A SYSTOLIC ARRAY COMPUTER

E. Arnould, H. T. Kung, O. Menzilcioglu and K. Sarocky

Department of Computer Science, Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

A high-performance systolic array computer has been designed by CMU and is currently under construction. The first copy of the machine, to be built by CMU together with its industrial partners before the end of 1985, will incorporate a programmable systolic array of ten linearly connected cells. Each cell in the systolic array is capable of performing 10 million floating-point operations per second (10 MFLOPS), giving the total machine a peak performance of 100 MFLOPS, or higher if additional cells are used. This particular systolic array computer is named *Warp*, suggesting that it can perform computations at a very high speed. The 10-cell systolic array, with one cell implemented on one board, can process 1024-point complex FFTs at a rate of one FFT every 600 μs. Under program control, the same array can perform many other primitive computations in signal, image, and vision processing, including two-dimensional convolution, dynamic programming, and real or complex matrix multiplication, at a rate of 100 million floating-point operations per second. Users may view the systolic array as an array of conventional "array processors," which can efficiently implement not only systolic algorithms where communication between adjacent cells is intensive, but also non-systolic algorithms where each cell operates on its own data independently from the rest. This paper describes the hardware organization of the Warp machine.

1. INTRODUCTION

Since the beginning of 1984, CMU has been working on the design of the Warp machine, which is intended to be a powerful computing engine for many low-level signal and image processing tasks, encountered especially in the vision area. The Warp architecture was heavily influenced by application needs as perceived by vision researchers at CMU [6]. Passive navigation for mobile robots or autonomous vehicles will be among the first demonstrations of the machine when it is built.

The Warp machine is composed of three parts—the host, the Warp processor array, and the interface unit (IU), as depicted in Figure 1. There is also an external program development workstation that can prepare and compile code for the Warp machine.

The Warp processor array provides the bulk of the computing power for the Warp machine, and is also called the Warp processor in the paper. This is a programmable, one-dimensional systolic array, whose cells are all replicas of the same processor, called the *Warp cell*. During computation, data, addresses and controls all flow between neighboring cells; there are no global communications at all. Because of this regularity and locality in communication, the Warp processor array can have hundreds of cells without major difficulties.

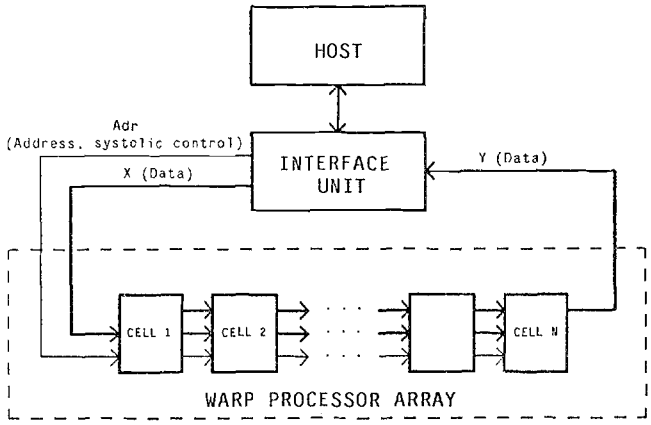

Figure 1. Warp machine overview

The Warp processor array supports two modes of programming. In the *systolic mode*, the array can efficiently implement a wide variety of systolic algorithms, including those for the FFT, matrix multiplication, and two-dimensional convolution [3]. In the *local mode*, the array can implement many non-systolic algorithms where each cell performs a complete computation on its own data independently from the rest. The local mode can be useful in applications for which either appropriate systolic algorithms are not available or the programmer finds it easy to program each cell individually. In this case, each cell can be programmed as a usual sequential machine working, say, on a separate region of an image. It is possible that all the cells of the Warp processor array will execute the same code, as in an SIMD machine, or different code as in an MIMD machine. To support the local mode, we have made the Warp cell a programmable processor, with a microsequencer, a writeable microcode store, register files, a data memory, and a rich interconnection.

The Warp cell, however, is not intended to be a general-purpose processor. In particular, it does not have powerful facilities to generate data-dependent addresses for its data and control memories, although it can generate simple addresses quite efficiently [6]. Complex, data-independent address patterns are usually generated by the interface unit outside, and then fed "systolically" to the cells of the Warp processor array along the Adr path.

This paper describes the hardware aspect of the Warp machine. More specifically, the Warp processor array, host, and interface unit are described in Sections 3, 4 and 5, respectively, and some concluding remarks are given in the last section.

CH2118-8/85/0000-0232 $1.00 © 1985 IEEE

Discussions on system software, program development supports, and applications for Warp will be covered by other papers.

2. WARP PROCESSOR ARRAY

As stated earlier, the Warp processor array is a one-dimensional systolic array, whose cells are all replicas of the same processor called the Warp cell. The entire array of cells is synchronized by a single clock issued by the interface unit. The following is a description of the Warp cell.

2.1. Warp Cell Datapath

The datapath of the Warp cell is shown in Figure 2. All datapaths are 16-bit wide. However, the Warp cell is architecturally a 32-bit processor, since it performs operations on 32-bit words. This is accomplished by implementing each 32-bit operation in two consecutive minor cycles on the 16-bit datapaths. The Warp cell has a (major) cycle time of 200 ns, which is composed of two minor cycles of 100 ns each. When dealing with 32-bit operations, the programmer sees only the major cycles and does not have to explicitly deal with minor cycles.

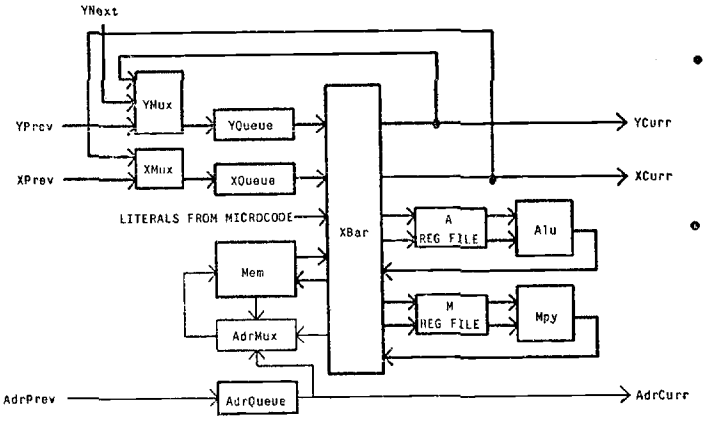

Figure 2. Warp cell datapath

In the rest of this section, operations and words refer to 32-bit operations and words, respectively, and cycles refer to 200 ns major cycles, unless otherwise stated.

Following are descriptions of the major blocks in the Warp cell and their functions.

- *Arithmetic units* (Alu, Mpy). These are implemented with the Weitek 32-bit floating-point multiplier and ALU chips [6]. The Alu is not a general purpose ALU, but it does include comparison, normalization and absolute value operations. To get the best-possible throughput for the Warp cell, these chips are used in their pipelined mode. That is, a chip starts a new 32-bit arithmetic operation every cycle, although the result of an operation will not emerge from the chip's output ports until five cycles after the operation starts.

- *Register files* (A, M). These are general register files for buffering operands for the arithmetic units. Each register file is implemented with a copy of the Weitek 32×32 six-port register file chip. The M register file can also compute approximate inverse and inverse square root functions with 8-bit precision, using the look-up table on the Weitek register file chip.

- *Data Memory* (Mem). Having a local memory at each cell for buffering data, implementing look-up tables, or storing constants and intermediate results, can substantially reduce the I/O bandwidth requirement of the cells. Also, by using the local memory to store temporary data, a cell can be multiplexed to implement the functions of multiple cells in a systolic algorithm. As a result the Warp processor can implement algorithms designed for two-dimensional systolic arrays, despite the fact that it is a one-dimensional processor array. The data memory in each cell has 4K words, and can be expanded up to 16K words. The memory can perform both a read and a write simultaneously every cycle, using addresses selected from the AdrQueue (for external addresses), crossbar (for computed addresses or addresses from microcode), or the data memory itself (for indirect addresses).

- *Input queues* (XQueue, YQueue, AdrQueue). These queues (of 128 words each) are provided mainly to implement programmable delays to ensure that data and address streams are properly synchronized over the Warp processor array, as required by systolic algorithms. To implement programmable delays, these queues are equipped with counters that can automatically increment read and write addresses every cycle. When not implementing programmable delays, the queues can be used as scratchpad register files, by setting or holding the counters via microcode.

- *Crossbar* (XBar) A completely general crossbar is used to link the different components described above. The crossbar has eight output ports, and six input ports including one that accepts literals from microcode. The crossbar can be reconfigured every cycle under microcode control to allow each output port to receive data from *any* of the six input ports.

- *Input muxes* (XMux, YMux). Normally, every cycle a cell inputs data from the previous cell (i.e., the cell to the left) via YPrev and XPrev, and outputs data to the next cell (i.e. the cell to the right) via YCurr and XCurr. Input muxes allow two additional modes to be implemented—the wrap-around mode and bi-directional mode. In the wrap-around mode the outputs of a cell are fed back to the inputs, so that one cell can be used to implement the function of several adjacent cells. This mode therefore increases the virtual size of the array for problems requiring larger array size. In this case, since each cell is multiplexed in time to perform the function of several cells, it does not communicate with other cells or the interface unit as frequently as before. This implies that the Warp processor will require only a reduced I/O bandwidth with the host. In the bi-directional mode, the Y outputs of a cell are used by the cell to the left, thus data on the Y path flow in the direction opposite to those on the X path. Bi-directional dataflows are used in many systolic algorithms [1].

2.2. Warp Cell Control

The Warp cell is microcode controlled. Implemented with the AMD 2910A, the sequencer supports branching, one-level looping, and nested subroutine calls. There are 4K words of microcode memory in the present system capable of holding about 100 simple routines. In general, one can expect that the entire set of routines required in one application session can be loaded into the microcode memory at the same time, eliminating the need to stop and reload during processing.

At every cycle, the condition code input to the sequencer can be selected from six condition codes determined by Alu.Sign, Alu.Exp and Sys.Cc, or can be directly set to true or false by microcode. The Alu sign bit (Alu.Sign) is useful for implementing inequality comparisons

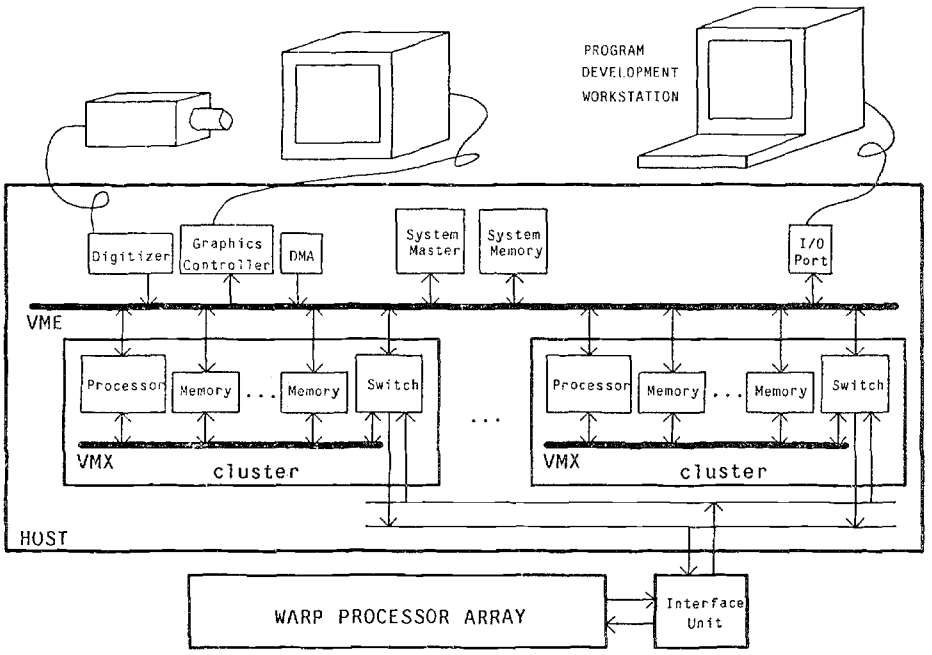

Figure 3. Host of the Warp machine

of floating-point numbers. The most significant two bits of the Alu exponent field (Alu.Exp) can be used to implement equality comparisons of integer values (represented in floating-point format). Four systolic condition codes, determined by the two systolic control bits (Sys.Cc) that are sent from cell to cell, are useful to indicate (to each cell) the beginning and end of loop, address or data patterns. The systolic condition bits can be generated by either the interface unit external to the Warp processor array, or by the microcode of an individual cell.

3. HOST

The host is mainly responsible for all the on-line operations that are needed to provide the Warp Processor with the necessary flow of data, through the interface unit. The host may also perform computations that cannot be efficiently done on the Warp processor array.

As shown in Figure 3 the host is based on the VMEbus, which is a family of three buses—VME, VMX, and VMS. The host has two or more identical clusters, each consisting of a local VMX bus, a 68020-based processor board, several memory boards, and a switch unit. All the boards in a cluster have both VME and VMX interfaces, and can communicate with boards outside the cluster via the VME bus.

During computation, two clusters may work in parallel, each handling a uni-directional flow of data to or from the Warp processor, through the interface unit. In this case the processor board in the cluster that sends data to the interface unit can generate random addresses to retrieve data from the memories and input them into the interface unit, via the switch unit. The switch unit in the other cluster that receives data from the interface unit can generate sequential addresses by a DMA device to store data coming from the interface unit into the memories.

While all this is happening, data may be shuffled by one of the processors in the host. For example, 8-bit or 16-bit integers may be packed into one, 32-bit wide data word (see the integer-to-floating-point conversion discussion regarding the interface unit below).

Other parts of the host include the digitizer, graphics and DMA boards on the VME bus that handle the I/O communication with TV camera(s) and the graphics display.

The system master on the VME bus controls the host and performs those computations that are not done on the Warp processor. In particular, it handles interrupts, coordinates the processors and DMA devices, and performs decision-making processes based on the results computed by the Warp processor.

4. INTERFACE UNIT

The interface unit (IU) handles all communication between the host and Warp processor, and provides all the control signals necessary to drive the Warp processor. As depicted in Figure 4, the IU has the following major components:

- Input/output FIFOs, each 32-bit wide and 512 deep, used to buffer data between the host and the Warp processor.

- Integer-to-floating-point and floating-point-to-integer converters. For signal, image and vision processing input/output data are usually 8-bit or 16-bit (signed or unsigned) integers. These integer inputs must be converted into 32-bit floating-point numbers before the (floating-point) Warp processor can process them, and the floating-point results computed by the Warp processor must be converted back into integers. Having these converters inside the IU has the following pleasant side effect: because the IU communicates with the host in 8-bit or 16-bit integers rather than in the corresponding 32-bit floating-point numbers, the data bandwidth requirement between the host and IU is reduced by a factor of 4 or 2, respectively. It is often due to this I/O bandwidth reduction one is able to fully utilize the Warp processor array in spite of the fact that the present hardware of the Warp machine provides only a moderate data bandwidth between the host and IU.

- Input/output crossbars. By programming the crossbars, the Warp processor can receive its inputs from either of the two input ports in the IU. Similarly, the Warp processor can deliver its outputs to either of the two output ports in the IU.

- Address generator. This component sends 16-bit addresses to the Warp at the rate of one per 100 ns. The addresses are either precomputed addresses stored in a table (32 K bytes), or ones generated on-the-fly.

- Host interface. This unit contains three 32-bit registers: the status register, the control register, and the interrupt register. These registers can be read by the host at any time. The status register contains the current status of both the Warp processor and the IU. The control register is used to send instructions to functional blocks of the IU as well as to the Warp processor. The interrupt register can be set up so that the IU has the ability to interrupt the host on eight different events. They include full/empty status signals for the input/output FIFOs, certain patterns of systolic control bits from the Warp processor, and parity error and arithmetic exception signals from the Warp processor.

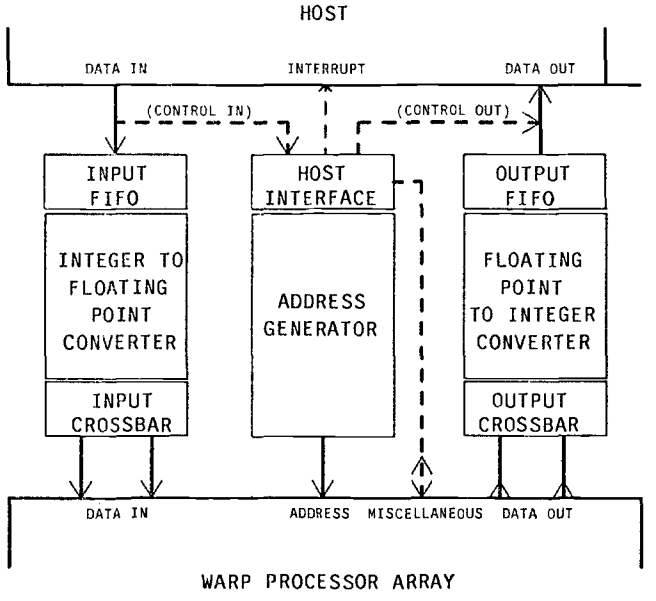

Figure 4. Interface unit block diagram

The IU has a similar micro-control structure to the Warp cell, except that for implementing nested loops, the AMD 2910A is augmented with a counter and 16 registers.

5. CONCLUDING REMARKS

A microinstruction language called W1, together with its assembler and simulator for the Warp processor array and IU are available; they are being used to develop application programs for Warp. Another language higher than W1 is under development. This language, called W2, will hide low-level details such as register allocation, memory management, scheduling, and packing of micro-operations. Eventually all application programs will be written in or compiled to W2 code, which will then get translated into code ready to be executed by Warp or its simulator.

As a separate project, CMU is designing a custom chip—the Link and Interconnection Chip (LINC)—which will substantially reduce the implementation cost of the Warp cell [2]. LINC is a super "glue" chip whose function is to serve as an efficient link between system functional modules, such as arithmetic units, memories, and I/O ports. Operating at the target cycle time of 100 ns, the LINC makes it possible to implement Warp-like cells with much reduced chip counts. More specifically, using the LINC, four or more Warp-like cells can fit a single PC-board. The chip is being laid out in CMOS technology, and is expected to be fabricated by the summer of 1985.

CMU is building another processor capable of performing fast divisions and square roots. Augmented with this processor, the Warp processor array can efficiently carry out a number of difficult matrix operations such as solving covariant linear systems, a crucial computation in real-time adaptive processing [1, 4].

ACKNOWLEDGMENTS

Many people have contributed to the evolution of the Warp architecture and hardware, including Marco Annaratone, C. H. Chang, Peter Dew, Thomas Gross, Takeo Kanade, Monica Lam, Andreas Nowatzyk, T. M. Parng, David Sher, and Jon Webb. The research was supported and in part by the Office of Naval Research under Contracts N00014-76-C-0370, NR 044-422 and N00014-80-C-0236, NR 048-659, and in part by the Defense Advanced Research Projects Agency (DoD), monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539, and Naval Electronic Systems Command under Contract N00039-85-C-0134.

REFERENCES

[1] Heller, D.E. and Ipsen, I.C.F.
Systolic Networks for Orthogonal Equivalence Transformations and Their Applications.
In *Proceedings of Conference on Advanced Research in VLSI*, pages 113-122. Massachusetts Institute of Technology, Cambridge, Massachusetts, January, 1982.

[2] Hsu, F.H., Kung, H.T., Nishizawa, T. and Sussman, A.
LINC: The Link and Interconnection Chip.
Technical Report, Carnegie-Mellon University, Computer Science Department, May, 1984.

[3] Kung, H.T.
Systolic Algorithms for the CMU Warp Processor.
In *Proceedings of the Seventh International Conference on Pattern Recognition*, pages 570-577. International Association for Pattern Recognition, 1984.

[4] Kung, H.T.
Systolic Algorithms.
In Parker, S.V. (editor), *Large Scale Scientific Computation*, pages 127-139. Academic Press, 1984.

[5] Webb, J. and Kanade, T.
Vision on a Systolic Array Machine.
To appear in the Proceedings of the Tanque Verde Workshop on High-Speed Image Processing, Tucson, Arizona, 1984, edited by K. Preston and L. Uhr.

[6] Woo, B.Y., Lin, L., Fandrianto, J. and Sun, E.
A 32 Bit IEEE Floating-Point Arithmetic Chip Set.
In *Proceedings of 1983 International Symposium on VLSI Technology, Systems and Applications*, pages 219-222. 1983.