I'll define terms used in this paper even though some should be obvious to you. - JT Wunderlich PhD
ALU = Arithmetic Logic Unit
FPU = Floating Point Unit
GPU = Graphics Processing Unit

DOI: 10.14529/jsfi170203

Simultac Fonton: A Fine-Grain Architecture for Extreme Performance beyond Moore's Law

Maciej Brodowicz[1], *Thomas Sterling*[1]

With nano-scale technology and Moore's Law end, architecture advance serves as the principal means of achieving enhanced efficiency and scalability into the exascale era. Ironically, the field that has demonstrated the greatest leaps of technology in the history of humankind, has retained its roots in its earliest strategy, the von Neumann architecture model which has imposed tradeoffs no longer valid for today's semiconductor technologies, although they were suitable through the 1980s. Essentially all commercial computers, including HPC, have been and are von Neumann derivatives. The bottlenecks imposed by this heritage are the emphasis on ALU/FPU utilization, single instruction issue and sequential consistency, and the separation of memory and processing logic ("von Neumann bottleneck"). Here the authors explore the possibility and implications of one class of non von Neumann architecture based on cellular structures, asynchronous multi-tasking, distributed shared memory, and message-driven computation. "Continuum Computer Architecture" is introduced as a genus of ultra-fine-grained architectures where complexity of operation is an emergent behavior of simplicity of design combined with highly replicated elements. An exemplar species of CCA, "Simultac" is considered comprising billions of simple elements, "fontons", of merged properties of data storage and movement combined with logical transformations. Employing the ParalleX execution model and a variation of the HPX+ runtime system software, the Simultac may provide the path to cost effective data analytics and machine learning as well as dynamic adaptive simulations in the trans-exaOPS performance regime.

HPC = High Performance Computing

CCA = "Continuum Computer Architecture"

Keywords: High-Performance Computing, parallel processing, exascale, non-von Neumann architecture, cellular architecture.

Introduction

MPP = Massively Parallel Processing
SIMD = Single Instruction Multiple Data (i.e., a Flynn Classification)
Vector = Vector-Register Architecture (e.g., good for Matrix Manipulation)
ILP = Instruction Level Parallelism ("Static" if Compiler decides; "Dynamic" if Processor decides)

Commercial computers have been predominantly von Neumann derivative architectures throughout the seven decades of digital electronic information processing although the enabling technologies and the logical structures have varied widely over this period. Such systems like MPPs, SIMD, vector, ILP, and multithreaded, while representing their own distinct ways of exploiting parallelism, have none the less been based on the fundamental principles of the von Neumann model of the 1940s. These include sequential instruction issue and sequential consistency, optimizing for ALU/FPU utilization as the precious resource, and separation of processing and memory. At one time this was the rational objective function as an FPU in the enabling technology of their respective times was the most expensive component in discrete components, size, cost, and power; thus, driving its throughput as most important for performance to cost. Even when this was less so, while Moore's Law dominated, component capacity and performance grew exponentially with time in conjunction with Dennard scaling [7] and ILP through incremental extensions of conventional practices and architecture. Since 2005, processor core speeds have not grown and overall system performance has been improved only through increased multiplicity of cores either through multicore/manycore architectures such as the Intel Xeon Phi or the higher order core structures like the NVIDIA family of GPUs. This will likely deliver an ExaFLOPS HPL R_{max} performance after 2020 but at significant cost and low efficiency. Further emerging applications of importance in data analytics and machine learning among others will demand very different design structures, balance, and programming methodologies. It is a

FLOPS = Floating Point OPerations per Second

HPL Rmax is a High Performance Computing Benchmark measure

[1]Center for Research in Extreme Scale Technologies, Indiana University, Bloomington, USA

premise of this paper that the future of high end computing exploiting CMOS nano-scale device technologies will require new classes of computer architecture to take full advantage of component capabilities through avoidance of many of the bottlenecks imposed by the von Neumann architecture model. One example of active interest and research pursuit is neuro-morphic architectures that are brain-inspired such as neural nets. The research discussed here, Continuum Computer Architecture (CCA), is proposed to reverse the stagnation of von Neumann architecture and dramatically increase key properties of scalability and memory bandwidth while dramatically reducing size, power, and cost. This paper establishes the fundamental principles of the future class CCA systems.

CCA is motivated by the end of Moore's Law at nanoscale semiconductor feature size and the opportunity for new architectures that exploit rational optimization strategies of current enabling technologies as opposed to conventional practices based on the legacy of the venerable von Neumann architecture model. An examination of conventional core architectures based on the three major contradictions imposed by the von Neumann model are the emphasis on FPU utilization, the separation of processor and memory, and the limited way in which parallelism is exposed and exploited due to sequential instruction issue. Much of the die area is dedicated to emphasizing the FPU/ALU utilization including speculative execution, branch prediction, out of order completion, and multiple cache layers and their control. The principal metric of operation should be the time delay between when an operation is logically ready to be executed (that is satisfies its precedent constraints) and when it is actually performed. Eliminating the von Neumann bottleneck associated with the latency and bandwidth constraints for memory access has been long recognized as an essential goal but current cache based techniques demand data reuse through temporal and spatial locality. The typical practice of organizing parallel computation through the BSP style [21] with global barriers imposes potentially severe bottlenecks in core usage with irregular tasks. Furthermore, both user productivity and performance portability are hampered by the myriad details of performance optimization resulting from the complexity of contemporary core designs. Ultimately, future architectures across the exascale performance regime will depend how they address the key fundamental operational factors of starvation, latency, overhead, and contention at every level with effective use of parallelism probably most critical.

This paper departs from the norm by considering these key issues and examining an architecture design space that falls into the broad family of non-von Neumann architectures to eliminate the deleterious effects of the legacy of von Neumann based machines. It further presents a genus of innovative architectures, Continuum Computer Architectures, that borrows from early consideration of a number of alternative strategies including cellular automata [14], dataflow [8], systolic arrays [5], and more recent work on Asynchronous Multi-Task computing [10, 12, 15, 20, 23] as well as the authors' own work on the ParalleX execution model [1, 11]. The paper concludes with an analysis that suggests that an exascale computer employing such concepts could be devised and fabricated using contemporary CMOS semiconductor technology and managed through current experimental dynamic adaptive software techniques in a relatively cost effective way compared to current projected approaches.

1. Continuum Computer Architecture

The architecture genus of Continuum Computer Architecture is first advanced as a Gedanken-experiment with the following attributes: presume that a finite space (rectangular

for convenience) is filled with many rows and columns of computers connected in a mesh topology. Within the finite space, replace each computer with four machines, each of a quarter lower performance, data storage, and foot print so that the same area has the same capability, just realized through more and more smaller and smaller computing elements. Repeat this cycle infinitely. In the limit, every point in a finite space has the properties of logical operations, data storage, and data movement but to a minimal, actually zero amount, in each case. But a finite area, no matter how miniscule beyond a point will have non-zero values in all three parameters. A medium of computing has been created, at least of the imagination a continuum computer. Such continuum computers can actually be produced for special purposes in the analog domain such as the determination of electrical fields in a conductive fluid; not exactly programmable in a general purpose sense but it does prove the point. For a more general CCA, such a medium must be discretized as is done with many algorithmic models such as Discrete Fourier Transforms [6] or Finite Element Methods [2]. The challenge is to derive the smallest fine grain element that embodies the properties of data storage, logic, and data transfer such that when employed in collectives can accomplish real world (programmable) parallel computation. In the special case of the species of CCA described in this paper, the element is referred to as a "fonton", but this is getting ahead of the story, to which we return in future sections. It is ironic that perhaps the first example of this genus of non-von Neumann architectures was created by John von Neumann himself in 1949 in the form of the cellular automata which he proved to be Turing equivalent. Many special purpose cellular automata have been devised over ensuing years; the most famous of which is most likely Conway's Game of Life [3]. The technical strategy of the CCA is summarized as comprising the following elements:

1. Non von Neumann Architecture – to eliminate the bottlenecks imposed by legacy structures.
2. Optimizing for most expensive property, not FPU utilization – emphasizing today's crucial bottlenecks such as memory bandwidth and latency rather than the obsolete notion of FPU utilization.
3. Cellular Structures -- maximizing best usage of die area through cellular structures which replaces complexity of design for complexity of operation with simplicity of design combined with massive replication.
4. Nearest Neighbor Access – exploitation of nearest neighbor communication between cells for extreme system bandwidth with minimum latency as appropriate.
5. Parallel Control Flow – the replacement of sequential issue based control flow with intrinsic parallel control flow combining dataflow and futures semantics with a high level global parallel control state.
6. Objective Function – parameter set for a multi-dimensional optimization space including memory bandwidth and latency, delay between enabled and executed actions, sustained OPS versus peak or Linpack FLOPS, and time to solution versus cost.

These strategy elements in concert establish the guidelines that govern the manifestation of continuum computer architectures for ultra-scale computing.

2. Simultac Fonton

The Simultac architecture currently investigated by the authors is but one of possible species of CCA. It stresses practical aspects of the design to promote cost-effective and efficient implementation while reducing the design time and enabling fast prototyping on FPGA technology.

The primary guiding principle is simplicity; the complexity of its component cells is significantly lower than that of a single RISC core. The desired performance level of the whole system is achieved through high level of replication of uniform component cells (fontons). The emergent properties of a large scale system manifest themselves due to synergies arising within the multitude of concurrent actions performed by many fontons. RISC = Reduced Instruction Set Comuting

Simultac design directly addresses performance degradation factors identified by the SLOWER model [16]. It supports hierarchy of parallel actions, ranging from dataflow-like interactions inside cell groups to parallel process instances, each occupying a macroscopic fraction of hardware resources. This limits starvation by identifying opportunities for parallelism extraction at multiple levels of program execution. The latency effects are suppressed by introduction of new structures, both in software and hardware that enable opportunistic scheduling of work. The overheads are minimized through development of optimized hardware mechanisms; due to simplicity of the design, the analysis of costs and implementation of such improvements are not difficult to perform. The contention is addressed by providing abundant execution resources. A critical element of reducing power consumption, proximity of data, is achieved through integration of storage and processing logic. Finally, component redundancy and high availability "anywhere in the medium" provides fault tolerance.

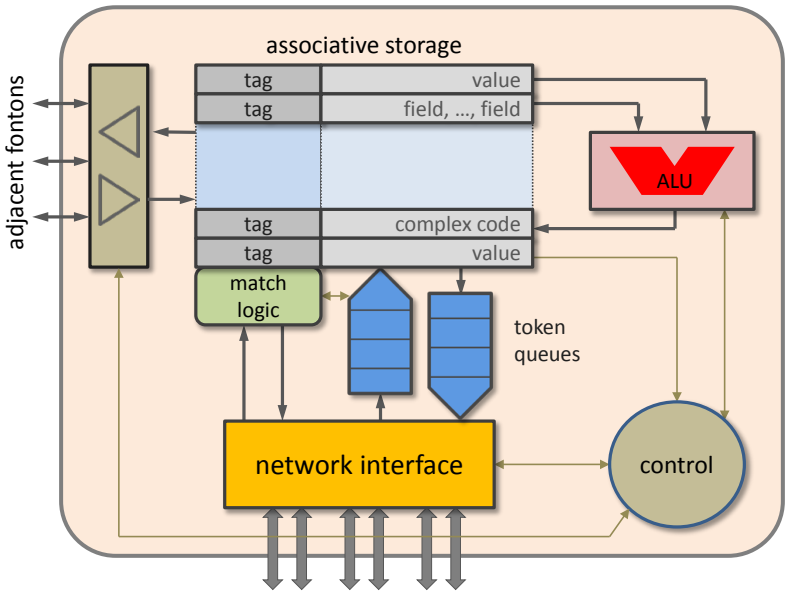


Figure 1. Internal structure of a fonton

A schematic diagram of fonton internals is shown in fig. 1. At the heart of the fonton is an associative storage array in which memory segments with capacity comparable to that of a cache line are combined with virtual tags to enable selection of the appropriate individual storage cells. Contents of the array may be transformed by an ALU controlled by logic executing locally stored strands of instructions. The (limited) code may be either kept in the memory array or delivered by an incident token (active message) from the cross-chip interconnect. The token destination address is matched by specialized logic monitoring all locally used tags. If a match is obtained, including forms of wildcard addressing, the token is absorbed by a fonton. The fonton has a dedicated adjacency interface permitting access and modification of the state of several neighbor fontons. This interface also permits realization of simple but high aggregate bandwidth massaging, alleviating the load of on-chip network if only intra-neighborhood communication is

necessary. The majority of fonton operations are performed within a single clock cycle eliminating elaborate pipelines and reducing the processing latency.

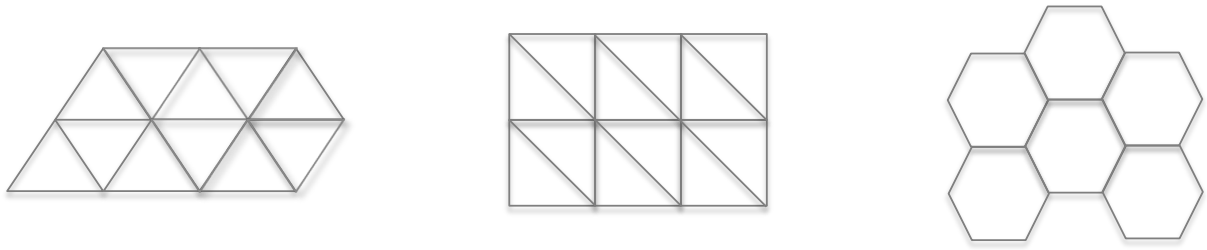


Figure 2. Some of the possible 2D tessellations

Scaling to chip size involves embedding a large number of fontons on a die. For this purpose, die area has to be tessellated using uniform shapes to completely fill the available space. This also determines the number of adjacent fontons involved in the nearest-neighbor interactions. Some of the considered tessellations are illustrated in fig. 2. The resultant geometry directly impacts the types of parallel processing that can be performed in local neighborhoods. They range from simple 1D pipelines, forked pipelines, overlapping multi-directional pipelines, trees and DAGs to arbitrary graphs. Determining the tradeoffs between the required resource count (such as number of links per cell with associated control logic) and the ability to emulate a broad range of structures required to represent data and/or control flow relationships is one of the subjects of the ongoing research.

3. Parallel Control Flow

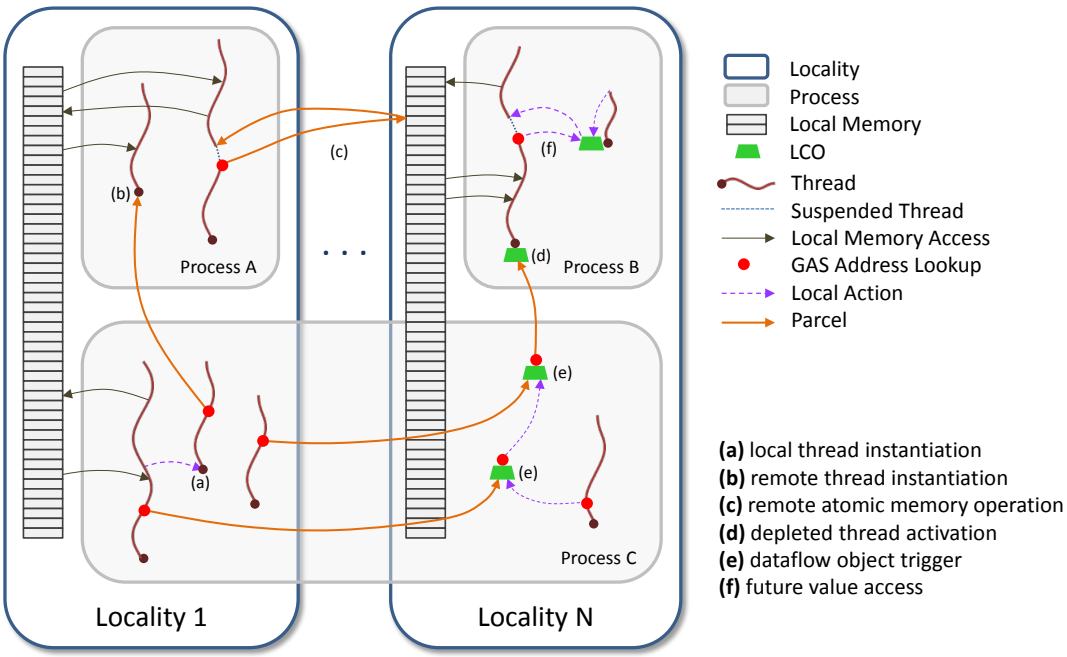


Figure 3. Semantic constructs of ParalleX enabling various types of parallel control flow

The CCA/Simultac architecture described thus far satisfies the need to define organization of primitive components. But the von Neumann architecture also provides the semantics of computing; the execution model that governs the logic of operations and the meaning of the

objects on which they perform as well as the order in which they are performed. Continuum Computer Architectures in general and the Simultac architectures in particular also require an execution model to transform the local rules and state to global general purpose computation, meeting the criteria of success required of this non von Neumann approach. The experimental ParalleX execution model serves this purpose as an abstraction to define and guide the semantics of computing implemented as a version of the HPX family of runtime systems [18, 19]. This also provides a low-level application programming interface (API) or a target for high-level language compilers. There are many aspects of the ParalleX execution model and its HPX embodiment that distinguish it from the classical von Neumann model, but a few functional constructs create the framework for the rest of the parallel operation. These include the following:

- Global name space – data, control, and program objects all exist in a single but hierarchical name space that permits virtual objects to migrate across physical space without requiring name change.
- Spanning processes – first class ephemeral contexts that include data, control state, task instantiations, child processes, and resource mapping that span multiple physical work units like conventional SMP nodes. They define the hierarchical name space.
- Compute complexes – the principal form of executing instantiated tasks that operate within a defined locality (contiguous physical space of bounded response time and guaranteed compound atomic sets of operations) that performs the work of the application.
- Parcel-driven computing – a form of active messages that move work to the data as opposed to always moving data to the work to reduce latencies and their effects while managing asynchronous operation of distributed systems.
- Local control objects – small objects that contain and transform control state a graph of which provides global parallelism control and continuations. Rich semantic constructs like dataflow and futures (actor's model [9]) provide a rich array of asynchronous control for managing the progress of complexes and the creation of new ones.

The HPX runtime systems manage the overall resource management and task scheduling of a ParalleX program anticipated for future CCA and Simultac application programs. HPX-5 is a recent experimental runtime system embodying many of these policies on conventional parallel computers for many applications in use today. Challenges in the area of overhead of runtime control can be minimized by architectural features within the fontons. This represents future research.

4. ExaOPS System Architecture

The proposed architecture may be implemented using the currently available CMOS technology (fig. 4). To lower the production costs, die sizes that result in best balance between the effective silicon area, yield, and ability to support the required number of I/O leads are selected, typically in the vicinity of $100\,\mathrm{mm}^2$. In many cases the yield may be improved further since chips containing a few damaged fontons need not be discarded. The practical fonton counts are expected to reach about 10,000 per die. The required packaging density may be achieved by stacking dies on top of each other and connecting them using Through-Silicon Vias [4] to provide high local interconnect bandwidth while sacrificing relatively minor fraction of usable silicon area. The bottom dies incorporate high-speed serial transceivers to enable communication be-

tween different stacks on the same board. For longer distances, integrated silicon photonics [17] may be deployed to allow the use of more efficient fiber optics.

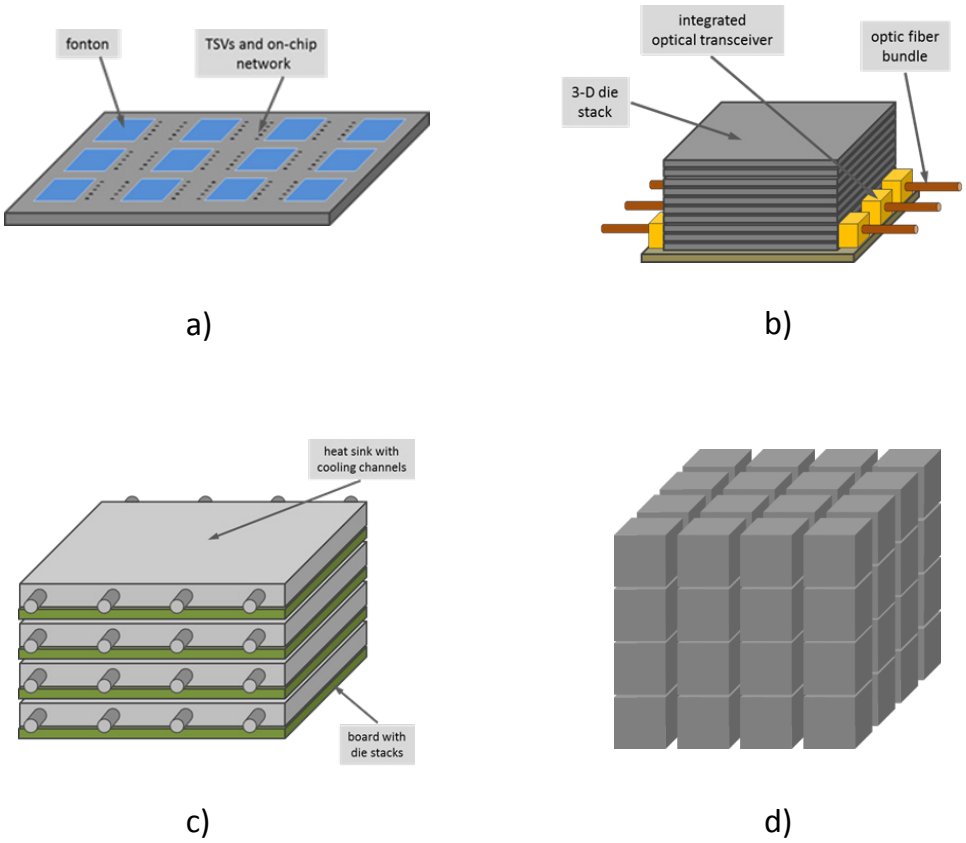
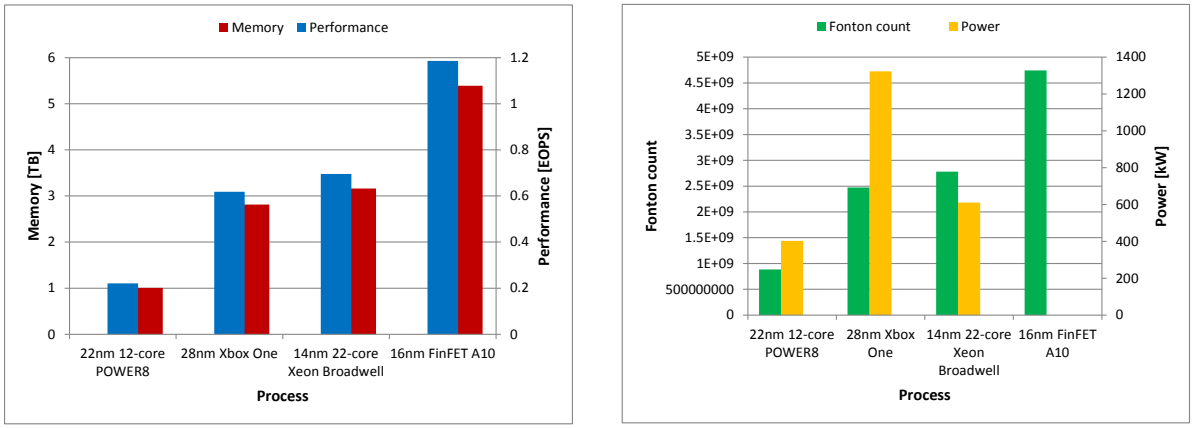


Figure 4. Simultac system architecture: a) single CMOS die, b) 3D die stack, c) board stack with integrated cooling, and d) exascale machine

The board space is mainly consumed by few thousands of die stacks. The remainder is occupied by voltage regulators and other conventional circuitry such as clock and reset generators, system monitors, etc. Relatively dense packaging will result in substantial heat dissipation thus preventing the use of conventional forced-air cooling. To cope with that, heat sinks with integrated cooling channels may be applied. This also permits the boards to be densely stacked as illustrated in fig. 4c. Finally, the complete system incorporates a number (64 shown) of board stacks arranged uniformly in three dimensions to minimize the length of cables connecting the individual units.

To estimate the resources needed to implement a Simultac system of exascale capability, a number of (rather conservative) assumptions have been made. To limit the power consumption, the clock frequency has been reduced to 250 MHz. The logic of a single fonton was estimated at the generous 16,000 gates, excluding memory. 50% of die resources are used for storage, while the logic and interconnect consume 40% and 10%, respectively. Each memory data cell uses 12 transistors per bit due to multi-ported access supporting both local operation and adjacency interfaces. The extra transistors may also be used in support circuits that improve noise margins in storage cells and reduce the current leakage arising as an effect of sub-1V supply voltage [13, 22], thus providing additional power saving. The assumed resource budget yields about 1 KB of useful storage per fonton, accounting for address tags and synchronization bits for individual words. The dies are stacked four-deep and the resulting stacks are spaced



a) b)

Figure 5. Comparison of one cubic meter Simultac implementation using four process technologies: a) overall performance and aggregate memory size and b) total fonton count and power dissipation. The estimated power consumption for the last technology node could not be calculated due to unavailability of reliable data

20 mm center-to-center on the boards, leaving 10% of board area for supporting circuitry. With boards mounted every 20 mm in vertical dimension, the properties of a prototype fitting into $1\,m^3$ are plotted in fig. 5 assuming transistor densities represented by four recent processor implementations: IBM POWER8, AMD APU for Xbox One, Intel Xeon Broadwell, and Apple A10. The power figures were approximated by linear scaling of the published TDP values by clock frequency and transistor count, hence do not take correctly into account static power dissipation and possible additional savings due to reduction of supply voltage. Properties of a full-scale system using arrangement of $4 \times 4 \times 4$ cubes are contrasted in tab. 1 with the currently fastest machine on the planet, TaihuLight. While Simultac compares favorably on nearly all metrics, it has significantly worse storage capacity. This may be corrected by integrating additional DRAM dies into 3D stacks and connecting them by TSVs to achieve high data throughput.

Table 1. Comparison of major system metrics for Simultac and TaihuLight

Parameter	Simultac	TaihuLight
Clock speed	250 MHz	1.45 GHz
Processing unit count	303.6 billion fontons	83.9 million FPUs
Peak performance	76 ExaOPS	125 PetaFLOPS
Total memory	345 TB	1.28 PB
Aggregate memory bandwidth	1821 EB/s	5.46 EB/s
Memory size to performance ratio	0.0000044 bytes/OPS	0.01 bytes/FLOPS
Memory bandwidth to performance ratio	24 bytes/OP	0.044 bytes/FLOP
Physical footprint	$25\,m^2$	$605\,m^2$

Conclusions

This paper has described a genus of computer architectures referred to here as the "Continuum Computer Architecture" and a particular specific instance, the "Simultac", intended to deliver superior performance to cost in the trans-exaOPS performance regime. It has taken

a distinctly different direction from typical incremental changes to conventional practice most frequently pursued. In particular, CCA is a genus of architectures that are non von Neumann in form and function for the purpose of eliminating the legacy artifacts of prior art that impose potentially severe bottlenecks. These include sequential instruction issue, separation of processing logic and memory often referred to as the von Neumann bottleneck, and the emphasis on FPU utilization for which much of the remaining architecture is dedicated. With the end of Moore's Law at nanoscale feature size, the development of revolutionary architectures suggests the most promising approach to dramatic improvements to efficiency and scalability. This paper has discussed at length the strategy of the CCA and has analyzed to first order the potential peak capability of the Simultac with respect to foot print and volume.

There are many design decisions yet to be determined for the Simultac class of cellular structures. Perhaps most significant is the granularity of the fonton components. How large and what is the form factor of the fonton storage? What are the logical functions that are directly implemented in hardware of the fonton both to perform the application operations and to eliminate sources of overhead for resource management and task scheduling? One parameter to be determined is the clock rate which needs to be slow enough to permit single cycle operation of the fonton but fast enough to maximize the sustained performance of the Simultac. The communication protocol has yet to be devised that can efficiently represent the Parcel semantics while minimizing latency and power. Managing the global name space and converting the virtual address space distribution to physical routing algorithms requires refinement with emphasis on race conditions due to asynchrony in the presence of migrating objects including continuations. An exciting opportunity for academic research in architecture is the design of the fonton. Conventional processor architectures are outside the scope of small teams of researchers. But this particular class of architecture, that is the CCA genus, yields itself to small groups of developers. Prototyping with FPGAs is entirely feasible that can lead to very small chips for proof of concept in semiconductor technology. The HPX runtime system software has been developed in multiple versions and is sufficiently mature to anticipate its use in this context. But changes to the low-level backend interface will have to be produced to work with the metal. This is a very exciting time in computer architecture beyond an exascale and at the end of Moore's Law. No longer should research be constrained by incrementalism of conventional practices. The need is too great to reconsider the opportunities of the non von Neumann family of architectures. An exaOPS in a cubic meter is an extraordinary goal, but realistic in its execution.

References

1. Anderson, M., Brodowicz, M., Kaiser, H., Sterling, T.L.: An application driven analysis of the ParalleX execution model. CoRR (2011), `http://arxiv.org/abs/1109.5201`, arXiv:1109.5201v1

2. Argyris, J.H., et al.: Finite element method – the natural approach. Computer Methods in Applied Mechanics and Engineering 17–18, 1–106 (January 1979), DOI:10.1016/0045-7825(79)90083-5,

3. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning Ways for your Mathematical Plays, vol. 4. A. K. Peters Ltd. (2001-2004), ISBN:978-1568811444

4. Black, B., et al.: Die stacking (3D) microarchitecture. In: Proceedings of the 39^{th} Annual IEEE/ACM International Symposium on Microarchitecture, MICRO'06. pp. 469–479 (December 2006)

5. Borkar, S., et al.: Supporting systolic and memory communication in iWarp. In: Proceedings of the 17^{th} Annual International Symposium on Computer Architecture. pp. 70–81 (1990), DOI:10.1109/ISCA.1990.134510

6. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. 19, 297–301 (1965), DOI:10.2307/2003354,

7. Dennard, R.H., Gaensslen, F., Yu, H.N., Rideout, L., Bassous, E., LeBlanc, A.: Design of ion-implanted MOSFET's with very small physical dimensions. IEEE Journal of Solid State Circuits 9(5) (October 1974), DOI:10.1109/JSSC.1974.1050511,

8. Dennis, J.B.: Data flow supercomputers. Computer 13(11), 48–56 (1980), DOI:10.1109/MC.1980.1653418

9. Hewitt, C., Baker, H.G.: Actors and continuous functionals. Tech. rep., Cambridge, MA, USA (1978)

10. Intel Corp.: Intel Threading Building Blocks (Intel TBB) (2017), website, `http://www.threadingbuildingblocks.org`

11. Kaiser, H., Brodowicz, M., Sterling, T.: ParalleX: An Advanced Parallel Execution Model for Scaling-Impaired Applications. In: Parallel Processing Workshops. pp. 394–401. IEEE Computer Society (2009), DOI:10.1109/ICPPW.2009.14

12. Kale, L.V., Krishnan., S.: Charm++: Parallel programming with message-driven objects. In: Wilson, G.V., Lu, P. (eds.) Parallel Programming using C++, pp. 175–213. MIT Press (1996), ISBN:9780262731188

13. Kim, T.H., Liu, J., Keane, J., Kim, C.H.: A high-density subthreshold SRAM with data-independent bitline leakage and virtual ground replica scheme. In: IEEE International Solid State Circuits Conference. pp. 330–331,606. IEEE (2007), DOI:10.1109/ISSCC.2007.373428

14. von Neumann, J.: Collected Works, vol. 5, pp. 288–326. Oxford: Pergamon Press (1961), ISBN:0080095666

15. Slaughter, E., Lee, W., Jia, Z., Warszawski, T., Aiken, A., McCormick, P., Ferenbaugh, C., Gutierrez, S., Davis, K., Shipman, G., Watkins, N., Bauer, M., Treichler, S.: Legion programming system (Feb 2017), version 16.10.0, `http://legion.stanford.edu/`

16. Sterling, T., Kogler, D., Anderson, M., Brodowicz, M.: SLOWER: A performance model for exascale computing. Supercomputing Frontiers and Innovations 1(2) (2014), DOI: 10.14529/jsfi140203

17. Syrbu, A., Mereuta, A., Iakovlev, V., Caliman, A., Royo, P., Kapon, E.: 10 Gbps VCSELs with high single mode output in 1310 nm and 1550 nm wavelength bands. In: Proceedings of the Optical Fiber Communication/National Fiber Optic Engineers Conference. pp. 1–3 (February 2008), DOI:10.1109/OFC.2008.4528529,

18. The Center for Research in Extreme Scale Technologies: HPX-5 (Nov 2016), version 4.0.0 http://hpx.crest.iu.edu/

19. The Ste||ar group: HPX (July 2016), version 0.9.99, http://stellar.cct.lsu.edu/

20. Tim, M., Romain, C.: OCR, the open community runtime interface (March 2016), version 1.1.0, https://xstack.exascale-tech.com/git/public?p=ocr.git;a=blob;f=ocr/spec/ocr-1.1.0.pdf

21. Valiant, L.G.: A bridging model for parallel computation. Comm. ACM 33(8), 103–111 (1990), DOI:10.1145/79173.79181

22. Verma, N., Chandrakasan, A.P.: A 256 kb 65 nm 8T subthreshold SRAM employing sense-amplifier redundancy. In: IEEE Journal of Solid-State Circuits. pp. 141–149. IEEE (2008), DOI:10.1109/JSSC.2007.908005

23. Wilke, J., Hollman, D., Slattengren, N., Lifflander, J., Kolla, H., Rizzi, F., Teranishi, K., Bennett, J.: DARMA 0.3.0-alpha specification (March 2016), version 0.3.0-alpha, SANDIA Report SAND2016-5397