ASSOCIATIVE ARCHITECTURE FOR FAST DCT

Y. Shain, A. Akerib, R. Adar

Associative Computing Ltd.
9 Hataasiya Street
Raanana, Israel 43100

ABSTRACT

This paper discusses an associative processor architecture designed to meet the demands of real-time image processing applications. In a single chip, this architecture provides thousands of processors – one for each pixel, in the form of associative memory. This paper focuses on a generic, proprietary associative processor architecture and discusses implementing the discrete cosine transform (DCT) using processors based on this architecutre. Associative Computing Ltd. has developed a commercial associative chip based on this architecutre, and while the DCT implementation discussed refers to future generations based on this architecture, reference is made throughout to the Company's present processor.

Processors based on our associative architecture can process the large amounts of data typically required in real-time imaging applications at a lower cost-performance ratio than conventional processors. The scalable nature of memory-based processor architecture allows developers to rapidly increase processing power without altering the fundamental processor, or system architecture. The underlying technologies used in the Company's present processor can significantly facilitate the development of associative processing as an alternative to conventional processing for video applications including compression and video editing.

1. INTRODUCTION

A host of new processor architectures embodying a number of process-enhancing principles have enabled image-processing systems to meet real-time demands for many basic functions. There are four basic principles for enhancing processor performance [7]:

1. Data distribution – concurrent processing of multiple data.
2. Task distribution - concurrent execution of tasks.
3. Instruction pipelining - parallel execution of operations.
4. Concurrent data transfer and processing.

These principles have found expression, in one way or another, in most video signal processors. For example, Intel's Pentium with MMX embodies the data distribution principle by featuring subword parallelism whereby one 32-bit word is operated upon as four, 8-bit words [5]. This principle finds further expression in SIMD architectures, such as Texas Instruments' Scan-line Video Processor (SVP) [10]. A further extension of the data distribution principle is MIMD architecture, such as Texas Instruments' TMS320C80 that includes a RISC core, a floating point processor and four, parallel DSPs [12]. Regrettably, the utility of SIMD processors has been limited to low level tasks, while the more potent MIMD processors are both expensive and difficult to program.

The task distribution principle is embodied in VLIW architectures, such as Texas Insturments' TMX320C6201 [9] (containing eight parallel units: two multipliers and eight adders) and Chromatics' Mpact2, the latter comprising a VLIW core capable of issuing one or two instructions packed into a 72b word per cycle [6]. Experience has shown it difficult to write efficient compilers for VLIW processors as the various memory and hardware addresses must be controlled throughout in order to execute these instructions. The ability to execute many instructions in parallel is further dependent on the specific application, i.e., the application must be of such a nature that its data and tasks are separable into parallel entities.

Task distribution is also realized in the adaptation of special instructions that execute conspicuous or prevalent operations, typically requiring a number of basic operations, in a single machine cycle. For example, Intel's Pentium with MMX instruction set includes an instruction to perform a multiply/accumulate + saturate operation in a single machine cycle [6]. This technique provides an application-specific solution as these instructions are necessarily designed for particular applications.

Another method of distributing processor tasks, based on the instruction pipelining principle, is to incroporate a co-processor into the chip. Most multimedia processors benefit from such an arrangement. On-chip SRAM's and memory caches are also frequently used to reduce access time to data and instructions. For example, Phillips' Trimedia TM-1 [4] includes a variable length decoder unit, an instruction cache and fast interface, in addition to a VLIW architecture controlling 27 units and the capacity to execute five instructions in parallel.

Our proposed associative processor architecture is highly cost-effective and can be feasibly implemented in consumer electronics in the future. This architecture is an SIMD architecture that executes instructions on thousands of data words in parallel. Processors based on this architecture comprise a one-dimensional array of thousands of memory words made up of content addressable memory (CAM) cells. These cells are capable of comparing values stored therein to an on-chip register and writing the contents of that register to themselves in parallel. By performing only these two operations, compare and write, this associative processor can implement a truth table; hence the processor is fully programmable and can implement all logical and arithmetic operations.

0-7803-4428-6/98 $10.00 © 1998 IEEE

The processor's power directly correlates to the density of the CAM. Memory manufacturing technologies are advancing at a prodigious rate. Finer manufacturing processes will allow condensing more processors into a single piece of silicon. Due to the linear relationship between the processor's power and the number of processors per chip, these advances will have a linear affect on the chip's performance.

The associative processing core consists of memory words (CAM cells), hence each word is easily divided into two parallel functional units: input/output and processing. Using this dichotomy, our architecture implements the principle of concurrent data transfer and processing with a minimal overhead.

Processors based on our architecture are highly modular. Connecting multiple chips increases the size of the associative memory and boosts performance linearly. For example, the Company's present chip can process up to 2K memory words in parallel; cascading two chips forms an associative memory twice as large and can process up to 4K memory words at once. This connection requires no glue hardware or software.

This paper is organized as follows. In Section 2 we discuss the associative processing concept. In Section 3 we describe an associative implementation of the DCT. In Section 4 we discuss architecture for an associative multimedia chip.

2. ASSOCIATIVE PROCESSING

The classical associative approach pioneered by Foster [3] is a unique SIMD architecture based on implementing only the most basic logical operations en masse on data stored in an associative memory. Basically, the philosophy behind this approach is that it is advantageous to perform complex operations, such as addition and multiplication, using a number of simple operations if the simplicity of these operations will enable the construction of a memory capable of executing them on large blocks of data. For example, addition of two eight-bit numbers takes 41 machine cycles using the Company's present chip, but executing the addition on 2000 pairs of eight-bit numbers at once reduces the rate to 0.021 cycles per eight-bit addition.

Foster taught that using only two primitives, compare and write, it is possible to implement the truth table maxim, "if condition, then action." Since all logical and arithmetic operations can be performed by implementing a truth table, the associative processing approach offered a fully programmable alternative to conventional processing.

According to the classical associative processing approach, a word of associative memory is assigned to each pixel and constitutes a primitive processor. Such a system may be regarded as an array of simple processors, one for each word in memory. All bits of all words in the associative memory are operated upon at the same time, in SIMD fashion. Ruhman & Scherson [13] introduced a shift mechanism in the responder (tag) register to provide communication between processors. Akerib, Ruhman & Ullman [2] elaborated upon this mechanism to facilitate operations over a neighborhood of the image, and adapted this approach to computer vision and VLSI implementation.

2.1 Basic Structure

Our proposed associative processing array comprises a one-dimensional array of memory words. For example, the Company's present chip features 2K memory words of 128 bits each, enabling this processor to process an image region of up to 2K pixels at once. Alternative architectures, featuring more words of shorter length are possible. The chip's target applications determine whether it is advantageous to have more words or longer ones.

Each word is divided into two sections: an I/O section of configurable size and a processing section. A block diagram of the associative array can be seen in Fig. 1. The left shaded block corresponds to the chip's video I/O section, whereas the right shaded block corresponds to the remaining bits of each associative word used for processing.

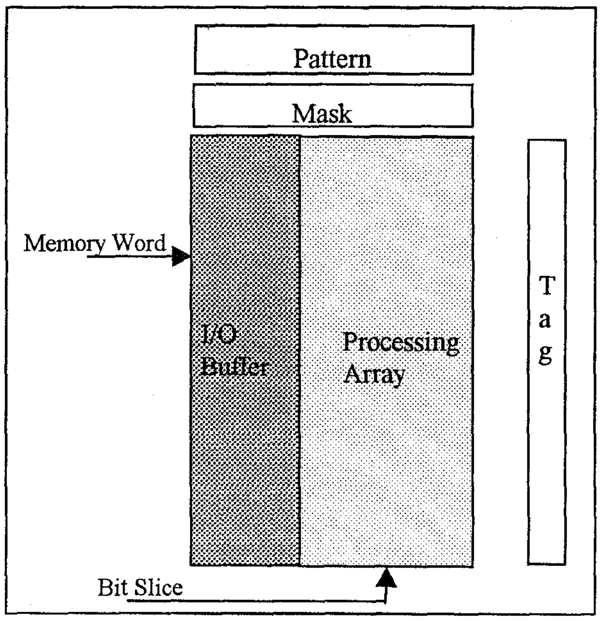

Figure 1. Block diagram of the associative array. The left shaded block represents a portion of the all memory words used for I/O, while the right shaded block represents the remainder of the array used for processing. Three registers assist in processing: Pattern and Mask select bit slices and Tag identifies memory words.

The associative processor, as depicted in Figure 1, is parallel by bit as well as by word. The associative memory implements only two primitives: compare and write. In a compare operation, a pattern register is matched against all words of memory simultaneously, and agreement is indicated by setting the corresponding tag bit. The comparison is only carried out on bits indicated by the mask register and only on words indicated by the tag register. Hence the tag register serves as the source for words participating in a compare operation at the outset of this operation, and also serves as a destination for collecting successful matches at the end of the compare operation.

The write primitive operates in a similar manner. The contents of the pattern register, in all bits indicted by the mask, are simultaneously written to all words indicated by the tag. In this case, the tag serves only as a source.

The associative array processes a region of pixels at once. The size of the region depends on the size of the associative memory. Although in the original image these pixels may occupy any area, in the associative memory they are arranged as a linear scan.

Communication between processors is carried out one bit at a time by means of a shift primitive implemented in the tag [1]. A bit slice is copied to the tag (by executing a compare primitive limited to a single bit slice), and the result is shifted the desired distance, then copied back to the associative array (by executing a write operation to a bit slice whose initial values were all zero. Recall that the tag serves as the source for words that participate in a write operation, hence only words having a corresponding set bit in the tag will be written to.)

Shift primitives can be implemented to shift various distances in a single machine cycle. The shift primitives implemented in the Company's present chip allow for shifts of 1, 8 and 16 bits in a single cycle. Communication between distant processors (e.g., distances other than 1, 8 and 16, in the Company's present processor) is done using a plurality of shifts, whereby the total distance of the shift is determined by the number of shifts and the size of each shift. Significantly, this architecture enables communication between any two memory words (processors), as opposed to other architectures, which allow only communication between eight neighboring processors. When the distance is uniform for all pixels in the processed region, communication between word processors is simultaneous. Most vision algorithms, including neighborhood operations, are fortuitously of such a nature and require a uniform communication pattern.

For example, if Y rows of X pixels are input to the associative memory one row at a time, horizontal neighbors are situated one on top of the other in the associative memory. Vertical neighbors, in this case, are situated at a distance of X words from each other. Communication between horizontal neighboring pixels for all pixels in the region requires a uniform shift of one (1); communication between vertical neighboring pixels requires a uniform shift of X.

Input and output of video data are executed in parallel to processing using a portion of each associative word as an I/O buffer. Each associative word includes a shift register of CAM cells, which can be configured in increments of eight bits for either I/O or processing. The configuration is uniform for all associative words, thus forming an I/O-buffer-array. At each clock cycle, one word's shift register can output a processed word and input a fresh word. In the Company's present processor, outputting a processed region and inputting a new region requires 2K cycles, during which time associative processing can be carried out. After 2K cycles, the input region is transferred into the processing portion of the associative memory via the tag, one bit slice at a time. The two clock cycles that this takes (compare a bit slice in the I/O-buffer-array; write that bit slice to a new location) for all 2K words essentially provide a bus of 1K for this transfer, making the I/O overhead negligible. After the input region has been transferred to the associative memory, a processed region can be transferred from this memory to the I/O-buffer-array in a similar fashion.

3. ASSOCIATIVE DISCRETE COSINE TRANSFORM (DCT)

Many image compression methods, including the JPEG, MPEG-X, and H.26X standards, are based on the discrete cosine transform (DCT), making it a good example application on which to demonstrate the power of our associative approach. The associative DCT implementation for 8x8 samples takes advantage of the separability of the DCT. Hence, this execution implements a 1D DCT twice: once along each axis.

In a processor based on our associative architecture mathematical operations are carried out by aligning operatives in a single associative word and implementing a truth table on respective pairs of bits [8]. For example, eight bit addition is performed by aligning two eight-bit values in a single associative word with an additional carry bit in that same word. Next, all possible combinations are tried (compare) and when a match is detected, appropriate new values are written to the sum bit and carry bit (write). Pairs of operatives are aligned, one bit slice at a time, using the shift primitive described above.

The associative DCT implementation we developed is parallel on two planes: it operates on a plurality of 8x8 blocks at once, and within each 8x8 block various tasks are carried out in parallel. Regarding operations on a plurality of blocks at once, an associative memory of 2K words can contain 32 blocks of 8x8 pixels and operates on all blocks at once; an array of 8K words can contain four times as many blocks (128).

The DCT operation is basically one of successive multiplication of the elements of a data vector by those of a coefficient vector and the summing of the products. Arranging the DCT for a parallel implementation that takes advantage of the associative capacity for parallelism within each 8x8 block (i.e., parallel addition between elements and parallel multiplication between elements), we arrived at only six additions and two multiplications. This compares favorably with the both, the straightforward implementation requiring 512 multiply operations and 448 additions, and the Chen-Smith algorithm requiring 96 multiply operations and 256 additions, per 8x8 1D DCT.

Another advantage of the associative architecture is its flexibility in performing calculations on vectors of various precisions. Because the associative processor performs these operations one bit-slice at a time, resources are used efficiently. For example, addition of two eleven-bit vectors or multiplication of a 13-bit vector by a 17-bit vector are carried out on relevant bit slices only. Our processor architecture is flexible down to a single bit position; it is not confined to operations on bytes, words or double-words. Hence, the DCT can be calculated in a minimum of cycles while adhering to the CCITT standard, since the length of an associative operation (such as addition) is linearly dependent on the width (in bits) of the operand vectors. Other processors, such as Intel's Pentium with MMX, optimized to work on elements at 8-bit resolutions require twice the number of cycles to perform 9-bit MPEG decoding as 8-bit JPEG decoding. Alternatively, these processors forfeit accuracy for efficiency's sake.

Assuming an associative core of 16Kx160 CAM cells (feasible using 0.25 micron technology with a 4 level metal process), a 9-bit DCT operation takes 32 machine cycles per 8x8 block in a YUV 4:2:2 image (calculated performance). As a benchmark, Texas Instruments' TMS320C62x processor requires 226 machine cycles to perform the same operation [11]. In addition, the concurrent input, output and processing enabled by our processor architecture is translated into very low overhead when inputting and outputting 8x8 blocks. As a result, a processor based on our architecture operates at peak performance while other processors cannot reach their peak performance levels in real-time processing due to I/O constraints.

4. ONE-CHIP VIDEO PROCESSOR

In the preceding section we described an associative implementation of the DCT. However, most of the fundamental operations that constitute the standard compression algorithms can be implemented in parallel. We believe that associative processing can efficiently perform the majority of operations required by the ISO and ITU compression standards.

Future generations of associative processors based on our proprietary architecture that feature dense arrays of CAM will perform all of the following operations at a greater efficiency than conventional processor architectures:

- Quantization
- Zigzag reordering
- Zero run length calculation
- Huffman coding
- Motion vector calculation based on a current image and a reference image
- Calculations associated with bit-rate control and gathering statistics on the nature of each block in an image in order to improve the quality of the compression

For example, zigzag reordering of 12-bit samples takes 953 machine cycles per region. For a region of 8K pixels this performance translates into less than 0.125 cycles per pixel.

Numerous processor architectures take advantage of data patterns in order to accelerate execution of functions. For example, some processors offer a fast inverse DCT by assuming that most of the block's values are zero. The same processors are much slower when performing a forward DCT. By contrast, an associative processor based on our architecture makes no assumptions regarding the data processed, and as a result, forward and inverse DCT operations enjoy the same levels of efficiency.

The only operations not efficiently performed associatively, and which take only a small amount of overall processing time, are variable length decoding (at the decoding stage), and arranging the bit stream and adding headers (at the compression stage). All of these operations can be performed by auxiliary units that can be added to the associative core without having a great impact on the die size. In addition, these operations can be pipelined together with operations in the associative core.

Our associative processor architecture is not a collage of units customized for compression featuring a limited instruction vocabulary. Rather, it describes a fully programmable unit capable of a wide variety of generic operations, in addition to being well suited for video compression. While DSP architectures suggest providing additional units to handle video processing tasks, an associative core can perform a variety of video-related tasks in addition to compression, such as improving image quality, video editing, graphics, as well as non-standard compression schemes, e.g., wavelets.

5. REFERENCES

[1] Avidan Akerib and Rutie Adar, "Associative Approach to Real Time Color, Motion and Stereo Vision". *IEEE International Conference on Acoustics, Speech, and Signal Processing*. Detroit, May 1995.

[2] Avidan Akerib, S. Ruhman. and S. Ullman, "Real Time Associative Vision Machine", Aritficial Intelligence and Computer Vision, Elsevier Science Publishers B.V. (North-Holland), 1991. Pp. 441-453.

[3] Foster, C.C. "Content Addressable Parallel Processors". *Van Nostrand Rignhold Co.*, 1976, chs. 2 & 5.

[4] Gerrit Slavenburg, Selliah Rathnam, and Henk Dijkstra, "The Trimedia TM-1 PCI VLIW Media Processor", *HotChips8 Symposium*, Berkely CA, August 1996. http://infopad.eecs.berkeley.edu/HotChips8/6.1/

[5] Intel, "Pentium Processor with MMX Technology" datasheet. June 1997.

[6] John A. Watlington, "Video Signal Processors",Video & Graphics Processors: 1997", MIT Media Laboratory, May 11, 1997. http://wad.www.media.mit.edu/people/wad/vsp/node1.html

[7] P. Pirsch, "Programmable Signal Processors for Video Coding", *Seminar on Low Bit Rate Video Coding*, Tel Aviv Israel, September 1997.

[8] R. Adar, A. Akerib, "Associative Architecture for Image Processing". *SPIE Conference on Parallel and Distributed Methods for Image Processing*. San Diego, July 1997.

[9] Texas Insturments TMX320C6201 Digital Signal Processor, Product Preview, June 1997.

[10] Texas Instruments SVP for Digital Video Signal Processing, Product Overview, 1995.

[11] Texas Instruments TMS320C62x Assembly Benchmarks http://www.ti.com/sc/docs/dsps/products/c6x/benchmk.htm #graphics

[12] Texas Instruments TMS320C80 Data Sheet, 1996.

[13] S. Ruhman, I. Scherson, "Associative Processor for Tomographic Image Reconstruction". *Proc. Medcomp 82 pp. 353-358.*