# PERFORMANCE ANALYSIS FOR HIGH-PRECISION INTERCONNECT SIMULATION

R. Heinzl[△], M. Spevak[°], P. Schwaha[△], T. Grasser[△] and S. Selberherr[°]

[△]Christian Doppler Laboratory for TCAD in Microelectronics
at the Institute for Microelectronics

[°]Institute for Microelectronics, Technical University Vienna,
Gußhausstraße 27-29/E360, A-1040 Vienna, Austria
E-mail: {heinzl|schwaha|spevak|grasser|selberherr}@iue.tuwien.ac.at

**KEYWORDS**
Mesh generation, error estimation, mesh quality, high performance computing, programming paradigms

## ABSTRACT

This work analyzes the performance of high-precision interconnect simulation tools on refined meshes with guaranted accuracy. On the one hand, the integrated circuits are subject to an ongoing miniaturization which results in ever increasing computing power. On the other hand, the simulation of these integrated circuits demands more sophisticated simulation methodologies such as better resolution of geometrical features or more complex surface topography. We show that modern microprocessor architectures and memory hierarchies impose performance limits on the simulation time.

## INTRODUCTION

Down-scaling of integrated circuits to the deep sub-micron regime and beyond increases the influence of interconnects on circuit behavior drastically. Parasitic effects are becoming more and more important as devices get faster and line widths smaller. These effects become the limiting factor for further improvements of circuit speed. An essential step in technology computer aided design (TCAD) is the optimization of these parameters, which demands vast amounts of computer resources, CPU-time and memory. Therefore the performance of current computer systems is essential for an optimal simulation flow. The overall performance of computer systems with a given set of applications depends on numerous factors and can not be attributed to only the speed of the central processing unit (CPU). Among the most important factors is the connection of the CPU to the computers main memory [6]. In the early days of computers the employed memories were faster or at least comparable in speed to the CPU. Naturally the focus of the evolution of CPUs was to increase their processing speed. This goal was greatly aided and in fact only made possible by continuous downscaling of the dimensions of the devices which they components are built on. This downscaling of the densely packed logic found in CPUs made it possible to attain ever higher clock speeds thereby increasing their maximum performance. The main effect on random access memory (RAM) modules, on the other hand, was to increase their sizes, again by an ever growing level of integration. While speed was also an important concern, it quickly lagged behind, as the signal to noise ratios in the highly integrated structures worsen, which leads to increased latencies due to the necessity of appropriate signal handling to insure proper operation.

The reduction of feature size and therefore the increase of operating frequencies is not going to continue without bounds and different strategies have to be used to increase the performance of the processing cores. Nevertheless this trend of increasing clock speeds, especially of CPUs, has already led to the problem that CPUs require data at a faster rate than memories are able to supply. This has resulted in the development of memory hierarchies introducing several levels of caches and instruction pipelines, thereby increasing the overall performance of the systems. The additional complexity induced by these measures makes it more and more important to employ appropriate compilers and techniques to obtain optimal performance [5, 12]. This is even more so, as the increase of computing power of future computer hardware is primarily obtained by multiplying the processor cores.

Not only the hardware of computers and the compilers have evolved and now provide a myriad of features, but new methodologies of software development and programming paradigms have also surfaced. Different approaches have focused on the development of high performance libraries for the area of scientific computing such as PETSc [3], CCA [14], or MTL [17].

## MODERN PROGRAMMING PARADIGMS

From a software point of view, numerous new paradigms have evolved recently, which allow the synthesis of highly efficient code on modern hardware. It is now the aim to combine the newly provided possibilities in such a way, that an optimal result not only in terms of run-time efficiency, but also maintainability, extendability, portability, and orthogonality of code is attained. While run-time efficiency, maintainability and extendability of code have classically been contradicting goals, with code tuned for high performance often becoming an unreadable maintenance nightmare, the advent of new compilers deploying new optimizers and feature sets supports the design of high performance code which no longer needs to be unreadable [1, 9, 16]. Especially generic

programming accomplishes both, a general solution for most of the application scenarios and highly specialized code parts for minor, but also important, scenarios without sacrificing performance [2, 11]. This has already been demonstrated in the field of numerics and yields figures comparable to Fortran [13, 18], the previously undisputed candidate for this kind of calculations.
Based on these techniques we developed a high performance simulation engine based on the SAP tools [15]. With template meta-programming [1], the functional specification can be used very similarly to the original mathematical formulation, as can be seen in this work. Due to this new programming technique and the corresponding evaluation at compile time the calculation associated with the specified equations is highly optimized by the compiler and thereby ensures excellent run-time performance often superior to highly hand-optimized code. In our case C++ was the language of choice, because currently no other language offers sufficient support for all the necessary programming techniques to enable the required level of abstraction.
Our own investigations in the field of compiler optimization and compiler comparison has shown significant differences in optimization behavior and run-time performance of modern programming techniques [8].

## INTERCONNECT MODEL

Our interconnect simulation tools use the finite element method to discretize the partial differential equations resulting in a system of equations that eventually has to be linearized, and thereafter solved with a preconditioned conjugate gradient algorithm [10]. To give a glimpse on details we consider a typical problem of forming the equation system.
The problem we consider is posed in the following way:

$$\mathcal{L}\Psi := \operatorname{div}(-\varepsilon \operatorname{grad}(\Psi)) - \varrho = 0 \qquad \text{in } \Omega \tag{1}$$

$$\Psi - \Psi_D = 0 \qquad \text{on } \partial\Omega\,, \tag{2}$$

where $\varepsilon$ denotes the (isotropic) permittivity of the considered domain, which is assumed to be constant in an element of the tessellation. Due to the weak formulation using Galerkin finite elements [19] weighting coefficients for the local element matrices have to be derived for tetrahedra:

$$g_1^e = \varepsilon \frac{K_{11}^2 + K_{21}^2 + K_{31}^2}{\det \mathbf{J}} \tag{3}$$

there, $\mathbf{K}$ is the adjoint matrix of the Jacobian $\mathbf{J}$ which is derived by the affine transformation of the mesh elements to the standard element. Due to operator overloading different mathematical structures such as scalars, vectors, and even matrices can be handled using identical notation. Therewith the transformation into code results in the following code snippet:

```
double g1 = epsilon*(K11*K11 + K21*K21 + K31*K31)/detJ;
```

To assemble the system matrix, a local element matrix has to be assembled: $\mathbf{S}^e$ stands for the local element stiffness matrix which is derived by:

$$\mathbf{S}^e = g_1^e \mathbf{S}_1 + g_2^e \mathbf{S}_2 + g_3^e \mathbf{S}_3 + g_4^e \mathbf{S}_4 + g_5^e \mathbf{S}_5 + g_6^e \tag{4}$$

The corresponding C++ code reads:

```
Se = g1*S1 + g2*S2 + g3*S3 + g4*S4 + g5*S5 + g6*S6;
```

`S1-S6` means the linear form function matrices and `g1-g6` are calculated at the nodes of the tetrahedra in the global coordinate system [4].

## PERFORMANCE ANALYSIS

It should be noted that for high precision simulations it is essential to model the simulation domain as exactly as possible. The accuracy and efficiency of a finite element and finite volume simulation strongly depends on the quality of the tessellation of the domain. As a consequence we introduced a comprehensive solid modeling and mesh generation and adaptation approach [7].
Figure 1 presents the example structure under investigation with a coarse mesh for the following performance analysis. In order to obtain sufficiently accurate results, the mesh size typically has to be in the order of $10^4$ to some $10^5$ nodes.

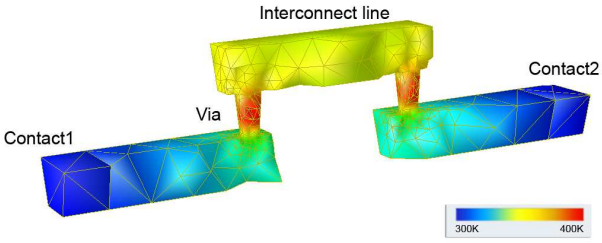


Figure 1: Temperature distribution due to self-heating in a tapered interconnect line with cylindrical vias.

For a rigorous analysis we evaluate three different implementations in C++ and compare them to a hand-optimized Fortran 77 implementation on different computer architectures. The first implementation is based on the GNU GCC `valarray` data-type which is a standardized data-structure representing a mathematical vector. This data-type has shown excellent performance on different computer architectures with recent compilers. Secondly, we utilize the Blitz++ [18] library, which introduced high performance calculation comparable to Fortran 77 directly in C++. Lastly, a naive C++ implementation is used that creates two temporary objects, one for the addition and one for the assignment. As a consequence all elements have to be accessed three times. The tests were performed on four different computer systems:

| CPU type | Clock speed | RAM | Compiler | MFLOPS |
|---|---|---|---|---|
| Pentium 4 | 2.8 GHz | 2 GB | GCC 4.0.2 | 2310.9 |
| AMD64 | 2.2 GHz | 2 GB | GCC 3.4.4 | 3543.0 |
| IBM P655 | 8x1.5 GHz | 64 GB | GCC 4.0.2 | 16361.7 |
| G5 | 4x2.5 GHz | 8 GB | GCC 4.0.0 | 24434.0 |

Figures 2-5 compare these different approaches on different hardware architectures. The y-axes is labeled with million operations per second. The vector addition consists of 3 operations, two additions and one assignment.

For vector lengths smaller than $10^4$, cache hits reveal the full computation power of the CPU, longer vectors show the limits imposed by memory bandwidth. The poor performance of naive C++ code is indeed remarkable.

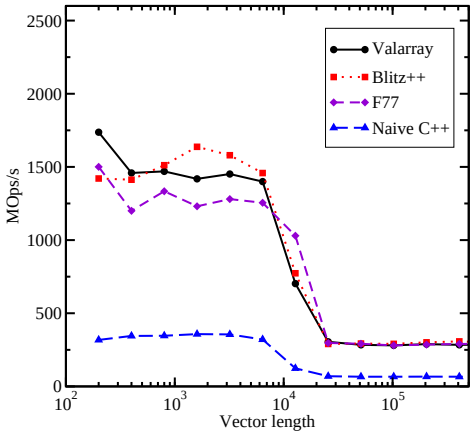


Figure 2: Comparison of different functional specification on the Pentium4.

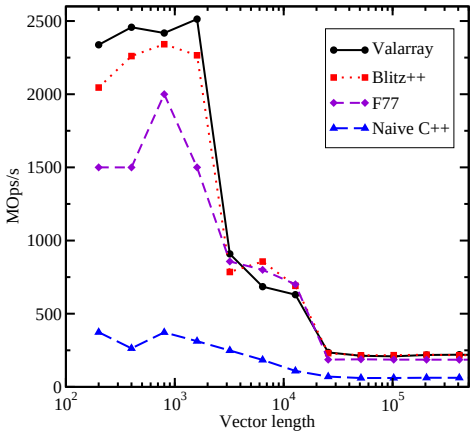


Figure 3: Comparison of different functional specification on the AMD64.

Based on these observations of restrictions due to the limited bandwidth, we illustrate the influence of a problem's size on the overall finite element. We therefore investigate our test structure with different levels of refinement. By resolving a three-dimensional simulation domain, the number of points easily exceeds the critical threshold and thereby leads to severe problems caused by memory bandwidth restrictions (Figure 6).

Investigations of parallelization attempts on multiprocessor machines (G5) show that the inner loop of the finite element assembly cannot be parallelized easily. On the one side, the update mechanisms of the element

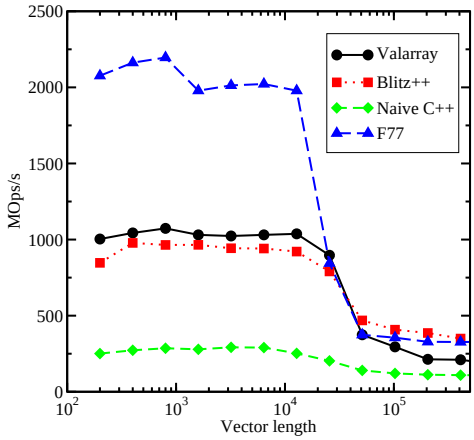


Figure 4: Comparison of different functional specification on the IBM.

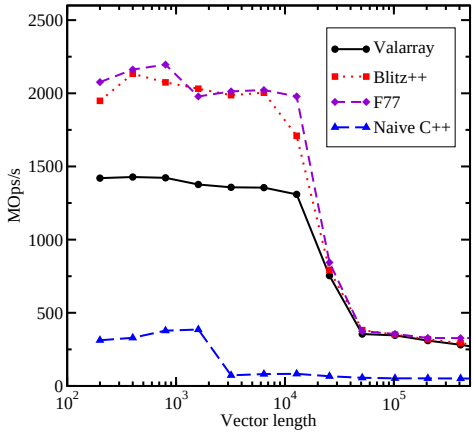


Figure 5: Comparison of different functional specification on the G5.

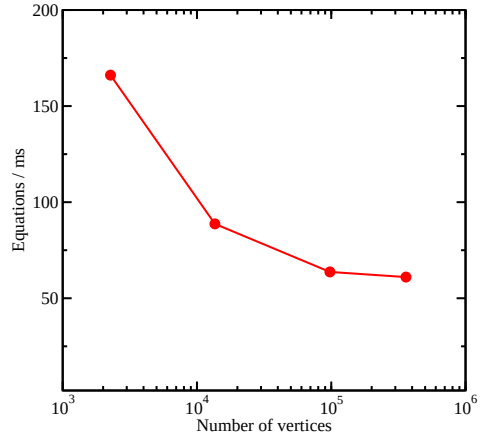


Figure 6: Comparison of the finite element assembly times.

matrices may require access to the same part of memory simultaneously, which could be avoided by a different assembly scheme, e.g. node-based assembly. On the

other hand, the inner loops are compiled very efficiently and only bounded by memory bandwidth. Tests with four CPUs have shown, that parallel assembly does not speed up the total assembly process at all. Restrictions resulting from memory bandwidth completly negate any benefit due to parallelization.

## CONCLUSION

Although the observed performance issues are presented for the field of interconnect simulation, the main findings are certainly transferable to other areas, such as process and device simulation. Memory bandwidth is the limiting factor as we have seen from our benchmarks.
In summary, highly expressive code in C++ on different platforms and computer architectures does not show any abstraction penalty, where naive C++ code does not perform well. Regarding parallelization, current memory links hardly provide enough bandwidth to accommodate the throughput required to satisfy the computational performance of multiple cores.

## REFERENCES


[1] D. Abrahams and A. Gurtovoy. *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series)*. Addison-Wesley Professional, 2004.

[2] A. Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[3] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2005.

[4] R. Bauer. *Numerische Berechnung von Kapazitäten in dreidimensionalen Verdrahtungsstrukturen*. PhD thesis, Technische Universität Wien, 1994.

[5] K. Beyls and E. H. D'Hollander. Generating Cache Hints for Improved Program Efficiency. *J. Syst. Archit.*, 51(4):223–250, 2005.

[6] Boost. *Stream - Sustainable Memory Bandwidth in High Performance Computers*. `http://www.cs.virginia.edu/stream/`.

[7] R. Heinzl and T. Grasser. Generalized Comprehensive Approach for Robust Three-Dimensional Mesh Generation for TCAD. In *Proc. Conf. in Sim. of Semiconductor Processes and Devices*, pages 211–214, Tokio, September 2005.

[8] R. Heinzl, P. Schwaha, M. Spevak, and T. Grasser. Performance Aspects of a DSEL for Scientific Computing with C++. In *Proc. of the POOSC Conf.*, Nantes, France, July 2006.

[9] R. Heinzl, M. Spevak, P. Schwaha, and T. Grasser. A High Performance Generic Scientific Simulation Environment. In *Proc. of the PARA Conf.*, page 61, Umea, Sweden, June 2006.

[10] M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[11] J. Järvi, J. Willcock, and A. Lumsdaine. Concept-Controlled Polymorphism. In *GPCE '03: Proc. of the 2nd Conf. on Generative Prog. and Comp. Eng.*, pages 228–244, New York, NY, USA, 2003. Springer-Verlag New York, Inc.

[12] D. Lacey, N. Jones, E. Van Wyk, and C.C. Frederiksen. Compiler Optimization Correctness by Temporal Logic. *Higher Order and Symbolic Computation*, 17(3):173–206, 2004.

[13] L. Lee and A. Lumsdaine. Generic Programming for High Performance Scientific Applications. In *JGI '02: Proc. of the 2002 joint ACM-ISCOPE Conf. on Java Grande*, pages 112–121, New York, NY, USA, 2002. ACM Press.

[14] S. Lefantzi, J. Ray, and H. N. Najm. Using the Common Component Architecture to Design High Performance Scientific Simulation Codes. In *IPDPS '03: Proc. of the 17th Symp. on Parallel and Distributed Proc.*, page 52, Washington, DC, USA, 2003. IEEE Computer Society.

[15] Rainer Sabelka and Siegfried Selberherr. A Finite Element Simulator for Three-Dimensional Analysis of Interconnect Structures. *Microelectronics Journal*, 32(2):163–171, 2001.

[16] P. Schwaha, R. Heinzl, M. Spevak, and T. Grasser. Advanced Equation Processing for TCAD. In *Proc. of the PARA Conf.*, page 64, Umea, Sweden, June 2006.

[17] J. G. Siek and A. Lumsdaine. The Matrix Template Library: A Unifying Framework for Numerical Linear Algebra. In *ECOOP Workshop*, pages 466–467, 1998.

[18] T. L. Veldhuizen. Arrays in Blitz++. In *Proc. Symp. on Comp. in Obj.-Oriented Parallel Env.*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[19] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*. McGraw-Hill, Berkshire, England, 1987.


## BIOGRAPHIES

**RENÉ HEINZL** studied electrical engineering at the Technische Universität Wien. He joined the Institute for Microelectronics in November 2003, where he is currently working on his doctoral degree. In April 2005 he achieved first place at the doctoral competition at the EEICT in Brno. His research interests include process simulation, solid modeling, and adaptive mesh generation for TCAD with special emphasis on three-dimensional applications.
**PHILIPP SCHWAHA** studied electrical engineering at the Technische Universität Wien. He joined the Institute for Microelectronics in June 2004, where he is currently working on his doctoral degree. His research activities include circuit and device simulation, device modeling, and software development.
**MICHAEL SPEVAK** studied electrical engineering at the Technische Universität Wien. He joined the Institute for Microelectronics in December 2004, where he is currently working on his doctoral degree.
**TIBOR GRASSER** received the Ph.D. degree in technical sciences, and the "venia docendi" in microelectronics from the Technische Universität Wien in 1999, and 2002, respectively. He is currently employed as an Associate Professor at the Institute for Microelectronics. Since 1997 he has headed the Minimos-NT development group, working on the successor of the highly successful MiniMOS program. In 2003 he was appointed head of the Christian Doppler Laboratory for TCAD in Microelectronics, an industry-funded research group embedded in the Institute for Microelectronics. His current scientific interests include circuit and device simulation and device modeling.
**SIEGFRIED SELBERHERR** received the Ph.D. degree in technical sciences from the Technische Universität Wien in 1981. Since that time he has been with the Technische Universität Wien as professor. Dr. Selberherr has been holding the "venia docendi" on "Computer-Aided Design" since 1984. As of 1988 he has been chair professor of the Institut für Mikroelektronik. From 1998 to 2005 he served as Dean of the "Fakultät für Elektrotechnik und Informationstechnik" at the Technische Universität Wien. His current topics of interest are modeling and simulation of problems for microelectronics engineering.