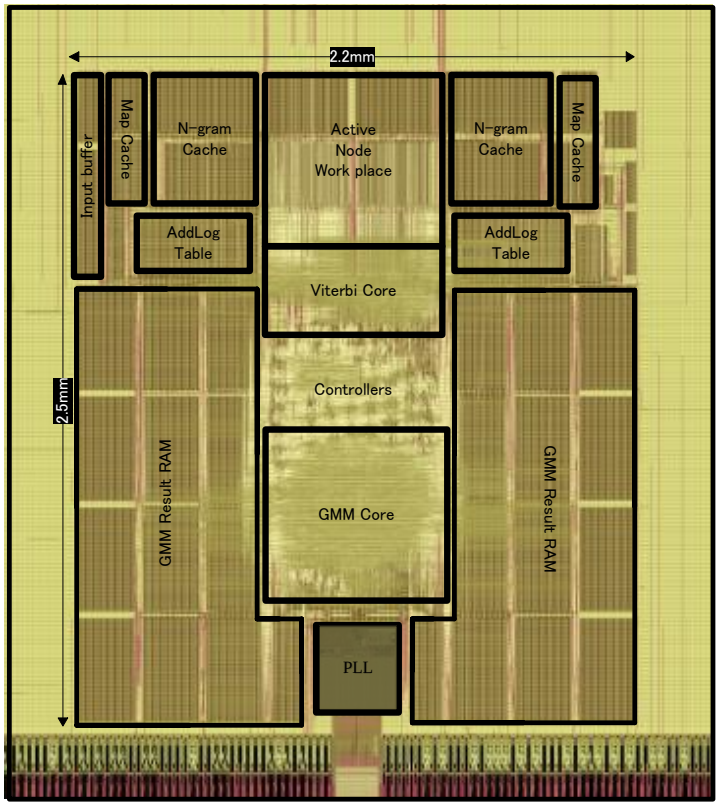

Fig. 4.12 Chip microphotograph.

Table 4.2 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 4.3 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)

4.5.2 Required frequency and memory bandwidth

Fig. 4.13 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.4.14).

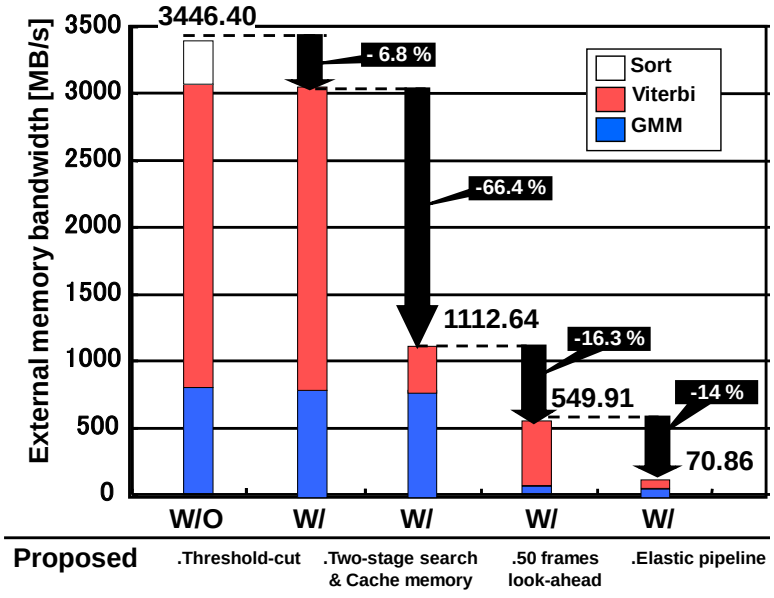

Fig. 4.13 Bandwidth reduction by the proposed schemes.

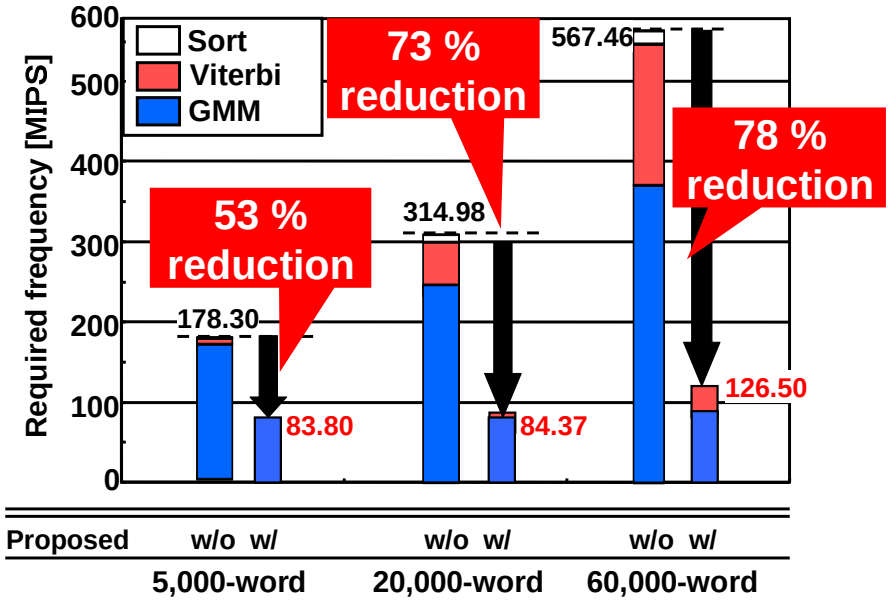

Fig. 4.14 Required frequency reduction for real-time processing.

4.5.3 Power consumption

Using the shmoo plot [18] presented in Fig. 4.15, we measured the power consumption with the lowest operation voltage. Fig. 4.16 and Fig. 4.17 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 4.4, The chip I/O 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 I/O power consumption was reduced greatly by our proposed schemes because it is mainly caused by external DRAM access.

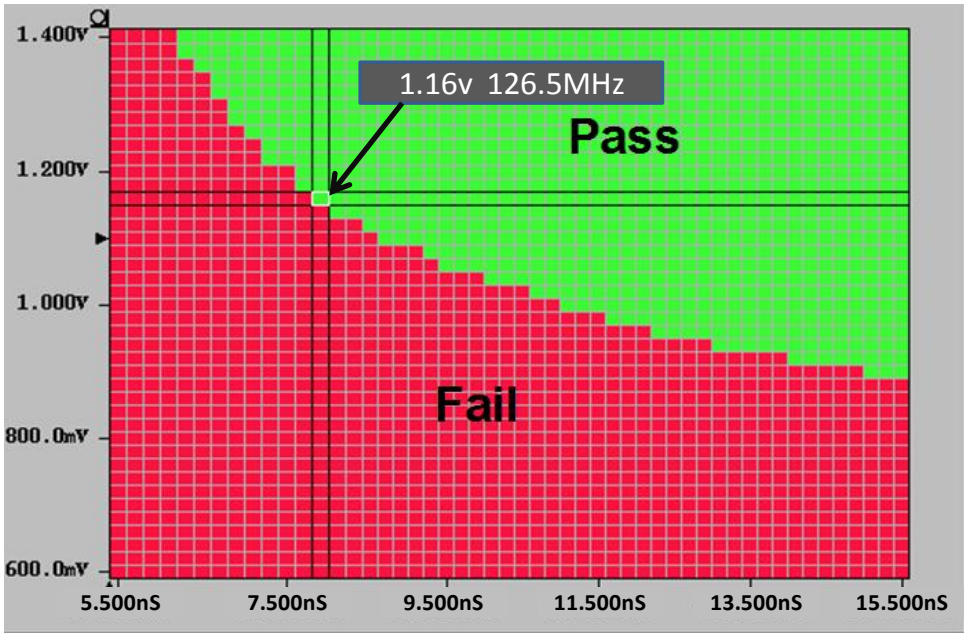

Fig. 4.15 Vdd versus period shmoo plot generated by SOC logic tester.

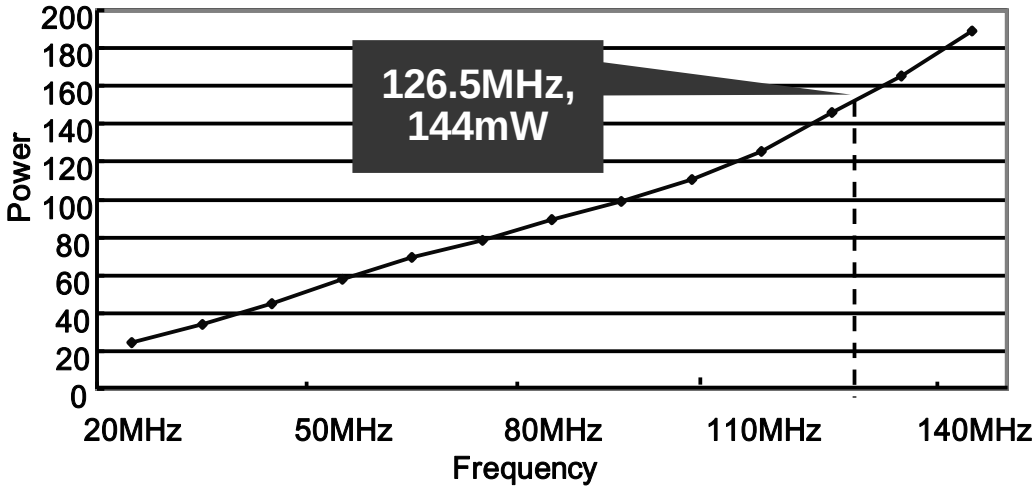

Fig. 4.16 Power consumption with the lowest operation voltage.

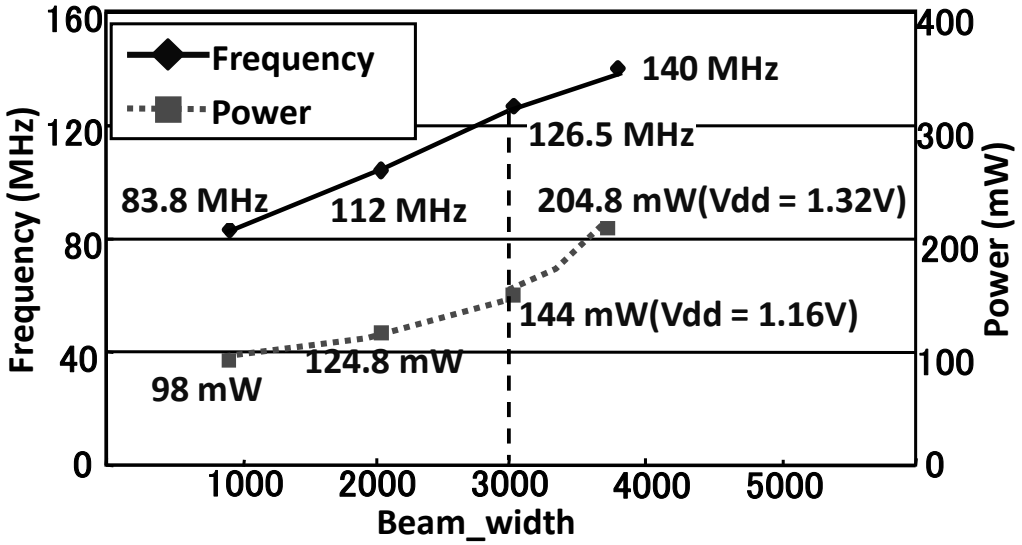

Fig. 4.17 Measurement results of frequency versus power consumption versus beam width.

Table 4.4 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

4.5.4 Comparison with other works

We compared the performance of our chip with those of other recently reported works (Table 4.5) 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 "×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 (Fig. 4.18).

Table 4.5 Comparison with recently reported works.

		[4]	[19]	[12]	[11]	This work
Vocabulary (k)		1	5	5	20	60
Technology		FPGA	VLSI (0.18 um)	VLSI (0.18 um)	FPGA	VLSI (40 nm)
GMM Module	# of states	NA	3,001	3,001	3,001	2,000
	# of distributions	8	16	16	16	16
	# of dimensions	39	39	39	39	25
LM	# of unigram	1,000	5000	5000	19,983	60,001
	# of bigram	2,385	835,000	835,000	1,440,272	4,000,273
Viterbi beam width		NA	NA	NA	500	3000
Real-time factor		2.3	0.55	0.42	0.66	1
Internal Frequency (MHz)		50	133	100	100	126.5
External memory BW (MB/s)		100	530	NA	800	70.86
Core area (mm^2)		NA	8.78	15.47	NA	5.5
Power consumption(mW)		NA	NA	NA	NA	144
Logic elements		13,449 slices	NA	NA	13,835 slices	1.9 MTr. (25,108 slices) *
Internal memory (KB)		138	60.2	140.1	416	975

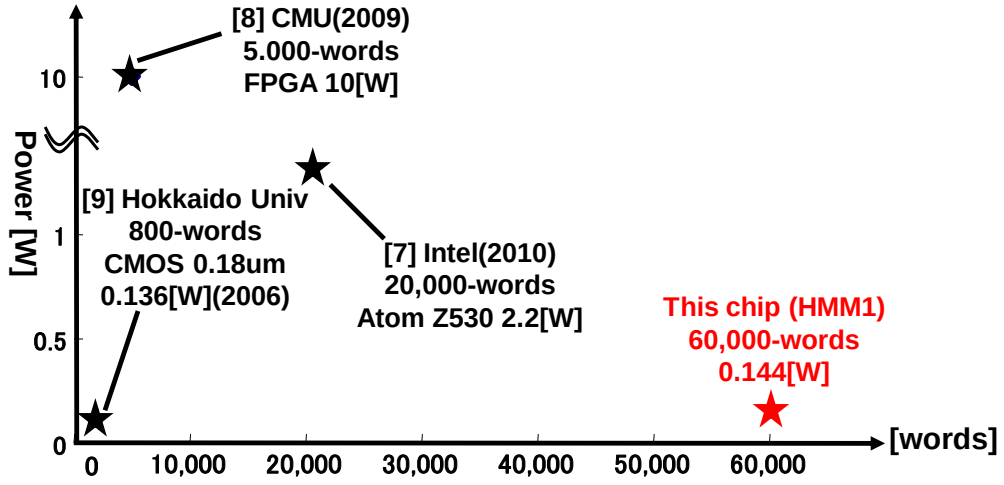

Fig. 4.188 Power consumption comparison.

4.6 Summary

As described in this chapter, several schemes including highly parallel architecture and cache architecture with high hit-rate are proposed to reduce the memory bandwidth and the operating clock frequency for a large-vocabulary of 60,000 words.

1) a variable-50-frame look-ahead scheme;

2) elastic pipeline operation between the Viterbi transition and GMM processing.

3) a highly parallel Gaussian Mixture Model (GMM) architecture based on the mixture level;

4) The Viterbi cache architecture using locality of speech recognition: as some of the

data that has been used in the current frame may be reused in the following frames, on chip caches are introduced for keeping some of them to reduce the external memory access, the specialized cache architecture achieve a hit-rate of 75%;

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 this chip can perform 60-kWord continuous real-time speech recognition at 126.5 MHz.

Chapter 5 2.4x-Real-Time 60-KWord Continuous Speech Recognition VLSI Processor

This chapter describes the second chip (HMM2) [15, 20] we implemented for 60-kWord real-time continuous speech recognition. Although the external memory bandwidth was reduced adequately in the previous work, it needed a large amount of on-chip memory and caused long latency. Therefore in HMM2, we try to find another way to reduce the external memory bandwidth. The following techniques are proposed to achieve a smaller chip area and higher processing speed .

1) Compression-decoding scheme

2) max-mixture approximation GMM calculation

3) 4-pass Viterbi transition unit

5.1 Speech Recognition System

The overall chip architecture, depicted in Fig. 5.1, comprises the GMM core, Viterbi core, double buffer for the GMM result, and the memory interface. We use a PowerMedusa [25] custom test board to construct a speech recognition system with a test chip. The MFCC feature vectors are extracted using a PC. The reason why we separate the feature extraction part from the recognizer is as follows:

1) Firstly, the use of fixed-point computation in the feature extraction part will cause big degradation [26] [27] in recognition accuracy (3%-5%);

2) Secondly, the computation workload for feature extraction is small and can be easily handled by PC or an embedded soft-core [4] [11] ;

3) Lastly, it is flexible for adopting other capabilities like noise reduction or speaker adaptation [11].

The input speech data can either be recorded as an audio stream or with real-time speaking. The database is set up at the beginning. The test chip accesses the DRAM through an on-board FPGA. The data-path of the DRAM is 64 bit, but only 48pin is available for the test chip according to the pin limitation. The data-path for GMM computation (16pin) and Viterbi search (32pin) is separated to support pipeline

operation.

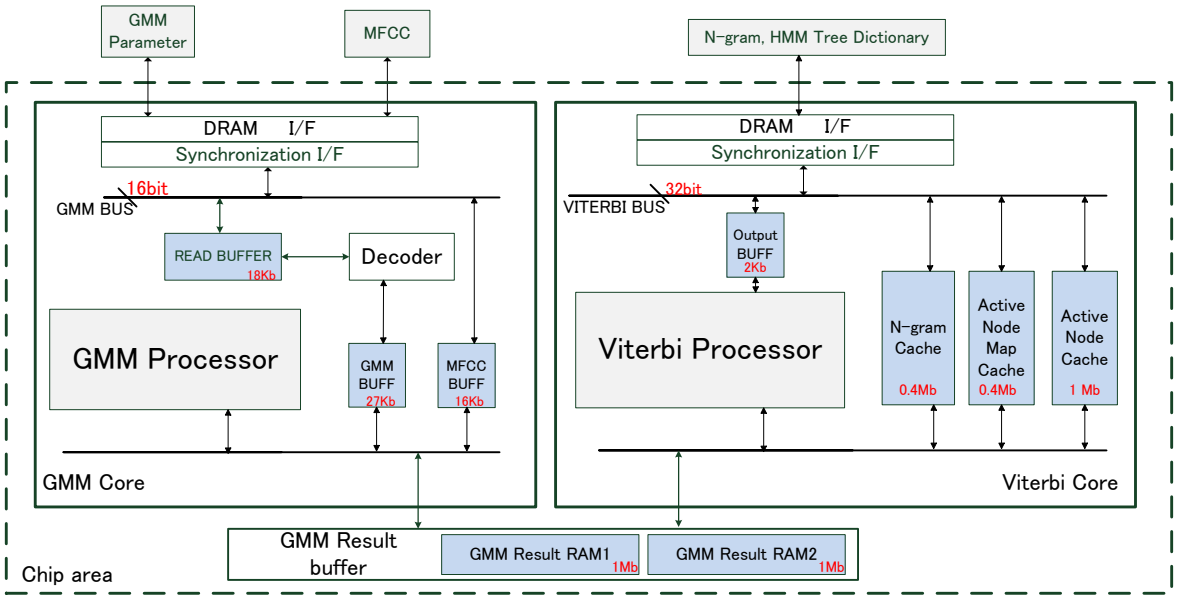

Fig. 5.1 Overall speech recognition architecture.

5.2 GMM Architecture with Compression-Decoding

The external memory bandwidth for GMM computation can be reduced by sharing the parameters for several frames. We can use this scheme to reduce the memory access as much as needed, theoretically. A variable 50-frame-look-ahead scheme was used in HMM1 [13, 14], which reduced the external memory bandwidth to 13.3 MB/S for GMM computation. However, it becomes cost-ineffective when the look-ahead frames are numerous because the reduction efficiency becomes worse while the on-chip memory for saving the GMM result is proportionally increased. For the proposed method, we adjust the number of look-ahead frames to optimize the area.

The maximal IO frequency of the test chip is 50 MHz and the data pins for GMM computation are 16pin. Therefore, the IO transfer capability is 100 MB/S. The GMM parameters include 1987 states each with 16 mixtures. There are 52 parameters for one mixture. The bit-width of one parameter is 32 bit. Therefore 2 IO cycles are necessary to load one GMM parameter. There are 100 frames in 1S speech. Assuming *n* frames look-ahead to support 2.4×-real-time processing (decided by Viterbi), *n* should be 32 according to the following limitation:

4B × 52 × 16 × 1987 × 100/n × 2 × 2.4 MB/S < 100 MB/S,

which requires roughly 3.2 Mbit of result RAM and causes latency of 0.32 S. For further optimization, We proposed a compression–decoding scheme. GMM parameters saved in DRAM are compressed state-by-state to a smaller size using a lossless data compression algorithm. When the chip starts processing, the compressed data are transferred to a buffer inside the chip and then decoded. The parameters are rebuilt completely and therefore have no degradation to recognition accuracy. Then the compression-decoding scheme can reduce the required external memory bandwidth by the compression ratio.

The GMM architecture and pipeline operation are portrayed respectively in Fig. 5.2 and Fig. 5.3 The GMM core comprise a MFCC buffer for feature vectors, two input buffers for loading the compressed data, a GMM buffer for the decoded parameters, one decoder and 20 GMM computation processors. As shown in Fig.5, the compressed data of state n is loaded to the input buffer while the decoding and the computation for state n-1 is treated.

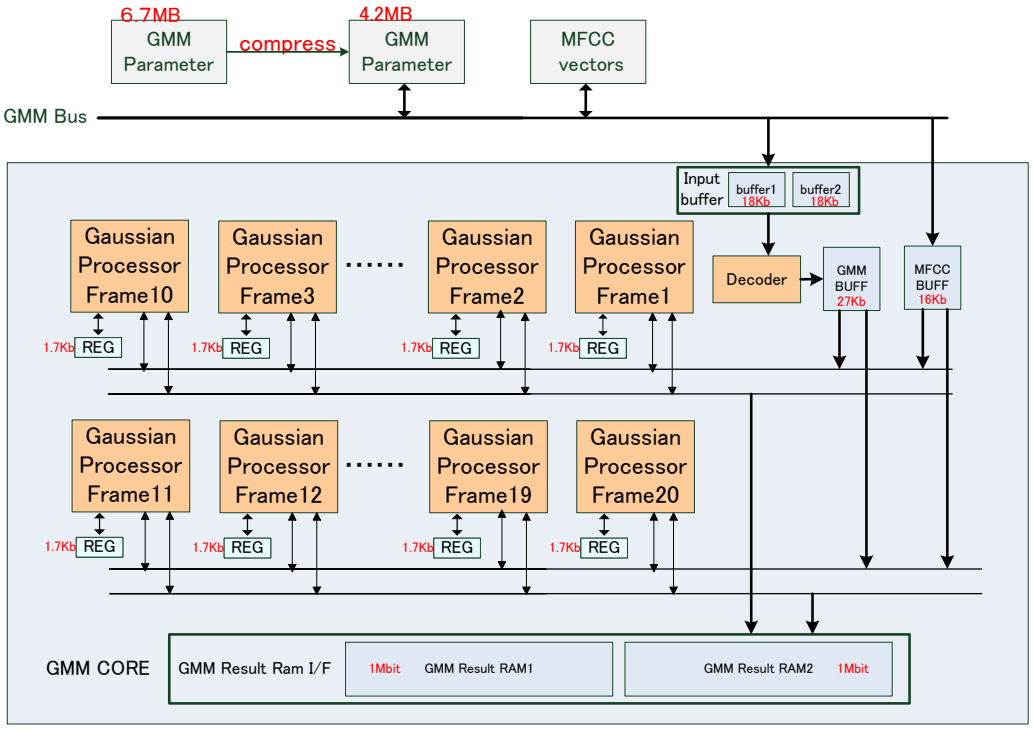

Fig. 5.2 GMM architecture with compression-decoding scheme.

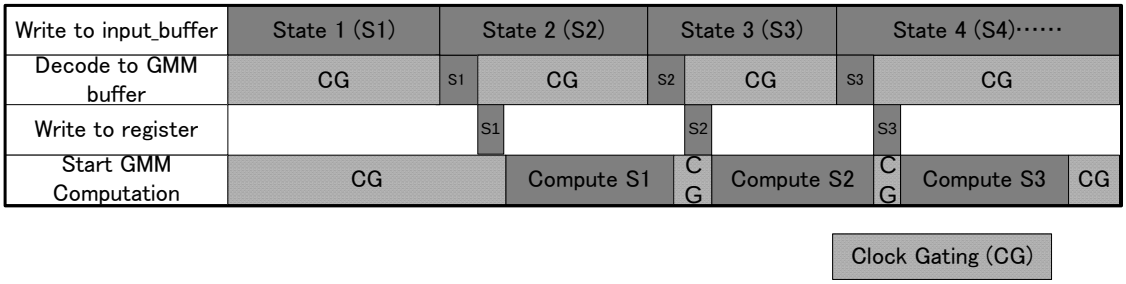

Fig. 5.3 Pipeline operation in GMM core.

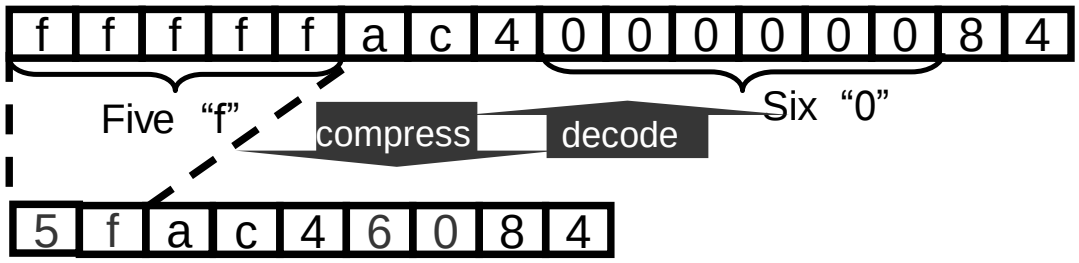

Fig. 5.4 Run Length Encoding (RLE) for GMM parameters.

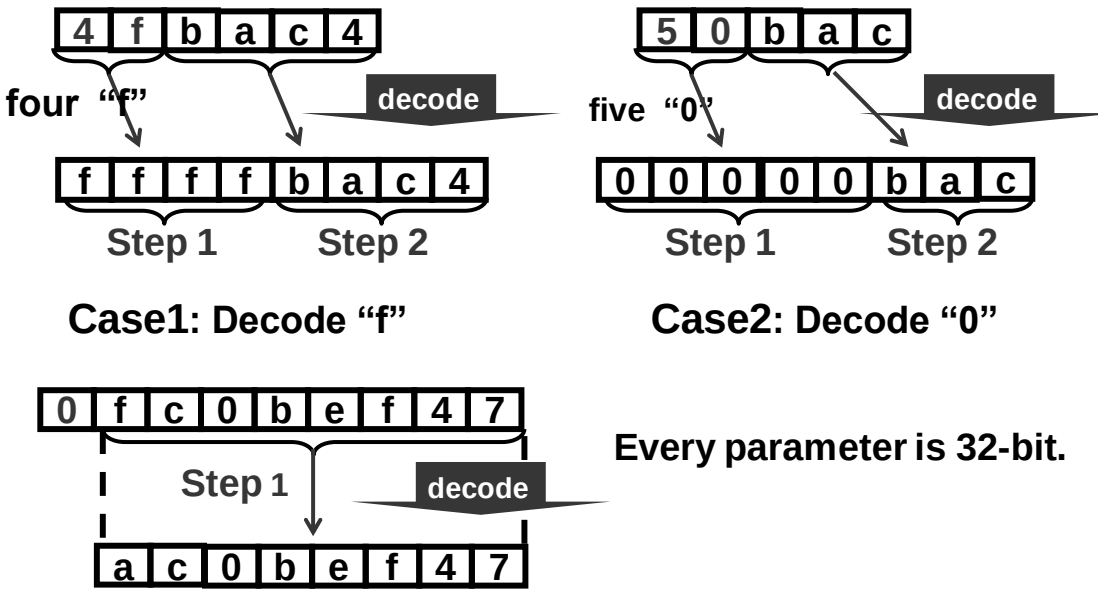

Fig. 5.5 Three cases for decoding operation.

Clock gating is implemented in the decoder and the GMM processors. After decoding state n, the decoder will be clock gated until the loading for state n+1 is accomplished. GMM processors are also clock gated in the same way. The power consumption reduction by clock gating is measured by simulation with 40nm library and power compiler. Because of the high leakage current in 40nm process, only 1.25% power reduction is confirmed.

The decoding time for one-compressed-state parameters must be shorter than the loading time for a complete state and the extra computation workload and logic elements for decoder should be small. Therefore, we choose the run-length-encoding (Fig. 5.4) algorithm for our decoder. To adopt the GMM parameter, we modified the RLE operation, instead of compressing all bit, we merely compressed the top *n* bit for one parameter. There are three cases for decoding as shown in Fig. 5.5. In Case one or case two 2 steps are needed to generate one parameter while in case three only one step is needed. As there are 832 parameters for one state, the total number of operation is less than 832*2 = 1664. The average compression ratio is 37.5% with only 0.01% extra computation workload and 0.005% area overhead of the GMM core. Consequently, the number of look-ahead frames is reduced further to 20.

5.3 Max-Mixture Approximation GMM Calculation

According to the required processing speed of this chip, it is needed to increase the degree of parallelism. However, mixture-level parallel implementation is not flexible because only limited degrees can be utilized (16, 32 etc.), which will cause unnecessary increment of logic elements. Therefore we need to implement the parallel architecture at the frame level and it is just suitable for the 20-frame look-ahead scheme. As described herein, we implement a 20-frame parallel architecture for GMM computation to reduce the cycle count for the computation part. Figure 5.6 presents both the 16-parallel processing at mixture level of the previous chip HMM1 and the 20-parallel processing at frame level of this chip.

We calculate the log probability density function (PDF) by its max approximation instead of the previous log-table based one, which saves about 1 Mbit of on-chip memory while cause 049% degradation in regocnition accuracy.

$$\log b_s(X_t) = \max_m \{C_m - \frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - \mu_{md})^2}{\sigma_{md}^2}\} \tag{5.1}$$

Therein, $\log b_s(X_t)$ represents the state output probability of a HMM state s for feature vector X_t at time t; x_d stands for the vector component of the feature vector X_t, D is the feature dimension, and C_m, μ_{md}, σ_{md} respectively denote the constant, the mean, and the standard deviation of Gaussian mixture model.

The GMM computation flow using the max-approximation algorithm is shown in Fig. 5.7. As the value of one mixture is getting smaller with the calculation going on, the computation for one mixture can be stopped when its value is less than the max result of the previous mixtures. Around 40% cycle count reduction is achieved by the above changes. The power consumption variety of 16-core and 20-core measured by simulation are shown in Fig. 5.8. Power consumption for real-time processing is reduced by 20%.

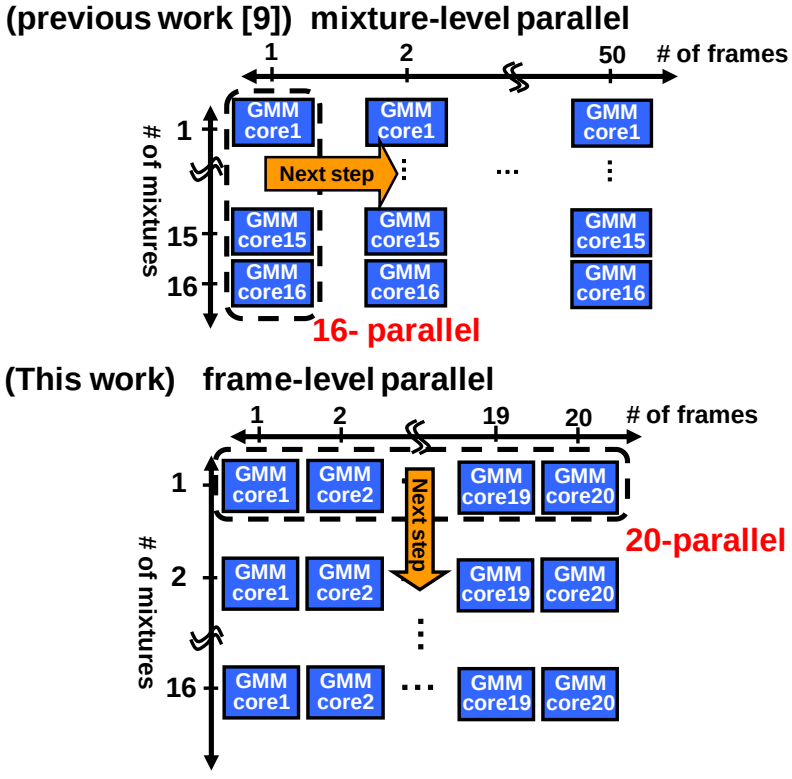

Fig. 5.6 20-frame parallel processing versus 16-mixture parallel processing.

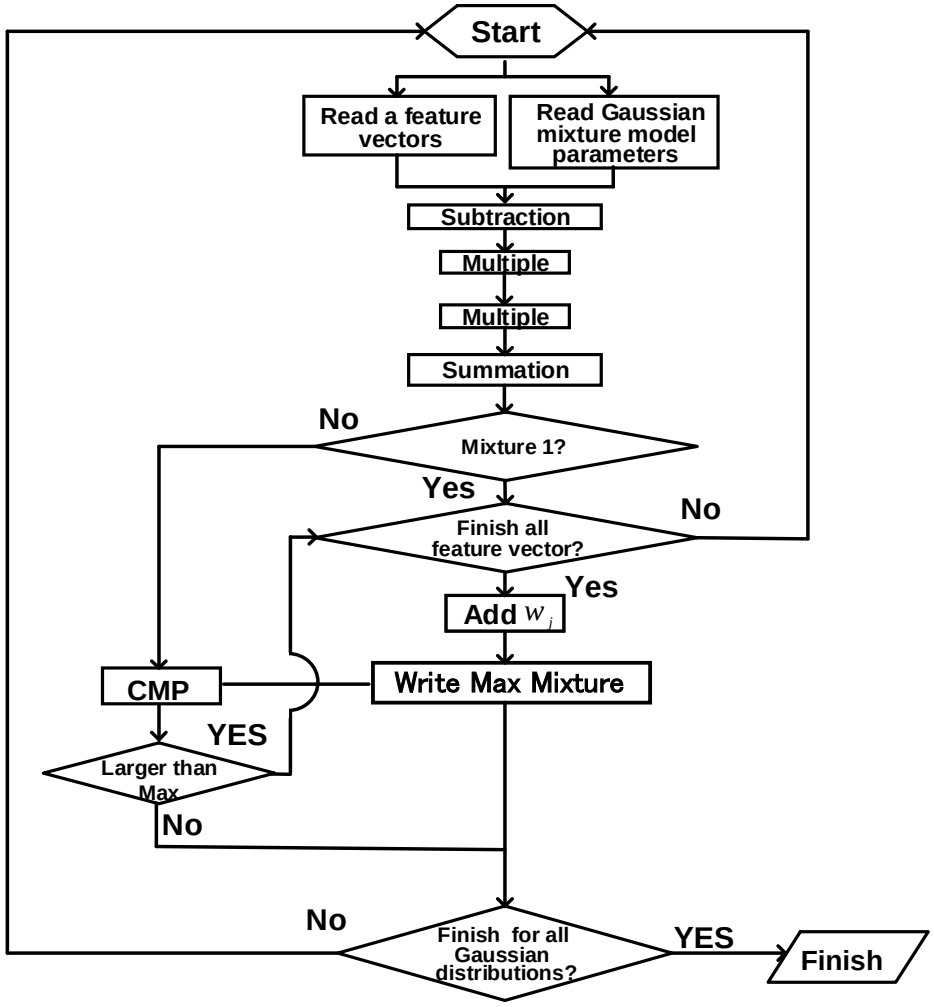

Fig. 5.7 Max-mixture approximation GMM computation flow.

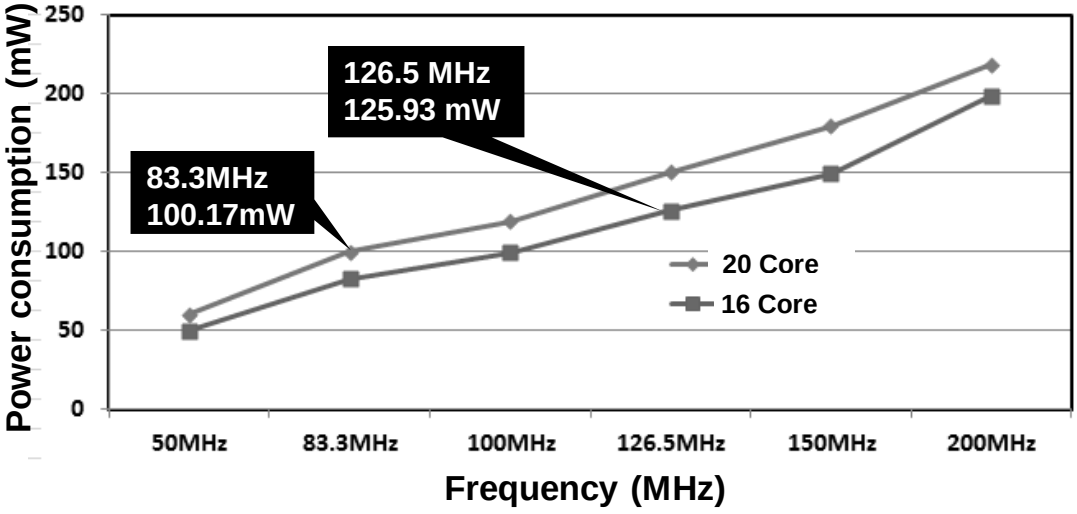

Fig. 5.8 Power comsumption for real-time GMM computation of 16-core and 20-core measured by simulation.

5.4 4-Path Viterbi Transition Unit

The external memory bandwidth required in Viterbi processing has been reduced greatly by two-stage language model (LM) search, the specialized cache memory, and the elastic pipeline described in the previous Chapters. All GMM probabilities and active node information can be read out from the internal RAM. However, when a mis-hit occurs at the *n*-gram cache or active-node map during cross-word transition, it will take two IO cycles, which is eight internal cycles to access the external database. In the previous chip [9], the processing is stopped to wait until the required data is got from the DRAM, this latency strongly delays the Viterbi processing. Although the internal data-path to the caches are available at this time, it is impossible to process the next transition. Consequently in this chip, we proposed a four-path Viterbi-processing units to hide the latency as portrayed in Fig. 5.9. Figure 5.10 shows the pipeline operation. When a mis-hit occurs at one of the transition paths, the other units will access the cache memory and continue processing other transitions. This scheme eliminates 34.2% of the cycle counts for the Viterbi transition. Because the computation time for Viterbi is the neck of the GMM-Viterbi pipeline processing, the total computation time is reduced by the same rate.

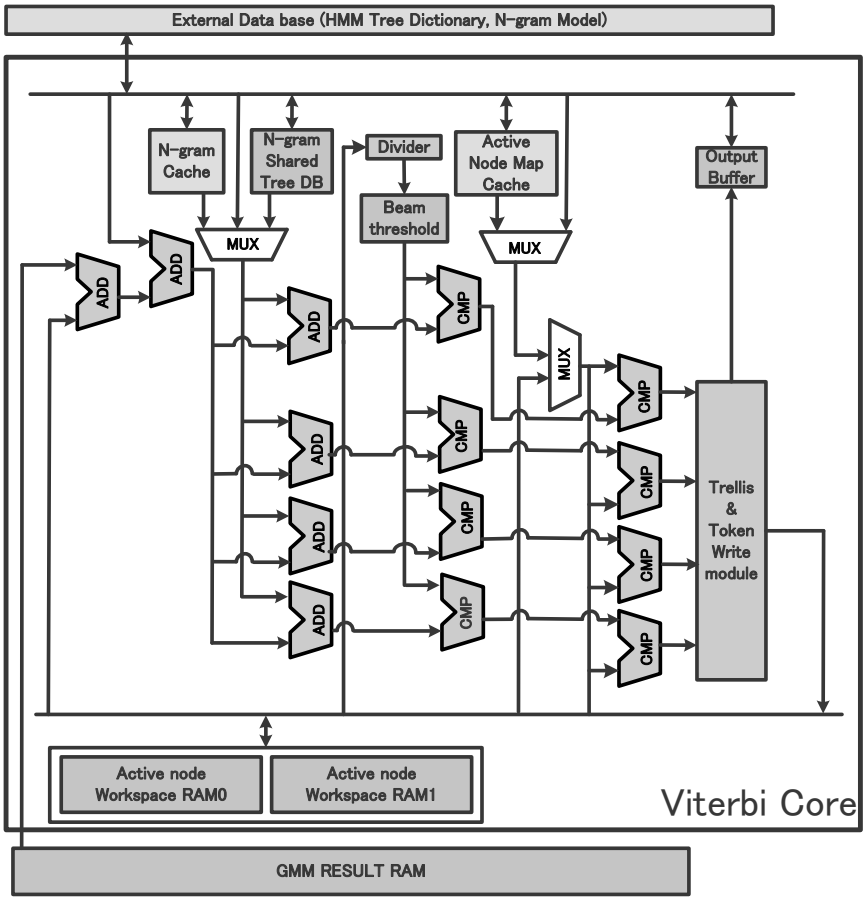

Fig. 5.9 4-path Viterbi transition architecture.

Path 1:	N-hit	ADD	CMP	Map-hit	CMP	write	N-hit	ADD	...
Path 2:		N-hit	ADD	CMP	Map-hit	CMP	write	N-hit	...
Path 3:			N-hit	ADD	CMP	Map-hit	CMP	write	...
Path 4:				N-hit	ADD	CMP	Map-hit	CMP	...
Clock count	1	2	3	4	5	6	7	8	...

Path 1:	Miss-hit	Access external DRAM								...
Path 2:		N-hit	ADD	CMP	Map-hit	CMP	write	N-hit	ADD	...
Path 3:			N-hit	ADD	CMP	Map-hit	CMP	write	N-hit	...
Path 4:				N-hit	ADD	CMP	Map-hit	CMP	write	...
Clock count	1	2	3	4	5	6	7	8	9	...

N-hit : N-gram cache Map-hit : Active node map cache

Fig. 5.10 Pipeline operation for viterbi transition.

5.5 Implementation

The chip, which was fabricated in 40 nm CMOS technology as shown in Fig. 5.11, occupies 1.77×2.18 mm^2 containing 2.52 M transistors for Logic and 4.29 Mbit on-chip SRAM. A summary of the chip statistics is shown in Table 5.1. The breakdown of the internal memory is shown in Table 5.2.

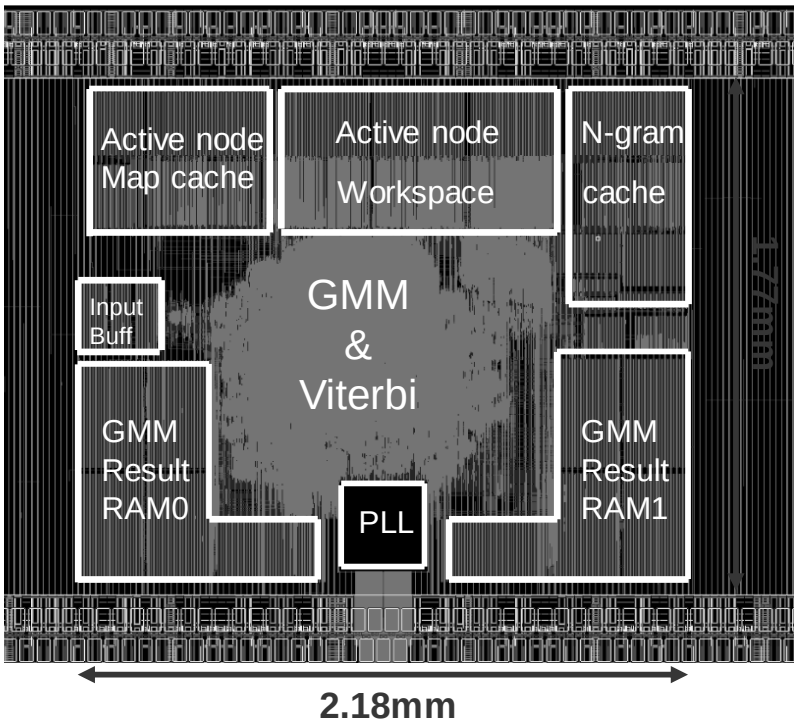

Fig. 5.11 Chip layout.

Table 5.1 Summary of chip implementation.

Process Technology		40-nm CMOS
Core area		2.18 mm × 1.77 mm
Chip area		5 mm × 2.5 mm
Transister Count (Logic	GMM	1.58 M
	Viterbi	0.43 M
	Other	0.51 M
	Total	2.52 M
On-Chip Memory (SRAM)		4.29 Mbit
Supply voltage		1.1V
I/O voltage		3.3V
Evaluation environment		SOC Tester System
Operating Frequency		83.3 MHz for 60 kWord real-time processing
Measured Power (without IO)		74.14 mW
Leakage current		1.35 mA

Table 5.2 Breakdown of Internal memory.

Description	# of SRAM	Size of SRAM (Kb)
gmm_result_ram	10	2000
gmm_buffer	1	26
Input_buffer	2	36
MFCC_buffer	1	16
active_node_work_space	6	1248
shared Tree DB	1	14
bi-gram	4	468
active_node_map	6	476
initial parameter	1	4
output buffer	1	2
Total	50	4290 (536kB)

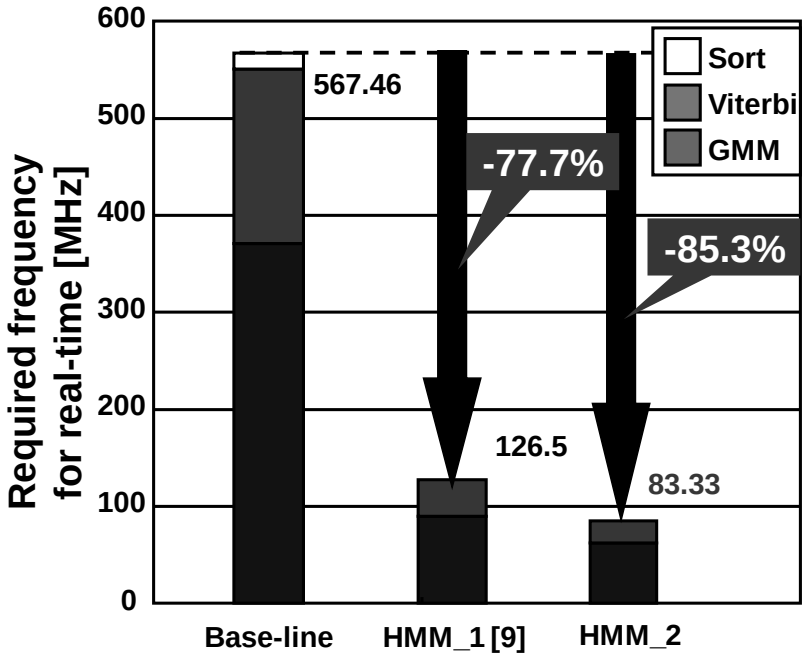

Fig. 5.12 Required frequency reduction for real-time processing.

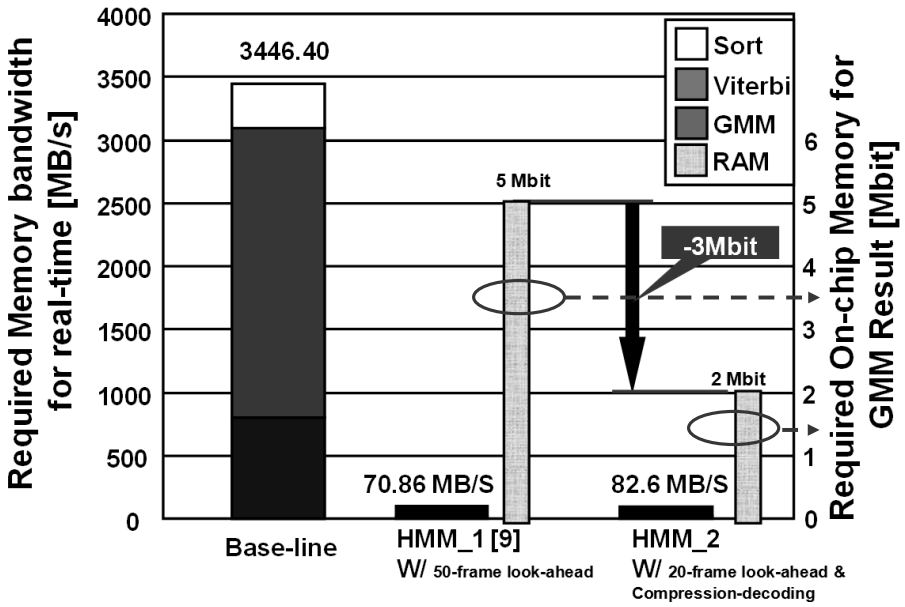

Fig. 5.13 Required memory bandwidth reduction & GMM result RAM reduction.

We implement a software prototype profiling with Microsoft Visual C++ and a referential hardware using hardware descrption language (HDL) to check the required memory bandwidth and frequency for real-time operation. The reduction of both frequency and external memory bandwidth are presented in Fig. 5.12 and Fig. 5.13. The required frequency for real-time processing is reduced by 34.2% compared to HMM1. This rate is just the same as Viterbi part because the computation time for Viterbi transition is the neck of the GMM-Viterbi pipeline operation. Although the external memory bandwidth was reduced greatly in HMM1, it needed a large amount of SRAM and caused long latency. Therefore in HMM2, we optimize the number of look-ahead frames cooperating with a compression decoding scheme to reduce the external memory bandwidth for GMM computation, which cuts down 3 Mbit of GMM result RAM as shown in Fig. 5.13.

We evaluated the test chip with a logic tester. The generated Shmoo plot is presented in Fig. 16. Figure 17 shows the power consumption at different frequency. This chip can process real-time 60-kWord continuous speech recognition at 83.3 MHz and 0.84 V with power consumption of 74.14 mW which is only half of the power consumed by HMM_1. It can maximally process at 2.4× faster than real-time at 200 MHz with

standard voltage of 1.1 V while dissipating 168 mW (Fig. 18). Table 5.3 presents a comparison between this work and some recently announced works in terms of the vocabulary size, GMM model, language model, beam-width, recognition accuracy, real-time factor, operation frequency, internal memory, external memory bandwidth, area and the logic element. Power consumption of both core and IO are also included. Figure 5.17 plots the vocabulary versus speed of the state of art hardware-based recognizers.

The beam-width is the number of active node we treat every frame, as described in the previous Chapters, larger beam-width can afford higher recognition accuracy while increase the computation workload. Consequently, although the use of GMM approximation computation cause 0.49% accuracy degradation comparing to HMM_1, because of the performance-improvement, this chip can recognize speech with a larger beam-width of 4000 which provide a better recognition accuracy of 91.45% as shown in Table 3. In that case, the power consumption for real-time processing is increased to 97.4 mW and the max-performance is decreased from 2.4x to 2.08x.

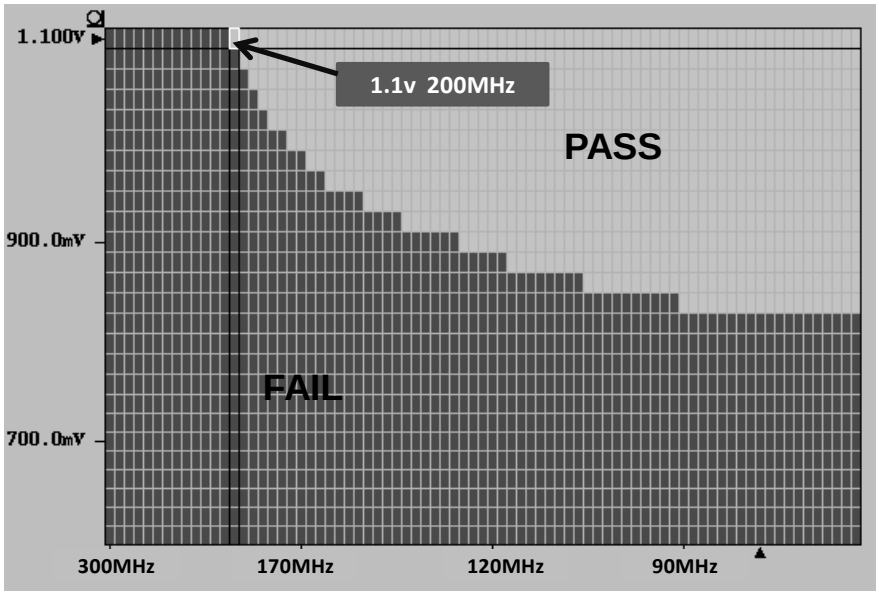

Fig. 5.14 Shmoo plot generated using a logic tester.

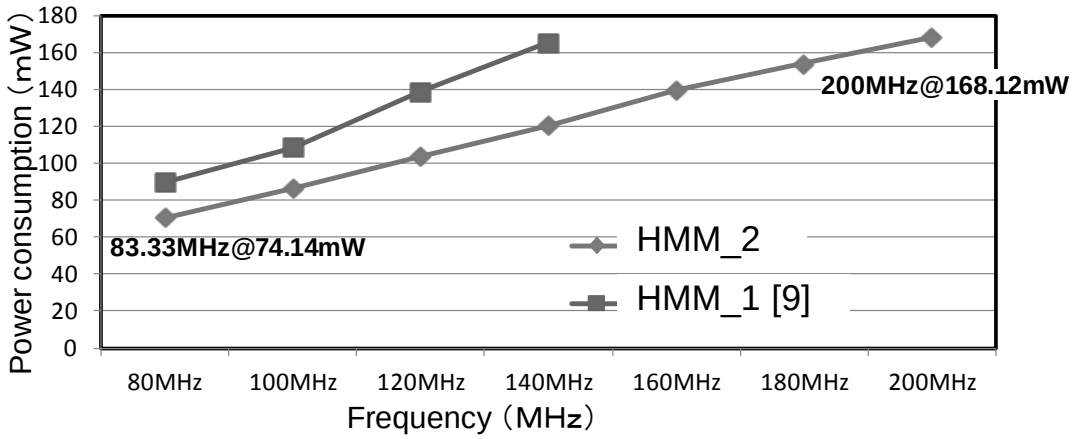

Fig. 5.15 Power consumption versus operation frequency.

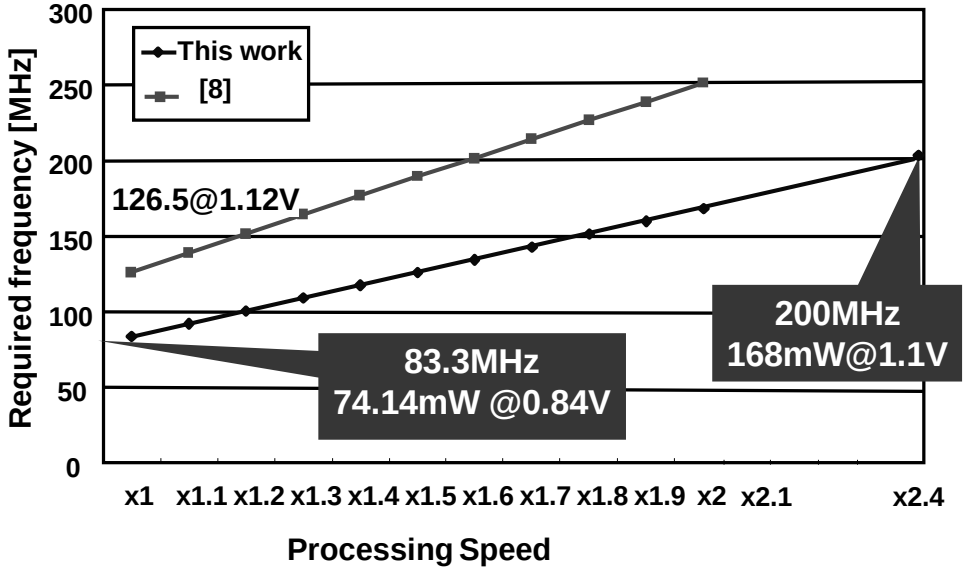

Fig. 5.16 Processing speed versus required frequency.

Table 5.3 Comparison with recently reported works.

		[12]	[11]	[13] (HMM_1)	This work (HMM_2)			
Vocabulary (k)		5	20	60	60			
Technology		VLSI (0.18 um)	FPGA	VLSI (40 nm)	VLSI (40 nm)			
GMM Mod	# of states	3,001	3,001	2,000	2,000			
	# of distributions	16	16	16	16			
	# of dimensions	39	39	25	25			
LM	# of unigram	5000	19,983	60,001	60,001			
	# of bigram	835,000	1,440,272	4,000,273	4,000,273			
Viterbi beam width		NA	500	3000	3000		4000	
Accuracy (%)		89	88	91.39	90.9		91.45	
Real-time factor		2.38x	1.51x	1x	1x	2.4x	1x	2.08x
Internal Frequency (MHz)		100	100	126.5	83	200	96	200
IO Frequency (MHz)		100	100	31.625	21	50	24	50
External memory BW (MB/s		NA	800	70.86	83	198	96	199
Power consumpt	Core (mW)	NA	NA	144	74	168	97	172
	IO (mW)	NA	NA	58.2	39	94	45	94
Core area (mm^2)		15.47	NA	5.5	3.86			
Logic elements		NA	13,835 slices	1.9 MTr	2.52 MTr.			
Internal memory (KB)		140.1	416	975	536			

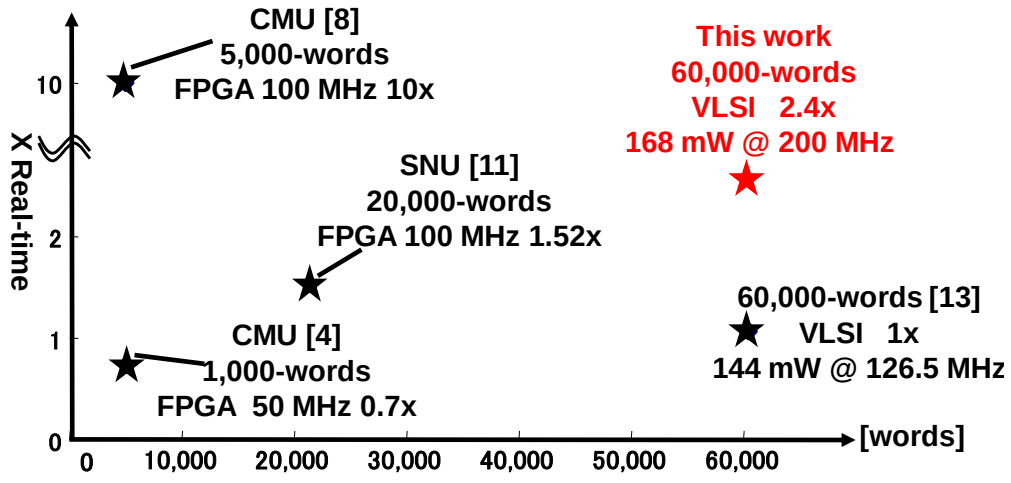

Fig. 5.17 Vocabulary versus speed.

5.6 Summary

As described in this chapter, several schemes are proposed to achieve a smaller area and higher processing speed.

1) a compression–decoding scheme to reduce the external memory bandwidth for GMM computation, which reduce the result ram;

2) max-mixture approximation GMM calculation, which reduce the add-log table.

3) a 4-path Viterbi transition unit to hide the DRAM latency when a mis-hit occurs.

The measured results show that our implementation achieves 34.2% required frequency reduction (83.3 MHz) , 48.5% power consumption reduction (74.14 mW) and 30% area reduction compared to the previous work for 60 k-Word real-time continuous speech recognition with recognition accuracy of 90.9%. This chip can maximally process 2.4× faster than real-time at 200 MHz and 1.1 V with power consumption of 168 mW. By increasing the beam width, better accuracy (91.45%) can be achieved. In that case, the power consumption for real-time processing is increased to 97.4 mW and the max-performance is decreased to2.08x. because of the increased computation workload.

Chapter 6 3x-Real-Time 60-KWord Continuous Speech Recognition VLSI Processor

This chapter describes the third chip (HMM3) [21, 22] we implemented for 60-kWord real-time continuous speech recognition. As the “on-chip” latency (stop processing until the data come to the compute unit) and the “off-chip” latency (loading latency caused by DRAM) strongly delay the processing speed. We try to reduce the “off-chip latency” by implementing a two-level cache architecture and propose a 8-path Viterbi transition unit to reduce the “on-chip latency”, which maximize the processing speed. Furthermore, we adopts tri-gram language model to improve the recognition accuracy. The following techniques are explained in detail.

1) Level-2 Cache
2) 8-path Viterbi transition unit
3) Simplified 3-gram transition

6.1 Speech Recognition System

The overall speech recognition system architecture is depicted in Fig. 6.1, The MFCC feature vectors are extracted using a PC, we separate the feature extraction from the chip because the use of fixed-point computation in the feature extraction part would cause big degradation in recognition accuracy and the computation workload for feature extraction is small thus can be easily handled by PC or an embedded soft-core. The input speech data can either be recorded as an audio stream or with real-time speaking. Before start working, the language and acoustic models are transferred from PC to SDRAM through USB to construct the database. The test chip accesses the DRAM through an on-board FPGA. The data-path of the SDRAM is 64 bit, The data-path for GMM computation (32pin) and Viterbi search (32pin) is separated to support pipeline operation. We implement a 20-frame parallel architecture for GMM computation to support 3× real-time processing (decided by Viterbi). In the previous chip HMM2, we suffered from the pin limitation that only 16 data-pin are available for GMM part. We optimize the pin-placement in this chip by sharing the output pin with Viterbi part because we don’t need to output result in GMM computation except the initial test. Therefore the available data-pin for GMM is increased to 32 which is

enough to support the required speed . There are two GMM result buffer to support the GMM-Viterbi pipeline operation, each of the result buffers will be accessed by GMM core and Viterbi core respectively.

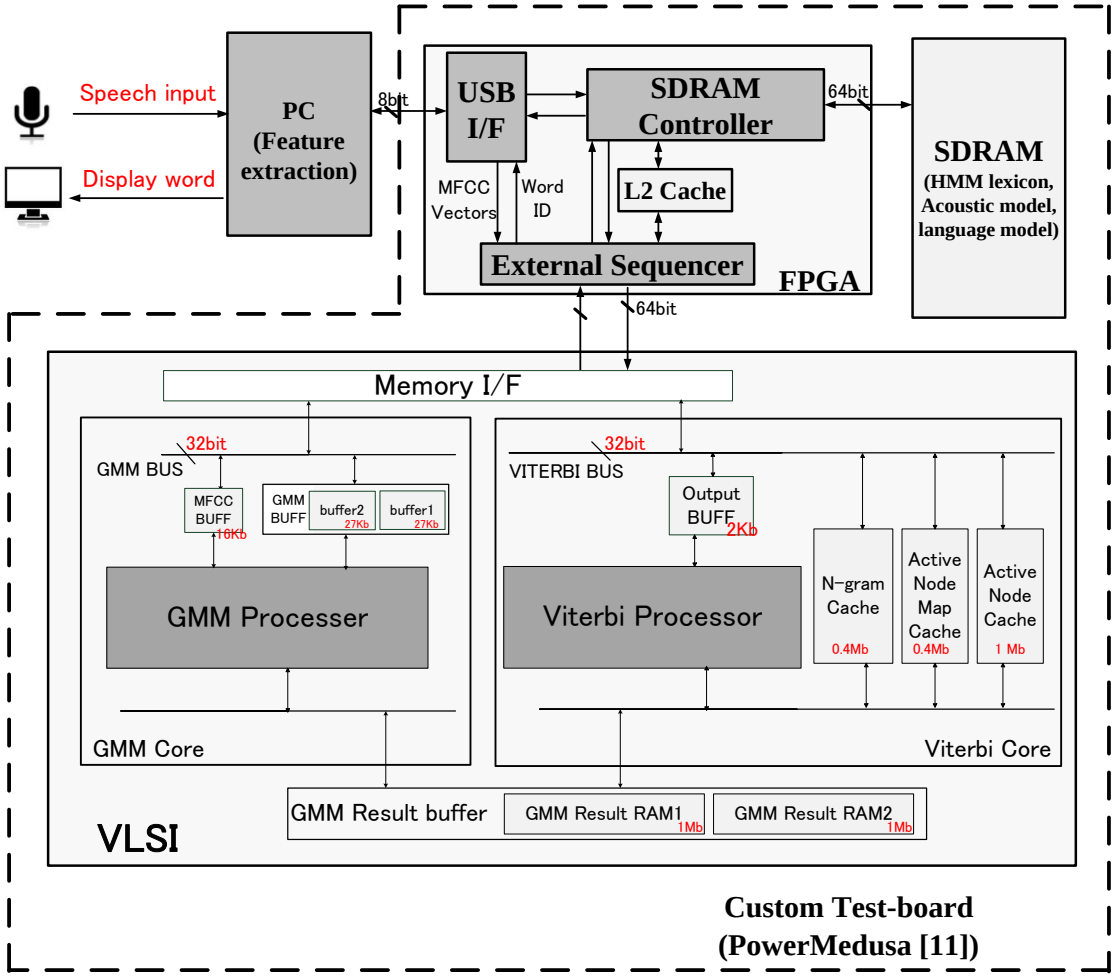

Fig. 6.1 Overall speech recognition system architecture.

6.2 Level-2 Cache

A two-level cache architecture is implemented to reduce the “off-chip latency” for accessing SDRAM. The Level-1 cache is the specialized caches we proposed in HMM1 which are implemented inside the chip and can offer a high hit-rate of 75%. However, when miss-hit occurs, the chip has to access the external SDRAM which causes long latency. In Image and Vidio processing system, the DRAM only acts as a buffer between camera and chip, the pixels are read from DRAM orderly and saved to the on-chip memory. Therefore the max length of burst mode for DRAM access can be utilized. However, in LVCSR system, the DRAM functions as a data-base which saves

the dictionary parameters and language models. These data will be accessed randomly during the processing. According to the characteristics of DRAM , there are several cycles of latency caused by pre-charge every time before we read from the DRAM, therefore if the required data are not accessed sequentially, the efficiency is bad.

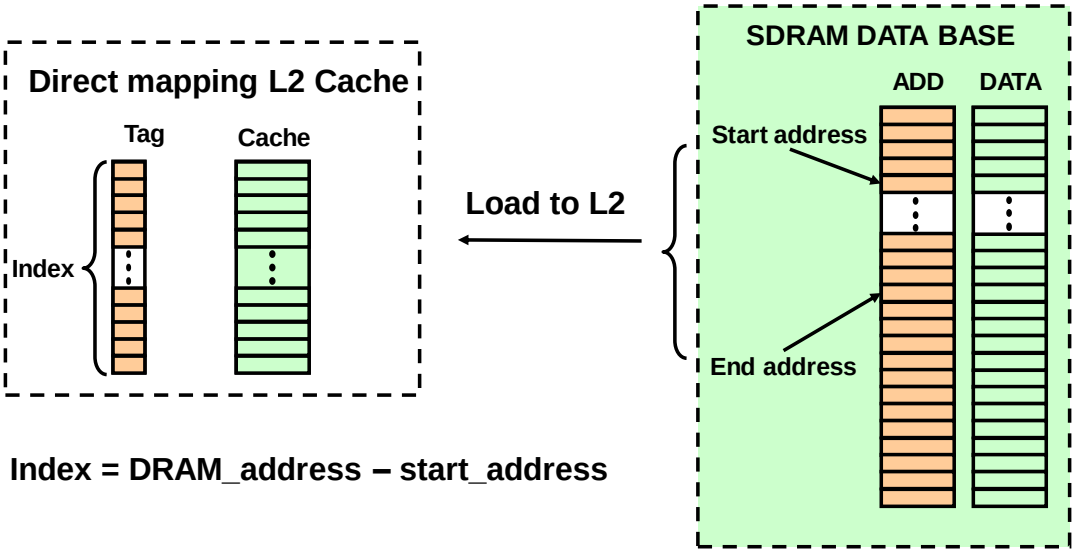

Fig. 6.2 Level-2 Cache architecture.

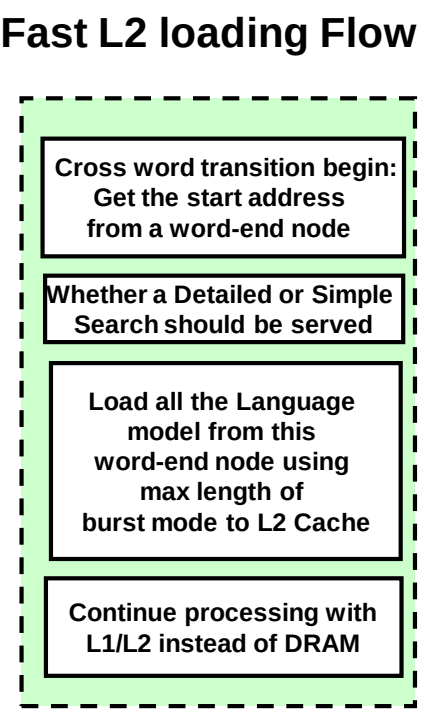

Fig. 6.3 L2 cache data loading flow .

As described herein, a Level-2 cache is created in the host FPGA as shown in Fig.

6.1 . The possible required data are loaded to the L2 Cache during processing as shown in Fig. 6.2. If the required data is found in L1 cache, it will not be transferred to the chip. When the required data is not saved in L1 cache, There's no need to access the SDRAM because the data can be read immediately from the L2 cache. The data loading flow is described in Fig. 6.3. This scheme is effective for many kind of data for speech recognition include the N-gram model for cross-word transition and destination node information for internal-word transition, which account for a big amount of DRAM access.

6.3 8-Path Viterbi Transition Unit

The architecture of Viterbi core is shown in Fig. 6.4. It comprises two active node workspace, an output buffer, a threshold calculator, trellis & token write module and the specialized cache consist of N-gram cache and active node map cache. During transition processing, the active node information and the GMM probabilities can be read from the on-chip memory immediately without miss-hit. However, the N-gram data and active node map data may not be found in the caches. Even the hit-rate of the proposed cache reach 75%, the miss-hit still cause big latency because the Viterbi processing will have to stop to wait until the required data is read from the external data-base, which strongly delays the Viterbi processing. Increasing the number of transition-path can hide the miss-hit latency, which improve the processing speed of the recognition system because the computation time for Viterbi transition is the bottle neck of the GMM-Viterbi pipeline operation. To maximize the utilization ratio of the internal data-path to caches and the external data-path to the SDRAM data-base, we analyze the tradeoff between the number of gates, the number of transition-path and processing speed at 200MHz as presented in Fig. 6.5. 200MHz is the maximum operating frequency of the test chip. Few speed improvement can be achieved while increasing the number of paths from 8 to 10. This means eight paths is enough to process most of the transitions in pipeline, as caches and external data-base has already been occupied, the extra paths will be in the waiting state most of the time during processing. Consquently, We choose to implement a 8-path Viterbi transition architecture, which offers a processing speed of 3.02×.

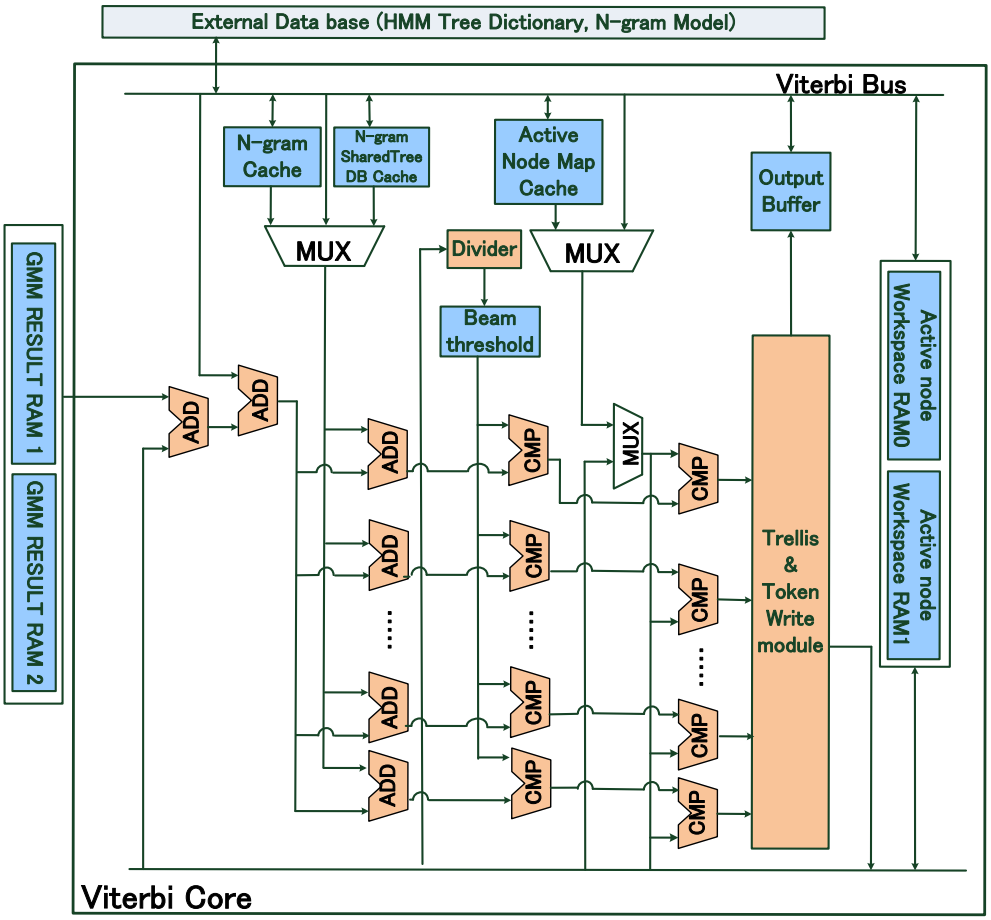

Fig. 6.4 8-path Viterbi transition architecture.

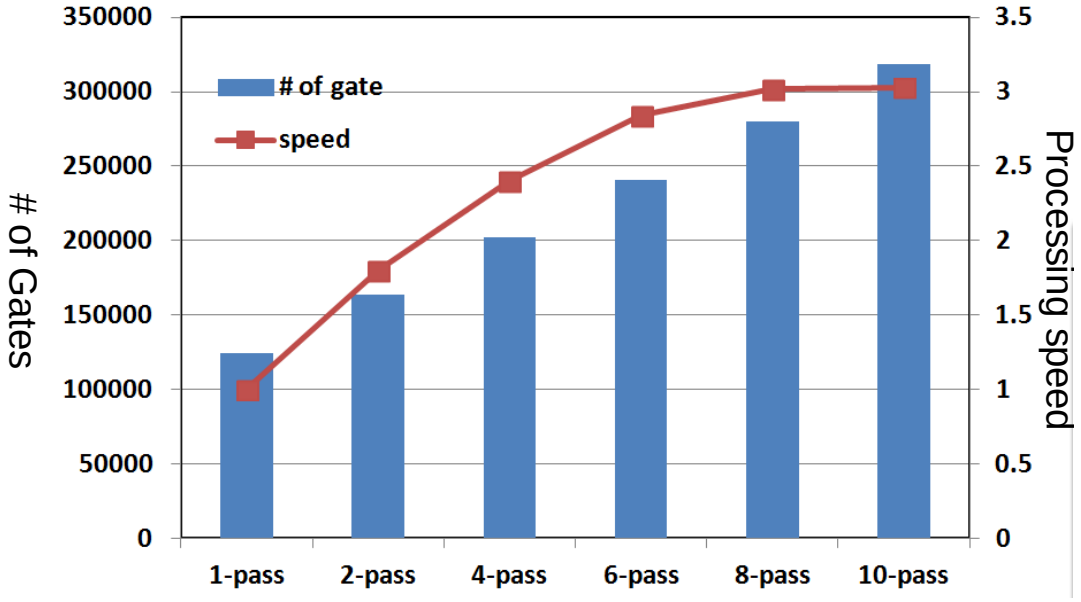

Fig. 6.5 # of paths versus # of logic elements versus processing speed at 200MHz.

6.4 Simplified 3-gram Transition

Both bigram and trigram are available in this chip. The 60k-word bigram language model consists of 60,001 unigram and 4,000,273 bigram transitions while the trigram language model (3-gram) consists of 60,001 unigram, 2,420 231 bigram and 8,368,507 trigram transitions. To adopt trigram transition, we need to treat much more transitions and remain a large network for the history of the two preceding words as shown in Fig. 6.6, which is too costly. Therefore we utilize a simplified trigram transition [11] which only consider the best predecessor word. After bigram transition is applied, the best predecessor word can be decided, then the trigram transition from this word is treated. In case of Fig. 6.6, after treating the bi-gram transitions, word B1 is chosen as the best word. Therefore only the tri-gram transitions from B1 is applied, the other tri-gram are not considered. The recognition accuracy for our test patterns is improved by 1.4% when the trigram restrict is added. In that case, the processing speed is decreased to 2.25× as shown in Fig. 6.7.

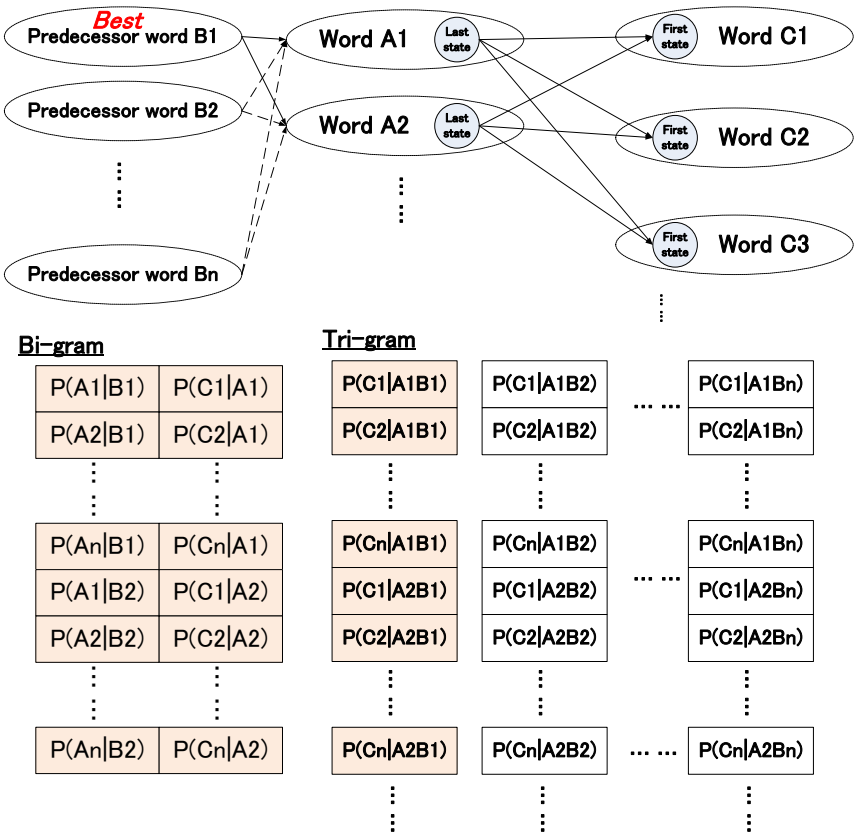

Fig. 6.6 Cross-word transition using bi-gram and tri-gram.

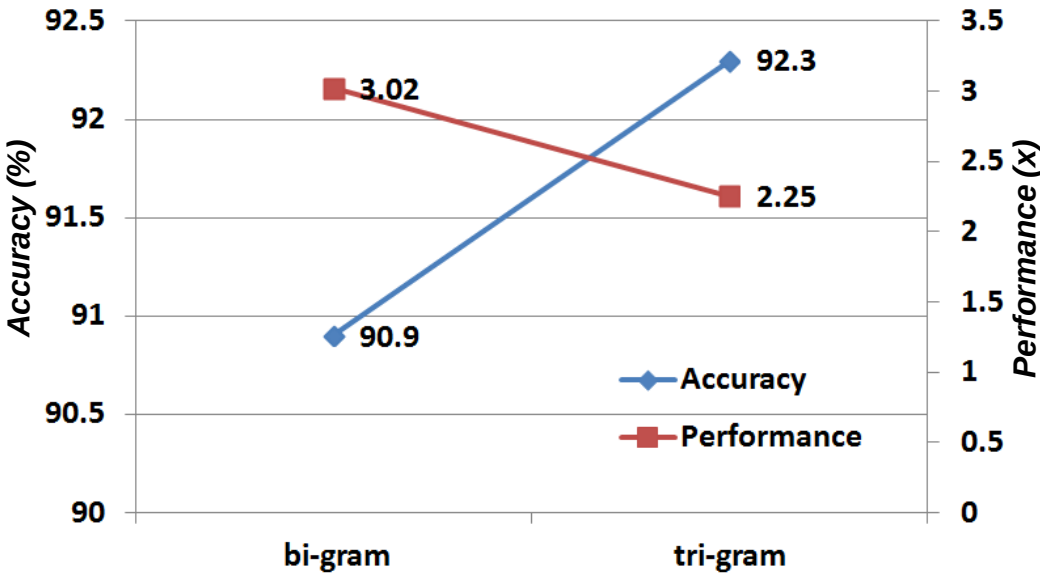

Fig. 6.7 Performance and recognition accuracy for bi-gram and tri-gram (Test speech pattern: 172Japanese sentences).

6.5 Implementation

We implement a software prototype profiling with Microsoft Visual C++ and a referential hardware using hardware description language (HDL) to check the required memory bandwidth and operating frequency for real-time operation. The required frequency reduction for real-time processing is reduced by 88.9% compared to the base-line system and 25% compared to our previous work HMM2 as shown in Fig. 6.8. The layout of the chip, which was fabricated in 40 nm CMOS technology is shown in Fig. 6.9. It occupies 1.77×2.18 mm^2 containing 2.98 M transistors for Logic and 4.29 Mbit on-chip SRAM. The logic part is placed in the center area and the cache memory is placed around. We evaluated the test chip with a logic tester. The generated Shmoo plot is presented in Fig. 6.10. The green area of the Shmoo plot shows the available frequency and operation voltage with which the chip can function correctly. 200MHz is the maximum operation frequency of the test chip under the standard operating voltage (1.1V).

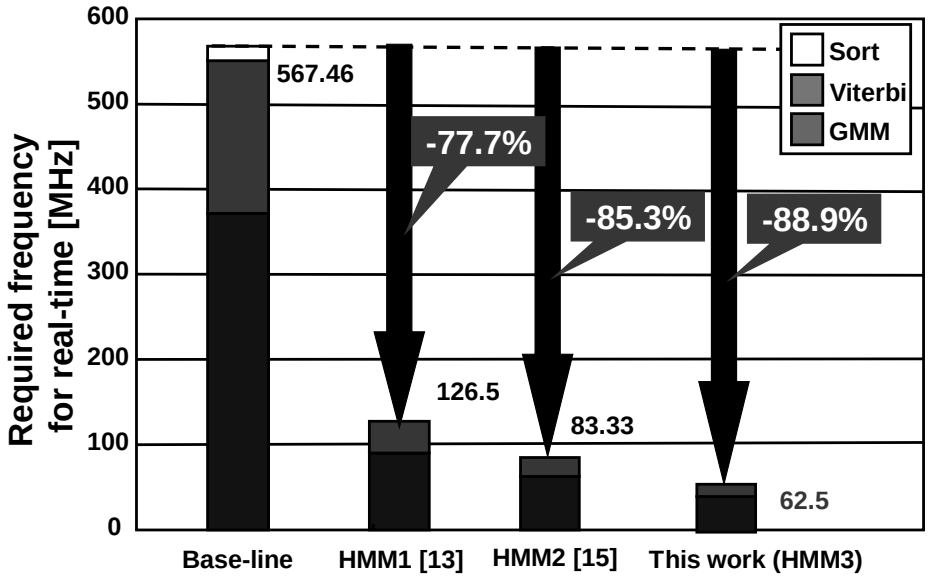

Fig. 6.8 Required frequency reduction for real-time 60k-word continuous speech recognition.

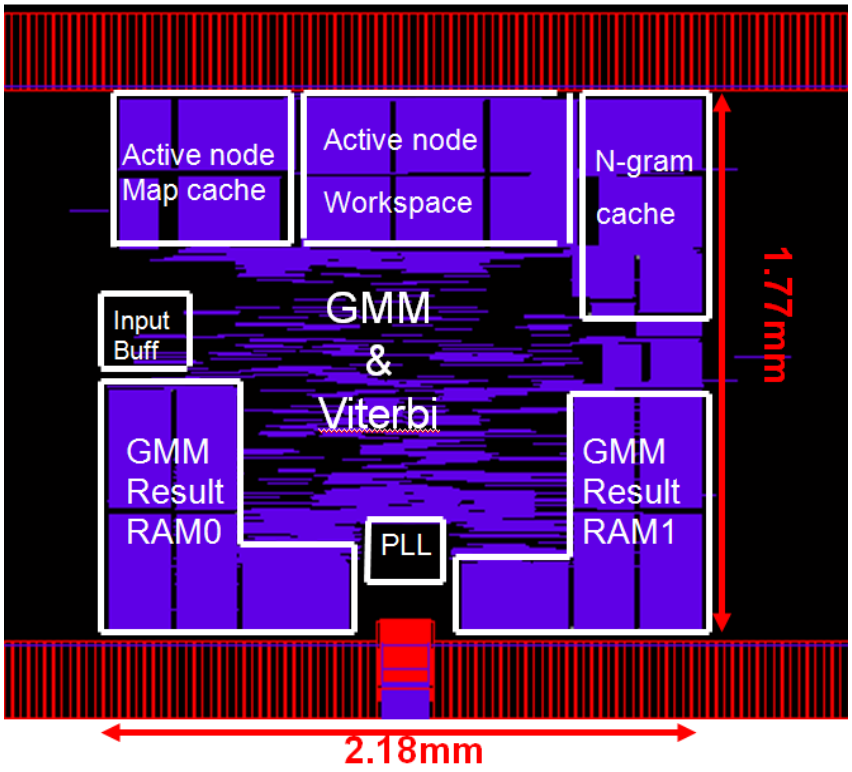

Fig. 6.9 Chip layout.

Table 6.1 Summary of chip implementation.

Process Technology		40-nm CMOS
Core area		2.18 mm × 1.77 mm
Chip area		5 mm × 2.5 mm
Transister Count (Logic)	GMM	1.58 M
	Viterbi	0.89 M
	Other	0.51 M
	Total	2.98 M
On-Chip Memory (SRAM)		4.29 Mbit
Supply voltage		1.1V
I/O voltage		3.3V
Evaluation environment		SOC Tester System
Operating Frequency		62.5 MHz for 60 kWord real-time processing
Measured Power (without IO)		54.8 mW
Leakage current		1.32 mA

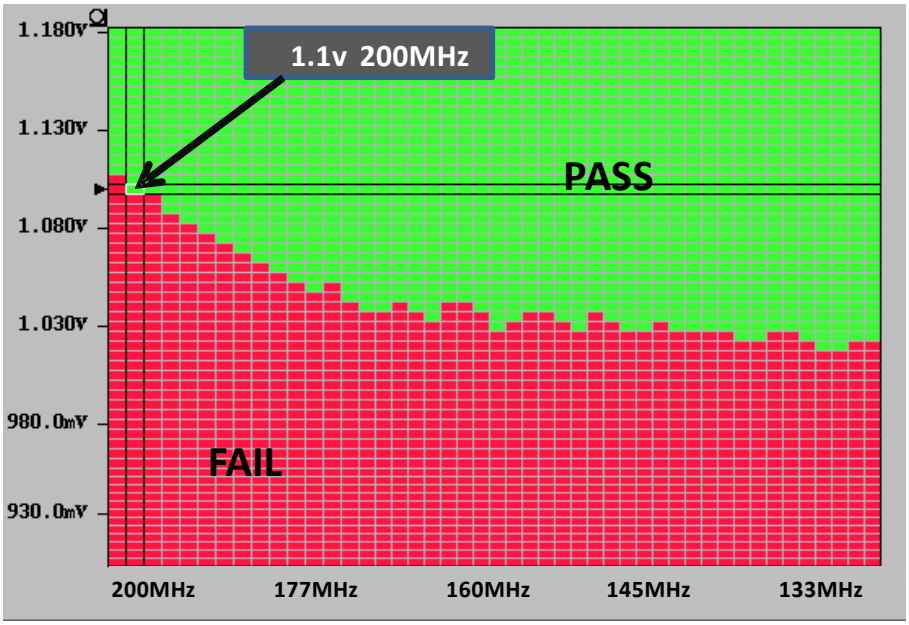

Fig. 6.10 Shmoo plot generated by a logic tester.

Processing speed versus required frequency and the measured power are presented in Fig. 6.11 and Fig. 6.12. This chip can process real-time 60-kWord continuous speech recognition with bi-gram model at 62.5 MHz while consumes 54.8 mW and maximally function 3.02× faster than real-time at 200MHz while consumes 177.4 mW. 26% power consumption reduction for real-time processing is achieved compared to HMM2. When tri-gram is used, HMM3 can process real-time operation at 88.9 MHz

with power consumption of 76.7 mW and maximally function 2.25× faster than real-time at 200 MHz with power consumption of 165 mW. Table 1 presents a comparison between this chip and some recently announced works in terms of the vocabulary size, GMM model, language model, beam-width, real-time factor, operation frequency, external memory bandwidth , area , logic element and power consumption. Figure 6.13 plots the vocabulary versus speed of the state of art hardware-based recognizers.

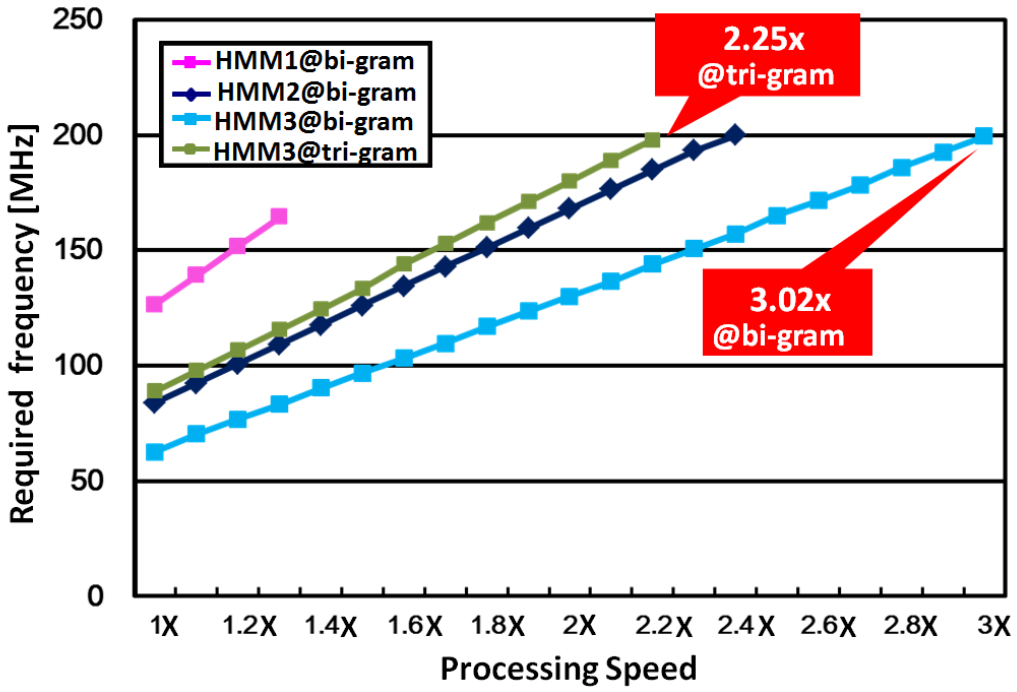

Fig. 6.11 processing speed versus frequency.

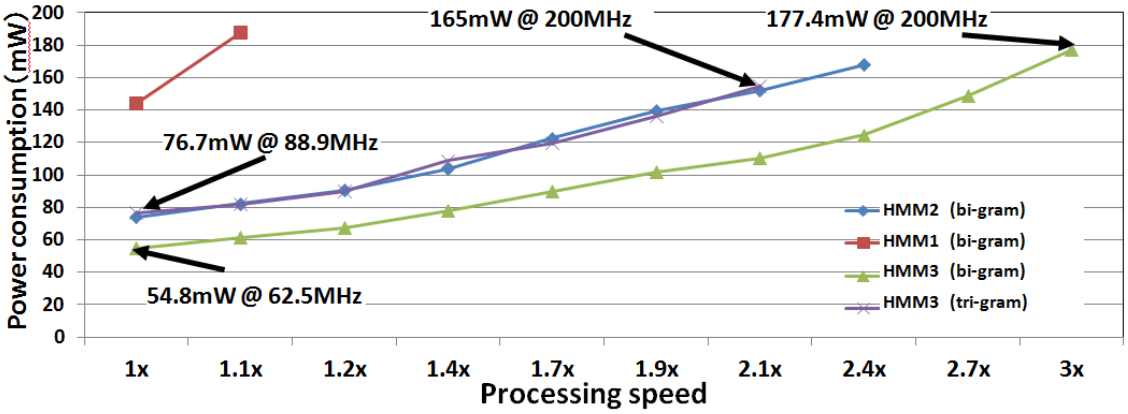

Fig. 6.12 Measured data of processing speed versus power consumption.

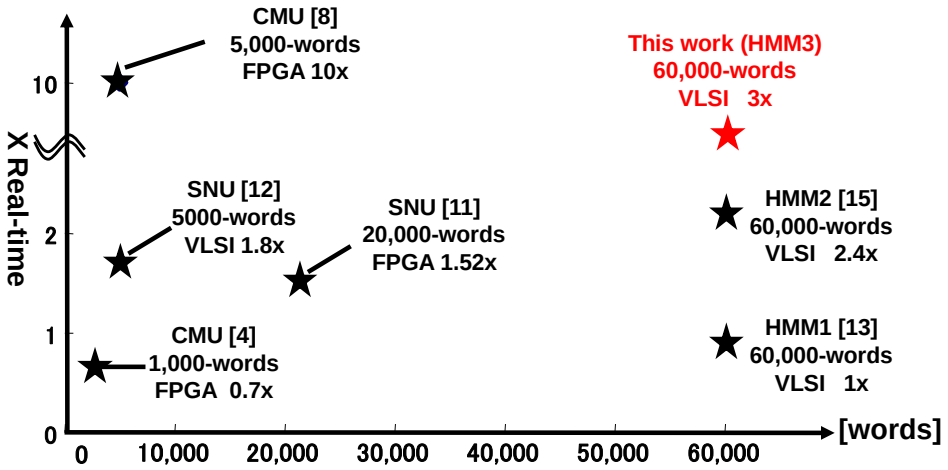

Fig. 6.13 Measured data of processing speed versus power consumption.

Table 6.2 Comparison with recently reported works.

		HMM1 [13]	HMM2 [15]		This work			
Vocabulary (k)		60	60		60			
Technology		VLSI (40 nm)	VLSI (40 nm)		VLSI (40 nm)			
GMM Modu	# of states	2,000	2,000		2,000			
	# of distributions	16	16		16			
	# of dimensions	25	25		25			
LM	# of unigram	60,001	60,001		60,001		60,001	
	# of bigram	4,000,273	4,000,273		4,000,273		2,420,231	
	# of trigram	NA	NA		NA		8,368,507	
Viterbi beam width		3000	3000		3000	3000	3000	
Real-time factor		1	1	2.4	1	3.02	1	2.25
Internal Frequency (MHz)		126.5	83.3	200	62.5	200	88.9	200
External memory BW (MB/s)		70.86	82.6	198	82.6	250	117	264
Core area (mm^2)		5.5	3.86		3.86			
Logic elements		1.9 MTr.	2.52 MTr.		2.98 MTr.			
Internal memory (KB)		975	536		536			

6.6 Summary

As described in this chapter, several schemes are proposed to achieve higher processing speed. As the “on-chip” latency (stop processing until the data come to the compute unit) and the “off-chip” latency (loading latency caused by DRAM) strongly delay the processing speed. We try to reduce the “off-chip latency” by implementing a two-level cache architecture and propose a 8-path Viterbi transition unit to reduce the “on-chip latency”, which maximize the processing speed. Furthermore, we adopts

tri-gram language model to improve the recognition accuracy. The measured results show that our implementation achieves 25% required frequency reduction (62.5 MHz) and 26% power consumption reduction (54.8 mW) for 60 k-Word real-time continuous speech recognition compared to the previous work. This chip can maximally process 3.02× and 2.25× times faster than real-time at 200 MHz using the bigram and trigram language models, respectively.

Chapter 7 Conclusion

This dissertation reports VLSI designs for large vocabulary real-time continuous speech recognition based on context-dependent Hidden Markov Model (HMM). As the background of this research area, the objective of this study and an overview of this dissertation are presented in Chapter 1. Then issues related to the VLSI implementation for real-time continuous speech recognition are noted in Chapter 2. The main issues are explained as four parts: 1) large-vocabulary are needed to support a higher word-cover rate. 2) low-power is required for longer working time and lower temperature, 3) high-speed is demanded for a more comfortable human interface, and 4) small area which means less recourse would be cost. In this study, the solutions to these issues are presented in Chapter 3 to Chapter 6 as shown in Table. 7.1.

In Chapter 3, some algorithm optimizations are presented: beam pruning using a dynamic threshold to cut the workspace and memory bandwidth for sort processing; the modified unigram language model which eliminated the updata process for word-internal transitions; two-stage language model searching to reduce cross-word transitions which reduce the memory access greatly. The accuracy degradation of the important parameters in Viterbi computation is strictly discussed. These optimizations are utilized in all the three chips introduced in the following chapters.

Chapter 4 describes the first chip (HMM1) we implemented for 60-kWord real-time continuous speech recognition. It includes a cache architecture using locality of speech recognition, as some of the data that has been used in the current frame may be reused in the following frames, on chip caches are introduced for keeping some of them to reduce the external memory access, the specialized cache architecture achieve a hit-rate of 75%; a highly parallel Gaussian Mixture Model (GMM) architecture based on the mixture level; a variable-50-frame look-ahead scheme; and elastic pipeline operation between the Viterbi transition and GMM processing. 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 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.

Chapter 5 describes the second chip (HMM2) we designed for 60-kWord real-time continuous speech recognition. It features a compression–decoding scheme to reduce the external memory bandwidth for Gaussian Mixture Model (GMM) computation and a 4-path Viterbi transition units. We optimize the internal SRAM size using the max-approximation GMM calculation and adjusting the number of look-ahead frames. The test chip, fabricated in 40 nm CMOS technology, occupies 1.77 mm × 2.18 mm containing 2.52 M transistors for logic and 4.29 Mbit on-chip memory. The measured results show that our implementation achieves 34.2% required frequency reduction (83.3 MHz), 48.5% power consumption reduction (74.14 mW) for 60 k-Word real-time continuous speech recognition compared to the previous work while 30% of the area is saved with recognition accuracy of 90.9%. It can maximally process 2.4× faster than real-time at 200 MHz and 1.1 V with power consumption of 168 mW.

Chapter 6 describes the third chip (HMM3) we designed for 60-kWord real-time continuous speech recognition. It includes a 8-path Viterbi transition architecture to maximize the processing speed by cutting the on-chip latency and adopts tri-gram language model to improve the recognition accuracy. A two-level cache architecture is implemented for the demo system to cut the off-chip latency. Measured results show that our implementation achieves 25% required frequency reduction (62.5 MHz) and 26% power consumption reduction (54.8 mW) for 60 k-Word real-time continuous speech recognition compared to the previous work. The chip can maximally process 3.02× and 2.25× times faster than real-time at 200 MHz using the bigram and trigram language models, respectively.

The conclusion of this study is presented in this Chapter. This thesis presents the VLSI designs for low-power speak-independent large vocabulary real-time continuous speech recognition based on context-dependent Hidden Markov Model (HMM). The work contributes to achieve a comfortable human interface for mobile systems and robotics.

Table 7.1 Mapping of the proposed techniques in Chapter 3-6 for the four issues.

Proposed techniques	Issue of Large Vocabulary	Issue of Low Power	Issue of High Speed	Issue of Small Area
3.2 Beam Pruning	○	○		
3.3 Modified Unigram	○	○		
3.5 Two-stage LM Search	○	○		
4.1 50-frame Look Ahead scheme	○	○		
4.2 Elastic Pipeline Operation	○	○		
4.3 Highly Parallel GMM Architecture	○	○		
4.4 Viterbi Cache Architecture	○	○		
5.2 Compression-Decoding scheme				○
5.3 Max-Mixture GMM Calculation		○		○
5.4 4-path Viterbi Transition unit		○	○	
6.2 Level-2 Cache		○	○	
6.3 8-path Viterbi Transition unit		○	○	

References

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

[2] K. Yu, and R. Rutenbar, "Profiling Large-Vocabulary Continuous Speech Recognition on Embedded Device: A Hardware Resource Sensitivity Analysis," in *Proc. 10th Conf. Interspeech*, Brighton, UK, pp. 995-998, Sep. 2009.

[3] E. C. Lin, K. Yu., R. Rutenbar, and T. Chen, "In silico Vox: Towards Speech Recognition in Silicon" in "*HOTCHIPS"*, August, 2006.

[4] E. C. Lin, K. Yu. R. 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, Feb. 2007.

[5] U. Pazhayaveetil, D. Chandra, and P. Franzon, "Flexible low power probability density estimation unit for speech recognition," in *Proc. IEEE ISCAS*, Honolulu, HI, 2007, pp. 1117-1120.

[6] B. Mathew, A. Davis, and Z. Fang, " A low-power accelerator for the SPHINX 3 speech recognition system," in *Proc. Int. Conf. CASES*, San Jose, CA, 2003, pp. 210-219.

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

[8] E. C. Lin, and R. A. Rutenbar, "A Multi-FPGA 10x-Real-Time High-Speed Search Engine for a 5000-Word Vocabulary Speech Recognizer," in *Proc. 17th ACM/SIGDA Intl. Symp FPGA*, Montery, CA, pp.83-92, Feb. 2009.

[9] 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

[10] 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.

[11] 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.

[12] K. You, Y.Choi, J. Choi, and W. Sung, "Memory Access Optimized VLSI for 5000-Word Speech Recognition," *JOURNAL OF SIGNAL PROCESSING SYSTEMS*, vol.63, no. 1, pp. 95-105, Apr. 2011.

[13] G. He, T. Sugahara, T. Fujinaga, 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 Integrated Circuits Conference (CICC),* pp.1-4 Sep. 2011.

[14] G. He, T. Sugahara, T. Fujinaga, 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," *IEEE Trans. Circuits Syst. I, Reg. Papers,* vol. 59, no. 8, pp.1656-1666, Aug. 2012.

[15] G. He, T. Sugahara,Y. Miyamoto, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A 40-nm 168-mW 2.4×-Real-Time VLSI Processor for 60-kWord Continuous Speech Recognition," in *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, Sep. 2012.

[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, pp. 132-133, Oct. 1996.

[19] Y. Choi, K. You, J. Choi, and W. Sung, "VLSI for 5000-word continuouse speech recognition," in *Proce. IEEE ICASSP*, Taipei, Taiwan, Apr. 2009, pp.

557-560

[20] G. He, T. Sugahara, Y. Miyamoto, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A 168-mW 2.4x-Real-Time 60-kWord Continuous Speech Recongnition Processor VLSI," *IEICE Transactions on Electronics,* Vol. E96-C, No. 4, pp. 444-453, Apr. 2013.

[21] G. He, Y. Miyamoto, K. Matsuda, S. Izumi, H. Kawaguchi, M. Yoshimoto, " A 40-nm 54-mW 3×-Real-time VLSI Processor for 60-KWORD Continuous Speech Recognition, " *Proceedings of IEEE Workshop on Signal Processing Systems (SiPS)*, pp.147-152, Oct. 2013.

[22] G. He, Y. Miyamoto, K. Matsuda, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A 54-mW 3x-Real-Time 60-kWord Continuous Speech Recongnition Processor VLSI," *IEICE Electronics Express*, vol. 11, no.2, Jan. 2014.

[23] 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.

[24] 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.

[25] "Power medusa custom test board," MMS, Amagasaki, Japan [Online] Available: http://www.mms.co.jp/powermedusa/concept/index.htl.

[26] T.W. Kohler, C. Fugen, S. Stuker and A. Waibel, "Rapid porting of ASR-systems to mobile devices," in *Proc. Interspeech,* pp. 233-236. Sep. 2005.

[27] D. Huggins-Daines, K. Mohit, C. Arthur, "Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices," in *Proc. ICASSP,* pp. 185-188. May. 2006.

List of Publications and Presentations

Publications in journals and transactions

1) G. He, Y. Miyamoto, K. Matsuda, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A 54-mW 3x-Real-Time 60-kWord Continuous Speech Recongnition Processor VLSI," *IEICE Electronics Express*, vol. 11, no.2, Jan. 2014.
2) G. He, T. Sugahara, Y. Miyamoto, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A 168-mW 2.4x-Real-Time 60-kWord Continuous Speech Recongnition Processor VLSI," *IEICE Transactions on Electronics*, vol. E96-C, no. 4, pp. 444-453, Apr. 2013.
3) G. He, T. Sugahara, T. Fujinaga, 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," *IEEE Transaction on Circuits and Systems I, Regular Papers*, vol. 59, no. 8, pp.1656-1666, Aug. 2012.
4) K. Mizuno, H. Noguchi, G. He, Y. Terachi, T. Kamino, T. Fujinaga, S. Izumi, Y. Ariki, H. Kawaguchi and M. Yoshimoto, "A Low-power Real-Time SIFT Descriptor Generation Engine for Full-HDTV Video Recognition," *IEICE Transaction on Electronics*, vol. E94-C, no. 4, pp. 448-457, Apr. 2011.

Presentations at international conferences

1) G. He, Y. Miyamoto, K. Matsuda, S. Izumi, H. Kawaguchi, M. Yoshimoto, " A 40-nm 54-mW 3×-Real-time VLSI Processor for 60-KWORD Continuous Speech Recognition, " *Proceedings of IEEE Workshop on Signal Processing Systems (SiPS)*, pp.147-152, Oct. 2013.
2) G. He, T. Sugahara, T. Fujinaga, Y. Miyamoto, H. Noguchi, S. Izumi, H. Kawaguchi and M. Yoshimoto, "A 40-nm 144-mW VLSI Processor for Realtime 60k Word Continuous Speech Recognition," *Proceedings of IEEE Asia and South Pacific Design Automation Conference (ASP-DAC) University LSI Design Contest*, pp. 71-72, Jan. 2013. **(University LSI Design Contest Best Design Award**)
3) G. He, T. Sugahara, Y. Miyamoto, S. Izumi, H. Kawaguchi, and M. Yoshimoto,

"A 40-nm 168-mW 2.4×-Real-Time VLSI Processor for 60-kWord Continuous Speech Recognition," *Proceedings of IEEE Custom Integrated Circuits Conference(CICC)*, Sep. 2012.

4) G. He, T. Sugahara, T. Fujinaga, Y. Miyamoto, H. Noguchi, S. Izumi, H. Kawaguchi, M. Yoshimoto, "A 40 nm 144 mW VLSI Processor for Realtime 60 kWord Continuous Speech Recognition," *Proceedings of IEEE Custom Integrated Circuits Conference (CICC)*, Sep. 2011.
5) K. Mizuno, H. Noguchi, G. He, Y. Terachi, T. Kamino, H. Kawaguchi and M. Yoshimoto, "Fast and Low-Memory-Bandwidth Architecture of SIFT Descriptor Generation with Scalability on Speed and Accuracy for VGA Video," *Proceedings of 20th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 608-611,Milano, ITALY, Aug. 2010.

Presentations at domestic conferences

1) 何　光霽, 宮本優貴, 松田 薫平, 和泉慎太郎, 川口　博, 吉本雅彦 "3 倍速実時間 6 万語彙連続音声認識のための 40-nm, 54-mW 音声認識専用プロセッサ" 信学技報, vol. 113, no. 235, ICD2013-76, pp. 29-34, 2013 年 10 月, 弘前.
2) 何　光霽, “6 万語彙実時間連続音声認識のための 40nm, 144mW 音声認識専用プロセッサ” VDEC デザイナーズフォーラム, 2013 年 8 月.（**VDEC デザインアワード優秀賞**）
3) 宮本優貴, 何　光霽, 和泉慎太郎, 川口　博, 吉本雅彦 "2.4 倍速実時間 6 万語彙連続音声認識プロセッサの開発" 信学技報, vol. 112, no. 365, ICD2012-101, pp. 49-53, 東京, 2012 年 12 月.
4) 何光霽, 菅原隆伸, 藤永剛史, 宮本優貴, 野口紘希, 和泉慎太郎, 川口博, 吉本雅彦, "6 万語彙実時間連続音声認識のための 40nm, 144mW 音声認識専用プロセッサの開発," LSI とシステムのワークショップ 2012, pp. 189-191, 北九州市, 2012 年 5 月.
5) 菅原 隆伸, 何 光霽, 藤永 剛史, 宮本 優貴, 野口 紘希, 和泉 慎太郎, 川口 博, 吉本 雅彦, ”6 万語彙実時間連続音声認識のための 40nm,144mW 音声認識専用プロセッサの開発," 信学技報, vol. 111, no. 327, ICD2011-96,

pp. 79-84, 2011 年 11 月.

6) 寺地 陽祐，水野 孝祐，何 光霽，川口 博，吉本 雅彦，"フル HDTV 実時間動画像認識のための低消費電力 SIFT 特徴量抽出プロセッサ," LSI とシステムのワークショップ 2011, pp.209-211, 北九州市, 2011年5月.（**ICD 最優秀ポスター受賞**）

Acknowledgement

I would like to express my deepest appreciation to Professor Masahiko Yoshimoto of Kobe University for offering me the great chance to study in this laboratory, his invaluable suggestions, appropriate guidance and enthusiastic encouragement helped me having a wonderful research life in Japan. I am grateful as well to Associate Professor Hiroshi Kawaguchi of Kobe University for providing me constructive comments and warm encouragement related to this research. My deepest appreciation goes to them.

I also would like to thank Professor Makoto Nagata, Professor Yasuo Ariki and Professor Osamu Matoba for giving me helpful suggestions for this dissertation.

I would like to express my deepest gratitude to my colleagues of the hyper project: Dr. Hiroki Noguchi, Dr. Kosuke Mizuno, Mr. Tetsuya Kamino, Mr. Junichi Tani, Mr. Kazuo Miura, Mr. Mitsuhiko Kuroda, Mr. Yusuke Shimai, Mr. Tomoya Takagi, Mr. Tsuyoshi Fujinaga, Mr. Koji Kugata, Mr. Takanobu Sugahara, Mr. Yosuke Terachi, Mr. Masanori Nishino, Mr. Yuki Miyamoto, Mr. Kenta Takagi, Mr. Kumpei Matsuda, Mr. Kotaro Tanaka. I owe my deepest gratitude to Assistant Professor Shintaro Izumi for giving me advices in many aspects of this research. I would like to thank Dr. Shusuke Yoshimoto for spending time in the laboratory for 5 years and giving me many help.

During the time I spent in the laboratory, I was fortune to meet so many nice members, with whom I spent a great time. I would like to thank all of them: Dr. Hidehiro Fujiwara, Dr. Shunsuke Okumura, Dr. Keita Konishi, Dr. Yohei Nakata, Dr. Takashi Takeuchi, Dr. Takashi Matsuda, Dr. Takashi Takeuchi, Mr. Hyeokjong Lee, Mr. Yu Otake, Mr. Yusuke Iguchi, Mr. Yoshinori Sakata, Mr. Koh Tsuruda, Mr. Akihisa Oka, Mr. Yukihiro Takeuchi, Mr. Masahiro Yoshikawa, Mr. Kosuke Yamaguchi, Ms. Mari Masuda, Mr. Keisuke Okuno, Mr. Takuro Amashita, Mr. Yuki Kagiyama, Mr. Masaharu Terada, Mr. Jung Jinwook, Mr. Koji Yanagida, Mr. Yusuke Takeuchi, Mr. Shimpei Soda, Mr. Masanao Nakano, Mr. Asuka Fujikawa, Mr. Yohei Umeki, Mr. Yuki Kitahara, Mr. Ken Yamashita, Mr. Yuta Kimi, Mr. Tomoki Nakagawa, Mr. Song Dae-Woo, Mr. Takahide Fujii, Mr. Go Matsukawa, Mr. Yozaburo Nakai, Ms. Kana Masaki Mr. Naoki Okawa, Mr. Yuta Kawamoto, Mr. Haruki Mori and Mr. Shuhei Yoshida. I would like to express my appreciation to Ms. Emi Go, Ms. Keiko Matsuoka, Ms. Aya Tsuboi, and Ms. Yurie Izumi for their kindless.

This work was supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems Inc., Mentor Graphics Corp., and Synopsys Inc. The VLSI chips used in this study was fabricated in the chip fabrication program of VDEC, the University of Tokyo in collaboration with Renesas Electronics. 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”.

I would like to appreciate SENSHUKAI CO. LTD for supply me scholarship during my doctoral course.

Finally, I would like to express my deepest gratitude to my parents, my aunt, my cousin for their plentiful support and encouragement.

Guangji He