iWarp Architecture Overview

May, 1988

Intel Corporation
5200 NE Elam Young Parkway
Hillsboro OR 97124-6497

1. Introduction

iWarp is a single-chip integrated version of the Warp systolic architecture, developed by Professor H. T. Kung, and the staff and students of Carnegie Mellon University. Significant enhancements have been incorporated in the iWarp component that extend the applicability of the architecture to a much broader class of applications than were supported in previous systolic systems. These features, their corresponding benefits, and a comparison with the Warp system are summarized as follows:

	iWarp Chip Capability	Warp Board Comparison
High Performance:	- 20 MFLOP 32-bit	2x increase
	- 10 MFLOPS 64-bit	Not supported
	- Scalar arithmetic section	7 stage pipeline
	- 20 MIPS Integer Logical Unit	2x increase
	- 320 MByte/Sec Systolic I/O	4x increase
	- 160 MByte/Sec Memory BW	2x increase
	- 1D & 2D arrays to 100's of cells (processors)	10 cells
More Flexible Programming Model	- Message passing communication	Not supported
	- Data Spooling & Streaming	Not supported
	- Broadcast, Gather, Scatter Communication Operations	Nearest neighbor
	- Asynchronous Queuing	Not supported
	- Program Cache from local memory	Fixed Program Store
Smaller size, lower cost:	- Single Chip & Memory	Discrete Circuit Board
	- 2" x 5" board 'footprint'	12" x 18" board
High performance I/O	- Up to 80 MBytes/sec./connection	12 MBytes/sec. Max.
	- I/O direct to Array	Special I/O Subsystem
	- Multiple access ports up to 1/2 number of cells	Using VME bus
	- Independent Communication/ Computation Agents	Communication linked to computation
Tools and Languages Support	- Fortran, C with message passing & systolic extensions	W2 Systolic High Level Language
	- Assembler	Supported
	- Debugger	Supported

A common thread among all systolic architectures is a near unity ratio between communication and computation. This allows full performance computation for fine-grain pipelined (systolic) algorithms, even though only one or two arithmetic operations may be performed in each cell as the data moves through the array. An example is a convolution operation for which one tap of the operator, $h(t)$, is stored in each cell, and data, $x(t)$, flows through the array. Each cell performs a multiply by its value of $h(t)$ and sums this result with the partial sum from the previous cell. Thus,

$x(t)$ goes in one end of the array and $\hat{x}(t)$ comes out the other. iWarp can support this application with all cells executing at peak performance.

A significant iWarp enhancement that distinguishes it from other systolic architectures is the support of data queuing mechanisms. These support message passing and global communication operations are required for more sophisticated applications than the example above. For instance, this capability is essential for performing algorithms like Gaussian Elimination, found in the LINPACK benchmark. In this case the matrix is distributed across the array by "dealing out" its columns to cells in the array. The cell with the left most column (the column to be eliminated) performs the pivot operation and then broadcasts the resulting X vector to all other cells in the array. They then perform a SAXPY (single-precision $y = ax + y$)or DAXPY (double-precision $y = ax + y$) operation against all remaining columns. The process continues until all elements below the diagonal of the matrix have been eliminated. Cells take turns performing the pivot operation, while all cells do _AXPYs on their resident columns.

If all three variables (X, Y and the Y result) are stored in memory, as would be the case for the message passing model, the computation will be limited in performance to the rate at which memory can be accessed. Eliminating one of the memory operations by combining the systolic and message passing communication models will nearly double performance. Thus, the _AXPY operation executes systolically. Since the X vector is used by all cells in the same order, it can be distributed as a systolic computation variable, reducing the number of memory cycles from three to two. This produces a corresponding halving in the number of arithmetic cycles required to do each _AXPY. (Two memory operations can be performed in a single CA arithmetic instruction, so the reduction in memory operations from three to two, make it possible to perform _AXPY in a single arithmetic cycle.)

2. The iWarp Cell/Component

The iWarp cell is composed of the iWarp component and its local memory (LM). The iWarp component is a single chip implementation of the iWarp processor architecture. Local memory is external to the iWarp component, and in a typical configuration requires only 18 chips (64-bits of data plus parity).

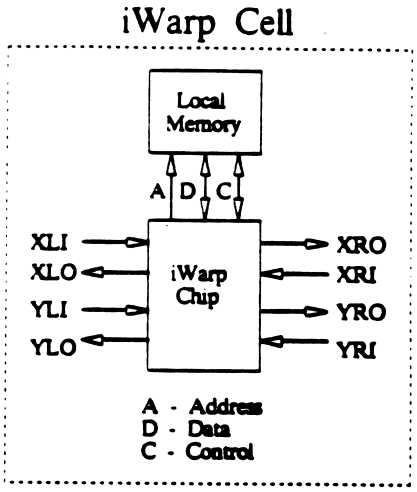


Figure 2.1: iWarp cell

iWarp processors are logically composed of two parts: a **computation agent** and a **communication agent**. (see Figure 2.2)

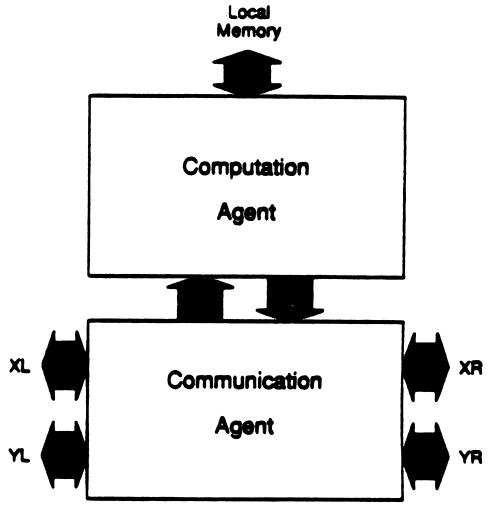


Figure 2.2: Computation and Communication Agents

The purpose of the computation agent is to execute a cell's part of some parallel programmed algorithm that has been partitioned over an array of iWarp cells. The computation agent is realized as a programmable processor composed of a number of physical units: Instruction

Sequencing Unit (ISU), the Floating Point Unit (FPU) consisting of an Adder (FPA) and Multiplier (FPM), the Integer/Logical Unit (ILU), the Local Memory Unit (LMU), and the Streaming/Spooling Unit (SSU). The computation agent provides instructions that allow several of these units to function in parallel during a single instruction.

The purpose of the communication agent is to provide largely independent, low level management of the communication mesh by which the cell is connected to its neighbor cells in an array. The communication agent is realized as a semi-autonomous communication controller called the Pathway Unit. Based upon setup information provided by software running on the computation agent, the pathway unit handles communication traffic through the cell decoupled from (and logically asynchronous to) computation unit operations. Only at well defined points (e.g., arrival of data wanted by a program running on the computation agent), do they need to synchronize. This synchronization is transparent to the programmer.

While there is significant potential for low level, intracellular parallelism achieved by parallel scheduling of a cell's multiple functional units and between the computation and communication agents, it is typically invisible to the algorithm designer. Algorithm level parallelism is usually achieved via partitioning an algorithm over several cells, each cell's algorithm having one basic thread of control. Requirements for synchronization between subalgorithms on different cells varies widely depending upon the model of parallel algorithm partitioning employed (see Figure 2.3). One way of explaining this variation is based upon the number of computations/data point/cell and its inverse, the number of computations/cell before new data is needed (implying some communication). At one end of the spectrum are "systolic" algorithms which are quite "fine-grained" requiring only 1-10 computations/data point/cell before passing the result on and requiring new input. At the other end of the spectrum are memory-based, matrix algorithms which are quite "coarse-grained" requiring 100-1000 computations/data point/cell. iWarp effectively spans this range because it supports both the systolic and message passing communication models implied by these computational requirements.

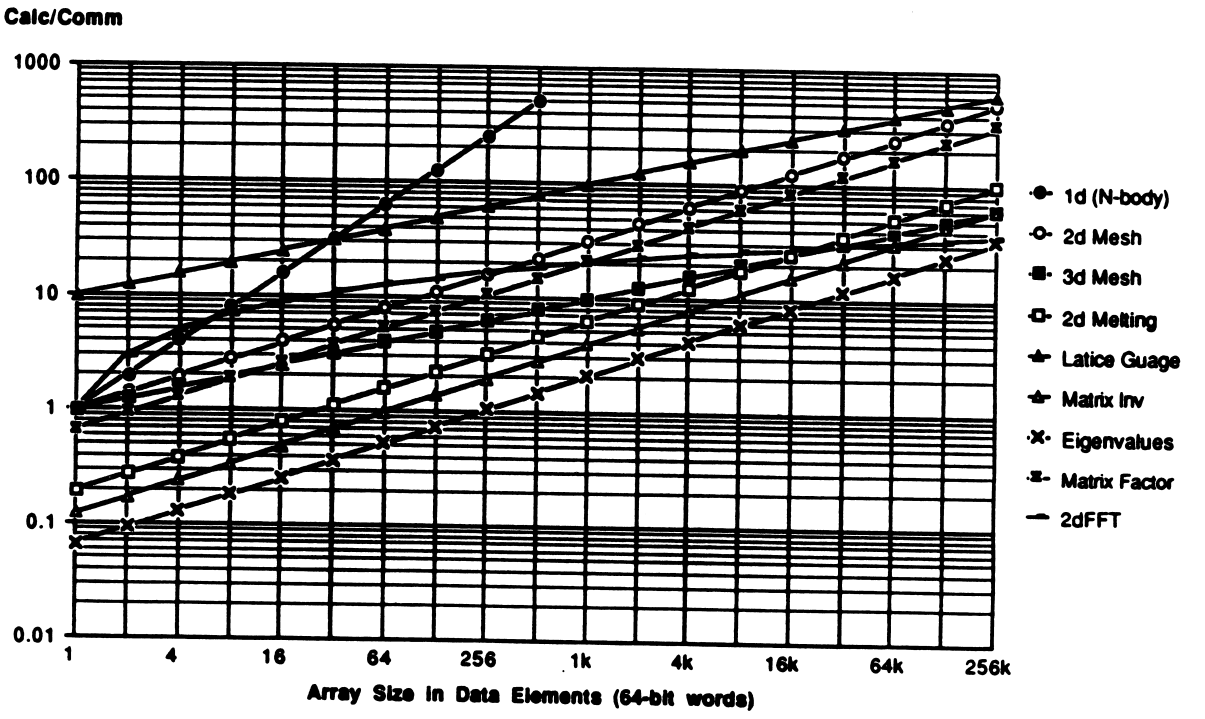


Figure 2.3: Granularity for a variety of algorithms

Fine-grained computations require synchronization on a word by word basis between participating cell programs. To support this synchronization at the hardware level, the only real difference between a "systolic" processor and any other type of processor is that a program running on a systolic processor can, on an instruction by instruction basis, synchronize with one or more inter-cell buses for the purpose of reading or writing a data value. On iWarp, this synchronization and data value I/O are accomplished, from the program's point of view, as references to particular registers. ***Thus, a single iWarp instruction can consume an input, compute and generate an output in a single arithmetic cycle.*** Support for this synchronization and data value I/O is provided by each cell's pathway unit.

Coarse-grained computations require synchronization far less often (e.g., over large blocks of data) and can be supported by memory to memory message passing. For example, a message containing the entire block of data to be manipulated can be passed from some source cell to the cell where the computation is to be done and then the computation can be started. This message passing may even be accomplished during some prior computation (e.g., overlapping I/O and computation) transparent to the programmer. On iWarp, support for such memory to memory message passing is provided via "spooling" by the Streaming/Spooling Unit (SSU) and the Pathway Unit (PWU).

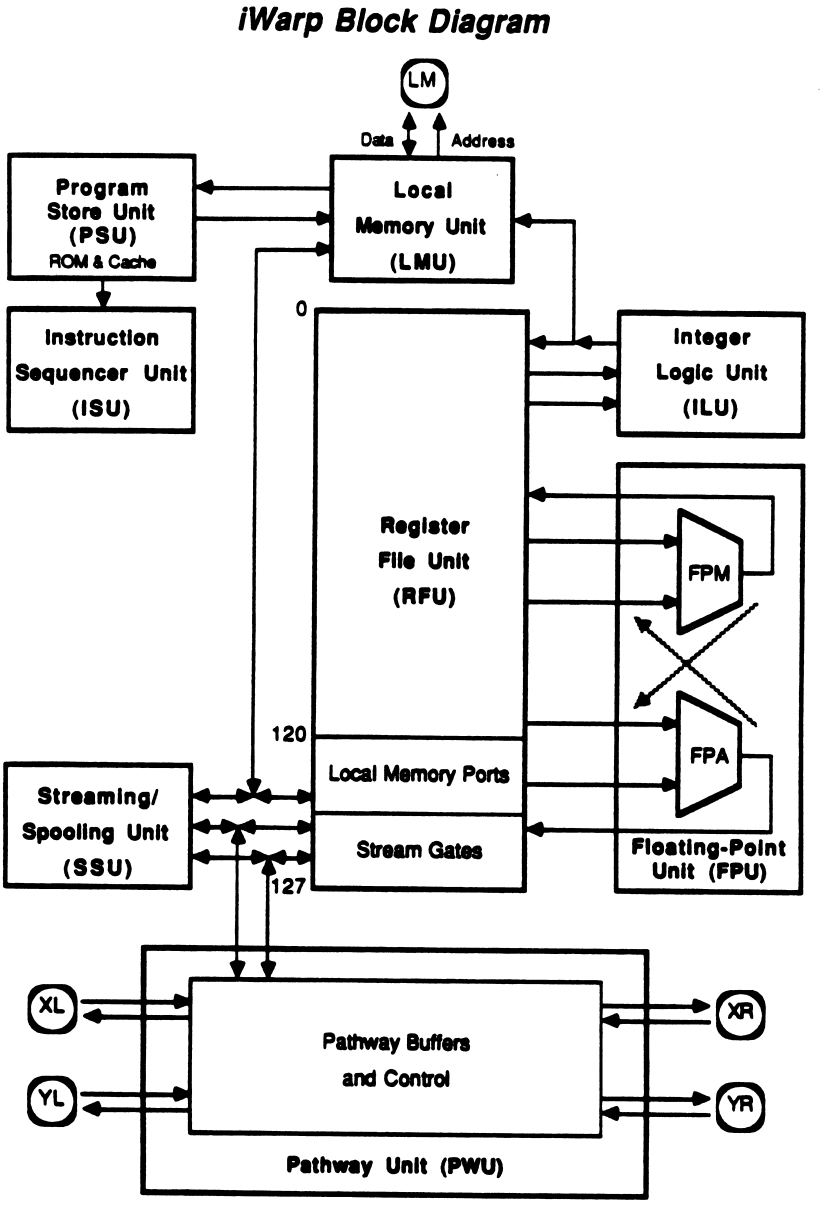
The iWarp cell architecture is designed to support fine and coarse-grained, single task (single thread of control), numeric computational model(s). These can be large-state applications (i.e., compared to I/O or controller applications which are generally small-state applications). Thus, iWarp provides the programmer with a large, flat, external local memory (LM), a large, flat, internal register space, and a large control space (used to maintain cell configuration and state information). iWarp has been designed to support high performance dedicated applications, consequently iWarp cells contain no support for large-grained multitasking (e.g., protection mechanisms).

High performance event support is included for small-state multitasking via a vectored invocation mechanism for a nested hierarchy of event handlers to perform error handling (i.e., overflow support) and function extension (i.e., IEEE denormalized number support).

Supporting multiple threads of control per cell (i.e., so called "light-weight" processes) within the single set of iWarp address space(s) per cell is probably feasible (i.e., considering the large state swap overhead), but not a primary design thrust for iWarp hardware.

Figure 2.4 shows the iWarp component architecture.

Figure 2.4: iWarp Component Architecture



3. Summary of Features of the iWarp Cell/Component

- Data Formats
  - Floating-Point Format: 32/64-bit IEEE 754 standard, draft 10.1
    - fully supported
  - Integer Format: 8/16/32-bit 2's complement including signed and unsigned mode
  - Format Conversion: integer to float, float to integer, pack and unpack
- Computational Units
  - Floating-Point Adder (FPA)
    - 10 MFLOPS peak performance on 32-bit operations
    - 5 MFLOPS peak performance on 64-bit operations
    - Nonpipelined
    - Rounding
  - Floating-Point Multiplier (FPM)
    - 10 MFLOPS peak performance on 32-bit operations
    - 5 MFLOPS peak performance on 64-bit operations
    - Full divide, remainder, and square root support
    - Nonpipelined
    - Rounding
  - Integer/Logical Unit (ILU)
    - 20 MIPS performance (up to 20 million addresses generated per second or up to 20 million integer/ordinal operations per second)
    - 8/16/32-bit integer and ordinal formats
    - Arithmetic, logical, and bit operations
  - All three units (FPA, FPM, and ILU) may be scheduled to operate in parallel in one instruction, generating a peak computing rate of 20 MFLOPS + 20 MIPS
- Memory Units
  - Local Memory
    - External
    - Data and instructions
    - separate address and data buses (24-bit address bus and 64-bit data bus)
    - up to 16 million 32-bit words (64 MBytes)
    - RAM/ROM
    - 20 million accesses per second (160 MBytes/sec.)

    * Read, Write, and Read/Modify/Write support (RMW for byte/double byte accesses and shared memory exclusion)
  - Internal Program Store
    * Internal 256 word Cache RAM
    * Internal 2K word ROM (built-in functions)
    * 32- and 96-bit instructions
    * 20 MHz performance (up to 20 million accesses per second)

- Intracell Communication
  - Via a shared, multiported, 128 word register file
  - 9 read and 6 write operations per cycle (random)
  - Special register file locations for LM and pathway interfaces
- Intercell Communication
  - Four configurable, point to point, bidirectional physical pathways: XLeft, YLeft, XRight, YRight.
  - Each pathway consists of two 8-bit unidirectional data buses and two 5-bit combined control, status and handshake buses, (one for each direction).
  - Independent Request/Acknowledge synchronization.
  - Configurable as a 1D or 2D array (2 physical pathways/dimension/1D or 1 physical pathway/dimension/2D)
  - Corner turning supported in 2D.
  - 10 Mword/sec maximum transfer rate per bus (80 Mword/sec aggregate intercellular bandwidth/bus)
- Other Facilities
  - Basic timer and counter facilities.
  - Both instruction and data breakpoint facilities
  - Maskable, hierarchical event reporting with vectored event handling.
  - ROM-resident component self test.

4. Information Flow

The following is an overview of the iWarp cell from the perspective of its interface to the outside world.

Inter-Cell

Data is transmitted between cells in the array in unit entities called messages. Messages are routed along pathways, which are comprised of links (through cells) and pathway buses (between cells). There are four physical pathway buses/cell (XLeft, XRight, YLeft and YRight) (see Figure 2.2), with independent in/out ports for each (full duplex communication is supported).

For one-dimensional (1D) linear arrays, the X/Y paths are used in parallel, as shown in Figure 4.1. For two-dimensional (2D) rectangular processor arrays, the X and Y paths are routed perpendicular to each other, as shown in Figure 4.2.

Figure 4.1: 1-Dimensional Array

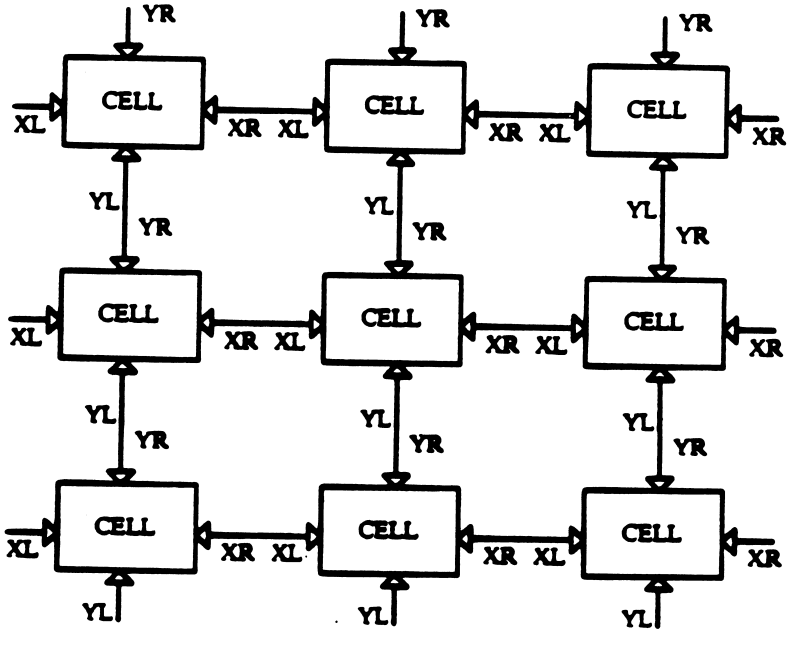


Figure 4.2: 2-Dimensional Array

Intra-Cell

In the internal block diagram (see Figure 4.3), the most important information flow features are:

- the shared, multi-ported, Register File,
- the pathway gates, and
- the LM interface.

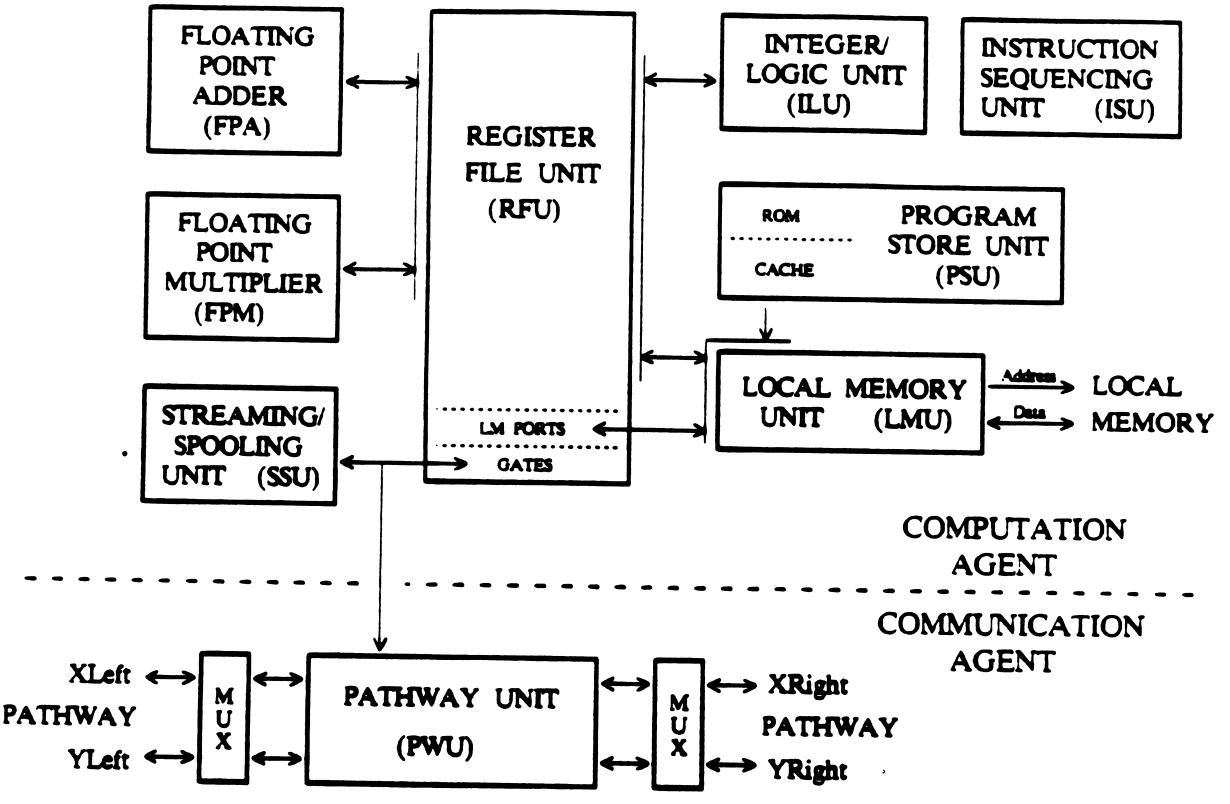


Figure 4.3: Structure of the iWarp Component

The shared, multi-ported Register File is central to routing data between the functional units in minimum time. It has nine primary ports, three each to the Integer/Logical Unit, the Floating Point Adder and the Floating Point Multiplier.

Additionally, there are three special purpose ports that provide direct connection between the external interfaces and locations in the Register File. These allow pathway or LM data to be manipulated, just as any other register.

5. iWarp Systems

iWarp systems consist of an iWarp processor array and one or more attached hosts, or external I/O ports as shown in Figure 6.1. An iWarp processor array consists of one or more iWarp cells. As depicted in Figure 2.1, an iWarp cell is implemented by an iWarp component and its local memory (LM), which involves only a few memory components. The rest of an iWarp system consists of host(s) and host interface(s), as described below (see Sections 6.1 and 6.2 for specific examples).

The system integration work required to attach/interface an array to a host is usually one of the most difficult parts of any special purpose design. Many proposals for systolic array architectures have involved configurations that are difficult to implement. The classic configuration for systolic systems is a linear array (see Figure 5.1) of one or more cells similar to the CMU Warp array. In contrast, iWarp linear arrays are connected as rings. That is, there are no "ends" and no preferred communication direction. The array is connected to a host or user port by a host interface that can connect into the array at any cell position in an iWarp array. The only constraint here is board level partitioning (e.g., multicell iWarp array boards). This allows more than one host to be connected to one array, and for a variety of I/O ports to be connected directly into the array as well. It also allows a host to have more than one host interface to a single array.

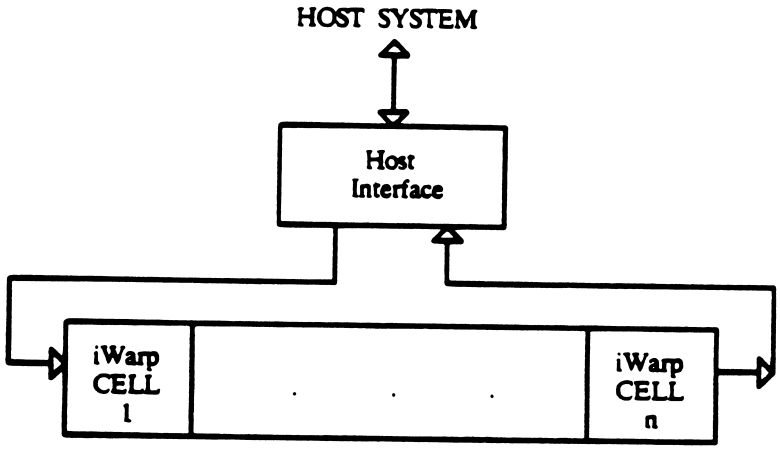


Figure 5.1: Linear Array System

The major problem in integrating an array with a host is an information flow problem. We must get information in and out of the host system, and must also get that information to the right place(s) in the array at the right time so that computations can take place.

Getting information, both control and data, in and out of the host system has three major facets: sheer volume, patterns of access, and staging/interleaving. These problems are managed by the host interface. Thus, the host interface is responsible for generating the address patterns required to read/write the data structures found in the host memory.

In terms of information volume, one of the attractive attributes of linear systolic array architectures is that, for certain classes of algorithms, they can achieve linear speedup with the addition of new processors to an array. That is, each incremental processor does not incur any incremental information bandwidth overhead beyond that required by the initial one-cell array. However, the bandwidth requirements of that one cell can be significant. To keep it fully loaded (and generate m Mflops), an iWarp cell could require as much as m Mwords/sec total throughput

(e.g., m/2 Mwords/sec in and m/2 Mwords/sec out). This would be a stringent requirement for any microcomputer host system. The positive side is that an arbitrary length array (n) has the same requirements and delivers up to n*m Mflops.

6. iWarp Demonstration System

The particular iWarp host, host interface, and array described below comprise a demonstration system configuration committed for delivery to CMU by intel as part of the iWarp contract. It should be taken as indicative of the kinds of possible iWarp system configurations.

6.1 Demonstration System Host

The iWarp demonstration configuration that intel will build is shown in Figure 6.1, and is configured to support a 74-cell array with a separate file server and UNIX host. The iWarp array will be housed in a single 34-slot container that is four feet high by sixteen inches square. As an example of the configurability provided by these systems, each container can contain up to 126 iWarp cells, and up to 144 containers can be configured into a single array. Additionally, interface connections can be inserted at any slot in the array, each supporting I/O data rates of 40 MBytes/sec. or more. (For the 74-cell array, eleven slots are unused and available for configuration options.)

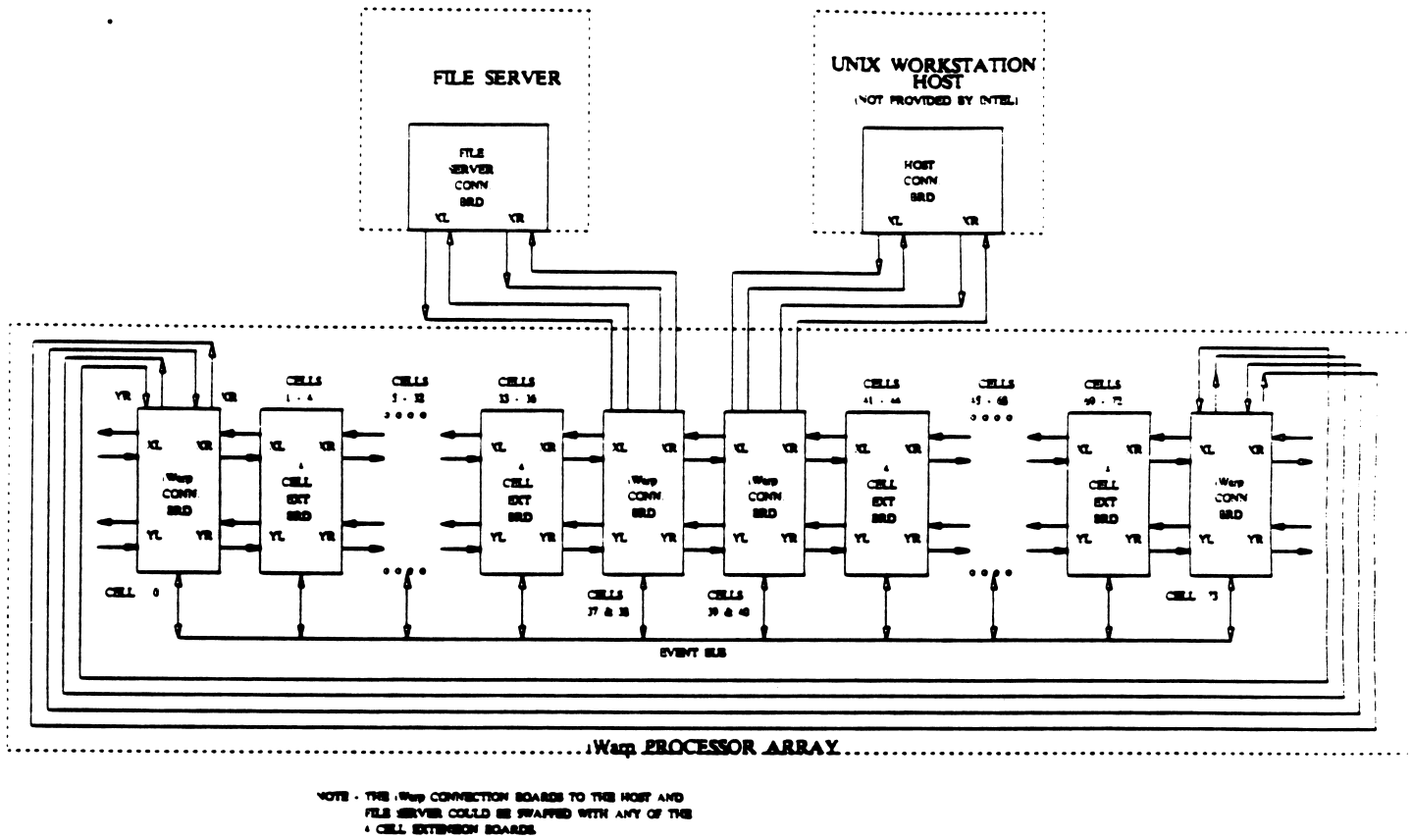


Figure 6.1: CMU iWarp Demonstration System

6.2 Demonstration System Host Interface

For the iWarp demonstration system host interface, a new board type (called a 'host connection' board) will be developed (see Figure 6.3). A host connection board provides an interboard, host system bus connection and an interchassis connection to a pair of iWarp cells (i.e., one to the left and one to the right) in the array. One host connection board comprises a host interface for an array.

As with Warp, host interface serves to drive data into an iWarp processor array and remove data from that array. Thus, host interfaces can generate addresses for streaming data in and out of host memory.

An iWarp host interface is composed of two active processing agents: an I/O processor and an iWarp processor (see Figure 6.2). The I/O processor is internally multi-tasked, and generally programmable. This programmability will add considerable flexibility to our design.

As shown in the figure, these processors each have their own private local memory (i.e., for local data and code). They communicate with each other via an asynchronous, shared, multiported memory and an event/interrupt signaling mechanism. By making the interprocessor interface asynchronous, we allow host(s) and the array to run at their own (potentially differing) natural clock rates. Both processors support Read/Modify/Write access to this shared memory. This allows software executing on them to synchronize safely over multiple, lockable, shared data structures in this shared memory. Software, running on these processors, implements a shared, buffered, message queue-based, communication protocol for array input/output.

On its other side (i.e., the side facing away from its interface to the iWarp processor), the I/O processor is capable of generating a variety of standard microprocessor backplane buses (e.g., Multibus II or VME) with very little external logic.

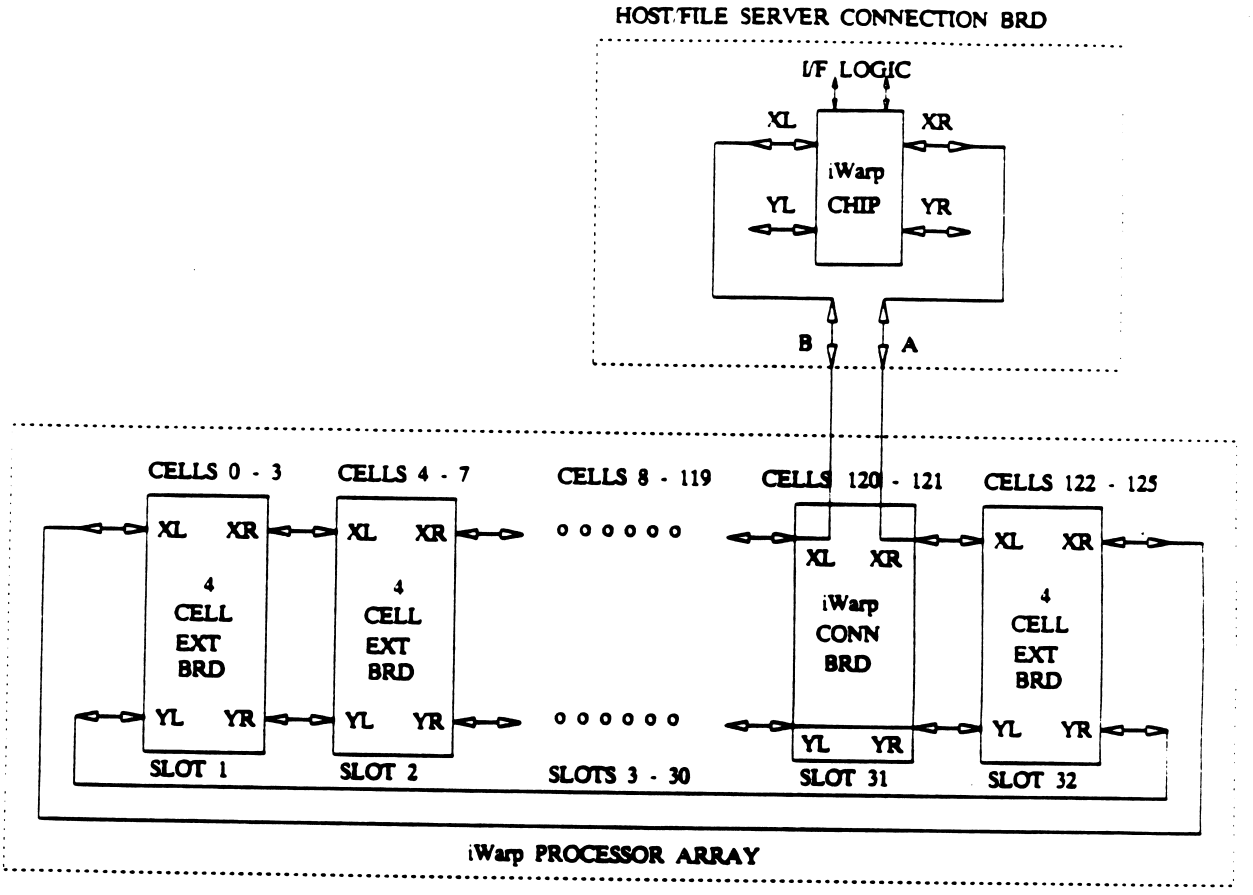


Figure 6.2: Single iWarp Linear Ring with Maximum Number of Cells, and One External Interface Connection

6.3 Demonstration System Array

For the demonstration system iWarp processor array, four new board types will be developed: a 'backplane' board, an 'array connection' board, an 'extension' board, and a 'clock' board (see Figure 6.3). The backplane board provides interboard connections between extension boards within a single chassis. The array connection board provides array extension via two iWarp cells, two interboard iWarp connections, and interchassis connections (e.g., to a host connection board in a host). The extension board provides array extension via four iWarp cells and two interboard iWarp connections. Thus, any iWarp array employs at least one host connection board (as host interface), at least one array connection board, at least one backplane board, some number of extension boards, and one clock board.

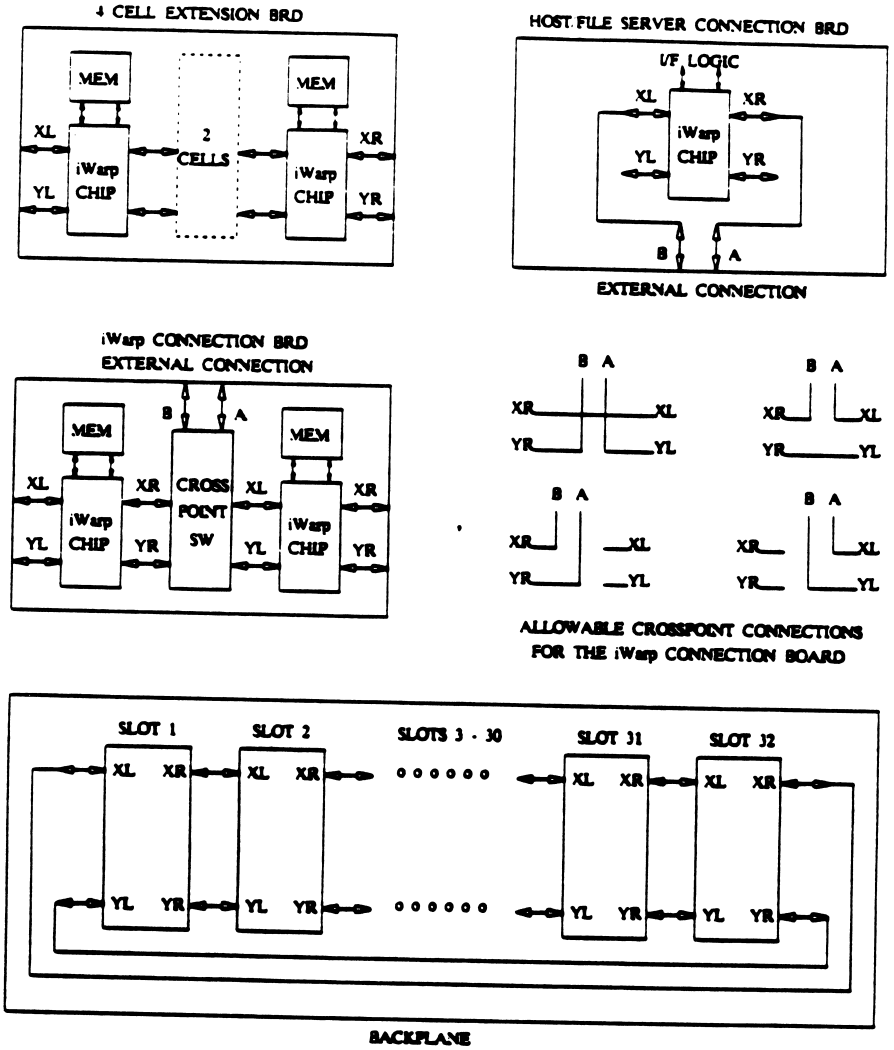


Figure 6.3: iWarp Array Building Blocks

While a wide range of local memory configurations are possible, for this discussion, assume that a basic iWarp demonstration system cell contains 128K words of local memory. With such a cell configuration, the iWarp component and the local memory have comparable footprints (about 2" x 2" and 2" x 3" respectively). Thus, a basic iWarp demonstration system cell has roughly a 2" x 5" footprint.

The iWarp demonstration systems will contain 17 4-cell extension boards, 2 1-cell host connection boards, 4 2-cell array connection boards for a total of 74 cells in the array. The backplane, array connection, clock, and extension boards will be housed in a single intel iPSC/2 chassis. Based upon a nominal array clock rate of 20 Mhz, this configuration will generate a peak single precision computing rate for the demonstration systems of 1.48 GFLOPS (e.g. 74 cells x 20MFLOPS/cell) and 1.48 GIPS (e.g. 74 cells x 20MIPS/cell).