A Fine-Grain, High-Throughput Architecture Using Through-Wafer Optical Interconnect

W. Stephen Lacy, Christophe Camperi-Ginestet, Brent Buchanan,
D. Scott Wills, Nan Marie Jokerst and Martin Brooke
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250

Abstract

This paper presents a highly parallel, three dimensionally interconnected system to process high-throughput stream data such as images. Optical interconnect at wavelengths to which silicon is transparent is used to create the 3D system. Thin film InP/InGaAsP-based emitters and detectors operating at 1.3 microns are bonded to the silicon circuitry, and emit through the silicon wafer to create the vertical optical interconnect. Foundry-fabricated Si circuits are post processed using standard, low cost, high yield microfabrication techniques to integrate the thin film devices with the circuits. In order to meet off-chip I/O requirements, a high-bandwidth, three dimensional optical network is also being designed. Using through-wafer optical interconnect, a new offset cube *topology has been created, and naming and routing schemes have been developed. Its performance is comparable to that of a three dimensional mesh. A processing architecture has also been defined that minimizes overhead for basic parallel operations. A complete processing node for high-throughput, low-memory applications can be implemented using a fraction of a chip.*

1 High-Bandwidth Parallelism

Many applications can benefit from large-scale parallelism. But their system requirements vary with a large number of factors, including I/O, storage requirements, and the type of operations required (i.e., integer versus floating point). High-throughput, low-memory problems form one class of applications. This class requires simple parallel tasks to be performed on massive streams of data with less local memory required since operands are provided primarily as I/O. Examples of this application class include image processing (e.g., filtering, edge detection, and convolution), object recognition, and data compression.

This type of problem does not map well onto existing general-purpose parallel computers which are designed for low-throughput, high-memory operation (i.e., a large data set is loaded into memory and extensively processed). Although the I/O bandwidth of these machines is often high, they are inefficient for high-throughput applications. Recent research has explored single-chip processing nodes [1][9][14] employing architectures which target more general applications and which use external dense memory chips. But because their access is sequential, dense memory arrays are a poorly utilized silicon resource. Since they are not required for high-throughput applications, the silicon area can be used for additional processing, or eliminated for lowered system cost.

This paper presents a high-throughput processing system which incorporates a new high-bandwidth integrated optoelectronic technology. The use of this technology changes the balance of communication, storage, and processing in the system allowing new approaches to processor and network design. A fine-grain message-passing processing architecture is being designed for handling high message traffic with minimum parallel overhead. Efficient support for task and storage management, communication, and synchronization is provided. Single cycle task swapping also reduces overhead for short tasks. High-throughput applications and a low-memory processing node make I/O a critical aspect of this architecture. Experiments with example programs indicate that each executed instruction requires approximately .1 - .5 words of I/O. If a processor executes at 50 MIPS, the I/O rate can be as high as 25 Mwords/sec. This requires 800 Mbits/sec of I/O. With dense, three dimensional arrays of multi-node chips, conventional interconnect technology is inadequate.

To satisfy the high I/O requirement, an integrated three dimensional optical network is being developed, incorporating integrated optical devices developed at Georgia Tech. Arrays of thin film InGaAsP/InP optical emitters and detectors are bonded to silicon chips which are then staggered to overlap chips in different planes. This forms an *offset cube* topology similar to a three dimensional mesh.

This paper describes each aspect of the system design. It begins with a description of through-wafer interconnect and continues with the design of a high-speed analog interface. The physical architecture built using these enabling technologies is then described and the Pica architecture is introduced.

2 3D Optical Interconnect

The manufacture of hardware capable of scalable three dimensional local connectivity has been a challenge for decades. The high interconnect density avail-

0-8186-5830-4/94 $3.00

able on VLSI circuits is inherently two dimensional in nature and a new technology is needed for massively parallel interconnection in the third dimension. We have demonstrated that through-wafer optical interconnect can be employed to pass information from one layer of silicon to another. This optical interconnect scheme uses thin film InGaAsP/InP emitters and detectors which are bonded directly to the silicon. These emitters and detectors operate at a wavelength of 1.3 microns, to which silicon is transparent. The use of thin film emitters and detectors bonded to silicon enables the use of the integration complexity and density of silicon coupled with reliable, low cost fabrication technology for both the circuitry and the integration: standard foundry silicon circuitry and standard microfabrication techniques for post-processing of the thin film devices are used to integrate the system.

The thin film detectors and emitters are deposited onto the silicon using epitaxial liftoff processing, thereby effectively separating the manufacture of the silicon circuitry from the manufacture of the InP-based optoelectronic devices used for vertical interconnect. Utilizing the process of thin film epitaxial lift off, high quality, single crystal epitaxial InP/InGaAsP materials and devices, either individually or arrays, are separated from the lattice matched growth substrate and can be deposited onto smooth host substrates such as Si. The compound semiconductor devices can be grown, separated from the growth substrate, aligned and selectively deposited as arrays or single devices out of an array onto host substrates. This type of mixed-material system is extremely attractive for high performance three dimensional systems using through-wafer optical interconnect; unfortunately, indium phosphide-based compounds have not been successfully directly grown on silicon substrates. Using epitaxial lift off (ELO), we have monolithically integrated high quality, single crystal thin film compound semiconductor InGaAsP/InP-based devices onto host substrates which include silicon circuitry, polyimide, glass, and lithium niobate.

Epitaxial lift-off is a promising technique for the integration of compound semiconductors onto smooth substrates since it involves no special growth techniques, has produced GaAs and InP materials whose high quality is unchanged by the lift-off process, and is a relatively simple process with potential for scale up. The process of epitaxial lift-off, which has currently been demonstrated in the GaAs and InP-based materials system [2], utilizes etch selectivity as a function of material composition. In the InP/InGaAsP material system, the etch selectivity between InP and InGaAsP is utilized to form a series of stop-etch layers which are used to separate the thin film epitaxial devices of interest from the growth substrate. These layers generally range from 0.1 to 5 microns thick. After the epitaxial layers have been separated from the growth substrate, they are commonly handled manually using a thick (100 micron) wax handling layer.

We have developed a polyimide transfer diaphragm that allows the alignment, back contact, and selective deposition of individual devices or arrays of devices [4]. After the epitaxial lift off of mesa etched and surface processed thin film devices, they are attached to a transparent transfer diaphragm. This diaphragm is constructed of either polyimide or mylar, and is supported by a ring of Si on the outer edge. The diaphragm is then inverted (with thin film devices facing the host substrate), and the thin film devices are visually aligned and using a pressure probe, bonded to the host substrate. Current deposition accuracy is to within 1 micron with respect to features on the host substrate.

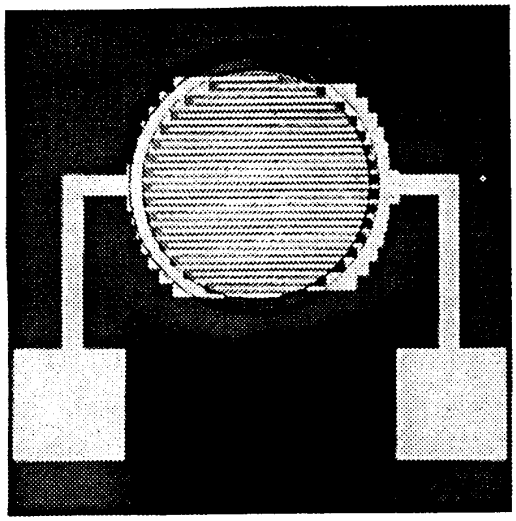

Figure 1: Thin film InGaAsP MSM detector.

Thin film InP/InGaAsP/InP pin and InGaAsP MSM detectors which detect at 1.3 microns and pn homojunction InGaAsP light emitting diodes which emit at 1.3 microns have been fabricated at Georgia Tech. We have also demonstrated through silicon wafer optical communication using these thin film emitters and detectors [3]. These through-wafer demonstrations include a front to back of silicon wafer demonstration as well as a wafer to wafer demonstration. The performance of these thin film devices is comparable to that of devices which have not been separated from the growth substrate. A number of studies have demonstrated that the quality of these thin film materials and devices is not affected by the process of epitaxial lift off and bonding to the host substrate [22]. Figure 1 is a photomicrograph of an InGaAsP thin film detector which detects at a wavelength of 1.3 microns. As an example of a thin film array, Figure 2 is a photomicrograph of a 4 X 4 array of thin film p-i-n detectors integrated on top of silicon neural network oscillator circuitry.

3 Analog Circuits

In the Pica architecture described herein, the operand requirement for instruction execution at each node requires 100 Mbps per channel optical through-wafer communication nodes. To communicate at the approximately 100 Mbps necessary for each through wafer optical connection, we have designed a detec-

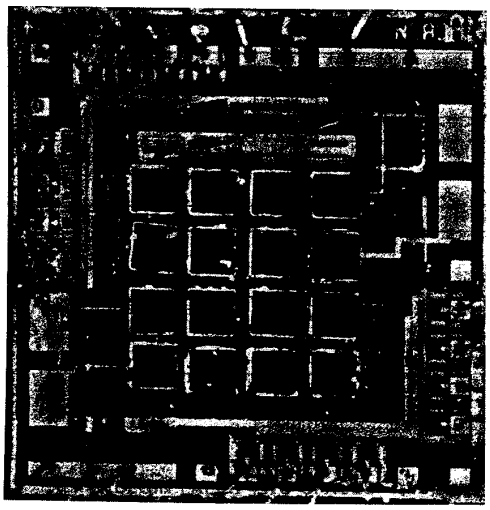

Figure 2: Photomicrograph of a 4 X 4 array of thin film detectors integrated onto silicon neural network oscillator circuits.

tor amplifier, laser driver, multiplexed/demultiplexer, and timing circuitry with nearly 100 MHz bandwidth (assuming non return to zero (NRZ) encoded serial data transmission) using 0.8 mm CMOS technology. The basic channel architecture proposed is similar to most 100Mbps digital fiber optic communication channels with the following notable exceptions. We assume that, due to the tightly controlled environment of the wafer to wafer communication channel, that low Bit Error Rates (BER) will be achieved without the use of error correction codes. We also assume that a global clock signal is available and that only a steady skew will perturb it, i.e., there will be very low clock jitter. The maintenance of low BER requires that signal to noise margins be kept at close to normal CMOS bus levels throughout the channel design procedure. This should be possible with careful detector amplifier design. BER is also impacted by the quality of the clock used to recover the detected digital signals. The presence of only a constant clock skew and no significant jitter will help to maintain low BER since the skew can be canceled by a calibration procedure.

To design the detector amplifiers, a number of assumptions were used to calculate the amplifier parameters. In the through wafer interconnect scheme demonstrated, a 1% optical through wafer efficiency was calculated and indirectly measured. Several factors, such as lensed emitters, use of vertical cavity surface emitting lasers, and use of standard thickness wafers, will improve this figure, but as a worst case we can use it for determining the specifications of the detector amplifier required. Further assumptions of a 50% electrical to optical power conversion in a semiconductor laser emitter, a 5mW laser power consumption and a 0.5 amp per watt detector efficiency yield a peak detector current of 12.5 mA. Thus to produce a 5 volt signal swing a detector amplifier must have a gain of 400 kW. Using the simulated performance of the Hewlett Packard 0.8 mm CMOS foundry process available thought the MOSIS service, a 3 stage open loop amplifier with a gain of 7.4 per stage and total power dissipation of 5mW will easily meet the gain specifications with a simulated bandwidth of 282 MHz, about 3 times that needed for a 100Mbps NRZ encoded serial signal. Similar amplifiers fabricated by the authors have had equivalent input noise currents of 10 to 100 nA. This gives a signal to noise ratio of 100 to 1000 or 40 to 60 dB which is superior to the noise margin typical for a CMOS data bus.

The primary concern with clock distribution is the calibration of the skewed clocks. We can assume that each chip in the processor array will have a slightly different but essentially constant clock skew. Adjacent processors will have very similar skews, and careful design of the clock distribution network will ensure that the skew between locally connected processors on any one processing plane will be negligible. Thus each processor must correct the skew in the clock from processors on the plane above or below it in a stack, but will need only one correction for all four connected processors on the same plane. One high end solution to this incorporates a single analog phased locked loop onto each chip in the array, and corrects the skew with a 4 or 5 bit digital to analog converter (DAC) to set the skew. This scheme assumes a short synchronization period at power-up, and a supply voltage stable to one least significant bit (LSB) for the DAC.

None of this proposed scheme presents a technological challenge, and simplified schemes may well suffice. The scheme will operate unless the skew between adjacent chips becomes significant; at that point one more correction phase locked loops per channel will be needed. However, that problem will not occur until clock speeds show significant skew in less than a centimeter, which would require GHz clocks.

The multiplexing of 100Mbps NRZ encoded data back to the 50 MHz chip clock data is simplified by using the same clock speed for the MUX/DEMUX as the rest of the processor. This clock speed is about one half of the 100 MHz 31 stage ring oscillator frequency of the HP 0.8 um CMOS process.

The design of laser drivers is similar to digital pad drivers. We propose to add a variable current ECL like differential output stage to allow a variable laser drive current. This will allow the power dissipation of the lasers to be controlled and adjusted to minimize heat dissipation. It will require a special low voltage high current DC power bus of around 2 volts for the lasers.

4 Network Architecture

In order to meet the I/O requirements of several nodes per chip, a high-bandwidth, three dimensional network is required. Traditional wire interconnects which are limited to perimeter chip contacts cannot scale as transistor densities increase. New MCM technologies (e.g., bump bonding) are not suitable for scalable three dimensional networks. Existing optical interconnect techniques such as index-guided and free-space networks are difficult to incorporate in three

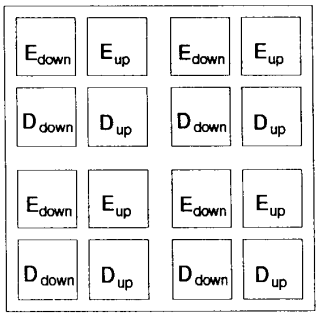

Figure 3: Chip Device Pattern

dimensional interconnection networks. We are currently developing a three dimensional optoelectrical network using hybrid integrated optoelectronic devices and through-wafer transmissions.

4.1 Background

To build a system, processing elements are designed and fabricated using existing Si VLSI techniques. No bonding pads are included except for power contacts. (Probe pads are provided for testing.) Instead, holes in the overglass allow GaAs and InP-based emitters or detectors to be attached in a post-processing step. Driver circuits, included on the silicon, interface parallel buses from routers on the chip to the faster optical devices (0.1 - 1.0 Gbits/second).

After the Si chips containing the optical devices are fabricated and tested, they are attached to a larger silicon substrate. This provides physical support and also participates in power distribution and cooling. A fully populated substrate forms one plane of the system. Completed substrates are then stacked to create the third dimension. To facilitate manufacturing, all chips and substrates are identical.

Each chip is broken into four quadrants which overlap with eight neighboring chips (four in the plane above, four in the plane below). Although any number of emitter and detectors can be included in each quadrant, the minimum case requires two emitter/detector pairs, shown in Figure 3. In order for emitter/detector pairs to be correctly aligned, the chips must be offset. To achieve this using identical plane substrates, a spacing equal to one half the chip width plus interchip spacing is added to two non-opposing sides of the substrate. Then alternating planes are rotated 180 degrees during assembly providing the correct device alignment.

System I/O is provided by I/O layers which inject data directly into the network on the top and bottom surfaces of the stack. These layers can interface to high-bandwidth electrical or optical connections. Alternatively, the input surface can be covered by visible light detectors for direct focal-plane imaging. The output surface can incorporate a frame buffer for direct video output. The sides of the stack are used for power distribution and cooling connections. Due to the dense packaging, liquid cooling between layers is anticipated.

4.2 Offset-Cube Topology

Since each chip in the system contains several nodes, the communication network is divided into two parts. The inter-chip network delivers messages between the chips containing the source and destination nodes. The intra-chip network is used to communicate messages within the source and destination chips. Messages sent between nodes within a chip are routed entirely via the intra-chip network.

Communication within a chip is accomplished using a wire-only network. This type of network is well-studied [8] [16]. Since on-chip wire density is high and the number of nodes per chip is initially low, the topology of the intra-chip network is straightforward.

Each chip connects to eight neighboring chips: four in the plane above and four in the plane below. There are no inter-chip connections within a substrate plane. This interconnection forms an offset cube topology which bears similarities to the well-studied k-ary 3-cube. While the physical architecture of the inter-chip network is not symmetric in the vertical and horizontal dimensions, the offset cube topology implemented by this network is isotropic. The topology, shown in Figure 4, is formed by vertices that have all even or all odd coordinates.

The topology is formally defined as follows. Consider a set of k^2L vertices (representing processing elements) where k is the length of each side of a square vertex array on each layer and L is the number of layers. The vertices are named by triples $\langle x, y, l \rangle$ where l is the layer number between 0 and $(L-1)$ inclusive. For even values of l, $x = 2i$ and $y = 2j$ where i and j are the vertex's coordinates within a layer assuming values between 0 and $k-1$. For odd values of l, $x = 2i+1$ and $y = 2j+1$ for i and j between 0 and $(L-1)$ inclusive. The offset cube topology is defined by the edges connecting each vertex $\langle x, y, l \rangle$ with its neighbors $\langle x \pm 1, y \pm 1, l \pm 1 \rangle$.

Boundary vertices have less than eight edges, and are defined in six overlapping sets.

1. vertices $\langle 0, y, l \rangle$ are connected with its neighbors $\langle 1, y \pm 1, l \pm 1 \rangle$
2. vertices $\langle 2k-1, y, l \rangle$ are connected with its neighbors $\langle 2k-2, y \pm 1, l \pm 1 \rangle$
3. vertices $\langle x, 0, l \rangle$ are connected with its neighbors $\langle x \pm 1, 1, l \pm 1 \rangle$
4. vertices $\langle x, 2k-1, l \rangle$ are connected with its neighbors $\langle x \pm 1, 2k-2, l \pm 1 \rangle$
5. vertices $\langle x, y, 0 \rangle$ are connected with its neighbors $\langle x \pm 1, y \pm 1, 1 \rangle$
6. vertices $\langle x, y, L \rangle$ are connected with its neighbors $\langle x \pm 1, y \pm 1, L-1 \rangle$

For a symmetric cube, $L = 2k-1$ creating a *k-ary offset cube*. The offset cube topology is isotropic, and is illustrated in Figure 4.

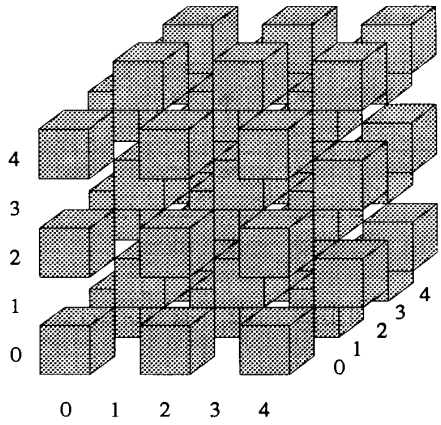

Figure 4: A 3-ary Offset Cube Topology

In a k-ary 3-cube, a shortest path routing algorithm uses channel routes to reduce the difference in coordinates of the source and destination vertices. A well-performing deterministic shortest path algorithm reduces the coordinate differences one at a time (i.e., x, then y, then z, or *dimension order routing*).

A shortest path routing algorithm for an offset cube is more complicated since each channel hop changes all three coordinates. If one coordinate difference reaches zero, it must be increased in the next hop to reduce differences in other dimensions. This creates "bouncing" routes where a message alternates between neighboring planes as it moves through one dimension. Several routing paths are shown in Figure 5.

The minimum path cost of routing a message between vertices A and B on an offset cube is:

$$hops_{min} = \max(|X_B - X_A|, |Y_B - Y_A|, |Z_B - Z_A|)$$

Deadlock avoidance in an offset-cube is similar to that in k-ary 3-cubes. Existing techniques [10] are being adapted to this network.

4.3 Network Performance

The analysis of this network is similar to that of wire-only networks [8], except that the optical channels require a different set of assumptions.

1. Unlike the wire density limits for wire networks, optical channels are limited entirely by characteristics of the transmitter and receiver. The communication medium (i.e., the space through which light travels) does not enforce any fundamental density or bandwidth limits.

2. The channel delay is not determined by channel length. Since the messages travels fast (near c), and the channel length is small (around 1 mm), the time of flight in a channel is around three picoseconds. The channel delay is dominated by transmitter and receiver response time (around 1 nS).

3. Power and heat dissipation are the dominant issues in communication costs. GaAs LEDs have a low quantum efficiency (<3%). GaAs lasers are are more efficient (75%), but still dissipate several milliwatts during operation. The cumulative effect of thousands of lasers operating in a few hundred cm^3 will require special consideration.

Performance of the offset cube topology has been analyzed through simulation. This section presents a summary of the results of this analysis. The complete details are available in [12].

To compare the k-ary 3-cube and offset cube topologies, a simulator has been constructed which analyzes network performance under similar conditions and loads. In these experiments, a 16-ary 3-cube is compared with a 13-ary offset cube. A random traffic load is simulated. Wormhole routing is employed with eight virtual channels for each physical channel. All message lengths are 25 flits.

Figure 6 compares the performance of deterministic and adaptive routing algorithms on the two topologies. For the deterministic case, the 16-ary 3-cube uses dimension-order routing. The offset cube routes along a diagonal shortest path. The offset cube utilizes less than half of the capacity of the 3-cube.

The capacity loss is due to loading in the center of the network. For random traffic, deterministic routing on a 3-cube does a reasonable job of distributing traffic across the entire network. The offset cube's diagonal routing creates much higher traffic in the center of the network. This has been verified by comparing cross-sectional loading profiles for a 3-cube with dimension-order and diagonal routing algorithms.

To better utilize the capacity of an offset cube network, a simple adaptive routing algorithm is employed. Local information (the number of virtual channels in use) is used to select between shortest paths at each vertex. The adaptive offset cube performance exceeds that of the best 3-cube strategy. The adaptive 3-cube performance is *lower* than the deterministic case because of increased diagonal routing.

5 Processor Architecture

The Pica execution architecture is designed for handling high message traffic consisting of small, ephemeral tasks. In order to achieve acceptable efficiency in this fine-grain domain, parallel overhead must be reduced to the minimum achievable level. Complex mechanisms to support general purpose applications are replaced by simpler, lower cost mechanisms for high-throughput problems.

The Pica execution architecture is designed specifically for high-throughput, low-memory operation. The design of a Pica node begins with a minimal sequential core architecture. Pica provides low overhead support for communication, synchronization, naming, and task and storage management. A small amount of memory (4096 36-bit words) and a network interface/router complete the node. This node complexity can be implemented using a fraction of the transistors available on a chip in current technology. This allows *multi-node chips*; the prototype chip will contain four nodes.

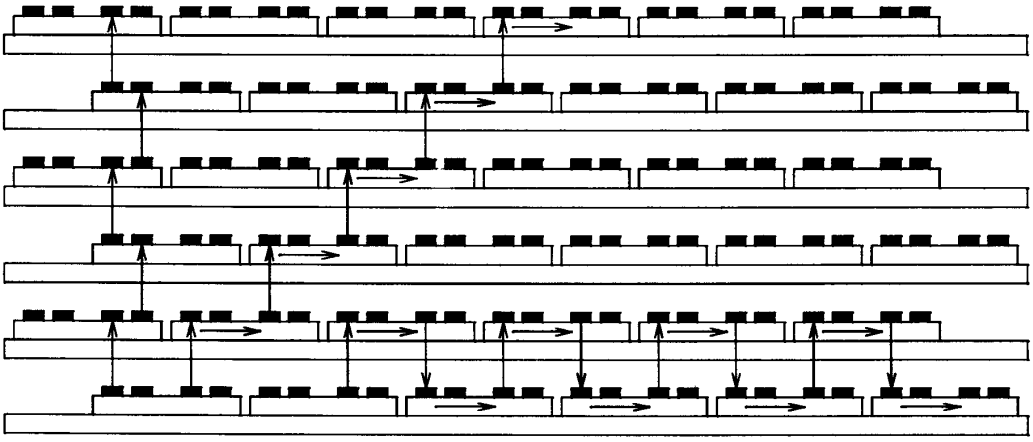

Figure 5: Routing Paths

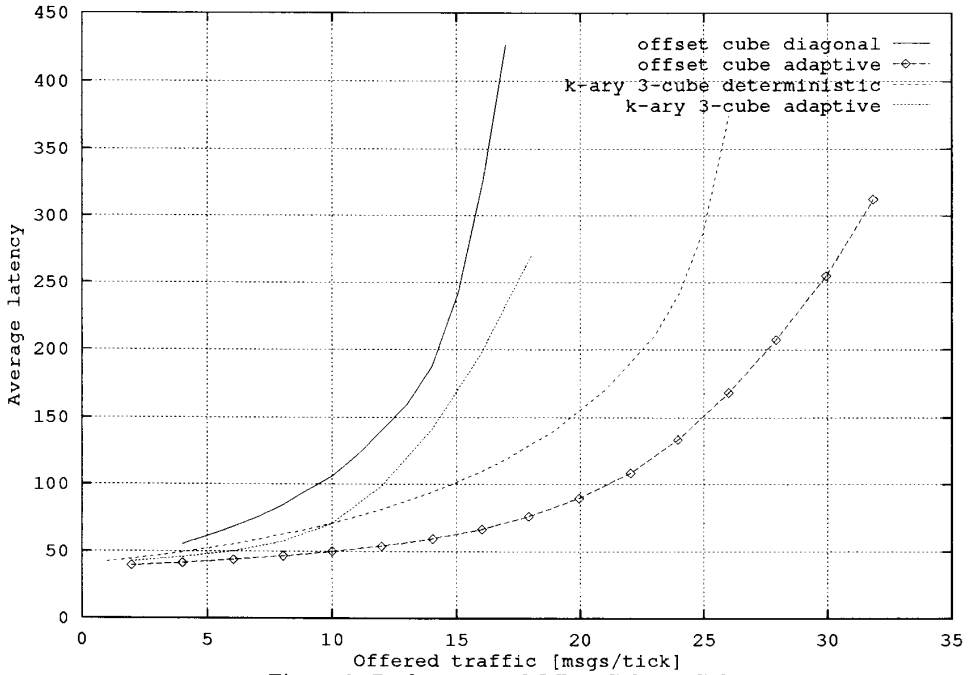

Figure 6: Performance of Offset Cube, 3-Cube

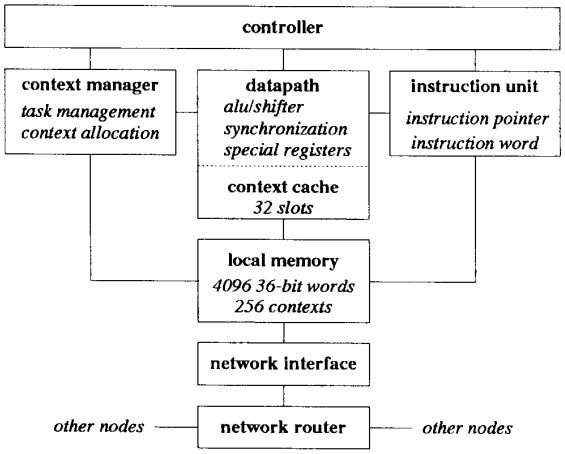

Figure 7: Pica Microarchitecture

The Pica architecture is designed to form a dense, three dimensional computational array for processing high-throughput data streams. While less general than other MIMD architectures, it is more efficient for this application area. The execution model supported by Pica is more flexible than other high-throughput architectures (e.g., systolic arrays, static dataflow).

The basic functional blocks of the Pica microarchitecture are shown in Figure 7. The network router routes messages through the node, forming that node's contribution to the communication network. The router implements a simple adaptive routing strategy based on current local virtual-channel allocation. The network interface buffers incoming messages and signals the context manager that a context is required. When it obtains access to local memory, the network interface writes the message contents directly into the allocated, fixed-length context. The datapath consists of a 32-bit integer ALU and shifter, and special-purpose registers. Operands are accessed from a 32 word context cache, which supports two read and one write accesses on each cycle. The instruction unit fetches and decodes instructions for execution. In order to keep design complexity and task swapping overhead low, the datapath implementation is not pipelined. The context manager serves three functions: (1) it maintains a queue of suspended and ready tasks for execution, (2) it allocates task storage for incoming messages and deallocates storage as the tasks complete, and (3) it arbitrates requests by both the network interface and cache controller for control of the local memory bus.

5.1 Storage Management

To support fast storage allocation and deallocation, node memory is divided into fixed-sized, 16 word blocks, or *contexts*. Contexts are used to store task data, persistent data objects, and program code. The size of a context is based on results from [19]. For a set of fine-grain benchmark programs, 95.8% of the tasks require less than eight words of storage. 99.0% require less than 16 words. Objects larger than 16 words are

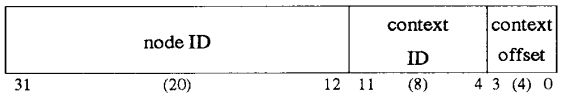

Figure 8: Global Address Format

constructed using collections of contexts with some increase in access cost.

Fixed sized contexts reduce storage management costs, but result in slack (unused space at the end of a context). This penalty is attenuated by the elimination of memory fragmentation. Also, small objects tend to be ephemeral. The median lifetime of task storage for the examples tasks is 100 operations. The primary benefit of fixed sized contexts is low-overhead context allocation and deallocation, which is accomplished by hardware.

In the initial Pica implementation, local memory consists of 4096 36-bit words. It is organized as 256 contexts, each containing 16 words. This storage is implemented with dense cell memories, and is organized in 256 rows of 16 words (576 bits) in length. As a row is accessed, it is stored in a row buffer to take advantage of spatial locality in accesses. The local memories of the nodes form a global address space. Figure 8 shows the address fields. The maximum address space is four billion words with one million nodes.

5.2 Task Management

Pica is a message-driven processor. When a message arrives at a node, a context is allocated to store the message and to provide local storage for th associated task. Context allocation for message arrival occurs without processor intervention so the current task is not interrupted. The context associated with the currently executing task (the *active context*) is indicated by the active context register in the datapath.

Messages contain no more than 16 words (excluding the header) so they can be stored in a single context. When a messages arrives, the context in which it is stored becomes the task context for the procedure that handles the messages (the message handler). Communication-only messages, which require no handling task, are also supported. Tasks requiring additional automatic or persistent storage can allocate additional contexts. Handler code is installed on the appropriate nodes during system configuration. Handler lengths are not fixed. Several handlers can be stored in a single context, or one handler can spread across several contexts. With only 4096 words of storage per node, long sequential procedures are prohibited.

Task execution on the Pica is non-preemptive. Once a task begins execution, it continues until (a) it completes, (b) it encounters a unready synchronization, or (c) it voluntarily suspends using the SUSPEND instruction. Task are restarted in a round-robin fashion based on information in the context management table. This table contains one entry for each context in local memory. The entry fields include an allocation bit, a task bit, and an instruction pointer (IP). The allocation bit indicates whether the corresponding context is currently in use. The task bit indicates whether the context has an associated task. If so, the current instruction pointer is stored in the IP field.

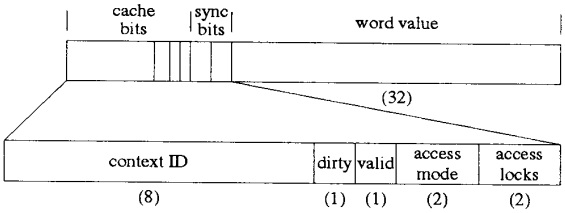

Figure 9: Context Cache Fields

Only entries with the task bit set are considered for execution.

When a messages arrives, the header word contains a handler ID which is the starting instruction pointer for the handler code on the node. This address is stored in the IP field of the context management table entry for the allocated context. When a task suspends, the IP is written back to the context management table, and the next task in the table is executed. In this scheme, a task blocked by a synchronization will poll when its turn arrives. While some extra work is performed over a non-polling scheme, this simpler approach reduces the overhead of task swapping and synchronization. Task swapping occurs in a single machine cycle.

5.3 Context Cache

Instruction operands are either 16-bit sign-extended immediate values or five bit slot operands. This 32 directly addressable slots include the sixteen slots of the active context and the sixteen slots of a specified data context associated with the task. The slot operand fields are concatenated with either the active context register or the data context register to form an address in local memory. There are no explicitly loaded registers. To avoid the latency associated with accessing dense cell memories, a context cache is included between local memory and the datapath.

The context cache assumes the role of registers in a traditional load-store architecture, providing fast, multi-ported access of variables. This is similar to the named-state storage described in [15]. Unlike traditional registers, context cache entries are managed automatically at runtime based on temporal access locality. In the current implementation, the context cache is organized as a direct-mapped cache with 32 sets and one word lines (N = 32, K = 1, L = 1). The cache employs a write-back update policy. The context cache provides low-overhead task swapping, since entries are not swapped back to local memory unless the slot is required by the active task (i.e., lazy context swapping).

Figure 9 shows the fields of a context cache entry. The context ID is the cache tag. Since Pica has a three operand instruction set architecture, three simultaneous accesses of the cache occur during each processor cycle. These comparisons occur in parallel with the datapath operation, (but before the destination write occurs). In the case of a context cache miss, the instruction is aborted and the missing lines are loaded from memory.

Simulations indicate that context cache miss frequencies are comparable to register load/store frequencies in a traditional register-based architecture. However register swapping due to task switches are substantially reduced.

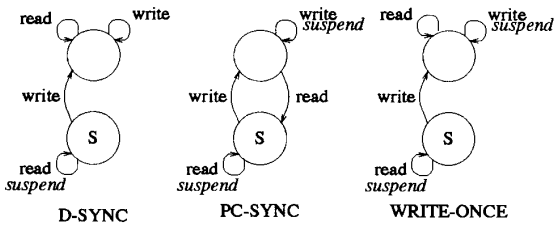

Figure 10: Synchronization State Tables

5.4 Synchronization

Synchronization in a parallel system is composed of two components: a piece of named synchronization state, and an inter-node communication mechanism to allow access of that state. In Pica, communication is provided via message sends. The named state, which indicates the status of synchronization events (both data and control), is maintained as annotated storage locations. In this way, synchronization event naming is supported via the existing storage naming scheme already described.

Each word in local memory includes a 32-bit data value and a four-bit synchronization tag. Two bits of the tag are read lock and write lock. If a lock bit is set, the corresponding access to the word is blocked, resulting in the suspension of the active task. The two remaining bits define the access mode for the word. The access mode defines how the lock bits are to be modified during an access. For example, a data synchronization (e.g., full/empty bits in HEP [17]) is accomplished by initially setting the read lock bit of a word. The mode is defined as D-SYNC, indicating that a write operation should clear the read-lock bit. Figure 10 illustrates the three access modes.

A fourth access mode, READ-WRITE does not change lock bits during accesses. In this mode, the lock bits are directly manipulated using the READ-TAG and WRITE-TAG instructions to efficiently create other synchronization types (e.g., barrier synchronizations).

To provide low-cost synchronization support, the testing and manipulation of the sync tag is performed by hardware in the context cache. When a three operand instruction executes, the synchronization tags of the three operands are accessed and modified in accordance with the access modes. Three state machines (two read access and one write access) determine whether the access of an operand (a) results in a suspension, and (b) requires modification of the access bits. Modifications of access bits are delayed until possible task suspensions and cache slot replacements are complete. Since these comparisons are completed concurrently with instruction execution, the cycle time is not significantly increased.

5.5 Communication

Low overhead communication is supported on Pica with the SEND family of instructions. This class of in-

structions can initiate a new message, inject one or two words from a context or an immediate operand into the network, and append an end of message marker. This is identical to the technique used in the Message Driven Processor [9].

5.6 Sequential Instructions

Efficient support for sequential code sequences remains an important requirement, even for fine grain tasks. The Pica instruction set retains the basic suite of sequential instructions: conditional branches, logical operations, shifts, and arithmetic operations.

Instruction word operands specify slots in the active and data contexts. The LOAD and STORE instructions support accesses to other contexts in local memory by specifying both the context ID and offset. Synchronization locks are tested for local memory accesses. During a load, the accessed value (including synchronization bits) is stored in a temporary register to avoid conflicts with slots in the context cache.

A no-exception datapath attempts to continue processing when errors occur, reducing the need for runtime software.

6 Status and Summary

The communications and processing components of a highly parallel, optically interconnected Pica architecture have been presented. The optical communication has been demonstrated using through-wafer thin film optoelectronic devices which operate at a wavelength to which silicon is transparent. Analog circuits have been designed which operate at data rates of 100 Mbps which interface to the Pica network interface. The initial processor design under implementation targets 73 Kbits of static storage (2048 words, 442K transistors, 30 mm^2) and 100K transistors for the datapath and controller. This should allow a four-node chip using current technology. Test components of the processing node and optical network are being fabricated. Simulations of the network performance are being conducted [20, 21, 13]. Plans for a full-scale prototype are underway, but are still preliminary. A full-scale prototype system will have the following characteristics. The node being designed contains 2048 36-bit words of local memory (256 contexts), a 32-bit integer processor, and a network interface. The estimated instruction rate is 50 MIPS. A chip contains four nodes and 3.2 Gbits/sec optical I/O bandwidth. A processing plane contains 64 chips (256 nodes, 12,800 MIPS) and measures approximately 10 cm by 10 cm. 16 planes contain 1024 chips (4096 nodes, 204,800 MIPS) and fit inside a cube 10 cm on a side. 819.6 Gbits/sec of I/O bandwidth is available from chips on the top and bottom of the cube. Sides of the cube are used for power and cooling mechanical connections. The object recognition application domain has been selected for evaluating the Pica. This domain is chosen because (a) massively parallel algorithms are available, (b) high-throughput processing is required, and (c) extensive floating-point performance is not needed. An assembler and instruction-level simulator are being used to develop application programs and evaluate the architecture [5]. A compiler for a higher-level language (Pica-C) is in progress.

Using these tools, several important image processing operations and applications have been implemented, including convolution, relaxation, FFT, edge detection, JPEG compression, and array manipulation [11, 18]. The largest implemented application for Pica is a Maximum Likelihood Expectation Maximization algorithm for positron emission tomography [7, 6], developed in conjunction with the proposer's research group and Crawford Long Hospital in Atlanta. This application, which reconstructs 21 slices of 256 x 256 pixels over 120 angles, offers a effective speedup of 900 as compared with the sequential workstation currently in use.

Using the instruction-level simulator, actual message traces from these applications can be collected and used by a second generation network simulator that is nearing completion. This network workload provides a more realistic comparison of offset-cube networks with other topologies (e.g., k-ary, 3-cubes).

References

[1] William C. Athas and Charles L. Seitz. Multicomputers: Message-Passing Concurrent Computers. *IEEE Computer*, 21(8):9–24, August 1988.

[2] G. Augustine, N. M. Jokerst, and A. Rohatgi. Single Crystal Thin Film InP: Fabrication and Absorption Measurements. *Applied Physics Letters*, 61(12):21–23, September 1992.

[3] K. H. Calhoun, C. B. Camperi-Ginestet, and N. M. Jokerst. Vertical Optical Communication Through Stacked Silicon Wafers Using Hybrid Monolithic Thin Film InGaAsP Emitters and Detectors. *IEEE Photonics Technology Letters*, 5(2):254–257, February 1993.

[4] C. Camperi-Ginstet, M. Hargis, N. Jokerst, and M. Allen. Alignable Epitaxial Liftoff of GaAs Material with Selective Deposition Using Polyimide Diaphragms. *IEEE Transactions Photonics Technology Letters*, 3(12):1123–1126, December 1991.

[5] Huy Cat, Jose Cruz-Rivera, W. Stephen Lacy, Mac Baker, John Eble, Abelardo Lopez-Lagunas, Mike Hopper, and D. Scott Wills. The Pica Architecture and its Application. *submitted to IEEE Transactions on Parallel and Distributed Systems*, February 1994.

[6] C. M. Chen, S. Y. Lee, and Z. H. Cho. Parallelization of the EM Algorithm for 3-D PET Image Reconstruction. *IEEE Transactions on Medical Imaging*, 10(4):513–521, December 1991.

[7] J. L. Cruz-Rivera, E. V. R. DiBella, D. S. Wills, T. K. Gaylord, and E. N. Glytsis. Parallelized Implementation of the Maximum Likelihood Expectation Maximization Algorithm on a Fine-grained Optically Interconnected Architecture. to be submitted to *IEEE Transactions on Medical Imaging*, 1994.

[8] William J. Dally. Performance Analysis of k-ary n-cube Interconnection Networks. *IEEE Transactions on Computers*, C-39(6):775–785, June 1990.

[9] William J. Dally, J.A. Stuart Fiske, John S. Keen, Richard A. Lethin, Michael D. Noakes, Peter R. Nuth, Roy E. Davison, and Gregory A. Fyler. The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms. *IEEE Micro*, 12(2):23–39, April 1992.

[10] William J. Dally and Charles L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnected Networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[11] W. Eric L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* The MIT Press, Cambridge, MA, 1990.

[12] Matthias Grossglauser and D. Scott Wills. Performance Analysis of an Offset Cube Network using Optical Interconnect. VLSI Architectures Group Technical Report, January 1993.

[13] W. Stephen Lacy, Jose Cruz-Rivera, Matthias Grossglauser, and D. Scott Wills. A Scalable Optical Interconnection Network for Fine-Grain Parallel Architectures. *submitted to IEEE Transactions on Computers*, March 1994.

[14] David May, Roger Shepherd, and Peter Thompson. The T9000 Transputer. In *1992 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 209–212, October 1992.

[15] Peter R. Nuth and William J. Dally. A Mechanism for Efficient Context Switching. In *1991 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 301–304, October 1991.

[16] Daniel A. Reed and Richard M. Fujimoto. *Multicomputer Networks: Message-Based Parallel Processing.* Scientific Computation Series. MIT Press, 1987.

[17] Burton Smith. The Architecture of HEP. In J. S. Kowalik, editor, *Parallel MIMD Computation: HEP Supercomputer and Its Application*, chapter 1, pages 41–55. MIT Press, 1985.

[18] Gregory K. Wallace. Overview of the JPEG (ISO/CCITT) Still Image Compression Standard. In Richard Feinberg, editor, *Current Overviews in Optical Science and Engineering I*, pages 358–371. SPIE Optical Engineering Press, Bellingham, Washington, 1990.

[19] D. Scott Wills and William J. Dally. Pi: A Parallel Architecture Interface. In *The Fourth Symposium on the Frontiers of Massively Parallel Computation*, pages 345–352. IEEE Computer Society Press, October 1992.

[20] D. Scott Wills and Matthias Grossglauser. A Scalable Optical Interconnection Network for Fine-Grain Parallel Architectures. In *1993 International Conference on Parallel Processing*, pages I–154 – I–157, 16-20 August 1993.

[21] D. Scott Wills and Matthias Grossglauser. A Three-Dimensional Optical Interconnection Network for Fine-Grain Parallel Architectures. In *Proceeding of the IEEE Lasers and Electro-Optics Summer Topical Meeting on Hybrid Optoelectric Integration and Packaging*, pages 21–22, 26-28 July 1993.

[22] E. Yablonovitch, E. Kapon, T. Gmitter, C. Yun, and R. Bhat. Double Heterostructure GaAs/AlGaAs Thin Film Diode Lasers on Glass Substrates. *IEEE Transactions Photonics Technology Letters*, 1(2):41–42, February 1989.