Journal of Computational Physics 231 (2012) 1293–1313

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A two-dimensional domain decomposition technique for the simulation of quantum-scale devices

Stephen Cauley [a,*], Venkataramanan Balakrishnan [b], Gerhard Klimeck [b], Cheng-Kok Koh [b]

[a] Athinoula A. Martinos Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital, Charlestown, MA 02129, USA
[b] School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-2035, USA

ARTICLE INFO

Article history:
Received 16 December 2009
Received in revised form 16 September 2011
Accepted 6 October 2011
Available online 19 October 2011

Keywords:
NEGF
Parallel
Density of states
Atomistic

ABSTRACT

The simulation of realistically sized devices under the Non-Equilibrium Greens Function (NEGF) formalism typically requires prohibitive amounts of memory and computation time. In order to meet the rising computational challenges associated with quantum-scale device simulation we offer a 2-D domain decomposition technique. This technique is applicable to a large class of atomistic and spatial simulation problems. Considering a decomposition along both the cross section and length of the device, the framework presented in this work ensures efficient distribution of both memory and computation based upon the underlying device structure. As an illustration we stably generate the density of states and transmission, under the NEGF formalism, for the atomistic-based simulation of square 5 nm cross section silicon nanowires consisting of over one million atomic orbitals.

1. Introduction

The problem of computing density of states and transmission through the NEGF formalism reduces to a mathematical problem of finding certain entries from the inverse of a typically large and sparse matrix. Although there has been research into numerically stable and computationally efficient serial computing methods [1] for the NEGF approach, when analyzing certain device geometries this type of method will result in prohibitive amounts of computation and memory consumption. In [2] a parallel divide-and-conquer algorithm (PDIV) was shown to be effective for NEGF-based simulation problems. Two applications were presented, the atomistic level simulation of silicon nanowires and the 2-D simulation of nanotransistors. However, similar to the single processor approach presented in [1], the PDIV algorithm was biased toward specific device geometries. Alternative, serial computing, NEGF-based approaches such as [3] rely on specific problem structure (currently not capable of addressing atomistic models), with computation limitations that again restrict the size of simulation. An approach that is similar to [3] in theory and computational complexity is demonstrated in [4]. This work shows promising scaling through the use of parallel computing resources, but the performance again is restricted to specific device structures. In addition to [3,4] that focused on effective mass approximated nano scale device simulation there have been alternate approaches, in other areas of research, for determining certain entries from a matrix inverse [5,6]. It is important to note that domain decomposition strategies such as the one presented in this work rely on solving smaller versions of the same underlying mathematical problem. Thus, any applicable version of these alternate computational methods could readily be included within our parallel framework.

* Corresponding author.
E-mail addresses: stcauley@nmr.mgh.harvard.edu, stcauley@ecn.purdue.edu (S. Cauley), ragu@ecn.purdue.edu (V. Balakrishnan), gekco@purdue.edu (G. Klimeck), chengkok@ecn.purdue.edu (C.-K. Koh).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.10.006

Nomenclature

N_x	cross-sectional size for the device. The number of elements (atomic orbitals or grid points) in each grouping along the transport direction
N_y	length of the device. The number of groupings or layers (consisting of orbitals or grid points) along the transport direction
K	the block tridiagonal coefficient matrix associated with the device. Based on the NEGF formalism, the matrix $K = (EI - H - \Sigma_1 - \Sigma_2)$. The matrix has N_y blocks each of size N_x
E	the energy of interest
H	the Hamiltonian matrix describing the relationship between each orbital or grid point
Σ_1, Σ_2	the left and right (first and last layer) boundary conditions
$\{A\}$, $\{C\}$	the diagonal and off-diagonal blocks that constitute the matrix K
G^r	the retarded Green's function, where $KG^r = I$
$\{D\}$, $\{R\}$, $\{S\}$	the generator or ratio matrices that can be used to construct G^r
p_y	number of domains/processors assigned along the length or transport device dimension
p_x	number of domains/processors assigned along the cross-sectional device dimension
p	total number of domains/processors for the 2-D decomposition of the device. This is a product based upon the number of separations along each dimension $p = p_x p_y$
d_i	the ith separator location along the length of the device, where $d_1 = 0$ and $d_{p_y+1} = N_y$
f_l^k	distribution factors for each layer l assigned to cross-sectional domains $1 \leqslant k \leqslant p_x$. The factors determine how the domain separators are drawn through neighboring layers
ϕ_i^j	the coefficient matrix corresponding to the domain for the $1 \leqslant i \leqslant p_y$ separation along the device length and the $1 \leqslant j \leqslant p_x$ separation along the device cross section
$P_x(\cdot)$	The permutation operator that re-orders the sub-matrix ϕ_i (corresponding to the ith domain along the length) sequentially by cross-sectional domains $\phi_i^1, \ldots, \phi_i^{p_x}$
C_i^j	the matrix of Hamiltonian connectivity between the elements of ϕ_i^j and ϕ_i^{j+1}, based upon the ordering of $P_x(\phi_i)$
$\mathscr{I}_i^j$, $\mathscr{J}_i^j$	the sets of rows and columns for C_i^j that contain non-zero entries
W_r, W_c	the upper bound for the cardinality of the sets $\mathscr{I}_i^j$ and $\mathscr{J}_i^j$. For a regular 2-D grid the values $W_r = W_c = N_y/p_y$
$\{X\}$	the cross-sectional mapping terms required to update the diagonal for each $\left[\phi_i^j\right]^{-1}$, in order to determine the diagonal for $[\phi_i]^{-1}$
$\{Y^k\}$	the boundary mapping terms required to transition from reconstruction along the cross section to reconstruction along the transport direction. There are $1 \leqslant k \leqslant p_x$ sets of boundary maps
$\{M\}$	the transport direction mapping terms required to update the diagonal for each $[\phi_i]^{-1}$, in order to determine the density of states for G^r

The prohibitive computational properties of NEGF-based simulation has prompted a transition to wave function-based methods, such as those presented in [7]. Given an assumed basis structure the problem translates from calculating certain entries from the inverse of a matrix to solving large sparse systems of linear equations. This is an attractive alternative because solving large sparse systems of equations is one of the most well studied problems in applied mathematics and physics. In addition, popular algorithms such as UMFPACK [8] and SuperLU [9] have been constructed to algebraically (based solely on the matrix) exploit problem-specific structure in an attempt to minimize the amount of computation. The performance of these algorithms for the wave function-based analysis of several silicon nanowires has been examined in [10]. However, wave-function methods are currently unable to address more sophisticated analyses that involve electron (phonon) scattering. Another approach [11] attempts to solve these problems by considering the use of the Pauli equation, as was demonstrated in the context of a 1-D device. Thus, there remains a strong need to further develop NEGF-based algorithms for the simulation of realistically sized devices considering these general modeling techniques.

In this work, we extend the approach of [2] to consider a 2-D decomposition of the device in order to reduce the computational bias toward specific device geometries. That is, the algorithms presented in [1,2] both scale linearly with respect to device length, but require a large number of cubic matrix inversion operations based upon the cross-sectional size of the device (for simplicity we represent dense matrix inversion as cubic order). We demonstrate that as the cross-sectional size of the device grows, considering a fixed number of processors, it becomes increasingly advantageous to allow for cross-sectional separations. The 2-D decomposition technique (PDIV-2D) sets a foundation for the NEGF-based simulation of realistically sized devices through the use of increasingly available distributed computer resources. The computational efficiency offered by the method is illustrated as we stably generate the dynamics for square 5 nm cross section silicon nanowires through the use of an atomistic model consisting of over one million atomic orbitals.

2. Mathematical preliminaries

Computing the density of states using the NEGF formalism for many tight-binding and spatial models translates into the problem of finding the diagonal entries of G^r [12], where $KG^r = I$ and K is a block tridiagonal matrix of the form

$$K = (EI - H - \Sigma_1 - \Sigma_2) = \begin{pmatrix} A_1 & -C_1 & & & \\ -C_1^T & A_2 & -C_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -C_{N_y-2}^T & A_{N_y-1} & -C_{N_y-1} \\ & & & -C_{N_y-1}^T & A_{N_y} \end{pmatrix}. \quad (1)$$

Here, E represents the energy of interest, H is the Hamiltonian matrix, with Σ_1 and Σ_2 being the left and right contact terms, respectively. The diagonal blocks $A_i \in \mathbb{C}^{N_x^{(i)} \times N_x^{(i)}}$, the off-diagonal blocks $C_i \in \mathbb{C}^{N_x^{(i)} \times N_x^{(i+1)}}$, and the notation $K = \mathrm{tri}(A_{1:N_y}, C_{1:N_y-1})$ can be used to compactly represent such a matrix. Fig. 1 shows an atomistic representation of a silicon nanowire with a square cross section of 2 nm. The connectivity seen in the semiconductor lattice exits between atomic orbitals in neighboring layers. This modeling of the nanowire results in a block tridiagonal Hamiltonian matrix. As can be seen in Fig. 1 the number of layers in the nanowire corresponds to the number of blocks in the matrix N_y. The number of atomic orbitals within each layer l corresponds to the block size N_x^l. The sparsity pattern seen in the matrix is determined by the lattice itself. Any atoms shown to be connected will have corresponding non-zero entries in the Hamiltonian.

The inverse of a block tridiagonal matrix can be computed explicitly, as demonstrated in [13]. Here, the diagonal blocks of the inverse, D_i, can be computed in a numerically stable fashion [14] using two sequences of "ratio" matrices $\{R_i\}$, $\{S_i\}$. If uniform block sizes are assumed, the time complexity associated with determining the ratio sequences (R_i and S_i) and the diagonal blocks (D_i) is $O\left(N_x^3 N_y\right)$, with a memory requirement of $O\left(N_x^2 N_y\right)$. An alternative formulation with the same computational and memory complexities was presented in [1]. Often these sequences are called "generators" for the inverse matrix because they allow for one to compute any entry through a "compact" representation. Details relating to both determining and utilizing this compact representation are provided in Appendix A.

In [2] a parallel divide-and-conquer framework was developed to determine the diagonal entries of G^r. In that work the block tridiagonal matrix referred to in (1) was separated along the device length, producing p_y sub-matrices. Fig. 2 shows $p_y + 1$ separators $d_1 < d_2 < \cdots < d_{p_y+1}$, where $d_1 = 0$ and $d_{p_y+1} = N_y$. The separators are used to assign a portion of the N_y total blocks to each of the p_y available processors. Each processor i was responsible for a block tridiagonal sub-matrix $\phi_i = \mathrm{tri}(A_{d_i+1:d_{i+1}}, C_{d_i+1:d_{i+1}-1})$. Fig. 3 shows a 1-D decomposition illustration for a block tridiagonal matrix into four sub-matrices. It is important to note that the 1-D decomposition scheme does not directly account for connectivity between the sub-matrices ϕ_i. That is, there are off-diagonal blocks that are not covered by this decomposition, namely $\{C_{d_2}, C_{d_3}, \ldots, C_{d_{p_y}}\}$, which we call bridge matrices. These bridge matrices are shown as the shaded blocks in the decomposition stage of Fig. 3. It was demonstrated in [2] that the bridge matrices, along with the compact representation for each sub-matrix inverse, allow for the accumulation of information needed to reconstruct the device characteristics. Through the introduction of mapping terms this accumulation can be performed efficiently by employing a standard radix-2 combining process. Following induction-based logic, the set of mapping terms takes information from $[\phi_i]^{-1}$ as an input and reconstructs relevant information from an accumulation of domains (based upon the position in the combining process). Specifically, the recursively updated matrix maps presented in [2] allow for each processor to collect any information needed to produce the diagonal and boundary elements from K^{-1}. Fig. 3 illustrates the efficiency of the matrix mapping process for four sub-matrices of the block tridiagonal matrix K.

The time complexity of the algorithm presented in [2] is $O\left(\frac{N_x^3 N_y}{p_y} + N_x^3 \log_2 p_y\right)$, with memory consumption $O\left(\frac{N_x^2 N_y}{p_y} + N_x^2\right)$. The first term $\left(\frac{N_x^3 N_y}{p_y}\right)$ in the computational complexity arises from the embarrassingly parallel nature of both determining the generators and applying the matrix maps to update the diagonal entries of K^{-1}. The second term $\left(N_x^3 \log_2 p_y\right)$ is dependent on the number of hierarchical levels ($\log_2 p_y$) needed to collect combining information from the p_y sub-matrices. Similarly, the first term in the memory complexity is due to the generators, and the second represents the memory required for the matrix maps associated with the sub-matrix. By limiting the decomposition to only separations along the length of the device, the parallel algorithm presented in [2] is still heavily dependent on $O(N_x^3)$ matrix inversion operations. This can be seen

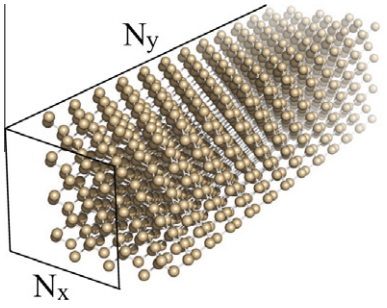

Fig. 1. Silicon nanowire with square cross section size of 2 nm. The dimension of the associated block tridiagonal Hamiltonian matrix is determined by the number of layers N_y and the number of atomic orbitals N_x^l for each layer l.

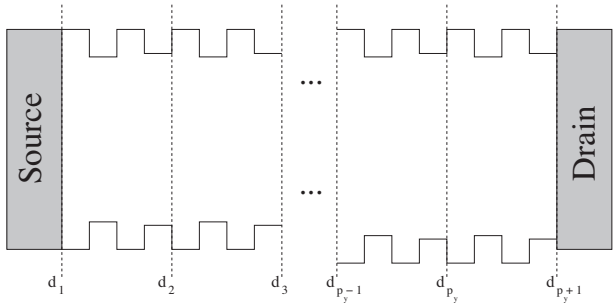

Fig. 2. Separator locations for 1-D decomposition along the length of the device. Each domain i corresponds to a block tridiagonal sub-matrix $\phi_i = \text{tri}(A_{d_i+1:d_{i+1}},\ C_{d_i+1:d_{i+1}-1})$.

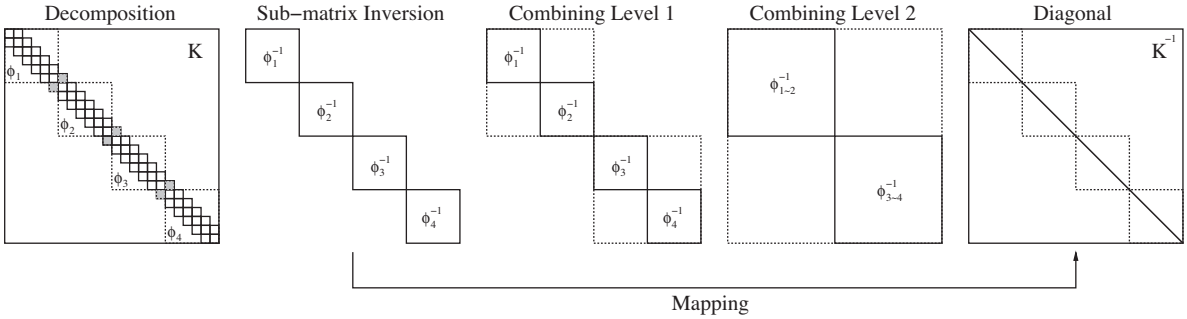

Fig. 3. Decomposition of block tridiagonal matrix K into four sub-matrices, where the shaded blocks correspond to the bridge matrices. The two combining levels follow the individual sub-matrix inversions, where $\phi_{i\sim j}^{-1}$ represents the inverse of sub-matrices ϕ_i through ϕ_j from the matrix K. Matrix mappings are used to capture the combining effects and allow for the direct computation of the diagonal portion of K^{-1}.

simply by observing that each sub-matrix $\phi_j = \text{tri}(A_{d_j+1:d_{j+1}}, C_{d_j+1:d_{j+1}-1})$ has the same block sizes as the original matrix K. In order to facilitate the simulation of large cross-sectional devices (i.e. large N_x values), such is the case with the atomistic simulation of silicon nanowires using the $sp^3d^5s^*$ tight-binding model [15,16], we offer a generalized 2-D domain decomposition technique. The method presented in this work allows for the calculation of the density of states and transmission considering a decomposition along both the cross section and length of the device. Specifically, an additional dimension of flexibility is introduced by separating each of the domains ϕ_i described above into p_x cross-sectional domains ϕ_i^j, where $1 \leqslant j \leqslant p_x$. This allows for a total of $p = p_x p_y$ processors to be assigned for the simulation task. Both the time and memory complexities are substantially reduced when compared to the algorithm of [2]. For simplicity of illustration, if we assume that matrix inversion and dense matrix–matrix multiplication are both of cubic order, then the time complexity of the proposed approach is:

$$O\left(\left(\frac{N_y}{p_y}\right)^3\left[\left(\frac{N_x}{p_x}\right)+\log_2 p_x\right]+\left(\frac{N_y}{p_y}\right)^2\left(\frac{N_x^2}{p_x}\right)+N_x^3\left[\left(\frac{N_y}{p}\right)+\log_2 p_y\right]\right),$$

with memory consumption $O\left(\frac{N_x^2 N_y}{p}+N_x^2\right)$. For large cross-sectional devices of interest, the number of layers N_y can be much smaller than the cross-sectional size of the device N_x. Assuming a 1-D decomposition we are limited to $p = p_x p_y = (1)p_y = p_y$ processors to distribute memory across, where each machine will often have only 2 GB per core. With our 2-D approach we can meet these stringent memory requirements by allowing for $p_x > 1$ and assigning more processors to the task. We compare the computational trade-off for both NEGF and wave function-based approaches. The efficiency of our algorithm is illustrated as we consider large (up to 5 nm) cross section nanowires with over one million atomic orbitals. Although the parallel decomposition framework presented here is applicable to many device geometries and materials, for ease of illustration we limit the focus of this work to silicon nanowires. We will consider a [100] orientation for the semiconductor lattice as defined through the standard Miller indices.

3. Two-dimensional decomposition and reconstruction

In order to motivate the 2-D decomposition and reconstruction technique we begin with a small Si [100] nanowire consisting of 8 layers with a square cross section size of 2 nm. The Si [100] nanowire is periodic every four layers along the transport

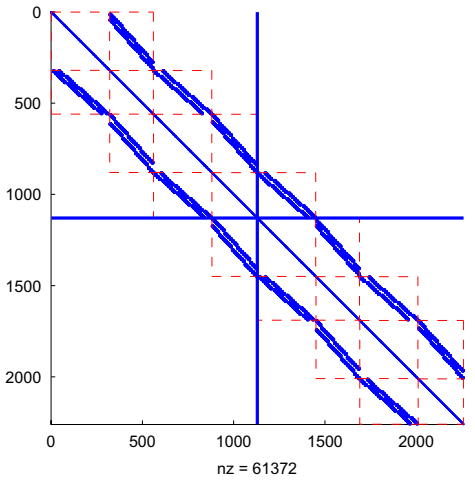

Fig. 4. Hamiltonian for a 2 nm cross section nanowire with 8 layers. The block structure of the Hamiltonian is illustrated through the use of dotted lines. Non-zero entries based upon the lattice connectivity are shown within the blocks. The separator $d_2 = 4$ is illustrated through the use of the solid vertical and horizontal lines. The bridge matrix b_{d_2} is the off-diagonal block not included in the sub-matrices ϕ_1 and ϕ_2.

direction, and for ease of illustration we will only consider separations that do not break apart unit cells of the lattice. Unit cells are small building blocks that can be used to construct semiconductor lattices. Any connectivity within the lattice can be determined solely by examining a unit cell of the material. If we assume $p_y = 2$ domains along the length, each will correspond to a smaller nanowire consisting of four layers. Fig. 4 shows the decomposition of the Hamiltonian matrix for the example, i.e. the assignment of separator locations $d_1 = 0$, $d_2 = 4$, and $d_3 = N_y = 8$. Assuming no spin–orbit coupling for the $sp^3d^5s^*$ tight-binding model, the number of atomic orbitals in each layer will be: $N_x^{(1)} = 320,\ N_x^{(2)} = 240,\ N_x^{(3)} = 320,$ and $N_x^{(4)} = 250$. In Fig. 4 the bridge matrix C_{d_2} is the off-diagonal block outside of the drawn vertical/horizontal separator. The non-zero locations are shown within the dotted lines corresponding to the block sizes.

If we wish to further decompose the Hamiltonian matrix to reduce both the memory and computational complexities, the effect of splitting each section of the nanowire along its cross section must be examined. However, the dense boundary condition blocks Σ_1 and Σ_2, as seen in (1), prevent an efficient decomposition of the matrix K. Therefore, we initially consider the problem of finding the diagonal entries of the matrix $K = (EI - H)$ and in Section 3.4 describe how the diagonal entries are modified by the inclusion of the boundary condition terms. In order to simplify notation we will refer to solving a sub-problem as finding the diagonal entries for the inverse of a block tridiagonal sub-matrix.

3.1. Cross-sectional decomposition

In order to illustrate the cross-sectional decomposition process we continue with the example from Fig. 4. The sub-matrix ϕ_1 corresponds to the first four layers of the nanowire (the blocks above the drawn separator line). Fig. 5 shows a cross-sectional decomposition of this lattice into $p_x = 4$ smaller domains. Each sub-matrix of ϕ_1 is related to an area encompassed by dotted lines. As can be seen in Fig. 5, the sub-matrices $\phi_1^1, \ldots, \phi_1^4$ will have smaller block sizes (fewer atoms per layer), but the same number of layers as ϕ_1. The bonds that bridge two neighboring domains represents the connectivity that is lost through the decomposition. We now introduce notation which will formalize the cross-sectional decomposition process.

Assigning distribution factors $f_l^k,\ k = 1, \ldots, p_x$, to each layer of size, $N_x^{(l)}$, allows for further decomposition. The elements (atomic orbitals or grid points) that make up the sub-matrices ϕ_i can be divided to form p_x smaller sub-matrices. This separation produces coefficient matrices $\{\phi_i^1, \phi_i^2, \ldots, \phi_i^{p_x}\}$, considering the Hamiltonian interactions for each subset of elements. The following conditions are put in place to ease the illustration for the domain reconstruction of the device characteristics:

1. Defining f_l^j to be the size of block l for a sub-matrix $1 \leqslant j \leqslant p_x$, the distribution factors must satisfy the property: $f_l^j \geqslant f_l^k,\ \forall j \leqslant k \leqslant p_x$. This results in a non-increasing allotment across all of the sub-matrices, i.e., the size of ϕ_i^j is at least as large as $\phi_i^k,\ \forall j \leqslant k \leqslant p_x$.
2. The connectivity lost by decomposing along the cross section of the device must result in a "consistent" sparsity pattern. Specifically, the number of bonds cut and the spatial locations of the cut bonds must be the same.

These conditions are in place to simplify both the presentation of our method and the implementation. The criteria allow for a consistent indexing to be used throughout any stage of the cross-sectional reconstruction. The decomposition for each of the $p_y = 2$ domains from Fig. 4 into $p_x = 4$ smaller domains is illustrated in Fig. 6. Here, the cross-sectional lattice decomposition from Fig. 5 (corresponding to ϕ_1) is repeated with respect to ϕ_2. We can conclude that the cross-sectional distribu-

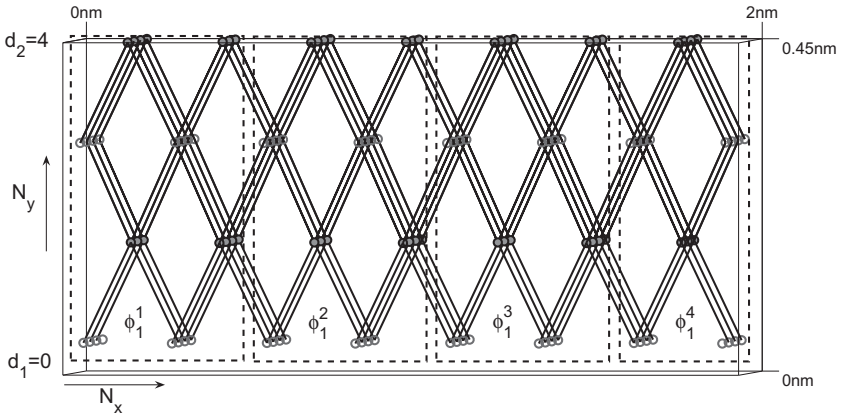

Fig. 5. An illustration for the cross-sectional decomposition of the Si [100] lattice into 4 domains. The lattice corresponds to the ϕ_1 sub-matrix from Fig. 4. The atoms encompassed in each dotted box correspond to the 4 sub-matrices $\phi_1^1, \ldots, \phi_1^4$.

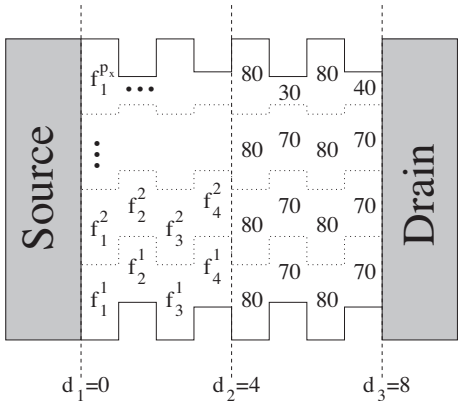

Fig. 6. Separator locations for 2-D decomposition along both the length and cross section of the device. The example from Fig. 4 is illustrated. Assuming there are $p = p_x p_y = (4)(2) = 8$ domains, the factors for each layer l are denoted as f_l^k, for $1 \leqslant k \leqslant p_x = 4$. As the Si nanowire lattice is periodic every four layers, the numeric values in the last four layers match the symbols provided in the first four layers.

tion factors, $f_1^j = 80,\ f_2^j = 70,\ f_3^j = 80,$ and $f_4^j = 70,\ 1 \leqslant j \leqslant 3$, with $f_1^4 = 80,\ f_2^4 = 30,\ f_3^4 = 80,$ and $f_4^4 = 40$, satisfy the criteria provided above.

The illustrations from Figs. 5 and 6 are useful to motivate our approach, but it is necessary to examine the effect of the cross-sectional decomposition on the Hamiltonian matrix. Fig. 7 shows the reordering of the sub-matrix ϕ_1 into a sequential arrangement of the sub-matrices: $\{\phi_1^1, \phi_1^2, \ldots, \phi_1^{p_x}\}$. That is, all atomic orbitals in ϕ_1^1 are listed first and correspond to the first diagonal block in Fig. 7. Similarly, the second diagonal block corresponds to ϕ_1^2 and the connectivity between these two sub-matrices is the first off-diagonal block. Thus, the general form of the ϕ_i sub-matrix under this cross-sectional domain permutation would be of the form:

$$P_x(\phi_i) = \begin{pmatrix} \phi_i^1 & C_i^1 & & \\ (C_i^1)^T & \phi_i^2 & C_i^2 & \\ & & \ddots & \\ & & \left(C_i^{p_x-1}\right)^T & \phi_i^{p_x} \end{pmatrix}. \tag{2}$$

In order to describe the method by which the density of states for sub-matrix ϕ_i can be constructed from each of the smaller sub-matrices $\{\phi_i^1, \phi_i^2, \ldots, \phi_i^{p_x}\}$, we begin by examining the connectivity matrices C_i^j. From the second decomposition criteria posed above we know that each $C_i^j,\ j < p_x$, will have a sparsity pattern that is consistent, i.e. for any non-zero entry in C_i^j there should be a corresponding non-zero entry in $C_i^k,\ j, k < p_x$. Therefore, if $\mathscr{I}_i^j$ and $\mathscr{J}_i^j$ are defined to be the ordered sets of non-zero rows and columns respectively for each matrix C_i^j, we can conclude that the sparsity patterns of $C_i^j\left(\mathscr{I}_i^j, \mathscr{J}_i^j\right)$ and $C_i^k\left(\mathscr{I}_i^k, \mathscr{J}_i^k\right)$ match $\forall j, k < p_x$. In order to construct the diagonal entries of $[\phi_1]^{-1}$ we use a radix-2 combining process (similar to the procedure shown in Fig. 3), the first stage of which involves joining the inverses of ϕ_1^1 and ϕ_1^2. The next stage involves joining the inverses of ϕ_1^3 and ϕ_1^4, and the final stage joining the combined inverses of $\phi_1^{1\sim2}$ and $\phi_1^{3\sim4}$.

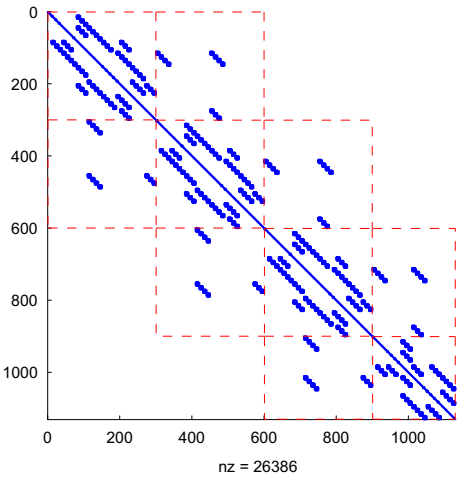

Fig. 7. Permutation $P_x(\cdot)$ of sub-matrix ϕ_1 based upon $p_x = 4$ domains along the cross section. The sub-matrices $\phi_1^1, \ldots, \phi_1^4$ are the diagonal blocks and the connectivity between the domains is represented in the off-diagonal blocks. The dotted lines are used to separate the four domains. The general matrix representation is provided in (2).

By examining how to combine the inverses of ϕ_1^1 and ϕ_1^2 through the use of the Sherman–Morrison–Woodbury formula [17], we motivate the general recursive combining process:

$$\begin{pmatrix} \phi_1^1 & C_1^1 \\ \left(C_1^1\right)^T & \phi_1^2 \end{pmatrix} = \begin{pmatrix} \phi_1^1 & \\ & \phi_1^2 \end{pmatrix} + \begin{pmatrix} C_1^1(:,\mathscr{I}_1^1) & 0 \\ 0 & \left[C_1^1(\mathscr{I}_1^1,:)\right]^T \end{pmatrix} \begin{pmatrix} 0 & I(\mathscr{J}_1^1) \\ I(\mathscr{I}_1^1) & 0 \end{pmatrix},$$

where $I(\cdot)$ is a matrix with ones such that $C_i^j = C_i^j\left(:,\mathscr{I}_i^j\right) I\left(\mathscr{I}_i^j\right)$. The inverse of this combined matrix can now be written as:

$$\begin{aligned} \begin{pmatrix} \phi_1^1 & C_1^1 \\ \left(C_1^1\right)^T & \phi_1^2 \end{pmatrix}^{-1} = \begin{pmatrix} \phi_1^1 & \\ & \phi_1^2 \end{pmatrix}^{-1} - \begin{pmatrix} \left[\phi_1^1\right]^{-1}(:,\mathscr{I}_1^1)\widetilde{C}_1^1 & 0 \\ 0 & \left[\phi_1^2\right]^{-1}(:,\mathscr{I}_1^1)\left[\widetilde{C}_1^1\right]^T \end{pmatrix} \\ * \begin{pmatrix} I & \left[\phi_1^2\right]^{-1}(\mathscr{I}_1^1,\mathscr{I}_1^1)\left[\widetilde{C}_1^1\right]^T \\ \left[\phi_1^1\right]^{-1}(\mathscr{I}_1^1,\mathscr{I}_1^1)\widetilde{C}_1^1 & I \end{pmatrix}^{-1} * \begin{pmatrix} 0 & \left[\phi_1^2\right]^{-1}(\mathscr{I}_1^1,:) \\ \left[\phi_1^1\right]^{-1}(\mathscr{I}_1^1,:) & 0 \end{pmatrix}, \end{aligned} \tag{3}$$

where $\widetilde{C}_1^1 = C_1^1(\mathscr{I}_1^1,\mathscr{J}_1^1)$ is the subset of rows and columns from the connectivity matrix that contain non-zero entries. That is, the matrix $\widetilde{C}_1^1$ will be $W_r \times W_c$, where the values $W_r = |\mathscr{I}_1^1|$ and $W_c = |\mathscr{J}_1^1|$ are used to represent the total amount of Hamiltonian connectivity between entries from neighboring sub-matrices ϕ_1^1 and ϕ_1^2. We will refer to the matrix:

$$J = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix} = \begin{pmatrix} I & \left[\phi_1^2\right]^{-1}(\mathscr{I}_1^1,\mathscr{J}_1^1)\left[\widetilde{C}_1^1\right]^T \\ \left[\phi_1^1\right]^{-1}(\mathscr{I}_1^1,\mathscr{I}_1^1)\widetilde{C}_1^1 & I \end{pmatrix}^{-1}$$

as the adjustment matrix associated with the combining stage. It is important to note that the adjustments to the diagonal entries from each sub-matrix inverse, as seen in (3), are completely determined by the connectivity matrix C_1^1 and certain rows and columns, $\mathscr{I}_1^1$ and $\mathscr{J}_1^1$, from each sub-matrix inverse. These properties allow for the formulation of a general recursive framework that can be used to map the sub-problem solutions for $\{\phi_1^1, \phi_1^2, \ldots, \phi_1^{p_x}\}$ to form the solution for ϕ_1. The problem of determining certain columns from the inverse of the sub-matrices is addressed in Appendix A.2. Determining the representation for each $\left[\phi_i^j\right]^{-1}$ and utilizing the representation to find the necessary columns from the inverse requires $O\left(\left(\frac{N_x}{p_x}\right)^3\left(\frac{N_y}{p_y}\right) + \left(\frac{N_y}{p_y}\right)^2\left(\frac{N_x}{p_x}\right)^2\right)$.

3.2. Cross-sectional reconstruction

Given that the sparsity patterns for all connectivity-based matrices $\widetilde{C}_i^j$, $j < p_x$ are assumed to be consistent for a given nanowire domain i (i.e. property (2) from Section 3.1), we only need to consider one row index set and one column index set for all sub-matrices ϕ_i^j. Specifically, the sets:

$$\mathscr{I} = \mathscr{I}_1^1 \quad \text{and} \quad \mathscr{J} = \mathscr{J}_1^1 \tag{4}$$

can be used to formulate all cross-sectional combining steps associated with any ϕ_i. The values $W_r = |\mathscr{I}|$ and $W_c = |\mathscr{J}|$ are used to represent the total amount of Hamiltonian connectivity between elements from neighboring sub-matrices ϕ_i^j and ϕ_i^{j+1}. It should be clear from (3) that the updates to the diagonal entries based upon any combining step are dependent on $\left[\phi_i^j\right]^{-1}(\mathscr{I},:)$ and $\left[\phi_i^j\right]^{-1}(\mathscr{J},:)$. Thus, associated with each domain we can define eight matrix mapping terms $\{X_1, X_2, \ldots, X_8\}$ that allow for the cross-sectional reconstruction of the diagonal entries for $[\phi_i]^{-1}$. The first four matrix maps ensure we can produce the necessary columns from any combined sub-matrices:

$$\begin{aligned}\left[\phi_i^j\right]^{-1}(\mathscr{I},:) &\leftarrow X_1\left[\phi_i^j\right]^{-1}(\mathscr{I},:) + X_2\left[\phi_i^j\right]^{-1}(\mathscr{J},:),\\ \left[\phi_i^j\right]^{-1}(\mathscr{J},:) &\leftarrow X_3\left[\phi_i^j\right]^{-1}(\mathscr{I},:) + X_4\left[\phi_i^j\right]^{-1}(\mathscr{J},:),\end{aligned} \tag{5}$$

where initially

$$X_k = \begin{cases} 0 & k = 2,3,\\ I & k = 1,4.\end{cases}$$

Note, the columns from each sub-matrix inverse are initialized correctly before any combining has taken place. The remaining four matrix maps can be used to produce the diagonal entries after any combining step:

$$\begin{aligned}\left[\phi_i^j\right]^{-1} \leftarrow \left[\phi_i^j\right]^{-1} &- \left[\phi_i^j\right]^{-1}(:,\mathscr{I})X_5\left[\phi_i^j\right]^{-1}(\mathscr{I},:) + \left[\phi_i^j\right]^{-1}(:,\mathscr{I})X_6\left[\phi_i^j\right]^{-1}(\mathscr{J},:) + \left[\phi_i^j\right]^{-1}(:,\mathscr{I})X_7\left[\phi_i^j\right]^{-1}(\mathscr{I},:)\\ &+ \left[\phi_i^j\right]^{-1}(:,\mathscr{J})X_8\left[\phi_i^j\right]^{-1}(\mathscr{J},:),\end{aligned} \tag{6}$$

where initially $X_k = 0$, $k = 5, 6, 7, 8$. As this is a recursive combining process, the matrix maps associated with a domain must be modified in such a way as to allow expression (6) to remain valid after any combining step. Each of the combing steps involves both a range of sub-matrices and a separator location (with a corresponding connectivity or bridge matrix). We define the start, bridge point, and stop positions (`st`, `bp`, and `sp` respectively) to describe the relevant domain indices. Using this notation, the updates to the matrix maps for each combining step can be described through a series of recursions.

If we define $[\texttt{bridge point} + \texttt{l}]^x = \left[\phi_i^{\texttt{bp}+1}\right]^{-1}(\mathscr{J},\mathscr{J})$, $[\texttt{bridge point}]^x = \left[\phi_i^{\texttt{bp}}\right]^{-1}(\mathscr{I},\mathscr{I})$, and $\widetilde{C} = C_i^{\texttt{bp}}$, the adjustment matrix for each combining step is defined as follows:

$$J = \begin{pmatrix} J_{11} & J_{12}\\ J_{21} & J_{22}\end{pmatrix} = \begin{pmatrix} I & [\texttt{bridge point} + \texttt{l}]^x\left[\widetilde{C}_i^{\texttt{bp}}\right]^T\\ [\texttt{bridge point}]^x\widetilde{C}_i^{\texttt{bp}} & I\end{pmatrix}^{-1}.$$

Next, if we define $[\texttt{start}]^x = \left[\phi_i^{\texttt{st}}\right]^{-1}(\mathscr{I},\mathscr{J})$ and $[\texttt{stop}]^x = \left[\phi_i^{\texttt{sp}}\right]^{-1}(\mathscr{J},\mathscr{I})$, the updates to the matrix maps for the upper sub-problems ($\texttt{st} \leqslant j \leqslant \texttt{bp}$) are summarized below:

$$\begin{aligned}&X_5 + [X_1]^T(\widetilde{C}J_{12})X_1 \to X_5,\\ &X_6 + [X_1]^T(\widetilde{C}J_{12})X_2 \to X_6,\\ &X_7 + [X_2]^T(\widetilde{C}J_{12})X_1 \to X_7,\\ &X_8 + [X_2]^T(\widetilde{C}J_{12})X_2 \to X_8,\\ &X_3 - (\widetilde{C}J_{12}[\texttt{start}]^x)^TX_1 \to X_3,\\ &X_4 - (\widetilde{C}J_{12}[\texttt{start}]^x)^TX_2 \to X_4,\\ &-(\widetilde{C}J_{11}[\texttt{stop}]^x)^TX_1 \to X_1,\\ &-(\widetilde{C}J_{11}[\texttt{stop}]^x)^TX_2 \to X_2.\end{aligned} \tag{7}$$

Those associated with the lower sub-problems ($\texttt{bp} + 1 \leqslant j \leqslant \texttt{sp}$) are exactly a reflection of the updates to upper sub-problems:

$$\begin{aligned}&X_5 + [X_3]^T(\widetilde{C}^TJ_{21})X_3 \to X_5,\\ &X_6 + [X_3]^T(\widetilde{C}^TJ_{21})X_4 \to X_6,\\ &X_7 + [X_4]^T(\widetilde{C}^TJ_{21})X_3 \to X_7,\\ &X_8 + [X_4]^T(\widetilde{C}^TJ_{21})X_4 \to X_8,\\ &X_1 - (\widetilde{C}^TJ_{21}[\texttt{stop}]^x)^TX_3 \to X_1,\\ &X_2 - (\widetilde{C}^TJ_{21}[\texttt{stop}]^x)^TX_4 \to X_2,\\ &-(\widetilde{C}^TJ_{22}[\texttt{start}]^x)^TX_3 \to X_3,\\ &-(\widetilde{C}^TJ_{22}[\texttt{start}]^x)^TX_4 \to X_4.\end{aligned} \tag{8}$$

Determining the adjustment matrix J and updating the matrices $\{X_1, \ldots, X_8\}$ requires $O\left(\left(\frac{N_y}{p_y}\right)^3 \log_2 p_x\right)$.

3.3. Mapping into transport direction

The recursions in (7) and (8) are used to compute the density of states for the domains associated with $\{\phi_1, \phi_2, \ldots, \phi_{p_y}\}$. The cross-sectional reconstructions are performed independently, i.e., without taking into consideration the related bridge matrices $\{C_{d_2}, C_{d_3}, \ldots, C_{d_{p_y}}\}$. As was seen in [2], accounting for separations along the length of the device requires the computation of the first block row and last block column (the boundary) from each $[\phi_i]^{-1}$. This is due to the fact that the connectivity between ϕ_i and ϕ_{i+1}, as demonstrated in Fig. 4, only exits between the boundary layers. Generating a distributed representation for the boundary of $[\phi_i]^{-1}$ can be accomplished by adding additional mapping terms into the cross-sectional combining process described in Section 3.2.

In order to motivate the type of mapping scheme required to produce the boundary of each $[\phi_i]^{-1}$, consider the example shown in Fig. 8. Here, we have removed everything from $[\phi_1]^{-1}$ except the (1,1) block and transformed the matrix based upon the cross-sectional permutation described in Section 3.1. As can be seen from this figure, entries from the first block row of $[\phi_1]^{-1}$ are not limited to the first block row of each sub-matrix inverse $\left[\phi_1^j\right]^{-1}$ in the permuted space, i.e. areas enclosed by dotted lines. In fact, the entries become distributed across every combination of the domain extent for the sub-matrices ϕ_1^j. Therefore, we must associate with each sub-matrix ϕ_i^j boundary mapping terms: $\left\{Y_1^k, Y_2^k, Y_3^k, Y_4^k\right\}$, where $1 \leqslant k \leqslant p_x$. It is important to note that only the mapping terms associated with a given combining step need to be updated. That is, if the domain associated with ϕ_1^j falls within the combining step $\mathtt{st} \leqslant j \leqslant \mathtt{sp}$, then all the maps $\left\{Y_1^k, \ldots, Y_4^k\right\}, \mathtt{st} \leqslant k \leqslant \mathtt{sp}$ must be updated. This process is illustrated by further analyzing the example from Section 3.1.

For each domain (sub-matrices ϕ_1^j shown in Fig. 7) the update to the kth set of boundary mapping terms will be dependent on the position of both j and k with respect to the bridge point $\mathtt{bp}$. Initially, the mapping terms $Y_1^j = Y_4^j = I$ and the remaining are set to 0. If we define $[\mathtt{top\ row}]_k^T = \left[\phi_i^k\right]^{-1}(1 : f_1^k, \mathscr{I}), [\mathtt{bottom\ row}]_k^T = \left[\phi_i^k\right]^{-1}((\mathrm{end} - f_{\mathrm{end}}^k + 1) : \mathrm{end}, \mathscr{I})$, and $\widetilde{C} = C_i^{\mathtt{bp}}$, for domains/boundary maps satisfying $\mathtt{st} \leqslant j,k \leqslant \mathtt{bp}$, we have the following update scheme:

$$\begin{aligned}
&Y_1^k - ([\mathtt{top\ row}]_k^T \widetilde{C} J_{12}) X_1 \to Y_1^k,\\
&Y_2^k - ([\mathtt{top\ row}]_k^T \widetilde{C} J_{12}) X_2 \to Y_2^k,\\
&Y_3^k - ([\mathtt{bottom\ row}]_k^T \widetilde{C} J_{12}) X_1 \to Y_3^k,\\
&Y_4^k - ([\mathtt{bottom\ row}]_k^T \widetilde{C} J_{12}) X_2 \to Y_4^k.
\end{aligned} \tag{9}$$

If we define $[\mathtt{top\ col}]_k^T = \left[\phi_i^k\right]^{-1}(1 : f_1^k, \mathscr{I}), [\mathtt{bottom\ col}]_k^T = \left[\phi_i^k\right]^{-1}((\mathrm{end} - f_{\mathrm{end}}^k + 1) : \mathrm{end}, \mathscr{I})$, for domains/boundary maps satisfying $\mathtt{st} \leqslant j \leqslant \mathtt{bp}$ and $\mathtt{bp} + 1 \leqslant k \leqslant \mathtt{sp}$, we have the following update scheme:

$$\begin{aligned}
&Y_1^k - \left([\mathtt{top\ col}]_k^T \widetilde{C}^T J_{22}\right) X_1 \to Y_1^k,\\
&Y_2^k - \left([\mathtt{top\ col}]_k^T \widetilde{C}^T J_{22}\right) X_2 \to Y_2^k,\\
&Y_3^k - \left([\mathtt{bottom\ col}]_k^T \widetilde{C}^T J_{22}\right) X_1 \to Y_3^k,\\
&Y_4^k - \left([\mathtt{bottom\ col}]_k^T \widetilde{C}^T J_{22}\right) X_2 \to Y_4^k.
\end{aligned} \tag{10}$$

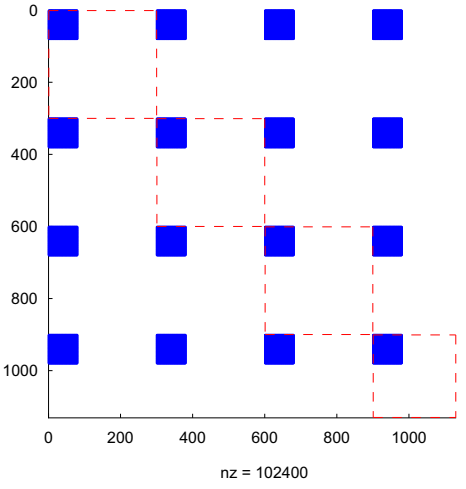

Fig. 8. The (1,1) block of $[\phi_1]^{-1}$ under the permutation $P_x(\cdot)$. The permutation has redistributed the 320×320 block into 16 smaller blocks of size 80×80. Similar to Fig. 7, the dotted lines show the domain extent for each $\left[\phi_1^j\right]^{-1}$, where $1 \leqslant j \leqslant 4$.

Similar recursions are used for the lower sub-problems $j > bp$. Note that the adjustment matrix for the combing step is the same as that seen in (7) and (8). When considering the $\log_2 p_x$ combining levels, each processor will be responsible for updating boundary maps dependent on the range of the combining steps. Cumulatively there will be $\sum_{k=1}^{\log_2 p_x} 2^k$ boundary maps updates. Therefore, when mapping into the transport direction the computational complexity required is $O\left(\left(\frac{N_y}{p_y}\right)^2 N_x\right)$.

After the cross-sectional combining process has been completed, the boundary mapping terms can be used to generate the boundary of $[\phi_i]^{-1}$. Specifically, the maps are used to generate two matrices $\left[\mathtt{Row}_i^j\right]$ and $\left[\mathtt{Col}_i^j\right]$ that contain entries from the first block row and last block column of $[\phi_i]^{-1}$, respectively. The mapping operation used to produce information for the first block row:

$$\left[\mathtt{Row}_i^j\right] = \begin{pmatrix} Y_1^1\left[\phi_i^j\right]^{-1}(\mathscr{I},:) + Y_2^1\left[\phi_i^j\right]^{-1}(\mathscr{J},:) \\ \vdots \\ Y_1^{p_x}\left[\phi_i^j\right]^{-1}(\mathscr{I},:) + Y_2^{p_x}\left[\phi_i^j\right]^{-1}(\mathscr{J},:) \end{pmatrix} \tag{11}$$

and last block column:

$$\left[\mathtt{Col}_i^j\right] = \begin{pmatrix} Y_3^1\left[\phi_i^j\right]^{-1}(\mathscr{I},:) + Y_4^1\left[\phi_i^j\right]^{-1}(\mathscr{J},:) \\ \vdots \\ Y_3^{p_x}\left[\phi_i^j\right]^{-1}(\mathscr{I},:) + Y_4^{p_x}\left[\phi_i^j\right]^{-1}(\mathscr{J},:) \end{pmatrix}^T. \tag{12}$$

Each of the matrices in (11) and (12) have sizes in the order of $\left(N_x \times \frac{N_x N_y}{p_x p_y}\right)$. Forming an entry in either of the matrices requires $2(N_y/p_y)$ multiplications, resulting in a total computational complexity $O\left(\left(\frac{N_y}{p_y}\right)^2\left(\frac{N_x^2}{p_x}\right)\right)$. Fig. 9 shows the distribution of the entries from $\left[\mathtt{Row}_i^1\right]$ across the first block row of $[\phi_i]^{-1}$. Since in this case we are examining the first cross-sectional sub-matrix ($j = 1$), the columns of $\left[\mathtt{Row}_i^1\right]$ correspond to the first several columns from each block entry of $[\mathtt{Row}_i]$, which is the unpermuted first block row $[\phi_i]^{-1}$ (see Figs. 9 and 10).

Therefore, when we consider the combining process detailed in [2] all information that is necessary to combine along the length of the device has been generated and is stored in a distributed fashion. For example, if we consider the transport direction matrix maps needed to update the boundary of $[\phi_i]^{-1}$ based upon the transport direction bridge matrices $\{C_{d_2}, C_{d_3}, \ldots, C_{d_{p_y}}\}$ we have:

$$\begin{aligned} [\phi_i]^{-1}\left(1 : N_x^{(d_i+1)}, :\right) &\leftarrow M_1[\mathtt{Row}_i] + M_2[\mathtt{Col}_i]^T, \\ [\phi_i]^{-1}\left(\mathrm{end} - N_x^{(d_{i+1})} + 1 : \mathrm{end}, :\right) &\leftarrow M_3[\mathtt{Row}_i] + M_4[\mathtt{Col}_i]^T, \end{aligned} \tag{13}$$

where initially

$$M_k = \begin{cases} 0 & k = 2, 3, \\ I & k = 1, 4. \end{cases}$$

Similarly, the updates to the matrix maps for the diagonal entries due to combining in the transport direction are:

$$[\phi_i]^{-1} \leftarrow [\phi_i]^{-1} - [\mathtt{Row}_i]^T M_5[\mathtt{Row}_i] - [\mathtt{Row}_i]^T M_6[\mathtt{Col}_i]^T - [\mathtt{Col}_i] M_7[\mathtt{Row}_i] - [\mathtt{Col}_i] M_8[\mathtt{Col}_i]^T \tag{14}$$

where initially $M_k = 0$, $k = 5, 6, 7, 8$. Thus, by following the combining procedure described in [2] we can modify the diagonal entries to account for the separators along the length of the device. During each of the $\log_2 p_y$ combining stages forming the

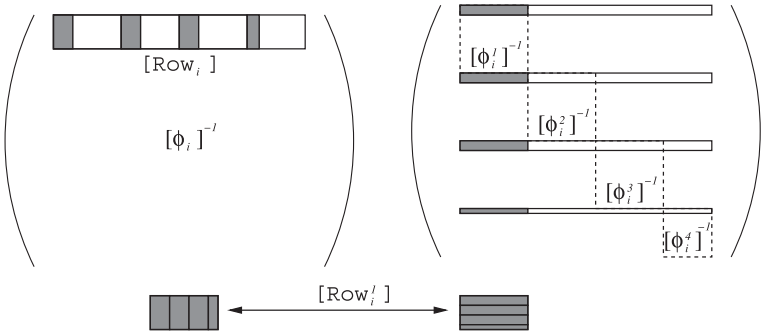

Fig. 9. Distribution of first block row for $[\phi_i]^{-1}$ based upon the cross-sectional permutation $P_x(\cdot)$. The entries for $\left[\mathtt{Row}_i^1\right]$ are represented by the dark region. The original ordering for ϕ_i is shown on the left and the permuted space on the right. The domain extent for each $\left[\phi_i^j\right]^{-1}$ are enclosed in dotted lines.

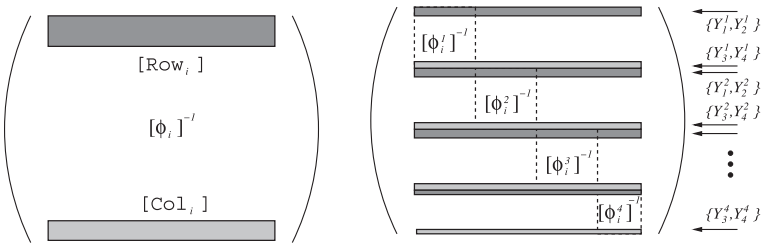

Fig. 10. Usage of boundary matrix maps to generate first block row and last block column of $[\phi_i]^{-1}$ based upon the cross-sectional permutation $P_x(\cdot)$. The original ordering for ϕ_i is shown on the left and the permuted space on the right. The domain extent for each $\left[\phi_i^j\right]^{-1}$ are enclosed in dotted lines.

adjustment matrix requires $O\left(N_x^3\right)$ and the distributed dense matrix–matrix multiplications require $O\left(N_x^3/p_x\right)$ The complexity needed to update the mapping terms $\{M_1,\ldots,M_8\}$ is $O\left(N_x^3\log_2 p_y\right)$. The remaining task is to account for the presence of boundary conditions when determining the diagonal entries of G^r.

3.4. Accounting for boundary conditions

In order to efficiently decompose the matrix, the dense boundary condition terms Σ_1 and Σ_2, which correspond to the left and right contacts, respectively, were initially excluded from the analysis. Specifically, in Sections 3.2 and 3.3 a framework was constructed to allow for the calculation of the diagonal entries for a block tridiagonal matrix $K = (EI - H)$ through the use of cross-sectional and transport direction mapping terms. We illustrate how the effect of the boundary conditions can be incorporated into the transport direction mapping terms $\{M_1,M_2,\ldots,M_8\}$. The inverse of the matrix $K = (EI - H)$ and G^r are related by the following relationship:

$$G^r = K^{-1} - \left(\begin{bmatrix}K^{-1}\left(:,1:N_x^{(1)}\right)\Sigma_1\end{bmatrix} \quad \begin{bmatrix}K^{-1}\left(:,\text{end}-N_x^{(N_y)}+1:\text{end}\right)\Sigma_2\end{bmatrix}\right) * \begin{pmatrix} I+[\mathtt{UL}]\Sigma_1 & [\mathtt{UR}]\Sigma_2 \\ [\mathtt{LL}]\Sigma_1 & I+[\mathtt{LR}]\Sigma_2 \end{pmatrix}^{-1} * \begin{pmatrix} \begin{bmatrix}K^{-1}\left(1:N_x^{(1)},:\right)\end{bmatrix} \\ \begin{bmatrix}K^{-1}\left(\text{end}-N_x^{(N_y)}+1:\text{end},:\right)\end{bmatrix} \end{pmatrix}, \tag{15}$$

where:

$$[\mathtt{UL}] = K^{-1}\left(1:N_x^{(1)},1:N_x^{(1)}\right),$$
$$[\mathtt{LL}] = K^{-1}\left(\text{end}-N_x^{(N_y)}+1:\text{end},1:N_x^{(1)}\right),$$
$$[\mathtt{UR}] = K^{-1}\left(1:N_x^{(1)},\text{end}-N_x^{(N_y)}+1:\text{end}\right),$$
and
$$[\mathtt{LR}] = K^{-1}\left(\text{end}-N_x^{(N_y)}+1:\text{end},\text{end}-N_x^{(N_y)}+1:\text{end}\right).$$

It is important to note that the modifications to K^{-1} due to the inclusion of the boundary condition terms are only a function of the first block row and last block column of K^{-1}. The generation of this information has been described in Section 3.3 through the use of the transport direction mapping terms $\{M_1,M_2,\ldots,M_8\}$. If we consider the adjustment matrix for (15) to be J we can describe the effect of the boundary conditions on the first block row of K^{-1}:

$$\begin{aligned} G^r\left(1:N_x^{(1)},:\right) &= K^{-1}\left(1:N_x^{(1)},:\right) - ([\mathtt{UL}]\Sigma_1[\mathtt{UR}]\Sigma_2)\begin{pmatrix}J_{11} & J_{12}\\ J_{21} & J_{22}\end{pmatrix}\begin{pmatrix} \begin{bmatrix}K^{-1}\left(1:N_x^{(1)},:\right)\end{bmatrix} \\ \begin{bmatrix}K^{-1}\left(\text{end}-N_x^{(N_y)}+1:\text{end},:\right)\end{bmatrix}\end{pmatrix} \\ &= K^{-1}\left(1:N_x^{(1)},:\right) - ([\mathtt{UL}]\Sigma_1 J_{11} + [\mathtt{UR}]\Sigma_2 J_{21})K^{-1}\left(1:N_x^{(1)},:\right) \\ &\quad - ([\mathtt{UL}]\Sigma_1 J_{12} + [\mathtt{UR}]\Sigma_2 J_{22})K^{-1}\left(\text{end}-N_x^{(N_y)}+1:\text{end},:\right). \end{aligned}$$

Therefore, updates to the maps $\{M_1,M_2\}$ for each domain would be:

$$M_1 - ([\mathtt{UL}]\Sigma_1 J_{11} + [\mathtt{UR}]\Sigma_2 J_{21})M_1 - ([\mathtt{UL}]\Sigma_1 J_{12} + [\mathtt{UR}]\Sigma_2 J_{22})M_3 \to M_1,$$

$$M_2 - ([\mathtt{UL}]\Sigma_1 J_{11} + [\mathtt{UR}]\Sigma_2 J_{21})M_2 - ([\mathtt{UL}]\Sigma_1 J_{12} + [\mathtt{UR}]\Sigma_2 J_{22})M_4 \to M_2.$$

In a similar fashion the update schemes for the transport direction maps $\{M_3,\ldots,M_8\}$ can be generated from (15). Accounting for the boundary conditions in the transport direction maps requires $O\left(N_x^3\right)$ operations. After these updates have been

performed the diagonal entries of G^r can be generated using expression (14). Correcting each of the $\left(\frac{N_x N_y}{p_x p_y}\right)$ diagonal entries associated with ϕ_i^j requires matrix multiplications with complexity $O\left(N_x^2\right)$.

The information needed to compute the transmission:

$$T = \text{Trace}[\Gamma_1 G^r \Gamma_2 G^{r\dagger}], \quad \Gamma_{1,2} = i[\Sigma_{1,2} - \Sigma_{1,2}^\dagger] \tag{16}$$

can be generated using the expressions from (13). Determining the transmission will require dense matrix–matrix multiplications with complexity $O\left(N_x^3\right)$. The complete procedure for determining both the diagonal entries of G^r and the transmission is summarized in the pseudo-code. A detailed analysis of the computational complexity can be found in Appendix B.1.

Pseudo-Code

1. *Start algorithm on $p = p_x p_y$ processors, where each processor is assigned a unique identifier (i,j) for the domain of the device to be governed, $1 \leqslant j \leqslant p_x$ and $1 \leqslant i \leqslant p_y$*
2. *Decompose the device along its length, separating the matrix $K = (EI - H)$ into p_y domains:*
   - *Define separator locations $d_1 < d_2 < \cdots < d_{p_y+1}$ with $d_1 = 0$ and $d_{p_y+1} = N_y$*
   - *Assign an appropriate domain of device to each processor: $\phi_i = \text{tri}(A_{d_{i+1}:d_{i+1}}, C_{d_{i+1}:d_{i+1}-1})$*
   - *Retain all off-diagonal blocks that are not covered by the decomposition: $\{C_{d_2}, C_{d_3}, \ldots, C_{d_{p_y}}\}$*
3. *Decompose the device along both its length and cross section, thus, separating each ϕ_i into p_x sub-matrices:*
   - *Assign distribution factors f_l^j which define the size of block l for a sub-matrix $j \leqslant p_x$, according to the criteria in Section 3.1*
   - *Apply the corresponding cross-sectional permutation matrix P_x for the given domain i*
   - *Assign sub-matrix ϕ_i^j to the appropriate to processor (i,j)*
   - *Retain each connectivity matrix C_i^j for the appropriate domain i*
4. *Determine compact representation for ϕ_i^j :*
   - *Use ratio formulation (17) to construct compact representation for $\left[\phi_i^j\right]^{-1}$*
   - *Assign appropriate row and column connectivity sets $\mathscr{I}$ and $\mathscr{J}$ based upon (4)*
   - *Determine $\left[\phi_i^j\right]^{-1}(:,\mathscr{I})$ and $\left[\phi_i^j\right]^{-1}(:,\mathscr{J})$ using procedure from Section A.2*
5. *Define a series of combining steps across all sub-matrices $1 \leqslant j \leqslant p_x$; for a given domain i, each combining stage is defined by locations* [st,bp,sp], *where for every* $\texttt{st} \leqslant j \leqslant \texttt{sp}$:
   - *Each processor responsible will generate the matrices:* $[\texttt{start}]^x$, $[\texttt{bridge point}]^x$, $[\texttt{bridge point + 1}]^x$, *and* $[\texttt{stop}]^x$
   - *Calculate* $[\texttt{top row}]_j$, $[\texttt{bottom row}]_j$, $[\texttt{top col}]_j$ *and* $[\texttt{bottom col}]_j$ as described in Section 3.3
   - *All generated matrices will be transferred between domains* $\texttt{st} \leqslant k \leqslant \texttt{sp}$
   - *Update boundary maps* $\left\{Y_1^k, \ldots, Y_4^k\right\}, \texttt{st} \leqslant k \leqslant \texttt{sp}$ *using the approach outlined in Section 3.3*
   - *Update cross-sectional matrix maps $\{X_1, \ldots, X_8\}$ according to (7) and (8)*
6. *Prepare for transport direction reconstruction:*
   - *Each processor will use cross-sectional maps $\{X_5, \ldots, X_8\}$ to modify diagonal entries according to (6)*
   - *Calculate* $[\texttt{Row}]_i^j$ *and* $[\texttt{Col}]_i^j$ *as described in Section 3.3*
7. *Update transport direction maps $\{M_1, \ldots, M_8\}$ using the procedure described in [2], given distribution scheme of* $[\texttt{Row}]_i$ *and* $[\texttt{Col}]_i$ *outlined in Section 3.3. Requires transferring* $[\texttt{start}]^y$, $[\texttt{bridge point}]^y$, $[\texttt{bridge point + 1}]^y$, *and* $[\texttt{stop}]^y$
8. *Account for boundary conditions by modifying transport direction maps, as outlined in Section 3.4*
9. *Use modified transport direction maps to adjust diagonal entries, according to (14)*
10. *Use modified transport direction maps to calculate transmission, according to (13) and (16).*

4. Results and discussion

The 2-D decomposition algorithm along with the RGF method have been implemented, in C, on clusters consisting of either 2.33 GHz Quad-Core Intel E5410 nodes with 2 GB of memory per core or 2.5 GHz Quad-Core AMD 2380 nodes with 4 GB of memory per core. All nodes were connected through 1GigE connections. The $sp^3d^5s^*$ tight-binding model was used in conjunction with the simulation procedure detailed in [18]. Due to the fact that the computation time for both methods is independent of energy level, only the time for single energy point computations of the density of states and transmission were analyzed.

Given the desire to perform simulations of realistically sized devices, the 2-D decomposition method offers an effective way to divide memory and computation across large numbers of distributed computers for NEGF-based simulation. In order to compare against the current state of the art for realistically sized device simulation we must perform comparisons with wave function-based methods. This is necessary as both the methods of [1,3] are not able to distribute memory on cost effective distributed computing resources. In this work, we compare against the MUMPS [19] algorithm which was shown to be the best performing of the linear solvers for the atomistic simulation of nanowires in [10]. We begin with two scaling analyses for short 10 nm Si [100] nanowires performed on Intel E5410 nodes with 2 GB of mem-

Table 1
Single energy time Si [100] nanowire simulation $4 \times 4 \times 10$ nm^3, in a (p_y, p_x) processor configuration.

p	4	8	16	16	32
Configuration (p_y, p_x)	(4,1)	(8,1)	(16,1)	(8,2)	(8,4)
PDIV-2D (min)	9.82	7.28	6.48	4.92	3.87
MUMPS (min)	0.78	0.89	0.89	–	–

Table 2
Single energy time Si [100] nanowire simulation $5.2 \times 5.2 \times 10$ nm^3, in a (p_y, p_x) processor configuration.

p	8	16	16	32
Configuration (p_y, p_x)	(8,1)	(16,1)	(8,2)	(8,4)
PDIV-2D (min)	36.35	31.36	21.76	16.61
MUMPS (min)	2.45	1.89	–	–

Table 3
Si [100] nanowire simulation for length of 80 nm.

	$4 \times 4 \times 80$ nm^3		$5 \times 5 \times 80$ nm^3	
	p (p_y, p_x)	Time (min)	p (p_y, p_x)	Time (min)
PDIV-2D	64 (32,2)	8.65	64 (32,2)	36.41
MUMPS	32	3.81	64	10.16

ory per core. We consider both 4 nm and 5.2 nm cross sections, shown in Tables 1 and 2, respectively. The reduced complexity of solving for only a few RHS (in this case less than 20) of a linear system is demonstrated by the superior performance of the MUMPS algorithm. The number of RHS is associated with the number of wave functions needed for the analysis. This is dependent on the model (i.e. scattering) and intrinsic properties of the device, see [18]. However, if the length of the device is extended to a more realistic size of 80 nm we have the results shown in Table 3. In this case, we see that the MUMPS algorithm was less than a factor of 4 $\times$ faster than the 2-D decomposition approach for the 5.2 nm cross section nanowire. Thus, for large-scale simulation problems we have demonstrated that NEGF-based algorithms can offer comparable computational performance when compared to wave function-based methods. In addition, the NEGF-based approach presented in this work is directly applicable for more sophisticated analyses that involve electron (phonon) scattering.

It is important to recall that the MUMPS algorithm is only a linear solver and not a NEGF-based approach. The distinction can be clearly seen if we stipulate on the efficiency of using a linear solver such as MUMPS to find the density of states through the diagonal entries of the inverse. That is, we consider the use of the MUMPS algorithm to compute the entire matrix G^r. Given the current state of applicable wave function-based algorithms, the inclusion of scattering would require all wave functions (size of matrix) to be determined. For the 5.2 nm cross section nanowire, this translates into solving for over 1 million RHS. By recording the time necessary to solve for a single RHS, over one second for this example, the time necessary would be over two orders of magnitude slower than the PDIV-2D approach for NEGF-based applications. Thus, we have included Tables 1–3 solely as a benchmark for simulation algorithms in the absence of scattering. There should be a clear distinction between the mathematical problems of finding the diagonal entries from the inverse and solving a linear system of equations.

In order to demonstrate the versatility of the proposed approach we offer two other types of computational analyses (performed on 2.5 GHz Quad-Core AMD 2380 nodes with 4 GB of memory per core). The first is a study of the performance for the algorithm when the length of a 3×3 nm^2 square cross section silicon nanowire is increased from 10 nm to 20 nm and 40 nm. Table 4 shows the times needed to compute the density of states and transmission using p processors with p_x cross-sectional separations and $p_y = p/p_x$ separations along the length of the nanowire. The PDIV-1D approach from [2] is represented by configurations with $p_x = 1$. There is no additional algorithmic overhead by considering $p_x = 1$ as a subset of the PDIV-2D approach. All results for the proposed approach were found to be within 10^{-6} relative error when compared to the RGF algorithm. When considering the modest 3 nm cross section nanowire, having at most 72 atoms (720 atomic orbitals assuming no spin–orbital coupling) in each layer, the benefits of cross-sectional separation can already be observed. For example, when considering the 10 nm length nanowire using $p = 32$ processors the $p_x = 2$ configuration required 3.6 min, while the $p_x = 4$ configuration required under 3.1 min. As the second study will demonstrate, these computational improvements are significantly more pronounced for devices with more realistic cross-sectional size. Although similar trends can be observed for the cases of 20 nm and 40 nm length, it is important to note that the amount of coupling lost through a cross-sectional separation is directly proportional to the length of the nanowire domain considered. Thus, it is important to

Table 4
Single energy run time (min) across 2-D separation configurations, for fixed total number of processors. PDIV-1D is represented under configurations with $p_x = 1$. "Cross" is used for decomposition schemes that cannot utilize the desired number of processors directly.

p	p_x	p_y	$3 \times 3 \times 10$ nm^3 $N_x \approx 720$ $N_y = 76$	$3 \times 3 \times 20$ nm^3 $N_x \approx 720$ $N_y = 152$	$3 \times 3 \times 40$ nm^3 $N_x \approx 720$ $N_y = 304$	$4 \times 4 \times 10$ nm^3 $N_x \approx 1130$ $N_y = 76$	$5.2 \times 5.2 \times 10$ nm^3 $N_x \approx 2000$ $N_y = 76$
1	RGF		6.87	13.52	27.64	40.20	236.42
2	1	2	9.81	17.16	32.93	53.13	260.96
4	1	4	7.63	12.52	21.81	37.61	185.46
8	1	8	6.32	8.28	13.36	32.29	149.72
	2	4	4.16	14.12	66.72	22.03	101.66
16	1	16	5.95	6.89	8.96	29.16	151.23
	2	8	3.32	4.70	14.73	17.52	85.19
	4	4	4.46	18.64	102.95	16.70	81.10
32	2	16	3.60	3.86	5.35	17.33	87.31
	4	8	3.07	4.83	18.32	13.32	69.83
	8	4	Cross	Cross	Cross	14.29	68.71
64	4	16	3.15	3.53	5.31	14.96	77.96
	8	8	Cross	Cross	Cross	11.77	60.85
128	8	16	Cross	Cross	Cross	14.93	76.17

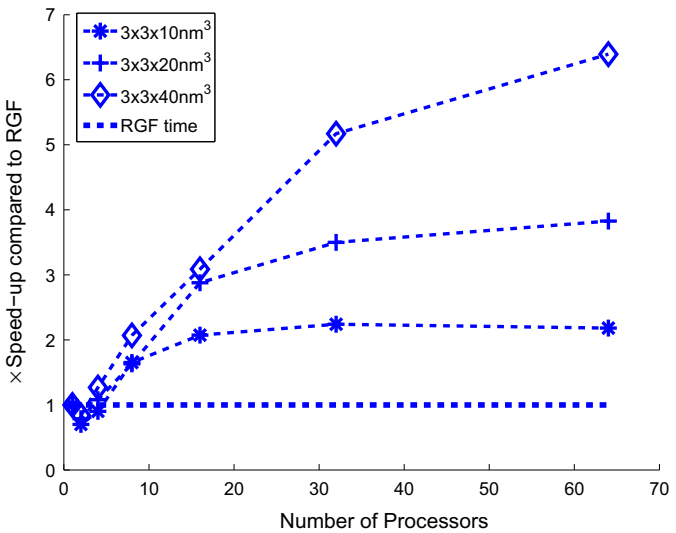

Fig. 11. Speedup comparison for single energy computation time of 3 nm cross section nanowire.

consider the "aspect" ratio between the size of the cross section and the nanowire length assigned to each processor in order to determine the most efficient separation scheme.

The second computational study considers a fixed length of 10 nm and varying the cross-sectional size from 3 nm to 4 nm and 5.2 nm. The cases have at most 720, 1130 and 2000 atomic orbitals respectively in each layer. It can be seen from Table 4 that as the cross-sectional size increases, it becomes increasingly advantageous to use more cross-sectional separations. When examining the minimum computation time for all $p = 32$ configurations, the best separation scheme was $p_x = 4$ for the 4 nm cross section device, and $p_x = 8$ for the 5.2 nm. The 10 nm-length examples described above also demonstrate another important contribution of the method. Given that the 10 nm nanowires have a total of 76 layers, they cannot be divided into more than 16 domains along the length, i.e., the PDIV-1D method cannot be used on 32 processors to simulate this device (4-layer periodic silicon [100] nanowire). However, with the use of cross-sectional separations, several times more computing resources can be allocated toward the simulation. The resulting benefits to memory

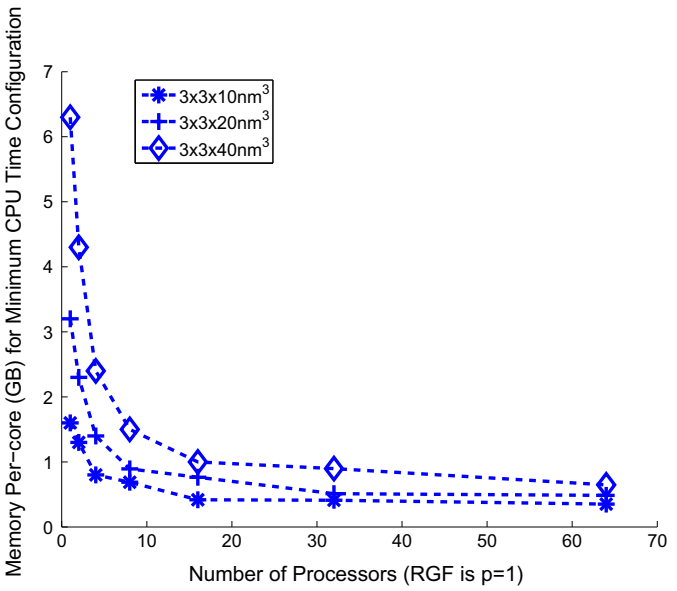

Fig. 12. Memory requirements of each core for single energy computation time of 3 nm cross section nanowire. Required memory is based upon fastest processor configuration. The RGF requirements are shown as $p = 1$.

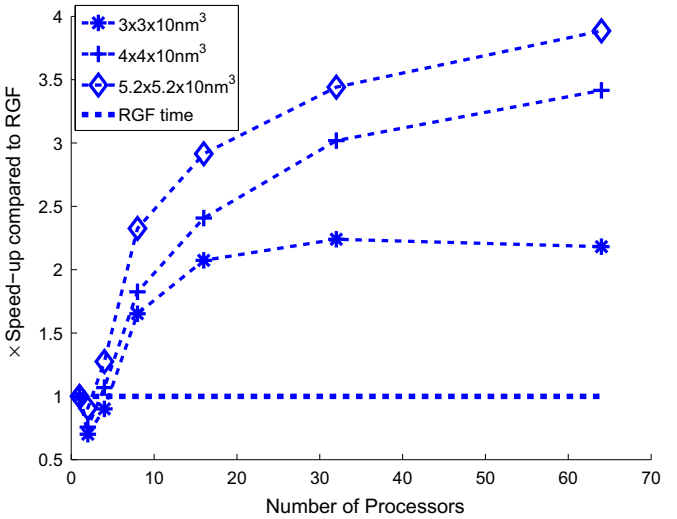

Fig. 13. Speedup comparison for single energy computation time of 10 nm length nanowire.

requirements can be seen in Table 6. Here, the 5.2 nm cross section nanowire requires 3.7 GB per core for the PDIV-1D approach with $p = p_y = 16$. This amount is reduced to 1.5 GB per core using PDIV-2D decomposition approaches with $p_x > 1$.

A summary of the minimum run time processor configurations are provided in Figs. 11 and 13, respectively. The overall memory reduction for these configurations (when compared to the single processor RGF algorithm) are shown in Figs. 12 and 14, respectively. In addition to the single energy run times provided in Table 4, the average communication time and memory requirements (per core) are summarized in Tables 5 and 6, respectively. In Appendix B.1 we have provided details regarding the computational complexity of our approach. This information can be used to predict an ideal processor configuration given the nature of the problem being solved. It is important to note that for a given device hundreds to thousands of similar problems must be solved during simulation. In a practical implementation it would be beneficial to attempt several processor configurations that are similar to the estimate shown in Appendix B.1. As communication costs will vary with the distributed computing resources used, this strategy will ensure an ideal configuration without incurring significant computational overhead.

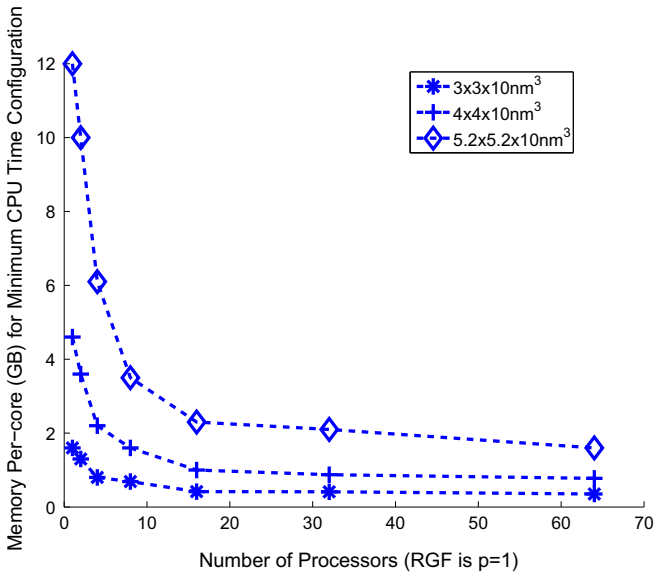

Fig. 14. Memory requirements of each core for single energy computation time of 10 nm length nanowire. Required memory is based upon fastest processor configuration. The RGF requirements are shown as $p = 1$.

Table 5
Single energy per-core communication time (min) across 2-D separation configurations, for fixed total number of processors. PDIV-1D is represented under configurations with $p_x = 1$. "Cross" is used for decomposition schemes that cannot utilize the desired number of processors directly.

p	p_x	p_y	$3 \times 3 \times 10$ nm^3 $N_x \approx 720$ $N_y = 76$	$3 \times 3 \times 20$ nm^3 $N_x \approx 720$ $N_y = 152$	$3 \times 3 \times 40$ nm^3 $N_x \approx 720$ $N_y = 304$	$4 \times 4 \times 10$ nm^3 $N_x \approx 1130$ $N_y = 76$	$5.2 \times 5.2 \times 10$ nm^3 $N_x \approx 2000$ $N_y = 76$
1	RGF		0.0	0.0	0.0	0.0	0.0
2	1	2	0.42	0.08	0.08	2.19	10.70
4	1	4	0.18	0.36	0.22	0.75	4.22
8	1	8	0.54	0.21	0.21	0.99	4.15
	2	4	0.11	0.07	0.33	0.45	2.42
16	1	16	0.41	0.24	0.27	1.64	8.28
	2	8	0.13	0.16	0.14	0.47	2.36
	4	4	0.26	0.41	1.49	0.83	4.61
32	2	16	0.21	0.23	0.26	0.59	3.18
	4	8	0.24	0.32	0.43	0.61	4.22
	8	4	Cross	Cross	Cross	1.03	5.69
64	4	16	0.30	0.33	0.40	0.90	5.52
	8	8	Cross	Cross	Cross	0.67	4.93
128	8	16	Cross	Cross	Cross	1.16	7.11

5. Conclusion

In this work we offer a framework that facilitates the simulation of large cross-sectional devices. The number of atomic orbitals in each layer of the nanowire contributes the dominant factor in both the computational and memory complexities for state of the art NEGF-based algorithms. In order for the NEGF-based approaches to be applicable for realistically sized devices, new techniques must be introduced that can support analyses involving hundreds of atoms (thousands of atomic orbitals) in each layer. The 2-D decomposition technique ameliorates this computational challenge by allowing for additional computing resources to be allocated toward the simulation of these devices. We use these computational resources to stably generate the density of states and transmission, under the NEGF formalism, for the atomistic-based simulation of square 5 nm cross section silicon nanowires consisting of over one million atomic orbitals.

Table 6
Single energy per-core memory requirement (GB) across 2-D separation configurations, for fixed total number of processors. PDIV-1D is represented under configurations with $p_x = 1$. "Cross" is used for decomposition schemes that cannot utilize the desired number of processors directly.

p	p_x	p_y	$3 \times 3 \times 10$ nm^3 $N_x \approx 720$ $N_y = 76$	$3 \times 3 \times 20$ nm^3 $N_x \approx 720$ $N_y = 152$	$3 \times 3 \times 40$ nm^3 $N_x \approx 720$ $N_y = 304$	$4 \times 4 \times 10$ nm^3 $N_x \approx 1130$ $N_y = 76$	$5.2 \times 5.2 \times 10$ nm^3 $N_x \approx 2000$ $N_y = 76$
1	RGF		1.6	3.2	6.3	4.6	12.0
2	1	2	1.3	2.3	4.3	3.6	10.0
4	1	4	0.8	1.4	2.4	2.2	6.1
8	1	8	0.6	0.9	1.5	1.6	4.4
	2	4	0.7	1.4	3.7	1.6	3.5
16	1	16	0.5	0.7	1.0	1.0	3.7
	2	8	0.4	0.8	1.5	1.0	2.3
	4	4	0.5	1.1	3.2	1.0	2.3
32	2	16	0.4	0.5	0.9	0.9	2.1
	4	8	0.4	0.5	1.5	0.9	2.0
	8	4	Cross	Cross	Cross	0.9	2.1
64	4	16	0.4	0.5	0.7	0.8	1.7
	8	8	Cross	Cross	Cross	0.8	1.6
128	8	16	Cross	Cross	Cross	0.7	1.5

In addition, the ability to map from the cross-sectional space to the transport space is applicable for determining the entire compact representation of G^r, not just limited to the diagonal entries. Therefore, the procedures seen in [1] would facilitate the computation of all information necessary for construction of the less-than and greater-than Green's functions, $G^<$ and $G^>$, respectively.

Acknowledgement

This work was partially supported by NSF grant CCF-1065318.

Appendix A. Block tridiagonal matrices

The inverse of a block tridiagonal matrix (1) can be completely captured through the diagonal blocks of the inverse along with sequences of ratio matrices. In this work, we introduce a decomposition method that separates the device (and the related matrix) along two dimensions. We then show how desired information from the inverse can be efficiently computed through a parallel reconstruction method. The derivation of our reconstruction scheme relies on detailed knowledge of mathematical operations related to block tridiagonal matrices.

A.1. Compact representation

As discussed in Section 2 the Hamiltonian of the device contains all interaction information for an atomistic represented nanowire. The Hamiltonian matrix and any sub-matrix ϕ_i^j created through the 2-D decomposition of the device (presented in Section 3.1) are of block tridiagonal structure. For simplicity of illustration, we will assume the block size N_x and number of blocks N_y for K can be evenly divided by p_x and p_y, respectively. Based upon the permutation $P_x(\phi_i)$ we can write $\phi_i^j = \mathrm{tri}(\widehat{A}_{1:(N_y/p_y)}, \widehat{C}_{1:(N_y/p_y)-1})$ where each diagonal and off-diagonal block is $(N_x/p_x) \times (N_x/p_x)$. The inverse of each ϕ_i^j can be described compactly through the use of sequences $\{R_{1:(N_y/p_y)-1}\}, \{S_{1:(N_y/p_y)-1}\},$ and $\{D_{1:(N_y/p_y)}\}$:

$$\left[\phi_i^j\right]^{-1} = \begin{pmatrix} D_1 & D_1S_1 & D_1S_1S_2 & \cdots & D_1 \prod_{k=1}^{(N_y/p_y)-1} S_k \\ R_1D_1 & D_2 & D_2S_2 & \cdots & D_2 \prod_{k=2}^{(N_y/p_y)-1} S_k \\ R_2R_1D_1 & R_2D_2 & D_3 & \cdots & D_3 \prod_{k=3}^{(N_y/p_y)-1} S_k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \left(\prod_{k=(N_y/p_y)-1}^{1} R_k\right) D_1 & \left(\prod_{k=(N_y/p_y)-1}^{2} R_k\right) D_2 & \left(\prod_{k=(N_y/p_y)-1}^{3} R_k\right) D_3 & \cdots & D_{(N_y/p_y)} \end{pmatrix},$$

where $(N_y/p_y) = d_{i+1} - d_i$ is the number of blocks in the ith domain of the device along its length. The sequences $\{R\}$, $\{S\}$, and $\{D\}$ can be determined as follows:

$$
\begin{aligned}
&R_1 = \widehat{A}_1^{-1}\widehat{C}_1,\\
&R_i = \left(\widehat{A}_i - \widehat{C}_{i-1}^T R_{i-1}\right)^{-1}\widehat{C}_i, \quad i = 2,\ldots,(N_y/p_y)-1,\\
&S_{(N_y/p_y)-1} = \widehat{C}_{(N_y/p_y)-1}\widehat{A}^{-1}_{(N_y/p_y)},\\
&S_i = \widehat{C}_i\left(\widehat{A}_{i+1} - S_{i+1}\widehat{C}_{i+1}^T\right)^{-1}, \quad i = (N_y/p_y)-2,\ldots,1,\\
&D_1 = \left(\widehat{A}_1 - S_1\widehat{C}_1^T\right)^{-1},\\
&D_{i+1} = \left(\widehat{A}_{i+1} - S_{i+1}\widehat{C}_{i+1}^T\right)^{-1}\left(I + \widehat{C}_i^T D_i S_i\right), \quad i = 1,\ldots,(N_y/p_y)-2,\\
&D_{(N_y/p_y)} = \widehat{A}^{-1}_{(N_y/p_y)}\left(I + \widehat{C}^T_{(N_y/p_y)-1} D_{(N_y/p_y)-1} S_{(N_y/p_y)-1}\right).
\end{aligned}
\tag{17}
$$

Reconstructing the diagonal entries for the inverse of the Hamiltonian involves operations with $\left[\phi_i^j\right]^{-1}$. Specifically, the diagonal entries are needed as well as certain columns (based upon the connectivity cut through the decomposition of the device, i.e. the sets of indices $\mathscr{I}$ and $\mathscr{J}$ from Section 3.1). The diagonal entries are merely a subset of the diagonal blocks themselves, so we will turn our attention to efficiently calculating columns from the inverse using the compact representation.

A.2. Calculation of columns from block tridiagonal inverse using compact representation

For each unique index n in the sets $\mathscr{I}$ and $\mathscr{J}$ we have to determine the corresponding column from the inverse of the submatrix ϕ_i^j. The index n can be written in terms of the block number (layer number) l and the orbital (grid point) number o within the layer. For simplicity of illustration, we will assume the block size N_x and number of blocks N_y for K can be evenly divided by p_x and p_y respectively. Thus, the expression $n = (l-1)(N_x/p_x) + o$ describes the relationship between the variables n, l, and o. The explicit expression for a column from the inverse can be written as follows:

$$
\left[\phi_i^j\right]^{-1}(:,n) = \begin{pmatrix}
\left(D_1 \prod_{k=1}^{l-2} S_k\right) S_{l-1}(:,o)\\
\left(D_2 \prod_{k=2}^{l-2} S_k\right) S_{l-1}(:,o)\\
\vdots\\
D_l(:,o)\\
(D_l(o,:)S_l)^T\\
\vdots\\
\left(D_l(o,:)\prod_{k=l}^{(N_y/p_y)-1} S_k\right)^T
\end{pmatrix}.
\tag{18}
$$

Therefore, we can construct two matrices $\left[\phi_i^j\right]^{-1}(:,\mathscr{I})$ and $\left[\phi_i^j\right]^{-1}(:,\mathscr{J})$ for use in the 2-D reconstruction process.

Appendix B. Computational complexity

A detailed list of the computational and memory requirements along with the information that needs to be transferred through inter-processor communication is provided in Appendix B.1. The relationship between the computational requirements and the asymptotic computational scaling is summarized in Appendix B.2.

B.1. Computation, memory, and communication complexities

In order to simplify the analysis of the computational complexity we will make certain assumptions:

- The block size is uniform for each layer and equal to N_x.
- The number of cross-sectional domains p_x evenly divides the block size N_x.
- The number of domains along the length p_y evenly divides the number of blocks N_y.
- The connectivity pattern cut from each domain due to cross-sectional separations can be reduced to a $W_r \times W_c$ matrix (described in Section 3.1). If we examine Fig. 7, this would be the information contained in C_i^j if we were to eliminate columns and rows that are entirely zero. The size of W_r and W_c for a regular 2-D grid would be exactly (N_y/p_y).
- The distribution factors (described in Section 3.1) for processor k and layer l are equal $f_l^k = N_x/p_x$.

- The order of matrix inversion is assumed to be cubic and for simplicity of illustration we will assume that dense matrix–matrix multiplication is of the same order. We will ignore the smaller computational cost of sparse matrix multiplication.

We will directly follow the flow of the pseudo-code to provide detail into the dominate computational operations. The numbering relates to steps from the pseudo-code:

4. Inversion of the sub-matrix associated with each domain of the device:
   The generator sequences are found through dense matrix–matrix multiplications along with dense matrix inversions, see (17). As the Hamiltonian is symmetric the computation required to form the $\left(2\frac{N_y}{p_y}-1\right)\left(\frac{N_x}{p_x}\right)^2$ entries is:

$$\left(3\frac{N_y}{p_y}-1\right)\left(\frac{N_x}{p_x}\right)^3.$$

The matrices needed for cross-sectional reconstruction, i.e. $\left[\phi_i^j\right]^{-1}(:,\mathscr{I})$ and $\left[\phi_i^j\right]^{-1}(:,\mathscr{J})$, are of sizes $\left(\frac{N_xN_y}{p_xp_y}\right)\times W_r$ and $\left(\frac{N_xN_y}{p_xp_y}\right)\times W_c$, respectively. As demonstrated in Appendix A.2 there are $\left(\frac{N_y}{p_y}-1\right)$ dense matrix–vector multiplications needed for each column from the sets $\mathscr{I}$ and $\mathscr{J}$. Therefore, the computational cost to form the matrices is:

$$(W_r+W_c)\left(\frac{N_y}{p_y}-1\right)\left(\frac{N_x}{p_x}\right)^2.$$

5. Collect combining information based upon separators along the cross-sectional dimension of the device:
   There are a total of $\log_2 p_x$ combining levels needed based upon the cross-sectional decomposition of the device. The computation and communication costs required for each stage are provided below.
   Use the cross-sectional matrix maps to generate the following cross-sectional combining terms: $\texttt{[start]}^x$, $\texttt{[bridge point]}^x$, $\texttt{[bridge point + 1]}^x$, and $\texttt{[stop]}^x$

$$3W_r^2W_c+3W_rW_c^2+W_r^3+W_c^3.$$

The dense matrices listed above are of sizes $W_r\times W_c$, $W_r\times W_r$, $W_c\times W_c$, and $W_c\times W_r$, respectively. Communication is required to transfer these $(W_r+W_c)^2$ values across at most p_x processors.
Use the cross-sectional matrix maps to generate the terms associated with reconstructing along the length of the device (transport direction):
$[\texttt{top row}]_j$, $[\texttt{bottom row}]_j$, $[\texttt{top col}]_j$ and $[\texttt{bottom col}]_j$

$$\left(W_r^2+W_c^2\right)\left(\frac{2N_x}{p_x}\right).$$

The dense matrices listed above are of sizes $\left(\frac{N_x}{p_x}\right)\times W_r$, $\left(\frac{N_x}{p_x}\right)\times W_r$, $\left(\frac{N_x}{p_x}\right)\times W_c$, and $\left(\frac{N_x}{p_x}\right)\times W_c$, respectively. Communication is required to transfer these $(W_r+W_c)\left(\frac{2N_x}{p_x}\right)$ values across at most p_x processors.
Calculating the $(W_r+W_c)\times(W_r+W_c)$ adjustment matrix requires:

$$(W_r+W_c)^3+W_r^2W_c+W_rW_c^2.$$

The associated matrix inversion can be performed in parallel for a potential complexity reduction.
The p_x versions of the boundary maps $\left\{Y_1^k,\ldots,Y_4^k\right\}$ each consist of $(W_r+W_c)\left(\frac{2N_x}{p_x}\right)$ entries. The updates for the boundary maps are separated into two categories based upon the location with respect to the bridge point:
If $k\leqslant \texttt{bp}$ then:

$$\left(2W_r^2+W_rW_c\right)\left(\frac{2N_x}{p_x}\right).$$

If $k>\texttt{bp}$ then:

$$\left(2W_c^2+W_rW_c\right)\left(\frac{2N_x}{p_x}\right).$$

It is clear that during the $\log_2 p_x$ combining levels, each processor will be responsible for updating maps $\left\{Y_1^k,\ldots,Y_4^k\right\}$, based upon the range of the combining step: $\texttt{st}\leqslant k\leqslant \texttt{sp}$. Cumulatively there will be $\sum_{k=1}^{\log_2 p_x}2^k$ or $O(p_x)$ map updates of this type. The updates to the cross-sectional matrix maps $\{X_1,\ldots,X_8\}$ requires the following computation cost:

$$4W_r^2W_c+9W_rW_c^2+3W_c^3.$$

Each processor updates only their own version of these mapping terms, consisting of $2(W_r + W_c)^2$ entries. There will be $O(\log_2 p_x)$ map updates of this type.

6. Use the cross-sectional and boundary matrix maps to prepare for reconstruction along the length of the device:
   Each of the $\left(\frac{N_x N_y}{p_x p_y}\right)$ diagonal entries for $\left[\phi_i^j\right]^{-1}$ must be updated based upon cross-sectional separations, using the relationship from (6) the computation required is:
   $$W_r^2 + 2W_r + 2W_r W_c + 2W_c + W_c^2.$$
   The first block row and last block column of each $[\phi_i]^{-1}$ must be reconstructed using $\left\{Y_1^k, \ldots, Y_4^k\right\}$ and the relationships shown in (11) and (12). There are $2(W_r + W_c)$ multiplications needed to form each entry in $\left[\texttt{Row}\ {}_i^j\right]$ and $\left[\texttt{Col}\ {}_i^j\right]$. The total computational complexity will be:
   $$\left(\frac{N_x N_y}{p_x p_y}\right)(2N_x)(W_r + W_c).$$
   As the first block row and last block column have been evenly divided among the $p = p_x p_y$ processors, the memory requirements are for $2N_x^2 N_y/p$ entries.
7. Collect combining information based upon separators along the length of the device:
   There are a total of $\log_2 p_y$ combining levels needed based upon the decomposition along the length of the device. Details pertaining to this stage of algorithm can be found in [2]. The computation required for each stage is provided below.
   Use the transport direction matrix maps to generate the following combining terms: $\texttt{[start]}^y$, $\texttt{[bridge point]}^y$, $\texttt{[bridge point + 1]}^y$, and $\texttt{[stop]}^y$ all of which require $2N_x^3/p_x$ multiplications. Communication is required to transfer each of the four $N_x \times N_x$ matrices across at most $p = p_x p_y$ processors.
   The $2N_x \times 2N_x$ adjustment matrix needs to be computed for updates to the matrix maps associated with the transport direction. The associated matrix inversion can be performed in parallel or in serial with complexity:
   $$5(N_x)^3.$$
   Modifying the eight $N_x \times N_x$ matrix maps $\{M_1, \ldots, M_8\}$ in order to update the boundary (13) and diagonal entries (14) requires:
   $$16\frac{N_x^3}{p_x}.$$
8. Modify the matrix maps $\{M_1, \ldots, M_8\}$ to account for the boundary conditions based upon the relationship shown in (15):
   $$14\frac{N_x^3}{p_x} + 20N_x^3 + 8N_x^3.$$
9. Each of the $\left(\frac{N_x N_y}{p_x p_y}\right)$ diagonal entries associated with $\left[\phi_i^j\right]^{-1}$ must be updated based upon separations along transport direction. As can be seen from (14), each diagonal entry will require:
   $$4\left(N_x^2 + N_x\right).$$
10. The matrix maps $\{M_1, \ldots, M_8\}$ can be used to calculate the transmission through the relationship shown in (13):
    $$6\frac{N_x^3}{p_x} + 15N_x^3.$$

B.2. Asymptotic computational scaling

The 2-D domain decomposition strategy presented in this work involves a cross-sectional combining/density of states reconstruction, a directional transition, and a transport direction combining/density of states reconstruction. The computation related to these five operations constitutes the overall asymptotic scaling for our approach. The numbering matches the algorithm flow from Appendix B.1.

5. $\log_2 p_x$ combining steps required to account for cross-sectional separations:
   $$O\left(\left(\frac{N_y}{p_y}\right)^3 \log_2 p_x\right) \leftarrow \log_2 p_x \cdot \begin{cases} 3W_r^2 W_c + 3W_r W_c^2 + W_r^3 + W_c^3, \\ (W_r + W_c)^3 + W_r^2 W_c + W_r W_c^2, \\ 4W_r^2 W_c + 9W_r W_c^2 + 3W_c^3. \end{cases}$$
6. Updates to the density of states for cross-sectional separations:

$$O\left(\left(\frac{N_y}{p_y}\right)^3\left(\frac{N_x}{p_x}\right)\right) \leftarrow \left(\frac{N_xN_y}{p_xp_y}\right)\cdot\{\, W_r^2 + 2W_r + 2W_rW_c + 2W_c + W_c^2.$$

Transitioning from cross-sectional to transport direction reconstruction:

$$O\left(\left(\frac{N_y}{p_y}\right)^2\left(\frac{N_x^2}{p_x}\right)\right) \leftarrow \left(\frac{N_xN_y}{p_xp_y}\right)(N_x)\cdot\{\, 2(W_r + W_c).$$

7. $\log_2 p_y$ combining steps required to account for transport direction separations:

$$O\left(N_x^3\log_2 p_y\right) \leftarrow \log_2 p_y \cdot \{\, 5(N_x)^3.$$

9. Updates to the density of states for transport direction separations:

$$O\left(N_x^3\left(\frac{N_y}{p}\right)\right) \leftarrow \left(\frac{N_xN_y}{p_xp_y}\right)\cdot\left\{\, 4\left(N_x^2 + N_x\right).$$

References

[1] A. Svizhenko, M.P. Anantram, T.R. Govindan, B. Biegel, R. Venugopal, Two-dimensional quantum mechanical modeling of nanotransistors, Journal of Applied Physics 91 (4) (2002) 2343–2354.
[2] S. Cauley, J. Jain, C.-K. Koh, V. Balakrishnan, A scalable distributed method for quantum-scale device simulation, Journal of Applied Physics 101 (2007) 123715.
[3] S. Li, S. Ahmed, G. Klimeck, E. Darve, Computing entries of the inverse of a sparse matrix using the FIND algorithm, Journal of Computational Physics 227 (22) (2008) 9408–9427.
[4] L. Lin, C. Yang, J. Lu, L. Ying, E. Weinan, A fast parallel algorithm for selected inversion of structured sparse matrices with applications to 2D electronic structure calculations, SIAM Journal of Scientific Computing, 33(3):1329–1351, 2011.
[5] K. Takahashi, J. Fagan, M.-S. Chin, Formation of a sparse bus impedance matrix and its application to short circuit study, in: Power Industry Computer Applications Conference Proceedings, 1973, pp. 63–69.
[6] Y.E. Campbell, T.A. Davis, Computing the sparse inverse subset: an inverse multifrontal approach, Technical Report TR-95-021, Computer and Information Sciences Department, University of Florida, 1995.
[7] M. Stadele, R. Tuttle, K. Hess, Tunneling through ultrathin SiO2 gate oxides from microscopic models, Journal of Applied Physics 89 (1) (2001) 348–363.
[8] T.A. Davis, Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method, ACM Transactions on Mathematical Software 30 (2) (2004) 196–199.
[9] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert, X.S. Li, J.-W.H. Liu, A supernodal approach to sparse partial pivoting, SIAM Journal Matrix Analysis and Applications 20 (3) (1999) 720–755.
[10] T.B. Boykin, M. Luisier, G. Klimeck, Multiband transmission calculations for nanowires using an optimized renormalization method, Physical Review B 77 (2008) 165318.
[11] M.V. Fischetti, Master-equation approach to the study of electronic transport in small semiconductor devices, Physical Review B 59 (1999) 4901.
[12] S. Datta, Electronic Transport in Mescoscopic Systems, Cambridge University Press, 1997.
[13] R. Nabben, Decay rates of the inverses of nonsymmetric tridiagonal and band matrices, SIAM Journal on Matrix Analysis and Applications 20 (3) (1999) 820–837.
[14] J. Jain, S. Cauley, H. Li, C.-K. Koh, V. Balakrishnan, Numerically stable algorithms for inversion of block tridiagonal and banded matrices, submitted for consideration, Purdue TR ECE 07-17, 2006.
[15] J.M. Jancu, R. Scholz, F. Baltram, F. Bassani, Empirical spds* tight-binding calculation for cubic semiconductors: general method and material parameters, Physical Review B 57 (1998) 6493.
[16] T.B. Boykin, G. Klimeck, F. Oyafuso, Valence band effective-mass expressions in the sp3d5s* empirical tight-binding model applied to a si and ge parametrization, Physical Review B 69 (2004) 15201.
[17] M.A. Woodbury, Inverting Modified Matrices, Statistical Research Group Memo. Rep., vol. 42, Princeton University, Princeton, NJ, 1950.
[18] M. Luisier, A. Schenk, W. Fichtner, G. Klimeck, Atomistic simulation of nanowires in the sp3d5s* tight-binding formalism: from boundary conditions to strain calculations, Physical Review B 74 (2006) 205323.
[19] P. Amestoy, I.S. Duff, C. Voeme, Task scheduling in an asynchronous distributed multifrontal solver, SIAM Journal on Matrix Analysis and Applications 26 (2) (2005) 544–565.