THE MGAP-2: AN ADVANCED, MASSIVELY PARALLEL VLSI SIGNAL PROCESSOR

Thomas P. Kelliher[2] *Eric S. Gayles*[1] *Robert M. Owens*[1] *Mary Jane Irwin*[1]

[1]Architecture and VLSI CAD Group, Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802

[2]Department of Mathematics and Computer Science
Westminster College, New Wilmington, PA 16172

ABSTRACT

The Micro-Grain Array Processor (MGAP) is a family of two-dimensional, micro-grained array processors. The processor cell architecture is extremely compact and simple, ensuring fine grainess, a very high processor density, and programming flexibility. Flexibility is maintained through a programmable interconnect which clusters array cells into larger computational units. In this paper, we will discuss the design and optimization issues of MGAP-2, both at the processor array and system levels. Various design strategies and tradeoffs are being investigated at both levels. The reader will see how lessons learned from building and using MGAP-1 have been applied in this new design effort. We also describe our MGAP programming environment and an application example — the two-dimensional discrete cosine transform, a powerful image compression tool.

1. INTRODUCTION

The Micro-Grain Array Processor (MGAP) is a family of two-dimensional, micro-grained VLSI array processors which maintains both a high degree of flexibility *and* fine grainess [1, 2, 3, 4]. The range of problems to which the MGAP may be applied is large, including signal and image processing, graph problems, sorting and searching, matrix computations, astronomy, computational biology, and computational fluid dynamics. Members of the MGAP family are variegated according to processor cell architecture, size of the 2-D mesh of processors, and cycle time. Each of the processor cell architectures is extremely compact and simple, ensuring fine grainess and a very high processor density. Ordinarily, fine-grainess and flexibility are mutually exclusive properties; we achieve flexibility, without sacrificing fine-grainess, with a programmable interconnect which clusters individual processor cells into larger, more powerful computational units (word cells) without a loss in performance. The array architecture is completely scalable to larger sizes and decreased cycle times because of the use of nearest neighbor communication between processor cells.

MGAP-2 is the second of three MGAP systems which have been planned for implementation. Here is a summary of all three MGAP generations:

MGAP-1 — Running, 16,384 digit processors (DPs) in 32 1.2 μm CMOS chips. Single 9U$\times$400 mm board operating at 25 MHz. Performance of 0.8 teraops.

MGAP-2 — In design, 65,536 DPs in 32 0.8 μm CMOS chips. Single 9U$\times$400 mm board operating at 50 MHz. Performance of 6.4 teraops.

MGAP-3 — Planned, 262,144 DPs on one MCM. Single board operating at 100 MHz. Performance of 51.2 teraops.

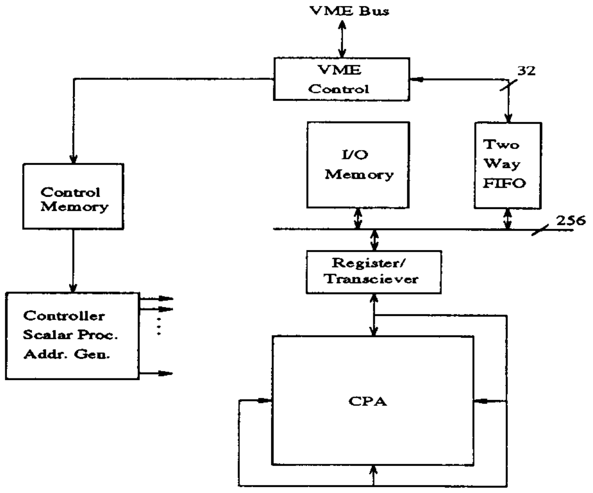

Fig. 1. MGAP-2 block diagram

The platform for the MGAP-1 and MGAP-2 is a SUN-4 style 9U$\times$400 mm VME board, communicating with the host SPARC CPU across the VME bus.

The motivation for developing the next generation MGAP system, MGAP-2, is to provide more DPs with larger local memories, giving applications more processing power and reducing the need for communication with the I/O memories (refer to Figure 1 for the block diagram). To achieve these goals, we have redesigned the VLSI custom processor array (CPA) chip and the support subsystems. The CPA redesign takes advantage of more advanced technology and better design. The redesign of the support subsystems has been driven by observations of MGAP-1's performance and usage, culling away seldom used, expensive features, adding a few features, and optimizing what remains so that the subsystems are fast, efficient, easy to program, and flexible.

This paper is organized as follows. In Section 2 we describe the design and optimization issues of the MGAP-2, both at the chip level and at the system level. Section 3 briefly describes MGAP development tools and some signal processing applications. An efficient algorithm for the 2-D discrete cosine transform (DCT) is discussed in Section 4.

2. MGAP-2 DESIGN

2.1. Custom Processor Array Design

The CPA is the core of the MGAP system. The MGAP-2's CPA chip is a 64 $\times$ 32 mesh of fine-grain processors (called digit processors). Compare this with the MGAP-1 CPA chip, which is a 32 $\times$ 16 mesh, with one-fourth of the DPs available in the MGAP-2. Each DP, one of which is il-

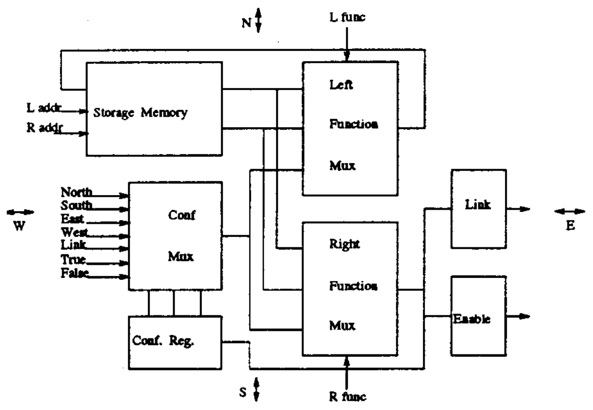

Fig. 2. MGAP digit processor

lustrated in Figure 2, has 32 bits of local memory (twice as much as an MGAP-1 DP). The computational ability of each DP comes from two 8-to-1 function multiplexers which serve as three input universal logic modules (ULMs). Each logic module receives two inputs from the DP's local memory and one input from the configuration multiplexer. The latter input can be a value computed by the cell, by one of the cell's nearest neighbors in the previous cycle, or either of the values 0 or 1. The output of one of the ULMs is connected to the local RAM. The *link* register stores the output of the second ULM. The output of the link register of each DP is connected to its immediate north, south, east, and west neighbors. The 32-bit dual-port memory in the processor is used to store local operands/results and can support two read operations simultaneously. Every DP accesses the same location within its own local memory when global control information is received, both when retrieving and storing values. Each cell contains a three-bit register which provides the selector inputs for the configuration multiplexer. The configuration register is set through global control statements issued to all DPs. Added flexibility is achieved by allowing a DP to conditionally configure itself depending on the output of the right ULM.

From studying the performance of our application programs, we have concluded that the original 16-bit local memory of the MGAP-1 DP is insufficient, requiring extra machine cycles for moving data between the processor array and I/O memories. Within each MGAP-2 DP is a 32 bit dual-ported RAM. The memory operates in three phases. First there is a precharge phase. This is followed by a read phase, where the bit lines are separately connected to the output of inverters within two of the cells inside the RAM. This may cause one or both of the bit lines to fall to 0V. Finally there is a write phase in which one of the bit lines is set to 0V; the other is in the high impedance state. The desired latency for reads is under 2 ns. We used Hspice [6] to test the DP memory design. Our simulations produced read times under 2 ns and write times under 3 ns.

The two ULMs in each DP were designed using pass transistor logic. The schematic is shown in Figure 3. The reader should note that for any of the eight possible input combinations, the selected input has two paths to the output, one through a p-network and one through an n-network. This allows both 5V and 0V to be passed undegraded through the ULMs.

Global control includes: selector inputs for the configuration multiplexer, inputs for both of the ULMs, control signals used within all DPs, and address lines to the dual port RAM. This control information is broadcast to all of the DPs. Each DP contains an *enable* register which can be set through one of the ULMs. As with the configuration register, the enable register can be set conditionally

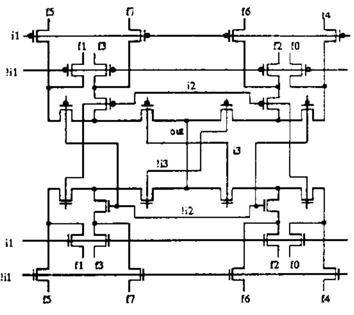

Fig. 3. Universal Logic Module

based on the output of the right ULM. A DP is active on any cycle if and only if its enable register is set. There are three general types of instructions on the MGAP-2: a compute instruction, a configure DP instruction, and an enable DP instruction. Only those DPs which have their enable register set will update their current state based on the instruction issued. Figure 4 shows the layout for one pair of DPs. It is $193\lambda \times 978\lambda$. One of the motivations for our floorplan strategy was that it allowed a pair of DPs to be mated as shown.

Metal-2 is run vertically over the array to deliver the ULM minterm inputs to the function multiplexers of each cell, this is an additional benefit of our floorplan strategy. Two of the function input bits come from the sense amplifier and one from the configuration multiplexer. All of these signals are run in polysilicon horizontally through the cell. With 1.2 μm technology this did not seem to introduce significant delays into the design. The ULMs were tested with their full load and displayed a worst case delay of 4 ns. All of the transistors in the function multiplexer are 4λ. The total size of a single ULM is $80\lambda \times 70\lambda$.

2.2. System-Level Design

The major system-level improvements are I/O memory organization, I/O memory/CPA communication, and control logic. Other than the CPA, system logic is off-the-shelf MSI and LSI, and programmable PLDs. The MGAP-1 used so-called ping-pong I/O memories for concurrent CPA I/O. Removing this feature allowed us to recover a fair amount of board real estate. The MGAP-1 communication buses between the I/O memories and CPA were designed to support what we thought were all the important dataflows. Hindsight shows that the buses weren't flexible enough, resulting in the current arrangement of bidirectional I/O in all four directions (refer to Figure 1).

Using off-the-shelf parts in MGAP-1 led to control pipelines with unequal lengths. This complicated the control model and forced us to insert a small number of no-ops to synchronize the I/O memory and CPA pipelines. The MGAP-2 will consolidate much of the control logic into a few FPGAs, making it possible for us to equalize the pipeline lengths.

Finally, the MGAP-1 VME slave interface is not optimized for maximum bandwidth. Some applications, such as image compression/decompression, utilize the MGAP as a pure pipeline. For this class of applications, the VME interface is a real bottleneck. The MGAP-2 will improve on VME bandwidth through DMA and block transfers. The two-way FIFO, which holds four bit planes in each direction, will also be extremely useful in these pipeline configurations.

3. MGAP DEVELOPMENT TOOLS AND DSP APPLICATIONS

In this section, we briefly discuss our suite of MGAP application development tools and highlight some applications.

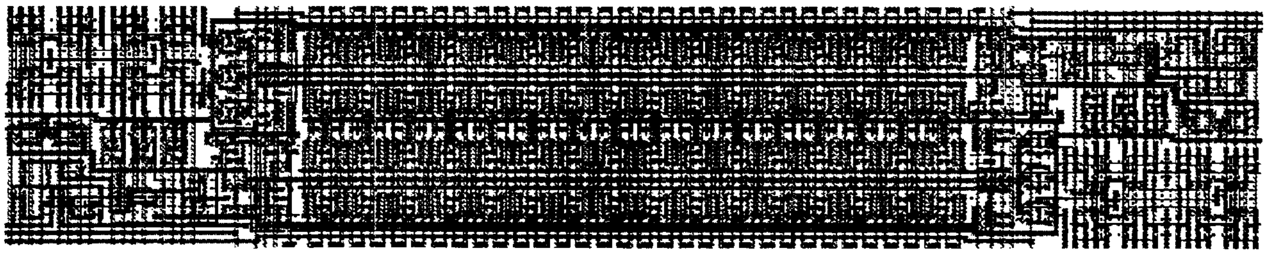

Fig. 4. Layout of Two Digit Processors

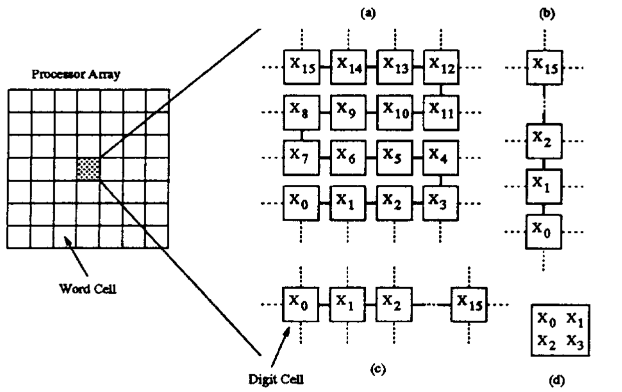

Fig. 5. Four possible word cell configurations: (a) snake; (b) vertical; (c) horizontal; and (d) local.

Table 1
Basic operation command counts for MGAP-2. Precision, p, is 16 MRR4 digits.

Operation	# Instructions	Ops/Second
Digit shift	8	$\approx 104 * 10^9$
Word shift	$3\sqrt{p}+2$	$\approx 56 * 10^9$
Comparison	$14\sqrt{p}+33$	$\approx 8 * 10^9$
Addition	25	$\approx 32 * 10^9$
Multiplication	$135p$	$\approx 376 * 10^6$

The tools may be broken down into three categories: low-, middle-, and high-level.

Digit cells are joined together, through programming their configurations, to form arbitrary precision word cells (see Figure 5) and arithmetic at the word level is performed using the maximally redundant radix four (MRR4) representation. This representation yields significant parallelism at the digit level, because many arithmetic operations can be performed in constant time. We have developed, and continue to develop, low-level arithmetic library routines for exploiting digit-level parallelism. Table 1 lists command counts for several of these fundamental operations. In the table, p is operand precision.

At the middle-level, we have have a suite of programming tools: graphical user interface, parallelizing compiler, assembler, linker/loader, simulator, and test and diagnostic software. The supported high-level programming language, *C++, is a cross between C++ and C*. Essentially, C*'s notion of a *shape* has been combined with a C++ compiler. A bit is the only primitive data type needed for the MGAP array. All other data types are derived from the bit type: digits are derived from bits, words from digits, and shapes from words.

Our high-level development efforts center about parallel algorithm design for solving specific problems on the MGAP. At this level, we can leverage two distinct forms of parallelism: digit- and word-parallelism. Digit-parallelism is obtained by using the low-level MRR4 arithmetic routines, while word-parallelism results from taking advantage of the MGAP architecture in designing new problem solving algorithms.

For maximum performance, it is imperative that high-level algorithms take advantage of the MGAP architecture's strengths, such as local communication and fast addition and subtraction, and avoid its weaknesses. An edge detection algorithm, using the Hough transform, which we have have developed, completely avoids expensive multiplications in favor of cheap additions and subtractions [8]. In cases where complex, expensive operations cannot be avoided or there is insufficient high-level parallelism, digit-level parallelism will still provide a significant speedup. This is illustrated by our one-dimensional wavelet transform algorithm [9]. Some other signal and image processing applications are discussed in [5, 10, 11].

4. DCT ON THE MGAP-2

The 2-D DCT is a powerful tool for image compression. We show how an algorithm based upon the so-called small-n algorithms and using the minimum number of multiplications for the DCT can be mapped onto the MGAP-2. More detail can be found in [12].

The small-n algorithms efficiently compute DFTs [13] and can be used, with a small modification, to compute the discrete Hartley transform (DHT) [14, 15]. The DHT thus computed can be used to compute a 1-D DCT using the relationships established in [16]. The 1-D DCT computation may be expressed as:

$$DCT(x) = ASC'TPx,$$

where x is a sequence of length N, P is a permutation matrix, T and S contain elements from $\{-1,0,1\}$, C' is diagonal, and A embodies the relationship between the DHT and DCT.

Using column-row decomposition, the 2-D DCT is computed using:

$$DCT(X) = (ASC'TP(ASC'TPX)^T)^T.$$

Matrices T and P are fixed and can be pre-computed as T'. Both T' and S contain elements from $\{-1,0,1\}$, so multiplication is replaced with systolic summation. Matrix C' is diagonal and we replace matrix multiplication with an element-by-element scaling after pre-copying each diagonal element to the remaining elements on the row. The multiplication by A involves a butterfly-style routing, two multiplications, and an addition. Figure 6 illustrates the algorithm on an MGAP sub-array. Numerous instantiations of the algorithm are processed in parallel on the array.

Time analysis for the MGAP-2 is given in Table 2. The steps given are for one 1-D pass. The "Total" row is adjusted for the 2-D DCT. "Whole image" is the time for a complete 256×256 pixel image. For realtime video processing, the 8×8 2-D DCT of 256×256 pixel images should be less than 32.55 μs. The MGAP-2 beats this by an order of magnitude.

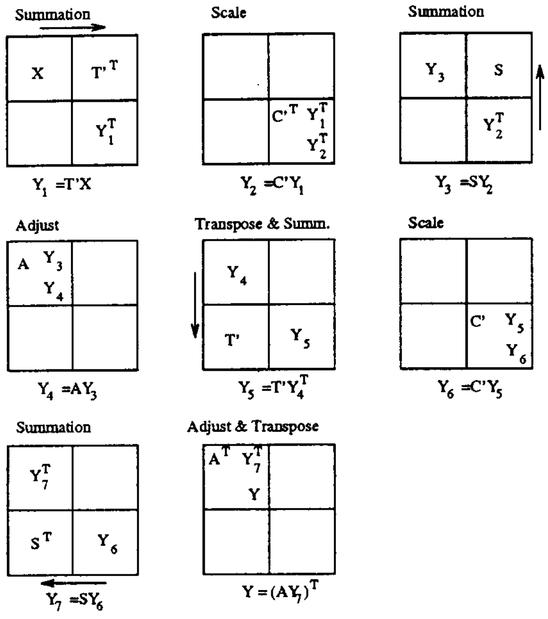

Fig. 6. DCT dataflow on an MGAP subarray

Table 2
MGAP-2 256 × 256 DCT time analysis

Step	8×8	16×16
Load image	0.2 μs	0.88 μs
Summation	0.35 μs	3.65 μs
Scaling	0.29 μs	1.47 μs
Summation	0.35 μs	3.65 μs
Adjust	0.63 μs	4.48 μs
Total	3.44 μs	25.38 μs
Whole image	3.52 ms	6.4 ms

5. CONCLUSION

The MGAP-2 has four times as many processors and twice as much local memory as the MGAP-1. The major optimization issues employed in the MGAP-2 CPA are utilizing advanced VLSI technology, designing a more compact, higher capacity dual-port local memory, and altering pass-transistor ULMs and latches. The clock frequency of the MGAP-2 has been increased from 25 MHz to 50 MHz.

At the system level, the structure of the I/O subsystem has been streamlined for efficiency without sacrificing processor array performance. This streamlining has been effected by reducing the number of I/O memories from two to one and simplifying the communication structure between the array and the I/O memory. The control subsystem has been optimized to handle the decreased cycle time, it has been made uniform, and we have worked to make its design even more flexible than in the MGAP-1. VME bus bandwidth will be improved, increasing the performance of pure pipelined applications.

Of course, the most powerful system imaginable is worthless if it cannot be used. To fully utilize the MGAP's potential, we have implemented, and continue to implement, a useful set of MGAP support tools. These tools make the MGAP quite usable. Finally, we have developed several useful applications which demonstrate the power and viability of the MGAP system. We, and others, continue to create novel, suitable uses for the system. This work was supported under NSF grants CDA-8914587 and MIP-9408921.

REFERENCES

[1] M. J. Irwin and R. M. Owens, "A Micro-Grained VLSI Signal Processor", In *ICASSP-92*, pp 641–644, March 1992.

[2] M. J. Irwin and R. M. Owens, "A two-Dimensional, Distributed Logic Processor", *IEEE Transactions on Computers*, 40(10):1094–1101, October 1991.

[3] R. S. Bajwa, R. M. Owens, and M. J. Irwin, "Area Time Tradeoffs in Micro-Grain VLSI Array Architectures", *IEEE Transactions on Computers*, 43(10):1121–1128, Oct. 1994.

[4] R. M. Owens, M. J. Irwin, T. P. Kelliher, M. Vishwanath, and R. S. Bajwa, "Implementing a Family of High Performance, Micrograined Architectures", In *Application Specific Array Processors*, August 1992.

[5] R. M. Owens, M. J. Irwin, C. Nagendra, and R. S. Bajwa, "Computer Vision on the MGAP", in *Proc. CAMP'93*, 1993.

[6] Meta-Software. *HSPICE User's Manual H9001.* Campbell, CA, 1990

[7] R. M. Owens, T. P. Kelliher, and M. J. Irwin, "Building High Performance Signal Processors Cheaply and Quickly," in *Proc. 1993 IEEE Workshop on VLSI Signal Processing*, Oct. 1993.

[8] C. Nagendra, M. Borah, M. Vishwanath, R. Owens, and M. J. Irwin, "Edge Detection using Fine-Grain Parallelism in VLSI," in *Proc. ICASSP '93*, vol. 1, pp. 401–404, Apr. 1993.

[9] C. Nagendra, M. J. Irwin, and R. Owens, "Digit Pipelined Discrete Wavelet Transform," in *Proc. ICASSP '94*, Apr. 1994.

[10] R. Bajwa, R. M. Owens, and M. J. Irwin, "Image Processing with the MGAP: A Cost Effective Approach," in *Proc. IPPS'93*, pp. 439–443, Apr. 1993.

[11] H. N. Kim, M. J. Irwin, R. M. Owens, and C.-M. Wu, "Dynamic Space Warping Algorithms on Fine-Grain Array Processors," in *Proc. IPPS'94*.

[12] H. N. Kim, M. Borah, R. M. Owens, and M. J. Irwin, "2-D Discrete Cosine Transforms on a Fine Grain Array Processor," in *Proc. 1994 IEEE Workshop on VLSI Signal Processing*.

[13] D. Elliot and K. Rao, "Fast Transforms: Algorithms, Analyses, Applications," Academic Press, 1982.

[14] R. Bracewell, "The Hartley Transform," Oxford University Press, 1986.

[15] C. Chakrabarti and J. JáJá, "Systolic Architectures for the Computation of the Discrete Hartley and Discrete Cosine Transforms Based on Prime Factor Decomposition," *IEEE Trans. Comp.*, vol. 39, pp. 1359–1368, Nov. 1990.

[16] H. Malvar, "Fast Computation of Discrete Cosine Transform Through Fast Hartley Transform," *Electronic Letters*, vol. 22, pp. 352–353, Mar. 1986.

[17] T. Fountain, "An Evaluation of Some Chips for Image Processing," in L. Uhr, K. Preston, S. Levialdi, and M. J. B. Duff, editors, *Evaluation of Multicomputers for Image Processing*, ch. 4, Academic Press, 1986.

[18] F. A. Gerritsen, "A Comparison of the CLIP IV, DAP, and MPP Processor-Array Implementations," in M. J. B. Duff, editor, *Computing Structures for Image Processing*, ch. 2, Academic Press, 1983.

[19] R. M. Hord, *Parallel Supercomputing in SIMD Architectures*, CRC Press, 1990.