A Functional Memory Type Parallel Processor for Vector Quantization

K. Kobayashi, M. Kinoshita, M. Takeuchi, H. Onodera and K. Tamaru

Department of Electronics and Communication, Graduate School of Engineering,
Kyoto University, Kyoto, JAPAN

kobayasi@kuee.kyoto-u.ac.jp

Abstract--- We propose a memory-based parallel processor for vector quantization called a functional memory type parallel processor for vector quantization (FMPP-VQ). It accelerates nearest neighbor search of vector quantization. All distances between an input vector and reference vectors in a codebook are computed simultaneously in all PEs. The minimum value of all distances is searched in parallel. The nearest vector is obtained in O(k), where k stands for the dimension of vectors. An LSI including four PEs has been implemented. It operates at 25MHz clock frequency.

I. Introduction

Vector quantization[1] is one of promising candidates for low bit rate image compression. It requires much less hardware for decompression. For compression, however, a large amount of computation is required. The most time-consuming factor on compression is ''nearest neighbor search'' to find the vector nearest to an input one among a large number of reference vectors. A set of reference vectors is referred to a codebook.

We have developed a memory-based parallel processor called a functional memory type parallel processor for vector quantization: FMPP-VQ, which is used to accelerate the nearest neighbor search. It has as many PEs as reference vectors. A shared bus connects all PEs. The nearest vector is searched exhaustively in parallel. Each PE has conventional memories to store reference vectors and a logic unit to compute the distance between an input vector and reference vectors. The nearest vector is obtained using CAM-based parallel search. These procedures are done in O(k), where k stands for the dimension of vectors. The number of reference vectors does not affect computation time. On the nearest neighbor search, only input vectors are given from a shared data bus. Distance computation is done locally in each PE. Thus, memory-based SIMD processors perform effective computation through a shared bus. Reference vectors can be easily updated, since they are stored into conventional memories. All PEs can be laid out into memory-like regular-array structure, which minimizes the hardware cost.

II. Functional Memory Type Parallel Processor for Vector Quantization

In the nearest neighbor search, we should find the nearest vector $\vec{q}$ among reference vectors $\vec{y}$ in a codebook Y for every input vector $\vec{x}$ according to (1).

$$\vec{q} = \min_{\vec{y} \in Y} d(\vec{x}, \vec{y}) \quad (1)$$

The distance measure of $d(\vec{x}, \vec{y})$ determines hardware cost. The FMPP-VQ uses the absolute error $\sum |\vec{x} - \vec{y}|$ as $d(\vec{x}, \vec{y})$ to minimize the hardware cost. The AE has enough quality for image processing compared with the squared error, since input vectors from images spread out sparsely in Euclidean space.

Figure 1 shows a block diagram of the FMPP-VQ. A processing element (PE) of the FMPP-VQ is called a ''block.'' All blocks are laid out in a two-dimensional regular array and connected through a global data bus like a conventional memory. Figure 2 shows a simplified schematic diagram of a block. A block consists of memories and a simple logic unit(LU). Codebook words w_i are conventional 6 transistor SRAMs to store all elements $y_0, y_1, ..., y_{k-1}$ of a reference vector $\vec{y}$. For vector quantization of images, the dimension of vectors k is usually 16. Thus, the FMPP-VQ has 16 codebook words. The LU computes $d(\vec{x}, \vec{y})$. The result word R stores $d(\vec{x}, \vec{y})$. The operand word B stores an operand for addition, subtraction and comparison. There is no conventional adder in the LU. Instead of that, the operand word, the carry chain and other logic gates work together for addition or subtraction according to (2, 3).

$$A + B = \overline{P + \overline{C}}, \quad \overline{C_i} = G_i + P_i \cdot C_{i-1} \quad (2)$$

$$P = A \oplus B, \quad G = A \cdot B \quad (3)$$

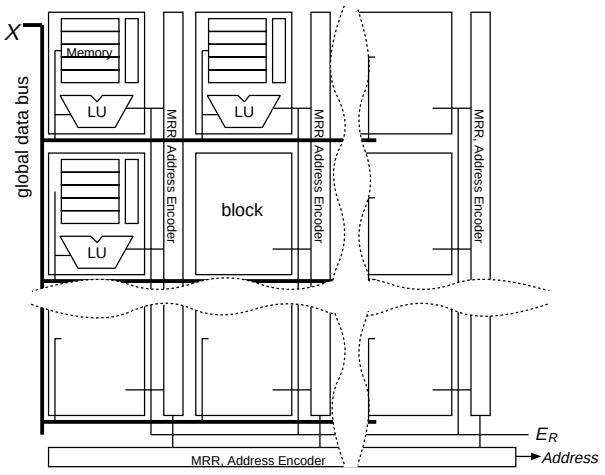

Figure 1: Block Diagram of the FMPP-VQ

The operand word is designed based on conventional SRAM-based CAM[2]. Logical operations (P, G) required for addition are obtained using four pass transistors in the operand word.

ASP-DAC '97
0-89791-851-7$5.00 © 1997 IEEE

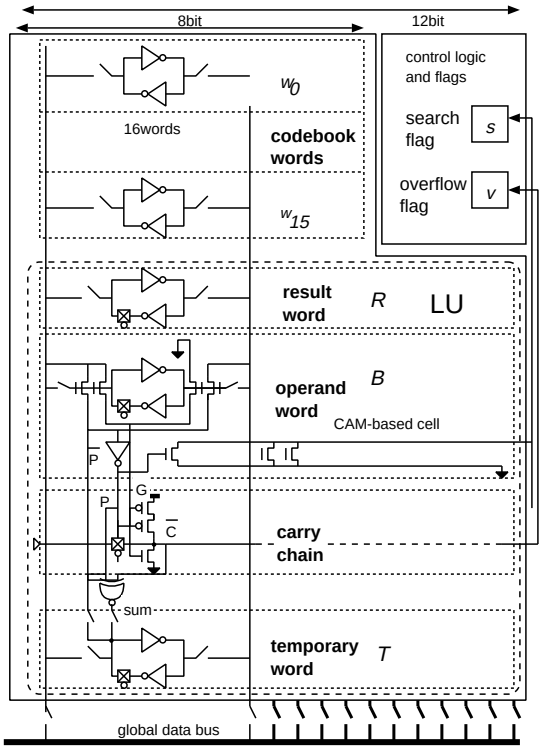

Figure 2: Schematic diagram of a block

The carry chain produces carries from P and G. The XOR gate at the bottom of the carry chain finally produces the sum. The temporary word T stores the sum. Note that operations including carry generation are done in bit parallel. Thus, we can get the sum in a single step. A codebook word is 8bit, while logic unit is 12bit, since absolute distances may become 12bit values. Two flags, an overflow and a search flag are located at the left of the codebook words. The overflow flag v stores overflow from addition or subtraction. The search flag s stores a search result. Some control logics at the left of the codebook words prohibit operations according to the state of these flags. Output signals of all search flags are sent to a wired-OR logic connected to the signal E_R, which becomes low when there is no true search flag. It accelerates the minimum search like conventional CAMs. The MRR resolve the place of the block whose search flag is true. The address encoder generates the address of the block.

Codebook words in each PEs store a reference vector. An input vector is broadcast to all PEs element by element. Thus the distance between the input vector and all reference vectors are computed element by element, but in all processors in parallel. The FMPP-VQ computes the distance in 165 clock cycles. Computation time is constant at any number of reference vectors.

III. Experimental Results

We have implemented an LSI including four PEs and some test circuits using a 0.7 μm CMOS process. Four PEs and sense amplifiers occupy 2.43mm^2. They are shown in the chip microphotograph Fig. 3. All components are designed with full custom methodology. All elements in a PE such as codebook words, an LU and flags are implemented into a rectangular area of 706μm$\times$389μm. Table 1 shows several specifications of the LSI. All functionalities for computing the nearest vector work properly at 25MHz clock frequency.

TABLE I : LSI Specifications

Process	0.7μm CMOS
Die size	26.3mm^2
Area of Fig. 3	2.43mm^2
# IOs	116
Frequency	25MHz
Power dissipation	3.8mW @25MHz (Whole LSI)
	0.87mW @25MHz (4PEs)

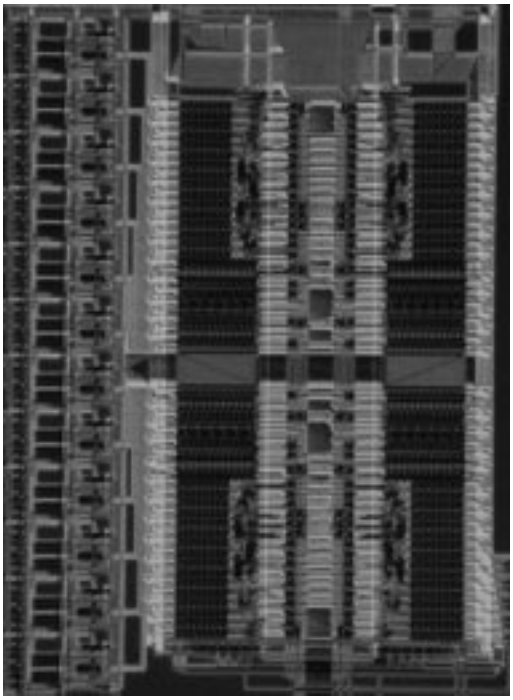

Figure 3: Chip microphotograph of the four block FMPP-VQ.

IV. Conclusion

The FMPP-VQ accelerates the nearest vector search on vector quantization. It achieve both high performance and small silicon area. The nearest vector search is done in O(k), where k stands for the dimension of vectors. Computation time does not depend on the number of reference vectors. An LSI including four PEs has been implemented. It works at 25MHz clock frequency. Now we just finished designing an FMPP-VQ LSI including 64 PEs. One of our goals is to develop a video compression system using the FMPP-VQ to transmit videophone images through a low-bit line.

References

[1] A. Gersho and V. Cuperman. Vector quantization. *IEEE Commun. Mag.*, 21(9):15--21, 1983.

[2] T. Ogura, S. Yamada, and T. Nikaido. A 4kb Associative Memory LSI. *IEEE J. Solid-State Circuits*, SC-20(6):1277--1282, 1985.