Fast Methods for Computing Selected Elements of the Green's Function in Massively Parallel Nanoelectronic Device Simulations

Andrey Kuzmin[1], Mathieu Luisier[2], and Olaf Schenk[1]

[1] Institute of Computational Science, Universita della Svizzera italiana
CH-6900 Lugano, Switzerland
{andrey.kuzmin,olaf.schenk}@usi.ch,@usi.ch
http://www.ics.inf.usi.ch

[2] Integrated Systems Laboratory, ETH Zürich, CH-8092 Zürich, Switzerland
mluisier@iis.ee.ethz.ch
http://www.iis.ee.ethz.ch

Abstract. The central computation in atomistic, quantum transport simulation consists in solving the Schrödinger equation several thousand times with non-equilibrium Green's function (NEGF) equations. In the NEGF formalism, a numerical linear algebra problem is identified related to the computation of a sparse inverse subset of general sparse unsymmetric matrices. The computational challenge consists in computing all the diagonal entries of the Green's functions, which represent the inverse of the electron Hamiltonian matrix. Parallel upward and downward traversals of the elimination tree are used to perform these computations very efficiently and reduce the overall simulation time for realistic nanoelectronic devices. Extensive large-scale numerical experiments on the CRAY-XE6 Monte Rosa at the Swiss National Supercomputing Center and on the BG/Q at the Argonne Leadership Computing Facility are presented.

1 Introduction

Ultrascaled nanowire field-effect transistors (NWFETs) [1,2] could become the next generation logic devices when it will no longer be possible to scale the dimensions of the currently manufactured fin-shaped field effect transistor (FinFETs) and keep improving their performance. Technology computer aided design (TCAD) has established itself as a great accelerator for the development of novel transistors. However, to simulate the characteristics of NWFETs, it is necessary to go beyond classical approximations such as the drift-diffusion model and to use a quantum transport approach. Energy quantization, quantum confinement, and quantum mechanical tunneling can only be accurately captured if the Schrödinger equation is directly solved in a full-band, atomistic basis and if open boundary conditions describing the coupling of the device with its environment are included [3,4].

F. Wolf, B. Mohr, and D. an Ney (Eds.): Euro-Par 2013, LNCS 8097, pp. 533–544, 2013.

The non-equilibrium Green's function formalism (NEGF) is one of the most efficient techniques for performing this task. It has been widely used to study the electronic and thermal properties of nanoscale transistors and molecular switches on massively parallel architectures. The NEGF formalism is used, e.g., in the highly successful nanoelectronics modeling tools OMEN [1] [5] and TranSIESTA[2] [6]. In these applications, a subset of the entries of the inverse of complex unsymmetric matrices must be repeatedly computed, which represents a significant computational burden. This is usually achieved with a so-called recursive Green's function (RGF) algorithm [7,8].

The calculation of a subset of the entries of the inverse of a given matrix also occurs in a wide range of other applications, e.g., in electronic transport simulation [9,10], the diagonal and sometimes subdiagonal of the discrete Green's function are needed in order to compute electron density. It is therefore of utmost importance to develop and implement efficient scalable algorithms targeting the diagonal of the inverse that are faster than, e.g., inverting the entire matrix based on successive application of a sparse direct LU decomposition of A or faster than the RGF algorithm.

Consider a general sparse unsymmetric matrix $A \in C^{n \times n}$, and assume that A is not singular so that it can be factorized as $A = LU$. If the matrix A is irreducible then A^{-1} is a dense matrix [11]. In this paper, the problem of computing selected elements of the inverse A^{-1} of A is addressed. This subset of selected elements is defined by the set of nonzero entries in the factorized matrix. It was proved in [12] that both the subset and the diagonal of A^{-1} can be evaluated without computing any inverse entry from outside of the subset.

A fast direct algorithm called fast inverse using nested dissection (FIND) that was proposed in [9] is used to compute the required components of the NEGF in the simulations of nanoscale devices. The method is based on the algorithm of nested dissection. A graph of the matrix is constructed and decomposed using a tree structure. An upward and downward traversal of the tree is used to perform the computation efficiently. An alternative method is based on the Schur-complement method described in [13,14]. The fast sequential algorithm proposed for symmetric indefinite matrices was implemented in the Selinv[3] library.

1.1 Contribution

To the best of our knowledge, there is no parallel efficient software package currently available for computing an inverse subset of a general unsymmetric sparse matrix. This paper fills this gap by describing an efficient BLAS level-3 algorithm and its parallel multithreaded implementation. It is available in

[1] OMEN was awarded an honorable mention at the 2011 ACM Gordon Bell Prize for reaching a sustained performance of 1.44 PFlop/s on the Cray-XT5 Jaguar. It is available at http://nanohub.org/resources/omenwire

[2] http://www.icmab.es/dmmis/leem/siesta

[3] https://web.math.princeton.edu/~linlin/software.html

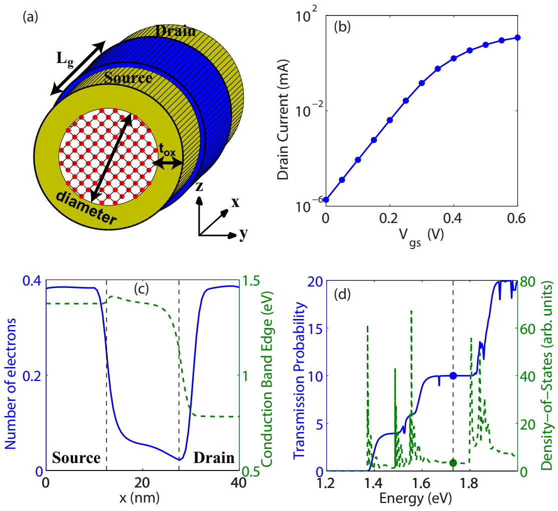

Fig. 1. (a) Schematic view of a Si ultrascaled NWFET composed of source, channel, and drain regions. Each single Si atom (red dot) is taken into account. The central semiconducting nanowire of diameter d is surrounded by an oxide layer of thickness t_{ox}. The gate contact (blue) has a length L_g. Electrons flow from source to drain along the x-axis. (b) Transfer characteristics I_d-V_{gs} (drain current vs. gate-to-source voltage) at a drain-to-source voltage V_{ds}=0.6 V of a Si NWFET with d=3 nm, L_g=15 nm, and t_{ox}=1 nm. (c) Number of electrons (solid blue line) and conduction band profile (dashed green line) along the x-axis of the same Si NWFET as in (b). (d) Electron transmission probability (solid blue line) and density of states (dashed green line) under the same conditions as in (c).

the latest version of the PARDISO[4] package. Extensive intranode performance studies were perfromed on Cray XE-6 and IBM BG/Q architectures. The method is fully integrated into the massively parallel nanoelectronic simulation code OMEN. Numerical experiments show significant advantage of method over the RGF approach.

The paper is organized as follows: in Section 2, an overview of the atomistic, quantum transport simulation approach including its RGF implementation is given. In Section 3 the idea of selected parallel inversion as an alternative to the RGF method is described. The performance of our code is finally described and analyzed in Section 4 before the paper is concluded in Section 5.

2 Overview of the Simulation Approach

2.1 Formulation of the Quantum Transport Problem

Simulating NWFET in a full-band, atomistic basis requires repeatedly solving the Schrödinger equation with open boundary conditions and dealing with large

[4] `https://www.pardiso-project.org`

matrices whose size depends on the number of atoms present in the simulation domain and on the number of atomic orbitals considered on each atom. A typical device structure is schematized in Fig. 1(a). The current flowing through the NWFET at a given gate-to-source (V_{gs}) and drain-to-source (V_{ds}) voltage represents the most relevant quantity of interest since it can be compared to experimental data. An example is given in Fig. 1(b). The electron charge density is also important since it induces an electrostatic potential through the Poisson equation, which in turn affects the Schrödinger equation. It must therefore also be calculated as illustrated in Fig. 1(c). In ballistic simulations, to obtain the current and charge density, the energy-resolved electron transmission probability $T(E)$ and density of states (DOS) $g(E)$ must be first evaluated. They are shown in Fig. 1(d).

In order to calculate $T(E)$ and $g(E)$ and then the electron and charge density, the following NEGF equation must be solved for each electron energy E:

$$\left(E\mathbf{I} - \mathbf{H} - \mathbf{\Sigma}^{RS}(E) - \mathbf{\Sigma}^{RD}(E)\right) \cdot \mathbf{G}^{R}(E) = \mathbf{I}. \tag{1}$$

In Eq. (1), the Hamiltonian matrix $\mathbf{H}$ is of size $N_A \times N_{orb}$, where N_A is the number of atoms and N_{orb} the number of orbitals in the full-band basis used to describe the device material properties. Here, a semiempirical, nearest-neighbor $sp^3d^5s^*$ tight-binding model without spin-orbit coupling is employed, which means that N_{orb}=10 [15]. In many applications, $\mathbf{H}$ is block tridiagonal, but it can also exhibit more complicated structures. The matrix $\mathbf{I}$ is the identity matrix while the self-energy matrices $\mathbf{\Sigma}^{RS}(E)$ and $\mathbf{\Sigma}^{RD}(E)$ refer to the source and drain open boundary conditions respectively, calculated as in [5]. Only the upper left corner of $\mathbf{\Sigma}^{RS}(E)$ and lower right corner of $\mathbf{\Sigma}^{RD}(E)$ are different from 0. In a device with more than two contacts, additional self-energies must be included in Eq. (1). Here, for simplicity, only a source and drain contact are accounted for. Finally, $\mathbf{G}^{R}(E)$ is the retarded Green's Function at energy E.

Once $\mathbf{G}^{R}(E)$ is calculated, either with a standard approach as in Section 2.2 or with the new algorithm proposed in Section 3, the electron density of states $g(E)$ and transmission probability $T(E)$ can be derived. First, $g(E)$ is considered. It contains as many components as contacts so that

$$g(E) = g^{S}(E) + g^{D}(E), \tag{2}$$

where $g^{S}(E)$ and $g^{D}(E)$ refer to the DOS coming from the source and drain, respectively. It can be demonstrated that [3]

$$g(E) = \frac{i}{2\pi}\mathrm{diag}\left(\mathbf{G}^{R}(E) - \mathbf{G}^{R\dagger}(E)\right), \tag{3}$$

$$g^{S}(E) = \frac{1}{2\pi}\mathrm{diag}\left(\mathbf{G}^{R}(E) \cdot \mathbf{\Gamma}^{S}(E) \cdot \mathbf{G}^{R\dagger}(E)\right), \tag{4}$$

$$g^{D}(E) = g(E) - g^{S}(E). \tag{5}$$

Similarly, the transmission probability $T(E)$ is given by

$$T(E) = \mathrm{trace}\left(\mathbf{G}^{R}(E) \cdot \mathbf{\Gamma}^{S}(E) \cdot \mathbf{G}^{R\dagger}(E) \cdot \mathbf{\Gamma}^{D}(E)\right). \tag{6}$$

In both Eqs. (4) and (6), broadening functions $\mathbf{\Gamma}^{S/D}(E)$ are introduced. They are defined as

$$\mathbf{\Gamma}^{S/D}(E) = i\left(\mathbf{\Sigma}^{RS/D}(E) - \mathbf{\Sigma}^{RS/D\dagger}(E)\right). \tag{7}$$

Based on Eqs. (3)–(6) and on the fact that only the upper left corner of $\mathbf{\Sigma}^{RS}(E)$ and $\mathbf{\Gamma}^{S}(E)$ is different from 0, it appears that not the entire $\mathbf{G}^{R}(E)$ matrix is needed, but only its diagonal and its first n columns, where n is the size of the nonzero square block of $\mathbf{\Gamma}^{S}(E)$. This simplification is valid in ballistic simulations only. As soon as scattering is included, e.g., electron-phonon interactions or interface roughness, the situation becomes more complicated. However, such cases are outside the scope of this paper.

With the knowledge of $g^{S}(E)$, $g^{D}(E)$, and $T(E)$, the charge density n_{el} and current I_d can be computed as

$$n_{el} = \int dE \left(g^{S}(E) f(E - E_{fS}) + g^{D}(E) f(E - E_{fD})\right), \tag{8}$$

$$I_d = \frac{q}{\hbar}\int \frac{dE}{2\pi} T(E)\left(f(E - E_{fS}) - f(E - E_{fD})\right), \tag{9}$$

where $f(E)$ is a distribution function (here Fermi-Dirac), E_{fS} and E_{fR} the source and drain Fermi levels, q the elementary charge constant, and $\hbar$ the reduced Planck constant. The electron density n_{el} takes the form of a vector with different values on each atom. The drain current I_d is a scalar.

2.2 RGF Algorithm

When the Hamiltonian matrix $\mathbf{H}$ has a block tridiagonal structure, as in most device simulations, an efficient RGF algorithm can be used to solve Eq. (1) [7,8]. To briefly sketch the functionality of the RGF algorithm, it is assumed that $\mathbf{H}$ contains N diagonal blocks and that M_{ij} is the block with row index i and column index j in the matrix $\mathbf{M}$. The algorithm involves two recursive steps, the first one starting at the lower right corner,

$$g_i^R = \left(E - H_{ii} - H_{ii+1} g_{i+1}^R H_{i+1i}\right)^{-1} C \tag{10}$$

with

$$g_N^R = \left(E - H_{NN} - \Sigma_{NN}^{RD}\right)^{-1}, \tag{11}$$

$$g_1^R = \left(E - H_{11} - H_{12} g_2^R H_{21} - \Sigma_{11}^{RS}\right)^{-1}. \tag{12}$$

The g_i^R's are approximate solutions to Eq. (1) when H_{ii-1} and H_{i-1i} are set to 0. It then becomes clear that the exact Green's Function G_{11}^R is equal to g_1^R. Then, in a second phase, two additional recursions can be derived to calculate the exact diagonal and first column blocks of $\mathbf{G}^R(E)$ starting from the upper left corner:

$$G_{ii}^R = g_i^R + g_i^R H_{ii-1} G_{i-1i-1}^R H_{i-1i} g_i^R, \tag{13}$$

$$G_{i1}^R = g_i^R H_{ii-1} G_{i-11}^R. \tag{14}$$

No other system of equations must be solved to evaluate Eqs. (3)–(6). Note finally that the computational complexity of the RGF algorithm amounts to $O(Nn^3)$, where n is the average block size, and that it cannot be efficiently parallelized. This becomes a serious issue in large simulation domains, for which the required memory to solve Eqs. (10)–(14) is larger than the one available on each CPU.

3 A Selected Sparse Inverse Matrix Algorithm (SINV)

3.1 Sparse Inverse Supernodal Factorization

The obvious way to compute selected entries of the inverse is to invert the entire matrix and than to extract the selected entries. The standard approach for matrix inversion is to perform the LU factorization first:

$$A = LU,$$

where L and U are unit lower triangular and upper triangular matrices respectively. Using such a factorization, $A^{-1} = (x_1, x_2, ..., x_n)$ could be obtained by solving a number of linear systems $Ax_i = e_i$. Each of the systems is solved using backward and forward substitution phases, $Ly = e_j$ and $Ux_j = y$. Before the algorithm is presented, the process of computing LU factorization is reviewed. Let A be a nonsingular unsymmetric matrix. Each step of LU factorization produces the following decomposition:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} I & \\ & S \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ & I \end{bmatrix},$$

where $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is the Schur-complement of A_{11} with respect to A. In order to simplify the derivation, it is assumed that no row or column permutation is required during the factorization. The discussion could be easily generalized for the case of pivoting algorithms used in order to improve stability. The main idea of the algorithm is that A^{-1} could be computed using the following expression:

$$A^{-1} = \begin{bmatrix} U_{11}^{-1}L_{11}^{-1} + U_{11}^{-1}U_{12}S^{-1}L_{11}^{-1}L_{21} & -U_{11}^{-1}U_{12}S^{-1} \\ -S^{-1}L_{11}^{-1}L_{21} & S^{-1} \end{bmatrix}.$$

Using the notation $\tilde{U}_{12} = -U_{11}^{-1}U_{12}$, $\tilde{L}_{12} = -L_{11}^{-1}L_{12}$ and $\tilde{D}^{-1} = U_{11}^{-1}L_{11}^{-1}$ this expression for the inverse can be simplified:

$$A^{-1} = \begin{bmatrix} \tilde{D}^{-1} + \tilde{U}_{12}S^{-1}\tilde{L}_{21} & \tilde{U_{12}}S^{-1} \\ -S^{-1}\tilde{L}_{21} & S^{-1} \end{bmatrix}.$$

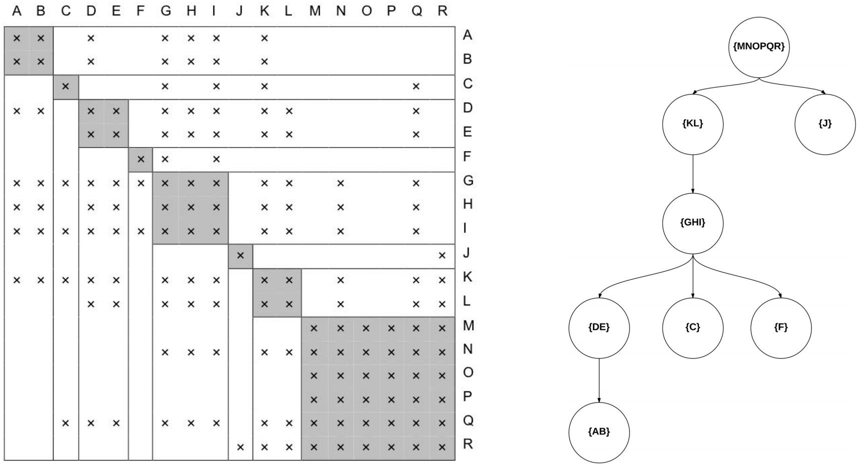

Fig. 2. Example of a supernodal partition for sparse LU factorization and the corresponding supernodal elimination tree

The idea was originally proposed in [16]. The more general approach for unsymmetric matrices is presented in this paper. This expression suggests that given the LU factorization, computation of the inverse can be reduced to computing S^{-1}. The computation of A^{-1} can be organized in a recursive manner similar to the LU factorization algorithm, but computing the sequence of the diagonal elements is organized in the opposite direction. The last diagonal element of the inverse equals a reciprocal of the last diagonal element of the U factor, $A^{-1}_{nn} = (U_{nn})^{-1}$. Starting from the last element, which is also the Schur-complement produced in the last step of the LU factorization algorithm, we proceed step by step, computing more and more blocks from the lower right corner to the upper left corner. Thus, more and more entries of A^{-1} are computed. The complexity of such an inversion algorithm is still $O(n^3)$ in the case of a dense matrix. However, computational cost can be drastically reduced if the matrix A is sparse. Rigorous consideration of this topic leads to the concept of sparse matrix elimination tree and its extension to the inverse factorization case [17,18,19]. As a consequence the complexity of the inversion process can be reduced to $O(n^2)$ for matrices from three-dimensional simulations.

The implementation is built on top of the PARDISO package that uses supernodal BLAS level-3 algorithm during the direct factorization phase. The METIS package is used to produce the fill-in reducing reordering [20]. An example of such a supernodal partitioning for LU factorization can be seen in Fig. 2. The nested dissection algorithm allows to reduce significantly the size of diagonal subblocks compared to RGF method that results in additional performance increase. Nonzero entries of LU factors are stored in dense subblocks, so that each supernode consists of many dense submatrices that could be used in efficient dense matrix-matrix multiplication subroutines.

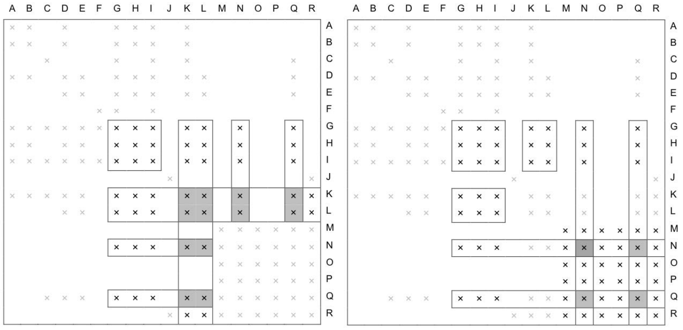
A B C D E F G H I J K L M N O P Q R
A B C D E F G H I J K L M N O P Q R
A B C D E F G H I J K L M N O P Q R
A B C D E F G H I J K L M N O P Q R

Fig. 3. Example of a dense subblock gathering during the inverse factorization process. The scheme on the left-hand side depicts the submatrix that contributes to the update of the supernode {G,H,I} by the supernode {K,L}. Contribution of the supernode {N,O,P,Q,R} to {G,H,I} is represented on the right-hand side.

In the implementation, the inverse factorization is computed in place so that the original LU factorization is overwritten by the entries of the inverse. The most computationally time consuming operations at this stage are two matrix-matrix products, namely $\tilde{U_{12}}S^{-1}$ and $-S^{-1}\tilde{L}_{21}$ where S^{-1} is a sparse matrix with supernodal storage and $\tilde{L}_{21}$, $\tilde{U}_{12}$ are, in turn, sparse block vectors stored by contiguous dense subblocks. First the computation of the product $\tilde{U_{12}}S^{-1}$ is considered. It could be split according to the supernodal partition of the Schur-complement: $\tilde{U_{12}}S^{-1} = \tilde{U_{12}}S^{-1}_{\{A,B\}} + \tilde{U_{12}}S^{-1}_{\{C\}} + \ldots + \tilde{U_{12}}S^{-1}_{\{M,N,O,P,Q,R\}}$, where $S^{-1}_{\{A,B\}}$, $S^{-1}_{\{C\}}$, ... are supernodes of S^{-1} consisting of a lower and upper triangular part each. Each of the terms in the sum can be computed in two steps. First the required entries are gathered into a dense block using indirect addressing schemes similar to techniques described in [21] (see Fig. 3). Then, the product is computed using two dense matrix-matrix multiplications. The sum is accumulated in a temporary buffer. After $\tilde{U_{12}}S^{-1}$ has been computed, the product $\tilde{U}_{12}S^{-1}\tilde{L}_{21} = (\tilde{U}_{12}S^{-1})\tilde{L}_{21}$ could be immediately calculated using the xGEMM function. The product $-S^{-1}\tilde{L}_{21}$ is calculated in a similar manner. Thus, all the computations were performed using BLAS level-3 subroutines that guarantees optimal use of vector operations on modern architectures and effecient usage of their cache hierarchies.

3.2 Intranode Parallelization

Data dependencies in the selected inversion algorithm are represented by the inverse elimination tree. Therefore, the inversion algorithm permits parallelization strategies similar to parallel direct solvers. The level of parallelism mainly utilized in this case is the tree level parallelism.

Consider the serial inversion algorithm described in the previous section. Factorization for each supernode consists of two parts. The first part could be called internal inverse factorization (computing $\tilde{U}_{12}$, $\tilde{L}_{21}$ and $\tilde{D}^{-1}$) since it is done independently. The second part (computing $\tilde{U_{12}}S_{\{I\}}^{-1}$ and $-S_{\{I\}}^{-1}\tilde{L_{21}}$ for each supernode of S) accumulates external contributions from other supernodes and could be called the external update phase. In order to parallelize the algorithm, data dependencies must be maintained. This suggests creation of the global tasks queue. Each element of the queue is a supernode and the size of the queue is bounded by the total number of supernodes. Tasks distribution is performed dynamically: each thread fetches one element S_j from the queue and proceeds with the internal inversion phase. Computation of external contributions requires synchronization with threads working on the descendants of S_J, i.e., the thread waits until the inversion of each of the dependents is finished.

4 Numerical Experiments

4.1 Experimental Testbase

In this section the parallel performance and efficiency of OMEN equipped with the PARDISO selected inversion implementation for the NEGF equation is reported. Before moving on to the parallel scalability of the atomistic, quantum transport simulation benchmarks, this section gives a brief description of the target hardware, namely, IBM BG/Q at the Argonne Leadership Computing Facility and a Cray XE6 system installed at the Swiss National Supercomputing Center CSCS. Intranode experiments were performed on one rack of the "Mira" BG/Q which has a 1.6-GHZ 16-way quad-core PowerPC processor and 16 GB of RAM. The Cray XE6 "Monte Rosa" compute nodes were used for intranode performance experiments. The Cray XE6 consists of 2 16-core AMD Opteron 6272 2.1-GHz Interlagos processors, giving 32 cores in total per node with 32 GBytes of memory. The Interlagos CPUs implement AMD's recent "Bulldozer" microarchitecture and each Interlagos socket contains two dies, each of which contains four so-called "modules."

4.2 Intranode Performance

In this section the results on performance of the inversion algorithm based on matrices from the NEGF solver implemented in OMEN are presented. Table 1 shows the speedup of the selected inversion over the full inversion algorithm for the set of 4 Hamiltonian matrices of the size 97900 to 581900 uknowns with LU factors containing around 10^6 nonzero elements. The advantage over the full inversion grows as the problem size increases that makes it practically possible to solve the problem with more than 10^6 nonzero elements. The new method is more than 250 times faster on both architectures for the largest matrix.

Table 2 demonstrates the scalability of the inversion algorithm for the set of 2 largest matrices on one node with 2 to 32 threads compared to the RGF method that was previously used in OMEN. The observed scalability is comparable to that of the direct factorization algorithm. The shared-memory parallel efficiency of the PARDISO-SINV implementation is considerable and compelling for larger

Table 1. Average time in seconds (and Gflop/s in brackets) in PARDISO and PARDISO-SINV on 32 cores when computing all diagonal entries of the inverse of A for four selected OMEN test matrices; n represents the size of the Hamiltonian matrix

n	CRAY XE6		BG/Q	
	PARDISO	PARDISO-SINV	PARDISO	PARDISO-SINV
97900 (d=2nm)	2434.0	0.8 (51)	4174.0	3.0 (22)
212300 (d=3nm)	4198.0	7.5 (89)	19571.1	8.7 (45)
375100 (d=4nm)	20525.1	42.7 (69)	65847.6	70.7 (63)
581900 (d=5nm)	26657.1	102.1 (119)	62296.7	217.2 (68)

Table 2. Average time in seconds to calculate the density of states and the transmission probability for one energy point, as indicated by the dashed line in Fig. 1(d), for nanowire transistors with two different diameters (4 and 5 nm). The first column gives the Hamiltonian matrix size in Eq. (1), the second the number of cores, the third the solution time with the RGF algorithm [7,8], while columns 4 to 9 are dedicated to the new approach proposed in this paper. The times to reorder the Hamiltonian matrix, factorize it, compute selected elements of its inverse, solve it to obtain G^R_{N1}, and derive the DOS with Eq. (4) are reported. The last column represents the sum of columns 4 to 8.

n	Cores	RGF	Reordering	Factorization	SINV	G^R_{N1}	Eq. (4)	Total
375100 (d=4nm) IBM BG/Q	1	4179.1	55.85	601.53	1122.41	1601.13	-	3380.92
	2	-	56.01	251.55	463.25	760.15	-	1530.96
	4	-	55.63	147.91	300.77	440.86	-	945.17
	8	-	54.99	67.60	162.49	237.69	-	522.77
	16	-	54.71	37.89	84.72	150.25	-	327.57
	32	-	55.85	28.72	67.56	112.17	-	264.3
581900 (d=5nm) IBM BG/Q	1	16644.2	92.93	2042.29	3393.81	5057.64	-	10586.67
	2	-	93.21	818.52	1799.18	2655.33	-	5366.24
	4	-	93.13	460.88	996.35	1790.10	-	3340.46
	8	-	91.49	229.03	622.89	1089.41	-	2032.82
	16	-	91.61	120.61	328.37	768.19	-	1308.78
	32	-	91.86	90.82	217.21	723.23	-	1123.12
375100 (d=4nm) Cray XE6	1	1393.9	15.6	222.7	392.2	1149.5	390.1	2191.3
	2	-	16.6	163.5	311.4	671.4	327.7	1515.6
	4	-	16.4	83	171	215.9	165.4	677.3
	8	-	17.5	47.1	105.1	108.6	87.7	394.4
	16	-	17.5	25.6	64.2	59.6	45.6	242.1
	32	-	17	15.5	13.7	51.7	24.1	153.5
581900 (d=5nm) Cray XE6	1	5548.9	23.5	697.8	1250.5	1466.1	1428.4	4892.7
	2	-	25.4	533.5	1033.3	1072.1	1171.7	3865.9
	4	-	25.5	270.1	529.2	564.2	609.2	2029.2
	8	-	27.6	153.1	333.5	328.6	322.3	1199.4
	16	-	27.2	79.2	166.9	302.3	168.2	779.7
	32	-	27.7	46.7	99.3	283.7	91.8	587.9

problems (e.g., d=5 nm). The experiments show that the new implementation has the performance lower or comparable to RGF method when using 1 or 2 threads; however the RGF implementation has a low potential for scalability and the new approach is one order of magnitude faster when using 32 threads. These results show that the selected inversion algorithm can be very efficiently applied in large-scale computational nanolectronics, significantly reducing the overall simulation time for realistic devices.

5 Conclusion

The recursive Green's function algorithm that is typically used in large-scale atomistic nanoelectronic device engineering has good algorithmic efficiency in the order of $O(N_z \cdot n^3)$, where $n = n_x \cdot n_y$ is the average block size, but significant disadvantages in terms of parallelism. An alternative method based on parallel selected inversion has been presented in this paper for the NEGF that is the central computation in atomistic, quantum transport simulations. The complexity of the selected inversion method is in the order of $O(N_z^2 \cdot n^2)$. It is used to extract all diagonal entries of the inverse of a complex sparse unsymmetric matrix. The new selected inversion method overcomes the scalabilty barrier of RGF by using parallel upward and downward traversals of the elimination tree to solve the NEGF equations very efficiently. The implementation of the inversion solver showed substantial speedup over the previous approach. PARDISO-SINV solved realistically sized examples in OMEN in about 5 min compared to over 1.5 hour on the Cray XE6.

Acknowledgment. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. We acknowledge the Swiss National Supercomputing Center and for giving us access to their hardware and for their support. The work done in Switzerland is funded partially under the HP2C initiative. Mathieu Luisier acknowledges the support of SNF grant PPOOP2_133591.

References

1. Cui, Y., Zhong, Z., Wang, D., Wang, W., Lieber, C.: High performance silicon nanowire field effect transistors. Nano Letters 3, 149–152 (2003)
2. Singh, N., Agarwal, A., Bera, L., Liow, T., Yang, R., Rustagi, S., Tung, C., Kumar, R., Lo, G., Balasubramanian, N., Kwong, D.: High-performance fully depleted silicon nanowire (diameter $\leq$ 5 nm) gate-all-around CMOS devices. IEEE Electron Device Letters 27, 383–386 (2006)
3. Datta, S.: Electronic transport in mesoscopic systems. Press Syndicate University of Cambridge, Cambridge (1995)
4. Cauley, S., Jain, J., Koh, C., Balakrishnan, V.: A scalable distributed method for quantum-scale device simulation. J. Appl. Phys. 101, 123715 (2007)

5. Luisier, M., Klimeck, G., Schenk, A., Fichtner, W.: Atomistic simulation of nanowires in the $sp^3d^5s^*$ tight-binding formalism: from boundary conditions to strain Calculations. Phys. Rev. B 74, 205323 (2006)
6. Brandbyge, M., Mozos, J., Ordejon, P., Taylor, J., Stokbro, K.: Density-functional method for nonequilibrium electron transport. Phys. Rev. B 74, 165401 (2002)
7. Lake, R., Klimeck, G., Bowen, R., Jovanovic, D.: Single and multiband modeling of quantum electron transport through layered semiconductor devices. J. Appl. Phys. 81, 7845–7869 (1997)
8. Svizhenko, A., Anantram, M., Govindan, T., Biegel, R., Venugopal, R.: Two-dimensional quantum mechanical modeling of nanotransistors. J. Appl. Phys. 91, 2343–2354 (2002)
9. Li, S., Ahmed, S., Klimeck, G., Darve, E.: Computing entries of the inverse of a sparse matrix using the FIND algorithm. J. Comp. Phys. 227, 9408–9427 (2008)
10. Petersen, D.A., Song, L., Stokbro, K., Sorensen, H.H.B., Hansen, P.C., Skelboe, S., Darve, E.: A Hybrid Method for the Parallel Computation of Green's Functions. J. Comp. Phys. 228(14), 5020–5039 (2009)
11. Duff, I., Erishman, A.: Sparsity structure and Gaussian elimination. ACM SIGNUM Newsletter 23, 2–8 (2006)
12. Erishman, A., Tinney, W.: On computing certain elements of the inverse of a sparse matrix. Comm. ACM 18, 177 (1975)
13. Lin, L., Lu, L., Ying, J.: Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems. Comm. Math. Sci. 7, 755–777 (2009)
14. Lin, L., Yang, C.: Selinv - an algorithm for selected inversion of a sparse symmetric matrix. ACM Trans. on Math. Software 37 (2011)
15. Slater, J., Koster, G.: Simplified LCAO method for the periodic potential problem. Phys. Rev. 94, 1498–1524 (1954)
16. Takahashi, L., Fagan, J., Chin, M.: Formation of a sparse bus impedance matrix and its application to short circuit study. In: Proc. 8th PICA Conference, Minneapolis, Minnesota, pp. 63–69 (1973)
17. Campbell, Y., Davis, T.: Computing the Sparse Inverse Subset: an Inverse Multifrontal Approach. Technical Report TR-95-021, Computer and Information Sciences Department, University of Florida (1995)
18. Amestoy, P., Duff, I., Robert, Y., Rouet, F., Ucar, B.: On Computing Inverse Entries of a Sparse Matrix in an Out-of-Core Environment. Technical Report TR/-PA/10/59, CERFACS, Toulouse, France (2010)
19. Slavova, T.: Parallel Triangular Solution in the Out-of-Core Multifrontal Approach for Solving Large Sparse Linear Systems. Ph.D. dissertation, Institut National Polytechnique de Toulouse, Toulouse, France (2009)
20. Karypis, G., Kumar, V.: A fast and highly quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Computing 20, 359–392 (1999)
21. Schenk, O., Gärtner, K.: Solving unsymmetric sparse systems of linear equations with PARDISO. Future Gener. Comp. Systems 20, 475–487 (2004)