

Plasmonics and the Parallel Programming Problem

Uzi Vishkin^{*a,b}, Igor Smolyaninov^a, Chris Davis^a

^aElectrical and Computer Engineering, University of Maryland; ^bUniversity of Maryland Institute for Advanced Computer Studies (UMIACS); College Park, MD 20742

ABSTRACT

While many parallel computers have been built, it has generally been too difficult to program them. Now, all computers are effectively becoming parallel machines. Biannual doubling in the number of cores on a single chip, or faster, over the coming decade is planned by most computer vendors. Thus, the parallel programming problem is becoming more critical. The only known solution to the parallel programming problem in the theory of computer science is through a parallel algorithmic theory called PRAM. Unfortunately, some of the PRAM theory assumptions regarding the bandwidth between processors and memories did not properly reflect a parallel computer that could be built in previous decades. Reaching memories, or other processors in a multi-processor organization, required off-chip connections through pins on the boundary of each electric chip. Using the number of transistors that is becoming available on chip, on-chip architectures that adequately support the PRAM are becoming possible. However, the bandwidth of off-chip connections remains insufficient and the latency remains too high. This creates a bottleneck at the boundary of the chip for a PRAM-On-Chip architecture. This also prevents scalability to larger “supercomputing” organizations spanning across many processing chips that can handle massive amounts of data. Instead of connections through pins and wires, power-efficient CMOS-compatible on-chip conversion to plasmonic nanowaveguides is introduced for improved latency and bandwidth. Proper incorporation of our ideas offer exciting avenues to resolving the parallel programming problem, and an alternative way for building faster, more useable and much more compact supercomputers.

Keywords: Plasmonics, Multiprocessing, Parallel programming, Parallel algorithmic computer architecture, PRAM

1. INTRODUCTION

To date, the outreach of parallel computing has fallen short of historical expectations. This has primarily been attributed to programmability shortcomings of parallel computers. The computer industry is considering this problem a key bottleneck for progress in the commodity processor space. Overall, there is a strong renewed interest in inventing new programming languages that accommodate simple representation of concurrency. However, during the previous decades thousands of papers have been written on this topic. This effort brought about a fierce debate between a considerable number of schools-of-thoughts. One of these approaches, the “PRAM approach”, emerged as a clear winner in this “battle of ideas”. In fact, we would like to defend an even stronger premise: “Had a parallel architecture that can look to the performance programmer like a PRAM been feasible in the early 1990s, its parallel programming approach would have become common knowledge and the prevailing standard by now”. As evidence to support this premise we point out that 3 of the main algorithms textbooks (taught in standard undergraduate computer science courses everywhere by 1990) [14-16] chose to include large chapters on PRAM algorithms. The PRAM was the model of choice for parallel algorithms in all major algorithms/theory communities and was taught everywhere. The only reason that this win did not register in the collective memory as the clear and decisive victory it really was is that, at about the same time (early 1990s), it became clear that it will not be possible to build such a machine (i.e., one that can look to the performance programmer as a PRAM) using early 1990s technology.

The Parallel Random Access Model (PRAM) is an easy model for parallel algorithmic thinking and for programming. It abstracts away architecture details by assuming that many memory accesses to a shared memory can be satisfied within the same time as a single access. As noted above, the PRAM was developed during the 1980s and 1990s in anticipation of a parallel programmability challenge. It provides the second largest algorithmic knowledge base right next to the standard serial knowledge base.

Although interesting, multi-chip multiprocessor designs that aim to support the PRAM (such as Tera/Cray MTA [8] and SB-PRAM [9]) are constrained by inter-chip interconnections. Latency and bandwidth problems have limited their

success in supporting PRAM. With the continuing increase of silicon capacity, it becomes possible to build a single-chip parallel processor. Such demonstration has been the purpose of the Explicit Multi-Threading (XMT) project [5,6] that seeks to prototype a PRAM-On-Chip vision. While on-chip interconnection networks provide enough bandwidth for connecting processors-to-memories, bandwidth to off-chip memories remains a challenge. The current paper suggests harnessing plasmonics to cope with this challenge.

2. THE BASIC XMT ORGANIZATION

In one example of an XMT on-chip multiprocessor organization, 64 clusters of 16 processors each (for a total of 1024 processors) will be placed on a single chip and, using an on-chip interconnection network, will be connected to 64 memory modules. For the current paper we henceforth only focus on the memory architecture. The memory architecture is based on partitioning the memory space to memory modules already from the first level of the cache. Namely, each logical address can reside in exactly one memory module. An important attribute of such memory architecture is that it frees the system from the need to provide cache coherence and its implied high overheads. This attribute translates directly into significant performance advantages, especially for computer applications where shortening single task completion time is sought, which is the main objective of the XMT approach. Going back to the example, each of the 64 memory modules provides on-chip the higher levels (“caches”) of the memory hierarchy for their respective component of the memory partition. A key bottleneck for such architectures is the bandwidth and latency of connecting each on-chip memory module to a memory chip comprising a lower level of its memory hierarchy. Figure 1 shows several on-chip memory modules, each connected to a separate memory chip.

What we propose is to use optics-based technologies in order to significantly alleviate this bottleneck. The E-to-P boxes depict conversion of an electronic signal to an optical signal on the chip. Much of the paper is devoted to this conversion and the implementation of the interchip communication.

This basic memory architecture and incorporation of hashing to avoid hot spots were discussed in [8-11]

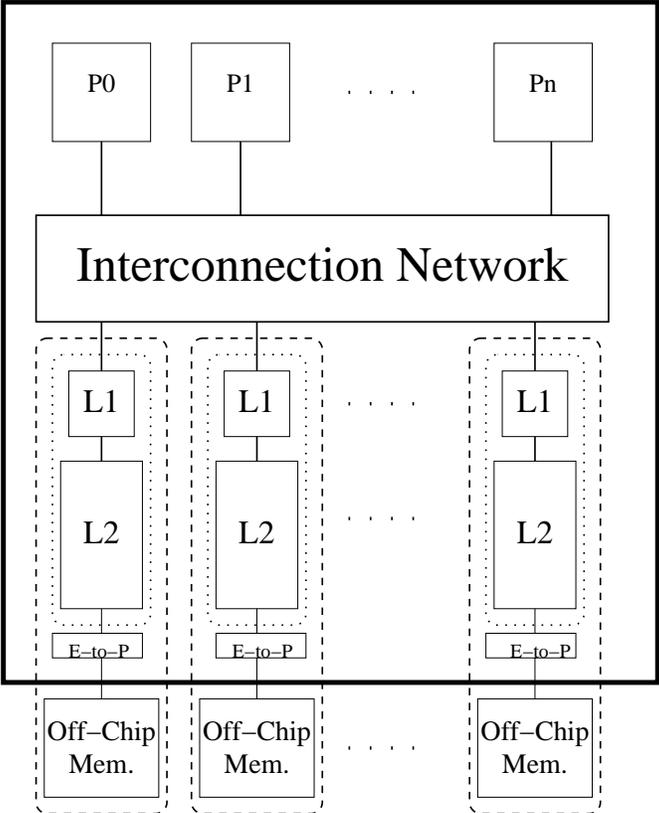


Figure 1: The chip boundary of a PRAM-On-Chip architecture (depicted by the square in bold lines)

For background, it is also important to understand the demand on the interconnection network in the XMT single-chip parallelism context. The messages between processor cluster and the cache modules are very small (e.g., one word for load instructions and at most two words for the store instructions). This may explain why the network should be designed to locally route packets with minimal overheads. Overall, the specs of the interconnection network require, at least in the short term, an electric solution. This constraint significantly affects the current paper. We expend a bit more on background. A hardware thread should be able to send memory requests to any memory location on the chip. Coupled with the objective of avoiding cache coherence issue, this requires placing an interconnection network between processing units and the first level of the cache and will cause a higher latency compared to cache access latencies of traditional serial processors. Multiple threads will run concurrently to hide this latency. In order to satisfy the steady and high demand of threads for data, the interconnection network needs to provide high throughput between the processing units (threads) and the memory modules. Second, the interconnection network needs to provide low on-chip communication latency. This will allow designating fewer threads just to hide latencies and will simplify the overall design. It will also improve performance in cases where a sufficiently large number of thread contexts is not available to overlap communication with computation. The interconnection network also needs to take as little chip area as possible. For the current paper we only state that studies we have done demonstrate that these interconnections network objectives can generally be accomplished. Also, as the reader can readily recognize, quite a few memory architecture consideration that are not directly relevant to the discussion that follows were suppressed. As one example for the type of suppressed issues, consider the possibility of using read-only caches in ways which do not require cache coherence within the processing clusters.

In any case, the main remaining bottlenecks are the off-chip latency and bandwidth. We first overview changes to the architecture that incorporate a plasmonics-based solution to this problem. Possible ways for scaling up the XMT architectures to a much larger machine are also discussed. Finally, the plasmonics-based solution is presented.

3. ORGANIZATION OF PLASMONICS-BASED ENHANCEMENTS OF XMT

Reaching larger off-chip memories (i.e., lower levels in the memory hierarchy, such as main memory) requires off-chip connections through pins on the boundary of the electric chips. Current all-electronic solutions for such interface require significant chip area (in one example we looked at, an all-electronic solution needs 40% of a large chip) in order to provide adequate bandwidth and even then the bandwidth is not optimal. One fundamental problem is that while the miniaturization trend of CMOS VLSI continues to look promising, the chip area needed for I/O pads and interfacing does not decrease at the same rate. It should also be clear that the general issues discussed in the current paper are important for almost any other architectures since it is quite rare that all processing power and memory fit into one chip.

The current paper provides a way for alleviating this general computer architecture problem. Instead of making connections through pins and wires we propose on-chip conversion to plasmonic nanowaveguides. Each such nanowaveguide can be interconnected to more standard (optical or microstrip) waveguides at suitable locations. Each of the latter waveguides will be orders of magnitude thicker and could be as long as required by the exact application. In addition, using techniques, which are either standard in the art, or a variation of the above ideas, namely, on-chip conversion between plasmonics and electronics and vice-versa, and similarly between standard waveguides and plasmonics, the current paper could also alleviate bandwidth problems among various components of a computer system.

Given the basic structure of Figure 1, where each of the on-chip memory modules is connected to a separate memory chip, a way for using an optics based solution to implement each such connection is presented. In addition to on-chip conversion to plasmonic nanowaveguides our basic idea is to use a "flower arrangement" type arrangement in order to create a layer about 1 cm above the chip level (Figure 2). This layer will provide an interface that will allow further conversion to more standard high bandwidth optical connection such as fiber, or waveguide technology, to memory chips. Figure 2 shows the central chip in the bottom. Later in the presentation we explain how each connection from within the chip to a memory chip will comprise the following: (i) Electronic signal are converted within the chip to plasmonic nanowaveguides. An example of one way in which this can be done is shown in Figure 5. (ii) Using a very thin metal wire-like plasmonic nanowaveguide (*there are different designs of such nanowaveguides which may be a few micrometers to a few centimeters long, depending on the optical frequency used; such waveguides typically start as two or more ~50-100 nm thin rectangular metallic electrodes separated by ~30 nm dielectric gaps, the transverse dimensions of the waveguide may change gradually so that near its end the plasmonic nanowaveguide becomes similar to*

regular one: see detailed description in the next section) each one of the on-chip memory modules will be connected to an optical interconnect interface in which each such nanowaveguide is converted to more standard optical waveguides. Each of the latter waveguides will be orders of magnitude thicker than the nanowaveguides. This interconnection will involve plasmon to optical conversion and insertion into the waveguide. (iii) Conversion back to plasmonics and into a memory chip where plasmonic nanowaveguides are converted back to electric signals (which is simple and well understood).

The flower arrangement of Figure 2 expands the optical interconnect interface to allow more space for the thicker connections. Each of the latter waveguides will connect to a memory chip. It should be able to provide high bandwidth with low latency between the memory chip and the on-chip memory module. The challenges to implementing such a solution involve: optimization of electronic to plasmonic on-chip conversion in terms of conversion efficiency and plasmonic scattering, and optimization of coupling between plasmonic and regular waveguides. The right metric for measuring performance improvement should be the sustainable bandwidth and latency of the connection to the memory chips. The proposed application should be able to drastically improve the performance of on-chip multiprocessors: (i) the area currently needed for the same interface could be devoted to processors (and functional units); (ii) better bandwidth and latency to memory chip may allow moving more of the memory to memory chips devoting even more area to processors; (iii) much greater memory scalability of the overall architecture will become possible, to a point where compact designs could outperform the most expensive and largest supercomputers built today for many more applications. This includes machines built out of hundreds of thousands of chips. What may be even more important than performance is that the memory architecture discussed will facilitate a much easier way to program parallel computers.

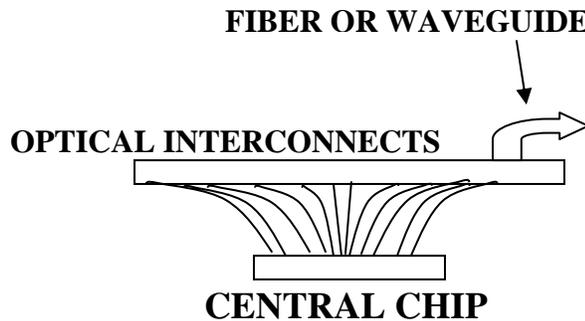


Figure 2: Schematics of a proposed solution: The central chip is in the bottom. Through plasmonic nanowaveguides each of the on-chip memory modules will be connected to the optical interconnects interface in which each such nanowaveguide will convert to more standard waveguides or fiber. Each of the latter waveguides will be orders of magnitude thicker. The flower arrangement expands the optical interconnect interface to allow more space for the thicker connections. Each of the latter waveguides will connect to a memory chip.

Two other multi-processing organizations are also considered:

1. In Figure 3(a), a separate chip comprises the first level of the memory hierarchy of each memory module. The processing clusters as well as the interconnection network connecting processor clusters and memories reside on the central chip. Incorporating this into a "flower arrangement" should be similar to the incorporation of Figure 1. The appeal of such a partition of components into chips relative to the one in Figure 1 above is that it allows devoting more chip real estate to the components remaining on chip as well as a different way for handling the power budget..
2. In Figure 3(b) a separate chip comprises each of the processing clusters, as well as each of the memory chips. The interconnection network connecting processor clusters and memories will still fully reside on the same central chip. The rationale for this organization is it appears that the interconnection network could still benefit from being located on a single chip. Recall that this assumes that the interconnection network is best implemented using an all-electronic chip as electronics appears to be best suited to handle fast heavily pipelined switching and routing of small packets. Incorporating this into a "flower arrangement" needs to take into account the extra number of external connections to processing cluster chips. One obvious thought would be to have optical wires to processing cluster chips connect through the bottom of the interconnection network chip, while optical wires to memory chips connect through the top of the chip. The appeal of such an organization is that it could allow a more effective way for building massively parallel machines.

Since the number of features that can fit on a single chip is limited, this will allow building larger machines (using more chips) while still using powerful single-chip components, such as an interconnection network. Power requirements can also be handled better using such an organization.

Culler and Singh [12] wrote that “to truly design a machine that can look to the programmer like a PRAM” could be a breakthrough for parallel computing. The PRAM-On-Chip approach that suggested how smaller scale machines could be designed in a way that can look to the programmer like a PRAM already demonstrated an opportunity for such a breakthrough. However, coupled with the current paper a scalable and more complete breakthrough can be derived. This would also complete a rebuttal to the 1993 LogP paper [13], which claimed that the PRAM model is unrealistic and should not even be taught.

A far reaching question is whether greater memory scalability of the overall architecture that the current paper points to could reach a point where compact designs (along the lines of the current paper) could outperform the most expensive and largest supercomputers built today for many more applications . This includes machines built out of thousands of chips, such as the machines at the Sandia National Laboratory. As already noted several times before in the current paper, a key advantage of our various proposed approaches is that they all facilitate a much easier way to program parallel computers.

The challenges to the solution presented in the next section involve: optimization of electronic to plasmonic on-chip conversion in terms of conversion efficiency and plasmonic scattering, and optimization of coupling between plasmonic and regular waveguides. The right metric for measuring performance improvement should be the sustainable bandwidth and latency of the connection to the memory chips. The proposed application should be able to drastically improve the performance of on-chip multiprocessors: (i) the area currently needed for the same interface (e.g., for I/O pad using bonding flip chips) could be devoted to processors (and functional units), interconnection network or on-chip memory, say in the structure of Figure 1; (ii) better bandwidth and latency to memory chip may allow moving more of the memory to memory chips (in Figure 3(a)). Alternatively, moving processing clusters to separate chips (in Figure 3(b)) would allow devoting even more area to the interconnection network that would remain on the central chip.

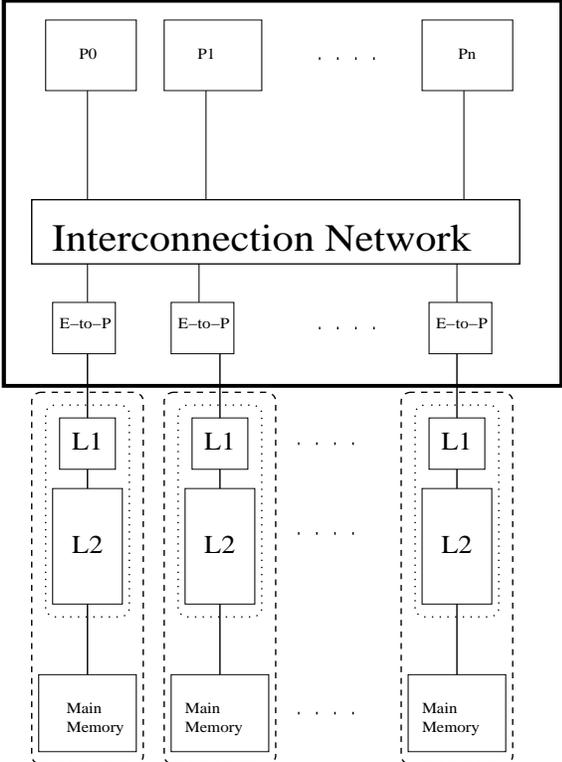


Figure 3a: On-chip multi-cores

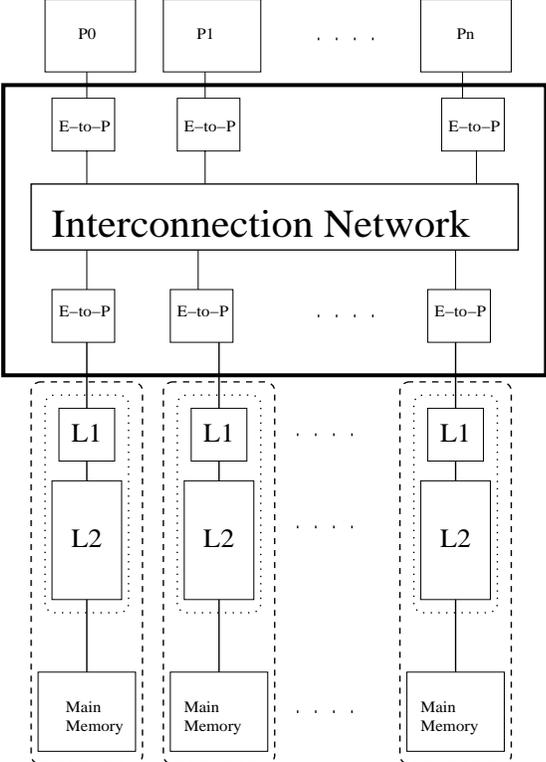


Figure 3b: Multi-chip multi-processor

4. PLASMONICS FOR IMPROVED INTER-CHIP BANDWIDTH AND LATENCY

4.1 Introduction to plasmonics: what is surface Plasmon?

“Plasmonics” is an emerging partner technology to electronics and photonics, based on surface-plasmon polaritons [1], which are commonly referred to as “plasmons”. These are classical charge density waves excited at a metal dielectric interface. They travel at a speed, which is close to the speed of light and can be guided by very small structures. These structures are much smaller than are conventionally used for the guiding of optical waves. Since plasmonic devices rely on the integration of metallic and dielectric nanostructures, their fabrication is compatible with a CMOS silicon-based fabrication technology. Gold, silver and copper nano-wires, nano-dots, and nano-dot arrays, as well as layered gold-dielectric structures support the excitation and propagation of plasmons.

Schematic views of two possible plasmonic nanowaveguides are presented in Figure 4. These waveguides may have physical dimensions that are at least an order of magnitude smaller than the dimensions of regular optical waveguides (<100 nm). In these plasmonic nanowaveguides the optical energy is guided via metal-dielectric interface in Figure 4a, or interfaces in Figure 4b. The fact that the wavelength

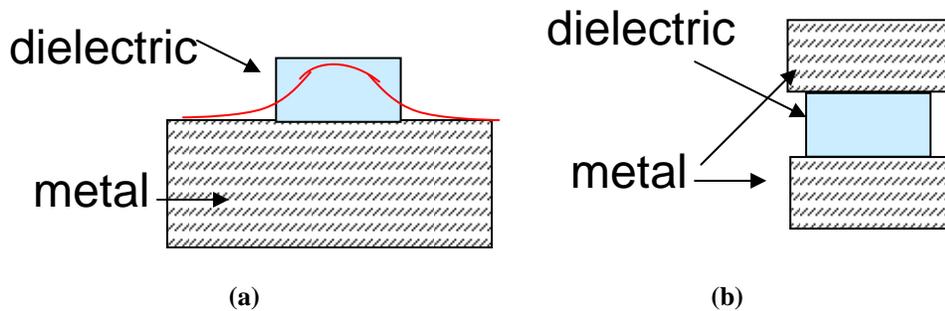


Figure 4: Two possible geometries of a surface plasmon waveguide

of surface plasmon propagating over the metal-dielectric interface may be much shorter than the wavelength of free space photons [1] indicates that there is no waveguide cutoff down to 30-50 nm waveguide dimensions. A waveguide shown in Figure 4a has been realized in our recent paper [2], in which a detailed description of issues related to fabrication, waveguiding, and energy coupling may be found. The main conclusions of [2] are that the plasmonic nanowaveguides are CMOS-compatible, and that they allow at least an order of magnitude compactification of regular optical interconnects and waveguiding devices. Another important result of [2] is that simple and practical ways for coupling of optical energy in and out of these nanowaveguides have been demonstrated.

A channel plasmonic waveguide shown in Figure 4b is basically a combination of two waveguides shown in Figure 4a. From the outside it looks very much like a metallic nanowire. We should emphasize that if the transverse dimensions of these waveguides are increased gradually up to a factor of 10, these waveguides can become regular optical waveguides, which are capable of long-range guiding of optical energy. This option would be very valuable for the “flower arrangement” geometry described above: we anticipate that these waveguides would start as plasmonic nanowaveguides at the central chip and will be gradually converted to regular waveguides to form the regular optics interconnect level (see Figure 2).

4.2 From electronics to plasmonics

Incorporation of plasmonic nano-waveguides into an electronic chip will have to involve fast and power-efficient conversion of electronic signals to optical ones and vice versa. While conversion of plasmonic signals to electronic ones is no different than the usual optical to electric signal conversion (such CMOS compatible converters have been built, even though not yet at 100Gbps), the electronic to optical conversion is more complicated. If regular optical waveguides or fibers were used, they would occupy considerable space on the chip. Microstrip connections reduce bandwidth, and become lossy at small scales. We propose to implement the dielectric plasmonic waveguides described above, which are much more compact. We have demonstrated in recent experiments that such waveguides may be as small as 100 nm

wide or even smaller [2]. To excite plasmons wherever they are needed for data transport either intra-or inter-chip, we propose the direct conversion from electronic to plasmonic propagation using inelastic electron tunneling [3]. This mechanism of direct conversion of electronic signals into plasmonic ones has been demonstrated experimentally in refs.[3,4]. In these experiments a voltage of $\sim 2.5\text{-}3\text{V}$ applied to a tunneling junction between a metal tip and a metal sample in a scanning tunneling microscope led to the generation of surface plasmons with quantum efficiency of the order of $10^{-1} - 10^{-3}$ plasmons per tunneling electron. Thus, approximately 10^{10} plasmons per second were generated in a single tunneling junction. We anticipate that multiple tunneling junctions can be used to generate plasmons in parallel, as shown in Figure 5 (we should emphasize that while some photons are created by the tunneling junction, the plasmon channel dominates the inelastic electron tunneling processes [3,4])

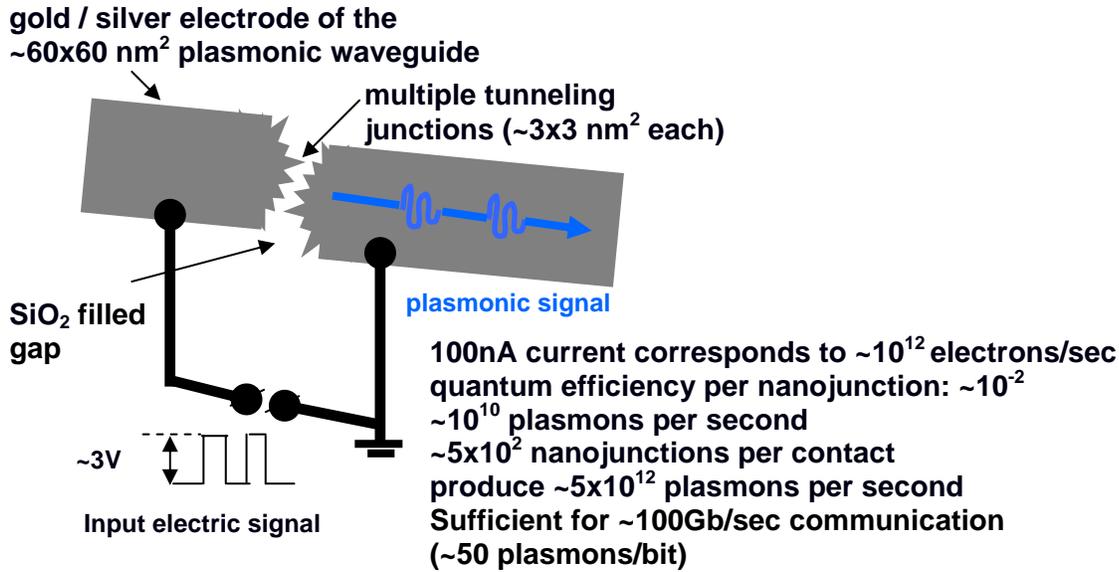


Figure 5: Proposed tunneling device for electric to plasmonic signal conversion (see description in the text)

The tunneling junction will be formed as a small dielectric-filled gap in the metal electrode of the plasmonic waveguide (Figure 5). The rough edges of the electrode will form multiple tunneling junctions so that $\sim 5 \times 10^2$ (note that this is an approximate figure: $60^2/3^2$ is 400, not 500) individual nanojunctions will be formed. As a result, $\sim 5 \times 10^{12}$ plasmons per second may be created if $\sim 3\text{V}$ electrical signal is applied to the junction. Assuming that an optical communication link may be operated at ~ 50 surface photons (plasmons) per bit, a communication rate of $\sim 100\text{ Gb/sec}$ may be achieved. This rate of communication is good enough to realize the parallel computer architectures described above. The directionality of the plasmonic signal will be determined by the direction of the plasmonic waveguide and the sign of the input electric signal. As the current figure of 100nA in Figure 5 suggests the power consumption will be minimal. However, it remains to be studied how the noise in the incoherent tunneling plasmon source would affect the overall performance of the electric to plasmonic signal converter.

4.3 Coupling of plasmon optics with normal 3D optics

Optical interconnects between chips in multi-chip systems, for intra-chip and inter-chip data communication, offer very high data rates, potentially to well beyond 100Gb/s . They provide connections with zero or minimal parasitic capacitance, and can provide very directional data flows, so crosstalk between multiple interconnects can be avoided. Such optical connections can be free-space, use optical fibers, or can be guided by on-board waveguides. The main bottleneck in application of optical connections to parallel computer architectures described above is the CMOS incompatibility of traditional on-chip light sources and the diffraction limit of normal 3D optics: the size of normal optical guiding devices is limited by the wavelength of light (>0.5 micrometers) and they are generally an order of magnitude larger. We propose that plasmonics-based waveguides provide a solution to this problem. Since plasmon wavelength may be considerably shorter than the wavelength of photons, the diffraction limit of plasmonic devices operation is at least an order of magnitude better than the diffraction limit of normal optics. On the other hand, as we

have seen in the previous section, the on-chip sources of plasmonic signal are CMOS-compatible. In addition, plasmonic optics is easy to integrate with normal 3D optics. We have shown that surface bumps or holes, either single ones, or arrays, convert propagating plasmons into light that propagates into the far field [2]. The additional incorporation of plasmonic nano-waveguides into an integrated structure provides an interconnect pathway of high bandwidth, and occupying minimal chip area. An example of such plasmonic to optical interconnect is shown in Figure 6. The plasmon to photon conversion is achieved using a combination of a plasmonic waveguide (Figure 4), parabolic waveguide coupler, and a periodic nanohole (nanobump) array. While we conjecture that this solution is CMOS-compatible, realization of this geometry using CMOS technology is yet to be demonstrated.

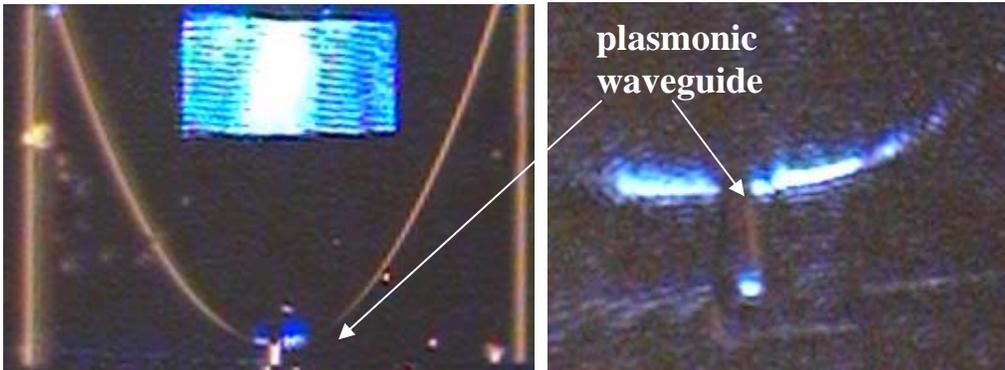


Figure 6: Plasmonic to optical interconnect: the plasmon to photon conversion is achieved using a combination of a plasmonic waveguide (Figure 4), parabolic waveguide coupler, and a periodic nanohole (nanobump) array, which appears as a bright rectangular area in the figure. The light emitted by the array is coupled into a fiber if necessary.

To demonstrate workability of plasmonic-based signal processing some other issues need to be resolved::

- 1) CMOS compatible optical detectors that would be capable of operation at the high speeds and/or with the quantum efficiency required at 100 Gbps will have to be built.
- 2) Since any optical channel (plasmonic or otherwise) is only as fast as the electrical signal driving it, there would be need to multiplex several slower electric signals either in time or in wavelength. Note also that remporal multiplexing and demultiplexing to high speeds will generally require high-speed electronic circuits (e.g., SiGe, InP HEMTs, etc.) which are not yet incompatible with standard CMOS technology.
- 3) Finally, a complete CMOS-compatible electric-->optical-->electrical link, comprised of an electrical multiplexer, source and high-speed modulator, plasmon waveguide, and optical detector with transimpedance amplifier, and electrical demultiplexer should be demonstrated in an experiment and its performance tested.

5. CONCLUSION

Motivation for the contributions in the current paper is mostly provided by references to previous, or related attempts, to develop an architecture that can look to the programmer like a PRAM, or to statements that this would be desirable. We end this paper with a more concrete motivation. The recent paper [7] presented a parallel gate level logic simulator implemented on an XMT platform and studied its performance. Test results demonstrated potential for achieving more than a hundred-fold speedup over a serial implementation. The results were achieved for the popular ISCAS89 benchmark suite. The largest circuit from ISCAS89 benchmark-[3] consumes less than 3MB memory. In other words, the case study in [7] is limited by the assumption that all the memory needs can be satisfied by on-chip resources. The type of architecture enhancements described in the current paper will allow sufficient bandwidth across chip boundaries. A closer inspection of the simulation results in [7] reveals that the hundred-fold speedups demonstrated for ISCAS89 circuits should extent to gate level simulations for circuits of any size, if the memory of the simulating computer would become large enough (assuming sufficient bandwidth and latency, which is not excessively high). Once demonstrated,

this may provide a “killer application” for our PRAM-like architectures. More speedup results from a previous paper [6] are given in Figures 7 and 8. They could also be extended to much larger problem sizes. The relevant assumptions made there about the memory architecture, such as latencies and bandwidth would apply to the new setting.

This type of results also suggests a more general yet powerful possibility for PRAM-related architectures: use an existing easy-to-program API, such as VHDL or Verilog, for reduced application-software development time and better performance over serial performance-driven languages, such as C.

While plasmonics, and optics in general, should be useful for other architectures as well, application of plasmonics for PRAM-related ones is of particular interest, since high bandwidth is so crucial for them.

*vishkin@umd.edu; phone 1 301 405-6763; fax 1 301 314-9658; www.umiacs.umd.edu/~vishkin. This work was partially supported by National Science Foundation grant CCF-0325393

REFERENCES

1. A.V. Zayats, I.I. Smolyaninov, and A. Maradudin, *Physics Reports*, 408, 131-314 (2005), "Nano-optics of surface plasmon-polaritons".
2. I.I. Smolyaninov, Y.J. Hung, and C.C. Davis, *Appl.Phys.Letters* 87, 241106 (2005) "Surface plasmon dielectric waveguides".
3. I.I.Smolyaninov, M.S.Khaikin and V.S.Edelman, *Phys.Letters A*, 149, 410-412 (1990), "Light emission from the tunneling junction of the STM."
4. I.I.Smolyaninov, V.S.Edelman and V.V.Zavyalov, *Phys.Letters A*, 158, 337-340 (1991), "Spectroscopic measurements of light emitted by the STM".
5. U. Vishkin, S. Dascal, E. Berkovich and J. Nuzman. Explicit Multi-Threading (XMT) Bridging Models for Instruction Parallelism (Extended Abstract). In *Proc. 10th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 1998.
6. D. Naishlos, J. Nuzman, C-W. Tseng, and U. Vishkin. Towards a First Vertical Prototyping of an Extremely Fine-Grained Parallel Programming Approach. Invited Special Issue for ACM-SPAA01: TOCS 36,5 pages 521-552, Springer-Verlag, 2003.
7. P. Gu and U. Vishkin. Case Study of Gate-level Logic Simulation on an Extremely Fine-Grained Chip Multiprocessor *Journal of Embedded Computing*, Special Issue on Embedded Single-Chip Multicore Architectures and related research - from System Design to Application Support, to appear.
8. R. Alverson, D.Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith. The tera computer system. In *Proc. Int. Conf. On Supercomputing*, pages 1--6, 1990.
9. P. Bach, M. Braun, A. Formella, et al. Building the 4 processor SB-PRAM prototype. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, volume 5, pages 14--23, Jan. 1997.
10. A. Gottlieb, R. Grishman, C. Kruskal, K. McAuliffe, L. Rudolph, and M. Snir. The NYU ultracomputer--designing an MIMD shared memory parallel computer. *IEEE Trans. Comput.*, pages 175--189, Feb. 1983.
11. K.Mehlhorn and U.Vishkin. Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories. *Acta Informatica* 21:339--374, 1984.
12. D.E. Culler and J.P. Singh. *Parallel Computer Architecture*. Morgan Kaufmann, 1999.
13. D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R.Subramonian, T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation, *Proc. Principles and Practice of Parallel Programming*, 1-12, 1993.
14. S. Baase. *Computer Algorithms: Introduction to Design and Analysis*, 2nd edition, Addison-Wesley, 1988.
15. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*, MIT Press, 1990.
16. U. Manber. *Introduction to Algorithms - A Creative Approach*, Addison Wesley, 1989

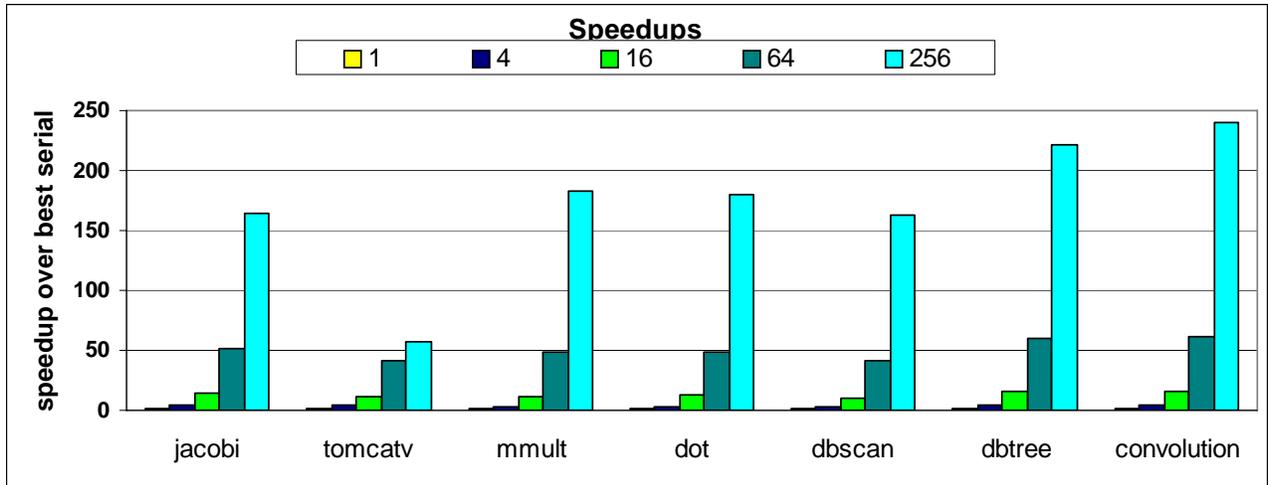


Figure 7: Speedups for some applications (256 processors in 32 processor clusters)

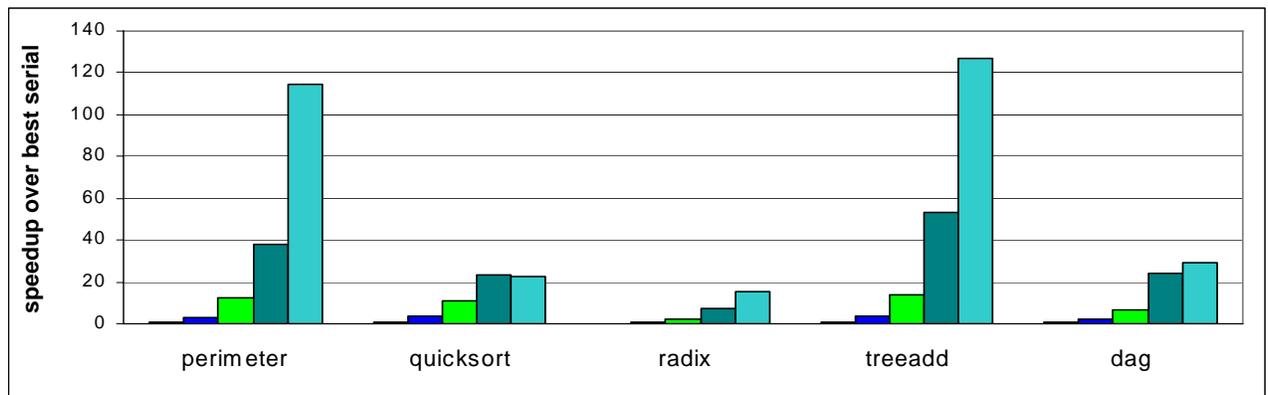


Figure 8: Speedups for some irregular applications (256 processors in 32 processor clusters)