DOI: 10.3745/KIPSTA.2009.16A.6.411

퍼지 벡터 양자화를 위한 대규모 병렬 알고리즘

Luong Van Huynh[†] · 김 철 홍[††] · 김 종 면[†††]

요 약

퍼지 클러스터링 기반 벡터 양자화 알고리즘은 퍼지 클러스터링 분석이 벡터 양자화 프로세스 초기단계에서 초기화에 덜 민감하게 하기 때문에 데이터 압축 분야에서 널리 사용되어 왔다. 하지만, 퍼지 클러스터링 처리는 훈련 벡터 공간에 포함된 불확실한 양적 공식의 복잡한 프레임워크 때문에 상당한 계산량이 요구된다. 이러한 상당한 계산량 부하를 극복하기위해 본 논문은 4,096 프로세싱 엘리먼트로 구성된 어레이 아키텍처를 이용하여 퍼지 벡터 양자화 알고리즘의 병렬 구현을 제안한다. 제안하는 병렬 구현은 4,096 프로세싱 엘리먼트를 이용하여 클러스터링 프로세스 동안 효과적인 벡터 할당 정책을 적용함으로써 계산적으로 효율적인 솔루션을 제공한다. 모의실험 결과, 제안한 병렬 구현은 기존의 다른 어레이 아키텍처를 이용한 구현보다 성능 및 효율 측면에서 상당한 향상을 보였다. 또한동일한 130nm 기술에서 제안한 병렬 구현은 오늘날의 ARM이나 TI DSP 프로세서를 이용한 구현과 비교하여 약 1000배의 성능 향상 및 100배의 에너지 효율 향상을 보였다. 이 결과들은 향상된 성능 및 에너지효율에서 제안한 병렬 구현의 잠재가능성을 입증한다.

키워드 : 퍼지 벡터 양자화, 퍼지 클러스터링, 이미지 압축, 병렬처리 아키텍처, 임베디드 프로세서

A Massively Parallel Algorithm for Fuzzy Vector Quantization

Luong Van Huynh[†] · Cheol Hong Kim[††] · Jong-Myon Kim[†††]

ABSTRACT

Vector quantization algorithm based on fuzzy clustering has been widely used in the field of data compression since the use of fuzzy clustering analysis in the early stages of a vector quantization process can make this process less sensitive to its initialization. However, the process of fuzzy clustering is computationally very intensive because of its complex framework for the quantitative formulation of the uncertainty involved in the training vector space. To overcome the computational burden of the process, this paper introduces an array architecture for the implementation of fuzzy vector quantization (FVQ). The arrayarchitecture, which consists of 4,096 processing elements (PEs), provides a computationally efficient solution by employing an effective vector assignment strategy during the clustering process. Experimental results indicatethat the proposed parallel implementation providessignificantly greater performance and efficiency than appropriately scaled alternative array systems. In addition, the proposed parallel implementation provides 1000x greater performance and 100x higher energy efficiency than other implementations using today's ARMand TI DSP processors in the same 130nm technology. These results demonstrate that the proposed parallel implementation shows the potential for improved performance and energy efficiency.

Keywords : Fuzzy Vector Quantization, Fuzzy Clustering, Image Compression, Parallel Processing Architecture, Embedded Processors

1. Introduction

Vector quantization (VQ) is a classical quantization technique that allows the modeling of probability density functions by the distribution of prototype vectors. VQ identifies an input vector with a member of a codebook which is a collection of codeword vectors. The encoding process replaces each constituent input block with its corresponding VQ codeword index. However, the traditional VQ method makes this process more sensitive to initialization for achieving the quality of vector quantizers. Fuzzy clustering provides a solution for designing a high quality of codebook by allowing the assignment of each training vector to multiple clusters in the early stages of the iterative codebook design process [1-2], resulting in

※ This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2008-331-D00459).
† 준 회 원 : Research & Development Center, BNF Technology Inc., Deajeon, Korea, Researcher
†† 종신회원 : 전남대학교 전자컴퓨터공학부 교수
††† 정 회 원 : 울산대학교 컴퓨터정보통신공학부 교수(교신저자)
논문접수 : 2009년 8월 31일
수 정 일 : 1차 2009년 10월 26일
심사완료 : 2009년 10월 27일

the reduction of dependence for the resulting codebook on the random initial codebook selection.

Fuzzy clustering based vector quantization algorithms [1-6] have been introduced to achieve the quality of vector quantizers by allowing the assignment of each training vector to multiple clusters in the early stages of the iterative codebook design process. The goal of fuzzy clustering is to allow each feature vector to belong to more than one cluster with different membership degrees (between 0 and 1). During the process, the fuzzy assignment is allocated at the earlier training stage. This guarantees that all input vectors are included in the formation of a new codebook represented by all the cordword-coupled with weights. The well-known fuzzy clustering is Fuzzy C-Means (FCM) [3] which allows overlapping clusters with partial membership ofindividuals in the clusters where c is a priori chosen number of clusters. However, the process of fuzzy clustering is computationally very intensive because of its sophisticated framework for the quantitative formulation of the uncertainty involved in a training vector space.

To overcome the computational burden of the complex process, this paper introduces a parallel implementation of the fuzzy vector quantization (FVQ) algorithm using a representative data parallel architecture which consists of 4,096 processing elements (PEs). Thus, the proposed parallel approach provides a computationally efficient solution by employing an effective vector assignment strategy for the transition from soft to crisp decisions during the clustering process. This paper evaluates the impact of the parallel FVQ implementation on both the processing performance and energy efficiency. In addition, this paper compares the proposed parallel implementation to other implementations using commercial processors. Experimental results show that the parallel implementation provides about 1000x greater performance and 100x higher energy efficiency than other implementations using commercial ARM [7] and TI DSP [9].

The rest of this paper is organized as follows. Section II presents fuzzy clustering based vector quantization for data compression, a brief introduction of the baseline data parallel architecture, and methodology infrastructure for the parallel FVQ implementation. Section III introduces our proposed parallel FVQ implementation using the specified data parallel architecture. Section IV analyzes and compares the performance and energy efficiency for the parallel implementation and the implementations using ARM families. Section V concludes this paper.

2. Background Information

2.1 Fuzzy Vector Quantization

This section briefly reviews key features of vector quantization (VQ) and fuzzy clustering [8]. VQ is defined as a mapping of k-dimensional vectors in vector space R_k on to a finite set of vectors $V=\{v_i \ i=1,\cdots,N\}$, where N is the size of the codebook. Each vector $y_i=(y_0,...,y_{k-1})$is called a code vector or codeword. Only index i of the resulting code vector is sent to the decoder. At the decoder, an identical copy of the codebook is retrieved as the encoder by a simple table-lookup operation. The compression ratio depends on the cardinality of the codebook, usually much smaller than that of the input domain.

In this paper, we study fuzzy clustering based vector quantization to design a high quality of codebook by allowing the assignment of each training vector to multiple clusters in the early stages of the iterative codebook design process. Fuzzy c-means (FCM) [3] is one of the most well-known methodologies in clustering analysis. The FCM clustering is an iterative algorithm of clustering technique which produces optimal c partitions and centers $V=\{v_1, v_2,\cdots, v_c\}$. Suppose the unlabelled data set $X=\{x_1, x_2,\cdots, x_n\}$ be the pixel intensity where n is the number of image pixels to determine their memberships. The FCM clustering performs to partition the data set X into c clusters, and the objective function of the standard FCM is defined as follows:

$$J_m(U,V)=\sum_{i=1}^{c}\sum_{k=1}^{n}u_{ik}^{m}d^2(x_k,v_i) \tag{1}$$

where $d(x_k,v_i)$ represents the distance between the pixel x_k and centroid v_i, along with constraint $\sum_{i=1}^{c}u_{ik}=1$, and the degree of fuzzification $m\geq 1$.

The data point x_k belongs to a specific cluster $v_i$which is given by the membership value u_{ik} of the data point to that cluster. Local minimization of the objective function $J_m(U,V)$ is accomplished by repeatedly adjusting the values of u_{ik} and v_i according to the following equations:

$$u_{ik}=\left[\sum_{j=1}^{c}\left(\frac{d^2(x_k,v_i)}{d^2(x_k,v_j)}\right)^{1/m-1}\right]^{-1} \tag{2}$$

$$v_i=\frac{\sum_{k=1}^{n}u_{ik}^{m}x_k}{\sum_{k=1}^{n}u_{ik}^{m}},\quad 1\leq i\leq c \tag{3}$$

As $J_m$is iteratively minimized, v_i becomes more stable. Iteration of pixel clustering is terminated when the termination measurement $\max_{1\le i\le c}\left\{\left\|v_i^{(t)}-v_i^{(t-1)}\right\|\right\}<\varepsilon$ is satisfied, where $v_i^{(t)}$ is new centers, $v_i^{(t-1)}$ is previous centers, and ε is the predefined termination threshold.

In the codebook design using FVQ, the input model consists of a set of training vectors X which are weighted with u_{ik}. In addition, the training vectors are mapped into clusters which are represented by codewords V. After several iteration processes, a high quality of codebook is generated. However, this demands tremendous computational requirements. To overcome this problem, we prefer to implement a parallel FVQ using a representative data parallel architecture which consists of 4,096 processing elements (PEs).

2.2 Data Parallel Architecture

Among many computational models available for imaging applications, single instruction multiple data (SIMD) processor arrays are promising candidates for 2-dimensional image processing algorithms since they often employ thousands of processing elements (PEs) while possibly distributing and co-locating PEs with the data I/O to minimize storage and data communication requirements. (Figure 1) shows the baseline data parallel architecture along with its interconnection network. When data are distributed, the processing elements (PEs) execute a set of instructions in a lockstep fashion. With 4x4 pixel sensor sub-arrays, each PE is associated with a specific portion (4x4 pixels) of an image frame, allowing streaming pixel data to be retrieved and processed locally. Each PE has a reduced instruction set computer (RISC) datapath with the following minimum characteristics:

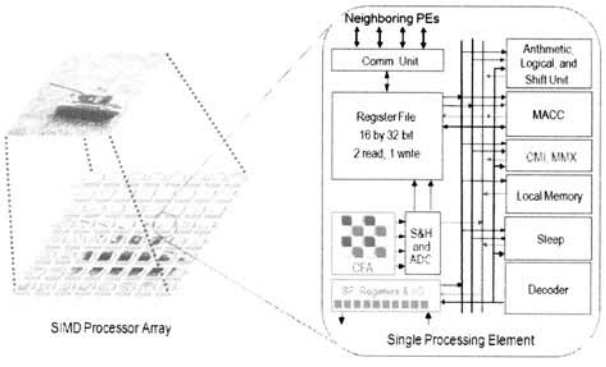
- ALU-computes basic arithmetic and logic operations,
- MACC-multiplies 32-bit values and accumulates into a 64-bit accumulator,
- Pixel unit-samples pixel data from the local image sensor array,
- ADC unit-converts light intensities into digital values,
- Three-ported general-purpose registers (16 32-bit words),
- Small amount of local storage (256 32-bit words),

Nearest neighbor communications through a NEWS (north-east-west-south) network and serial I/O unit.

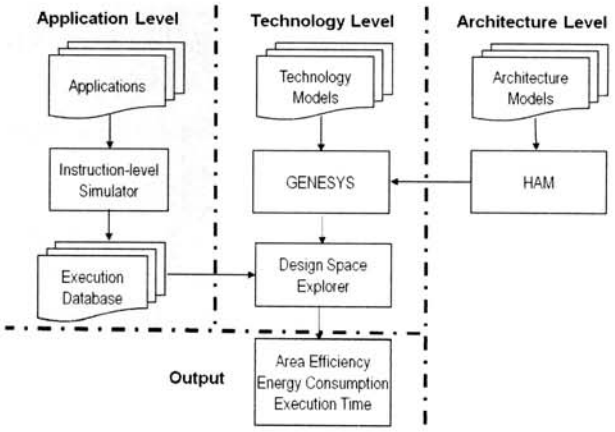
Using the baseline data parallel architecture, we implement a parallel FVQ algorithm to meet the computational requirements. The next section presents a simulation methodology to evaluate the performance of the parallel FVQ implementation.

2.3 Methodology Infrastructure

(Figure 2) shows our methodology infrastructure which is divided into three levels: application, architecture, and technology. At the application level, we use an instruction-level simulator to profile execution statistics, such as issued instruction frequency, PE utilization, and PE memory usage. At the architecture level, we use the heterogeneous architectural modeling (HAM) of functional units for processor arrays, proposed by Chai et al. [10]. The design parameters are then passed to the technology level. At the technology level, we use the Generic System Simulator (GENESYS) [11] to calculate technology parameters (e.g., latency, area, power, and clock frequency) for each configuration. Finally, we combine the database (e.g., cycle times, instruction latencies, instruction counts, area, and power of the functional units), obtained from the application, architecture, and technology levels, to determine execution times, area efficiency, and energy efficiency for each case.



(Figure 1) Baseline data parallel architecture



(Figure 2) Simulation methodology

3. Parallel Implementation of Fuzzy Vector Quantization

This section presents a parallel FVQ algorithm implemented on the specified data parallel architecture, which consists of fuzzy c-means (FCM) clustering for a codebook design and vector quantization (VQ) for image compression.

3.1 Parallel Fuzzy C-Means Clustering Algorithm

The pseudocode of the parallel FCM algorithm is given in (Figure 3), along with a pictorial description of the FCMmechanism and of the communication patterns for ahypothetical 16 node SIMD array system. Each system node is directly interfaced to a 4 x 4 pixel matrix. In step 1, each node computes distance between input pixels and the current center to determine their membershipvalues, new centers, and the termination measurement value. If the termination measurement value of distortion is less than the threshold value, current codeword will be replaced with the new center value. In step 2, the 16 components of the membership values are passed to the next neighbor in the same row; the new termination measurement value is computed; and then it is compared against the threshold value toreplace with the new center value in case the termination value is less than the threshold. This process is iterated until every node in the same row has been visited by the membership values. When the row completes, the membership values are transferred vertically to the next row in step3, and the same process is iterated along the row in step 4. A key enabling role is played by the toroidal structure of the interconnection network, which enables the communications among the nodes in the opposite of the mesh.

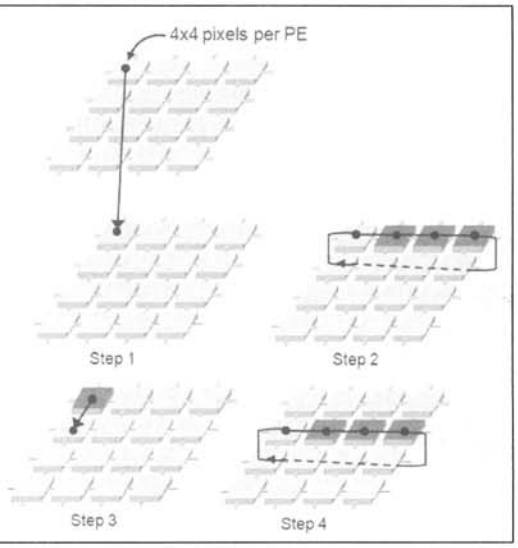
3.2 Parallel Vector Quantization Algorithm

Using the codebook generated by the FCM algorithm in Section 3.1, the parallel encoding operation of a vector quantizer is implemented. The pseudocode of the parallel encoding algorithm is given in (Figure 4) As a first step, each node computes the distortion between the input blockand the codeword. In a second step, the 16 components of the input block, the distortion value, and the index of the codeword are passed to the neighbor in the same row of the PE array. Then, the new distortion value is computed, and compared against the received distortion value. If a new minimum value is found, then the computed distortion and index values are will replace the

```
REPEAT for all rows
  REPEAT for all columns
    Transfer input block, current distortion, and current index to the eastern node;
    Calculate local distortion between input block and codeword;
    IF local distortion < current distortion THEN
        current index = local index;
        current distortion =local distortion;
    END if
  END repeat
  Transfer input block, current distortion, and current index to the southern node;
END repeat
Output current indexes;
```

(Figure 4) Parallel VQ encoding algorithm

```
REPEAT for all rows
  REPEAT for all columns
    Transfer current membership values to the eastern node;
    Calculate distances, new centers, and new membership
    values, and the termination measure of distortion;
    IF termination measure of distortion < threshold THEN
        current codeword = new center value;
    END if
  END repeat
  Transfer current membership values to the southern node;
END repeat
Output current codewords;
```



(Figure 3) Parallel fuzzy C-means clustering algorithm

received ones, and transferredto the next node of PEs. This process is iterated until every node in the same row has been visited by the input block. When the row completes, the data are transferred vertically to the next row, and the same process is iterated along the row.

4. Simulation Results

In this section, the performance evaluation of the parallel FVQ implementation is presented. We evaluated the parallel implementation with the following parameters: (1) the degree of fuzzification m=2, (2) the termination threshold E=0.0001, and (3)the number of codewords C=2 4 8; 16; 32; 64; 128; 256 (codeword size 4x4 pixels). In addition, a cycle-accurate simulator was used to simulate and evaluate the performance of the parallel FVQ with eight different codewords, where the parallel FVQ algorithms were developed in their respective assembly languages for the PE array system. In this study, the image size of 256x256 pixels was used. For a fixed 256x256 pixel system, the number of 4,096 PEs is used because each PE contains 4x4 pixels. <Table 1> summarizes the parameters of the PE array system configuration.

The metrics of execution time, sustained throughput, and energy efficiency of each case form the basis of the study comparison, defined in <Table 2>.

where C is the cycle count, f_{ck} is the clock frequency, O_{exec} is the number of executed operations, U is the PE utilization, and N_{PE} is the number of processing elements.

〈Table 1〉 PE array system parameters

Parameter	Value
Number of PEs	4,096
Pixels/PE	16 (4x4)
Memory/PE [word]	256 [32-bit word]
VLSI Technology	130 nm
Clock Frequency	133 MHz
Interconnection Network	Torus
intALU/intMUL/Barrel Shifter/intMACC/Comm	1 / 1 / 1 / 1 / 1

〈Table 2〉 Summary of performance evaluation metrics

Execution Time	Sustained Throughput	Energy Efficiency
$t_{exec} = \frac{C}{f_{ck}}$	$\eta_E = \frac{O_{exec} \cdot U \cdot N_{PE}}{t_{exec}} [\frac{Gops}{sec}]$	$\eta_E = \frac{O_{exec} \cdot U \cdot N_{PE}}{Energy} [\frac{Gops}{Joule}]$

〈Table 3〉 Performance of the FCM algorithm for the codebook design plus VQ encoding

Number of codewords	System utilization [%]	Total cycles [millions]	Execution time [sec]	Sustained throughput [Gops/sec]	PSNR [dB]	Compression rate [bpp]
C=2	0.76	22.21	0.17	368	18.7	0.0625
C=4	0.78	59.59	0.45	374	21.9	0.125
C=8	0.76	90.64	0.68	367	22.2	0.1875
C=16	0.64	244.68	1.84	307	24.8	0.25
C=32	0.62	286.39	2.15	298	26.1	0.3125
C=64	0.50	410.39	3.09	239	27.3	0.375
C=128	0.46	823.32	6.19	222	28.6	0.4375
C=256	0.45	1,620.9	12.2	219	29.7	0.5

<Table 3> summarizes the performance of the FCM algorithm for generating differentnumber of codewords and the VQ encoding processin terms of execution time, system utilization, sustained throughput, PSNR, and compressed rate parameters where the system utilization is calculated as the average number of active processing elements.

4.1 Performance Comparison with other Array Systems

<Table 4> compares three parallel VQ implementations of different hardware platforms [12-14]. The performance is computed for 4x4 blocks. The MasPar1 performance is compensated for its larger codebook. Although the MasPar and Inmos systems are several years old, the performance offered by the proposed parallel implementation provides significantly greater performance and efficiency than appropriately scaled alternative systems.

〈Table 4〉 Performance comparison of selected parallel VQ implementations

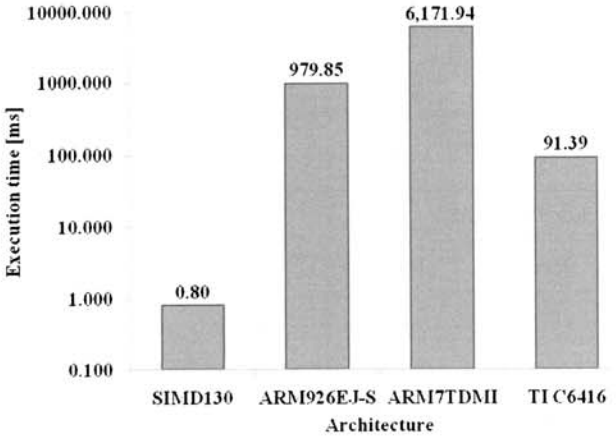
Parameter	MasPar1	Inmos	MasPar2	SIMD130
No. of Processing Elements	2,048	32	16,384	4,096
Clock Rate [MHz]	12	20	12.5	130
Image Size	128 x 128	512 x 512	512 x 512	256 x 256
Execution Time [msec]	1,790	4,800	585	0.8
Performance (Kblocks/sec)	1.14	3.41	28	5,120

4.2 Performance Comparison with Today's Commercial Processors

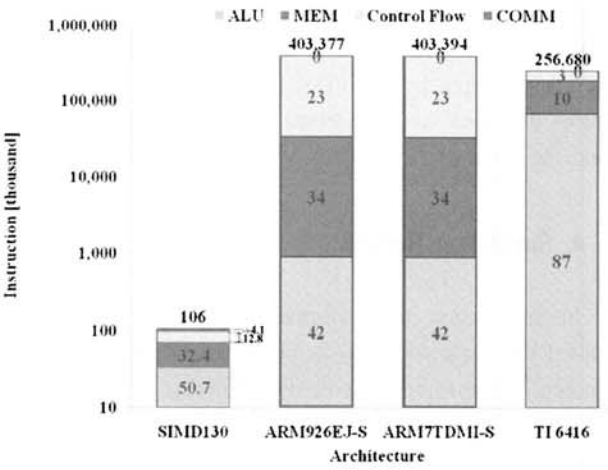
This section compares the performance of our parallel FVQ implementation to that of current commercial ARM and TI DSP processors including ARM7TDMI, ARM926EJ-S, and TMS320C6416 implemented with the same 130nm technology in terms of instruction count, execution time, and energy efficiency. A comparison between the parallel implementation and the implementations using the commercial products carries unavoidable errors. The objective of this section is to show the potential for improved performance and energy efficiency, rather than a precise performance comparison, since ARM chips are commercially designed and manufactured products and our baseline architecture is an architectural exploration.

(Figure 5) shows the performance of the VQ encoding implementations using fourdifferent architectures (SIMD 130, ARM926EJ-S, ARM7TDMI, TMS320C6416 DSP). Our SIMD130 outperforms ARM and TI DSP processors in execution time.

(Figure 6) shows additional data presenting instruction distribution of the VQ encoding implementations using four processors. Each bar divides the instructions into the arithmetic-logic-unit (ALU), memory (MEM), Control flow, and communication (COMM). The ALU and MEM instructions are computation cycles while COMM instructions are necessary for data distribution among processor elements. Control flow consists of conditional (or branch) instructions. An interesting observation is that the parallel encoding implementation using SIMD130 is dominated by ALU and MEM operations (51% ALU and 32% MEM instruction types) due to the inherent characteristic of VQ encoding. These performance results are combined with power parameters for each function unit to calculate energy efficiency.



(Figure 5) Execution time of VQ encoding using three architectures



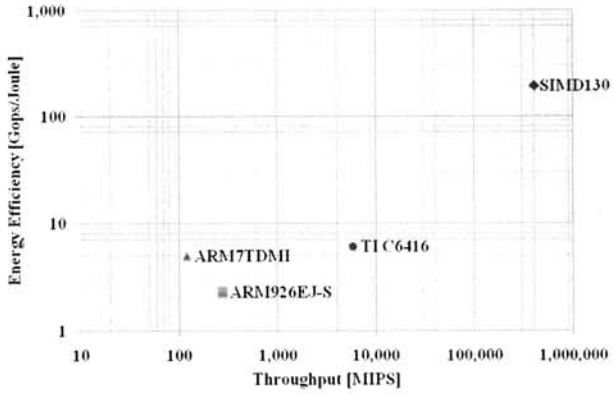
(Figure 6) Instruction distribution of VQ encoding using four different processors

<Table 5> showsan overall performance for each case. SIMD130 outperforms both ARM and TI DSP processors with the same 130nm technology in terms of execution time and energy efficiency which is defined as the task throughput achieved per unit of Joule. SIMD130 achieves about 100x higher energy efficiency than the ARM processors. This is because SIMD130 achieves higher sustained throughputs with a small increase in the system power. Increasing energy efficiency improves sustainable battery life for given system capabilities. Moreover, our parallel implementation provides 1000x better performance than the ARM processors.

(Figure 7) graphically shows the throughput-energy efficiency for each case.

〈Table 5〉 Performance comparison of the VQ encoding implementation using four different processors

Parameter	Units	SIMD130	ARM926EJ-S	ARM7TDMI	TI C6416
Technology	[nm]	130	130	130	130
Clock frequency	[MHz]	133	250	133	720
Average power	[mW]	2,083	120	24	950
Average throughput	[MIPS]	402,821	275	120	5,760
Encoding time	[ms]	0.8	6,172	980	91.4
Consumed time (10W • battery)	[hr]	5,997	13.5	426.3	115.1
Energy consumption	[uJoule]	1,667	740,632	23,458	86,854
Energy efficiency	[Gops/Joule]	193.4	2.3	5	6.1



(Figure 7) Performance comparison of each processor in energy efficiency and throughput plane

5. Conclusion

In this paper, we have presented and evaluated the impact of our proposed parallel FVQ implementation using a specified data parallel architecture in terms of the performance and energy efficiency. Experimental results show that the proposed parallel implementation provides greater performance and efficiency than appropriately scaled alternative parallel systems. In addition, the proposed parallel implementation provides 1000x greater performance and 100x higher energy efficiency than other implementations using today's ARM and TI DSP processorswith the same 130nm technology. These results suggest that he proposed parallel implementation has the potential for the improved performance and energy efficiency.

References

[1] N. B. Karayiannis, P.I. Pai, "Fuzzy Vector Quantization Algorithms and Their Application in Image Compression," *IEEE Trans. on Image Processing*, Vol.4, No.9, pp.1193-1201, September, 1995.

[2] N. Gkalelis, A. Tefas, and I. Pitas, "Combining Fuzzy Vector Quantization with Linear Discriminant Analysis for Continuous Human Movement Recognition," *IEEE Trans. On Circuits and Systems for Video Technology*, Vol.18, No.11, Nov., 2008.

[3] G. E. Tsekouras, "*A Fuzzy Vector Quantization Approach to Image Compression*," Applied Mathematics and Computation, Elsevier, pp.539-560, 2005.

[4] F. Masulli, and S. Rovetta, "Fuzzy Concepts in Vector Quantization Training," Springer, LNAI 2955, pp.279-288, January, 2006.

[5] H. S. Jayanna and S. R. Mahadeva Prasanna, "Fuzzy Vector Quantization for Speaker Recognition under Limited Data Conditions," *in IEEE Conference on TENCON 2008*, pp.1-4, Nov., 2008.

[6] W. Pedrycz, J. Valente de Oliveira, "A Development of Fuzzy Encoding and Decoding Through Fuzzy Clustering," *IEEE Trans. on Instrumentation and Measurement*, Vol.57, No.4, pp.829-837, April, 2008.

[7] ARM hompage, www.arm.com

[8] N. B. Karayiannis, J. C. Bezdek, "An Integrated Approach to Fuzzy Learning Vector Quantization and Fuzzy c-Means Clustering," *IEEE Trans. on Fuzzy Systems*, Vol.5, No.4, pp.622-628, November, 1997.

[9] TI DSP homepage, www.ti.com

[10] S. M. Chai, T. M. Taha, D. S. Wills, and J. D. Meindl, "Heterogeneous Architecture Models for Interconnect-motivated System Design," *IEEE Trans. VLSI Systems*, special issue on system level interconnect prediction, Vol.8, No.6, pp.660-670, December, 2000.

[11] J. C. Eble, V. K. De, D. S. Wills, and J. D. Meindl, "A Generic System Simulator (GENESYS) for ASIC Technology and Architecture Beyond 2001," *in Proc. of the Ninth Ann. IEEE Intl. ASIC Conf.*, pp.193-196, September, 1996.

[12] H. J. Lee, J. C. Liu, A. K. Chan, and C. K. Chui, "A Parallel Vector Quantization Algorithm for Single-Instruction-Multiple-Data (SIMD) Multiprocessor Systems," in *Proc. of the 5th Data Compression Conference*, p. 479, 1995.

[13] K. S. Prashant, V. J. Matthews, "A Massively Parallel Algorithm for Vector Quantization," in *Proc. of the 1995 NASA Space and Earth Sciences Workshop*, 1995.

[14] T. B. Henderson and K. S. Thyangarajan, "Implementation of VQ Algorithms on a Reconfigurable Array Processor," *Professional Paper No. AD-A236 483/4/XAB*, Naval Ocean Systems, 1991.

Luong Van Huynh

e-mail : huynhlvd@yahoo.com

2005년 Electronics and Telecommunications, Hanoi University of Technology, Vietnam (BS)

2005년~2007년 Infrastructure Department, FPT Information System Com., Hanoi, Vietnam, System Engineer

2007년~2009년 울산대학교 컴퓨터정보통신공학부(석사)

2009년~현 재 Research & Development Center, BNF Technology Inc., Deajeon, Korea, Researcher

관심분야 : 병렬처리, 임베디드시스템, 멀티미디어 등

김 철 홍

e-mail : cheolhong@gmail.com

1998년 서울대학교 컴퓨터공학과(학사)

2000년 서울대학교 컴퓨터공학부(공학석사)

2006년 서울대학교 전기컴퓨터공학부(공학박사)

2005년~2007년 삼성전자 반도체총괄 SYS.LSI 사업부 책임연구원

2007년~현 재 전남대학교 전자컴퓨터공학부 교수

관심분야 : 임베디드시스템, 컴퓨터구조, SoC 설계, 저전력 설계 등

김 종 면

e-mail : jongmyon.kim@gmail.com

1995년 명지대학교 전기공학과(학사)

2000년 Electrical & Computer Engineering, University of Florida, USA(공학석사)

2005년 Electrical & Computer Engineering, Georgia Institute of Technology, USA(공학박사)

2005년~2007년 삼성종합기술원 전문연구원

2007년~현 재 울산대학교 컴퓨터정보통신공학부 교수

관심분야 : 임베디드시스템, 시스템-온-칩, 컴퓨터구조, 병렬처리, 신호처리 등