Electronic Design Automation with Graphic Processors: A Survey

Electronic Design Automation with Graphic Processors: A Survey

Yangdong Deng

Tsinghua University
China
dengyd@tsinghua.edu.cn

Shuai Mu

Tsinghua University
China
mus04ster@gmail.com

now
the essence of knowledge

Boston – Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is Y. Deng and S. Mu, Electronic Design Automation with Graphic Processors: A Survey, Foundations and Trends® in Electronic Design Automation, vol 7, nos 1–2, pp 1–176, 2013

ISBN: 978-1-60198-680-1

Foundations and Trends® in Electronic Design Automation

Volume 7 Issues 1–2, 2013

Editorial Scope

Foundations and Trends® in Electronic Design Automation will publish survey and tutorial articles in the following topics:

- System Level Design
- Behavioral Synthesis
- Logic Design
- Verification
- Test
- Physical Design
- Circuit Level Design
- Reconfigurable Systems
- Analog Design

Information for Librarians
Foundations and Trends® in Electronic Design Automation, 2013, Volume 7, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Foundations and Trends® in
Electronic Design Automation
Vol. 7, Nos. 1–2 (2013) 1–176
© 2013 Y. Deng and S. Mu
DOI: 10.1561/1000000028

Electronic Design Automation with Graphic Processors: A Survey

Yangdong Deng[1] **and Shuai Mu**[2]

[1] *Institute of Microelectronics, Tsinghua University, Beijing, 100084, China, dengyd@tsinghua.edu.cn*
[2] *Institute of Microelectronics, Tsinghua University, Beijing, 100084, China, mus04ster@gmail.com*

Abstract

Today's Integrated Circuit (IC) architects depend on Electronic Design Automation (EDA) software to conquer the overwhelming complexity of Very Large Scale Integrated (VLSI) designs. As the complexity of IC chips is still fast increasing, it is critical to maintain the momentum towards growing productivity of EDA tools. On the other hand, single-core Central Processing Unit (CPU) performance is unlikely to see significant improvement in the near future. It is thus essential to develop highly efficient parallel algorithms and implementations for EDA applications, so that their overall productivity can continue to increase in a scalable fashion. Among various emergent parallel platforms, Graphics Processing Units (GPUs) now offer the highest single-chip computing throughput. A large body of research, therefore, has been dedicated to accelerating EDA applications with GPUs. This monograph is aimed to develop a timely review of the existing literature on GPU-based EDA computing. Considering the substantial diversity of VLSI CAD algorithms, we extend a taxonomy of EDA computing patterns, which can be used as basic building blocks to construct complex EDA applications. GPU-based acceleration techniques for these patterns are then reviewed. On such a basis, we further survey recent works on building efficient data-parallel algorithms and implementations to unleash the power of GPUs for EDA applications.

Categories and Subject Descriptors: J.6 [**Computer-Aided Engineering**] — Computer-aided design (CAD).

General Terms: Algorithms, Design, Performance

Additional Keywords and Phrases: Electronic Design Automation (EDA), VLSI, GPU, Graphics Processor, GPGPU, logic simulation, circuit simulation, matrix, linear algebra, sparse matrix, graph traversal, graph algorithm, dynamic programming, simulated annealing, structured grid

Contents

1

Introduction

As the foundation of information technology, Integrated Circuits (ICs) are playing a fundamental role in our society. In the foreseeable future, IC technology will still be one of the major enablers for sustainable development. To further improve the working efficiency and living standards of the human beings, the number of ICs deployed around the world will still be rapidly increasing in the future. It is predicted that 15X more transistors are going to be deployed in the next 5 years to “manage, store, and interpret data” [194].

At the same time, the complexity of ICs has been growing as indicated by Moore’s law to maintain the momentum towards increasing performance and functionality. Today, it is already feasible to integrate over 7 billion transistors on a consumer IC chip [187]. To conquer the overwhelming complexity of modern ICs, circuit designers depend on Electronic Design Automation (EDA) software to convert a design intention into working silicon. EDA tools, therefore, have to be scalable with the growing IC complexity, so that the design turnaround time can be kept in a reasonable level. Current EDA tools are facing challenges from two ends, big system and small physics [216]. The former means the integration of a whole hardware/software system onto a single chip, while the latter involves the manufacturability, reliability, and other issues incurred by the shrinking physical size of IC fabrication processes. Both trends pose significant requirements to the processing throughput of EDA software.

In the past, the performance scalability of EDA tools had always been the result of two interacting factors, smarter algorithms and faster CPUs. The latter factor is especially handy because the same EDA algorithm automatically runs faster on a CPU with higher performance. In early 2000s, however, single-core CPU performance is saturating due to the inability to extract more instruction-level parallelism and improve power efficiency. Such a stall in computing performance had serious implications on the design turnaround time of IC design projects. Given the complexity of today's IC designs, the runtime of EDA applications can still be excessive even using the best algorithm to date. For instance, a timing analysis will take a couple of hours to perform on a 5M-gate design. Such a runtime seriously constrains the number of optimization steps that can be conducted in a given design turnaround time, since virtually every post-synthesis optimization operation requires a run of timing analysis to validate the correctness. A runtime of a few hours suggests that only a small portion of the complete solution space can be explored and the design quality has to be relaxed. Another example is the circuit simulation problem. Given a Giga-Hertz phase-lock loop (PLL) circuit, a transient analysis needs to simulate the circuit for millions of cycles before the frequency can be stabilized. Thus a complete run will take months to finish on a single CPU. Besides, the continuously shrinking market window of today's electronic appliances also poses challenging requirements to the productivity of EDA software.

In spite of the relative saturation of single-core CPU performance in the conceivable future, the semiconductor processes are still offering continuously growing integration capacity. As a result, all major CPU vendors switched to offer multi-core products since 2006. Multi-core processors are inevitably becoming the dominant computing platform for EDA applications. Accordingly, it is crucial to develop parallel solutions to EDA software such that the momentum of function increase in VLSI designs can be maintained [46].

In the past few years, major EDA vendors proposed R&D initiatives to take advantage of the computing power of multi-core processors [220]. At the present time, the POSIX threads or Pthreads [115] based multithreading has been the most popular programming model for multi-core CPUs. Multithreaded versions of cutting-edge EDA software have already been released. Such applications include parallel circuit simulator (e.g., [42, 240]), router (e.g., [241]), and physical verification (e.g., [126]). Among these, multithreaded parallel circuit simulation proves to be especially successful. Meanwhile, the academia also introduced parallel algorithms for many EDA applications (e.g., [110, 157, 170, 220]).

Despite their many successful applications, the multithreaded parallel programming model on multi-core CPUs still has serious limitations. A CPU thread is associated with a relatively high overhead in initialization, context switching, and synchronization [40]. Accordingly, P threads and similar programming models belong to the category of coarse-grain multithreading, which suggests parallel processing of tasks and/or large chunks of data of a problem. However, many complex EDA applications feature abundant fine-grain parallelism (i.e., data parallelism) exemplified by matrix and graph operations. A multi-core microprocessor at most supports a few tens of threads and cannot fully take advantage of the inherent fine-grain parallelism. In addition, the scalability of a coarse-grained multithreaded program is seriously limited by the thread management overhead. A context switching of a thread on a multi-core CPU takes a few hundreds of microseconds [145]. Generally, such an overhead will outweigh the speed-up of increasing parallelism when the number of threads is beyond a given level. A recent work showed that the performance of a highly optimized parallel logic simulator saturated at 15 threads on a 10-core CPU [201].

The above problems of multi-core processors as well as the pursuit for more computing power motivate EDA researchers and engineers to explore alternative parallel computing platforms. Recently, Graphic Processing Units (GPUs) have emerged as a new general-purpose computing platform [28, 195, 196]. GPUs were originally designed as application-specific ICs for graphics rendering. Pushed by the relentless pursuit for better visual experiences, GPUs evolved to offer both high

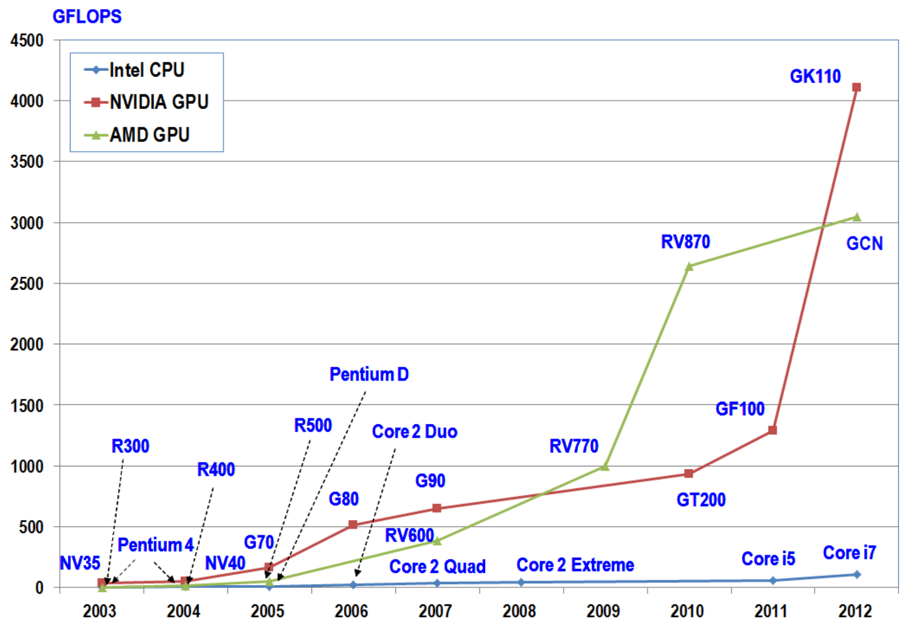

Fig. 1.1 Comparison of peak throughput of CPUs and GPUs.

programmability and superior computing throughput. In 2004, NV35 GPU began to deliver a higher level of performance than the best CPU at that time. Current GPUs outperform their multi-core CPU equivalents by a factor of over 30 in terms of peak computing throughput.

The above performance trend is depicted in Figure 1.1, where the computing throughputs of NVIDIA and AMD GPUs and Intel CPUs are compared in terms of Giga FLoating Operation Per Second (GFLOPS). We collected performance data from publically available datasheets [5, 186]. GPU chip makers usually release multiple GPUs with varying performance levels at each technology node. Meanwhile, the above three companies have different schedules for releasing new products. In Figure 1.1 we only show the “flagship” GPU for each generation and take NVIDIA’s release schedule as the time reference. Clearly, GPU has been outperforming CPU since 2004 and the performance gap is still broadening.

Along with the high computing throughput, GPUs are also equipped with a high bandwidth memory bus because it is installed on the

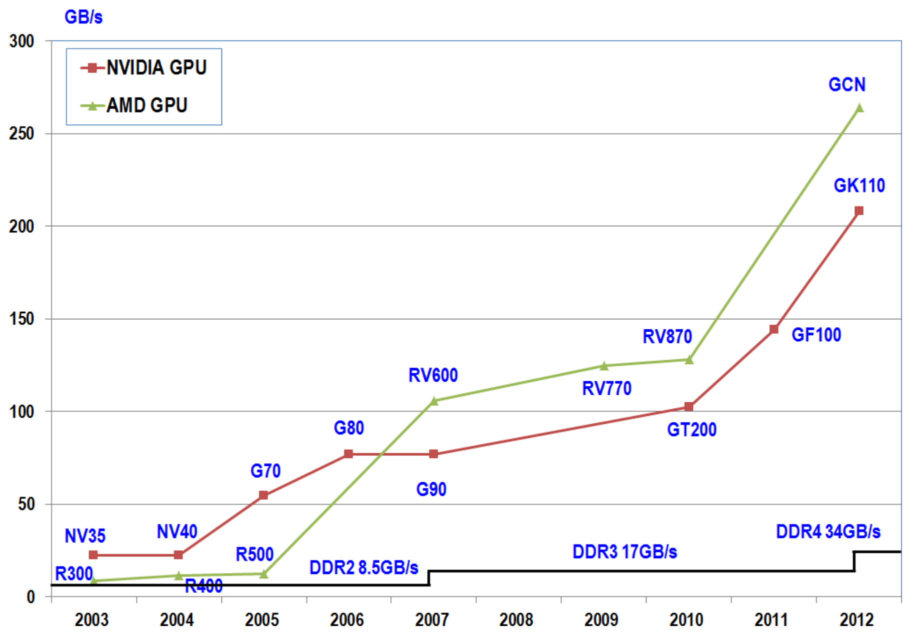

Fig. 1.2 Comparison of peak throughput of CPUs and GPUs.

graphics card and dedicated to GPU applications. The memory characteristics of major GPUs are demonstrated in Figure 1.2. The bandwidth values of four generations of DDR memories, i.e., the memory standard for CPUs, are also depicted as reference. The latest NVIDIA and AMD GPUs have a peak memory bandwidth of 208 GB/s and 264 GB/s, respectively, while the current DDR3 memory standard only supports 17GB/s (the next generation DDR4 will double the bandwidth to 34GB/s) [122]. Certainly the superior memory bandwidth of GPUs will significantly benefit memory-intensive EDA applications.

Traditionally, GPUs are programmed with shading languages like OpenGL [191]. Although OpenGL can be used for general-purpose computing on GPUs (GPGPU), the resultant programming process is laborious and error-prone. To ease the programming effort of GPGPU, NVIDIA introduced the Compute Unified Device Architecture (CUDA) technology [178, 183] so that programmers can develop GPGPU programs in a C/C++ alike language with a few extensions. While CUDA can only be used on NVIDIA GPUs, OpenCL is defined by a group of industry players as a standard cross-platform GPGPU language [134].

The synergy of GPU hardware and software has resulted in successful applications in a diverse range of scientific and engineering domains [28, 195]. On workloads with appropriate computing and memory accessing patterns, GPU can even attain a speed-up of over 100X. It is thus appealing to unleash the computing power of GPU for EDA applications.

Different CPUs, GPUs adopt a fine-grain multithreading model. Equipped with dedicated hardware for context switching, GPU threads are light-weighted and excel in massively data-parallel processing. Such an execution model makes GPU proper for EDA applications featuring data-parallelism. There is already a large body of literature presenting encouraging results on utilizing GPU to solve various EDA problems. GPGPU proved to be effective in such time consuming applications as system level design, logic simulation, timing analysis, power grid analysis, placement, and routing. The positive results suggest that the superior computing power of GPUs can be unleashed by developing carefully designed data-parallel algorithms and highly tuned implementations.

On the other hand, EDA applications pose unique challenges to the GPGPU model. The nature of circuits determines that the underlying data structures capturing IC designs tend to be irregular. Typical EDA applications are thus constructed on the basis of such irregular data structures as sparse matrix, tree, and graph.[1] The resultant memory accessing patterns are less amenable to GPUs, which only have a limited capacity of cache and assume regular memory accesses to fully utilize its large memory bandwidth. Accordingly, current works on GPU-based EDA computing generally resort to two strategies: (1) identifying regular sub-problems in an EDA application and then use GPU as an accelerator for them; and (2) re-designing or re-structuring algorithms on GPU so as to convert irregular data accesses into (at least partially) regular ones. Another challenge is that EDA applications are

[1] There exist special cases where the data structure can be quite regular. One such typical example is the power distribution network, which in many designs consists of a relatively regular power mesh.

extremely complex and cover many different domains of computations. Accordingly, an application-by-application parallelization approach can be infeasible. A viable line of attack, instead, is to identify the fundamental computing patterns and perform parallelization on them. Such a pattern-based strategy of parallel programming proves to be crucial for many other software applications [162].

In this monograph, we present an up-to-date survey on the progresses in GPU-accelerated EDA computing. Considering the high complexity of EDA applications, an essential objective of this work is to extract key computing patterns of EDA and present state-of-the-art GPU programming techniques to resolve such patterns. We believe that this approach will substantially ease the deployment of GPUs in future EDA software. This monograph focuses on using GPU to accelerate applications in the EDA domain, while the techniques also have wide applications in many other scientific and engineering domains. Interested readers please also refer to [Owens et al. 2007; Refs. [28, 195]] for surveys on applications in other disciplines.

The remainder of this monograph is organized as follows. Section 2 provides an overview of GPU hardware architectures and the corresponding data-parallel programming model. In Sections 3 and 4, we develop a taxonomy for the basic computing patterns of EDA applications and then review relevant GPU programming techniques for these patterns. In Section 5, we survey successful applications of GPU-accelerated EDA computing. In Section 6, we conclude this work and propose future research directions.

References

[1] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical path tracing — an alternative to fault simulation," in *Proceedings of Design Automation Conference*, pp. 214–220, 1983.

[2] E. Agullo et al., "LU Factorization for accelerated-based systems," in *Proceedings of International Conference on Computer Systems and Applications*, pp. 217–224, 2011.

[3] AMD, "ATI stream computing — technical overview," (http://developer.amd.com/gpu_assets/Stream_Computing_Overview.pdf), 2009.

[4] AMD, "Graphics core next architecture," http://www.amd.com/us/products/technologies/gcn/Pages/gcn-architecture.aspx, 2011.

[5] AMD, "AMD Radeon$^{\text{TM}}$ and AMD FirePro$^{\text{TM}}$ graphics cards," http://www.amd.com/us/products/Pages/graphics.aspx, 2012.

[6] M. Anderson, G. Ballard, J. Demmel, and K. Keutzer, "Communication-avoiding QR decomposition for GPUs," Technical Report No. UCB/EECS-2010-131, 2010.

[7] E. Anderson et al, "LAPACK: A portable linear algebra library for high-performance computers," in *Proceedings of Conference on Supercomputing*, pp. 2–11, 1990.

[8] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*. Springer, 2006. ISBN 0-387-22196-4.

[9] C. Augonnrt, S. Thibault, and R. Namyst, "StarPU: A runtime system for scheduling tasks over accelerator-based multicore machines," INRIA, Technique Report 7240, http://hal.archives-ouvertes.fr/inria-00467677, 2010.

[10] M. Baboulin, S. Donfack, J. Dongarra, L. Grigori, A. Remy, and S. Tomov, "A class of communication-avoiding algorithms for solving general dense linear systems on CPU/GPU parallel machines," in *Proceedings of International Conference on Computational Science*, pp. 17–26, 2012.
[11] D. L. Baggio, "GPGPU based image segmentation livewire algorithm implementation," Master Thesis of Technological Institute of Aeronautics, SãoJos'e dos Campos, 2007.
[12] M. L. Bailey, J. V. Brinerjr, and R. D. Chamnerlain, "Parallel logic simulation of VLSI systems," *ACM Computing Survey*, vol. 26, no. 3, pp. 255–294, 1994.
[13] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic Programming — An Introduction*. San Francisco, CA: Morgan Kaufmann, 1998. ISBN 978-1558605107.
[14] S. Barrachina, M. Castillo, F. D. Igual, R. Mayo, E. S. Quintana-Ortí, and G. Quintana-Ortí, "Exploiting the capabilities of modern GPUs for dense matrix computations," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 18, pp. 2457–2477, 2009.
[15] R. Barret et al., *Templates for the Solution of Linear Systems*. SIAM, 2nd ed., 1994.
[16] M. M. Baskaran and R. Bordawekar, "Optimizing sparse matrix–vector multiplication on GPUs," IBM Technical report RC24704, 2009.
[17] M. Bauer, H. Cook, and B. Khailany, "CudaDMA: Optimizing GPU memory bandwidth via warp specialization," in *Proceedings of Conference on Supercomputing*, 2011.
[18] S. Beamer, K. Asanovi'c, and D. Patterson, "Direction-optimizing breadth-first search," in *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012.
[19] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," NVIDIA Technical Report, 2008.
[20] N. Bell and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009.
[21] Y. Ben-Asher, D. Egozi, and A. Schuster, "2-D SIMD algorithms in the perfect shuffle networks," in *Proceedings of International Symposium of Computer Architecture*, pp. 88–95, 1989.
[22] Berkeley Logic Synthesis and Verification Group, "ABC: A system for sequential synthesis and verification," http://www.cad.eecs.berkeley.edu/∼alanmi/abc, 2005.
[23] A. Biere, M. Heule, H. Van Maaren, and T. Walsh, *Handbook of Satisfiability*. IOS Press, 2009.
[24] BLAS, "Basic linear algebra subprograms," http://www.netlib.org/blas/, 2008.
[25] A. Bleiweiss, "GPU accelerated pathfinding," in *Proceedings of Symposium on Graphics Hardware*, pp. 65–74, 2008.
[26] A. Bleiweiss, "Multi agent navigation on the GPU," in *Proceedings of Game Developers Conference*, 2009.
[27] G. E. Blelloch, *Vector Models for Data-Parallel Computing*. MIT Press, 1990.

[28] D. Blythe, "Rise of the Graphics Processor," *Proceedings of IEEE*, vol. 96, no. 5, pp. 761– 778, 2008.

[29] N. Bombieri, S. Vinco, V. Bertacco, and D. Chatterjee, "SystemC simulation on GP-GPUs: CUDA vs. OpenCL," in *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis*, pp. 343–352, 2012.

[30] U. D. Bordoloi and S. Chakraborty, "GPU-based acceleration of system-level design tasks," *International Journal of Parallel Programming*, vol. 38, no. 3–4, pp. 225–253, 2010.

[31] A. Boukedjar, M. E. Lalami, and D. El-Baz, "Parallel branch and bound on a CPU-GPU system," in *Proceedings of Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pp. 392–398, 2012.

[32] W. Briggs, *A Multigrid Tutorial*. SIAM Press, 1987.

[33] S. Brown, "Cadence contributes ESL methodology to TSMC reference flow," http://www.cadence.com/community/blogs/sd/archive/2010/06/11/system-realization-costs-seen-as-critical-barrier-to-ic-development-and-potentially-impacting-foundry-business.aspx, 2010.

[34] R. E. Bryant, "Simulation of packet communications architecture computer system," MIT Technical Report MITLCS-TR-188, 1977.

[35] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[36] L. Buatois, G. Caumon, and B. Levy, "Concurrent number cruncher: A GPU implementation of a general sparse linear solver," in *International Journal of Parallel, Emergent and Distributed Systems*, vol. 24, no. 3, pp. 205–223, 2009.

[37] I. Buck et al., "Brook for GPUs: Stream computing on graphics hardware," in *Proceedings of SIGGRAPH*, pp. 777–786, 2004.

[38] A. Buluç, J. R. Gilbert, and C. Budak, "Solving path problems on the GPU," *Parallel Computing*, vol. 36, no. 5–6, pp. 241–253, 2010.

[39] M. Burton and A. Morawiec, eds., *Platform Based Design at the Electronic System Level: Industry Perspectives and Experiences*. Springer, 2006 ed., 2006. ISBN 978-1-4020-5138-8.

[40] D. R. Butenhof, *Programming with POSIX Threads*. Addison-Wesley Professional, 1997. ISBN-10: 0201633922.

[41] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra, "A class of parallel tiled linear algebra algorithms for multicore architectures," *Parallel Computing*, vol. 35, no. 1, pp. 38–53, 2009.

[42] Cadence, "Virtuoso accelerated parallel simulator," http://www.cadence.com/products/cic/accelerated_parallel/pages/default.aspx, 2008.

[43] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can recursive bisection alone produce routable placements?," in *Proceedings of Design Automation Conference*, pp. 693–698, 2000.

[44] CAPS, "OpenHMPP directives," http://www.caps-entreprise.com/openhmpp-directives/, 2007.

[45] B. Catanzaro, M. Garland, and K. Keutzer, "Copperhead: Compiling an embedded data-parallel language," in *Proceedings of Symposium on Principles and Practice of Parallel Programming*, pp. 47–56, 2011.

[46] B. Catanzaro, K. Keutzer, and B.-Y. Su, "Parallelizing CAD: A timely research agenda for EDA," in *Proceedings of Design Automation Conference*, pp. 12–17, 2008.

[47] I. Chakroun, M. Mezmaz, N. Melab, and A. Bendjoudi, "Reducing thread divergence in a GPU-accelerated branch-and-bound algorithm," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 8, pp. 1121–1136, 2012.

[48] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proceedings of International Symposium on Physical Design*, pp. 185–192, 2005.

[49] K. M. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *IEEE Transactions on Software Engineering*, vol. SE-5, no. 5, pp. 440–452, 1979.

[50] B. Chapman, G. Jost, and R. van der Pas, *Using OpenMP: Portable shared memory parallel programming*. The MIT Press, 2007. ISBN: 9780262533027.

[51] D. Chatterjee and V. Bertacco, "EQUIPE: Parallel equivalence checking with GPUs," in *Proceedings of International Workshop on Logic & Synthesis*, pp. 486–493, 2010.

[52] D. Chatterjee, A. DeOrio, and V. Bertacco, "High-performance gate-level simulation with GP-GPUs," in *Proceedings of Design, Automation, and Test in Europe (DATE) Conference*, pp. 1–3, 2007.

[53] D. Chatterjee, A. DeOrio, and V. Bertacco, "Event-driven gate-level simulation with GP-GPUs," in *Proceedings of Design Automation Conference*, pp. 557–562, 2009.

[54] D. Chatterjee, A. DeOrio, and V. Bertacco, "Gate-level simulation with GPU computing," *ACM Transactions on Design Automation of Electronic Systems*, vol. 16, no. 3, 2011.

[55] D. Chen and D. Singh, "Parallelizing FPGA technology mapping using graphics processing units (GPUs)," in *Proceedings of International Conference on Field Programmable Logic and Applications*, pp. 125–132, 2010.

[56] J. W. Choi, A. Singh, and R. W. Vuduc, "Model-driven autotuning of sparse matrix-vector multiply on GPUs," in *Proceedings of Principles and Practice of Parallel Computing*, pp. 115–126, 2010.

[57] A. Choong, R. Beidas, and J. Zhu, "Parallelizing simulated annealing-based placement using GPGPU," in *Proceedings of International Conference on Field Programmable Logic and Applications*, 2010.

[58] K.-W. Chu, Y. Deng, and J. Reinitz, "Parallel simulated annealing by mixing of states," *Journal of Computational Physics*, vol. 148, pp. 646–662, 1999.

[59] J. H. Clark, "The geometry engine: A VLSI geometry system for graphics," in *Proceedings of Annual Conference on Computer Graphics and Interactive Techniques*, pp. 127–133, 1982.

[60] E. M. Clarke and O. Grumberg, "Avoiding the state explosion problem in temporal logic model checking," in *Proceedings of Symposium on Principles of Distributed Computing*, pp. 294–303, 1987.

[61] J. Cong and Y. Zou, "Parallel multi-level analytical global placement on graphics processing units," in *Proceedings of International Conference on Computer Aided Design*, pp. 681–688, 2009.

[62] R. L. Cook, L. Carpenter, and E. Catmul, "The Reyes image rendering architecture," in *Proceedings of Conference on Computer Graphics and Interactive Techniques*, pp. 95–102, 1987.

[63] T. H. Cormen, C. E. Leiserson, R. L. Riverst, and C. Stein, *Introduction to Algorithms*. MIT Press, 2nd ed., 2001.

[64] CUDPP, "CUDA data-parallel primitives library," http://code.google.com/p/cudpp/, 2010.

[65] F. Darema, "SPMD model: Past, present and future. Recent advances in parallel virtual machine and message passing interface," in *Proceedings of European PVM/MPI Users' Group Meeting*, Lecture Notes in Computer Science 2131.1, 2001.

[66] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proceedings of Symposium on Operating System Design and Implementation*, pp. 137–149, 2004.

[67] Y. Deng and S. Mu, "The potential of GPUs for VLSI physical design automation," in *Proceedings of International Conference on Solid-State and Integrated-Circuit Technology*, pp. 2272–2275, 2008.

[68] Y. Deng, B. Wang, and S. Mu, "Taming irregular EDA applications on GPUs," in *Proceedings of International Conference on Computer Aided Design*, pp. 539–546, 2009.

[69] J. Dongarra, V. Eijkhout, and P. Luszczek, "Recursive approach in sparse matrix LU factorization," *Scientific Programming*, vol. 9, no. 1, pp. 51–60, 1998.

[70] Y. Dotsenko, N. K. Govindaraju, P.-K. Sloan, C. Boyd, and J. Manferdelli, "Fast scan algorithms on graphics processors," in *Proceedings of International Conference on Supercomputing*, pp. 205–213, 2008.

[71] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proceedings of Design Automation Conference*, pp. 269–274, 1998.

[72] K. Fatahalian and M. Houston, "GPUs: A closer look," *ACM Queue*, vol. 6, no. 2, pp. 18–28, 2008.

[73] Z. Feng and P. Li, "Multigrid on GPU: Tackling power grid analysis on parallel SIMT platforms," in *Proceedings of International Conference on Computer Aided Design*, pp. 647–654, 2008.

[74] Z. Feng and P. Li, "Fast thermal analysis on GPU for 3D-ICs with integrated microchannel cooling," in *Proceedings of International Conference on Computer Aided Design*, pp. 551–555, 2010.

[75] Z. Feng and Z. Zeng, "Parallel multigrid preconditioning on graphics processing units (GPUs) for robust power grid analysis," in *Proceedings of Design Automation Conference*, pp. 661–666, 2010.

[76] J. A. Fisher, "Very long instruction word architecture and the ELI-512," in *Proceedings of International Symposium on Computer Architecture*, pp. 140–150, 1983.

[77] T. Folley, "Parallel programming on Larrabee," Special Tutorial on SIGGRAPH 2008, 2008.

[78] R. M. Fujimoto, *Parallel and Distributed Simulation Systems*. Wiley-Interscience, 2000. ISBN: 0471183830.

[79] W. L. Fung, I. Sham, G. Yuan, and T. M. Aamodt, “Dynamic warp formation and scheduling for efficient GPU control flow,” in *Proceedings of International Symposium on Microarchitecture*, pp. 407–420, 2007.
[80] V. Galiano, N. H. Migalló, V. MigallóN, and J. PenadéS, “GPU-based parallel algorithms for sparse nonlinear systems,” *Journal of Parallel and Distributed Computing*, vol. 72, no. 9, pp. 1098–1105, 2012.
[81] N. Galoppo et al., “LU-GPU: Efficient algorithms for solving dense linear systems on graphic hardware,” in *Proceedings of ACM/IEEE Conference on Supercomputing*, pp. 3–14, 2005.
[82] M. Garland, “Sparse matrix computations on many-core GPUs,” in *Proceedings of Design Automation Conference*, pp. 2–6, 2008.
[83] A. Ghuloum et al., “Ct: A flexible parallel programming model for tera-scale architectures,” Intel White Paper. http://download.intel.com/pressroom/kits/research/Flexible_Parallel_Programming_Ct.pdf, 2007.
[84] J. R. Gilbert and T. Peierls, “Sparse partial pivoting in time proportional to arithmetic operations,” *SIAM Journal of Scientific Statistical Computing*, vol. 9, no. 5, pp. 862–874, 1988.
[85] Graph500, “The graph 500 list,” http://www.graph500.org/, 2010.
[86] L. Greengard and V. Rokhlin, “A fast algorithm for particle simulations,” *Journal of Computing Physics*, vol. 73, no. 2, pp. 325–348, 1987.
[87] D. Grewe and A. Lokhmotov, “Automatically generating and tuning GPU code for sparse matrix-vector multiplication from a high-level representation,” in *Proceedings of Workshop on General Purpose Processing on Graphics Processing*, 2011.
[88] R. Grimes, D. Kincaid, and D. Young, “ITPACK 2.0 user’s guide,” Technical Report CNA-150. Center for Numerical Analysis, University of Texas, 1979.
[89] Z. Gu, J. Wang, R. P. Dick, and H. Zhou, “Incremental exploration of the combined physical and behavioral design space,” in *Proceedings of Design Automation Conference*, pp. 208–213, 2005.
[90] K. Gulati, J. F. Croix, S. P. Khatri, and R. Shastry, “Fast circuit simulation on graphics processing units,” in *Proceedings of Conference on Asia and South Pacific Design Automation*, pp. 403–408, 2009.
[91] K. Gulati and S. P. Khatri, “Towards acceleration of fault simulation using graphics processing units,” in *Proceedings of Design Automation Conference*, pp. 822–827, 2008.
[92] N. Gumerov and R. Duraiswami, “Fast multipole methods on graphics processors,” *Journal of Computing Physics*, vol. 227, no. 18, pp. 8290–8313, 2008.
[93] J. A. Gunnels, F. G. Gustavson, G. M. Henry, and R. A. van de Geijn, “FLAME: Formal linear algebra methods environment,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 27, no. 4, pp. 422–455, 2001.
[94] P. Guo et al., “A model-driven partitioning and auto-tuning integrated framework for sparse matrix-vector multiplication on GPUs,” in *Proceedings of TeraGrid*, 2011.
[95] T. Hamada et al., “42 TFLOPS hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence,” in *Proceedings of Conference on High Performance Computing Networking, Storage and Analysis*, 2009.

[96] T. D. Han and T. S. Abdelrahman, "hiCUDA: High-level GPGPU programming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 1, pp. 78–90, 2011.

[97] Y. Han, D. M. Ancajas, K. Chakraborty, and S. Roy, "Exploring high throughput computing paradigm for global routing," in *Proceedings of International Conference on Computer Aided Design*, pp. 298–305, 2011.

[98] Y. Han, K. Chakraborty, S. Roy, and V. Kuntamukkala, "Design and implementation of a throughput-optimized GPU floor planning algorithm," *ACM Transactions on Design Automation of Electronic Systems*, vol. 16, no. 3, 2011.

[99] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the GPU using CUDA," in *Proceedings of High Performance Computing (HiPC)*, pp. 197–208, 2007.

[100] P. Harish, V. Vineet, and P. J. Narayanan, "Large graph algorithms for massively multithreaded architectures," IIIT Technical Report IIIT/TR/2009/74. http://web.iiit.ac.in/∼vibhavvinet/Publications/GraphAlgos_TechRep.pdf, 2009.

[101] M. Harris, "Parallel prefix sum (scan) with CUDA," http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/scan/doc/scan.pdf, 2007.

[102] M. Harris, "Optimizing parallel reduction in CUDA," http://developer.download.nvidia.com/compute/DevZone/C/html/C/src/reduction/doc/reduction.pdf, 2008.

[103] Z. He and B. Hong, "Dynamically tuned push-relabel algorithm for the maximum flow problem on CPU-GPU hybrid platforms," in *Proceedings of International Parallel and Distributed Processing Symposium*, pp. 1–10, 2010.

[104] B. He et al., "Mars: A MapReduce framework on graphics processors," in *Proceedings of Parallel Architectures and Compilation Techniques*, pp. 260–269, 2008.

[105] M. T. Heath, E. Ng, and B. W. Peyton, "Parallel algorithms for sparse linear systems," *SIAM Review*, vol. 33, pp. 420–460, 1991.

[106] R. Helfenstein and J. Koko, "Parallel preconditioned conjugate gradient algorithm on GPU," *Journal of Computational and Applied Mathematics*, vol. 236, no. 15, pp. 3584–3590, 2012.

[107] M. Herlihy and N. Shavit, *The Art of Multiprocessor Programming*. Burlington: Morgan Kaufmann Publishers, 2008.

[108] S. Hong, T. Oguntebi, and K. Olukotun, "Efficient parallel graph exploration on multi-Core CPU and GPU," in *Proceedings of Parallel Architectures and Compilation Techniques*, pp. 78–88, 2011.

[109] S. Hong et al., "Accelerating CUDA graph algorithms at maximum warp," in *Proceedings of Symposium on Principles and Practice of Parallel Programming*, pp. 267–276, 2011.

[110] C.-J. Hsu, J. L. Pino, and S. S. Bhattacharyya, "Multithreaded simulation for synchronous dataflow graphs," in *Proceedings of Design Automation Conference*, pp. 331–336, 2008.

[111] J. Huang, "Keynote speech," in *Mini GPU Technology Conference*, Beijing, 2010.

[112] C. Huijs, "A graph rewriting approach for transformational design of digital systems," in *Proceedings of EUROMICRO Conference*, pp. 177–184, 1996.

[113] J. R. Humphrey et al., "CULA: Hybrid GPU accelerated linear algebra routines," in *Proceedings of SPIE Modeling and Simulation for Defense Systems and Applications*, pp. V 770502–770502-7, 2010.

[114] M. Hussein, A. Varshney, and L. Davis, "On implementing graph cuts on cuda," in *Proceedings of the First Workshop on General Purpose Processing on Graphics Processing Units*, 2007.

[115] IEEE, "The open group base specifications Issue 6," IEEE Standard 1003.1, 2004 Edition. (http://pubs.opengroup.org/onlinepubs/007904975/basedefs/pthread.h.html), 2004.

[116] IEEE, "SystemC language — 2011," http://standards.ieee.org/getieee/1666/download/1666-2011.pdf, 2011.

[117] Intel, "Intel® array building blocks," http://software.intel.com/en-us/articles/intel-array-building-blocks, 2010.

[118] Intel, "The Intel® Xeon Phi™ coprocessor 5110P highly-parallel processing for unparalleled discovery," http://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/high-performance-xeon-phi-coprocessor-brief-2.pdf, 2012.

[119] T. B. Jablin et al., "Automatic CPU-GPU communication management and optimization," in *Proceedings of Conference on Programming Language Design and Implementation*, pp. 142–151, 2011.

[120] A. Jantsch, *Modeling Embedded Systems and SoCs.* Morgan Kaufmann, 2004.

[121] JEDEC, "JEDEC standard: GDDR5 SGRAM," http://www.jedec.org/standards-documents/docs/jesd212, 2009.

[122] JEDEC, "JEDEC standard: DDR4 SDRAM," http://www.jedec.org/standards-documents/docs/jesd79-4, 2012.

[123] H. Jooybar, W. W. L. Fung, M. O'Connor, J. Devietti, and T. M. Aamodt, "GPUDet: A deterministic GPU architecture," in *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1–12, 2013.

[124] T. Jost, S. Contassot-Vivier, and S. Vialle, "An efficient multi-algorithms sparse linear solver for GPUs," in *Proceedings of International Conference on Parallel Computing (ParCo)*, 2009.

[125] J. Jung and D. P. O'Leary, "Cholesky decomposition and linear programming on a GPU," in *Proceedings of Workshop on Edge Computing Using New Commodity Architectures*, 2006.

[126] R. Kapoor, M. Adan, and L. Schaffer, "Achieving optimal performance scalability for physical verification," http://www.synopsys.com/Tools/Implementation/PhysicalVerification/CapsuleModule/hercules_achiv_wp.pdf, 2004.

[127] K. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Applications in VLSI domain," *IEEE Transactions on VLSI Systems*, vol. 7, no. 1, pp. 69–79, 1997.

[128] G. J. Katz and J. J. T. Kider, "All-pairs shortest-paths for large graphs on the GPU," in *Proceedings of Symposium on Graphics Hardware*, pp. 47–55, 2008.

[129] J. Keller, C. Keßler, and J. Träff, *Practical PRAM Programming.* John Wiley and Sons, 2001. ISBN 0-471-35351-5.

[130] A. Kerr, D. Campbell, and M. Richards, "QR decomposition on GPUs," in *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 71–78, 2009.

[131] K. Keutzer, "DAGON: Technology binding and local optimization by DAG matching," in *Proceedings of Design Automation Conference*, pp. 341–347, 1987.

[132] A. Khabou, J. W. Demmel, L. Grigori, and M. Gu, "LU factorization with panel rank revealing pivoting and its communication avoiding version," UCB Technical Report No. UCB/EECS-2012-15, 2012.

[133] A. Khajeh-Saeed, S. Poole, and J. B. Perot, "Acceleration of the Smith–Waterman algorithm using single and multiple graphics processors," *Journal of Computational Physics*, vol. 229, no. 11, pp. 4247–4258, 2010.

[134] Khronos Group, "OpenCL — the open standard for parallel programming of heterogeneous systems," http://www.khronos.org/opencl/, 2012.

[135] K. Kim, V. Eijkhout, and R. A. Geijn, "Dense matrix computation on a heterogeneous architecture: A block synchronous approach," TACC Technique Report TR-12-04, 2012.

[136] G. Kleinhans, F. J. Sigl, and K. Antreich, "Gordian: VLSI placement by quadratic programming and slicing optimization," *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, vol. 10, no. 3, pp. 356–365, 1991.

[137] M. A. Kochte, M. Schaal, H.-J. Wunderlich, and C. G. Zoellin, "Efficient fault simulation on many-core processors," in *Proceedings of Design Automation Conference*, pp. 380–385, 2010.

[138] J. Kurzak, S. Tomov, and J. Dongarra, "Scheduling dense linear algebra operations on multicore processors," *Concurrency and Computation: Practice & Experience*, vol. 22, no. 1, pp. 15–44, 2010.

[139] J. Kurzak, S. Tomov, and J. Dongarra, "Autotuning GEMM kernels for the Fermi GPU," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2045–2057, 2012.

[140] W. B. Langdon, "A fast high quality pseudo random number generator for graphics processing units," in *Proceedings of Congress on Evolutionary Computation*, pp. 459–465, 2008.

[141] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proceedings of International Symposium on Code Generation and Optimization*, pp. 75–86, 2004.

[142] E. Lee, "The problem with threads," *IEEE Computer*, vol. 39, no. 5, pp. 33–42, 2006.

[143] A. Levinthal and P. Porter, "Chap — a SIMD graphics processor," in *Proceedings of Conference on Computer Graphics and Interactive Techniques*, pp. 77–82, 1984.

[144] D. Lewis, "A hierarchical compiled code event-driven logic simulator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 6, pp. 726–737, 1991.

[145] C. Li, C. Ding, and K. Shen, "Quantifying the cost of context switch," in *Proceedings of Workshop on Experimental Computer Science*, 2007.

[146] J. Li, X. Li, G. Tan, M. Chen, and N. Sun, "An optimized large-scale hybrid DGEMM design for CPUs and ATI GPUs," in *Proceedings of International Conference on Supercomputing*, pp. 377–386, 2012.

[147] M. Li and M. S. Hsiao, "3-D parallel fault simulation with GPGPU," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 10, pp. 1545–1555, 2011.

[148] Y.-L. S. Lin, *Essential Issues in SOC Design: Designing Complex Systems-on-Chip*. Springer. Softcover reprint of hardcover 1st ed., 2006 ed., 2010.

[149] J. Liu, "Computational models and task scheduling for parallel sparse Cholesky factorization," *Parallel Computing*, vol. 3, no. 4, pp. 327–342, 1986.

[150] J. Liu, "The multifrontal method for sparse matrix solution: Theory and practice," *SIAM Review*, vol. 34, no. 1, pp. 82–109, 1992.

[151] X. Liu, Z. Liu, S. X.-D. Tan, and J. A. Gordon, "Full-chip thermal analysis of 3D ICs with liquid cooling by GPU-accelerated GMRES method," in *Proceedings of IEEE/ACM International Symposium on Low-Power Electronics and Design*, pp. 123–128, 2012.

[152] Y. Liu and J. Hu, "A new algorithm for simultaneous gate sizing and threshold voltage assignment," in *Proceedings of the International Symposium on Physical Design (ISPD)*, 2009.

[153] Y. Liu and J. Hu, "GPU-based parallelization for fast circuit optimization," *ACM Transactions on Design Automation of Electronic Systems*, vol. 16, no. 3, 2011.

[154] Y. Liu, B. Schmidt, and D. L. Maskell, "CUDASW++2.0: Enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions," *BMC Research Notes*, vol. 3, no. 93, 2010.

[155] W. Liu et al., *BSIM3v3.2.2 MOSFET Model*. Users' Manual, 1999.

[156] Y. Low et al., "GraphLab: A new parallel framework for machine learning," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 2010.

[157] Y. Lu, H. Zhou, L. Shang, and X. Zeng, "Multicore parallel min-cost flow algorithm for CAD applications," in *Proceedings of Design Automation Conference*, pp. 832–837, 2009.

[158] L. Luo, M. Wong, and W.-M. Hwu, "An effective GPU implementation of breadth-first search," in *Proceedings of Design Automation Conference*, pp. 52–55, 2010.

[159] G. Martin, B. Bailey, and A. Piziali, *ESL Design and Verification: A Prescription for Electronic System Level Methodology*. Morgan Kaufmann, 1st ed., 2007.

[160] K. K. Matam and K. Kothapalli, "Accelerating sparse matrix vector multiplication in iterative methods using GPU," in *Proceedings of International Conference on Parallel Processing*, 2011.

[161] K. Matsumoto, N. Nakasato, and S. G. Sedukhin, "Blocked all-pairs shortest paths algorithm for hybrid CPU-GPU system," in *Proceedings of High Performance Computing and Communications (HPCC)*, pp. 145–152, 2011.

[162] T. G. Mattson, B. A. Sanders, and B. L. Massingill, *Patterns for Parallel Programming.* Addison-Wesley Professional, 2004.

[163] M. D. McCool, “Scalable programming models for massively multicore processors,” *Proceedings of IEEE*, vol. 96, no. 5, pp. 816–831, 2008.

[164] S. McGlaun, “GPU market shows impressive growth in Q2 2009,” Slippery-Brick. http://www.slipperybrick.com/2009/07/gpu-market-shows-impressive-growth-in-q2-2009/, 2009.

[165] N. Melab, I. Chakroun, M. Mezmaz, and D. Tuyttens, “A GPU-accelerated branch-and-bound algorithm for the flow-shop scheduling problem,” in *Proceedings of International Conference on Cluster Computing*, 2012.

[166] J. Meng, D. Tarjan, and K. Skadron, “Dynamic warp subdivision for integrated branch and memory divergence tolerance,” in *Proceedings of International Symposium on Computer Architecture*, pp. 235–246, 2011.

[167] D. Merrill, M. Garland, and A. Grimshaw, “Scalable GPU graph traversal,” in *Proceedings of the Symposium on Principles and Practice of Parallel Programming*, pp. 117–128, 2012.

[168] U. Meyer and P. Sanders, “Delta-stepping: A parallel single source shortest path algorithm,” in *Proceedings of Annual European Symposium on Algorithms*, pp. 393–404, 1998.

[169] Microsoft, “DirectX 11,” http://windows.microsoft.com/zh-CN/windows7/products/features/directx-11, 2000.

[170] D. Muller, “Optimizing yield in global routing,” in *Proceedings of International Conference on Computer Aided Design*, pp. 480–486, 2006.

[171] K. Nabors and J. White, “FastCap: A multipole accelerated 3-D capacitance extraction program,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 11, pp. 1447–1459, 1991.

[172] L. Nagel, “SPICE: A computer program to simulate computer circuits,” University of California, Berkeley UCB/ERL Memo M520, 1995.

[173] G.-J. Nam, J. C. Alpert, and P. G. Villarrubia, *ISPD 2005/2006 Placement Benchmarks. Modern Circuit Placement.* US: Springer, 2007.

[174] M. Nanjundappa, H. D. Patel, B. A. Jose, and S. K. Shukla, “SCGPSim: A fast SystemC simulator on GPUs,” in *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 149–154, 2010.

[175] V. Narasiman, M. Shebanow, C. J. Lee, R. Miftakhutdinov, O. Mutlu, and Y. N. Patt, “Improving GPU performance via large warps and two-level warp scheduling,” in *Proceedings of International Symposium on Microarchitecture*, pp. 308–317, 2011.

[176] R. Nath, S. Tomov, and J. Dongarra, “An improved magma GEMM for Fermi GPUs,” Technical Report 227. LAPACK Working Note, 2010.

[177] M. Naumov, L. S. Chien, P. Vandermersch, and U. Kapasi, “CUSPARSE Library,” in *GPU Technology Conference*, 2010.

[178] J. Nickolls, I. Buck, M. Garland, and K. Skadron, “Scalable parallel programming with CUDA,” *ACM Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[179] J. Nie, “Evaluating the potential of Intel Ct technology for EDA Computing,” Bachelor Thesis. Tsinghua University, 2009.

[180] NVIDIA, “NVIDIA GeForce 8800 GPU architecture overview,” Technical Brief, 2006.

[181] NVIDIA, "CUBLAS," https://developer.nvidia.com/cublas, 2007.

[182] NVIDIA, "CUFFT," https://developer.nvidia.com/cufft, 2007.

[183] NVIDIA, "NVIDIA CUDA Compute Unified Device Architecture programming guide," Version 1.0, 2007.

[184] NVIDIA, "PTX: Parallel Thread Execution ISA," Version 1.0, 2007.

[185] NVIDIA, "NVIDIA's next generation CUDATM compute architecture: Fermi$^{\text{TM}}$," http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf, 2009.

[186] NVIDIA, "CUDA GPUs," https://developer.nvidia.com/cuda-gpus, 2012.

[187] NVIDIA, "NVIDIA's next generation CUDATM compute architecture: Kepler$^{\text{TM}}$ GK110," (http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf), 2012.

[188] T. Okuyama, F. Ino, and K. Hagihara, "A task parallel algorithm for computing the costs of all-pairs shortest paths on the CUDA-compatible GPU," in *Proceedings of International Symposium on Parallel and Distributed Processing with Applications*, pp. 284–291, 2008.

[189] E. Onbaşoğlu and L. Özdamar, "Parallel simulated annealing algorithms in global optimization," *Journal of Global Optimization*, vol. 19, no. 1, pp. 27–50, 2001.

[190] OpenACC, "The OpenACC$^{\text{TM}}$application programming interface," http://www.openacc.org/sites/default/files/OpenACC.1.0_0.pdf, 2011.

[191] OpenGL, "The OpenGL graphics system: a specification," http://www.opengl.org/registry/doc/glspec43.core.20120806.pdf, 2012.

[192] OSCI, "SystemC 2.2," http://www.accellera.org/members/download_files/check_file?agreement=systemc-2_2-draft, 2007.

[193] V. Osipov, P. Sanders, and J. Singler, "The Filter-Kruskal minimum spanning tree algorithm," in *Proceedings of Workshop on Algorithm Engineering and Experiments*, 2009.

[194] P. Otellini, "Keynote speech," in *Intel Developer Forum*, 2011.

[195] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[196] J. D. Owens, D. Luebke, N. Govindaraju, M. Houston, J. Krüger, A. E. Lefohn, and T. A. Purcell, "A survey of general-purpose computation on graphics hardware," in *Eugographics: State of the Art Reports*, pp. 21–51, 2005.

[197] PCI-SIG, *PCI Express Base Specification.* Revision 1.0, 2002.

[198] J. C. Pichel, F. F. Rivera, M. Fernandez, and A. Rodríguez, "Optimization of sparse matrix–vector multiplication using reordering techniques on GPUs," *Microprocessors and Microsystems*, vol. 36, no. 2, pp. 65–77, 2012.

[199] J. C. Pichel, D. E. Singh, and J. Carretero, "Reordering algorithms for increasing locality on multicore processors," in *Proceedings of the IEEE International Conference on High Performance Computing and Communications*, pp. 123–130, 2008.

[200] F. Pinel, B. Dorronsoro, and P. Bouvrya, "Solving very large instances of the scheduling of independent tasks problem on the GPU," *Journal of Parallel Distributed Computing*, vol. 73, no. 1, pp. 101–110, 2013.

[201] K. Pingali et al., "The tao of parallelism in algorithms," in *Proceedings of Programming Language Design and Implementation*, pp. 12–25, 2011.

[202] C. Pinto et al., "GPGPU-accelerated parallel and fast simulation of thousand-core platforms," in *Proceedings of International Symposium on Cluster, Cloud and Grid Computing*, 2011.

[203] QEMU, "Full system processor emulator," http ://wiki.qemu.org.

[204] H. Qian and Y. Deng, "Accelerating RTL simulation with GPUs," in *Proceedings of International Conference on Computer Aided Design*, pp. 687–693, 2011.

[205] H. Qian, Y. Deng, B. Wang, and S. Mu, "Towards accelerating irregular EDA applications with GPUs," *Integration: The VLSI Journal*, vol. 45, no. 1, pp. 46–60, 2012.

[206] S. Raghav, M. Ruggiero, and D. Atienza, "Scalable instruction set simulator for thousand-core architectures running on GPGPUs," in *Proceedings of International Conference on High Performance Computing and Simulation*, 2010.

[207] S. Raghav et al., "Full system simulation of many-core heterogeneous SoCs using GPU and QEMU semihosting," in *Proceedings of GPGPU Workshop*, 2012.

[208] A. Ramalingam et al., "An accurate sparse matrix based framework for statistical static timing analysis," in *Proceedings of International Conference on Computer Aided Design*, 2006.

[209] C. Ranger et al., "Evaluating MapReduce for multi-core and multiprocessor systems," in *Proceedings of International Symposium on High-Performance Computer Architecture*, pp. 13–24, 2007.

[210] P. Rashinkar, P. Paterson, and L. Singh, *System-on-a-Chip Verification: Methodology and Techniques*. Springer, 1st ed., 2000.

[211] L. Ren et al., "Sparse LU factorization for parallel circuit simulation on GPU," in *Proceedings of Design Automation Conference*, 2012.

[212] M. Rhu and M. Erez, "CAPRI: Prediction of compaction-adequacy for handling control-divergence in GPGPU architectures," in *Proceedings of International Symposium on Computer Architecture*, pp. 61–71, 2012.

[213] Rocketick, "Harnessing the power of GPU," http://www.rocketick.com/technology/harnessing-the-power-of-gpu, 2012.

[214] S. Rostrup, S. Srivastava, and K. Singhal, "Fast and memory-efficient minimum spanning tree on the GPU," *International Journal of Computational Science and Engineering*, vol. 8, no. 1, pp. 21–33, 2013.

[215] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[216] R. Rutenbar, "Next-generation design and EDA challenges: Small physics, big systems, and tall tool-Chains," in *Proceedings of Asia and South Pacific Design Automation Conference*, 2007.

[217] S. Ryoo et al., "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proceedings of Principles and Practice of Parallel Programming*, pp. 73–82, 2008.

[218] Y. Saad, "Iterative methods for sparse linear systems," SIAM, 2000.

[219] S. Sapatnekar, *Timing*. Springer, 1st ed., 2004. Ch. 5.

[220] S. Sapatnekar et al., "Reinventing EDA with many-core processors," in *Proceedings of Design Automation Conference*, 2008.

[221] I. Savran and J. D. Bakos, "GPU acceleration of near-minimal logic minimization," in *Proceedings of Symposium on Application Accelerators in High-Performance Computing*, 2010.

[222] L. Seiler et al., "Larrabee: A many-Core X86 architecture for visual computing," *ACM Transaction on Graphics*, vol. 3, no. 8, pp. 18:1–18.16, 2008.

[223] A. Sen, B. Aksanli, and M. Bozkurt, "Speeding up cycle based logic simulation using graphics processing units," *International Journal of Parallel Programming*, vol. 39, no. 5, pp. 639–661, 2010.

[224] S. Sengupta, M. Harris, Y. Zhang, and J. D. Owens, "Scan primitives for GPU computing," in *Proceedings of Graphics Hardware*, pp. 97–106, 2007.

[225] N. Seoane and A. J. Garcia-Loreiro, "Study of parallel numerical methods for semiconductor device simulation," *International Journal of Numerical Modeling: Electronic Networks, Devices, and Fields*, vol. 19, no. 1, pp. 15–32, 2005.

[226] Y. Shapira, *Matrix-Based Multigrid: Theory and Applications*. Springer, 2009.

[227] N. A. Sherwani, *Algorithm for VLSI Physical Design Automation*. Springer, 3rd ed., 1998.

[228] J. Shi et al., "GPU friendly fast Poisson solver for structured power grid network analysis," in *Proceedings of Design Automation Conference*, pp. 178–183, 2009.

[229] A. L. Shimpi and D. Wilson, "Intel's Larrabee architecture disclosure: A calculated first move," (http://www.anandtech.com/showdoc.aspx?i=3367), 2008.

[230] Si2, "OpenEDA," http://www.si2.org/openeda.si2.org/oso_news.php, 2004.

[231] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, *Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Professional, 2002.

[232] T. Smalley, "RV770: ATI Radeon HD 4850 & 4870 analysis. bit-tech.net," (http://www.bit-tech.net/hardware/graphics/2008/09/02/ati-radeon-4850-4870-architecture-review/1), 2008.

[233] SMIC, "SMIC-Cadence reference flow 2.1," http://www.smics.com/eng/design/reference_flows05.php, 2013.

[234] F. Song, S. Tomov, and J. Dongarra, "Enabling and scaling matrix computations on heterogeneous multi-core and multi-GPU systems," in *Proceedings of ACM International Conference on Supercomputing*, pp. 365–376, 2012.

[235] R. Stephens, "A survey of stream processing," *Acta Informatica*, vol. 34, no. 7, pp. 491–554, 1995.

[236] B. Stroustrup, *The C++ Programming Language*. ISBN 0-201-88954-4, 3rd ed., 1997.

[237] R. Strzodka, "Accelerated ANSYS fluent: Algebraic multigrid on a GPU," Available on http://developer.download.nvidia.com/GTC/PDF/GTC2012/PresentationPDF/RobertStrzodka_Accelerated_ANSYS_Fluent_SC12.pdf, 2012.

[238] L. Subramany, "GPU based lithography simulation and OPC," Master Thesis. Department of Electrical and Computer Engineering. University of Massachusetts Amherst, 2011.

[239] B. Suri, U. D. Bordolo, and P. Eles, "A scalable GPU-based approach to accelerate the multiple-choice knapsack problem," in *Proceedings of Design, Automation, and Test in Europe (DATE) Conference*, pp. 1126–1129, 2012.

[240] Synopsys, “The gold standard for accurate circuit simulation,” http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx, 2008.

[241] Synopsys, “Synopsys unveils new IC compiler router delivering 10X speed-up,” http://news.synopsys.com/index.php?s=43&item=575, 2008.

[242] G. Tan, L. Li, S. Trieche, E. Phillips, Y. Bao, and N. Sun, “Fast implementation of DGEMM on Fermi GPU,” in *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.

[243] R. O. Topaloglu, “Fast variational static IR-drop analysis on the graphical processing unit,” in *Proceedings of International Symposium on Quality Electronic Design*, pp. 1–6, 2011.

[244] T. Van Luong, N. Melab, and E. G. Talbi, “Large neighborhood local search optimization on graphics processing units,” in *Proceedings of International Parallel and Distributed Processing Symposium*, pp. 1–8, 2010.

[245] A. Vincenzi, A. Sridhar, M. Ruggiero, and D. Atienza, “Fast thermal simulation of 2D/3D integrated circuits exploiting neural networks and GPUs,” in *Proceedings of International Symposium on Low-Power Electronics and Design*, pp. 151–156, 2011.

[246] S. Vinco, D. Chatterjee, V. Bertacco, and F. Fummi, “SAGA: SystemC acceleration on GPU architectures,” in *Proceedings of Design Automation Conference*, pp. 115–120, 2012.

[247] V. Vineet, P. Harish, S. Patidar, and P. J. Narayanan, “Fast minimum spanning tree for large graphs on the GPU,” in *Proceeding of High Performance Graphics*, pp. 167–171, 2009.

[248] V. Vineet and P. Narayanan, “Cuda cuts: Fast graph cuts on the GPU,” in *Proceedings of Conference on Computer Vision and Pattern Recognition: Workshop on Visual Computer Vision on GPUs*, pp. 1–8, 2008.

[249] V. Volkov, “Better performance at lower occupancy,” in *Proceedings of GPU Technology Conference*, 2010.

[250] V. Volkov and J. Demmel, “Benchmarking GPUs to tune dense linear algebra,” in *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, 2008.

[251] B. Wang, Y. Zhu, and Y. Deng, “Distributed time, conservative parallel logic simulation on GPUs,” in *Proceedings of Design Automation Conference*, 2010.

[252] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability.* Morgan Kaufmann, 1st ed., 2006.

[253] M. S. Waterman, *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman and Hall/CRC, 1st ed., 1995.

[254] J. Williamson, Y. Lu, L. Shang, H. Zhou, and X. Zeng, “Parallel cross-layer optimization of high-level synthesis and physical design,” in *Proceedings of Design Automation Conference*, pp. 467–472, 2011.

[255] C. M. Wittenbrink, E. Kilgariff, and A. Prabhu, “Fermi GF100 GPU Architecture,” *IEEE Micro*, vol. 21, no. 2, pp. 50–59, 2011.

[256] X. Xiao, A. M. Aji, and W.-C. Feng, “On the robust mapping of dynamic programming onto a graphics processing Unit,” in *Proceedings of International Conference on Parallel and Distributed Systems*, pp. 26–33, 2009.

[257] X. Xiao and W.-C. Feng, "Inter-block GPU communication via fast barrier synchronization," in *Proceedings of IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–12, 2010.
[258] Y. Yang, P. Xiang, J. Kong, and H. Zhou, "A GPGPU compiler for memory optimization and parallelism management," in *Proceedings of Conference on Programming Language Design and Implementation*, pp. 86–97, 2010.
[259] Z. Yu, X. Guan, Y. Deng, and Y. Wang, "Riding bandwagon of parallel computing for nanoscale device simulation," in *Proceedings of International Workshop on Quantum Systems and Semiconductor Devices: Analysis, Simulations, Applications*, 2009.
[260] Z. Zeng, X. Ye, Z. Feng, and P. Li, "Tradeoff analysis and optimization of power delivery networks with on-chip voltage regulation," in *Proceedings of Design Automation Conference*, pp. 831–836, 2010.
[261] J. Zhang et al., "GPU-accelerated inverse lithography technique," in *Proceedings of SPIE*, vol. 7379, 2009.
[262] X. Zhao and Z. Feng, "Fast multipole method on GPU: Tackling 3-D capacitance extraction on massively parallel SIMD platforms," in *Proceedings of Design Automation Conference*, pp. 558–563, 2011.
[263] Y. Zhu, B. Wang, and Y. Deng, "Massively parallel logic simulation with GPUs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 16, no. 3, 2011.