Speech Recognition Chip for Monosyllables

Kazuhiro Nakamura Qiang Zhu[†] Shinji Maruoka[‡] Takashi Horiyama Shinji Kimura Katsumasa Watanabe

Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0101 JAPAN
[†]Presently Fujitsu Laboratories Ltd. [‡]Presently Hitachi Ltd.

Abstract— In the paper, we present a real-time speech recognition chip for monosyllables such as A, B, ..., etc. The chip recognizes up to 64 monosyllables based on the Hidden Markov Model (HMM), which is a well known speaker-independent recognition method. The chip accepts a short-speech frame including 256 16-bit digitized samples corresponding to 11.6 msec period, and outputs the 6-bit symbol code of monosyllables for 16 short-frames (corresponding to 185.6 msec). A learning circuit to update HMM parameters for the recognition chip has also been designed, and the recognition chip includes an interface to the learning circuit. We have fabricated the recognition chip by VDEC Rohm 0.6 μm process on a 4.5 mm x 4.5 mm chip. We have also made a layout of the entire circuit including the learning circuit by VDEC Rohm 0.35 μm process on a 4.9 mm x 4.9 mm chip.

I. Introduction

Speech recognition has become one of popular human interfaces, and many speech recognition softwares and consumer products (such as cellular phones and car navigation systems) with speech recognition function have been developed[1, 2, 3]. To increase the use of speech recognition in embedded systems, we need a speech recognizer chip with small area, less memory usage, low power and online leaning mechanism.

In this paper we present a real-time speech recognition chip for monosyllables such as A, B, ..., etc. The chip recognizes up to 64 monosyllables based on the Hidden Markov Model (HMM) which is a well known speaker-independent recognition method. The chip accepts a short-speech frame corresponding to 11.6 msec period, and outputs the code of the monosyllable with the highest probability on the HMM. The chip also includes a mechanism to work with a learning circuit to update HMM parameters.

We first implemented the recognition algorithm in C to check the behavior, and we converted the C programs into register-transfer level programs for hardware implementation. In the conversion, floating point operations are changed to fixed-point operations keeping the recognition quality. Based on the algorithm described in C, we have designed the speech recognition chip using VHDL and fabricated a 4.5 mm $\times$ 4.5 mm chip using 0.6 μm process via VDEC Rohm.

II. Speech Recognition Based on Hidden Markov Model (HMM)

We adopt the speaker-independent HMM-based method for our hardware design and adopt monosyllables as a recognition unit. Monosyllables are basic unit of speech and are suitable for a human interface such as a speech-recognizing keyboard (this would not be a keyboard anymore). Monosyllables are known to be hard to recognize, but special hardware modules would have a chance to solve the problem.

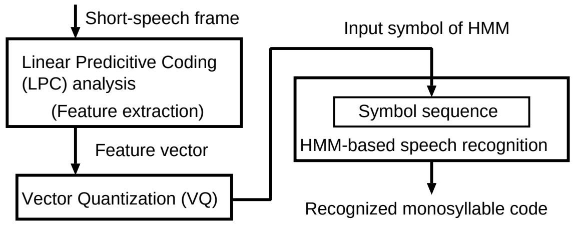

Short-speech frame
Linear Predicitive Coding (LPC) analysis (Feature extraction)
Feature vector
Vector Quantization (VQ)
Input symbol of HMM
Symbol sequence
HMM-based speech recognition
Recognized monosyllable code

Fig. 1. Structure of Our HMM-based Speech Recognition System.

Fig.1 shows the structure of our HMM-based speech recognition system that includes three major tasks: linear predictive coding (LPC) analysis, vector quantization (VQ) and HMM-based speech recognition. In recognition processes, a feature vector is extracted form a short-speech frame by the LPC, and then the feature vector is mapped to an input symbol of HMMs by the VQ. Finally, the code of recognized monosyllable is output after the HMM-based recognition.

III. Speech Recognition Algorithms and Hardware Design

A. Linear Predictive Coding (LPC) Analysis

We use the LPC analysis to extract a feature vector from a short-time speech frame. The length of the short-time speech frame is 11.6 msec, which includes 256 speech samples of 16-bit value. The sampling frequency is 22.05 kHz. The dimension of an extracted feature vector is 24, each element of which has 32-bit value.

A.1. Linear Predictive Coding (LPC) Analysis Algorithm

Let $x_0, x_1, x_2, \cdots, x_{255}$ and $(C_0, C_1, \cdots, C_{23})$ be a sequence of speech samples and that of feature vector respectively. Feature vector is computed based on the following formulae (1), (2) and (3).

First, $r_0, r_1, \cdots, r_{23}$ is obtained by the autocorrelation function (1) with respect to $x_0, x_1, x_2, \cdots, x_{255}$.

$$r_m = \frac{\sum_{n=0}^{255-m} x_n x_{n+m}}{256}, \ (m = 0, 1, \cdots 23) \qquad \cdots (1)$$

Then, LPC coefficients $\alpha_1, \alpha_2, \cdots, \alpha_{23}$ are obtained by solving the following equation.

$$\begin{pmatrix} r_0 & r_1 \cdots r_{22} \\ r_1 & r_0 \cdots r_{21} \\ \vdots & \ddots \ \vdots \\ r_{22} r_{21} & \cdots r_0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{23} \end{pmatrix} = - \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{23} \end{pmatrix} \qquad \cdots (2)$$

Finally, LPC cepstral coefficients $C_0, C_1, \cdots, C_{23}$ (feature vector) are obtained by the following formula.

```
Durbin_method ($r$[])
 $r$[0, ..., 23]:  elements of a Toepliz matrix;
 $\alpha$[1, ..., 23]:  LPC coefficients;
 $e$, $cm$, $k$[], $b$[]:  intermediate results;
begin
 $e$ = $r$[0];
 $k$[0] = - $r$[1] / $e$;  $\alpha$[1] = $k$[0];
 $e$ = (1.0 - $k$[0]$^2$) $\times$ $e$;
 for ($i$ = 1; $i$ < 23; $i$ = $i$+1) {
  $cm$ = $r$[$i$+1];
  for ($j$ = 1; $j \leq i$; $j$ = $j$+1) {
   $cm$ = $cm$ + $\alpha$[$j$] $\times$ $r$[$i$+1$-j$];
  }
  $k$[i] = - $cm$ / $e$;  $\alpha$[$i$+1] = $k$[$i$];
  $e$ = (1.0 $-$ $k$[$i$]$^2$) $\times$ $e$;
  for ($j$ = 1; $j \leq i$+1 ; $j$ = $j$+1) {
   $b$[$j$] = $\alpha$[$i$+1$-j$];
  }
  for ($j$ = 1; $j \leq i$ ; $j$ = $j$+1) {
   $\alpha$[$j$] = $\alpha$[$j$] + $k$[$i$] $\times$ $b$[$j$];
  }
 }
end
```

Fig. 2. Durbin algorithm for computing LPC coefficients.

$$C_n = \begin{cases} \log\left(\frac{\sum_{i=1}^{23}\alpha_i r_i}{2\pi}\right), \ (n=0) \\ -\alpha_1, \ (n=1) \\ -\alpha_n - \sum_{m=1}^{n-1}\left(1-\frac{m}{n}\right)\alpha_m C_{n-m}, \ (2 \leq n \leq 23) \end{cases} \cdots(3)$$

A.2. Hardware Module for LPC Analysis

When 256 speech samples are stored in external memory, the LPC analysis module starts the feature extraction. Then 24-dimensional feature vector is computed and stored in external memory, and a start signal is sent to the VQ module. We used 32-bit value for each element of the feature vector.

Equation (2) is efficiently solved with the Durbin method[3] since the equation is a Toepliz matrix. Fig.2 shows the Durbin algorithm that accepts $r[\ ]$ and outputs $\alpha[\ ]$. Hardware module for LPC analysis computes values of $\alpha[\ ]$ in 10,992 clock cycles based on the algorithm.

The LPC module takes 127,080 clock cycles for the whole computations of LPC analysis with a 32×20-bit combinational adder array multiplier, a 32-bit sequential divider, a 32-bit adder and a 32-bit logarithm calculating circuit. The LPC module also has 462 flip-flops.

B. Vector Quantization (VQ)

B.1. Vector Quantization (VQ) Algorithm

VQ translates a feature vector obtained by LPC analysis into an input symbol of HMMs. In VQ, the VQ codebook which is a table that consists of pairs of symbol index and representative feature vector is used for the translation. A feature vector is translated into a symbol index whose representative feature vector is closest to the feature vector.

B.2. Hardware Module for Vector Quantization (VQ)

The VQ module accepts a feature vector and translates the vector into a symbol. We used 8-bit values as the symbol.

Fig.3 shows a vector quantization algorithm. The size of the VQ codebook is 32-bit$\times$24 (representative feature vector) $\times$ 2^8 (symbol) bits. The algorithm starts from searching *CodeBook*[ ][ ], and if a representative feature vector that is

```
VQ ($C$[], $CodeBook$[][])
 $C$[0, ..., 23]:  a feature vector;
 $CodeBook$[0,...,255][0,...,23]:  VQ codebook;
 $symbol$:  a resulting symbol ;
begin
 $min\_dist$ = $\infty$;
 for ($i$ = 0; $i$ < 256; $i$ = $i$+1) { /* search */
  $tmp\_dist$ = 0.0;
  for ($j$ = 0; $j$ < 24; $j$ = $j$+1) {/* distance */
   $tmp\_dist$ += $|C[j]$ - $CodeBook[i][j]|$;
  }
  if ($tmp\_dist$ < $min\_dist$) {
   $min\_dist$ = $tmp\_dist$; $symbol$ = $i$;
  }
 }
 return $symbol$;
end
```

Fig. 3. A Vector Quantization algorithm.

closer to the input feature vector $C[\ ]$ is found, then the index of the representative feature vector is set to *symbol*. Finally, the index whose representative feature vector is closest to $C[\ ]$ is obtained and used as an 8-bit symbol. 61,952 clock cycles are used to map a feature vector to an 8-bit symbol.

C. Hidden Markov Model (HMM)

Monosyllables are recognized using Hidden Markov Model (HMM), and one HMM is constructed for each monosyllable. We show a definition of an HMM.

Definition 1 An HMM M is a 6-tuple $(S, \Sigma, A, B, \Pi, F)$ where

- S is a finite set of states,
- Σ is an input alphabet,
- $A : S\times S \to \mathbf{R}$ is a state transition probability function, where $\mathbf{R}$ is the real number set, $\sum_j A(q_i,\ q_j) = 1$,
- $B : S\times S\times\Sigma \to \mathbf{R}$ is an output probability function, $\sum_k B(q_i,\ q_j,\ k) = 1$,
- $\Pi = \{\pi_{q_i} \mid$ probability that $q_i \in S$ is an initial state $\sum_i \pi_{q_i} = 1\}$ is a set of initial probabilities,
- $F(\subseteq S)$ is a finite set of final states.

In the following, we describe $A(q_i,\ q_j)$ and $B(q_i,\ q_j, k)$ as $a_{q_i q_j}$ and $b_{q_i,q_j}(k)$ respectively. The behavior sequences of M with respect to an input sequence $X = x_1x_2\ldots x_N$ $(x_i \in \Sigma)$ are sequences whose length is N, and each sequence starts from a state whose initial state probability is not 0 and ends a final state. Let $Q_j = q_0^{(j)}q_1^{(j)}\cdots q_N^{(j)}$ $(q_N^{(j)} \in F)$ be jth sequence. Probability $P(x_1x_2\cdots x_N \mid M)$ that X is accepted by M is obtained by the following formula:

$$P(x_1x_2\cdots x_N \mid M) = \sum_{Q_j}\pi_{q_0^{(j)}}\prod_{i=1}^{N} a_{q_{i-1}^{(j)},q_i^{(j)}} b_{q_{i-1}^{(j)},q_i^{(j)}}(x_i).$$

D. Speech Recognition Based on Hidden Markov Model

In the following, 64 HMMs are used to recognize 64 monosyllables. We use 5-state left-right HMMs based on [3], which have only one-way state transition as shown in Fig.4. HMMs accept 16 input symbols corresponding to 16 short-speech frames (185.6 msec speech) and decide the monosyllable.

D.1. Recognition Algorithm

The acceptance probability $P(X|M_h)$ $(h = 1, 2, ..., 64)$ with respect to 16 symbols $X = x_1x_2\ldots x_{16}$ is computed for all HMMs. Then the monosyllable whose HMM has the greatest probability is recognized.

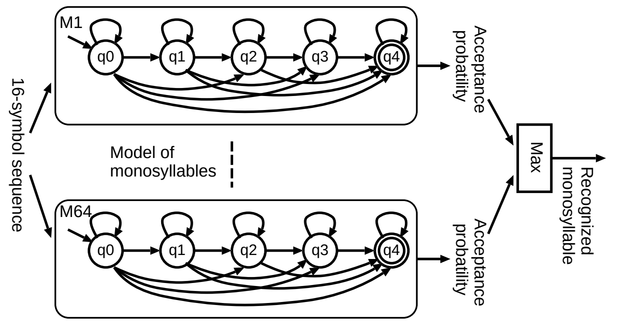


Fig. 4. HMM of monosyllables and HMM-based speech recognition.

Since hardware cost of a multiplier is expensive, $\log(\max_Q P(X|M_h))$ is used instead of the probability $P(X|M_h)$. $\max_Q P(X|M_h)$ is the probability of a behavior sequence with the highest probability. $\alpha'_{q_i,n}$ is the probability to stay the state q_i after the n-th symbol.

$$\alpha'_{q_i,n} = \begin{cases} \log\pi_{q_i},\ (n=0,\ \pi_{q_i}\neq 0) \\ -\infty,\ (n=0,\ \pi_{q_i}=0) \\ \max\limits_{q_j\in S,\ a_{q_j,q_i}\neq 0}\left(\alpha'_{q_j,n-1}+\log\left(a_{q_j,q_i}\cdot b_{q_j,q_i}(x_n)\right)\right) \end{cases} \cdots(4)$$

$$\log(\max_Q P(X|M_h)) = \max_{q\in F}\alpha'_{q,16}$$

D.2. Hardware Module for Recognition

The HMM module starts recognition when 16 symbols are stored in a register array X (8-bit$\times$16). The X is used as a queue. When a new symbol arrives at the HMM module, the symbol is put into X and the oldest symbol is removed from X. In the recognition process, 64 acceptance probabilities $P(X|M_h)$ $(h=1,2,...,64)$ are sequentially computed based on the above formula (4). Then, a monosyllable whose HMM has maximum acceptance probability in the 64 HMMs is chosen as recognized monosyllable, and finally the 6-bit symbol code of the monosyllable is output with the availability of recognition (1-bit) that is decided based on a threshold value. Note that if the output symbol is available, all symbols in X are erased. Thus the chip restarts to accept another monosyllable. Fig.5 shows detailed algorithm for computing 64 acceptance probabilities and choosing the monosyllables with highest probability. The HMM module takes 139,072 clock cycles for the computation. The HMM module has a 16-bit adder. 16-bit HMM parameters (for functions A and B) are generated by software in advance and stored in external memory.

E. HMM Learning Based on Expectation Maximization (EM) Algorithm

E.1. Learning Algorithm

We use the expectation maximization (EM) algorithm to construct HMMs of monosyllables. Let there be $L+1$ training data whose length is 16, and $X_l = x_1^{(l)}x_2^{(l)}\cdots x_{16}^{(l)}$ be symbol sequence of lth training data. First, we compute α and β from a symbol sequence $X_l = x_1^{(l)}x_2^{(l)}\cdots x_{16}^{(l)}$ based on the following formulae (5) and (6).

$$\alpha^{(l)}_{q_i,n} = \begin{cases} \pi_{q_i},\ (n=0) \\ \sum_{q_j\in S,\ a_{q_j,q_i}\neq 0}\left(\alpha^{(l)}_{q_j,n-1}\cdot a_{q_j,q_i}\cdot b_{q_j,q_i}(x_n^{(l)})\right) \end{cases} \cdots(5)$$

$$\beta^{(l)}_{q_i,n} = \begin{cases} 1,\ (n=16,\ q_i\in F) \\ 0,\ (n=16,\ q_i\notin F) \\ \sum_{q_j\in S,\ a_{q_i,q_j}\neq 0}\left(\beta^{(l)}_{q_j,n+1}\cdot a_{q_i,q_j}\cdot b_{q_i,q_j}(x_{n+1}^{(l)})\right) \end{cases} \cdots(6)$$

```
Highest_logP ($X$[])
  $X[n]$:  a symbols sequence $(n=1,2,...,16)$;
  $logP$:  a probability $\log P(X|M)$ ;
  $highest\_logP$:  the highest probability ;
  α[][], $tmp\_\alpha$:  intermediate results ;
  $max\_\alpha$:  maximum value of α[4][16] ;
  $HMM_h.A$[][]:  HMM parameters of $HMM_h$ ;
  $HMM_h.B$[][][]:  HMM parameters of $HMM_h$ ;
  $HMM_h.code$:  symbol code of a monosyllable ;
begin
  $highest\_logP$ = -∞;   $symbol\_code$ = 0 ;
  /* foreach $HMM_h$ */
  for ($h$ = 1; $h \le$ 64; $h$ = $h$+1) {
    /* computing probability */
    $max\_\alpha$ = -∞;   α[0][0] = log 1.0;
    for ($i$ = 1; $i$ < 5; $i$ = $i$+1)
      { α[$i$][0] = -∞; }
    for ($n$ = 1; $n \le$ 16; $t$ = $t$+1) {
      for ($i$ = 0; $i$ < 5; $i$ = $i$+1) {
        $tmp\_\alpha$ = 0.0;
        for ($j$ = 0; $j \le i$; $j$ = $j$+1) {
          $tmp\_\alpha$ = α[$j$][$n$−1] + $HMM_h.A$[$j$][$i$]
                  + $HMM_h.B$[$j$][$i$][$X$[$n$]];
          if ($tmp\_\alpha$ > $max\_\alpha$)
            { $max\_\alpha$ = $tmp\_\alpha$; }
        }
        α[$i$][$n$] = $max\_\alpha$;
      }
    }
    $logP$ = $alpha$[4][16]; /*computed probability*/
    if ($logP$ > $highest\_logP$)
      { $highest\_logP$ = $logP$; $symbol\_code$ = $HMM_h.code$; }
  }
end
```

Fig. 5. An Algorithm for computing the highest probability.

Then, we update a_{q_i,q_j} $(i,j=0,1,...,4)$ and $b_{q_i,q_j}(k)$ $(i,j=0,1,...,4,k=0,1,...,255)$ based on the following formulae (7) and (8), until a_{q_i,q_j} and $b_{q_i,q_j}(k)$ converge.

$$a'_{q_i,q_j} = \frac{\sum_{l=0}^{L}\sum_{n=1}^{16}\alpha^{(l)}_{q_i,n-1}\cdot a_{q_i,q_j}\cdot b_{q_i,q_j}(x_n^{(l)})\cdot\beta^{(l)}_{q_j,n}}{\sum_{l=0}^{L}\sum_{n=1}^{16}\sum_{q'_j\in S}\alpha^{(l)}_{q_i,n-1}\cdot a_{q_i,q'_j}\cdot b_{q_i,q_j}(x_n^{(l)})\cdot\beta^{(l)}_{q'_j,n}} \cdots(7)$$

$$b'_{q_i,q_j}(k) = \frac{\sum_{l=0}^{L}\sum_{n=1}^{16}\alpha^{(l)}_{q_i,n-1}\cdot a_{q_i,q_j}\cdot b_{q_i,q_j}(k)\cdot\beta^{(l)}_{q_j,n}}{\sum_{l=0}^{L}\sum_{n=1}^{16}\alpha^{(l)}_{q_i,n-1}\cdot a_{q_i,q_j}\cdot b_{q_i,q_j}(x_n^{(l)})\cdot\beta^{(l)}_{q_j,n}} \cdots(8)$$

Finally, we obtain $new\_a_{q_i,q_j}$ and $new\_b_{q_i,q_j}(k)$ with respect to the weights W and M by the following formulae (9) and (10). We adopt obtained $new\_a_{q_i,q_j}$ and $new\_b_{q_i,q_j}(k)$ as HMM parameters.

$$new\_a_{q_i,q_j} = \frac{Ma_{q_i,q_j}+Wa'_{q_i,q_j}}{M+W} \cdots(9)$$

$$new\_b_{q_i,q_j}(k) = \frac{Mb_{q_i,q_j}(k)+Wb'_{q_i,q_j}(k)}{M+W} \cdots(10)$$

F. Interface to Learning Module

In learning mode, HMM parameters of all monosyllables are sequentially updated from the monosyllable of small code. First, the speech recognition chip enters a wait state to listen to approximately 1 sec speech, and extracts 16 short training frames whose sum of 16-bit speech samples is the greatest in the speech corresponding to 81 short frames. Second, the 16 training frames is converted to 16 training symbols by the LPC and the VQ module, and the training symbols are stored in external memory (LPC/VQ-Mem). Third, the learning module computes new HMM parameters by using the 16 training symbols based on the above formulae (6), (7), (8), (9) and (10). Fi-

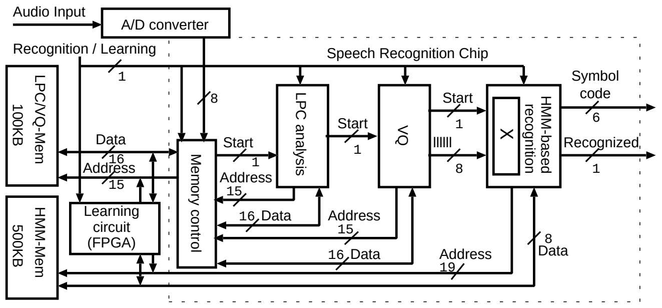


Fig. 6. Block diagram of the speech recognition and learning hardware.

nally, HMM parameters stored in external memory is updated to the new HMM parameters by the learning module.

IV. Implementation and Evaluation

A. Architecture Design

Fig.6 shows a block diagram of the speech recognition and learning hardware. The speech recognition chip includes a LPC analysis module, a VQ module, an HMM-based recognition module as described in Section III. A memory control module is also included in the chip to control memory accesses from the LPC module and the VQ module to LPC/VQ-Mem. Note that an input data from the A/D converter is stored via the memory controller. Speech samples (16-bit$\times$256$\times$81) in learning mode and the VQ codebook are stored in LPC/VQ-Mem of word length 16-bit, and HMM parameters of 64 HMMs are stored in HMM-Mem of word length 8-bit.

B. Design of Speech Recognition Chip and On-chip Learning Mechanism

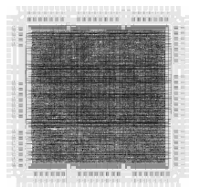
We have implemented HMM-based speech recognition system (including a LPC analysis, a VQ and an HMM-based recognition program) in C language in advance of the hardware design. The C program is about 1,700 lines. We used these programs to optimize the bit length of registers in each hardware module. We also made VQ codebook generator and an HMM parameter generator.

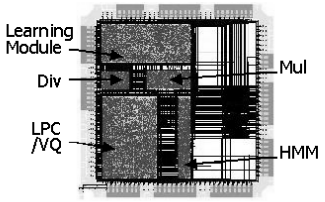
We translated the C program into VHDL. The VHDL source is about 2,300 lines. VHDL description of the speech recognition chip is synthesized with Synopsys Design Compiler, and the chip layout is performed with Avant! Apollo. We have fabricated a 4.5 mm $\times$ 4.5 mm chip via VDEC Rohm 0.6 μm process. The picture of the chip is shown in Fig.7(a). The chip includes about 15,000 cells. According to the result of post-layout simulation, the chip works at 40 MHz. The learning module is not included the chip because of the area constraint of the chip. That can be implented on an FPGA as shown in Fig.6.

We have already designed a learning circuit for the recognizer chip, and done the layout of the entire recognition chip including the learning circuit with VDEC Rohm 0.35 μm process on 4.9 mm$\times$4.9 mm chip. The VHDL source for the entire circuit is about 3,800 lines. The chip layout is in Fig.7(b). The chip includes 18,000 cells, and uses about 55 % of the chip area.

C. Evaluation of The Speech Recognition Chip

We have compared the speech recognition chip and the C program with fixed-point operations executed on a PC (CPU


(a) Speech Recognition Chip, 0.6μm, 4.5 mm $\times$ 4.5 mm



(b) Speech Recognition and Learning Chip, 0.35μm, 4.9 mm $\times$ 4.9 mm

Fig. 7. Layout of Speech Recognition Chips.

Pentium III 750 MHz, Main memory 512 MB). We have measured the elapsed CPU time of the software to recognize 60 sec speech. The elapsed CPU time was 7.2 sec and the elapsed clock cycles were 5,400$\times 10^6$. On the other hand, the real-time speech recognition chip takes 60 sec and 1,020$\times 10^6$ clock cycles since the chip recognizes in real-time at 17 MHz. Hence, the chip is 5.3 times faster than the software if the clock frequency is the same. This is because, the chip includes pipelined modules considering parallel operation. Note that the software takes 3,750$\times 10^6$ clock cycles for the LPC and VQ, and 1,650$\times 10^6$ clock cycles for the HMM. The chip takes 923$\times 10^6$ (81 frame$\times$ 60 sec$\times$189,072) clock cycle for the LPC and VQ, 631$\times 10^6$ (81 frame$\times$ 60 sec$\times$139,072) clock cycle for the HMM. Also note that DSP which executes product-sum operation in one clock cycle is not effective since the HMM module does not include product-sum operation.

V. Conclusions

In this paper, we have presented a real-time speech recognition chip for monosyllables. The speech recognition chip recognizes up to 64 monosyllables using HMM-based method. The recognition circuit also has a learning mechanism to update HMM parameters.

We are now designing a test board for the speech recognition chip including FPGA for a learning module. We would like to develop more hardware oriented speech recognition algorithms for real-time speech dictation in the future.

Acknowledgments

We would like to express our thanks to members of Watanabe Laboratory at Nara Institute of Science and Technology for their discussions and comments. We also thank to Lecturer Kazuyoshi Takagi at Nagoya University for his support on the layout of our chip. The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration by Rohm Corporation and Toppan Printing Corporation. The part of this work is supported by the NAIST IS fund 1999 and by Grants in Aid for Scientific Research in Aid for Scientific Research from the Ministry of Education, Science, Sports and Culture.

References

[1] Kai-Fu Lee, *Automatic Speech Recognition,* Kluwer Academic Publishers, 1989.
[2] Chin-Hui Lee, Frank K. Soong and Kuldip K. Paliwal, *Automatic Speech and Speaker Recognition,* Kluwer Academic Publishers, 1996.
[3] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition,* Prentice Hall PTR Prentice-Hall, 1993.
[4] X.D Huang, Y. Ariki and M.A. Jack, *Hidden Markov Models For Speech Recognition,* Edinburgh University Press, 1990.