Technique Report For An Adaptive IC For Automatic Speech Recognition on 130nm TSMC

Hoang Trang, Pham Dang Lam, Tran Van Hoang

Fac. of Electronics & Telecommunication, HCM Univ. of Technology, Ho Chi Minh City, Vietnam

{lamd.pham, tvhoang, hoangtrang}@hcmut.edu.vn

Abstract: Hidden Markov Model (HMM) scheme has been well known in Auto Speech Recognition (ASR) systems as well as experienced effective performance in comparison to others such as Artificial Neural Network (ANN), Vector Quantization (VQ), or Dynamic Time Wrapping (DTW). Furthermore, scaling of More's law for transistor density on chip is able to approach HMM in hardware approach. However, more different languages there are, more HMM hardware configurations are created to achieve the best performance. As the result, reusing the previous HMM configurations to apply in new schemes has become one of critical issues and costs highly. In order to be able to tackle this problem, a dynamic hardware configuration following continuous HMM scheme is proposed in this work. In other to present superiorities to others as well as approach the handset applications basing dynamic setup, well-design chip on TSMC 130nm technology integrating feature extraction Mel Frequency Cepstral Coefficients (MFCC) and HMM decode has implemented to confirm high recognition probability with 96.2 % and adapt maximum frequency recorded at 100 MHz.

Index Terms: *Hidden Markov Model (HMM), Application specific integrated circuit (ASIC), register transfer level (RTL), Taiwan Semiconductor Manufacturing Company (TSMC), auto speech recognition (ASR), field programmable gate array (FPGA).*

I. Introduction

To start with, the daily higher transistor density brings with it a number of advantages of applying HMM algorithm to hardware models. Firstly, one FPGA-base speech recognition system was presented by Hoang et al. as the Fig .1. In this structure, both MFCC extraction and decode processes approaching continuous HMM scheme were implemented on Nios II core integrated in Altera FGPA Cyclone II completely [1].

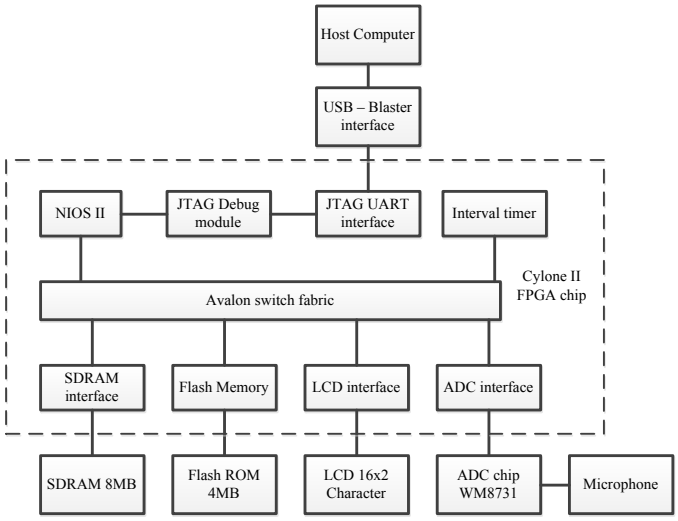

Fig. 1 A FPGA base speech recognition system

However, resource limitation of FPGA kit correctly with low performance of NIOS II core affects this hardware architecture negatively. Particularly, the enormous number of timing consumption was concerned. To be able to

enhance real-time requirement strictly, a hybrid architecture was illustrated by Octavian Cheng et al. in Fig. 2 [2]. Obviously, only Gaussian probability distribution function was implemented in hardware while the other functions were undertaken by NIOS II processor as [1]. This HMM scheme comprising three states in which each state included four mixtures also confirmed high recognition probability at 93.33% in word level. The time consumption for recognition process was confirmed that time cost on GMM step decreased from 78% recorded from full software approach to only 2% reported by hardware-software base. However, Viterbi decoding step following software on Nios II core estimated to cost 27% time requirement in comparison to whole recognition process was not improved yet.

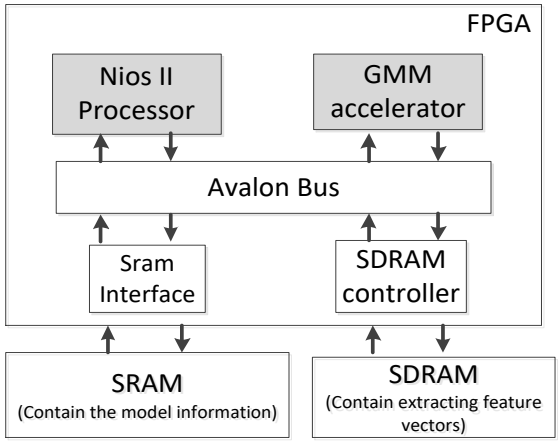

Fig. 2 The co-processor for recognition machine

Another architecture applying parallel technique was proposed to tackle real-time requirement by Richard Veitch et al. as Fig. 3 [3]. In this research, an experience with 4000 models is main reason to employ parallel technique to innovate the speed.

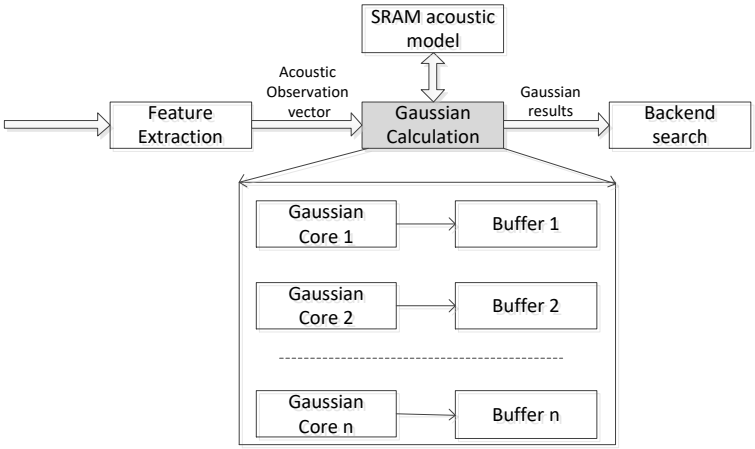

Fig. 3 Parallel technique for Gaussian calculation

Likewise, many approaches were promoted and innovated to meet the real-time criterion strictly as well as limited resource requirements mainly on FPGA architecture base [4, 5, 6, 7] or ready MCU [8]. To be able to success a complete hardware and correctly estimate the trade-off between recognition speech and resource utilization, Kazuhiro Nakamura et al. promoted an effective recognition system applied parallel technique [9]. In this research, experiment results on FPGA obviously support to decide the best number of parallel lines for calculating Gaussian probability and Viterbi scorer in Viterbi decode architecture. While a majority number of researches approach HMM, someone referred to ANN and achieved high probability such as [9, 10] with similar 92%,respectively.

From overall perspective, it is not ignored fact that most designs previously analyzed have experienced fixed configurations or evaluated on FPGA targets mostly. The reason is explanted that it may be suitable to obtain highest performance or causes of acoustic characteristics belong to different languages. Furthermore, there is few research to promote unified HMM scheme to be suitable to any language. Another aspect need to be concerned is that approaching ASIC product from fixed HMM configuration will become critical issue mainly because of being not reused. In order to tackle this issue, a dynamic HMM architecture following ASIC base design is illustrated in this paper. To be more specific, the below sections are described.

First and foremost, Section II presents the background knowledge to HMM necessarily. Base on the HMM theory, the proposed dynamic HMM hardware architecture is explanted in detail in Section III. Hence, the necessary experiments are introduced to confirm the efficient performance in Section IV. Finally, that is Section V including the conclusion and future works.

II. Background

2.1 HMM model applied in ASR

It is fact that HMM is one of the well-known mathematic models employed in many researches. In ASR application, HMM is strongly recommended as the best method to obtain the best results. In many HMM variants, the forward HMM illustrated in Fig. 4 is the most popular scheme to approach in ASR system.

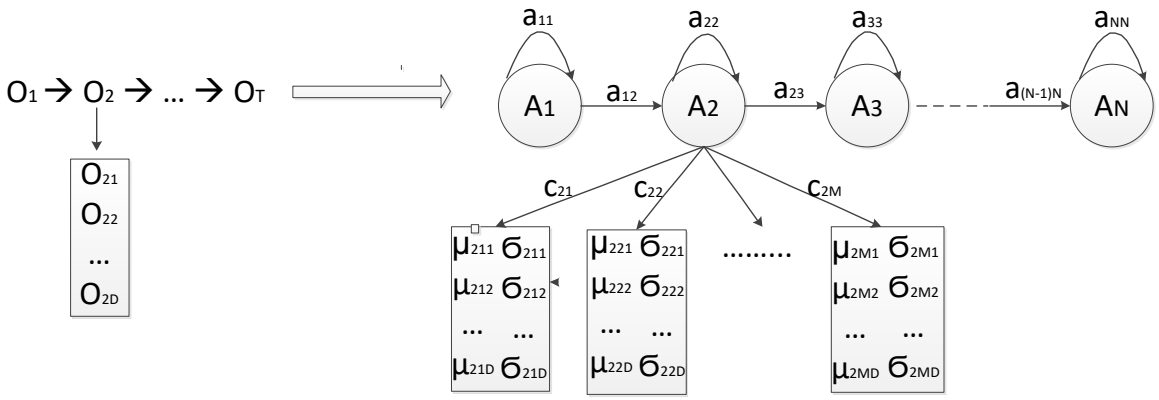

Fig. 4 A forward HMM scheme for speech recognition application

Basically, considered forward HMM configuration comprises *N* distinct states indexed by *{A1, A2, …, AN}* in which each state comprises *M* mixtures. Be able to specify each mixture, there are two matrixes generally named *Mean - μ* and *Covariance - б* which have same dimension *D*. Besides, other parameters such as transferred state probabilities $a_{(n-1)n}$ and chosen mixture probability c_{nm} also need to be concerned. Total of presented parameters, results of training process, are called as the set of model information symbolized *λ*. The given issue is that if there is a discrete time observation $O_1, O_2, \ldots, O_T$ in which each serial member symbolized O_t is the *D* dimension vector, how to obtain maximum dependent probability following below statement.

$$P(O|A,\lambda) = \prod_{\substack{t=1\\n=1}}^{T,N} P(O_t|A_n).a_{(n-1)n} \quad (1)$$

$$\Leftrightarrow P(O|A,\lambda) = P(O_1|A_1).(a_{11\ or\ 12})P(O_2|A_{1\ or\ 2}).(a_{11\ or\ 12\ or\ 23}).P(O_3|A_{1\ or\ 2\ or\ 3}) \ldots a_{(11\ or\ 12\ or \ldots (n-1)n)}P(O_T|A_{1\ or\ 2\ or \ldots N}) \quad (2)$$

In the formula (2), the probability member which is Gaussian probability distribution is described by the formula (3) and (4)

$$P(O_t|A_n) = \sum_{m=1}^{M} c_{nm}.£(O_t,\mu,6) \quad (3)$$

$$\Leftrightarrow P(O_t|A_n) = \sum_{m=1}^{M} c_{nm}.\frac{1}{(2\Pi)^{\frac{D}{2}}.(\prod_{d=1}^{D} 6_{nmd})^{\frac{1}{2}}}\ e^{-\sum_{d=1}^{D}\frac{(O_{td}-\mu_{nmd})^2}{26_{nmd}^2}} \quad (4)$$

From the formulas (2) and (4), it is irrefutable fact that the Gaussian probability distribution is recalled in many times to have the final results. Particularly, if there is certain *T* and *N* numbers, the number of calculating Gaussian probability named *y* is followed statement (5)

$$y = N.T \quad (5)$$

To be more specific, if there is two dimension matrix *(N:T),* the number of Gaussian function are all of remarked members in matrix as Fig. 5. Base on the all probability members in this matrix, the final maximum probability is obviously calculated as formula 6

$$P(O|A,\lambda) = P(O_1|A_1).a_{12}.P(O_2|A_2).a_{23}P(O_3|A_3).a_{33}P(O_4|A_3).a_{34}.P(O_5|A_4) \quad (6)$$

In order to obtain this maximum value, all probabilities remarked in the matrix have to be calculated absolutely.

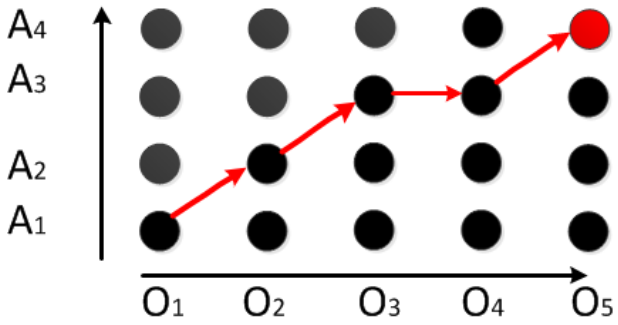

Fig. 5 Matrix states and observations

The large number of using Gaussian probability is one of the major issues negatively to affect the recognition speed in hardware system. To be able to enhance this problem, the Viterbi algorithm is recommended obviously. To be more detail, one observation O_t only need two times of calculating Gaussian probability function as Fig. 6 and formula (7)

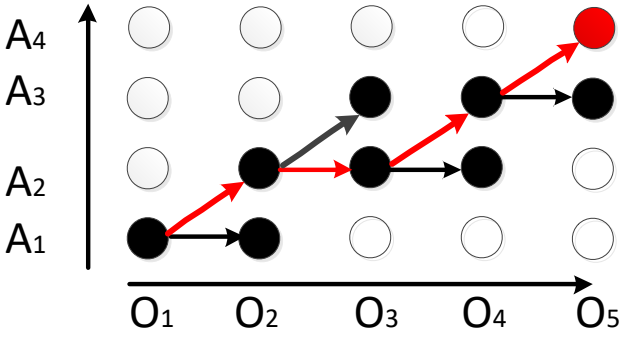

Fig. 6 Matrix states and observations for Viterbi algorithm

$$P(O|A,\lambda) = P(O_1|A_1).a_{12}.P(O_2|A_2).a_{22}P(O_3|A_2).a_{23}P(O_4|A_3).a_{34}.P(O_5|A_4) \quad (7)$$

Comparing the number of recalled Gaussian probability function between Viterbi decoding method and the previous method, the Viterbi scheme is always superior confirmed by maximum reused number is *2T - 1* in comparison to *N.T* as formula (5). However, the researched experiments really concern that although Viterbi algorithm is efficient to enhance the decoding speed, the final probability is not always the maximum target that affects and reduces performance of whole design. It becomes critical issue to apply in the special applications requiring high correction absolutely. Some approaches still keep strong perspective to ignore the Viterbi decoding method in order to obtain the best results.

2.2 Approaching HMM hardware base

In preceding sections, a basic HMM algorithm in ASR application is presented. Hence, this continuously illustrates how to approach HMM hardware. To be able to solve the complicated formula illustrated as (1), most researches utilized logarithm function to convert serial multiplies to serial adders that is feasible to approach hardware absolutely. As the result, the original formula (1) is converted to below statements

$$ln[\,P(O|A,\lambda)\,] = ln\,[\,\prod_{t=1,\,n=1}^{T,N} P(O_t|A_n).a_{(n-1)n}\,] \quad (8)$$

$$ln[\,P(O|A,\lambda)] = ln[P(O_1|A_1)] + ln(a_{11\,or\,12}) + ln[P(O_2|A_{1\,or\,2})] + \cdots$$
$$\ldots + ln(a_{(11\,or\,12\,or\ldots(n-1)n)}) + ln[P(O_T|A_{1\,or\,2\,or\ldots N})] \quad (9)$$

Moreover, the complicated Gaussian probability distribution function is also transferred to simpler variant in comparison to formula 4.

$$ln[\,P(O_t|A_n)] = ln\left[\sum_{m=1}^{M} c_{nm}.\frac{1}{(2\Pi)^{\frac{D}{2}}.(\prod_{d=1}^{D}6_{nmd})^{\frac{1}{2}}}\; e^{-\sum_{d=1}^{D}\frac{(O_{td}-\mu_{nmd})^2}{26_{nmd}^2}}\right] \quad (10)$$

$$\Leftrightarrow ln[\,P(O_t|A_n)] = \sum_{m=1}^{M}\left[ln(c_{nm}) - \sum_{d=1}^{D}\frac{(O_{td}-\mu_{nmd})^2}{2.6_{nmd}^2} + ln\left(\frac{1}{(2\Pi)^{\frac{D}{2}}.(\prod_{d=1}^{D}6_{nmd})^{\frac{1}{2}}}\right)\right] \quad (11)$$

$$\Leftrightarrow ln[\,P(O_t|A_n)] = \sum_{m=1}^{M}\left[\sum_{d=1}^{D}(O_{td}-\mu_{nmd})^2\frac{-1}{2.6_{nmd}^2} + ln\left(\frac{c_{nm}}{(2\Pi)^{\frac{D}{2}}.(\prod_{d=1}^{D}6_{nmd})^{\frac{1}{2}}}\right)\right] \quad (12)$$

From the formula (12), Gaussian probability function is possible to approach the hardware implement as the below architecture as Fig.7. Obviously, only simple structures such as addition and multiplication are used in promoted structure, however, addition or multiplication loops are up to *D, M,* and *T* indexes.

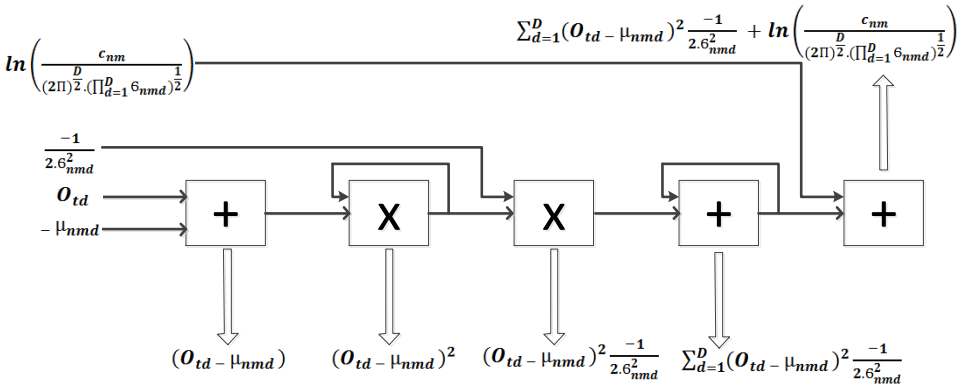

Fig. 7 Gaussian probability hardware approaching

From such circumstances, a complete HMM scheme is approached in hardware viewpoint possibly, however, this traditional way also has some troubles such loop controller, accuracy proportion, or real-time requirements with the large number of calculation steps. In order to cover all critical problems, a dynamic setting HMM hardware apply parallel and pipeline techniques is promoted in below sections.

III. Proposed Dynamic HMM hardware architecture

In our proposed architecture, both parallel and pipeline techniques are utilized to meet real-time requirement. Basing on two techniques, the traditional schemes described in previous section are adjusted as below sections.

3.1 Apply parallel technique in HMM hardware

By inheriting traditional HMM hardware approaches illustrated in the section 2.2, the issue of enormous number of recalled Gaussian probability distribution function is solved by parallel technique. Particularly, eight Gaussian probability structures named as GU unit in Fig. 8 are implemented. Correctly, one GU unit takes the Gaussian probability calculation of one state and one observation vector O_t. As the result, eight observation vectors corresponding to one state will be finished at the same time.

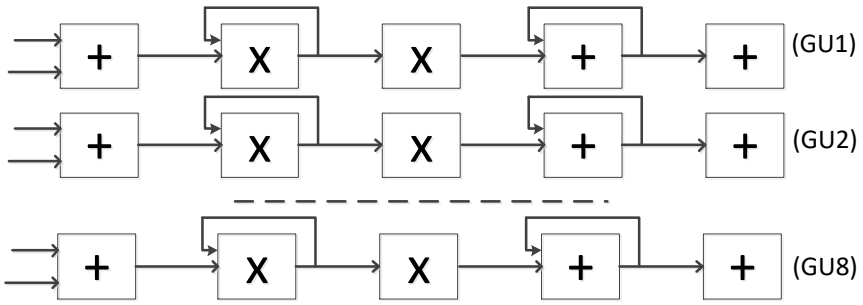

Fig. 8 Parallel technique to solve recalled enormous number of probability function

To be able to apply this parallel technique, eight data flows corresponding eight acoustic observation vectors have to be ready to read any time. Furthermore, the model information of each state obtained by training process is also been ready simultaneously. From such circumstances, internal memories are necessary to be instanced to HMM hardware as Fig. 9. Normally, parallel technique is applied to observation series instead of states for main reason so that the state number is fixed normally while the vector number of observation series are dynamic up to feature extraction methods.

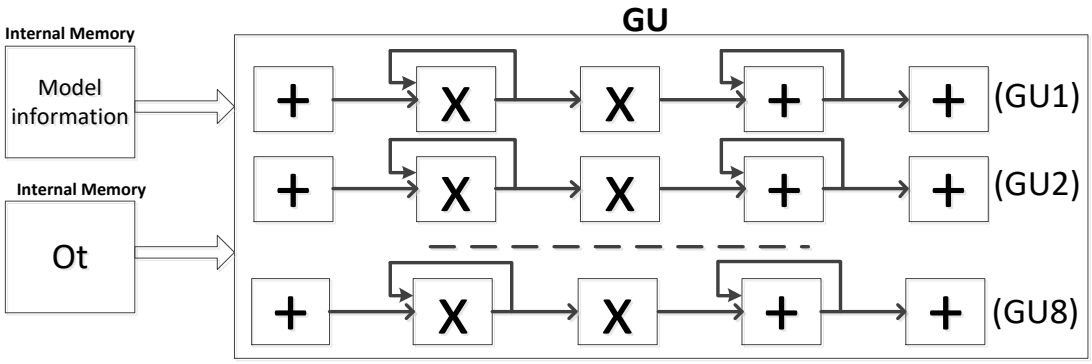

Fig. 9 Necessary internal memories containing input data to GU

After eight GUs finish operation, eight values of Gaussian probability functions are ready for decoding process. Therefore, in order to choose the largest probability from eight results, a decode hardware is implemented whenever receiving new eight data as Fig.10

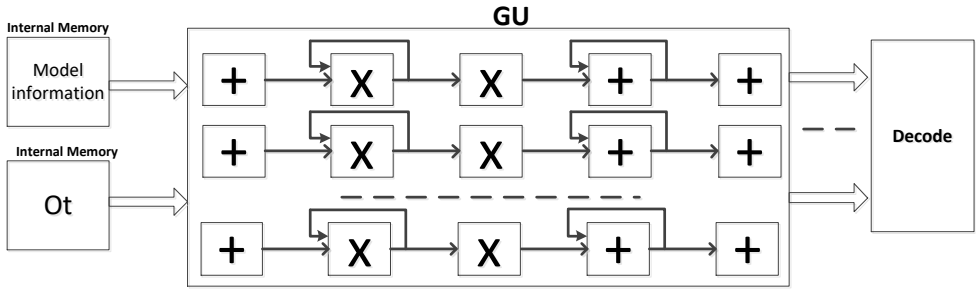

Fig.10 Architecture to decode outputs of GU

In order to describe the decode structure in detail, the data flow of decode architecture following to cover all probability as formula 5 and Fig. 5 instead of Viterbi algorithm is presented in the Fig.11 to be able to achieve high performance. After every recognition cycle, the maximum probability is confirmed again. For instance in the Fig. 11

(a), the maximum probability is obviously at O_8 observation following the statement (13)

$$P_1(O_8|A_1) = P(O_1|A_1).a_{11}.P(O_2|A_2)\dots a_{11}.P(O_8|A_1) \quad (13)$$

However, at the second recognition cycle when 8 observation comprising O_1 to O_8 finish calculating at the state A_2. The maximum probability can be changed or unchanged depending on result of comparisons among previous probabilities as formula (14). As the example result in the Fig.11 (b), the maximum probability is still kept.

$$P_2(O_8|A_1 \; and \; A_2) = Max\{P(O_1|A_1).(a_{11\;or\;12}).P(O_2|A_{1\;or\;2}) \ldots (a_{11\;or\;12\;or\;22}).P(O_8|A_{1\;or\;2})\}$$
$$= P(O_1|A_1).a_{11}.P(O_2|A_2)\ldots a_{11}.P(O_8|A_1) \quad (14)$$

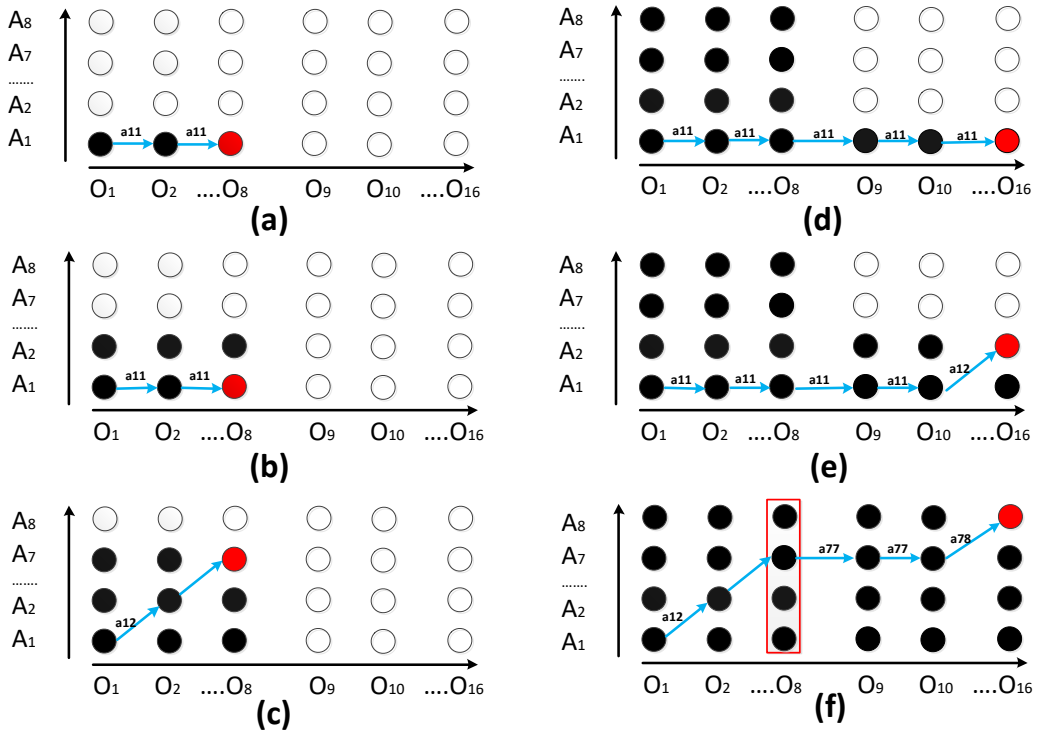

Fig. 11 Eight input data processing in decode structure

Continuously, the maximum value of Gaussian probability is updated after recognition cycle from Fig. 11 (a) to Fig. 11 (f) until that the probability between final acoustic observation vector and final state are calculated. In order to obtain the maximum value for each cycle correctly, all eight data need to be utilized to compare absolutely. Revealing the Fig. 11 to approach the hardware architecture is really challenge, however, the Fig. 12 and Fig. 13 as other aspect is illustrated to apply the hardware possibly but still keeping the algorithm correctly.

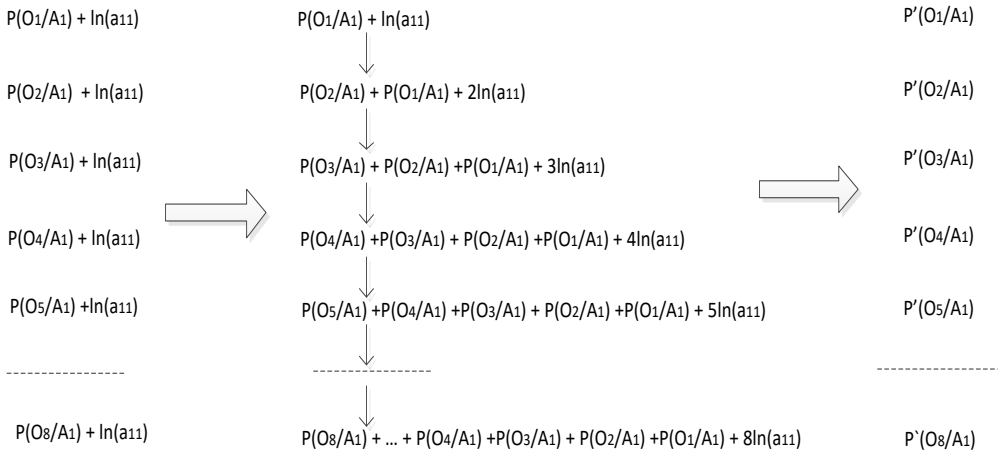

Fig. 12 Decoding algorithm step

where $P(O_t/A_1)$ is Gaussian probability result of every GU unit with state A_1, *Adding* $ln(a_{11})$ is used for applying hardware architecture more easily and $P'(O_t/A_1)$ is maximum Gaussian probability result at time when acoustic observation O_t and state A_1 are calculated.

After getting 8 values of probabilities corresponding state A_1 at the first recognition cycle, these values are added together step by step. The final results symbolized by $P'(O_t/A_1)$ are the maximum probabilities corresponding to O_t with state A_1 following below statement

$$P'(O_t|A_1) = P(O_1|A_1) + ln(a_{11}) + P(O_2|A_1) + \ldots + P(O_t|A_1) + ln(a_{11})$$
$$= P(O_1|A_1) + P(O_2|A_1) + \ldots + P(O_t|A_1) + t\, ln(a_{11}) \quad (15)$$

Continuously, when the next eight values of the second recognition cycle are updated, the next calculation steps which are not same in the first recognition cycle are described in the Fig. 12. In order to get the maximum probability for O_t at state A_2, adding previous maximum probability $P'(O_{t-1}/A_2)$ *or* $P'(O_{t-1}/A_1)$ to $P'(O_t/A_2)$ is decided by the maximum value of comparison between $P'(O_{t-1}/A_2)$ and $P'(O_{t-1}/A_1)$. Next, the probability values of second cycle are updated step by step to the final probability $P'(O_8/A_2)$ with the same way.

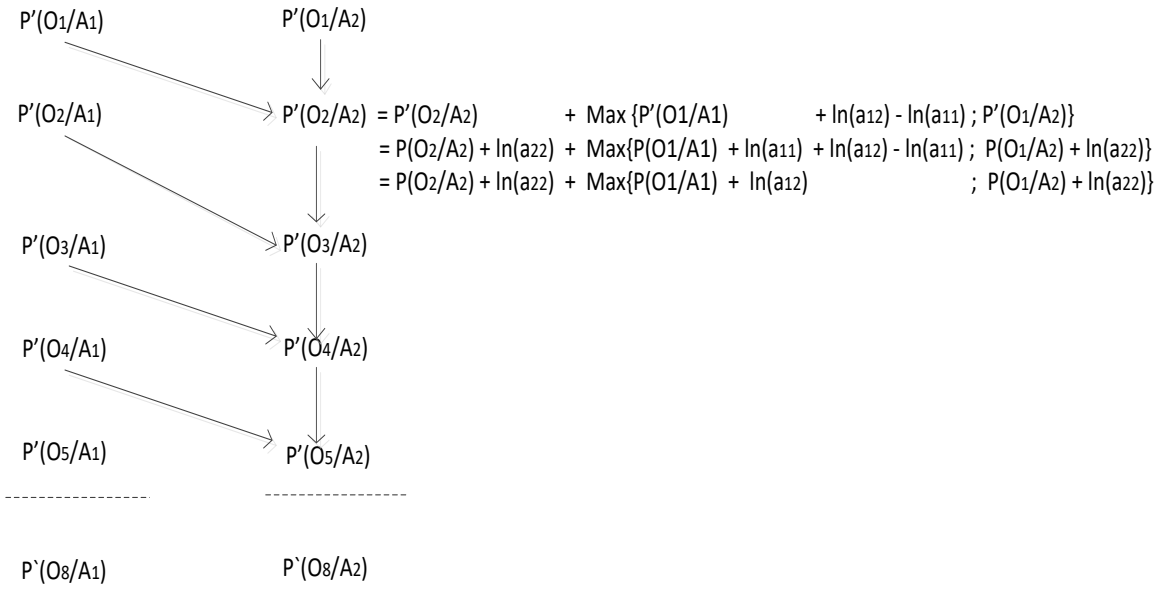

Fig. 13 Update the maximum probability at the second recognition cycle

Hence, the new values of recognition cycle are updated at the same way as the second recognition cycle. Following these calculations, the hardware approached to HMM decode architecture is described in the Fig. 14 in which only two statements comprising addition and comparison are integral.

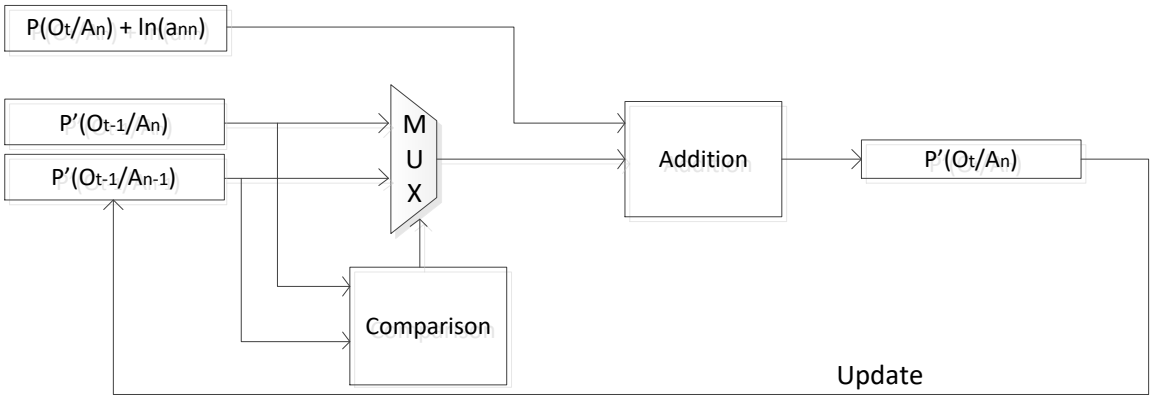

Fig. 14 Hardware approach for decoding

3.2 Apply pipeline technique

From overall of decode architecture, three main tasks including preparing data by writing to internal memory, probability calculation by GU unit, and decoding task also need to be implemented continuously. In these tasks, the GU unit takes the most time by many statements comprising addition and multiplication obviously. As the result, in order to enhance the speed of recognition process, the pipeline technique is applied in both GU unit and among three main tasks of complete recognition process.

a. Pipeline inner GU unit

As discussing in upper sections, Gaussian probability calculation step was chosen to improve by many researches mainly because its time cost is critical issue. In this work, pipeline technique is applied to speech up the GU unit as the state machine in the Fig. 15.

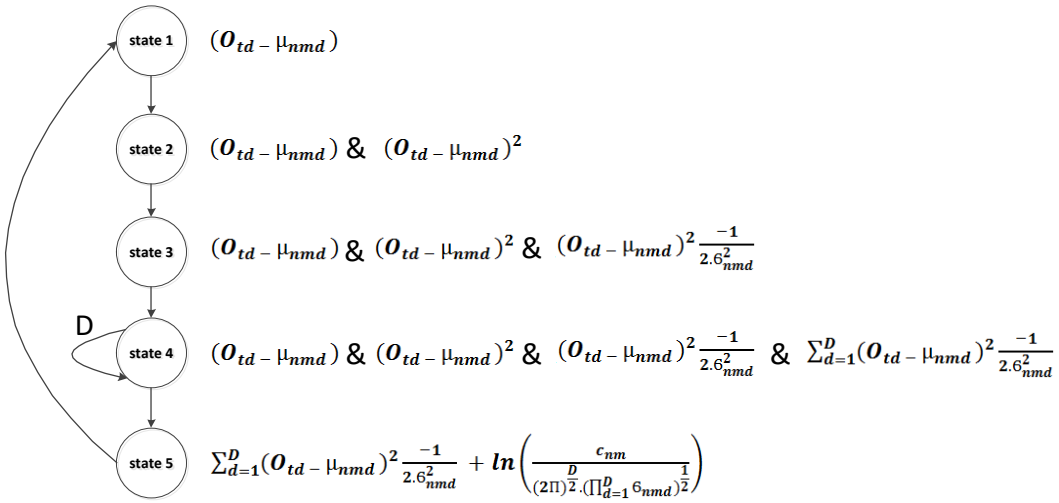

Fig. 15 State machine for applying pipeline technique in GU unit

Assuming that S_{GU} is the same cycle clock number for one state in Fig. 5, the below statement is used to calculate the number of clock cycle on GU unit symbolized as *ClockNumGU* if *D* index is clear before.

$$ClockNumGU = (4 + D)S_{GU} \quad (16)$$

b. Pipeline for complete recognition process

According to Fig. 10, complete recognition process comprises three main tasks in which the GU takes the most time estimated by formula 16. In order to continuous enhancing the speed of recognition process, the parallel technique is applied again as the state machine in Fig. 16

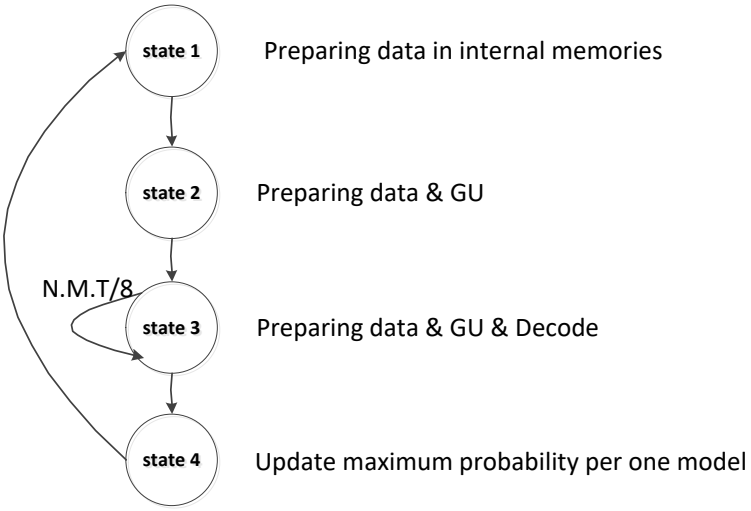

Fig. 16 State machine for recognition process

If assuming that time cost per each state in Fig.16 symbolized as S_{RP} is same, the below statement presents the total time for recognition process as *ClockNumRP*

$$ClockNumRP = (3 + N.M.T/8)S_{RP} \quad (17)$$

From formulas (16) and (17), the total time for recognition process per one model as *ClockNumRP* is presented

$$ClockNumRP = (3 + N.M.T/8)(4 + D)S_{GU} \quad (18)$$

As the result, if there are W discrete words corresponding to W models, the total time for complete recognition process is presented by below statement

$$ClockNumRP = W(3 + N.M.T/8)(4 + D)S_{GU} \quad (19)$$

Conclusion, the total time is depend on the word number, state number, mixture number, acoustic feature vector number and GU delay. In proposed hardware, these numbers can set up flexibly at first time before starting recognition process.

3.3 Complete recognition process

In full ASR system, feature extraction is one of major steps for both training and recognition processes. In order to estimate performance of recognition process in complete ASR system correctly, a feature extraction method commonly named Mel Frequency Cepstral Cofficient (MFCC) method, is also developed in both software and hardware in our work.

To be able to approach hardware, MFCC configuration is illustrated as the table I and Fig.17 in detail.

Table I. MFCC hardware configuration

MFCC configuration	Value
Energy coefficient number	1
Pre_emphasis coefficient	31/32
Sample number per frame	80
FFT number	80
Filter bank number	23
Cepstrum number	12
Delta level	Level
MFCC vector	26

After connecting MFCC to HMM base recognition scheme, a complete recognition process is built perfectly as the Fig. 17 to perform an ASR application.

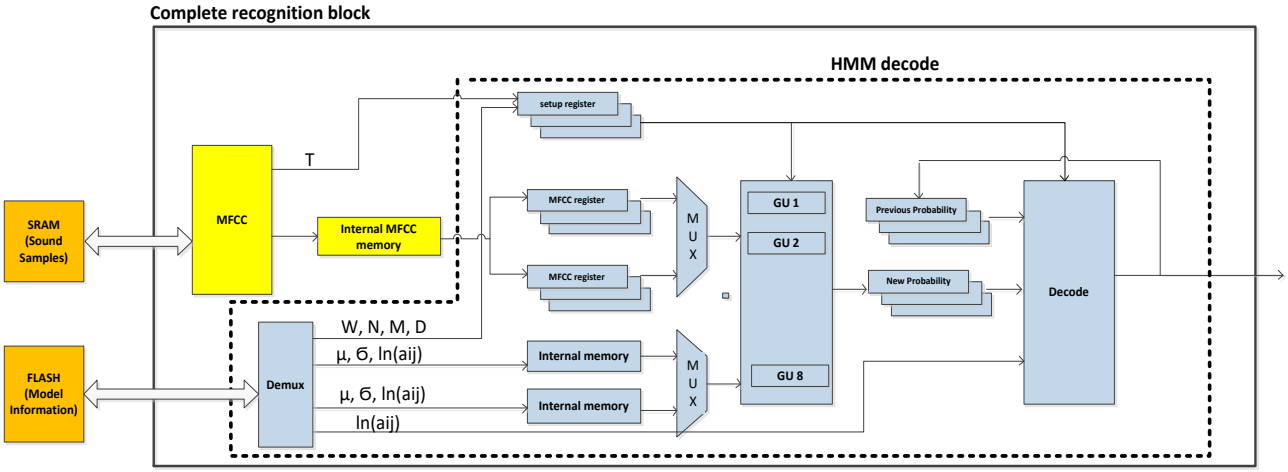

Fig.17 Complete recognition process

IV. Experiments and results

Approaching ASIC base design, this architecture is followed the ASIC design flow as the Fig. 18. Following the flow as Fig 18, there are three times to verify the functions of design. First, it is RTL verification on RTL source code (Verilog language). Next is gate level net list verification after synthesis step. The first two times only confirm the hardware functions called as dynamic verification and detect the mismatches between hardware and software development. Finally, that is post-lay out verification including the real timing information after physical design steps with TSMC 130nm technology in Place & Route process. In three tasks, the co-simulation environment is always used to satisfy as the Fig. 19.

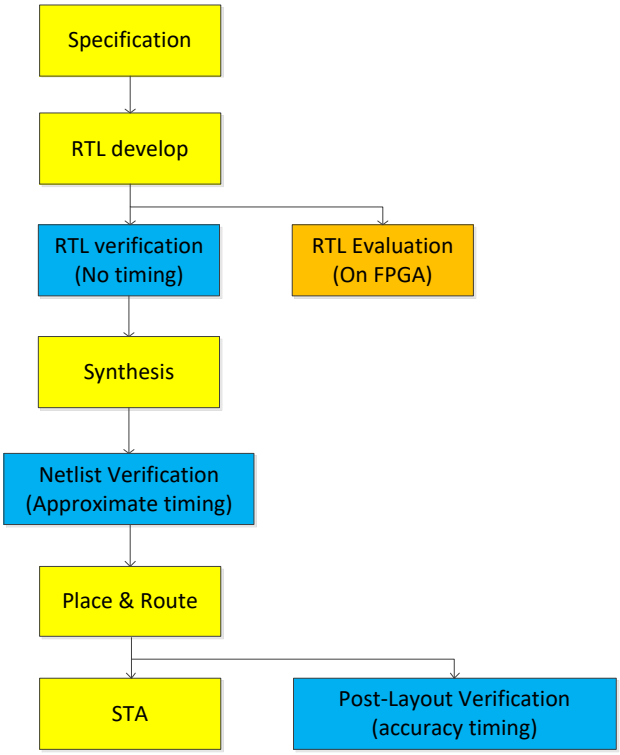

Fig.18 ASIC base design flow

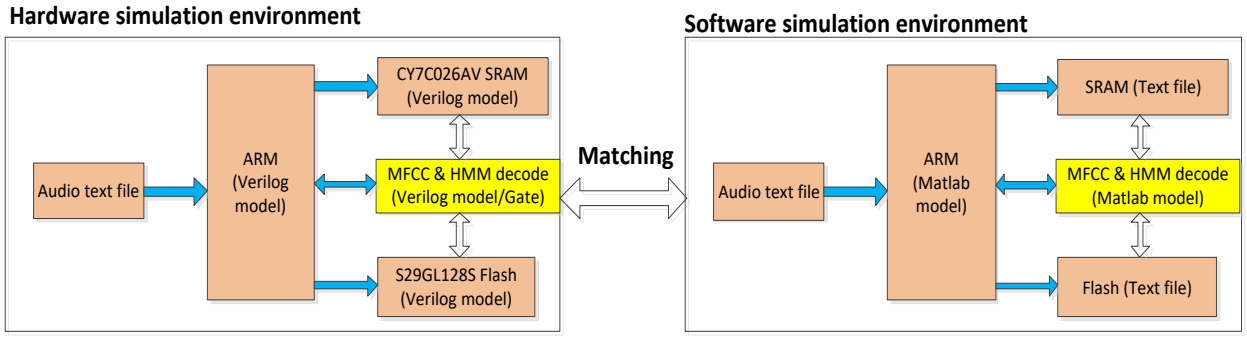

Fig. 19 Co-simulation environment

The corresponding result with 50 Vietnamese discrete words is verified to confirm the high probability as Table II. Particularly, there are total 100 samples per word and 5000 samples for 50 discrete words. And 4500 words are used to train before while 5000 words are recognized to obtained final probability on both RTL simulation and post-layout Gate level netlist simulation.

Table II. Timing consumption for verificaiton

Verification step	Timing Property	Simulation time average /sample	Probability
Software Verification	No Timing	1 second	97 %
RTL Verification	No timing	3 minutes	96.2 %
Pre-layout Verification	Approximate timing	7 minutes	96.2 %
Post-Layout Verification	Accuracy timing	30 minutes	96.2 %

Table III illustrates the limitation of dynamic configuration of HMM architecture. According to such constrain, a wide range of HMM configuration can adapt a huge variety of application not only in ASR but also for others.

Table III. Limitation of dynamic configuration

Maximum word	1000
State number limitation	0 → 16
Mixture number limitation	0 → 8
Acoustic feature vector dimension	0 → 32
Acoustic feature vector number	No Limit

The Place & Route results also confirm the physical property of design as Fig.20. In Fig. 20, three are three hard macro comprising MFCC memory that is used to store MFCC results and others used to store ready model information data for GU unit. Therefore, other internal memories are synthesized from standard cells. After Place & Route step, power consumption is calculated exactly and reported as Table IV.

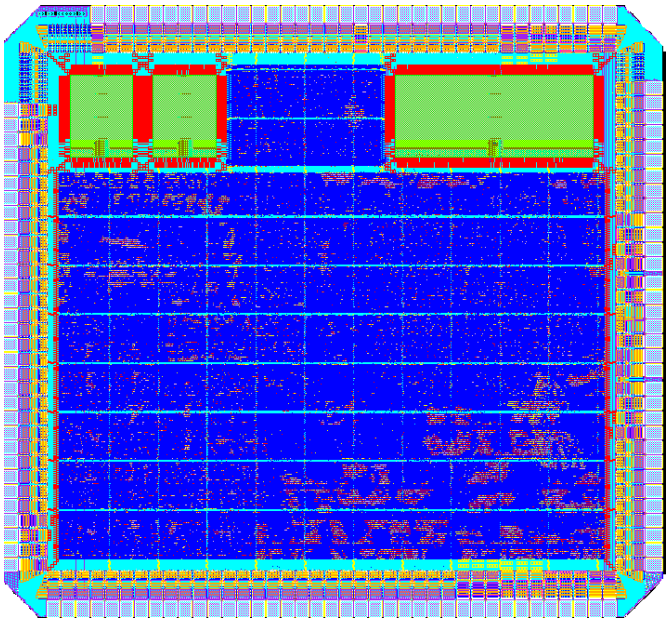

Fig. 20 Place & Route result

Table IV. Power consumption summary

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	%
io_pad	3.6309	9.0035	2.3121e+06	12.6367	27.43%
memory	6.2802e-02	4.9302e-03	1.5400e+08	0.2217	0.48%
black_box	0.0000	0.0000	0.0000	0.0000	0%

clock_network	3.2084	6.8134	3.0023e+07	10.0518	21.82%
register	21.0930	8.3054e-03	4.7970e+08	21.5803	46.85%
sequential	0.0000	0.0000	0.0000	0.0000	0%
combinational	0.1683	0.6115	7.9106e+08	1.5709	3.41%
Total	28.1634 mW	16.4416 mW	1.4571e+09 pW	46.0615 mW	100%

Furthermore, in order to ensure the successful evaluation on real chip, evaluation on FPGA is implemented. In order to match with the simulation environment, an application board is also implemented as Fig.21. To be more specific, FPGA models behavior of target IC while SRAM and ARM trigger FPGA active and control audio data into target IC. After successful fabrication, real chip known as iHear Tech is replaced the place of FPGA and evaluation is rerun and gain the same results that confirms successful design.

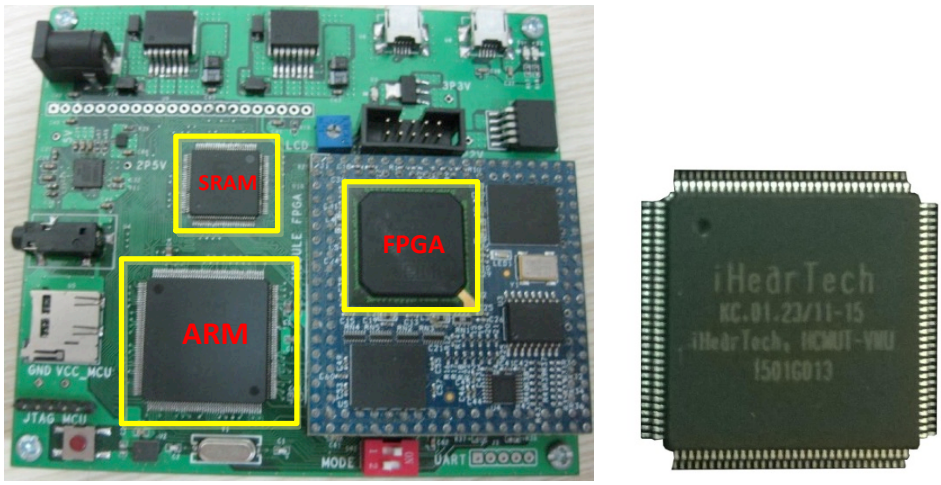

Fig 21 Application board integrated EP2C50F484C8 Cyclone II FPGA module

In order to compare with other designs, table V is created to obtain exact estimation among different criteria. Through such comparison, proposed hardware architecture experience high probability in comparison to [1] and [6] with small number of word. While achieved frequency is similar for all surveys, timing consumption of proposed architecture presents superior to other design. In addition, almost ASR systems approach both software and hardware that experience a drawback as regards software components. This does not happen in proposed solution in which complete feature extraction and HMM model are implemented by hardware followed ASIC flow and perform well-done chip.

Table V. Power consumption summary

Authors	The number of words	Frequency	Timing consumption	Technology	Range of configuration	Probability (%)
Ge Zhang [1]	100	Depend on FGPA kit	0.2 s	C language applied on NIOS II	Dynamic software configuration	93.76
Octavian Cheng [2]	993	120 MHz only for GMM	0.54 s	Hardware (GMM)– software co-processing systems	Dynamic software configuration	93.16
Richard Veitch [3]	5000	120 MHz	-	Virtex 5 SX95T FPGA	3 states, 8 mixtures	80
Hui Geng [4]	256	24 MHz	2.5 s	M1 ARM7-TDMI (Master) and FPGA	358 state, 8	

				M7A3P1000 (Slave for HMM)	mixture	
Mohammad Mosleh [6]	24	-	1.67 ms	FPGA XC4VLX15	6 states	83.92
G. Tamulevičius [7]	-	6 KHz	-	Xilinx ML402	-	86
Peng Li [8]	800	10MHz for MCU and 10 MHz for co-processor	22.3 us for MCU and 9.7 um for co-processor	MCU (32bit RISC (ARM)) and Co-processor (Xinlinx XCV2000)	Dynamic software configuration 358 state, 3 mixture	82
Proposed design	50	100 MHz	2 ms	130nm TSMC	State from 1 to 6 Mixture from 1 to 8	96.2

V. Conclusion

In this research, a dynamic hardware architecture of complete ASR is proposed in which a wide range of HMM configuration has experienced high performance in comparison to other designs. According to ASIC design flow and process of evaluation FPGA, well done chip that support speech recognition not only for Vietnamese people but also for any languages is fabricated successfully. In the future, a hybrid architecture ANN-HMM will be proposed by approaching hardware design to be able to improve performance of ASR.

References

[1] Ge Zhang, Jinghua Yin-, Qian Liu, Chao Yang, "A Real-Time Speech Recognition System Based on the Implementation of FPGA", Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), October 2011.

[2] Octavian Cheng, Waleed Abdulla, Zoran Salcic, "Hardware–Software Codesign of Automatic Speech Recognition System for Embedded Real-Time Applications", IEEE Transactions On Industrial Electronics, Vol. 58, No. 3, march 2011, pp. 850 – 859.

[3] Richard Veitch, Louis-Marie Aubert, RogerWoods, and Scott Fischaber, "FPGA Implementation of a Pipelined Gaussian Calculation for HMM-Based Large Vocabulary Speech Recognition", International Journal of Reconfigurable Computing, Volume 2011, No. 4, 2011.

[4] Hui Geng1, Yiyu Shi1, Ming Dong2, Runsheng Liu3, "A Master-Slave SoC Structure for HMM Based Speech Recognition", VLSI Design, Automation, and Test (VLSI-DAT), June 2012.

[5] Young-kyu Choi, Kisun You, and Wonyong Sung, "FPGA-Based Implementation of A Real-Time 5000-word Continuous Speech Recognizer", Signal Processing Conference, April 2008

[6] Mohammad Mosleh1, Saeed Setayeshi2, Mohammad Kheirandish3 "Hardware Implementing of a New Decoding Algorithm HGSCA for HMM Based Speech Recognition Systems", Journal of Basic and Applied Scientific Research, 2012, pp 12124-12133.

[7] G. Tamulevičius, V. Arminas, E. Ivanovas, D. Navakauskas, "Hardware Accelerated FPGA Implementation of Lithuanian Isolated Word Recognition System", Elektronika Ir Elektrotechnika Journal, Vol 99, No.3, 2010, pp. 57-62.

[8] Peng Li, Hua Tang, Weiqian Liang, "Low Power Embedded Speech Recognition System Based On A MCU And A Coprocessor", Acoustics, Speech and Signal Processing, ICASSP 2009.

[9] Md Salam, Dzulkifli Mohamad, and Sheikh Salleh, "Malay Isolated Speech Recognition Using Neural Network: A Work in Finding Number of Hidden Nodes and Learning Parameters", The International Arab Journal of Information Technology, Vol. 8, No. 4, October 2011.

[10] Shing-Tai Pan, Chih-Chin Lai, Bo-Yu Tsai, "The Implementation Of Speech Recognition Systems On FGPA-Based Embedded Systems With SOC Architecture" International Journal of Innovative Computing, Information and Control Volume 7, Number 11, November 2011.