ELSEVIER

Future Generation Computer Systems 795 (2000) 1–18

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

A new-generation parallel computer and its performance evaluation☆

Sotirios G. Ziavras [a,*], Haim Grebel [a], Anthony T. Chronopoulos [b], Florent Marcelli [a]

[a] *Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA*
[b] *Division of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249, USA*

Received 15 May 1999; received in revised form 15 May 2000; accepted 15 May 2000

Abstract

An innovative design is proposed for an MIMD distributed shared-memory (DSM) parallel computer capable of achieving gracious performance with technology expected to become feasible/viable in less than a decade. This New Millennium Computing Point Design was chosen by NSF, DARPA, and NASA as having the potential to deliver 100 TeraFLOPS and 1 PetaFLOPS performance by the year 2005 and 2007, respectively. Its scalability guarantees a lifetime extending well into the next century. Our design takes advantage of free-space optical technologies, with simple guided-wave concepts, to produce a 1D building block (BB) that implements efficiently a large, fully connected system of processors. Designing fully connected, large systems of electronic processors could be a very beneficial impact of optics on massively parallel processing. A 2D structure is proposed for the complete system, where the aforementioned 1D BB is extended into two dimensions. This architecture behaves like a 2D generalized hypercube, which is characterized by outstanding performance and extremely high wiring complexity that prohibits its electronics-only implementation. With readily available technology, a mesh of clear plastic/glass bars in our design facilitate point-to-point bit-parallel transmissions that utilize wavelength-division multiplexing (WDM) and follow dedicated optical paths. Each processor is mounted on a card. Each card contains eight processors interconnected locally via an electronic crossbar. Taking advantage of higher-speed optical technologies, all eight processors share the same communications interface to the optical medium using time-division multiplexing (TDM). A case study for 100 TeraFLOPS performance by the year 2005 is investigated in detail; the characteristics of chosen hardware components in the case study conform to SIA (Semiconductor Industry Association) projections. An impressive property of our system is that its bisection bandwidth matches, within an order of magnitude, the performance of its computation engine. Performance results based on the implementation of various important algorithmic kernels show that our design could have a tremendous, positive impact on massively parallel computing. 2D and 3D implementations of our design could achieve gracious (i.e., sustained) PetaFLOPS performance before the end of the next decade.

Keywords: PetaFLOPS computer performance; Advanced computer architecture; Generalized hypercube; Parallel computer; Optical interconnection network

☆ The work presented in this research was supported in part by NSF and DARPA, also co-sponsored by NASA, under the New Millennium Computing Point Design Grant ASC-9634775.

* Corresponding author. Tel.: +1-201-596-5651; fax: +1-201-596-5680.
E-mail address: ziavras@njit.edu (S.G. Ziavras)

0167-739X/00/$ – see front matter
PII: S0167-739X(00)00082-0

1. Introduction

The demand for ever greater performance by many computation problems has been the driving force for the development of computers with thousands of processors. Two important aspects are expected to dominate the massively parallel processing field. High-level parallel languages supporting a shared address space (for DSM computers) and point-to-point interconnection networks with workstation-like nodes. Near PetaFLOPS (i.e., 10^{15} floating-point operations per second) performance and more is required by many applications, such as weather modeling, simulation of physical phenomena, aerodynamics, simulation of neural networks, simulation of chips, structural analysis, real-time image processing and robotics, artificial intelligence, seismology, animation, real-time processing of large databases, etc. Dongarra pointed out in 1995 that the world's top 10 technical computing sites had peak capacity of only about 850 GigaFLOPS, with each site containing hundreds of computers. The goal of 1 TeraFLOPS (i.e., 10^{12} floating-point operations per second) peak performance was reached in late 1996 with the installation of an Intel supercomputer at Sandia Laboratories.

The PetaFLOPS performance objective seems to be a distant dream primarily because of the, as currently viewed, unsurmountable difficulty in developing low-complexity, high-bisection bandwidth, and low-latency interconnection networks to connect thousands of processors (and remote memories in DSM systems). To quote Dally, "wires are a limiting factor because of power and delay as well as density" [6]. Several interconnection networks have been proposed for the design of massively parallel computers, including, among others, regular meshes and tori [5], enhanced meshes, fat trees (direct binary), hypercubes [17], and hypercube variations [15,24,26,27]. The hypercube dominated the high-performance computing field in the 1980s because it has good topological properties and rather rich interconnectivity that permits efficient emulation of many topologies frequently employed in the development of algorithms [17,29]. Nevertheless, these properties come at the cost of often prohibitively high VLSI (primarily wiring) complexity due to a dramatic increase in the number of communication channels with any increase in the number of PEs (processing elements). Its high VLSI complexity is undoubtedly its dominant drawback, that limits scalability [25] and does not permit the construction of powerful, massively parallel systems. The versatility of the hypercube in emulating efficiently other important topologies constitutes an incentive for the introduction of hypercube-like interconnection networks of lower complexity that, nevertheless, preserve to a large extent the former's topological properties [26,27].

To support scalability, current approaches to massively parallel processing use bounded-degree networks, such as meshes or k-ary n-cubes (i.e., tori), with low node degree (e.g., FLASH [11], Cray Research MPP, Intel Paragon, and Tera). However, low-degree networks result in large diameter, large average internode distance, and small bisection bandwidth. Relevant approaches that employ reconfiguration to enhance the capabilities of the basic mesh architecture (e.g., reconfigurable mesh, mesh with multiple broadcasting, and mesh with separable broadcast buses) will not become feasible for massively parallel processing in the foreseeable future because of the requirements for long clock cycles and precharged switches to facilitate the transmission of messages over long distances [32].

The high VLSI complexity problem is unbearable for generalized hypercubes. Contrary to nearest-neighbor k-ary n-cubes that form rings with k nodes in each dimension, generalized hypercubes implement fully connected systems with k nodes in each dimension [2]. The nD (symmetric) generalized hypercube $GH(n, k)$ contains k^n nodes. The address of a node is $x_{n-1}x_{n-2}\cdots x_1x_0$, where x_i is a radix-k digit with $0 \leq x_i \leq k-1$. This node is a neighbor to the nodes with addresses $x_{n-1}x_{n-2}\cdots x'_i\cdots x_1x_0$ for all $0 \leq i \leq n-1$ and $x'_i \neq x_i$. Therefore, two nodes are neighbors if and only if their n-digit addresses differ in a single digit. For the sake of simplicity, we restrict our discussion to symmetric generalized hypercubes where the nodes have the same number of neighbors in all dimensions. Therefore, each node has $k-1$ neighbors in each dimension for a total of $n(k-1)$ neighbors per node. The nD $GH(n, k)$ has diameter equal to only n. Fig. 1 shows the $GH(2, 7)$ with two dimensions (i.e., $n = 2$) and $k = 7$. For $n = 2$ and k an even number, the diameter of the generalized hypercube is only 2 and its bisection width is the immense $k^3/4$. The increased VLSI/wiring cost of generalized hypercubes results in outstanding performance that permits optimal emulation of hypercubes and k-ary n-cubes, and efficient implementation of complex communication patterns.

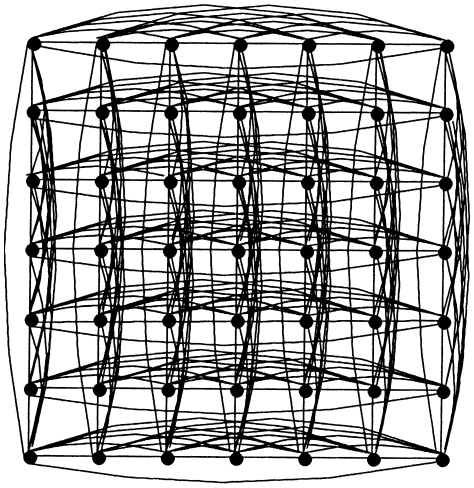

Fig. 1. The generalized hypercube $GH(2, 7)$.

Table 1 compares the numbers of channels in the binary hypercube (i.e., m-cube), the k-ary n-cube, and the generalized hypercube $GH(n, k)$, all with the same number N of processors. We assume bidirectional data channels for full-duplex communication, and that $N = k^n = 2^m$ (therefore, $k = N^{1/n} = 2^{m/n}$). For example, for $m = 14$ (i.e., for systems with N=16 384 processors) and 64-bit channels, the numbers of wires for data transfers are

- 14 680 064 wires for the 14-cube;
- 4 194 304 wires for the 128-ary 2-cube; and
- 266 338 300 wires for the $GH(128, 2)$.

In order to reduce the number of communication channels in systems similar to the generalized hypercube, the spanning bus hypercube uses a shared bus for the implementation of each fully connected subsystem in a given dimension [24]. However, shared buses result in significant performance degradation because of the overhead imposed by the protocol that determines each time ownership of the bus. Similarly, hypergraph architectures implement all possible permutations of their nodes in each dimension by employing crossbar switches [21]. Reconfigurable generalized hypercubes interconnect all nodes in each dimension dynamically via a scalable mesh of very simple, low-cost programmable switches [28]. However, all these proposed reductions in hardware complexity may not be sufficient for very high performance, such as PetaFLOPS-related, computing. To quote Patterson, "Currently the most expensive scheme is a crossbar switch, which provides an explicit path between every communicating device. This becomes prohibitively expensive when connecting thousands of processors" [14].

Table 1
Comparison of interconnection networks, assuming full-duplex bidirectional data channels

Network	Number of channels	Diameter
m-cube	$m \cdot 2^m$	$m = n \cdot \log_2 k$
k-ary n-cube	$2 \cdot n \cdot 2^m$	$\left\lceil \frac{k}{2} \right\rceil$
$GH(n, k)$	$(k-1)n \cdot 2^m$	n

To summarize, low-dimensional massively parallel computers with full connectivity for nodes in each dimension, such as generalized hypercubes, are very desirable because of their outstanding topological properties (e.g., extremely small diameter and average internode distance, and immense bisection width), but their electronic implementation is a Herculean task because of packaging (and primarily wiring) constraints. Therefore, the introduction of pioneering technologies for the implementation of such systems could give life, for the first time, to scalable and feasible computing platforms capable of very high performance. This is our main objective. We have chosen a combination of electrical and free-space optical technologies to satisfy this objective. Our free-space optics approach results in a dramatic reduction in the number of wires. There is another major drawback for pure electrical implementations of systems with thousands of processors; processor speeds increase much faster than memory and interconnection network speeds, and therefore there is an utmost need for the development of very advanced memory-latency hiding mechanisms, namely prefetching, cache coherence, multithreading, and relaxed memory consistency. However, mitigation of the memory-latency problem is possible if free-space optical technologies are used for the implementation of large, almost fully connected, point-to-point interconnection networks.

Optical technologies have been enlisted before in the design of parallel computers [1,8,10,21]. However, past efforts were often plagued by large power consumption (often due to redundant broadcasts), inefficient reconfiguration schemes with mechanical components that did not match electronic speeds, unreliability, strict alignment requirements, large complex-

ity prohibiting scalability, etc. Also, interconnects with wavelength selectivity for channel allocation have been under extensive study recently [1,8,16].

An optical crossbar-like multichannel switch is employed in [21] to support full connectivity in 1D subsystems for the implementation of a "hypermesh". A single fiber is used with WDM techniques to implement permutations among all nodes in the subsystem. The maximal size of the WDM optical crossbar is limited to only 16 nodes for the foreseeable future due to constraints on wavelength tunability with a single fiber. To build a massively parallel hypermesh machine, one must use many smaller WDM optical switches arranged into a chosen network architecture. Such a machine, however, may be slow in data transfers, routing may become cumbersome, and the cost and packaging complexity of the interconnection network may be prohibitively high. On the other hand, our design is scalable, has very low packaging complexity, uses fast point-to-point interconnection technology, and is characterized by low power consumption. The latter design can implement not only permutations among the nodes but also more powerful communications operations such as multicasting, broadcasting, all-to-all personalized, etc. Optical hypermeshes can implement a class of systems that are a subset of our optics-based generalized hypercube architecture.

Other proven architectural features, in addition to the chosen interconnection network technologies, are required for the success of any proposed system. DSM systems already dominate the massively parallel processing field [4,11] because the simultaneous incorporation of the message-passing and shared-memory communication paradigms introduces versatility in programming [4,12]. Around the year 2005, 8- to 16-way multithreaded microprocessors are expected to be common [9,13]. A DSM system with thousands of processors then may be handling simultaneously hundreds of thousands to a million threads, thus making the problems of cache coherence, debugging, scheduling, and performance monitoring extremely difficult to handle [9]. Hardware/software codesign will be needed to develop relevant solutions for such systems.

To summarize, we strongly believe that free-space optical technologies will have a very significant influence on massively parallel processing because of reduced packaging complexity that facilitates the construction of powerful systems with increased connectivity. Optical technologies employing simultaneously TDM and WDM techniques eliminate the need for wires in the implementation of communications channels, and could be used to implement densely populated, fully connected BBs with large numbers of processors and small packaging complexity. Ours is a meticulous effort towards formulating a relevant, attainable objective and presenting a viable solution to fulfill this objective. This effort covers innovative architectures, feasibility analysis for corresponding designs, applications development, and performance evaluation [30,31].

Our paper is organized as follows. Section 2 presents the basic architecture of our NSF/DARPA/NASA-funded New Millennium Computing Point Design. Section 3 contains a detailed description of our design for a system capable of 100 TeraFLOPS by the year 2005. Overall performance characteristics of this system and a feasibility analysis are also included. Section 4 presents an analysis for the optical component of the interconnection network. Section 5 contains performance results for some important computation- and/or communication-intensive problems. Finally, conclusions are presented in Section 6.

2. Basic architecture

Our architecture encompasses a 2D interconnection network that employs electrical and optical technologies. Section 2.1 presents the structure of the 1D building block (BB) and issues related to its implementation. The 2D complete system is constructed by repeating this 1D BB in two dimensions, and also incorporating additional glue logic. Section 2.2 describes the 2D complete structure.

2.1. 1D building block

Our basic design takes advantage of free-space optical technologies to produce a 1D fully connected, scalable BB capable of implementing bit-parallel communications channels. The first objective is to produce a low-cost, powerful, free-space, reliable, point-to-point communications system of low packaging complexity that incorporates some guided-wave concepts. Guided, planar, optical interconnects can produce a robust system with built-in optical filtering at the expense of sys-

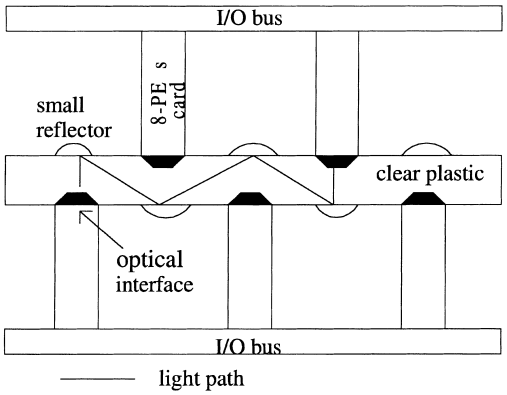

Fig. 2. Block diagram for part of the 1D BB.

tem complexity [22]. On the other hand, free-space interconnects are relatively simple to implement. However, they are more prone to system vibrations and source/detector misalignment. Misalignment problems are solved gracefully in our design, as shown in this section. Free-space interconnects possess an energy-bandwidth product which is larger than their electronic counterpart [3].

Part of the BB is schematically shown in Fig. 2. It is a 1D array of 8-PE cards attached to an inexpensive clear plastic/glass bar that provides alignment for optical transmissions. Each card carries the entire processing and memory power of eight processors fully interconnected via an electronic crossbar. This approach was chosen because of the high efficiency of small electronic crossbars. Each card is interfaced with optical transmitter/receiver modules and attached prismatic elements for inter-card data transfers, and the destination address for a data transfer is decoded to determine the prismatic element and associated modules to be used for the appropriate path.

In an optical cycle, 32 bits of information can be sent in parallel from one card to another via a card-to-card (i.e., point-to-point) color-coded interconnect, using 32 distinct colors (i.e., by the WDM technique); these 32 colors are the same for all of the cards. Actually, each inter-card channel is 128 bits wide because of the chosen format for messages (shown in Fig. 3), and therefore 128-bit information is transmitted each time (i.e., during a PE's cycle) by utilizing four (i.e., 128/32) optical cycles (i.e., by the TDM technique). All eight PEs on a card share the same lasers and receivers for inter-card transmissions, so that a PE P_{ij} on a given card, i, and with position j on the card, uses the same color set of 32 wavelengths, λ_m, however with a different RF carrier frequency (i.e., by the TDM technique). To summarize, the WDM and TDM techniques are used as follows:

- WDM with 32 wavelengths for bit-parallel transmissions involving 32 bits;
- TDM with four communication cycles to implement 128-bit transmissions, where each of these four cycles is of the aforementioned WDM type implementing 32-bit transmissions; and
- TDM with eight communication cycles, so that the eight PEs on a card can share the optical transmit/receive modules assigned for the exchange of information with another 8-PE card.

The chosen prismatic element for a data transfer determines a specific optical path via the set of reflectors used between the transmitting and receiving cards. Since any two cards communicate via dedicated prismatic elements, multi-access node communication is available. Common colors from different cards are detected by different detector arrays at different locations on the card's interface. Separation among the messages sent to a given card from other cards is made by separating the fields of view, and therefore activating different detectors on the receiving card. The receiver demultiplexes the information and sends it to the destined PE on the given card. The receiver may utilize a coherent detection system to increase its sensitivity by employing a distributed optical clock. The clock frequency may be transmitted on a color different from

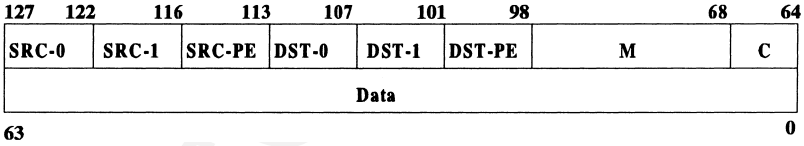

Fig. 3. Basic format for 128-bit messages, assuming DSM addresses.

the information or may be incorporated in the address field.

2.2. Complete 2D system

Extension of the 1D fully connected BB into a 2D configuration is now in order. In addition to interfacing a horizontal clear plastic bar (used for interconnects in the first dimension), each 8-PE card now also belongs to a similar 1D structure in the second dimension. Therefore, each card also interfaces a vertical plastic bar. The clear plastic columns are patterned with small metallic reflectors and prismatic interfaces, as for the horizontal bars. All in all, the system may be viewed as a 2D array, with rows and columns containing fully interconnected PEs. It behaves like a 2D generalized hypercube; each node of the generalized hypercube contains eight fully interconnected PEs (they are fully interconnected via an on-card electronic crossbar network).

Assume, as earlier, 128-bit channels and a communications frequency for each PE of f_c MHz. With the TDM technique applied twice (as described earlier) each laser source of hue, λ_m, has to operate at $8 \times (128/32) f_c = 32 f_c$ MHz, in order to facilitate simultaneous inter-card data transfers involving all eight PEs on the card. For example, according to SIA (Semiconductor Industry Association) projections for the year 2005 [13], f_c will be 375 MHz (i.e., frequency for inter-chip data transfers), and therefore lasers operating at $32 \times 375\,\text{MHz} = 12\,\text{GHz}$ will then be needed. Such lasers and corresponding multiplexers/demultiplexers already exist [19,20]. A typical clear plastic has a transmission factor of 0.25 dB/cm. A typical coherent detection system is able to detect −30 dB of optical signals, which is translated to a maximum optical distance of 30/0.25=120 cm. Assuming that the distance between adjacent cards on the same side of the bar is about 5 cm (about 2 in.) and the thickness of the plastic bar is equal to 2.5 cm, about 40 cards with up to 320 PEs (i.e., 40 cards× 8 PEs/card) can be accommodated in the BB. Fig. 4 shows an implementation for the card interface that follows this logic, assuming a 2D system with 40 cards per dimension. This complete system may contain up to $40^2 \times 8 = 12\,800$ PEs. If the plastic bars are replaced by more transparent material, such as glass, the system can accommodate much larger numbers of PEs. In addition, a DMA controller is part of each PE in our detailed design, as shown in Fig. 5 for the year

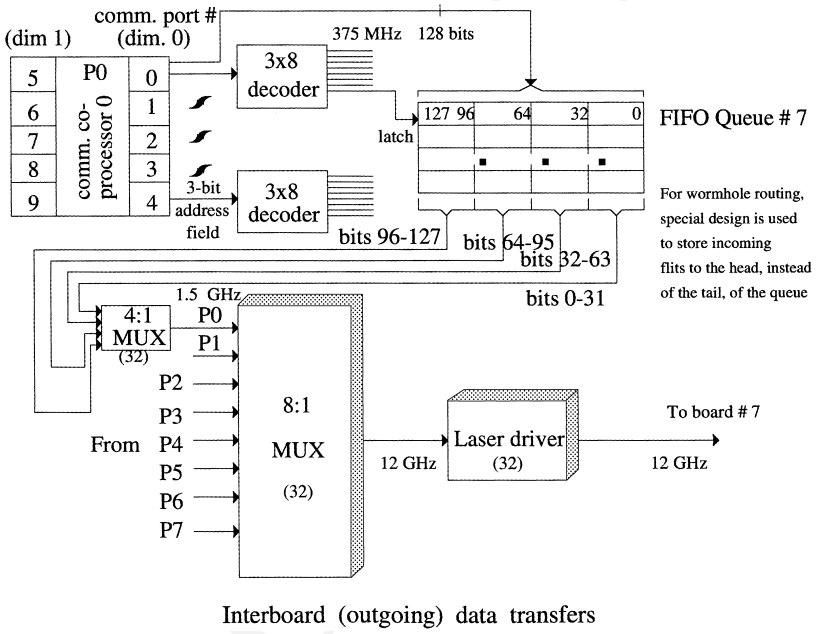

Fig. 4. Employing TDM and WDM techniques for inter-card transmissions (design for the year 2005 based on SIA projections for semiconductor components).

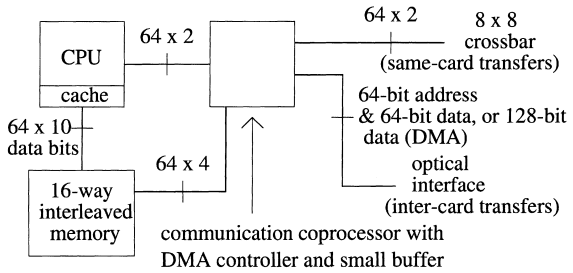

Fig. 5. PE schematic diagram (design for the year 2005 conforming to SIA projections).

Table 2
Bandwidth of communication links, including all associated latencies[a]

Type of data transfer	Bandwidth
CPU/local-memory	30 GB/s
CPU/RMSC	1.540 GB/s
CPU/RMAC	1.075 GB/s
Memory/memory via DMA	6 GB/s
Bisection bandwidth	31.104 TB/s

[a] All CPU to/from memory bandwidths are for single data transfers (RMSC: remote memory on the same card; RMAC: remote memory on another card).

2005 implementation (the numbers of pins for the chips conform to SIA projections); large numbers of DMA controllers are essential for very high, sustained performance. We assume memory modules with two ports that permit simultaneous load/store by the CPU and the communication coprocessor.

3. Case study: 100 Tera(FL)OPS performance

Our design is the result of a project funded jointly by NSF, DARPA, and NASA, under their New Millennium Computing Point Design program that has funded eight projects nationwide. The objective of this federal program is the design and feasibility analysis of massively parallel computers that could deliver 100 TeraFLOPS performance by the year 2005. They believe that the success of this aggressive program will lead by the year 2007 to the development of computers capable of PetaFLOPS (i.e., 10^{15} floating-point operations per second) performance. In contrast, the expected evolution in computer architecture and semiconductor technology will not make this possible before the year 2015!

Based on SIA projections [13], commodity microprocessors will be capable of 10 GigaFLOPS by 2005, and therefore we need at least 10 000 processors for a system capable of delivering 100 TeraFLOPS peak performance. We assume 10-way multithreaded processors of 10 GigaFLOPS and 1 GHz each (SIA projections). We can assume CPUs with 10 floating-point units because of their 1 GHz clock and their 10 GFLOPS performance. Each processor operates on 64-bit numbers.

For such a massively parallel system to be viable, its physical volume (i.e., size) must be reasonably small, and its communications and I/O capabilities should match (within an order of magnitude) the speed of its computation engine. As emphasized earlier, free-space optical technologies eliminate the need for wires in the implementation of communication channels, and could be used to realize fully connected BBs with large numbers of processors and small physical volume. Our design takes advantage of free-space optical technologies to produce a 1D fully connected and scalable BB, as described in Section 2. Since the complete system is a 2D configuration of 8-PE cards, we need a BB with 36 cards; the four additional cards in each BB of Section 2 can be used for I/O and/or fault tolerance. Then, the total number of computing PEs is $36^2 \times 8$ or 10 368 (for 103.68 TeraFLOPS peak performance). This BB is actually a fully connected system of 288 computing PEs (i.e., 36 cards$\times$ 8 PEs/card) because the eight PEs on each card are fully interconnected via an electronic crossbar network, and the bandwidth of the optical interface is such that all eight PEs on the card can be involved simultaneously in inter-card data transfers without any performance degradation (due to the TDM approach and the bit-parallel communication channels). Our system is scalable in terms of both its architecture and the optics technology, and therefore further performance improvement is possible, if desired.

Table 2 summarizes the performance characteristics of the BB and the bisection bandwidth of the complete system. It is easy to see the outstanding performance of the system's interconnection network. There cannot be any realistic electronic implementation of an interconnection network that matches these characteristics.

3.1. Feasibility analysis for optical components

Although the required number of 8-PE cards in the BB is 36 for 100 TeraFLOPS performance, our analysis here is for 40 cards. The four additional cards in each BB could be used for other services (e.g., I/O and fault tolerance), or to increase the size of the proposed system to 12 800 processors for 128 TeraFLOPS peak performance.

3.1.1. Optical Interface

The optical interface is composed of prismatic elements, and two dedicated arrays of laser diodes (LDs) and detectors arranged underneath each element. The element is acting like a collimating lens that directs the light out of 32 lasers to the detector array on the destination card through the appropriate reflectors. The same optical element focuses the incoming light from the sender onto the dedicated detector array. Each detector in the 32-element array is equipped with a color filter that allows only a particular hue to pass through. In this way, we separate further the 32 bits from one another. With current technology, each LD or detector could occupy a square area of 0.5 mm × 0.5 mm (including its electrodes), and the entire area occupied by the 32 LD/detector pair array will be 1 mm × 16 mm. Since the interface extends throughout the width of the card (about 20 cm), we may divide the 39 LD/detector pair arrays into 10 groups. Therefore, the width of the optical interface, as viewed from above, will be about 4 mm.

3.1.2. Light sources and detectors

The light sources will be LDs made of GaAs at wavelengths between 0.8 and 0.9 μm. The GaAs technology is a mature technology which is able to produce high-speed lasers at a reasonable cost. The detectors will make use of silicon technology. Each detector is equipped with a thin layer which serves as an optical filter. The filter for an array of detectors is easily made in an incremental manner. Chromatic dispersion is negligible at these distances (only 10^{-11} s for 1 m of propagation).

3.1.3. Optical/electrical power consumption

Our analysis here is very conservative, even with current optical components. Each LD puts out an average of 250 μW of optical power. This power is smaller or larger for short or long data transmissions, respectively. Each card transmits information to each of the 39 other cards in the BB via 32 dedicated LDs, and radiates, on the average, $39 \times 32 \times 250\,\mu\text{W} = 312\,\text{mW}$ of optical power. The electrical-to-optical power conversion ratio is normally 30%, thus the RMS (root-mean-square) value of the electrical power consumption for each card is about 1 W. Expected improvements in optoelectronic devices will further enhance these numbers.

4. Analysis of the optical interconnection network in the BB

Here we present results of analysis, simulation, and feasibility study for the optical interconnection network of our design.

4.1. Optical filters

A Fabry–Perot (FP) etalon is used as an optical filter [23]. In the case of a passive FP filter, the transmission characteristics are wavelength-dependent, as the filter transmits only frequencies that correspond to the longitudinal mode frequencies. For a FP filter, the channel spacing can be as small as $3\Delta\nu_{\text{FP}}$, where $\Delta\nu_{\text{FP}}$ is the spectral width of the filter. By using $F = 100$ (F is the finesse of the filter), the maximum number of channels, N, for the FP filter is restricted by $N < \pi(\sqrt{R}/3(1-R))$ with R being the reflectivity of the mirrors. A filter with $R = 99\%$ reflecting mirrors can be used to select up to $N = 104$ channels, which is much larger than the 32 bits (colors) per word used in our case.

4.2. Simulation of a simple free-space optical interconnect

First, an alignment tolerance is chosen between the lenses at the transmitting and the receiving ends. Then, the displacement as a function of lens separation (d12) is computed. The tolerance reflects the mechanical accuracy and operational stability that must be achieved with the mounting of the component. We will use a Gaussian beam propagation method [23] to account for the beam divergence. The overlap integral between

the optical beam and the aperture of the collection lens is assessed to determine the cross talk. A second overlap integral between the Gaussian beam focused by the collection lens and the detector aperture determines the power delivered to the detector. The collected power and cross talk give values for the received signal contrast that affects the bit error rate, the signal-packing density, and the optical power required for the laser diode.s

4.2.1. Bit error rate and signal-packing density

The bit error rate (BER) indicates the required source power and signal-to-noise levels necessary to achieve a desired signal fidelity, and represents an important measure of system performance. Unlike optical communication links where BER $< 10^{-9}$ at GHz-transmission rates, here we choose BER $< 10^{-17}$. The requirements for achieving this BER can be determined from communication theory [18]. With Gaussian statistics we find that the probability of error (PE) is given by

$$\mathrm{PE} \cong \frac{1}{(2\pi Q)^{1/2}} \mathrm{e}^{(-Q^2/2)},$$

where Q is a normalized number that quantifies the quality of the current signal.

In order to achieve a BER of 10^{-17}, we need $Q = 8.5$. The average optical power, P, required from a source to drive a receiver at a desired Q and BER is given by

$$P = \frac{1-r}{1+r} Q \frac{hc}{\lambda e} \langle i_{\mathrm{NA}}^2 \rangle^{1/2} \left(\frac{N}{\eta} \right),$$

where r is the ratio of currents when the detector is in the low-illumination state and the high-illumination state, $\langle i_{\mathrm{NA}}^2 \rangle^{1/2}$ the RMS current noise generated by the detector and preamplifier circuit, η the product of the quantum efficiency of the detector and the efficiency of the optical system, N the system fanout, and $hc/\lambda e$ is the voltage source equivalent of the optical field; hc/λ the photon energy and e is the electron charge. Typical experimental quantum efficiencies are on the order of 0.7–0.8. In addition, the contrast parameter r will decrease with optical and electrical cross talk and must also be determined to assess its effects on BER.

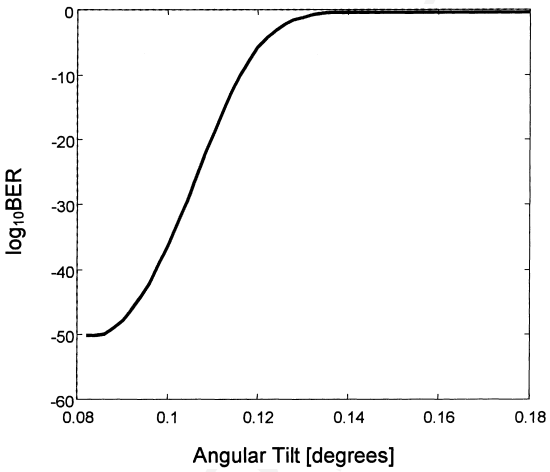

Fig. 6. The BER, in the logarithmic scale, as a function of the angular tilt for the most distant data transfers.

4.3. Alignment of the free-space optical system

Alignment of a free-space optical system is critical to its operation. We carried out a simulation of our optical system for 10 GHz transmission frequency. The mathematics that drive our simulation follow in Sections 4.3.1 and 4.3.2. The major constraint in our system is the angular tilt of the optical beam with respect to its original path. A value of 0.1° was found to optimize the system. A larger angular tilt will require an increase in the lens radius and, consequently, the system dimensions will increase too. On the other hand, we cannot compensate for the tilt by merely increasing the laser power because of the exponential behavior of the curve. In Fig. 6, we show the BER for laser power Plas=5 mW as a function of the angular tilt. We also show the BER as a function of the laser power in Fig. 7. Some of the optimized parameters are: the distance of propagation is 1 m, the angle of alignment tolerance is 0.1°, the reflective power loss is 0.25 dB/cm, the laser wavelength is 0.83 μm, the detector radius is 10 μm, the Q parameter of the receiver is 8 for a BER of 10^{-17}, the RMS current noise generated by the detector and preamplifier circuit is 814.6 nA for a bit rate of 10 Gbit/s; the relative received optical power becomes with the simulation 0.27.

We analyze below the possible causes for the misalignment and assess the tolerance for each case, based on an assessment of the BER.

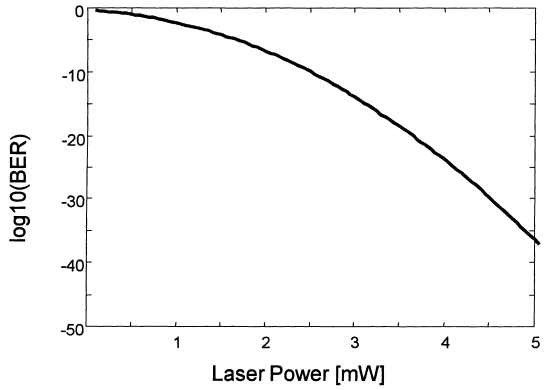

Fig. 7. The BER, in the logarithmic scale, as a function of the laser power.

4.3.1. Offsets in an integrated planar-optics interconnect

We resort to Fig. 8 in assessing the misalignment in the system. The offset Δx_i in the beam path at the ith reflection point is

$$\Delta x_i = ti\ [\tan\theta - \tan\theta_0],$$

where θ_0 is determined by the ideal path that is not affected by any other offset. The induced transverse offset Δy_i which is caused by an angular offset γ is

$$\Delta y_i = 2ti\,\frac{\tan\gamma}{\cos\theta}.$$

Angular offsets also change the path length of the beam inside the substrate. From Fig. 8, we find that the path length from the input to the ith reflection point is

$$z_i = \frac{ti}{\cos\theta\cos\gamma}.$$

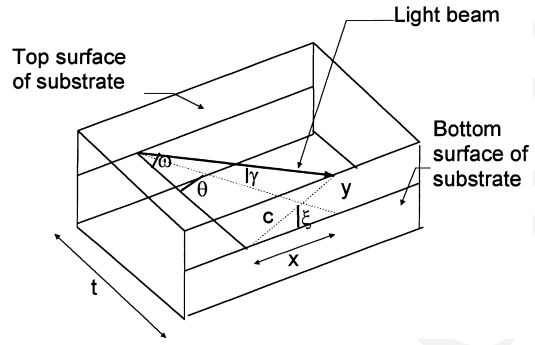

Fig. 8. Section of the substrate showing the path of light between two successive reflections.

This difference in path length results in a change in the spot size, caused by diffractive spreading at the ith reflection, and affects the efficiency. The third effect of angular offsets is a change in the eccentricity and orientation of the beam shape on the surface of the optical bar. Referring to Fig. 8, we find that the major axis of the elliptic spot is

$$2a = \frac{2s}{\cos\omega}, \tag{1}$$

where s is the spot size and $\omega = \cos^{-1}[\cos\theta\cdot\cos\gamma]$. Thus, as $\Delta\theta$ or γ increases, $\cos\omega$ decreases and the major axis and the eccentricity increase. This spreading of the spot may decrease the efficiency, depending on the relative sizes of the beam and the device, and on the intensity distribution of the beam. When $\gamma = 0, \omega = 0$ and the major axis of the ellipse remains on the x-axis. But if $\gamma \neq 0$, then the center of the ellipse moves to the point (x, y) and the major axis of the ellipse is along c. This new orientation of the major axis is found by a rotation of ξ rad, where $\xi = \tan^{-1}[\tan\gamma/\sin\theta]$. Depending on the shape of the element or receiving port of interest, this rotation of the ellipse may affect the efficiency.

4.3.2. Efficiency and alignability of the integrated planar-optical system for the case study

To obtain the efficiency of the system, we need to know the intensity distribution of the beam. We assume a circularly symmetric Gaussian beam. The intensity of such a beam with a total power P_0 can be expressed as

$$I(x, y) = \frac{2P_0}{\pi s_0^2}\exp\left[-\frac{2(x^2+y^2)}{s_0^2}\right].$$

Here, s_0 is the spot size of the beam at the source, and x and y are the coordinates of the intensity distribution. Considering the angle of incidence and the offsets in the integrated planar-optical system (IPOS), we can show that at the ith reflection point the intensity becomes

$$I(x, y) = \frac{2P_0\cos\theta}{\pi s_i^2} \times \exp\left\{-\frac{2[(x-\Delta h)^2\cos^2\theta + (y-\Delta y)^2]}{s_i^2}\right\}.$$

Here, $\Delta h = \Delta x_i + \Delta x$, s_i is the diverging spot size, with

$$s_i = s_0 + \tan\beta \frac{ti}{\cos\theta_0}$$

and β is the beam divergence.

At this point the major axis of the beam's elliptic profile on the IPOS surface is aligned with the x-axis of the IPOS and its elements. When γ offsets are added, however, the major axis of the ellipse is rotated away from the x-axis of the IPOS. In the rotated coordinate system of the ellipse, the total x and y transverse offsets, h and k, respectively, become

$$\Delta h = \Delta k \sin\xi + \Delta h \cos\xi,$$

$$\Delta k = \Delta k \cos\xi - \Delta h \sin\xi.$$

In addition to introducing Δy_i, the offset also changes the angle of propagation of the beam. The new angle is defined in Eq. (1). Using w, h, and k, we get the final form of the intensity distribution

$$I(x, y) = \frac{2P_0 \cos\omega}{\pi s_i^2} \times \exp\left\{\frac{-2[(x-h)^2\cos^2\omega + (y-k)^2]}{s_i^2}\right\},$$

where x and y now refer to the directions of the major and minor axes, respectively, of the elliptic spot.

We find the efficiency at the output port by setting $i = N$ and integrating (A9) over the area of the element or port and then dividing by the total beam power P_0. For our case study with circular elements, we obtain

$$\eta = \frac{2\cos\omega}{\pi s_i^2}\int_{-d}^{d} \exp\left[\frac{-2(y-k)^2}{s_i^2}\right] \times \left\{\int_{-(d^2-y^2)^{1/2}}^{(d^2-y^2)^{1/2}} \times \exp\left[\frac{-2(x-h)^2\cos^2\omega}{s_i^2}\right] \mathrm{d}x\right\} \mathrm{d}y.$$

One can perform the integration on x analytically and the integration on y numerically. This value is then used in assessing the BER.

To conclude, free-space optical systems are usually limited by system alignment and stability. We analyzed such a free-space optical system and found that the limiting factor is the angular tilt of the optical path. Owing to the highly dense interconnection scheme in our design, we were able to achieve very high bit rates with very moderate and practical angular tilt values.

5. Performance evaluation

Performance evaluation results are presented here for our case-study system capable of 100 TeraOPS, focusing on its communications capabilities and its implementing frequently used computation- and/or communication-intensive algorithmic kernels.

5.1. Data communications

Our general-purpose MIMD system targets the majority of the computation- and communication-intensive applications. The communications capabilities of its optical interconnection network resemble those of the extremely powerful 2D generalized hypercube, which is by far much better than any interconnection network that has ever been built for massively parallel processing. The generalized hypercube can emulate in optimal manners (i.e., with dilation and congestion of source edges equal to 1) the majority of the widely used topologies, such as binary hypercubes, k-ary n-cubes, etc. In addition, the incorporation of efficient memory-latency hiding techniques (e.g., cache coherence and prefetching) then becomes a viable task because of the system's extremely small diameter (i.e., 2), immense bisection width, and high-speed network. Similar tasks are of extraordinary difficulty for pure electronic designs of the same size. Our complete DSM design is consistent in terms of inter-PE data-transfer speeds and dense connectivity patterns throughout, thus supporting scalability and ease of mapping application tasks to the system. Other designs are characterized by limited bandwidth and substantial latencies that result in unpredictable performance. In contrast, the very efficient and uniform interconnection of resources in our system makes performance prediction much more accurate. Also, we expect algorithms for our system to be developed easily, by assuming an MIMD fully connected architecture as part of the programming model.

Wide usability of our design is further substantiated here through additional performance evaluation. Theoretical analysis and simulations were used for a highly accurate performance evaluation of the proposed system. For a system to potentially have a niche in the

massively parallel processing field, it must provide direct support for some very frequently used communication operations that are very costly to implement by repeating some of the basic communication primitives. Such operations are: multicasting, broadcasting, reduction using associative operators, prefix computations, and barrier synchronization. Other less frequent operations are one-to- all personalized (i.e., scattering) and its dual single-node gather communications, and total exchange. Our results show that these communications can be carried out consistently and efficiently throughout the entire system.

More specifically, for the implementation of one-to-all broadcasting, all-to-all broadcasting, one-to-all scattering, and all-to-all scattering, the optimal techniques in [7] were used for the generalized hypercube of 8-PE cards. A balanced spanning tree rooted at the node with address 0, which is employed by the latter techniques, is created at static time. For more than one source nodes, appropriate transformations are applied at run time in a distributed fashion in order to generate spanning trees rooted at other nodes; a sophisticated algorithm performs the latter transformations in constant time. For the aforementioned all-to-all communication operations, messages originating at individual nodes never compete for the same edge at the same time. We have also developed algorithms for multicasting that use the same spanning tree as the basis. For the sake of brevity, details of these implementations are omitted.

5.2. Implementation of algorithmic kernels

The efficient implementation of application algorithms on the proposed system is vital for its success. This task benefits tremendously from the versatility of the communications structure. For a highly accurate evaluation of the system, its performance is also estimated here for the kernels of important algorithms that were assigned to the New Millennium Computing Point Design groups, for the evaluation of their designs, during the PetaFLOPS Architecture Workshop (April 1996).

Such kernels are frequently encountered in scientific codes. In the following, let $n = 10^8$ and the arrays be appropriately dimensioned. The algorithmic kernels and their expected implementation on our system follow.

5.2.1. Algorithm I: Vector update or SAXPY loop

This algorithmic kernel is

do $i = 1, n$
  $c(i) = \pi * a(i) + b(i)$
enddo

where $\pi = 3.141592653589793$.

We assume double-precision (i.e., 8-byte) floating-point arithmetic for all the kernels. Also, the array data are uniformly distributed in the PE memories (i.e., each PE contains $\lceil 10^8/10368 \rceil = 9646$ elements), each arithmetic operation requires one clock cycle, and the CPU contains 10 floating-point units (FPUs), supports memory interleaving, fetches/stores ten 8-byte words from/to memory per cycle, and overlaps FPU operations with load/store operations. We can assume 10 FPUs in CPUs because of their 1 GHz clock and their 10 GFLOPS performance. All these assumptions conform to SIA projections. The execution time is given by

$$T^{\mathrm{I}} = \underbrace{2t_{\mathrm{m}} + t_{\mathrm{d}} + t_{\mathrm{c}}}_{\alpha} + \underbrace{2t_{\mathrm{m}} + t_{\mathrm{d}}}_{\beta} + \underbrace{\left(\left\lceil \frac{9646}{10} \right\rceil \times 2 - 1 \right) t_{\mathrm{d}}}_{\gamma} + \underbrace{t_{\mathrm{m}} + t_{\mathrm{d}}}_{\delta} + \underbrace{964 t_{\mathrm{d}}}_{\varepsilon},$$

where t_{m} the inverse of the inter-chip communications frequency, which is 1/375 MHz=2.66 ns [13], t_{c} the CPU clock-cycle time (i.e., clock period), which is 1/1 GHz=1 ns, t_{d} the memory clock-cycle time, which is 1/500 MHz=2 ns, α represents the time taken to fetch and decode the first instruction, β the time taken for the first set of 10 elements from a to reach the CPU, over the 10-word wide data bus. γ represents the time taken to fetch the remaining elements from a and b using memory interleaving (and loading 10 elements at a time due to the 640-bit data bus). These fetch operations fully overlap arithmetic CPU operations. δ represents the time taken by the group of the first 10 stores into the memory (these values first are stored in the local cache as they are produced and then written back into the memory when the data bus becomes available). ε represents the time taken to complete the remaining stores in groups of 10 each time using memory interleaving.

The amount of parallelism available in the program is 2×10^8 operations. The parallel execution time T^{I} is 5.80 μs and the execution rate (E-R) is 2×10^8 operations/5.80 μs = 34.48 Tera(FL)OPS (i.e., Tera-Operations per second). The degradation in E-R, compared to the peak performance of 103.68 TeraOPS, is due to the heavy amount of load/store operations; despite the application of memory interleaving, since the number of memory accesses exceeds the number of internal CPU operations by 50%, it has very significant effect. With prefetching into the data cache and leaving the results in the cache, the execution rate for this SAXPY loop approaches the peak rate.

5.2.2. Algorithm II: Large-stride vector fetch and store

This algorithmic kernel is

do $i = 1, n$
  $b(121 * i) = a(131313 * i)$
enddo

For Algorithm II, assume that each PE sends and receives $\lceil 10^8/10368 \rceil = 9646$ elements of the array a. We also assume no bulk-data transfers (i.e., no use of the DMA communications subsystem), simultaneous load/store by the CPU and the communications co-processor, and that each message contains an 8-byte address (i.e., 64 bits long) field and an 8-byte element from a (conforming to the format in Fig. 3). That is, a PE containing a value of a for a given i calculates the address in the global address space of the element $b(121 * i)$ and sends the value of $a(131313 * i)$ to the corresponding processor (this global address is contained in the address field of the message). The non-involvement of DMA controllers is the result of the irregular array stride and the rather small number of array elements per PE compared to the total number of PEs (with 9646 elements per PE and a total of 10 368 PEs, all source messages for a PE may be going to distinct PEs).

The amount of parallelism available is 10^8 operations. The total execution time is given by

$$T^{\mathrm{II}} = \underbrace{2t_{\mathrm{m}} + t_{\mathrm{d}} + t_{\mathrm{c}}}_{\alpha} + \underbrace{9646 \times (2t_{\mathrm{m}} + t_{\mathrm{d}})}_{\beta} + \underbrace{t_{\text{PE-to-remote\_memory}}}_{\gamma},$$

where α represents the time needed to fetch and decode the instruction, β the time needed to fetch the elements from the local memory individually (memory interleaving cannot be applied because of the irregular array stride), to calculate each time the address of the next element to fetch, and to prepare the messages by appending 64-bit destination addresses, all these operations are performed in parallel. γ represents the time needed to send the value of an element from a, in this case the last value, to a remote memory on another 8-PE card. It is derived using the values of t_{d}, t_{c}, and t_{m}. Previous data transfers overlap local-memory fetch operations.

The value of $t_{\text{PE-to-remote\_memory}}$ is 14.89 ns, and therefore $T^{\mathrm{II}} = 70.76$ μs. Also, the execution rate (E-R) is (10^8 elements × 8 Bytes/element)/70.76 μs = 11.30 TeraBS (i.e., Tera-Bytes per second). For the sake of comparison, the maximum network bandwidth for non-sequential memory accesses is obtained from Table 2 and is equal to 1.075 GigaBS/PE × 10 368 PEs = 11.14 TeraBS; however, the latter does not distinguish between addresses and data in the messages, and therefore the actual data bandwidth is 11.14/2 = 5.57 TeraBS for 64-bit data in 128-bit messages. The improvement of the E-R for this algorithm, when compared to the aforementioned maximum bandwidth, is due to the pipelining of multiple messages along the communication paths.

5.2.3. Algorithm III: Irregular gather/scatter

This algorithmic kernel is

/* This loop initializes i dx with pseudo-random number mod n. */
do $i = 1, n$
  i d$x(i) = mod(13131313 * i, n)$
enddo
/* Tested loop. */
do $i = 1, n$
  $b(i$ d$x(i)) = a(i)$
enddo

For Algorithm III, elements of the array a are accessed sequentially. Similarly to Algorithm II, however, the irregular stride for destination elements of the array b does not permit DMA transfers. Therefore, each PE fetches elements of the array a in groups of 10 from its local memory, but sends out only the value of one element at a time. The execution time is

$$T^{\mathrm{III}} = \underbrace{2t_{\mathrm{m}} + t_{\mathrm{d}} + t_{\mathrm{c}}}_{\alpha} + \underbrace{2t_{\mathrm{m}} + t_{\mathrm{d}}}_{\beta} + \underbrace{9645t_{\mathrm{m}}}_{\gamma} + t_{\text{PE-to-remote\_memory}},$$

where α represents the time needed to fetch and decode the tested-loop instruction, β the time needed to fetch the first set of 10 elements of a from the local memory (memory interleaving can be used because sequential elements are fetched). Simultaneously, destination addresses are produced (the implementation of the operations in the first loop is also included). γ represents the time needed for all but the last value to leave the PE. $t_{\text{PE-to-remote\_memory}}$ is the time needed for the last value to reach the memory of its corresponding destination PE.

This time is equal to 25.75 μs. It results in an execution rate for the tested loop of (10^8 elements × 8 Bytes/element)/25.75 μs = 31.06 TeraBS. Compared to Algorithm II, the much improved execution rate is the result of accessing sequential elements from the array a. In fact, this E-R is much better than the maximum of 5.57 TeraBS for non-sequential local-memory accesses, because of sequential accesses for a and the pipelining of messages along communication paths.

5.2.4. Algorithms IV and V: 3D Jacobi kernels

These algorithmic kernels are typical of many 3D physical modeling codes. They are

```
do k = 1, nz
  do j = 1, ny
    do i = 1, nx
      do m = 1, nc
        do mp = 1, nc
          a(i, j, k, m) = u(mp, m) *
            (b(i + 1, j, k, mp) + b(i − 1, j, k, mp)
            + b(i, j + 1, k, mp) + b(i, j − 1, k, mp)
            + b(i, j, k + 1, mp) + b(i, j, k − 1, mp))
            + a(i, j, k, m)
        enddo
      enddo
    enddo
  enddo
enddo
```

For Algorithm IV, we execute the 3D Jacobi kernel with $nc = 5$ and $nx = ny = nz = 1000$. For Algorithm V, we execute the 3D Jacobi kernel with $nc = 150$ and $nx = ny = nz = 100$.

Algorithms IV and V represent 3D grid Jacobi kernels with each (i, j, k) node being a space supernode with $nc \times nc$ interior nodes. This computation is a convolution-and-reduction operation applied for all values of mp for a given (i, j, k, m). We can observe the data reuse, for all values of m, in the terms multiplied by $u(mp, m)$. The corresponding sum of b terms is computed only once for each (i, j, k, mp) and is used in all iterations of m (i.e., nc times). The number of iterations involving all three "space" indices i, j, and k is larger than the number of PEs for both algorithms, so we distribute the corresponding three outer loops among all PEs. Therefore, each PE performs $\theta = \lceil nx \cdot ny \cdot nz / 10368 \rceil$ iterations involving these three outer loops. The number of these iterations for Algorithms IV and V is 96 451 and 97, respectively.

For a given (i, j, k), a PE performs the following operations:

- Five additions involving six elements from b for any given value of mp. Since mp assumes nc values, these are $5 \cdot nc$ additions. We assume that the results of these additions are stored in the local cache in order to be (re)used nc times.
- For a given value of (m, mp), one multiplication (involving $u(mp, m)$ and one of the aforementioned summation results) and one addition (involving the result of the multiplication and the previous value of $a(i, j, k, m)$). These operations are performed nc^2 times, the same as the total number of distinct values for the pair (m, mp).

Therefore, each PE performs a total of $(5 \cdot nc + nc^2)\theta$ additions and $nc^2 \cdot \theta$ multiplications. We can assume that the values of b, u, and a are prefetched into the cache. The values of elements from b do not change during the computation and data transfers between processors fully overlap computations. To better facilitate these data transfers, it is wise to choose a 3D grid mapping of the arrays a and b onto the processors. This mapping can be implemented in an optimal manner because of the GH's rich interconnection network. For example, we can view our generalized hypercube of 36^2 (i.e., 1296) 8-PE cards as a logical 3D cube of size $16 \times 9 \times 9$. We partition the 3D (i, j, k) grid for mapping onto the logical 3D cube. Only local data transfers between adjacent *GH* nodes or PEs on the same card are then needed to get elements of b corresponding to

offsets of ±1 in the dimensions traversed by i, j, and k. However, these data transfers are carried out transparently because of predictable data prefetches using DMA control and a very large number of CPU computations before external data is used, and therefore they do not contribute to the total execution time. It is easy to see that arithmetic operations fully overlap operations that store the results into the local memory (the results may first be stored into the local cache), except for the last set of 10 stores. The execution time of each algorithm is then given by

$$T^{\text{IV or V}} = \underbrace{2t_{\text{m}} + t_{\text{d}} + t_{\text{c}}}_{\alpha} + \underbrace{(5nc + 2nc^2)\left\lceil \frac{\theta}{10} \right\rceil t_{\text{c}} + t_{\text{d}}}_{\beta},$$

where α represents fetch and decode time for the first instruction, and β the total execution time for arithmetic operations by each PE. The denominator is the speedup resulting from the 10 FPUs in each CPU. There is no overhead for fetching data because of prefetching them into the cache. The last term represents the time for storing the last 10 results (the actual number of results is the remainder of $nc^2/10$ if different from 0, or 10 otherwise, corresponding to the m and mp loops; it is 5 and 10 for Algorithms IV and V, respectively). t_{m} is not added because of memory interleaving.

Because of the 10 FPUs in each PE and the fact that $nc = 5$ for Algorithm IV, operations corresponding to two consecutive values for the index m are performed each time. The execution times are $T^{\text{IV}} = 723.46\,\mu\text{s}$ and $T^{\text{V}} = 457.51\,\mu\text{s}$. The amount of parallelism available in Algorithms IV and V is given by $nx{\cdot}ny{\cdot}nz(5{\cdot}nc + 2{\cdot}nc^2)$. It is 75×10^9 and 45.75×10^9 operations for Algorithms IV and V, respectively. Therefore, the execution rates for Algorithms IV and V are 103.66 TeraOPS and 100.00 TeraOPS, respectively. Both execution rates are close to the peak rate of 103.68 TeraOPS because of data prefetching and fully overlapping computations and communications.

The expected execution times for all algorithmic kernels are summarized in Table 3. These results further prove the suitability of our case-study system for 100 Tera(FL)OPS performance.

Table 3
Performance results for algorithmic kernels (Algorithm I: SAXPY loop; Algorithm II: large-stride vector fetch and store; Algorithm III: irregular gather/scatter; Algorithm IV: Jacobi 1; Algorithm V: Jacobi 2)

Algorithm	Execution time (μs)	Execution rate
Algorithm I	5.80	34.48 TeraOPS
Algorithm II	70.76	11.30 TeraBS
Algorithm III	25.75	31.06 TeraBS
Algorithm IV	723.46	103.66 TeraOPS
Algorithm V	457.51	100.00 TeraOPS

5.2.5. Further performance results for altered Jacobi kernels: Algorithms VI and VII

We have altered the Jacobi kernel of the preceding subsection to heavily test the vector and, primarily, the communications capabilities of our design. The new algorithm is

```
do k = 1, nz /* dimension z */
  do j = 1, ny /* dimension y*/
    it do i = 1, nx /* dimension x*/
      do m = 1, nc /* iteration number */
        do mp = 1, nc /* different species */
          b(i, j, k, mp) = u(mp, m) *
            (b(i + 1, j, k, mp) + b(i − 1, j, k, mp)
            +b(i, j +1, k, mp)+b(i, j −1, k, mp)
            +b(i, j, k+1, mp)+b(i, j, k−1, mp))
            + a(i, j, k, mp)
        enddo
      enddo
    enddo
  enddo
enddo
```

For Algorithm VI, we execute the altered 3D Jacobi kernel with $nc = 5$ and $N = nx = ny = nz = 1000$. For Algorithm VII, we execute the kernel with $nc = 150$ and $N = nx = ny = nz = 100$. We cannot assume data reuse and heavy prefetching with these new kernels because values for the array b are produced continuously. The new Jacobi kernel requires a tremendous amount of communication operations. Every innermost iteration requires new data transfers among PEs.

For Algorithm VI, we restructure the loops j and i after loop exchange in order to take advantage of the vector capabilities (i.e., via memory interleaving) of our design. Assume the 2D $N \times N$ array corresponding to these two indices, as shown in Fig. 9 (I and J replace i and j, respectively).

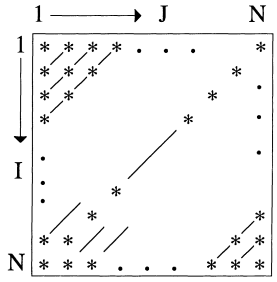

Fig. 9. The 2D $N \times N$ array corresponding to the loops j and i in Algorithm VI.

Assume the following parameters:

- NDIAGS $= 2N - 1$: the number of diagonals.
- LDIAG: the length of a diagonal.
- (I, J): indices for traversing diagonals.
- (ISTART,JSTART): starting indices of diagonals.

The average vector length of a diagonal is given by

$$\frac{\text{number of computed points}}{\text{number of diagonals}} = \frac{N^2}{2N-1} \simeq \frac{N}{2}$$

The new Jacobi kernel after loop exchange becomes

```
ISTART=0
JSTART=1
LDIAG=0
do k = 1, N /* divided among ... */
  do mp = 1, nc /*  ... the processors */
    do m = 1, nc /* serial loop */
      do 41 IDIAGS=1,NDIAGS
        if (IDIAGS.LE.N) then /* for upper left */
          ISTART=ISTART+1
          LDIAG=LDIAG+1
        else /* for lower right */
          JSTART=JSTART+1
          LDIAG=LDIAG−1
        endif
        I=ISTART+1
        J=JSTART−1
        do 42 IPOINT=1, LDIAG /* vectorizes */
          I=I−1
          J=J+1
          b(I,J,k,mp)=...
          42 continue
        41 continue
      enddo
    enddo
  enddo
```

The loops m and IDIAGS are serial. The loop IPOINT vectorizes, so vectoring is applied when accessing the 2D array $b(., ., k, mp)$ diagonally. We distribute the outermost loops k and mp among the PEs. However, a subset of the PEs are used. More specifically, we use $max\_k \cdot max\_mp = 5000$ Pes, out of the 10 368 PEs which are available. The arrays $b(., ., k-1, mp)$ and $b(., ., k+1, mp)$ are received from two other PEs at the beginning of each iteration of the loop m and the chosen 5000 PEs are synchronized. Because of the transmission of these large arrays, the DMA controllers of the PEs are employed. Each PE, except for those corresponding to $k = 1$ and $k = max\_k = N$, transmit their array $b(., ., k, mp)$ to both neighbors in dimension k corresponding to $k - 1$ and $k + 1$.

The total execution time is given by

$$T^{\mathrm{VI}} = t^{\mathrm{VI}}_{\mathrm{comm}} + t^{\mathrm{VI}}_{\mathrm{comp}},$$

where the two terms represent total communication and computation times, respectively. The time $7t_c\lceil nx \cdot ny/10\rceil = 700\,\mu$s for arithmetic operations in each iteration of m is fully overlapped by all local memory accesses (because of memory interleaving with vectoring, new data is accessed in $1/t_{\mathrm{d}} = 2$ ns, while the CPU operates in $1/t_{\mathrm{c}} = 1$ ns). However, these arithmetic operations and local-memory accesses are, in turn, fully overlapped by DMA transfers that fetch data in advance, as they are produced, for the next iteration. Therefore, the only component of the computation time that is not overlapped by DMA transfers is given by

$$t^{\mathrm{VI}}_{\mathrm{comp}} = max\_m(2t_{\mathrm{m}} + t_{\mathrm{d}}) = 0.03\,\mu\mathrm{s}$$

and corresponds to the time needed to fetch the first set of fresh data from the memory in each step, for a total of $max\_m = nc$ steps.

With 64-bit elements and DMA transfers, we have the very good approximation

$$\begin{aligned} t^{\mathrm{VI}}_{\mathrm{comm}} &= max\_m \\ &\quad \times \frac{(nx \cdot ny)\text{ elements} \times 8\,\text{Bytes/element}}{6\,\text{GBytes/s(via DMA)}} \\ &= 6.66\,\text{ms}. \end{aligned}$$

Therefore, $T^{\mathrm{VI}} = 6.66$ ms. The amount of parallelism is $7 \cdot nx \cdot ny \cdot nz \cdot nc \cdot nc$ or 175×10^9 operations, and therefore the execution rate is 26.27 TeraOPS. This is a very impressive execution rate considering the heavy

Table 4
Performance results for the altered Jacobi kernels (Algorithm VI: altered Jacobi 1; Algorithm VII: altered Jacobi 2)

Algorithm	Execution time	Execution rate
Algorithm VI	6.66 ms	26.27 TeraOPS (with 5000 PEs)
Algorithm VII	481 μs	95.73 TeraOPS (with 10 000 PEs)

amount of communications and the fact that less than half of the PEs are used.

For Algorithm VII, we parallelize the loops k and j, vectorize the loop mp, and the loop i remains serial. Therefore, we use $max\_k \cdot max\_j = N^2 = 100^2 = 10\,000$ PEs. Each PE corresponds to a specific value for the pair (k, j) and contains the elements $b(i, j, k, mp)$ for $i = 1, 2, \ldots, 100$ and $mp = 1, 2, \ldots, 150$. For each $b(i, j, k, mp)$, the PE receives the values of the four elements $b(i, j, k-1, mp)$, $b(i, j, k+1, mp)$, $b(i, j-1, k, mp)$, and $b(i, j+1, k, mp)$; these values are to be used in the next iteration of i. The new values are sent as they are produced using DMA control. $max\_mp = nc = 150$ values are sent between given pairs of PEs. DMA control during each iteration i then consumes time equal to $t^{\mathrm{VII}}_{i,\mathrm{DMA}} = (nc \text{ elements} \times 8 \text{ Bytes/element})/6 \text{ GBytes/s} = 0.2\ \mu\text{s}$. Again, we have data reuse for the summations of b elements for various values of m. The computation time for each iteration i is equal to

$$t^{\mathrm{VII}}_{i,\mathrm{comp}} = (2t_{\mathrm{m}} + t_{\mathrm{d}}) + 7\left\lceil \frac{max\_mp}{10} \right\rceil t_{\mathrm{c}} + 2 \cdot max\_m \left\lceil \frac{max\_mp}{10} \right\rceil t_{\mathrm{c}} = 4.61\ \mu\text{s}.$$

The first term is the time required to fetch the first set of 10 values from the local memory; memory interleaving is then applied, resulting in full overlap of computations and subsequent local-memory accesses. The second term is the time required to calculate the summations of b terms for all values of mp, while the third term is the time needed for two arithmetic operations (i.e., a multiplication and an addition) per mp value. An upper bound on the total execution time is $T^{\mathrm{VII}} = max\_i\,(t^{\mathrm{VII}}_{i,\mathrm{comp}} + t^{\mathrm{VII}}_{i,\mathrm{DMA}}) = 481\ \mu\text{s}$.

The amount of parallelism is $nx \cdot ny \cdot nz(7 \cdot nc + 2 \cdot nc^2)$ or 4.605×10^{10} operations. Therefore, the execution rate is 95.73 TeraOPS, which is very close to the peak performance for 10 000 PEs. The actual execution rate is even better considering that potentially there exists higher overlap between computations and DMA transfers. Table 4 summarizes the results for the altered Jacobi kernels.

6. Conclusions

We have proven in this paper the suitability of our "point design" for very high performance computing. The complete system is characterized by immense bisection bandwidth and other outstanding properties. Not only can our proposed system graciously achieve its performance objective, but also its dramatically low interconnect complexity renders it viable. Such a dramatic reduction in the system interconnect complexity is not possible with any other existing or expected technology. Performance results for important algorithmic kernels were also employed to further support our claim for outstanding performance.

References

[1] K. Aly, P. Dowd, Parallel computer reconfigurablility through optical interconnects, in: International Conference on Parallel Processing, 1992, pp. I:105–137.
[2] L.N. Bhuyan, D.P. Agrawal, Generalized hypercube and hyperbus structures for a computer network, IEEE Trans. Comput. 33 (4) (1984) 323–333.
[3] L. Camp, et al., Guided-wave and free space optical interconnect for parallel processing systems: a comparison, Appl. Opt. 33 (1994) 6168–6180.
[4] A.L. Cox, et al., Software versus hardware shared-memory implementation: a case study, in: International Symposium on Computing Architecture, 1994, pp. 106–117.
[5] W.J. Dally, C.L. Seitz, The torus routing chip, Distrib. Comput. 1 (1986) 187–196.
[6] W.J. Dally, Network and processor architecture for message-driven computers, in: R. Suaya, G. Birtwistle (Eds.), VLSI and Parallel Computation, Morgan Kaufmann, Los Altos, CA, 1990, pp. 140–222.
[7] P. Fragopoulou, S.G. Akl, H. Meijer, Optimal communication primitives on the generalized hypercube network, J. Parallel Distrib. Comput. 32 (1996) 173–187.
[8] E. Frietman, et al., Parallel optical interconnects: implementation of optoelectronics in multiprocessors architecture, Appl. Opt. 29 (1990).
[9] Findings and Recommendations, PetaFLOPS Software Summer Study (PetaSoft'96), June 1996.
[10] M. Kajita, K. Kasahara, T.J. Kim, I. Ogura, I. Redmond, E. Schenfeld, Free-space wavelength division multiplexing optical interconnections for massively parallel processing systems, in: Int'l. Top. Meet. Opt. Comput. 1996, pp. 24–25.

[11] D. Lenoski, et al., The stanford FLASH multiprocessor, IEEE Comput. 3 (1992) 63–79.
[12] X. Li, S.G. Ziavras, C.N. Manikopoulos, Parallel DSP algorithms on turboNet: an experimental hybrid message-passing/shared-memory architecture, Conc.: Pract. Exp. 8 (5) (1996) 387–411.
[13] The National Technology Roadmap for Semiconductors, Semiconductor Industry Association (SIA), 1994.
[14] D.A. Patterson, Teraflops design in eight easy steps, Sci. Am. (January 1991) 104–105.
[15] M.C. Pease III, The indirect binary *n*-cube microprocessor array, IEEE Trans. Comput. C-26 (5) (1977) 458–473.
[16] C. Qiao, R. Melhem, Reducing communication latency with path multiplexing in optically interconnected multiprocessor systems, IEEE Trans. Parallel Distrib. Syst. 8 (2) (1997) 97–108.
[17] C.L. Seitz, Concurrent VLSI architectures, IEEE Trans. Comput. C-33 (12) (1984) 1247–1265.
[18] J. Senior, Optical Fiber Communication, 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1992.
[19] T. Seshita, Y. Ikeda, H. Wakimoto, K. Ishida, T. Terada, T. Matsunaga, T. Suzuki, Y. Kitaura, N. Uchitomi, A 20 GHz 8 bit multiplexer IC implemented with 0.5μm WNx/W-Gate GaAs MESFET's, IEEE J. Solid-State Circ. 29 (12) (1994) 1583–1587.
[20] M. Soda, H. Tezuka, F. Sato, T. Hashimoto, S. Nakamura, T. Tatsumi, T. Suzaki, T. Tashiro, Si-Analog IC's for 20 Gb/s optical receiver, IEEE J. Solid-State Circ. 29 (12) (1994) 1577–1581.
[21] T. Szymanski, Hypermeshes: optical interconnection networks for parallel computing, J. Parallel Distrib. Comput. 26 (1995) 1–23.
[22] S.C. Tsay, H. Grebel, Design of transverse holographic optical interconnect, Appl. Opt. 33 (1994) 6747–6752.
[23] A. Yariv, Optical Electronics, 4th Edition, Saunders, Belmont, CA, 1991.
[24] L.D. Wittie, Communication structures for large networks of multicomputers, IEEE Trans. Comput. C-30 (4) (1981).
[25] S.G. Ziavras, On the problem of expanding hypercube-based systems, J. Parallel Distrib. Comput. 16 (1) (1992) 41–53.
[26] S.G. Ziavras, RH: a versatile family of reduced hypercube interconnection networks, IEEE Trans. Parallel Distrib. Syst. 5 (11) (1994) 1210–1220.
[27] S.G. Ziavras, Generalized reduced hypercube interconnection networks for massively parallel computers, in: D.F. Hsu, A. Rosenberg, D. Sotteau (Eds.), Networks for Parallel Computations, American Mathematical Society, Providence, RI, 1995, pp. 307–325.
[28] S.G. Ziavras, Scalable multifolded hypercubes for versatile parallel computers, Parallel Process. Lett. 5 (2) (1995) 241–250.
[29] S.G. Ziavras, A. Mukherjee, Data broadcasting and reduction, prefix computation, and sorting on reduced hypercube parallel computers, Parallel Comput. 22 (1996) 595–606.
[30] S.G. Ziavras, H. Grebel, A.T. Chronopoulos, A low-complexity parallel system for gracious, scalable performance. Case study for near PetaFLOPS computing, in: Proceedings of the Sixth Symposium on Front. Mass. Parallel Computations, Special Session New Millennium Computing Point Designs, 1996, pp. 363–370.
[31] S.G. Ziavras, H. Grebel, A.T. Chronopoulos, A scalable/feasible parallel computer implementing electronic and optical interconnections for 156 TeraOPS minimum performance, in: PetaFLOPS Arch. Workshop, 1996, pp. 179–209.
[32] S.G. Ziavras, Investigation of various mesh architectures with broadcast buses for high-performance computing, VLSI Des. 9 (1) (1999) 29–54.

Dr. Sotirios G. Ziavras received the Diploma in EE from the National Technical University of Athens, Greece, in 1984, the MSc in ECE from Ohio University in 1985, and the DSc in Computer Science from George Washington University in 1990, where he was a Distinguished Graduate Teaching Assistant and a Graduate Research Assistant. From 1988 to 1989, he was also with the Center for Automation Research at the University of Maryland, College Park. He was a Visiting Assistant Professor in the ECE Department at George Mason University in Spring 1990. He is currently an Associate Professor in the ECE and CIS Departments at NJIT. His research interests are unconventional processor and computer systems designs, and parallel computing systems and algorithms.

Dr. Haim Grebel is a Professor of Electrical and Computer Engineering at NJIT. His research interests are in the area of linear and non-linear optics which are applicable to optoelectronics networks and devices.

Dr. Anthony T. Chronopoulos received his PhD at the University of Illinois in Urbana-Champaign in 1987, in Computer Science. He is an IEEE senior member. He has performed research under 10 different research grants, and has given over 50 conference and invited research lectures. He has advised over 10 graduate students. He has co-authored 50 refereed journal papers and conference proceedings papers in the areas of high performance computing, distributed systems and applications and computational science. He has been awarded four NSF grants.

Mr. Florent Marcelli was an exchange student at NJIT in 1997.