Submit to: Workshop on Multimedia Technologies

Design of Modular Scalable HMM-based Continuous Speech Recognition / Convolutional Decoder IP

Yeu-Horng Shiau, Jer-Min Jou and Tsung-Chih Wang

Abstract

This paper presented a design method of modular scalable HMM-based continuous speech recognition / convolutional decoder IP. This IP includes three major functions: (i) Hidden Markov Model based continuous speech recognition (ii) convolutional decoder of error control coding (iii) modular scalable IP design. Since the recognition kernel of HMM-based speech recognition system and the decoding kernel of convolutional coding system are similar, we integrate the two functions in one IP by working with same hardware modules. Besides, in order to satisfy the number of recognizable words requirement of most speech recognition applications, we develop the modular scalable IP architecture that one can increase the number of recognizable words by cascoding connection with speech recognition IPs and extension modules.

Key words: Speech recognition, HMM, Convolutional code, Viterbi algorithm, Modular scalable, IP design

Yeu-Horng Shiau,

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: huh@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62431-821

Jer-Min Jou

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: jou@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62365

Tsung-Chih Wang, (the contact author)

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: tcw@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62431-821

1. Introduction

A wide variety of approaches to the problem of recognizing a spoken utterance have been proposed and evaluated. An approach based on Hidden Markov Models (HMM) helps recognizers to deal with most of the variability in the way people speak. Since the HMM technology is well-consolidated and its effectiveness is well proved, many current speech recognizers use the technique of HMM for pattern matching. An HMM is a type of model based on a doubly stochastic process in which there is an unobservable Markov chain. It provides a way of dealing with both temporal structure and the variability within speech patterns representing the same perceived sounds.

Convolutional coding with hard- and soft-decision Viterbi decoding can be used for error detection or error detection and correction. It has found application in many diverse systems such as majority-logic decoding, burst-error correction, concatenated coding system, and satellite communication systems, etc. The Viterbi algorithm provides a maximum likelihood decoding procedure that is practical for decoding short constraint length Convolutional codes.

When we observe the relation between HMM-based speech recognition and Convolutional codes, it is easily to find out that these two applications have the same computing kernel-Viterbi algorithm. Hence, we design a general architecture that can work for both applications without any conflict to increase the practical utilization of IP's functionality.

The remainder of this paper is organized as follows. In Section 2, we describe the design of modular scalable IP. In Section 3, the combination design of speech recognition and convolution decoder is presented. Then, the hardware/software (HW/SW) co-verification of this IP is introduced in Section 4, and the experiment results are presented in Section 5. Finally, we give the conclusion in Section 6.

2. Design of Modular scalable Speech Recognition IP

Generally, the number of recognizable words in one speech recognition IP is designed in constant. If one integrates speech recognition IP into his applications but the number of recognizable words is not enough to satisfy the requirements of system, the only one solution is adjusting its application to overcome this limitation.

In our modular salable IP design, one can cascode IPs with extension modules to increase the number of recognizable words and does not need to re-code the IP hardware description language. Fig. 1 shows *n* IPs cascoding connection with related extension modules and the extension modules are shown in Fig. 2.

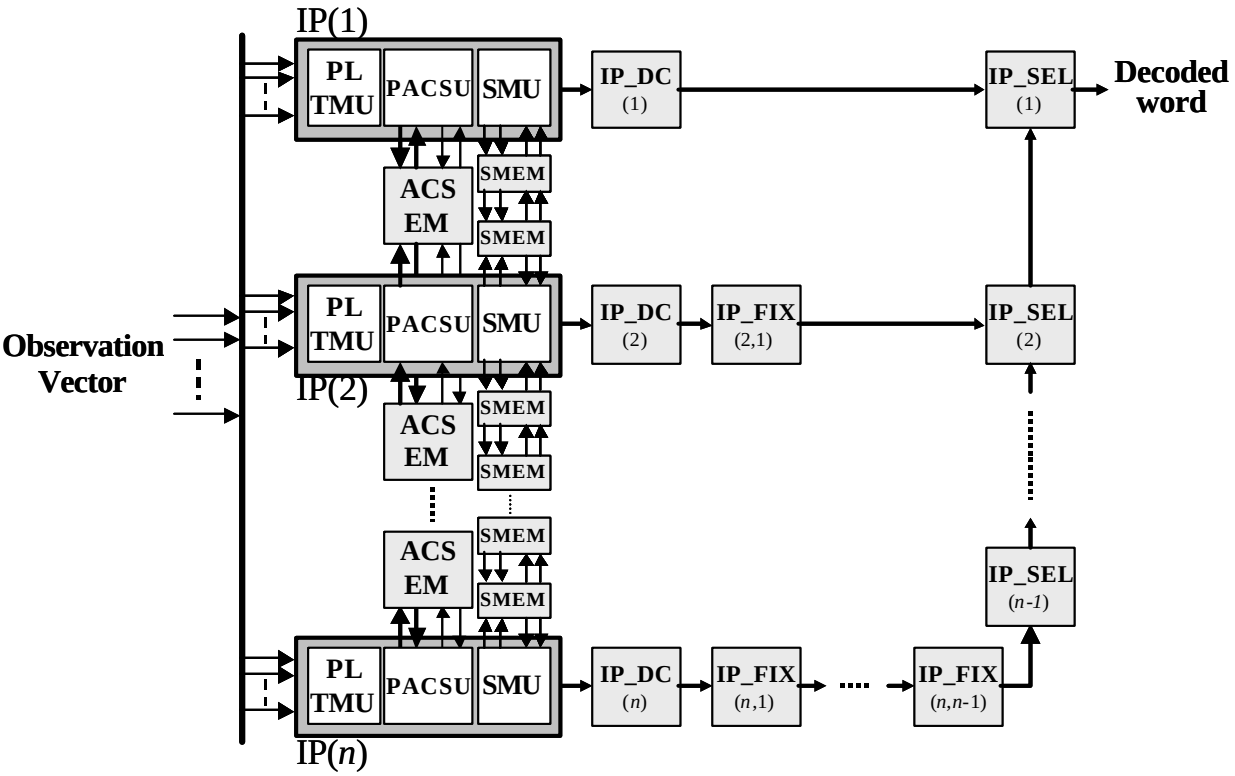


Figure 1 *N* speech recognition IPs cascading connection with extension modules.

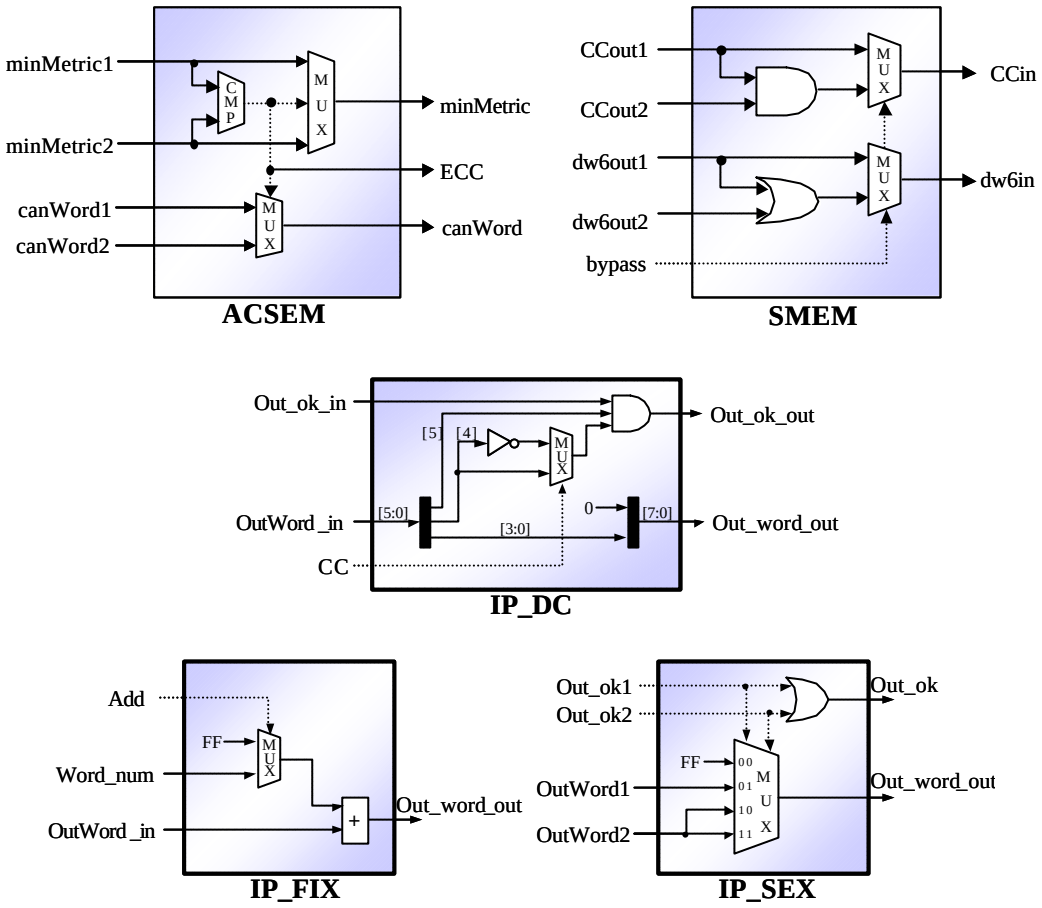


Figure 2 The architecture of extension modules.

The Extension module is described as follow:

ACSEM: The ACSEM compare two minimum paths from different IPs and send the minimum word with its related candidate word back to each IP.

SMEM: The SMEM switches extending control signals with And/Or gate among IPs.

IP_DC: The IP_DC Replaces the 6 bits output of IP to 8 bits output. Hence, the number of recognizable words is increase to 2^8=256 words.

IP_FIX: When we cascode connection more than one IP, we must add one IP_FIX to fix the recognized word tag. For example, if IP is connected in the fifth situation, the output of IP must connect five IP_FIX to fix the tag.

IP_SEL: The IP_SEL is utilized to combine the output of two IPs into one output when the number of cascoded connection IPs is over two.

3. Combinational architecture design of speech recognition and Convolutional decoder

The block diagram of IP architecture shown in Fig. 3 is divided into four major units. The Pipelined Transition Metric Unit (PLTMU) is only for speech recognition mode to compute transition metric in each state. The Hard-decision Branch Metric Unit (HDBMU) is only for convolutional decoding mode to compute branch metric in each state. Since these two operations are definitely distinct, we implement them in different hardware units. We can select convolutional decoding function or speech recognition function by switching these two units to work with the R/V signal.

The survivor data extraction tasks in the trellis diagram of each state and the survivor path back tracking tasks in memory accessing operation in convolutional decoding procedure and speech recognition procedure are similar. Hence, the survivor data extraction tasks implemented in Parallel Add-Compare-Select Unit (PACSU) and survivor path back tracking tasks implemented in Survivor Memory Unit (SMU) are designed in general architecture that can correctly work with both applications.

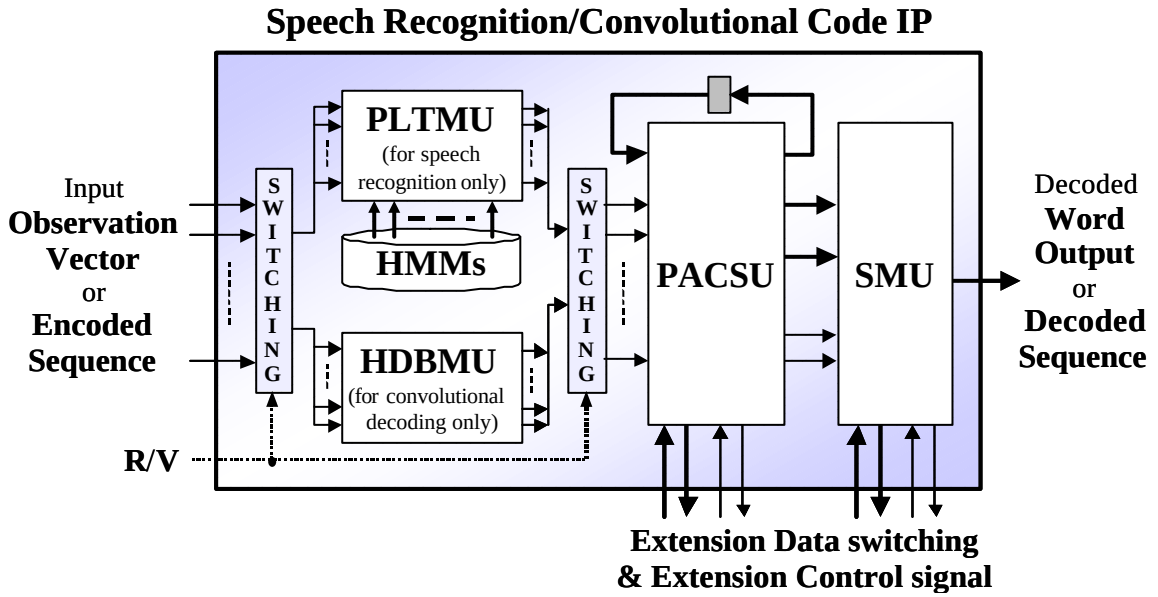


Figure 3 The architecture of speech recognition/Convolutional decoder IP

3.1 Pipelining design of PLTMU architecture

In speech recognition tasks, we first enumerate the probability $P(O|\lambda)$ of the observation sequence $O=(o_1,o_2,\ldots,o_T)$, given the model λ and can be approximated as:

$$P(O|\lambda)\approx \max_{s_1,s_2,\ldots,s_T}[\pi_{s_1}\cdot b_{s_1}(o_1)\cdot a_{s_1s_2}\cdot b_{s_2}(o_2)\ldots a_{s_{T-1}s_T}\cdot b_{s_T}(o_T)] \quad (1)$$

The observation probability of feature symbol o_t at state i ($b_i(o_t)$) can be represented as

$$b_i(o_t)=\sum_{j=1}^{m}C_{ij}\frac{1}{\sqrt{(2\pi)^D\cdot\prod_{k=1}^{D}\sigma_{kk}^2}}\cdot e^{-\frac{1}{2}\sum_{k=1}^{D}\frac{(o_{tk}-\mu_{ijk})^2}{\sigma_{kk}^2}} \quad (2)$$

where M is the number of Gaussian mixtures, D is the order of MFCC parameters, C_{ij} is the weight of cluster j in state i, μ_{ijk} is the k-th order mean vector in cluster j of state i and σ_{kk}^2 is the element [k,k] of covariance matrix.

In order to reduce the complexity for hardware design, equation (2) is simplified as

$$b_i(o_t)\approx \underset{j=1\sim M}{Max}\frac{1}{\sqrt{(2\pi)^D\cdot\prod_{k=1}^{D}\sigma_{kk}^2}}\cdot e^{-\frac{1}{2}\sum_{k=1}^{D}\frac{(o_{tk}-\mu_{ijk})^2}{\sigma_{kk}^2}} \quad (2)$$

Then, we take the negative logarithms in equation (2) and obtain the observation probability as:

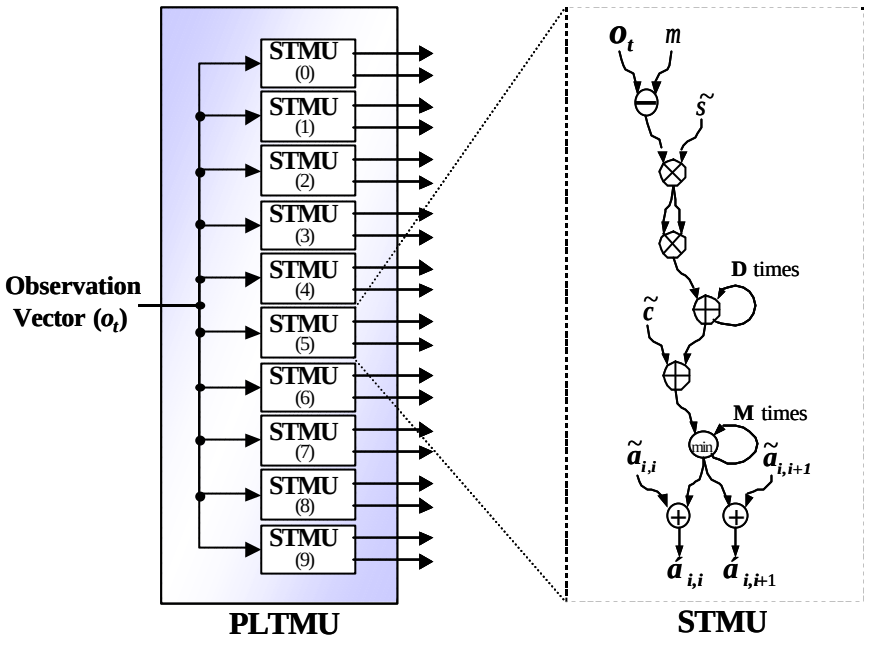
$$\tilde{b}_i(o_t)=-\log[b_i(o_t)]=\underset{j=1\sim M}{Min}\left\{\tilde{C}_{ij}+\sum_{k=1}^{D}\left[\tilde{\sigma}_{kk}(o_{tk}-\mu_{ijk})\right]^2\right\} \quad (3)$$

$$\text{where } \tilde{C}_{ij}=-\log\left(C_{ij}\frac{1}{\sqrt{(2\pi)^D\cdot\prod_{k=1}^{D}\sigma_{kk}^2}}\right),\ \tilde{\sigma}_{kk}=\frac{1}{2\sigma_{kk}}$$

Finally, by adding the observation probability and transition probability ($a_{i,i}$ and $a_{i,i+1}$), the output of PLTMU, transition metric $\delta_{i,i}$ and $\delta_{i,i+1}$, can be obtained.

$$\begin{cases}\delta_{i,i}=a_{i,i}+\tilde{b}_i(o_t)\\ \delta_{i,i+1}=a_{i,i+1}+\tilde{b}_i(o_t)\end{cases} \quad (4)$$

We draw the data flow of one state in Fig. 4 accord to the Eq. (5) and the computation elements of one state in PLTMU include 2 multipliers, 5 adders/subtractors and 1 comparator.



(a) (b)

Figure 4 (a) The architecture of PLTMU including 10 STMU computation elements (b) The data flow of STMU

As shown in Fig. 4, if we use fully parallel architectures to design PLTMU with *S* state in each HMM to reduce the clock cycles of transition metrics computation, the number of states in PLTMU is 10*S*. Hence, the totally number of computation elements of IP need 10**S**2 multipliers, 10**S**5 adders and 10**S* comparators.

Consider an example with the number of states in each HMM is 4 and design the architecture with fully parallel techniques, there will be 40 states in one IP and the computation elements include 80 multipliers, 200 adders and 40 comparators. This shows that the fully parallel architecture design is not efficient in IP area saving.

In order to achieve an efficient IP design with lower area utilization, we utilize one state computation element set to compute one HMM with *S* states computations. Hence, there are 10 computation elements called STMU in PLTMU architecture shown in Fig. 4(a).

Besides, for improving the computation performance of STMU, we utilize the pipelining technique to break the summing and minimizing operation loops. The architecture of STMU can partition into PE1 and PE2 modules shown in Fig. 5.

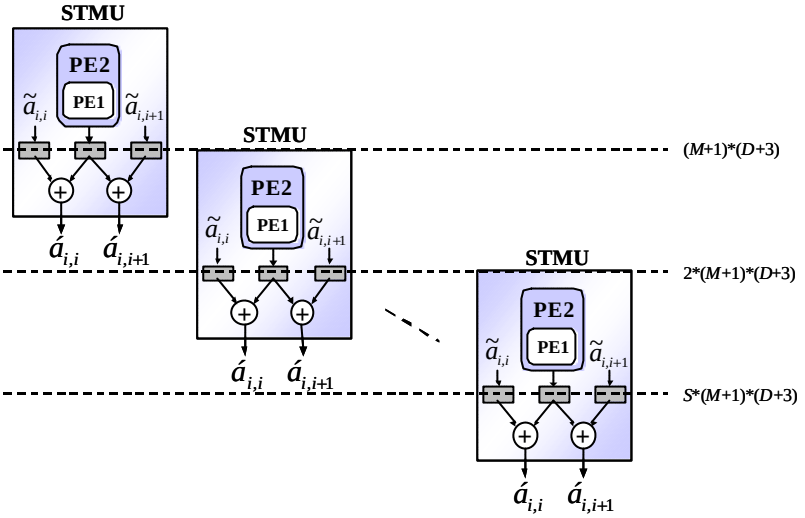
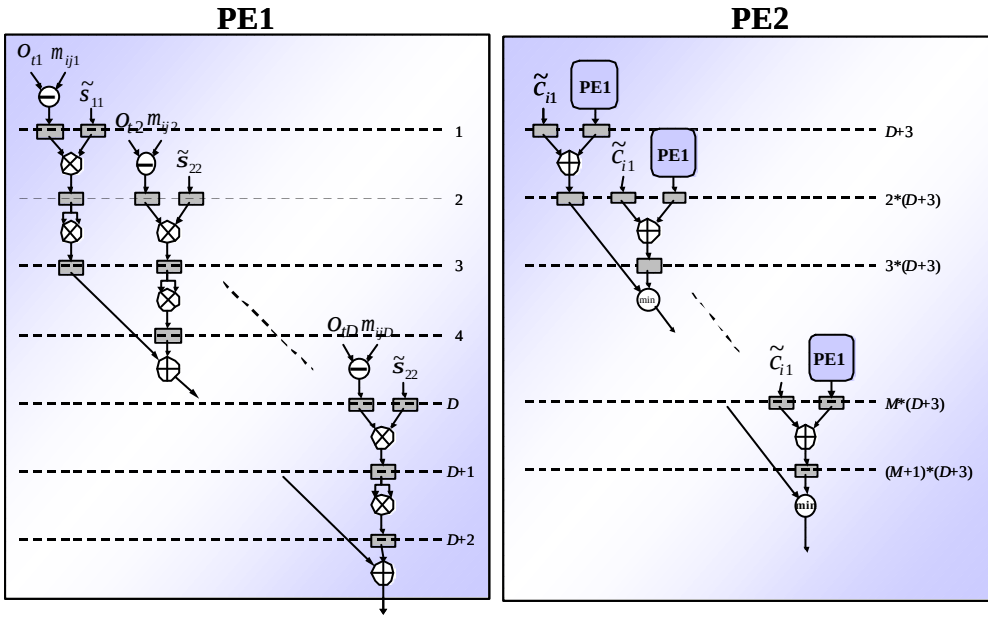


Figure 5 STMU pipeline scheduling

The PE1 and PE2 of STMU pipeline scheduling is drawn in Fig.6.



(a) (b)

Figure 6 (a) PE1 pipeline scheduling (b) PE2 pipeline scheduling

In PE1, the original *D* summation loop takes *D**3 clock cycles can be reduced to *D*+3 clock

cycles after using pipeline techniques. The same improvement occurs in PE2, the original *M* loops take 2*(*M*+1)*(*D**3) clock cycles and can be reduced to (*M*+2)*(*D*+3) clock cycles. It really more effect than the design without using pipeline techniques.

3.2 HDBMU architecture design

The function of HDBMU is to compute branch metrics in each state of the trellis diagram of convolutional decoder. The architecture of HDBMU includes one register, two inverters and four adders shown in Fig. 7 can be used in any code-rate-1/2, hard-decision convolutional decoder.

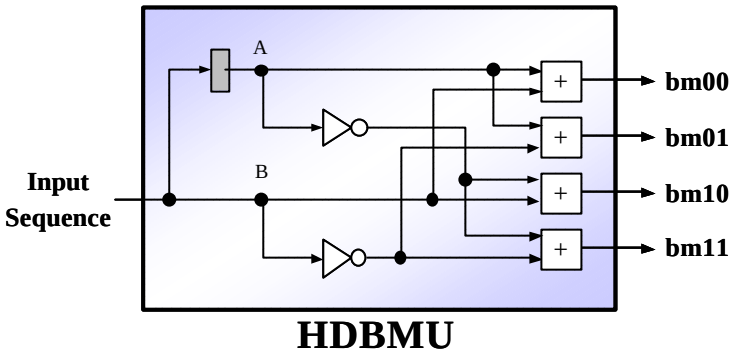


Figure 7 The architecture of HDBMU

The branch metric equations bm00, bm01, bm10 and bm11 is illustrated as

$$\begin{cases} \textbf{bm00} = A + B \\ \textbf{bm01} = A + \sim B \\ \textbf{bm10} = \sim A + B \\ \textbf{bm11} = \sim A + \sim B \end{cases}$$

3.3 PACSU architecture design

The PACSU accumulates path metrics by adding the transition metric from PLTMU or the branch metrics from HDBMU with previous time step path metrics for all possible paths in trellis diagram and find the matching path with minimum error, which is also called survivor path, of each state. The data flow of each state is shown in Fig. 8.

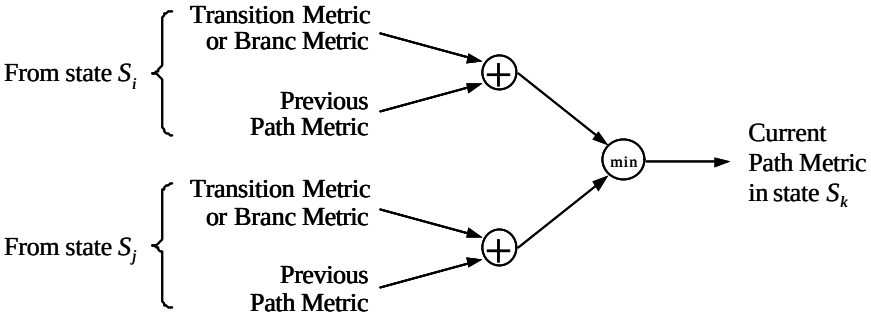


Figure 8 The data flow in each state.

Because the computing speed requirement is more important in convolutional decoder designs, we design the PACSU with parallel techniques to reduce the performance diminishing of feedback loop latency as shown in Fig. 9.

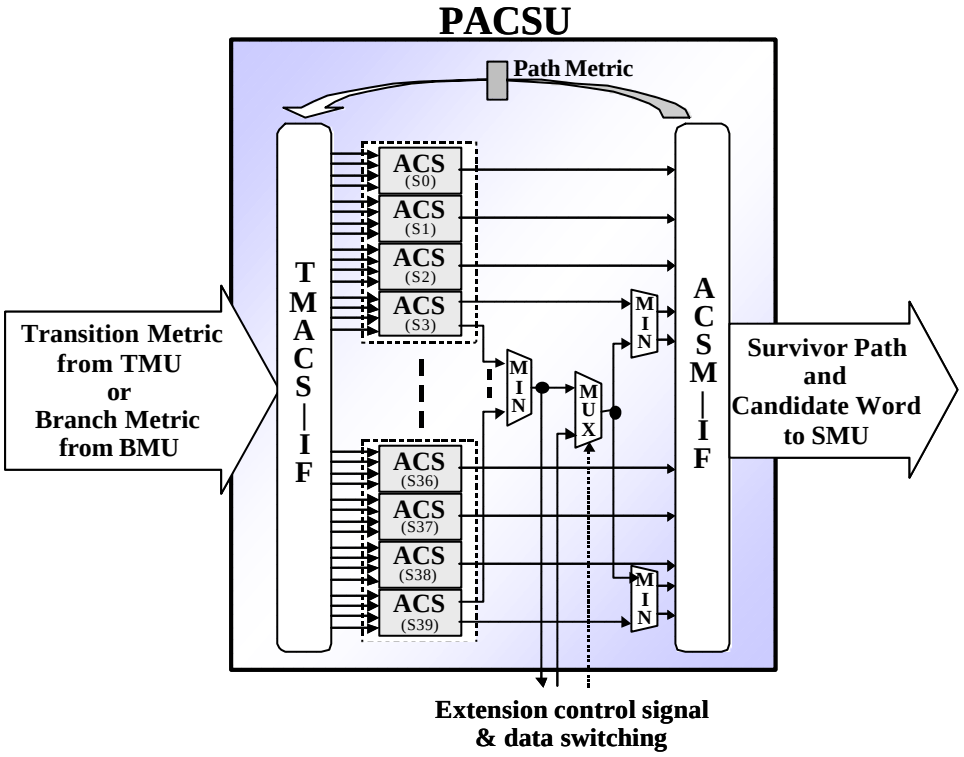


Figure 9 Parallel design of PACSU architecture.

In Fig. 9, the transition metrics generated by PLTMU or branch metrics generated by HDBMU and previous path metric from ACSM-IF will send to TMACS-IF module and distribute to related ACSs according to its own trellis diagram. The architecture of ACS shown in Fig. 10 includes two adders, one comparator and one multiplexer. Two pairs of branch inputs in one state are compared

to find the minimum path metric and survivor path information.

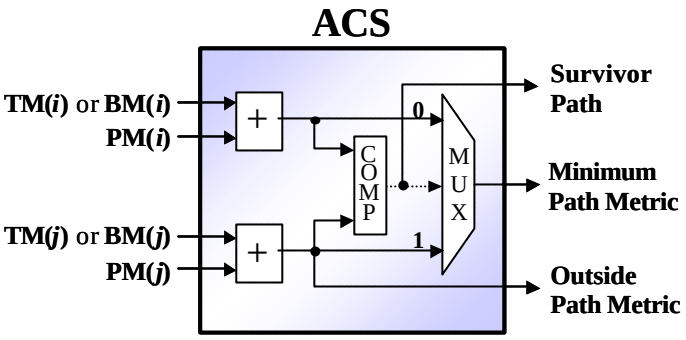


Figure 10 The architecture of ACS module.

After finished the survivor path information computed by parallel ACSs architecture, the 40 survivor bits output from 40 ACSs will be combined into one survivor word before sending to SMU in ACSM-IF module. Simultaneously, current path metric in each state will update and send back to TMACS-IF module. Besides, in speech recognition function mode, the candidate word information in each time step will be sent to SMU with the survivor path information, too.

3.4 SMU architecture design

The survivor path can be regarded as a circular *N* block of memory as shown in Fig. 11. We can break the memory into four blocks, M1, M2, M3, M4 and these blocks corresponds to different processes of traceback called Survivor Data Write (SDW), TraceBack Read (TBR), and DeCode Read (DCR) describe as follows.

1. Survivor Data Write (SDW)

Decision bits of all states from the PACSU combined into one word are written in one of the memory blocks M1, M2, M3 or M4 that are used cyclically.

2. TraceBack Read (TBR)

The TBR processing is performed in one of the memory blocks. An arbitrary state S_n at time n is chosen and recalls the trellis connections in the reverse order that they were stored to trace the

corresponding survivor path. The decision bit $D_n^{S_n}$ for S_n is obtained from survivor memory using S_n as the selection signal. The previous state of S_n in the survivor path can be computed by $S_{n-1} = D_n^{S_n} | (S_n >> 1)$. This process will be repeated for 2*D iterations to reach the convergence state of survivor paths. Hence, we obtain 2*D bits of the surviving path corresponding to starting state S_n. These bits are not used for the output, but the oldest bit is the starting state for a DCR process in the next period.

3. DeCode Read (DCR)

The DCR is performed in one of the memory block. The output bit of DCR process is the correct input sequences can be decoded. In fact, the DCR and TBR processing involve the same memory read operation. The difference is the 2*D bit traceback outputs from TBR are the selected path, but the traceback outputs from DCR are the reverse order of decoder actually read out results.

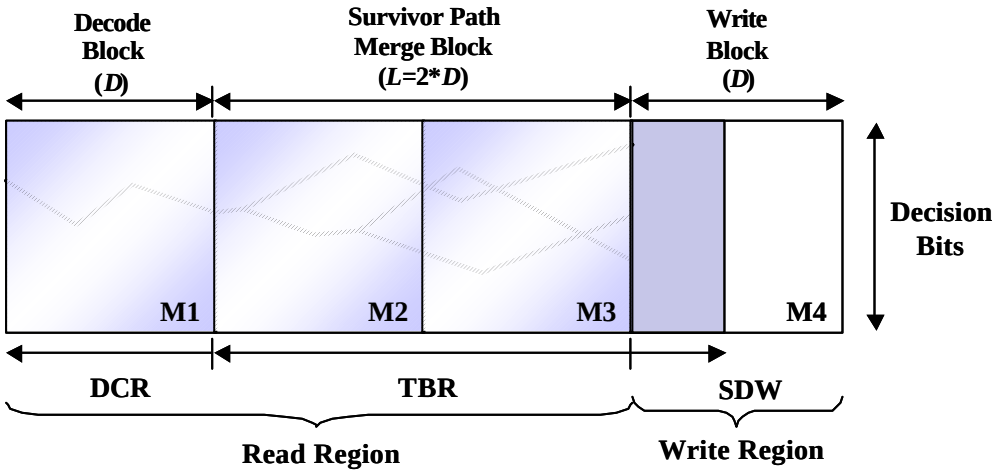


Figure 11 Survivor memory organization.

The dynamic scheduling among four RAM blocks for 10 periods of the traceback procedure is shown in Fig. 12. The horizontal axis of Fig.12 represents the time in terms of the clock cycles, and the vertical axis represents the column address of the memory. During SDW procedure, one decision path (40 bits) is written to survivor memory and a WR point is used to keep track of its position. Simultaneously, the RE pointer is used to keep track of the position of TBR and DCR

operations. The WR pointer accesses the memory in the increasing order but the RE pointer accesses in the opposite direction. Note that in the traceback processes, all bits in the same column of the memory are accessed during write operation, but only one bit needs to be accessed during read operation; the memory access time for both pointers is the same, but the RE pointer moves three times faster than the WR pointer.

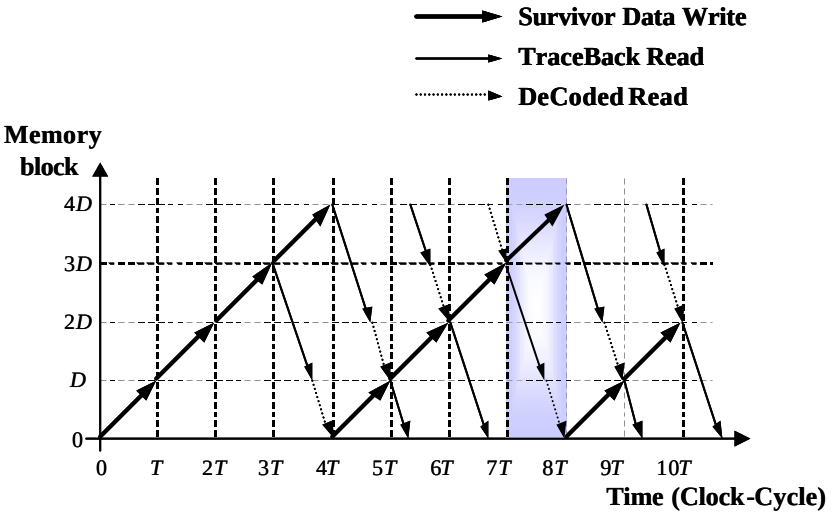


Figure 12 Survivor memory scheduling.

We design the SMU with survivor memory as shown in Fig. 13.

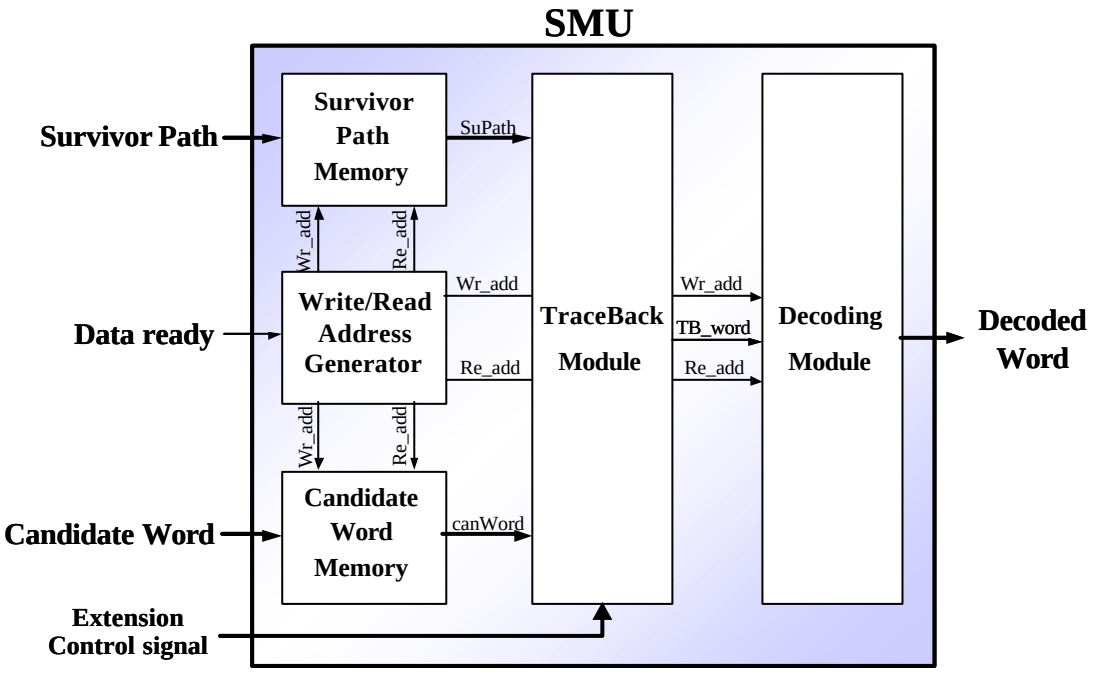


Figure 13 The architecture of SMU

4. Hardware and Software Co-Design

The rapid hardware prototyping system is built based on hardware/software co-design methods. The complete co-verification environment includes, software program run in a PC, hardware module (IP) implemented in FPGA target board and communicate between hardware and software with ISA bus architecture. Since software programs can access input/output data of hardware module through the memory map read/write scheme of ISA, one can send test patterns generated by software program to hardware module and observe the simulation result to verify the functionality of IP.

We draw the configuration of the prototyping board in Fig. 14.

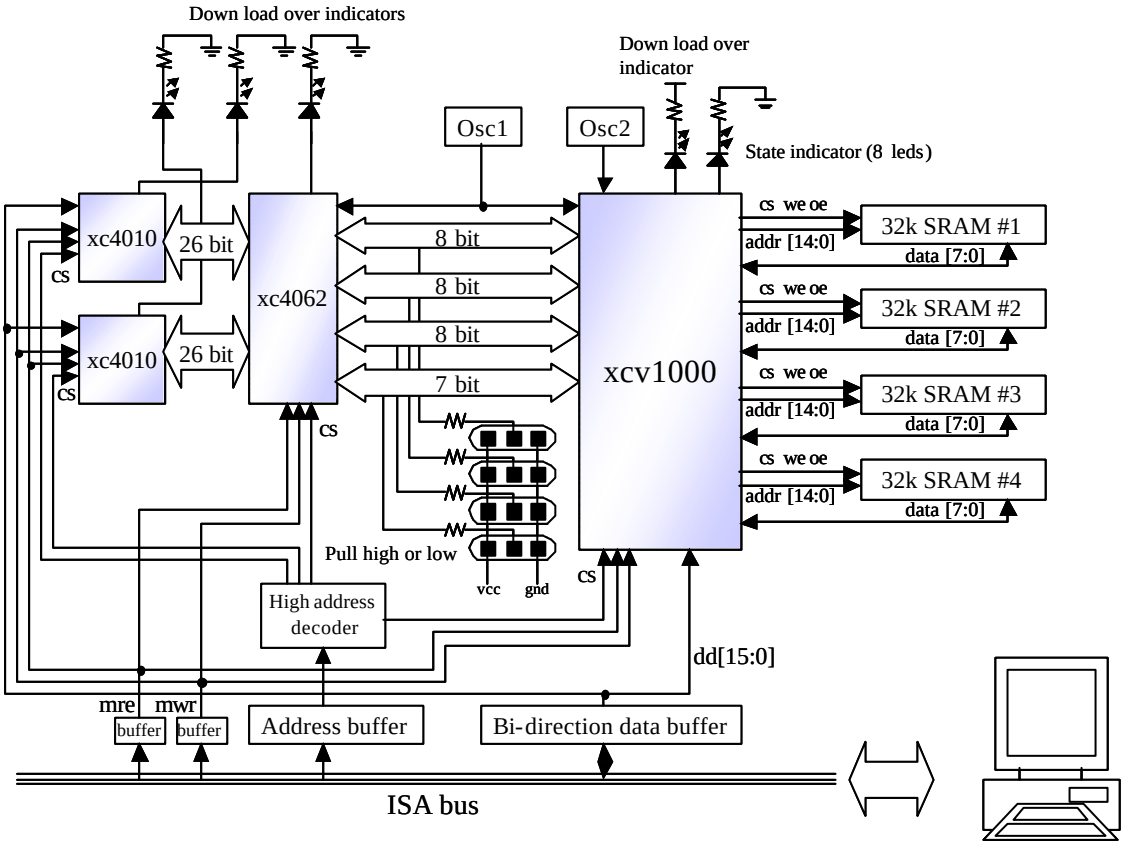


Figure 14 Configuration of prototyping system.

4.1. Continuous Speech Recognition System

The block diagram of the speech recognition system is shown in Fig. 15. It is separated into two parts, software and hardware parts. The software part is implemented with C code running on a PC including speech data collecting, front-end processing and parameters modeling for HMM training tasks. The hardware part (IP) is implemented in the FPGAs prototyping board including observation probability estimation and Viterbi processing computations.

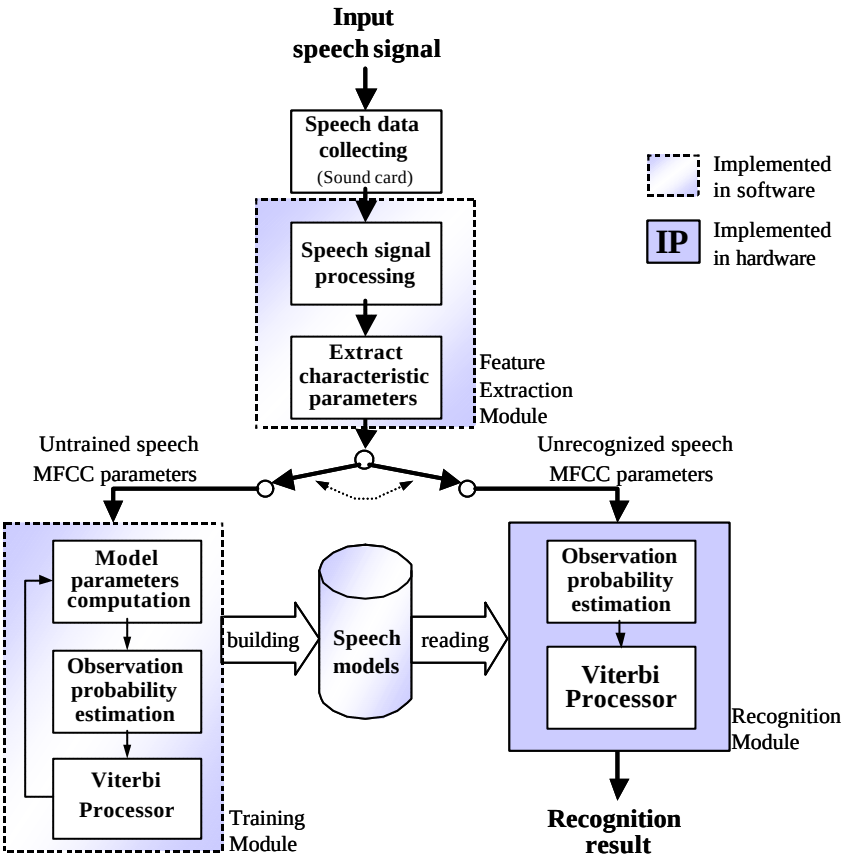


Figure 15 The block diagram of Speech recognition system.

4.2. Convolutional Coding System

The block diagram of the Convolutional coding system is shown in Fig. 16. The software running on a PC includes convolution encoding and AWGN channel simulating. The encoded sequence is then sent to the IP on the FPGAs target board to begin the Convolutional decoding

procedure.

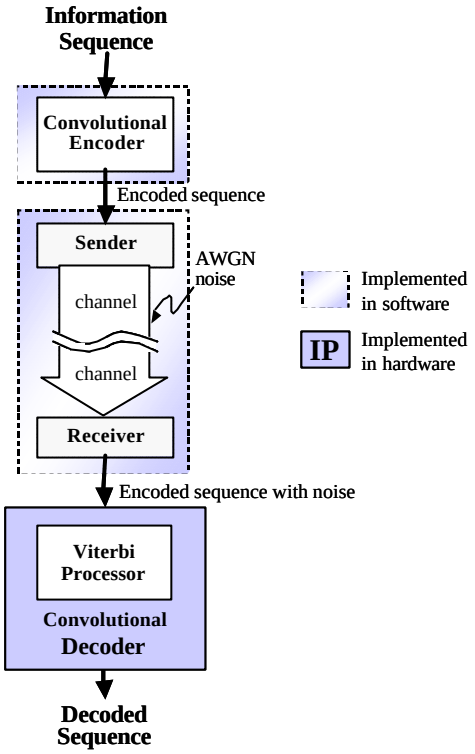


Figure 16 The block diagram of Convolutional coding system .

5. Experiment Result

5.1 Synthesis Result

Fig. 17 and Table 1 show the layout diagram and utilization rate of resource for our IP in Xilinx xcv1000 FPGA chip respectively.

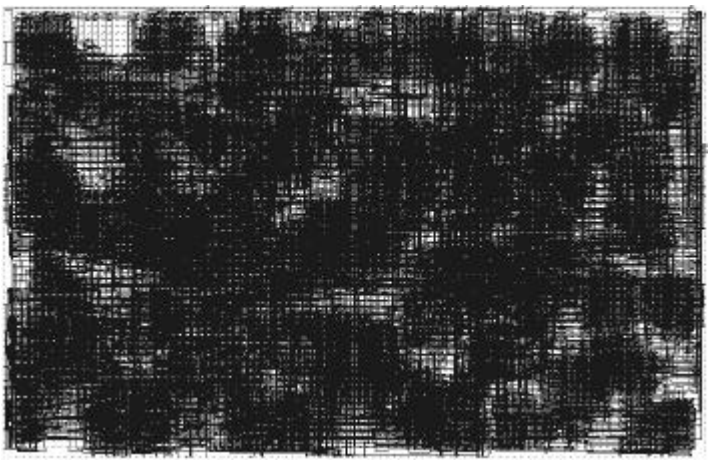


Figure 17 FPGA Layout diagram (Target: Xilinx xcv1000 BG580).

Table 1 Resource utilization of speech recognition IP.

Resource	USED	MAX available	% USED
Number of Slices	8,446	12,288	68%
Total Number Slice Flip Flops	4,722	24,576	19%
Total Number 4 Input LUTs	14,469	24,576	58%
Number of bounded IOBs	41	404	10%
Number of Block RAMs	26	32	81%
Number of GCLKs	1	4	25%
Number of GCLKIOBs	1	4	25%
Total equivalent gate count	600,756		
Additional JTAG gate count for IOBs	2,016		
Minimum period	69.412ns (Maximum frequency: 14.407MHz)		
Maximum net delay	16.119ns		

The resource utilization listed in Table 1 shows that the maximum operating clock rate is above 14 MHz and available to process about 14 million trellis steps per second.

5.2 Speech Recognition Rate Analysis

In order to analyze recognition rate in our speech recognition system, we gather 200 random generated sentences include 1370 digits to be recognized by 10 users. The random generated sentences are listed in Table 2 and the analyzed results are shown in Table 3. The experimental results show that the average speech recognition rate is above 90%.

Table 2 Random generated test sentences.

8182	6603	8012	50947	83512
01640	618984	143988	087783	8371073
4965109	9683484	99255333	74380880	68198973
282890781	586124258	626539246	1821197629	5200391819

Table 3 Recognition rate for 10 users

Samples	Recognition Errors			Word Error Rate (%)
	Insertion	Deletion	Substitution	
User1	15	67	53	90.15
User2	17	60	65	89.64
User3	18	55	61	90.22
User4	16	59	58	90.29
User5	17	61	60	89.93
User6	16	57	66	89.85
User7	18	65	55	89.93
User8	15	56	60	90.44
User9	16	64	57	90.00
User10	17	57	59	90.29

5.3 Bit Error Rate Analysis

We draw the bit error rate simulation results for a (2, 1, 5) convolutional coding on an AWGN channel with hard-decision FPGA simulation decoding, soft-decision software simulation decoding, and uncoded result of our convolutional coding system.

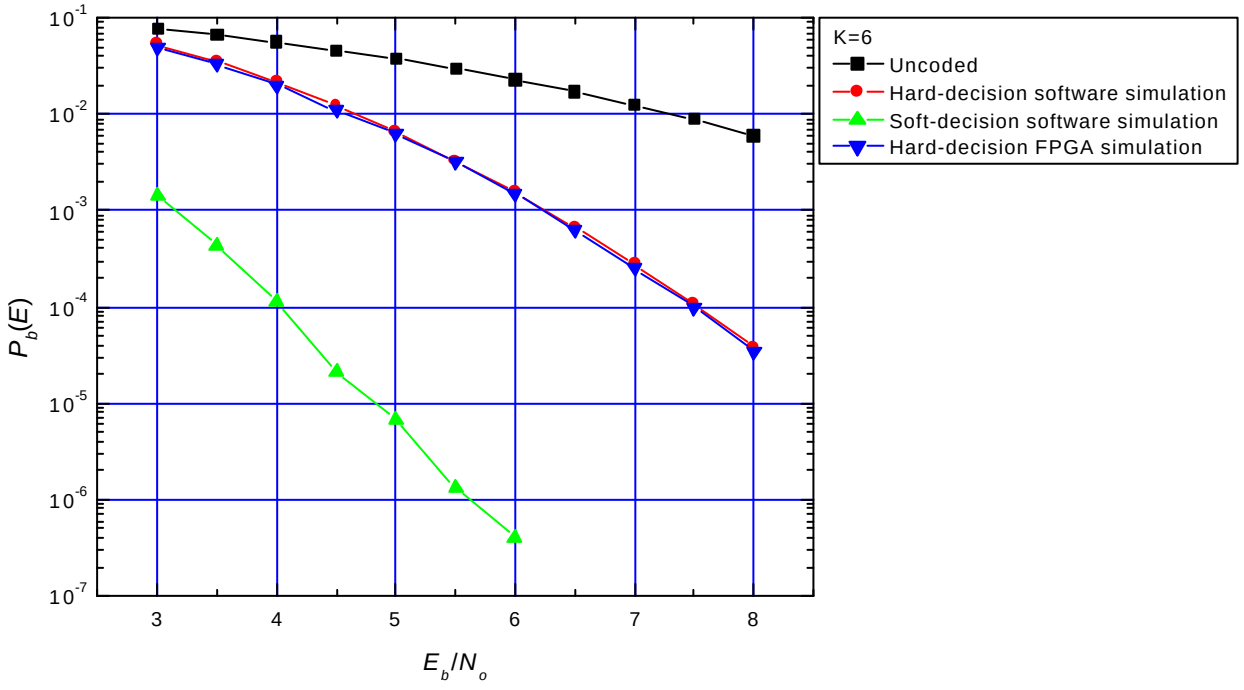


Figure 18 Simulation results for a (2, 1, 5) convolutional coding on an AWGN

channel.

6. Conclusion

This speech recognition IP integrates diversity function includes continuous speech recognition, convolutional decoder of error control coding and IP function scalable extension. General architecture of module design methods make two different application domains, speech recognition function and convolutional coding function can work together without any conflict.

Besides, the concept of modular scalable IP implementation overcomes the limitation of IP function extension. This modular scalable IP design method simultaneously releases the IP user from the utilization overhead in many applications. We can call it an originative design in speech recognition filed.

The hardware/software co-verification prototyping system ensures the reliability of IP design. Of course improves the completeness of IP design with system integration. Therefore, we built the continuous speech recognition system and the convolutional coding system to emphasize the advantage of easy to integrate with our designed IP.

Reference

[1] S.-H. Choi, J.-J. Kong, "State parallel Viterbi decoder soft IP and its applications," in *Proc. of IEEE Region 10 Int. Conf. Electrical and Electronic Technology, TENCON*, Vol. 1, pp. 355-358, 2001.

[2] R.V.K. Pillai, P. D'Arcy, "On high speed add-compare-select for Viterbi decoders," in *Proc. of Canadian Conf. Electrical and Computer Engineering*, Vol. 2, pp. 1193-1198, 2001.

[3] F.L.Vargas, R.D.R. Fagundes, D.B. Junior, "A FPGA-based Viterbi algorithm implementation for speech recognition systems," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Vol. 2, pp.1217-1220, 2001.

[4] M.P.C. Fossorier, Shu Lin, "Differential trellis decoding of Convolutional codes," in *IEEE Trans. Information Theory,* Vol. 46, Issue 3, pp.1046–1053, May 2000.

[5] C.-W. Wang, Y.-N. Chang, "Design of Viterbi decoders with in-place state metric update and hybrid traceback processing," in *IEEE Workshop on Signal Processing Systems*, pp. 5–15, 2001.