Feasibility Study of Scaling an XMT Many-Core

Sean O'Brien*†, Uzi Vishkin*†, James Edwards*†, Edo Waks†, Bao Yang‡
* *University of Maryland Institute for Advanced Computer Studies (UMIACS), and*
† *Electrical and Computer Engineering Department*
‡ *Mechanical Engineering Department*
College Park, Maryland, USA
seobrien@umd.edu, vishkin@umd.edu, jedward5@umd.edu, edowaks@umd.edu, baoyang@umd.edu

***Abstract*—The reason for recent focus on communication avoidance is that high rates of data movement become infeasible due to excessive power dissipation. However, shifting the responsibility of minimizing data movement to the parallel algorithm designer comes at significant costs to programmer's productivity, as well as: (i) reduced speedups and (ii) the risk of repelling application developers from adopting parallelism.**

The UMD Explicit Multi-Threading (XMT) framework has demonstrated advantages on ease of parallel programming through its support of PRAM-like programming, combined with strong, often unprecedented speedups. Such programming and speedups involve considerable data movement between processors and shared memory. Another reason that XMT is a good test case for a study of data movement is that XMT permits isolation and direct study of most of its data movement (and its power dissipation).

Our new results demonstrate that an XMT single-chip many-core processor with tens of thousands of cores and a high throughput network on chip is thermally feasible, though at some cost. This leads to a perhaps game-changing outcome: instead of imposing upfront strict restrictions on data movement, as advocated in a recent report from the National Academies, opt for due diligence that accounts for the full impact on cost. For example, does the increased cost due to communication avoidance (including programmer's productivity, reduced speedups and desertion risk) indeed offset the cost of the solution we present?

More specifically, we investigate in this paper the design of an XMT many-core for 3D VLSI with microfluidic cooling. We used state-of-the-art simulation tools to model the power and thermal properties of such an architecture with 8k to 64k lightweight cores, requiring between 2 and 8 silicon layers. Inter-chip communication using silicon compatible photonics is also considered. We found that, with the use of microfluidic cooling, power dissipation becomes a cost issue rather than a feasibility constraint.

Robustness of the results is also discussed.

***Keywords*-data movement, parallel algorithms, PRAM, parallel architecture, many-core, 3D VLSI, microfluidic cooling**

I. Introduction

Approaching the end of the so-called Dennard scaling is an important concern as it implies decreasing improvement in power consumption of computers. This concern has led to a remarkable consensus: communication avoidance must drive both the design of computer systems and their programming. Salient examples of this consensus include the report [25], and the extensive work done by Jim Demmel's group at UC-Berkeley [10] initially on communication avoidance upper bounds, and later also on lower bounds.

The impact of this consensus has been quite dramatic. The viewpoint article [35] argued that it has led industry to prioritize energy saving over programmer's productivity to the point of derailing the promise of shifting the general-purpose serial computing paradigm to a general-purpose parallel computing based on many-core hardware. While current-day parallel architectures allow good speedups on some regular applications that can be programmed to use limited communication bandwidth, such as dense-matrix type programs, these architectures are mostly handicapped on other programs: high communication-bandwidth programs, "irregular" programs, or when seeking "strong scaling." Strong scaling is the ability to translate an increase in the number of cores to faster run time for problems of fixed input size. Overall, the use of multicore parallelism for speeding up completion time of a single computational task has been quite limited. On a more principled level, this consensus seems to "go against history." Much of the progress attributed to the Industrial Revolution is due to using more power for reducing human effort. The consensus seems to seek progress in computing by using human programmer effort in order to save power.

The current paper makes two main contributions: a *technical insight* and a *higher-level one*.

The technical insight: Even for an architecture example that does not seek to save on data movement (henceforth, data movement would often mean the opposite of communication avoidance), it would be feasible to accommodate the thermal behavior of highly parallel programs that require much data movement, assuming the use of modern 3D-VLSI technology and microfluidic cooling. The conclusion is that *a more balanced approach* than categorically avoiding data movement at all costs is feasible. Namely,

A reality of trade-offs among different costs: Each of the following resources involves cost: 1. Power consumption. 2. Integration of cooling for mitigating power consumption and dissipation. 3. Lowering programmer's productivity for communication avoidance. 4. Lower performance due to communication avoidance. Thus, putting communication avoidance ahead of other cost considerations has been an arbitrary decision. It will be prudent for all players in the many-core space to consider various alternatives for both costs and benefit, and, in particular, feel free to question the above consensus since contrary to prior understanding, our paper demonstrates that other alternatives are also feasible. Hence, the choice among the various alternatives should be based on cost and benefit.

A. Literature (Summary)

For background on the cooling technology discussed in this paper, see [5]. The use of cooling for alleviating heat dissipation is, of course, not new. More specifically, the use of microfluidic cooling for 3D-VLSI co-design of multicore architectures has been the topic of recent work [36]. However, the current paper appears to be the first to draw a direct connection of such cooling for enabling

data movement, and, in particular, for overcoming the stifling of parallel programming due to communication avoidance.

Some broader perspective follows. The approaching end of Dennard scaling brought about the recognition that to maintain high levels of performance growth something must change in the reliance on traditional VLSI technologies. Photonics provides an appealing advantage, as it can move enormous amounts of data across a great distance using low power. Indeed, the advantage of silicon photonics for communication across distances that are not too small (e.g., one meter) is already well recognized. Recent work, summarized in [6], has suggested drastic conversions to silicon photonic technologies, where even communication on chip will be photonics-based. However: given 1. the large investment in current VLSI methods, 2. the advantage electronics has over photonics for switching, 3. the rather immature state of using photonics for on-chip communication, and, last, but not least, 4. our parallel (PRAM) algorithms motivation to enable high-bandwidth low latency fine-grained irregular communication on-chip and off-chip; it is only natural that we are looking for a proper hybrid of technologies that will allow mitigating the data movement problem for parallel algorithms. Our most modest objective was already quite ambitious: to at least demonstrate that such mitigation is feasible contrary to common wisdom. Towards that end, our main insight is that microfluidic cooling allowed us to pull together the remaining pieces of the puzzle.

Using silicon photonics for power-efficient scaling of multi-chip systems has also received attention, e.g., [22]. However, they do not use microfluidic cooling and 3D-VLSI, which when coupled with silicon photonics yield improved bandwidth and latency that, in turn, facilitate scaling of PRAM-like programming.

II. Background

A. XMT Architecture

The Explicit Multi-Threading (XMT[1]) general-purpose architecture [37] is a many-core architecture which aims to improve single-task completion time and ease-of-programming for parallel applications by supporting Parallel Random Access Model (PRAM) programming [18], [21]. For some advantages of XMT, see section V-B.

The XMT processor includes a master thread control unit (MTCU); processing clusters, each comprising several light-weight thread-control units (TCUs); a high-bandwidth low-latency interconnection network; memory modules (MM), each comprising on-chip cache and off-chip memory; prefix-sum (PS) unit(s); and global registers. The shared-memory-modules block (bottom left of fig. 1) suppresses the sharing of a memory controller by several MMs. The processor alternates between serial mode (in which only the MTCU is active) and parallel mode. The MTCU has a standard private data cache (used in serial mode) and a standard instruction cache. The TCUs, which lack a write data cache, share the MMs with the MTCU.

The overall XMT design is guided by a general design ideal we call no-busy-wait finite-state-machines, or NBW FSM, meaning the FSMs, including processors, memories, functional units, and interconnection networks comprising the parallel machine, never cause

[1]XMT at the University of Maryland, not to be confused with the Cray XMT

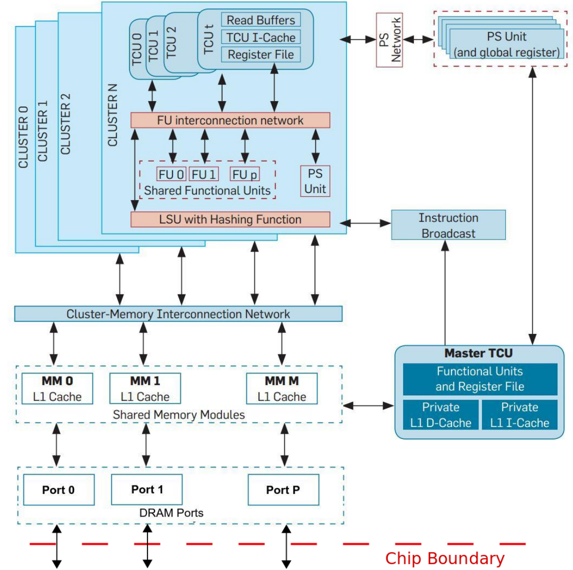

Figure 1. Block Diagram of the XMT Architecture

one another to busy-wait. It is ideal because no parallel machine can operate that way. Nontrivial parallel processing demands the exchange of results among FSMs. The NBW FSM ideal represents our aspiration to minimize busy-waits among the various FSMs comprising a machine.

We cite the example of how the MTCU orchestrates the TCUs to demonstrate the NBW FSM ideal. The MTCU is an advanced serial microprocessor that also executes XMT instructions (such as spawn and join). Typical program execution flow can also be extended through nesting of sspawn commands. The MTCU uses the following XMT extension to the standard von Neumann apparatus of the program counters and stored program. Upon encountering a spawn command the MTCU broadcasts the instructions in the parallel section starting with that spawn command and ending with a join command on a bus connecting to all TCU clusters. The largest ID number of a thread the current spawn command must execute (Y) is also broadcast to all TCUs. The largest ID (index) of the executing threads is stored in a global register X. In parallel mode, a TCU executes one thread at a time. Executing a thread to completion (upon reaching a join command), the TCU does a prefix-sum using the PS unit to increment global register X. In response, the TCU gets the ID of the thread it could execute next; if the ID is $\leq$Y, the TCU executes a thread with this ID. Otherwise, the TCU reports to the MTCU that it finished executing. When all TCUs report they have finished, the MTCU continues in serial mode. The broadcast operation is essential to the XMT ability to start all TCUs at once in the same time it takes to start one TCU. The PS unit allows allocation of new threads to the TCUs that just became available within the same time as allocating one thread to one TCU. This dynamic allocation provides run-time load-balancing of threads coming from an XMTC program.

We are now ready to connect with the NBW FSM ideal. From the moment the MTCU starts executing a spawn command until each TCU terminates the threads allocated to it, no TCU can cause any

Table I
XMT ARCHITECTURE CONFIGURATIONS

	8k	16k	32k	64k
TCUs	8192	16384	32768	65536
Clusters	256	512	1024	2048
Memory Modules	256	512	1024	2048
NoC MoT Levels	16	12	10	8
NoC Butterfly Levels	0	3	5	7
TCUs per Cluster		32		
ALUs per Cluser		32		
MDUs per Cluster		1		
FPUs per Cluster		1		
LSUs per Cluster		1		

other TCU to busy-wait for it. An unavoidable busy-wait ultimately occurs when a TCU terminates and begins waiting for the next spawn command.

TCUs, with their own local registers, are simple in-order pipelines, including fetch, decode, execute/memory-access, and write-back stages. A cluster includes functional units shared by several TCUs and one load/store port to the interconnection network shared by all its TCUs. In this paper, we consider XMT configurations with 8k, 16k, 32k, and 64k TCUs, grouped into clusters of 32 TCUs, with each cluster sharing a single load/store unit (LSU), multiply/divide unit (MDU), floating point unit (FPU), and read only cache. See table I for a summary of the XMT configurations considered for this paper.

The global memory address space is evenly partitioned into the MMs through a form of hashing. The XMT design eliminates the cache-coherence problem, a challenge in terms of bandwidth and scalability. In principle, there are no local caches at the TCUs. Within each MM, the order of operations to the same memory location is preserved.

Quite a few performance enhancements have been incorporated into the XMT hardware, including compiler and run-time scheduling methods for nested parallelism and prefetching methods.

B. NoC (Network on Chip)

The high-throughput interconnection network required for the XMT architecture presents an implementation challenge. A unique data path can be provided for each pair of clusters and cache modules, such that there is no blocking in the network, using a mesh of trees (MoT) network. However, the number of switches required is proportional the product of the number of clusters and the number of cache modules, which translates to a large silicon area. For example, an XMT architecture in 22 nm technology with 8k TCUs requires silicon area of 190 mm^2 just for an MoT NoC. The area required for an MoT NoC of an XMT architecture with 16k TCUs is 760 mm^2, and would not fit on a single silicon layer. In order to reduce network area, a hybrid MoT and butterfly network can be used, where the inner levels of the "pure" MoT network are replaced with butterfly levels [2]. See fig. 2. In this paper, we model a duplicated interconnection network, so data traveling from clusters to caches will not interfere with data traveling in the opposite direction. The interconnection network is duplicated because we found that a single silicon layer could contain duplicated networks with the number of MoT levels chosen; a single network did not offer more MoT levels or any other advantage.

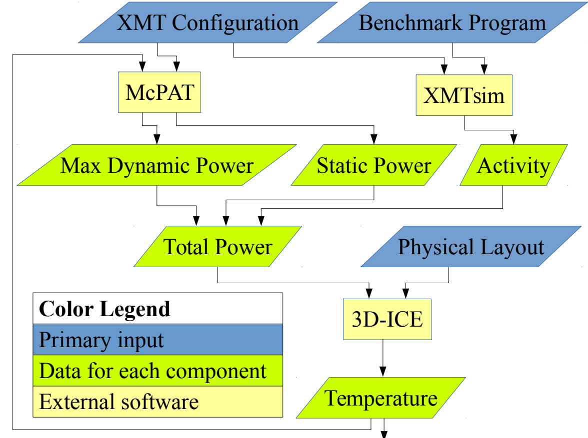

Figure 3. Simulation workflow with temperature feedback loop

III. EXPERIMENTAL METHODOLOGY

Three software tools were used to simulate an XMT system on a 3D integrated circuit (IC) (fig. 3).

- XMTSim [19], [20]: A cycle-accurate simulator of the XMT architecture, which can simulate many configurations of XMT and reports component activity data.
- McPAT [23]: A framework for modeling power, area, and timing of many-core architectures. For this paper, we use it to model power.
- 3D-ICE [30], [31]: Performs thermal modeling for 3D ICs with microfluidic channels.

Because the leakage power for the XMT components depends on temperature, and the temperature of components depends on the dissipated power, the simulation workflow incorporates a feedback loop. In the first iteration, some baseline temperature profile is assumed and used in McPAT to determine the power dissipation of the components. In later iterations, the temperature profile generated by 3D-ICE is used, and the process is repeated as long as the difference between the temperature profile of the final and preceding iterations is larger than some threshold.

A. Activity

We model the activity of the XMT system using the cycle-accurate simulator XMTSim. Given a description for an XMT architecture and a benchmark program, XMTSim will report the activity profile of all components. For the TCU clusters, we consider the activity in the TCUs, register files, instruction caches, ALUs, floating point units, multipy/divide units, read only caches, and load/store units. For the memory modules, we consider the activity of the caches and memory controllers.

A limitation of this study is that the activity level is not tracked on a per-TCU basis. Instead, the average activity level across all TCUs is recorded, and this is used as the activity level for each individual TCU. Additionally, the activity levels are averaged over intervals of 5000 clock cycles and the moment of peak activity

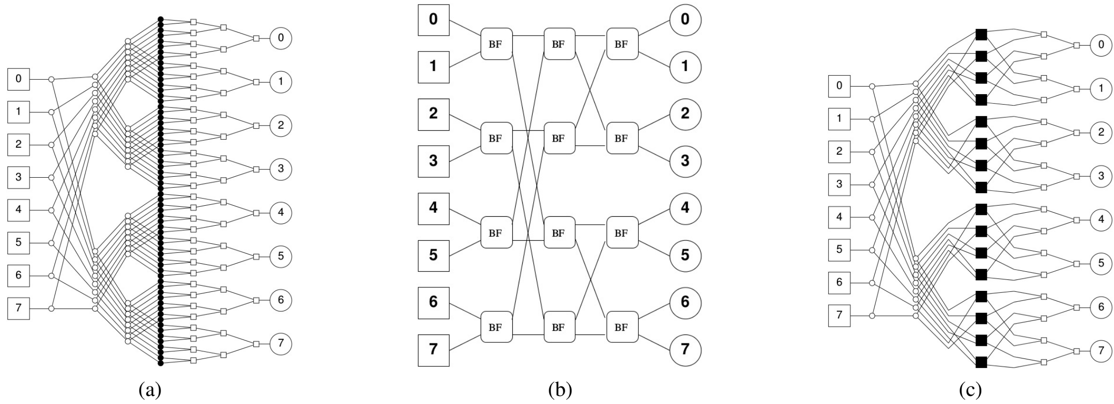

Figure 2. (a) 8-terminal MoT network. (b) 8-terminal butterfly network. (c) 8-terminal MoT / butterfly hybrid network, with one level of butterfly switches.

is used as the basis for power and temperature calculations in the later stages of the analysis pipeline. The load balancing used in XMT, per section II-A, ensures that deviation from average levels will be minimal when applications provide sufficient parallelism, as in the benchmarks evaluated. Using the peak activity level over 5000 clock cycles provides a conservative estimate of the maximum steady-state power dissipation.

B. Power Dissipation

Using McPAT (for Multicore Power, Area, and Timing), we estimate the leakage and maximum dynamic power dissipation of the XMT components, for temperatures between 27 °C and 127 °C. For all configurations we use a 22 nm technology node, 3000 MHz clock frequency, and 0.8 V gate voltage. McPAT cannot directly estimate the power dissipation of the NoC in the MoT, butterfly, or hybrid arrangements. Therefore, we use as a baseline the power profile of a 90 nm ASIC prototype [4] for an 8-terminal pure MoT network. We use the number of switch primitives to scale to the larger NoC configurations, and account for the change in clock frequency, voltage, and technology node using McPAT. Because McPAT only estimates power at intervals of 10 °C, we interpolate intermediate values.

Using the activity profile reported by XMTSim, we calculate the total power of a component as:

$$P = P_{dyn,max} * ACT * CF + P_{dyn,max} * (1 - CF) + P_{leak}$$

$P_{dyn,max}$ and P_{leak} are the dynamic and leakage power values reported by McPAT for each subcomponent of the TCUs, ACT is the activity level reported by XMTsim for each subcomponent, and CF is the activity correlation factor. In this paper we use an activity correlation factor of 0.9, which is the value that Wattch power simulator uses [8]. The activity correlation factor describes the proportion of the maximum dynamic power which is dissipated when the subcomponent is not in use.

C. Physical Model

We use an ASIC implementation of a 64-TCU XMT prototype and an ASIC implementation of an 8-terminal MoT NoC [3], fabricated in 90 nm technology, as the basis for area estimation. The area used in these prototypes was scaled quadratically to 22 nm technology with a safety factor of 30%, yielding a final area scaling factor of $\left(\frac{22}{90}\right)^2 * 1.3 = 0.078$. The area of the NoC is further scaled to account for the increased wire complexity and register count using the equations in chapters 4.7 and 7.3 of [4]. For example, the register count of the hybrid MoT and butterfly network is given by $R = 6N(N/2^h - 1) + (N/2^h)^2 * 2h * 2^h$, where N is the number of terminals on either side of the network, and h is the number of butterfly stages.

We limit the NoC to one silicon layer. This is necessary, at least for MoT, because the central levels of the interconnection network require hundreds of thousands of wire connections or more, depending on the number of MoT levels. To allow a multilevel NoC, these connections would need to cross silicon layers by the use of Through Silicon Vias (TSVs). A practical limit to the number of TSVs on a single layer may be ten thousand [34], as beyond this point manufacturing cost quickly increases and total TSV footprint becomes a significant percentage of silicon area. Multiplexing signals across TSVs is possible, but this can only increase the number of signals by a factor of 3 [33], not enough to enable a useful multi-layer NoC.

We limited the silicon footprint of each layer to a maximum of 400 mm^2. Given this constraint, we fit each configuration into as few silicon layers as possible. Then we reduce the length of the silicon footprint, leaving the width at 20 mm. In this manner we maximize the number of (cooling) microchannels per layer and reduce their length. Reducing the length of the microchannels provides two benefits: (1) it allows us to increase fluid velocity while maintaining a constant pressure drop across the length of the channel, and (2) it reduces the distance the fluid must travel while being heated by the IC, reducing the maximum temperature the fluid will reach, and allowing more efficient heat transfer.

D. Pressure and Pumping Power

The pressure drop from the inlet to the outlet of a microfluidic channel is calculated as $\Delta p = 2\gamma\mu L v D_h^{-2}$ [29] where Δp is the pressure drop, γ is an empirical factor depending on the microchannel aspect ratio, μ is the fluid viscosity, L is the microchannel length, v is the microchannel velocity, and D_h is the hydraulic diameter of the microchannel. We considered pressure drops in the range of 0 to 200 kPa. Pressure is a measure of force per area, and

Table II
XMT PHYSICAL CONFIGURATIONS

	8k	16k	32k	64k
Silicon Layers	2	3	5	8
Microchannel Layers	1	2	4	7
Silicon Area per Layer (mm^2)	276	302	328	380
Total Silicon Area (mm^2)	551	906	1641	3046
Microchannels per Layer	100			

100 kPa is close to 1 atmosphere of pressure. Pressure is a limiting factor in the design of microfluidic flow, as high pressure in the microchannels can cause damage to the IC. In [27], pressure drops as high as 1,000 kPa were considered. In this paper, we consider pressure drops of up to 200 kPa. Beyond this point, we see only minimal further temperature reduction.

Pumping power is calculated as $P_{pump} = Nf\Delta p$ [29] where P_{pump} is the pumping power, N is the total number of microchannels, f is the fluid flow rate per microchannel, and Δp is the fluid pressure drop. The maximum pumping power for any configuration considered was 25.6 W, and the highest ratio of pumping power to power dissipated by the IC was 1.85%.

E. Thermal Model

3D-ICE (for 3D Interlayer Cooling Emulator) is a thermal simulation tool for 3D ICs with microfluidic cooling [30], [31]. 3D-ICE uses a thermal resistance approach, which is motivated by an analogy between heat and electrical conduction. The 3D stack is divided into thermal cells, which can be represented as an electrical node connected to surrounding nodes with resistors. The resistance of these resistors depends on the thermal conductance of the material, the geometry of the cell, and the fluid flow in the cell. Similar connections are made between heat capacity and capacitance, heat sources and current sources, etc.

Figure 4 shows a representative cross-section view of the 3D ICs as described to 3D-ICE. We modeled a heat sink connected to an ambient temperature of 40 °C, and assumed a constant microchannel inlet temperature of 50 °C, with water as the coolant. Any configurations with no fluid flow were modeled without microchannels, as in fig. 4a. Figure 4 also shows the thermal conductivity of each layer in the 3D stack.

Figure 5 gives the flavor of the layout of the active silicon layers. The interconnection network is confined to one silicon layer, which it shares with some number of clusters and caches.

The power dissipated in each component depends on the activity of that component and on the temperature calculated for that component in the last iteration (or, in the first iteration, an assumed temperature of 47 °C). The simulation is complete when the temperature profile between two consecutive iterations is identical to within 0.1 °C, the finest distinction reported by 3D-ICE.

F. Benchmarks

We used parallel implementations of breadth first search (BFS), sparse matrix-vector multiplication (matvec), mergesort, and 3-dimensional fast Fourier transform (FFT) as benchmark programs.

- BFS: We use a parallel version of breadth first search.

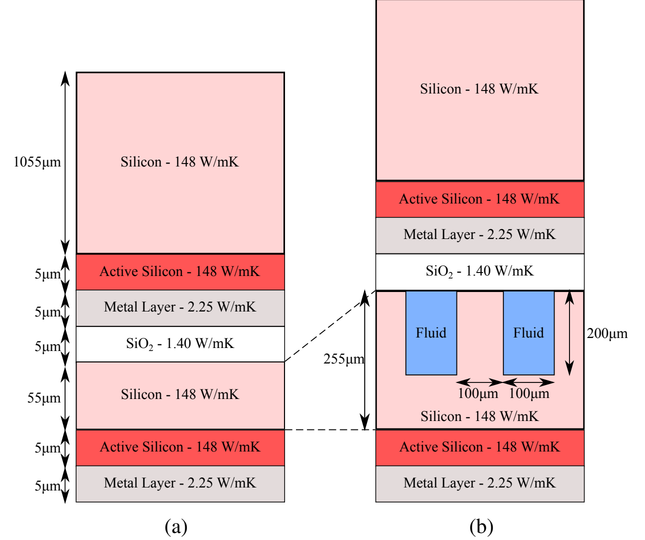

Figure 4. Cross-section view of the (a) uncooled and (b) cooled 3D IC stack for the 8k XMT configuration. Figures are not to scale.

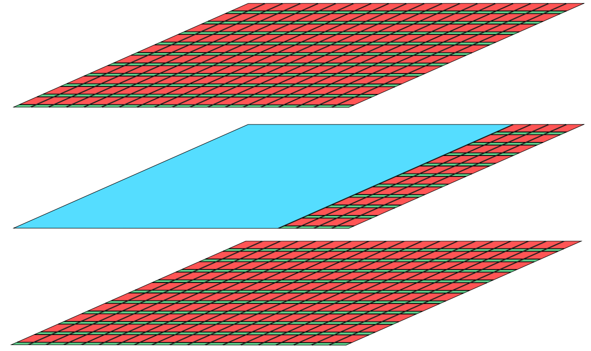

Figure 5. Representative floorplan for the active silicon layers with and without the NoC. Red blocks represent TCU clusters, green blocks represent memory modules, and the blue block represents the NoC. The NoC is confined to one layer, which it shares with some TCU clusters and memory modules.

- matvec: All entries in the result vector are computed in parallel with each other. However, the work of calculating each result is done serially.
- mergesort: All merges of the same size are done in parallel, requiring $\log n$ rounds. In the early rounds, when the merge size is small, the merges are done with a serial algorithm. After a defined crossover point, the merges are done with a parallel merging algorithm.
- FFT: The FFT is computed using a radix-8 decimation-in-frequency form of the Cooley-Tukey algorithm. An input of size $n \times n \times n$ is processed in $3 \log_8 n$ rounds. In each round, the input is divided into $n^3/8$ subproblems of size 8. The subproblems are solved in parallel by applying a serial algorithm to each subproblem.

IV. RESULTS

Because McPAT can only determine the power dissipation for components with temperatures between 27 °C and 127 °C, if any

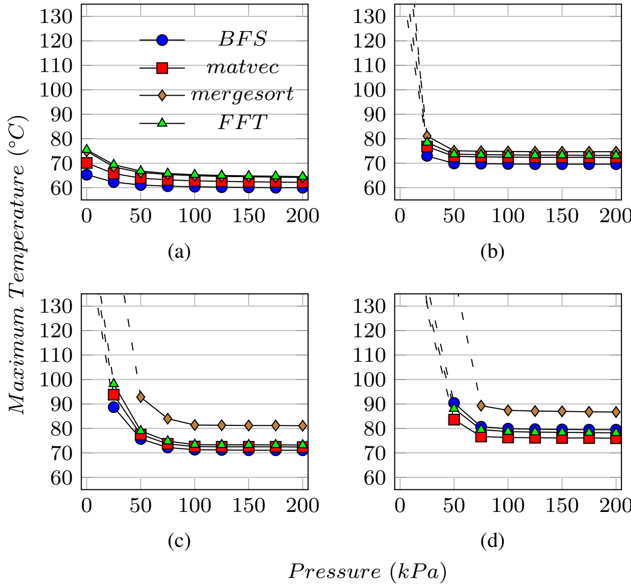

Figure 6. Maximum temperature reached while running various benchmarks on (a) 8k TCU, (b) 16k TCU, (c) 32k TCU, and (d) 64k TCU configurations. Dotted lines represent temperature values which exceed 127 °C, beyond which McPAT cannot accurately model power consumption. Results with a pressure drop of 0 kPa represent configurations which employ only air-cooling, not microfluidic cooling.

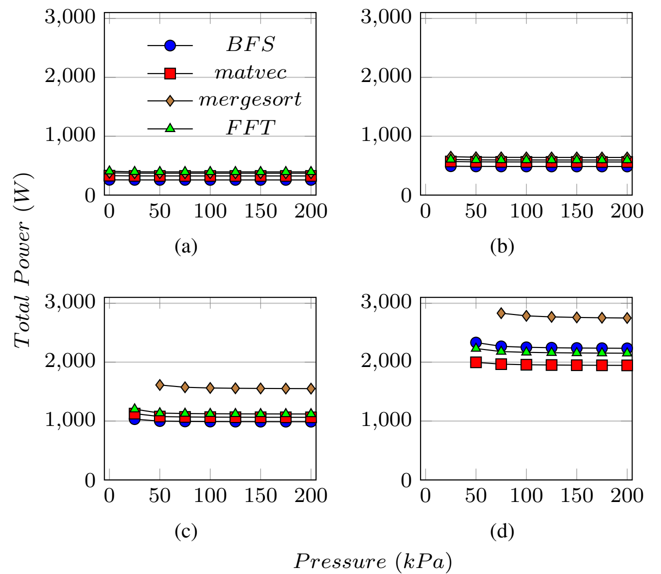

Figure 7. Maximum power dissipated while running various benchmarks on (a) 8k TCU, (b) 16k TCU, (c) 32k TCU, and (d) 64k TCU configurations. Total power presented only for cases where maximum temperature is feasible, namely under 127 °C, see fig. 6.

component exceeded 127 °C then simulations were halted. In figs. 6 and 7, therefore, some data points are excluded for simulations where the maximum temperature exceeded 127 °C. In fact, the temperature of some of the omitted data points exceeded 400 °C, even ignoring any increase in leakage power. So, it is safe to assume that many of the omitted data points and their configurations are infeasible.

Figure 6 shows the maximum temperatures reached in each architecture for fluid pressure between 0 and 200 kPa, for the four benchmark programs, with a pressure drop of 0 kPa representing a configuration with no micro-fluidic channels, only air cooling. There is a dramatic decrease in maximum temperature as fluid pressure increases from 0 to 100 kPa. At 100 kPa fluid pressure, the maximum temperature reached for any benchmark is 65.3 °C on the 8k TCU configuration, 74.8 °C on the 16k TCU configuration, 81.4 °C on the 32k TCU configuration, and 87.4 °C on the 64k TCU configuration. As fluid pressure increases from 100 to 200 kPa, further reduction in maximum temperature is small.

An IC temperature of 125 °C is considered the upper limit in military specification [17]. For the 16k TCU and larger XMT configurations, 3D-ICs which employ air cooling only and no microfluidic cooling exceed this threshold, and adding microfluidic cooling brings the maximum temperature well below the threshold.

See fig. 7 for a comparison of the power dissipated in various configurations. At 100 kPa fluid pressure, the maximum power dissipated for any benchmark is 395 W on the 8k TCU configuration, 641 W on the 16k TCU configuration, 1562 W on the 32k TCU configuration, and 2786 W on the 64k TCU configuration. The 8k TCU configuration is the only one for which air-cooling alone would be sufficient to keep the 3DIC at a low enough temperature for McPAT to model its power dissipation.

An additional benefit of microfluidic cooling is that by reducing temperature, one also reduces the leakage power dissipated by the processor (and, in turn, its snow balling effect on increasing temperature). In fact, for all configurations and all benchmarks with a pressure drop of 125 kPa or less, the leakage power is reduced by an amount greater than the pumping power required for the cooling fluid. Figure 8 shows a comparison of the leakage power saved by microfluidic cooling and the pumping power required for the 8k TCU configuration.

V. Discussion

A. Robustness of Results

The results presented in this paper are specific to the XMT architecture. However, it can be argued that any parallel architecture which is capable of the same performance will require silicon area and power consumption that are not very different. The main architectural element specific to XMT is the interconnection network. In Figure 9, we see that the total power consumption by the interconnection network is less than 18% for all configurations with 16k TCUs or more. Figure 10 shows a similar trend in the silicon area taken by the interconnection network choices in this paper. Driven by the need to keep the interconnection network in a single layer and at the same time not compromising performance, these were not arbitrary choices.

In this paper we present architectures in the 22 nm technology node. However, we believe the claim that microfluidic cooling can enable large many-core architectures remains valid in smaller technology nodes. Intel claims that logic area in their 14 nm process will be 0.54 times the logic area in their 22 nm process. They likewise claim ∼0.54 scaling in power consumption between the two technology nodes [7]. Thus, the power density of an

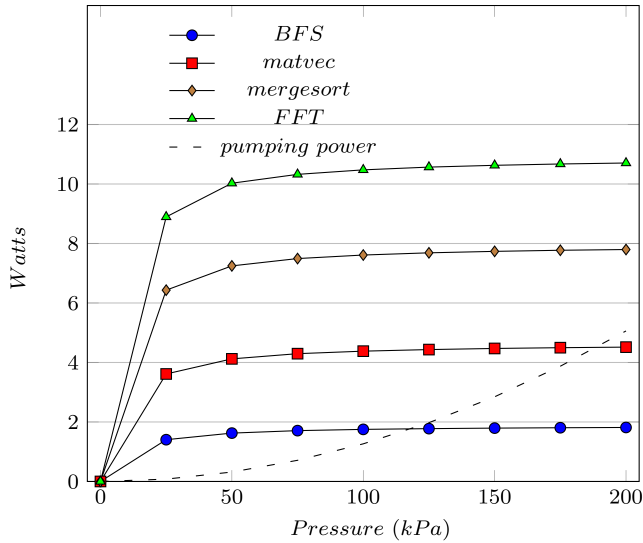

Figure 8. BFS, matvec, mergesort, FFT: Reduction in leakage power used by the 8k TCU configuration (versus air-cooled only). Dotted line: The pumping power required by the 8k TCU configuration for a given pressure.

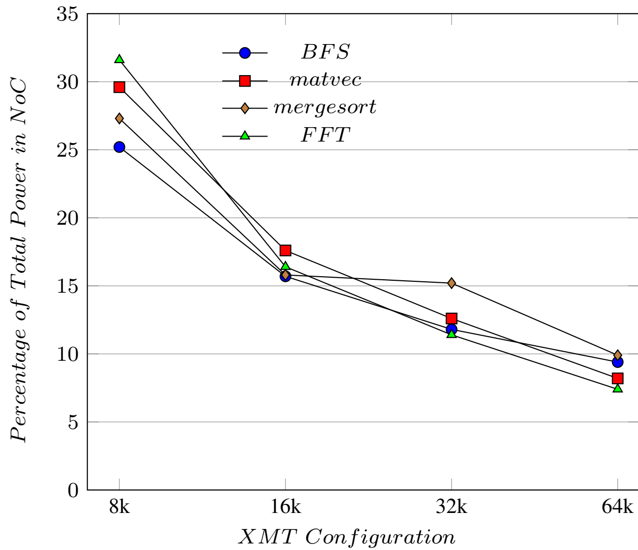

Figure 9. Percentage of total power used in the interconnection network for with 100 kPa cooling fluid pressure.

architecture implementation in both nodes would be roughly equal. Our 64k TCU configuration, which we estimate to require 3046 mm^2 and 2786 W in 22 nm would require around 1645 mm^2 and 1504 W in 14 nm (for the mergesort benchmark with 100 kPa cooling fluid pressure). These are a close approximation to the area and power estimates for the same benchmark, but with 32k TCU in 22 nm (1641 mm^2 and 1562 W). This means that we would expect to draw the same conclusions for a 64k TCU configuration in 14 nm as for a 32k TCU configuration in 22 nm—that it is thermally infeasible with air-cooling only, but that adding microfluidic cooling makes it feasible.

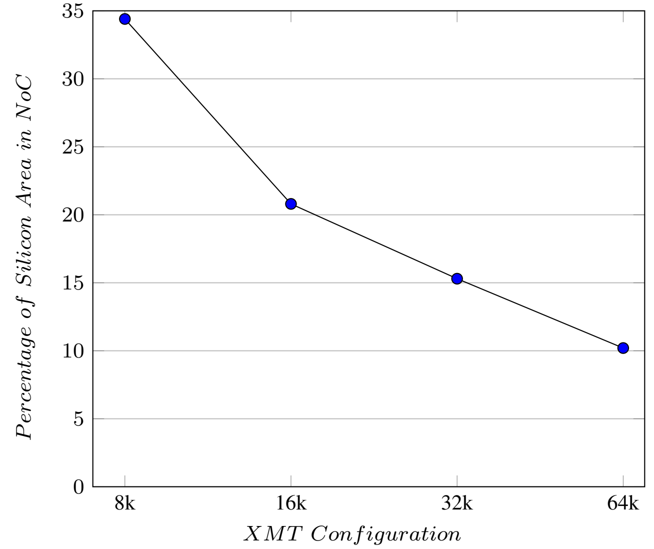

Figure 10. Percentage of total area used by the interconnection network.

B. Summary of Relevant Advantages of the XMT Architecture

To keep this presentation short, we summarize below the main nuggets from numerous publications listed on the XMT home page www.umiacs.umd.edu/users/vishkin/XMT:

1) Speedups. PRAM is that the main theory of parallel algorithms. A "proof-of-performance" with respect to PRAM algorithms demonstrated speedups between 1 and 2 orders of magnitude (up to 129X) on the most advanced parallel algorithms in the literature relative to the best known results on any machine (e.g., on GPUs) for any algorithms for the same problem. See table III. Other published speedups include 20.4X on a 64-TCU XMT versus 4X on a 16-core AMD (using the same silicon area) for FFT [28] and 100X on a gate-level simulation benchmark suite [14].
2) Ability to achieve "strong scaling", defined as getting larger speedups with more processors and the same problem size; and, the less rigorous, but very important:
3) Programmer's productivity. Evidence on ease of programming includes:
    a) Direct comparison with other platforms, supervised by third party software engineering experts, done in collaboration with: (i) UC Santa Barbara: 50% development time on XMT over MPI [15]; and (ii) U. Illinois: strong speedups by all 42 joint students on XMT versus no speedups on OpenMP [26]
    b) 370 high-school students who have already programmed XMT. 2014 update to [32].
    c) A national award winner high-school teacher described XMT as the one platform that strips away industrial accident from the parallel computing student's mind and published a paper in SIGCSE'10 supporting that. [32]
    d) In a graduate theory class on parallel algorithms, students submit 5-6 programming assignments delivering (in a 2-week project) significant speedups over best serial even on problems that are research level on cur-

Table III
XMT SPEEDUPS

	XMT	GPU/CPU	Factor
Graph Biconnectivity [11]	33X	4X, but only on random graphs	≫8
Graph Triconnectivity [13]	129X	Only serial result	129
Max Flow [9]	108X	2.5X	43
Burrows Wheeler Compression [12]	25X	X/2.5 on GPU	70
Transform - bzip2 Decompression [12]	13X	1.1X	11

Table IV
FFT SPEEDUPS

Configuration	8k	16k	32k	64k
No photonics	41X	79X	129X	225X
With photonics	53X	106X	207X	381X

rent commercial platforms (e.g., for finding biconnected components in graphs).

e) A Department of Defense employee was able to solve in an MS scholarly paper (a 2-month effort, which is order of magnitude less than an MS thesis) problems that would be the basis for at least one PhD on other platforms.

C. Extending the Work Beyond the Chip Boundary

Overall, bandwidth and latency matter at all levels (DRAM as well as cache) though they may matter in different ways for different applications. To that end, it would seem natural to incorporate photonic transceivers to the architecture, allowing high speed and high bandwidth access to main memory. Simulations with XMTSim show that increasing bandwidth to main memory can improve the execution speed of the FFT benchmark program by a factor of up to 1.69X, as shown in table IV. FFT was a natural here as it is known to under-perform when problem size does not fit into cache and bandwidth to main memory is limited [16].

Papers such as [24] and [39] show that high bandwidth, low power silicon photonic transceivers are possible. Using the ANSYS [1] simulation tool, companion work in progress with Mechanical Engineering Professor Bao Yang simulated a photonic structure of 10x10x2 µm^3 capable of sending and receiving at a rate of 25 Gb/s. The simulations, see fig. 11, showed the temperature reached by the photonic transceiver under a range of assumptions about the power required to send and receive messages. The simulations assumed that a substrate of 50x50x50 µm^3 comprises the photonic structure coupled with standard high thermal conductivity material for connecting the transceiver to a microfluidic channel. We found that if the transceiver uses less than 1.1 pJ/b it can be cooled with a microfluidic channel and remain below 100 °C.

A rough calculation (20X20 per mm^2 times 400 mm^2) indicates that one could fit on the order of 100,000 such transceivers on 20x20 mm of silicon. An interesting avenue of future study is to determine what bandwidth to main memory is feasible with photonic transceivers, and how this increased bandwidth can improve the performance of a parallel architecture. However, what matters for the current paper, and the reason for citing this companion work, is that there are feasible ways to extend mitigation of data movement beyond the chip boundary.

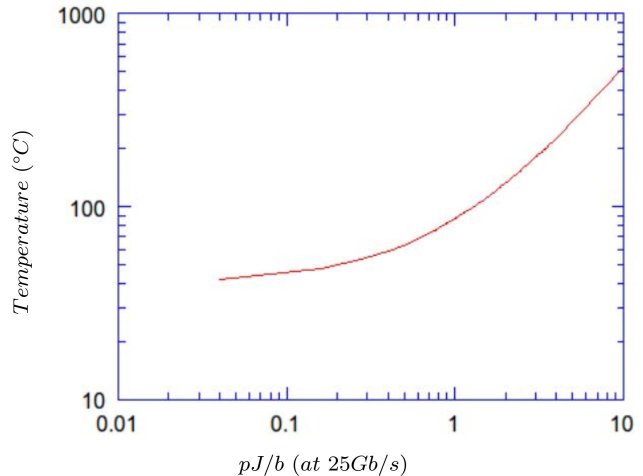

Figure 11. Temperature reached by a photonic device operating at 25 Gb/s, with a range of assumptions on power usage.

D. Work in Progress on Extensions Beyond the Scope of This Paper

We believe that the ideas in the current paper are quite powerful. We briefly point out their potential for several other significant applications. We start with further scaling of XMT, but then take the basic ideas to a rather different domain: an upgraded switch for connecting modules of (current) high-performance computing systems.

1) Further scaling of XMT: The following idea can help scale XMT further. Consider an XMT system in which a separate chip comprises the cluster-memory interconnection network. Other XMT components could either form a separate chip each, or form together joint chips in any of many possible combinations.

The current idea is to implement the interconnection network chip (and possibly other chips) using 3D-VLSI, microfluidic cooling and silicon photonics, along the lines presented earlier in this paper. Since TCUs could be spread over many chips, the XMT design could be much larger than even the enlarged version discussed in prior chapters of this paper.

2) Upgraded switch: more ports, larger bandwidth and lower latency: The above cluster-memory interconnection network chip actually provides a flexible all-to-all switch that enables routing in parallel data from every incoming port to every outgoing port. Per [38], the number of ports on a chip is a bottleneck in state-of-the-art switches. That paper explains that to reduce the number of switches, the topology of an interconnection network should utilize

all the available switch ports. Our approach of greatly increasing the number of switch ports would improve the overall bandwidth and latency of the topology.

VI. Conclusion

In this paper, we considered the thermal feasibility of many-core architectures that do not attempt to avoid data movement. For the test case we studied, we found that standard air cooling was insufficient but that on-chip temperatures can be reduced from levels in excess of 400 °C with standard air cooling to below 90 °C with microfluidic cooling, using pumping power less than 2% of the power dissipated by the processor. In fact, the pumping power used to drive the cooling fluid is lower than the leakage power saved by reducing on-chip temperature. This advances the argument that data movement in particular and power dissipation in general are not feasibility constraints, but a trade-off to be considered among other trade-offs, such as manufacturing cost, ease-of-programming, and performance.

This work represents non-standard outreach of parallel algorithms. Efforts such as the SB-PRAM project in Germany and the "PRAM-On-Chip" XMT stood out in reaching architecture, as well as compiler and run-time. This work, however, transcends computer science, deep into enabling technologies benefiting from a breadth of areas, beyond SPAA, covered by the authors. Sean O'Brien is a mechanical engineer by training, and UMD faculty Edo Waks specializes in photonics and Bao Yang in cooling. The work itself makes an original contribution by combining three technologies (3D-VLSI, microfluidic cooling and silicon photonics) in new ways for support of parallel algorithms, which is more scalable and more effective, making parallel algorithms both the driver behind this work and its primary beneficiary.

Acknowledgements

We gratefully acknowledge support from DARPA, NIH and NSF. We also acknowledge discussions with Avram Bar-Cohen (DARPA) on microfluidic cooling, Fuat Keceli (Intel) on power, and Caleb Serafy on 3D-VLSI.

References

[1] ANSYS, Inc. [Online]. Available: http://www.ansys.com

[2] A. O. Balkan, G. Qu, and U. Vishkin, "An area-efficient high-throughput hybrid interconnection network for single-chip parallel processing," in *Proceedings of the 45th annual Design Automation Conference*, 2008, pp. 435–440.

[3] A. O. Balkan, G. Qu, and U. Vishkin, "Mesh-of-trees and alternative interconnection networks for single-chip parallelism," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 10, pp. 1419–1432, 2009.

[4] A. O. Balkan, "Mesh-of-trees interconnection network for an explicitly multi-threaded parallel computer architecture," Ph.D. dissertation, Electrical and Computer Engineering, University of Maryland, 2008.

[5] A. Bar-Cohen, J. J. Maurer, and J. G. Felbinger, "DARPA's Intra/Interchip Enhanced Cooling (ICECool) Program," *CS MANTECH*, pp. 171–172, 2013.

[6] K. Bergman, L. P. Carloni, A. Biberman, J. Chan, and G. Hendry, *Photonic Network-on-chip Design*. Springer, 2014.

[7] R. Borkar, M. Bohr, and S. Jourdan, "Advancing Moore's Law on 2014," Intel, Aug. 2014. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/presentation/advancing-moores-law-in-2014-presentation.pdf

[8] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proceedings of the 27th Annual International Symposium on Computer Architecture*, 2000.

[9] G. C. Caragea and U. Vishkin, "Brief Announcement: Better Speedups for Parallel Max-flow," in *Proceedings of the Twenty-third Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2011, pp. 131–134.

[10] J. Demmel, "Communication-avoiding algorithms for linear algebra and beyond," in *IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, May 2013, pp. 585–585.

[11] J. A. Edwards and U. Vishkin, "Better Speedups Using Simpler Parallel Programming for Graph Connectivity and Biconnectivity," in *Proceedings of the International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM)*, 2012, pp. 103–114.

[12] J. A. Edwards and U. Vishkin, "Empirical speedup study of truly parallel data compression," University of Maryland, Tech. Rep., 2013. [Online]. Available: http://hdl.handle.net/1903/13890

[13] J. A. Edwards and U. Vishkin, "Brief Announcement: Speedups for Parallel Graph Triconnectivity," in *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2012, pp. 190–192.

[14] P. Gu and U. Vishkin, "Case study of gate-level logic simulation on an extremely fine-grained chip multiprocessor," *Journal of Embedded Computing*, vol. 2, no. 2, pp. 181–190, 2006.

[15] L. Hochstein, V. R. Basili, U. Vishkin, and J. Gilbert, "A pilot study to compare programming effort for two parallel programming models," *Journal of Systems and Software*, vol. 81, no. 11, pp. 1920–1930, 2008.

[16] J.-W. Hong and H. T. Kung, "I/O complexity: The red-blue pebble game," in *Proceedings of the thirteenth annual ACM Symposium on Theory of Computing (STOC)*, 1981, pp. 326–333.

[17] B. Hunt and A. Tooke, "High temperature electronics for harsh environments," in *18th European Microelectronics and Packaging Conference (EMPC)*, Sept 2011, pp. 1–5.

[18] J. JáJá, *An introduction to parallel algorithms*. Addison-Wesley Reading, 1992, vol. 17.

[19] F. Keceli, A. Tzannes, G. C. Caragea, R. Barua, and U. Vishkin, "Toolchain for Programming, Simulating and Studying the XMT Many-Core Architecture," in *Proc. 16th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS), in conjunction with IPDPS*, 2011.

[20] F. Keceli and U. Vishkin, “XMTSim: A Simulator of the XMT Many-core Architecture,” University of Maryland, Tech. Rep., 2011. [Online]. Available: http://hdl.handle.net/1903/13893

[21] J. Keller, C. Kessler, and J. Träff, *Practical PRAM programming*. Wiley-Interscience, J. Wiley & Sons, Inc., 2001.

[22] P. Koka, M. McCracken, H. Schwetman, X. Zheng, R. Ho, and A. Krishnamoorthy, “Silicon Photonic Network Architectures for Scalable, Power-Efficient Multi-Chip Systems,” in *Proceedings of the 37th ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2010.

[23] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, “McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures,” in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*, 2009, pp. 469–480.

[24] F. Liu, D. Patil, J. Lexau, P. Amberg, M. Dayringer, J. Gainsley, H. Moghadam, X. Zheng, J. Cunningham, A. Krishnamoorthy, E. Alon, and R. Ho, “10 Gbps, 530 fJ/b optical transceiver circuits in 40 nm CMOS,” in *Symposium on VLSI Circuits (VLSIC)*, June 2011, pp. 290–291.

[25] L. I. Millett, S. H. Fuller *et al.*, *The Future of Computing Performance: Game Over or Next Level?* National Academies Press, 2011.

[26] D. Padua, U. Vishkin, and J. C. Carver, “Joint UIUC/UMD parallel algorithms/programming course,” *Proc. EduPar*, 2011.

[27] M. Sabry, A. Sridhar, and D. Atienza, “Thermal balancing of liquid-cooled 3D-MPSoCs using channel modulation,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, March 2012, pp. 599–604.

[28] A. B. Saybasili, A. Tzannes, B. R. Brooks, and U. Vishkin, “Highly Parallel Multi-Dimensional Fast Fourier Transform on Fine-and Coarse-Grained Many-Core Approaches,” in *Proceedings of the 21st IASTED International Conference*, vol. 668, no. 018, 2009, p. 107.

[29] B. Shi, A. Srivastava, and P. Wang, “Non-uniform micro-channel design for stacked 3D-ICs,” in *Proceedings of the 48th Design Automation Conference*, 2011, pp. 658–663.

[30] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschwiler, “3D-ICE: A Compact Thermal Model for Early-Stage Design of Liquid-Cooled ICs,” *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2576–2589, Oct 2014.

[31] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, “3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2010, pp. 463–470.

[32] S. Torbert, U. Vishkin, R. Tzur, and D. J. Ellison, “Is Teaching Parallel Algorithmic Thinking to High School Students Possible? One Teacher’s Experience,” in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*. ACM, 2010, pp. 290–294.

[33] W.-P. Tu, Y.-H. Lee, and S.-H. Huang, “TSV sharing through multiplexing for TSV count minimization in high-level synthesis,” in *IEEE International SOC Conference (SOCC)*, 2011, pp. 156–159.

[34] D. Velenis, M. Stucchi, E. J. Marinissen, B. Swinnen, and E. Beyne, “Impact of 3D design choices on manufacturing cost,” in *IEEE International Conference on 3D System Integration (3DIC)*, 2009, pp. 1–5.

[35] U. Vishkin, “Is Multicore Hardware for General-purpose Parallel Processing Broken?” *Commun. ACM*, vol. 57, no. 4, pp. 35–39, Apr. 2014.

[36] Z. Wan, H. Xiao, Y. Joshi, and S. Yalamanchili, “Co-design of multicore architectures and microfluidic cooling for 3D stacked ICs,” in *19th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*. IEEE, 2013, pp. 237–242.

[37] X. Wen and U. Vishkin, “FPGA-based prototype of a PRAM-on-chip processor,” in *Proceedings of the 5th conference on Computing frontiers*, 2008, pp. 55–66.

[38] E. Zahavi, I. Keslassy, and A. Kolodny, “Quasi Fat Trees for HPC Clouds and Their Fault-Resilient Closed-Form Routing,” in *IEEE 22nd Annual Symposium on High-Performance Interconnects (HOTI)*, 2014, pp. 41–48.

[39] X. Zheng, F. Y. Liu, J. Lexau, D. Patil, G. Li, Y. Luo, H. D. Thacker, I. Shubin, J. Yao, K. Raj *et al.*, “Ultralow power 80 Gb/s arrayed CMOS silicon photonic transceivers for WDM optical links,” *Journal of Lightwave Technology*, vol. 30, no. 4, pp. 641–650, 2012.