A 66fps 38mW Nearest Neighbor Matching Processor with Hierarchical VQ Algorithm for Real-Time Object Recognition

Joo-Young Kim, Kwanho Kim, Seungjin Lee, Minsu Kim, and Hoi-Jun Yoo
School of EECS
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea

Abstract— **A 66fps 38mW nearest neighbor matching processor for real-time object recognition has been fabricated in 0.13μm CMOS technology. It consists of RISC processing core, pre-fetch DMA, and two independent sets of logic merged memories. Based on hierarchical vector quantization (H-VQ) algorithm, implemented processor achieves 22.5X cycle time reduction in matching process without any accuracy loss in VQ operation. As a result, 66fps frame rate is obtained for QVGA (320x240 pixels) video images with 5632-entry database.**

I. INTRODUCTION

Object recognition is the process of identifying pre-learned objects from video input images. It is a fundamental operation in computer vision system and can be widely applied to real-time visual applications such as vehicle automation, robot localization, and face recognition [1-4]. Generally, it is divided into image pre-processing stage and post matching stage. An image pre-processing stage extracts key points from the input image and generates descriptor vectors which represent the features of the objects in image. While, a post matching stage matches extracted descriptor vectors with the pre-defined object vector database and decides the final recognition result via voting process [5]. For real-time operation of object recognition, hardware acceleration for image pre-processing stage is actively researched because it involves data intensive and regular processing functions. On the other hand, a post matching stage is, in many cases, processed by general purposed processor since it has more irregular and software oriented characteristics compared to image pre-processing stage.

Although a post matching stage has less been emphasized in a computational aspect, however, its computational load is not simple but can be processing bottle neck in real time operation. Moreover, for recent sophisticate object recognition system, a dedicated processor for post matching stage is essential because matching time is linearly increased with the size of database. As the previous works of matching processor, several dedicated vector quantization (VQ) processors that search the nearest vector in codebook database have been developed for image compression applications [7]-[9]. However, these VQ processors suffer from large silicon area caused by expanded processing logic elements for distance calculation. And one of them [8] generates computational error in VQ operation instead of obtaining computational benefits. However, different from previous VQ processors for image compression, the nearest neighbor matching processor cannot utilize full area of the chip and dedicated off-chip memories because it is a sub system block of object recognition SoC. In addition, computation error in VQ should not be admitted in matching processor because it can degrade recognition rate of whole object recognition system.

In this paper, a high frame rate nearest neighbor matching processor is proposed for high performance object recognition SoC [6]. To achieve high frame rate, both of software and hardware optimization are applied. First, based on newly proposed hierarchical VQ (H-VQ) algorithm, the searching region of the database is greatly reduced without any accuracy loss. Second, proposed hardware architecture for early database fetch and parallel vector distance calculation accelerates VQ processing further. As a result, implemented nearest neighbor matching processor realizes 22.5X cycle time reduction compared to conventional general purposed processor and achieves 66fps frame rate for QVGA (320x240 pixels) input image with 5632-entry database.

II. HIERARCHICAL VQ ALGORITHM

Vector quantization (VQ) operation is the process of searching the nearest neighbor vector in database with the input query vector. In our design, Manhattan distance is employed as the distance metric and the dimension of a vector is decided to 16. Then, VQ process can be expressed that to find the nearest vector Vm in database satisfying the following equation.

$$\sum_{d=0}^{D=15} |I_d - V_{m,d}| = MIN(\sum_{d=0}^{D=15} |I_d - V_{n,d}|), \ where \ 0 \le n \le N \quad (1)$$

In order to decrease the search region and processing time, some of previous VQ processors employ hierarchical approach like two-step search or tree search [8], [10]. However, both of two VQ algorithms have the same problem of accuracy loss. To improve this, we propose a new hierarchical VQ (H-VQ) algorithm that not only reduces search region abundantly but also guarantees the optimum VQ results. Proposed H-VQ algorithm searches a part of database to get a pseudo minimum distance as a reference and estimate the possible existence region for the exact nearest neighbor vector. After the main search operation for estimated search region is performed, the optimum nearest neighbor vector search is guaranteed.

For the proposed hierarchical VQ (H-VQ) algorithm, object vector database should be organized as shown in Fig. 2. Object vectors are sorted in ascending order according to their intensity sum and each of them has 1 tag for object identifier. Then, whole database is divided into M segment groups which

978-1-4244-2605-8/08/$25.00 ©2008 IEEE

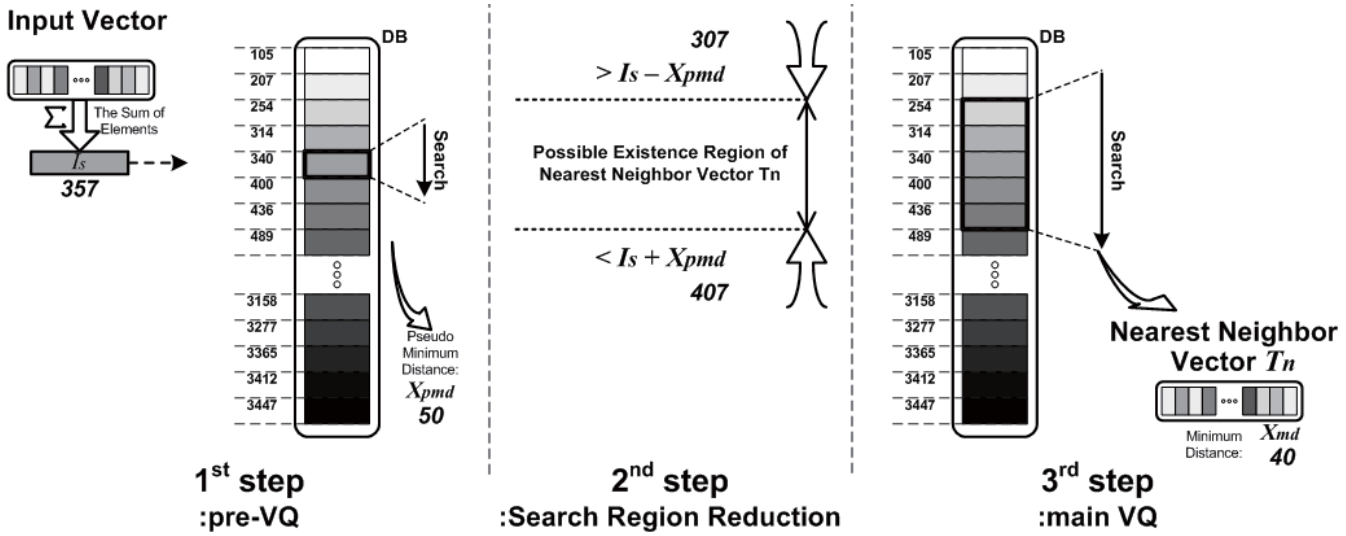


Fig. 1 Proposed H-VQ Algorithm

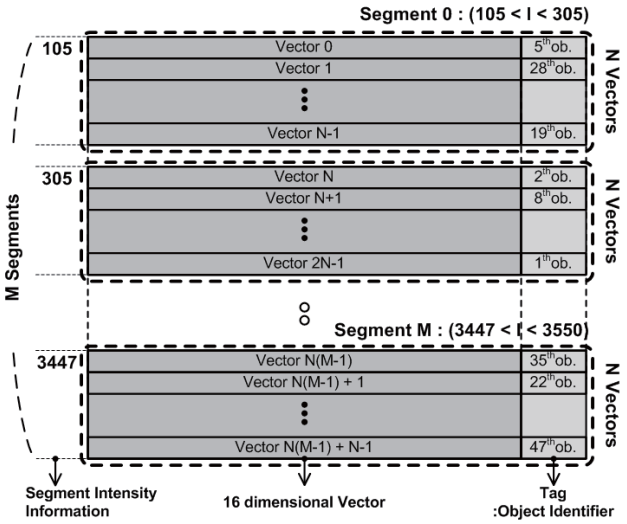


Fig. 2 Database Structure for proposed H-VQ algorithm

are composed of N vectors respectively. The intensity sum of the first vector of each segment is also included as the representative intensity of the segment. With this database, H-VQ algorithm can be summarized to 3 steps as shown in Fig. 1. First, the intensity sum of an input key point vector is calculated and the segment block that includes this intensity sum is selected. Then, pre-VQ operation is performed on this selected segment. The minimum distance value obtained from the pre-VQ operation is referred to as pseudo minimum distance *Xpmd*. Second, the possible existence region of the target nearest neighbor vector *Vm* is decided by the intensity sum of the input vector and pseudo minimum distance *Xpmd* as shown in next three following equations. *Xmd* is the minimum

$$X_{md} = \sum_{d=0}^{D=15} |I_d - V_{m,d}| \geq \left| \left(\sum_{d=0}^{D=15} I_d \right) - \left(\sum_{d=0}^{D=15} V_{m,d} \right) \right| \quad (2)$$

$$X_{pmd} \geq X_{md} \geq |I_s - V_{ms}| \quad (3)$$

$$I_s - X_{pmd} \leq V_{ms} \leq I_s + X_{pmd} \quad (4)$$

distance value between input vector and whole database and is also the distance between input vector and vector *Vm*. It is always smaller than or equal to *Xpmd* value. As a result, using *Xpmd* value, possible existence region of the nearest vector can be computed in intensity sum space. Lastly, the main VQ operation is performed on the segment blocks that cover the intensity sum region of (4) to search the nearest neighbor vector *Vm*.

Fig. 1 also explains how H-VQ algorithm operates with a simple example case. First, the intensity sum of the input vector is calculated as 357. Since a 357 is included in the 5th segment, the pre-VQ operation is performed on the 5th segment and *Xpmd* is obtained to 50 as a result. Second, the search region of the nearest neighbor vector is calculated as from (357-50) to (357+50) and this region is covered by the segments from the 3rd to the 7th. Finally, the main VQ searches the nearest neighbor vector *Vm* in these segments with the minimum distance *Xmd* 40. As a result, H-VQ algorithm reduces average search region to 19.1% in case that the whole database includes 5632 vectors organized into 22 segments. With a negligible computation overhead such as a few add and compare operations, H-VQ algorithm greatly reduces the average search region which is directly related to the execution time without any loss of exactness of VQ operation.

III. HARDWARE ARCHITECTURE

Fig. 3 shows the overall hardware architecture of the proposed nearest neighbor matching processor. It consists of RISC processing core, pre-fetch DMA, and two independent sets of memories for database vector and tag. RISC core has simple 5 pipeline stage and 32-bit datapath. RISC core plays a role in administrating overall data and control flow of matching processor. It performs H-VQ algorithm for current database set and prepare the next database set by setting pre-fetch DMA. Pre-fetch DMA initiated by RISC core transfers external database data to the internal database memories via network-on-chip interface. The sizes of the database and tag memory are 8K byte and 1K byte respectively. To increase vector data read bandwidth, bit-width of database memory is increased to

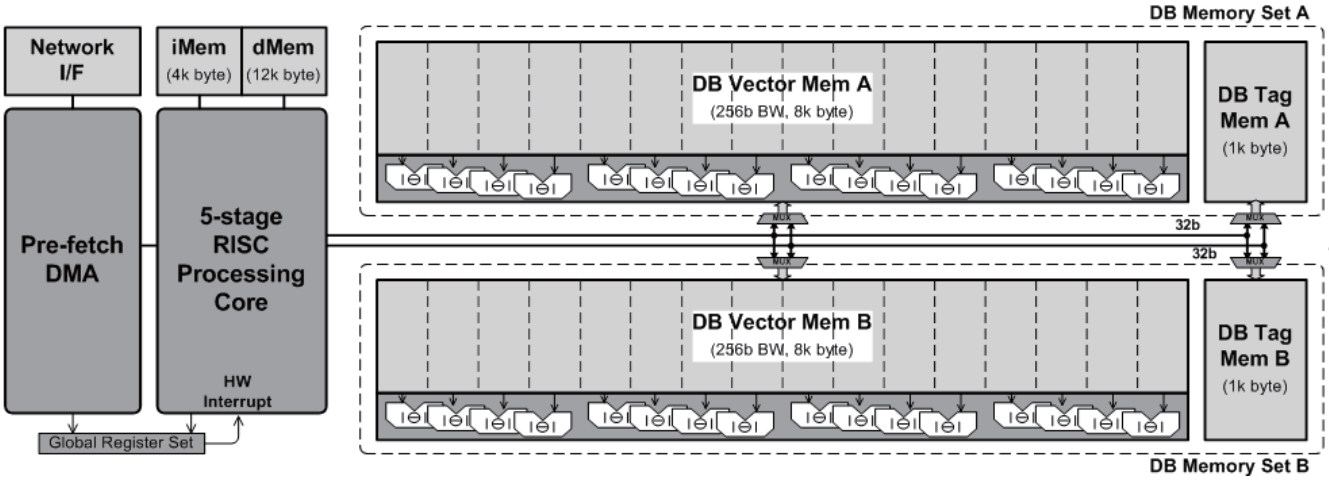


Fig. 3 Proposed Hardware Architecture

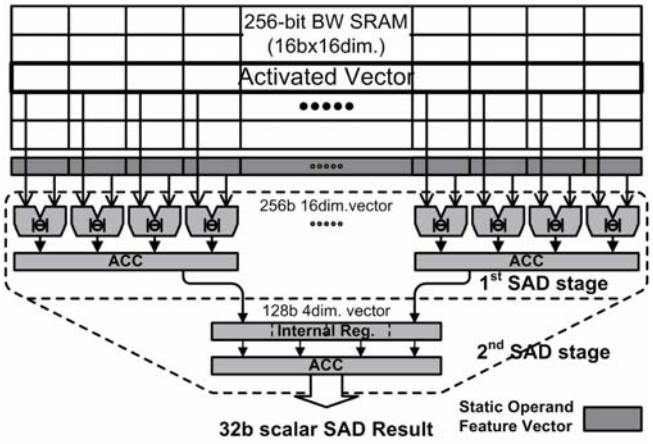


Fig. 4 SAD Memory Merged Logic

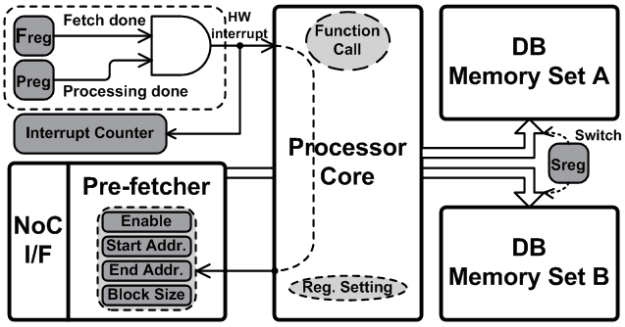


Fig. 5 Database Refilling Scheme

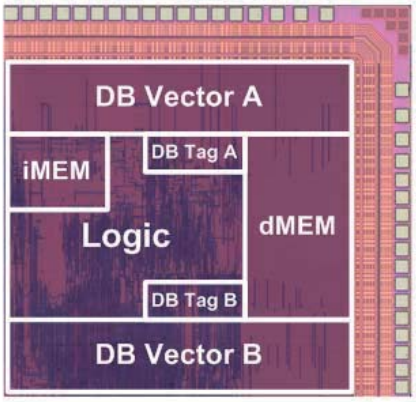
256-bit. Therefore, a 16-bit 16-dimensional vector can be read at one cycle. To resolve bandwidth conversion between 256-bit vector data and 32-bit scalar RISC processing core, SAD accumulation logic is merged into database memory.

A. SAD Merged Memory Logic

In nearest neighbor matching process, most of the computation time is spent by repeated distance calculation between input vector and database vectors. The distance calculation between the two vectors is not simple task to general purposed processor even if the distance metric is assumed to Sum of Absolute Differences (SAD). This is not efficient to SIMD type processor either because vector results of SIMD processor should be accumulated into one scalar value to get the final distance result. To resolve these problems in distance calculation, we merged 2-stage pipelined tree-structure SAD accumulation logics into 256-bit wide database memory as shown in Fig. 4. By accumulating 4 absolute difference results per stage, 16 absolute difference results are accumulated into one scalar value in every cycle at 200MHz frequency. This sufficiently resolve vector to scalar conversion between vector database memory and scalar RISC core in distance calculation. As a result, SAD merged memory logic performs distance calculation between two 16-bit 16-dimension vectors by 2 clock cycle latency and 1 clock cycle throughput at 200MHz frequency and this is equal to 12.8 giga operations per second (GOPS) processing ability.

B. Database Refilling Scheme

Since the size of database is large, having all database information in on-chip memory is not suitable. Moreover, database size can be varying with the target recognition objects. Therefore, for matching processor, iterative approach that repeats fetching and processing a part of database until the whole database is covered is desirable. This method is beneficial to on-chip memory reduction and gives flexibility to cope with the size variation of the database. However, bulk data transfer for database refilling amounts significant portion of the execution time. After the acceleration of distance calculation, this transfer time can be critical in system. To resolve this, we employ two independent sets of internal database memories and double buffers external database data. When the RISC core operates matching process with the current database part, pre-fetch DMA fetches the next part of the database simultaneously. To notify the end of the current processing / pre-fetch and to switch processing memory and pre-fetch memory, global registers and hardware interrupt is used as shown in Fig. 5. F register and P register are set by pre-fetch DMA and RISC core respectively when they finish their job. When two registers are all set, hardware interrupt is occurred to RISC core and function call which sets the global



Technology	0.13mm 1P 8M CMOS
Area	2.0mm^2
Power supply	1.2V core, 2.5 I/O
Operating freq.	200MHz
# of gates	35K gates
Memory	34K byte iMem/dMem: 4K/12K byte DB Vector/Tag Mem: 8K/1K byte
Power	38mW @ 1.2V, 200MHz
Frame rate	66 fps
Input image	QVGA(320x240 pixels) video image
Interface	Network-on-Chip

Fig. 6 Chip Micrograph and Features

registers and pre-fetch DMA registers is called for the next database iteration.

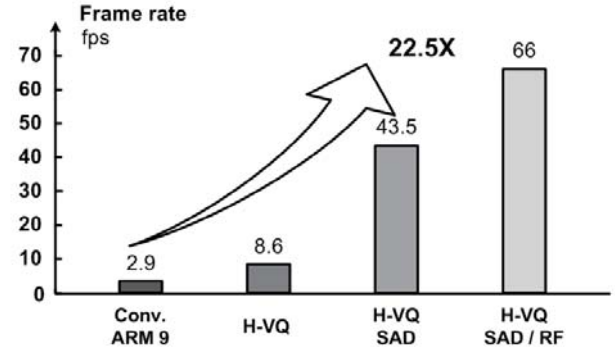
IV. IMPELEMENTATION RESULTS

The proposed nearest neighbor matching processor is fabricated in a 0.13µm CMOS technology. Fig.6 shows a chip micrograph and features of the implemented processor. It integrates 35K logic gates and 34K byte 6 memory macros and occupies 2.0mm^2 area. 38mW average power is dissipated when it operates nearest neighbor matching processing at 200MHz frequency.

Fig. 7 shows the experimental results of the proposed processor. Based on conventional ARM9 processor, the achievable frame rate of proposed processor is measured when H-VQ algorithm, SAD merged memory logic, and database refilling scheme is applied in turn. The experiment is performed in the case when 100 key point vectors are matched with 5632-entry database. As a result, H-VQ algorithm, SAD merged memory logic, and database refilling scheme achieves about 3X, 5X, 1.5X cycle time reduction, respectively. In overall, 22.5X cycle time reduction is achieved and 66fps frame-rate is obtained for QVGA (320x240 pixels) video input image.

V. CONCLUSIONS

A 66fps 38mW nearest neighbor matching processor for real-time object recognition has been fabricated in 0.13µm CMOS technology. It consists of RISC core, pre-fetch DMA,



	ARM9 only	SW opt. (H-VQ)	SW + HW opt. 1 (H-VQ, SAD)	SW + HW opt. 2 (All applied)
Exe. time/frame	341.3 ms	116.0 ms	20.6 ms	15.2 ms
Frame / sec	2.93 fps	8.62 fps	43.48 fps	65.80 fps
Improvement	1X	2.94X	14.82X	22.47X

Fig. 7 Experimental Results

and two independent sets of logic merged memories. Based on hierarchical vector quantization (H-VQ) algorithm, applied SAD merged memory logic and database refilling scheme achieves overall 22.5X cycle time reduction in matching process without any loss of accuracy. As a result, 66fps frame rate is achieved for QVGA (320x240 pixels) video images with 5632-entry database.

REFERENCES

[1] Donghyun Kim, Kwanho Kim, Joo-Young Kim, Seungjin Lee and Hoi-Jun Yoo, "An 81.6 GOPS Object Recognition Processor Based on NoC and Visual Image Processing Memory," IEEE Custom Intergrated Circuits Conference 2007, pp. 443-446.

[2] Wolfgang Raab, et al., "A 100-GOPS programmable processor for vehicle vision systems," IEEE Design & Test of Computers, Vol. 20, Issue 1, pp. 8-15, Jan.-Feb. 2003

[3] Juan Nieto, Jose Guivant, and Eduardo Nebot, "Real time data association for FastSLAM," Proceedings of IEEE International Conference on Robotics and Automation, Vol. 1, pp. 412 – 418, Sept. 2003

[4] Y. Hori, M. Kusada, and T. Kuroda, "A 0.79mm2 29mW Real-Time Face Detection Core," IEEE Symposium on VLSI circuits, Dig. of Tech. Papers, pp. 188-189, June 2006

[5] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, pp. 91-110, Jan 2004

[6] Kwanho Kim, et al., "A 125GOPS 583mW Network-on-Chip Based Parallel Processor with Bio-inspired Visual Attention Engine," IEEE International Solid State Circuits Conference, Dig. of Tech. Papers 2008, pp.308-309.

[7] A. Nakada, T. Shibata, M. Konda, T. Morimoto, and T. Ohmi, "A fully parallel vector-quantization processor for real-time motion-picture compression," JSSC, pp.822-830, June 1999

[8] T. Nozawa, M. Konda, M. Fujibayashi, M. Imai, T. Ohmi, "A Parallel Vector Quantization Processor Eliminating Redundant Calculations for Real-time Motion Picture Compression," IEEE International Solid State Circuits Conference, Dig. of Tech. Papers 2000, pp.234-235

[9] M. Fujibayashi, et al., "A Still-Image Encoder Based on Adaptive Resolution Vector Quantization Featuring Needless Calculation Elimination Architecture," JSSC, pp.726-733, May 2003

[10] C.-Y. Lee, S.-C. Juan, and Y.-J. Chao, "Finite state vector quantization with multipath tree search strategy for image/video coding," IEEE Trans. on Circuits and Systems for Video Tech., vol. 6, pp. 287-294, June 1996