Accelerating simulation of nanodevices based on 2D materials by hybrid CPU-GPU parallel computing

M. Poljak*, M. Glavan, and S. Kuzmić
Micro and Nano Electronics Lab, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
*e-mail: mirko.poljak@fer.hr

Abstract—We describe our new C-based software that combines atomistic quantum transport with the solution of the 2D Poisson’s equation for nanodevice simulation. A significant acceleration of about ~100× is demonstrated in comparison to our old Matlab code, by using numerical libraries BLAS and LAPACK, shared-memory parallelization with OpenMP, and GPU acceleration based on CUDA libraries. The new code has enabled the analysis of 10 nm-gate length nanotransistors based on silicene nanoribbons, for which we report electronic, transport and device properties.

Keywords—atomistic simulation; quantum transport; NEGF; silicene nanoribbon; CUDA; OpenMP

I. INTRODUCTION

Over the past 20 years, many innovations have been introduced into the complementary metal-oxide-semiconductor (CMOS) technology, such as applying strain and using high-k/metal gate stack, in order to meet device requirements for extremely scaled nodes under 45 nm. For the upcoming post-silicon era, two-dimensional (2D) materials could reshape the industry due to their exceptional electronic and charge transport properties [1]–[3]. Since the discovery of graphene [4], more than 50 single-layer materials have been demonstrated [5], [6]. These include monoelemental crystals (X-enes) such as graphene, silicene and germanene, transition metal dichalcogenides (TMDCs) such as MoS$_2$, etc. In comparison to graphene, silicene [7], [8] could be a better candidate for future electronic applications due to higher compatibility with the existing silicon-based semiconductor industry. Furthermore, its crystal lattice is buckled (see Fig. 1), which should facilitate an easier opening and engineering of the bandgap via the electric field [9], making this material more suitable for digital applications than graphene [10].

The suitability of 2D materials for future nanoelectronic devices, i.e. field-effect transistors (FETs), can be assessed with numerical modeling. Generally, modeling and simulation can support experimental work in the design process since it allows a physical insight into device operation. Moreover, it can reduce the number of trial-and-error iterations, thus reducing the cost of development that is very high at the nanoscale. In order to accurately predict the performance of future nanoscale FETs based on 2D materials, it is necessary to incorporate quantum mechanical effects and atomistic nature of materials into the simulation software. The direct solution of the Schrödinger’s equation with open boundary conditions (OBCs) can be obtained with the non-equilibrium Green’s function (NEGF) formalism [3], [11], [12]. However, atomistic NEGF simulations are computationally very demanding since they involve numerically intensive matrix operations on large matrices. For realistic sizes, the device consists of thousands of atoms, which demands efficient implementation of the NEGF code [13], [14].

In this paper, we present the atomistic NEGF simulator in which heterogeneous CPU-GPU computing and parallel code execution is used to significantly accelerate nanoscale FET simulation. This software’s capabilities are illustrated by studying the electronic and transport properties of silicene nanoribbons (SNRs), and device characteristics of SNR FETs. This analysis provides physical insight into the feasibility of SNR FETs for future digital/logic applications at ~10 nm gate lengths.

II. THEORY

A. Overview

As illustrated in Fig. 2, the device under study consists of an SNR connected to source and drain contacts (S/D) and two gates (G1/G2). The NEGF part of the code gives charge density and current, while the Poisson equation provides the potential profile in the device. The NEGF-Poisson loop iterates until self-consistence is achieved.

B. Atomic Hamiltonian

For each SNR, we construct a tight-binding Hamiltonian ($H$) that accounts for nearest-neighbor interactions:

$$H = \sum_i c_i^\dagger c_i + t \sum_{i,j} c_i^\dagger c_j + H.c.,$$  (1)
where \( \varepsilon_i \) is the on-site energy, \( c_i^\dagger \) (\( c_i \)) is the creation (annihilation) operator, while \( t \) is the hopping parameter \cite{15}. A 5 nm-wide and 10 nm-long SNR consists of 1536 atoms, i.e. \( H \) is an 1536-by-1536 matrix with 2.36 million elements. A 30 nm-long SNR of the same width consists of 4352 atoms and \( H \) has 18.9 million elements. Nevertheless, \( H \) is very sparse (most elements are zero) which can be utilized to accelerate the computations.

C. NEGF Formalism

The Schrödinger's equations with OBCs is solved directly using the NEGF formalism. The NEGF equations are as follows \cite{12}:

\[
(E - H - \Sigma_s - \Sigma_d) G = I \tag{2}
\]

\[
\Gamma = i \left[ \Sigma - \Sigma^* \right] \tag{3}
\]

\[
G^* = G \left[ \Gamma, \Gamma, f_1 + \Gamma, f_2 \right] G^\dagger \tag{4}
\]

\[
T = T \left[ \Gamma, G^\dagger, \Gamma, G^\dagger \right] \tag{5}
\]

\[
I = \frac{2q}{h} \int T \left[ f_1, f_2 \right] dE \tag{6}
\]

where diagonal matrix \( E \) contains carrier (injection) energy, \( G \) is the retarded Green's function and \( \Sigma \) is the retarded self-energy that accounts for OBCs in S/D contacts. Charge density matrix (electron correlation) \( G^\dagger \) depends on \( G \), broadening matrices \( \Gamma \) and Fermi functions \( f \) for the two contacts. \( T \) is the transmission between contacts that also determines the current \( I \) via the Landauer's equation. Position- and energy-dependence in (2)-(6) is omitted for simplicity.

D. 2D Poisson's Equation

Electrostatic potential profile caused by the applied bias and charge distribution is found by solving a two-dimensional Poisson's equation (2D PE):

\[
\nabla [\varepsilon(r) \nabla U(r)] = q \rho(r) \tag{7}
\]

where \( \varepsilon \) is the spatially-dependent dielectric constant, \( U \) is the potential and \( \rho \) is the net charge density distribution (depends on doping profile and carrier densities). The 2D PE is solved using finite differences method, applying Dirichlet BCs at gates and Neumann BCs elsewhere.

III. IMPLEMENTATION

A. Hardware

All simulations were done on a heterogeneous workstation with a 6-core processor (Intel i5-8400, 2.80 GHz to 4 GHz), 16 GB RAM (Kingston DDR4, 2400MHz), and Nvidia Titan Xp GPU (30 streaming multiprocessors (SMs), 3840 cores, 12 GB GDDR5X, 1.4 GHz). In double precision, processor performance is about 26 GFLOPS, while Titan performance is about 380 GFLOPS.

B. Numerical Libraries and Parallel Computing

The simulation software is implemented in C, a general purpose language in contrast to Matlab, and uses several numerical libraries for both CPU and GPU computation. BLAS (from Basic Linear Algebra Subprograms) contains low-level routines for operations such as vector/matrix addition, multiplication, etc. \cite{16}-\cite{18}. LAPACK (from Linear Algebra Package) provides routines for solving linear systems of equations, eigenvalue problems, matrix factorizations, etc. \cite{19}. In this work, we use OpenBLAS, an open-source BLAS/LAPACK implementation with some general optimizations \cite{20}. For GPU programming, we use CUDA (Compute Unified Device Architecture), a parallel computing platform and API model for Nvidia GPUs. Numerical libraries used for GPU acceleration include cuBLAS (a CUDA C implementation of BLAS), cuSPARSE (mostly BLAS-like functions for sparse matrices) and cuSOLVER (LAPACK-like routines for sparse and dense matrices) \cite{21}, \cite{22}. In order to use GPU computational resources, the host application must transfer data to and from the GPU memory space, which could lead to performance decrease due to bandwidth and latency limitations. Regarding parallel execution on the CPU, we either set OpenBLAS to use multi-threading (pthreads) or we parallelize compute-intensive parts of the code using OpenMP (Open Multi-Processing) which is an API for shared-memory multi-processing \cite{23}, \cite{24}. As for the GPU, CUDA libraries internally optimize the parallel execution by setting appropriate grid and thread block sizes across the 30 SMs in Titan Xp GPU. The code we developed in C/CUDA is compared to Matlab implementation. Matlab is a package and domain specific implementation. Matlab and CUDA store complex matrices in different formats, which internally employs numerical libraries such as BLAS, LAPACK, etc. and has a built-in multithreading.

C. Dealing with Compute-Intensive Steps

Contact self-energy matrices \( \Sigma_s \) and \( \Sigma_d \) in (2) are obtained by the Sancho-Rubio method \cite{25}. The procedure involves complex matrix addition and multiplication in double precision with BLAS routines \texttt{zgadd} and \texttt{zgemm}. In CPU-only version of the code, the retarded Green's function \( G \) in (2) is obtained by inversion, using LAPACK routines \texttt{zgetrf} and \texttt{zgetri}. Similarly, in CUDA version, \( G \) is found by solving a linear system \( AX = B \) (with \( B = I \)) using cuSOLVER routines \texttt{Zgetrf} and \texttt{Zgetrs}. Transmission in (5) is calculated using \texttt{Zgemm} in both versions. Finally, in the case of sparse CUDA version, the transmission is found using \texttt{Zcsrmm}. In all of the above steps several format conversions are necessary since BLAS, LAPACK and CUDA store complex matrices in different formats, which adds time to simulation. Acceleration is possible by
setting maximum number of pthreads in OpenBLAS, or by OpenMP parallelization. The NEGF equations must be solved for many energy points (e.g. 500–1000) in the chosen spectrum in order to capture fine changes, e.g. singularities in density of states. Therefore, OpenMP is a natural choice to divide the workload between cores.

The discretized 2D Poisson's equation (7) is in the form $Ax = b$, where $A$ is a real matrix in double precision. It is solved using LAPACK $dgetrf$ and $dgetrs$ routines in CPU-only version, cuSOLVER routines $Dgetrf$ and $Dgetrs$ in dense CUDA version, and cuSOLVER routine $SpDcsrlsvqr$ in the sparse CUDA version. In sparse versions of the code, reformat into compressed sparse row (CSR) is necessary, which requires calls to several subroutines from the cuSPARSE library. Acceleration on GPU, i.e. dividing workload between SMs, is handled internally.

IV. ANALYSIS OF TIME-TO-SOLUTION

A. Accelerating NEGF Transport Computation

NEGF code implementations (Matlab, C and C-CUDA) are tested under the same conditions, i.e. for the simulation of a 5 nm-wide and 10 nm-long SNR with $N = 1536$ atoms, with an energy spectrum with $NE = 504$ energy points. Relatively small structure ($N = 1536$) is chosen due to large $NE$ in order to keep time-to-solution manageable, especially for the Matlab version. Time-to-solution for NEGF transport is reported in Fig. 3 for four different implementations. Dense Matlab (MAT-Dn-6T) and dense C+BLAS+LAPACK (CBL-Dn-OMP6) versions are multithreaded over 6 cores. Matlab code is parallelized using $spmd$ from the Parallel Computing Toolbox, while the C version is parallelized with OpenMP. In both versions, the energy spectrum with $NE$ points is divided between the 6 cores. Parallelization in the dense (CUDA-Dn) and sparse (CUDA-Sp) implementations in Fig. 3 are done internally. Duration is measured by $tic$/$toc$ commands in Matlab, while $clock$/$gettime$ function is used in C versions.

The time-to-solution obtained using MAT-Dn-6T is approx. 24,000 s, while dense C and CUDA versions finish in 255 s and 307 s, respectively. The acceleration compared to Matlab equals 94× and 78× for dense C and CUDA code, respectively. An additional speed-up is offered by the sparse CUDA version, which takes only 206 s to calculate the transmission of an SNR with 1536 atoms over 504 energy points. In this case, the acceleration obtained in comparison to Matlab is 117×. The sparse CUDA implementation takes approx. 410 ms per energy point, out of which roughly two-thirds are taken by reformatting two $N$-by-$N$ matrices into CUDA double complex format. This reformat is needed because part of the host code is based on BLAS/LAPACK. Nevertheless, when the issue of reformatting into CUDA format is resolved in future work, the CUDA-Sp version in Fig. 3 could have a time-to-solution of approx. 70 s, which would present an acceleration of 3.6× and 343× in comparison to dense C and Matlab solutions, respectively.

B. Accelerating Poisson Electrostatics Computation

Different implementations of our 2D PE solver are tested on the same device structure: double gate Si FET with 10 nm-long source and drain regions, and 10 nm channel length; channel thickness is 5 nm; SiO$_2$ thickness is set to 2 nm. The PE is discretized on a rectangular mesh with $\Delta x = \Delta y = 0.1$ nm, leading to 300 points in $y$-direction and 92 points in $x$-direction. Coefficient matrix in (7) is of the size $N = 27,600$, i.e. more than 760 million elements in double precision (8 bytes) so it takes more

![Fig. 3. Comparison of time-to-solution for different code implementations of the NEGF solver. Device under study is a 5 nm-by-10 nm silicene nanoribbon, with 1536 atoms. The reported results are obtained by solving the NEGF equations over 504 energy points.](image)

![Fig. 4. An example output from the 2D Poisson's equation solver. The structure is a double-gate Si FET with overall length of 30 nm and thickness of 9 nm. $N_{S/D} = 10^{20}$ cm$^{-3}$ and inversion carrier concentration is constant (for simplicity) in the channel. $V_G = V_{G2} = 1$ V.](image)
than 5.6 GB of memory. An example of the solution for $V_{G1} = V_{G2} = 1$ V is given in Fig. 4.

Figure 5 shows time-to-solution for the 2D Poisson’s equation for (a) Matlab, (b) C-LAPACK and (c) C-CUDA implementations. For the Matlab implementation, we used sparse matrix storage and mldivide to solve $Ax = b$, which should provide the fastest possible Matlab solution. In addition, the number of internal threads is increased from 1 to 6 (MAT-1T to MAT-6T in Fig. 5a) to assess performance improvement with parallelization. C implementation using LAPACK is also multithreaded (C/LPK-1T to C/LPK-6T in Fig. 5b) with internal parallelization. While C/LPK-6T gives the result in 72 s, CUSLV-Sp version accelerates the simulation by almost 22×.

In order to solve the 2D PE with $N = 27,600$, MAT-1T version runs 306 s, while MAT-6T is done in 167 s. C/LPK-1T version finishes in 264 s, while C/LPK-6T gives the result in 72 s. Therefore, C/LAPACK version offers faster simulation (2.3× for 6T) and better scaling (3.7× acceleration over 6 cores compared to 1.8× in Matlab). The same problem is solved much faster within the CUDA framework, as it takes only 39 s and 7.6 s for the dense and sparse version, respectively. In comparison to the fastest sparse Matlab solution (MAT-6T), CUSLV-Sp version accelerates the simulation by almost 22×.

C. Estimating Time-to-Solution for I-V Characteristics

In order to obtain current-voltage characteristics, bias on the gate ($V_G$) and drain ($V_D$) electrodes has to be swept across different voltages. For each bias combination ($V_G$, $V_D$), roughly ten NEGF-Poisson iterations are needed to achieve self-consistency. Figure 6 reports time-to-solution for one NEGF-Poisson iteration. The sparse C-CUDA implementation offers 113× shorter time than the sparse Matlab version.

We can estimate that it would take approx. 5.93 hrs to calculate a single transfer characteristic even with sparse C-CUDA code (10 bias points, 10 iterations per bias point assumed). Nevertheless, this simulation duration pales in comparison with the Matlab case, where the same simulation would take approx. 27.8 days.

V. Properties of Silicene Nanoribbon FETs

In this section, we analyze several electronic, transport and device properties of SNR-based FETs. The SNR length is $L = 30$ nm (channel and S/D extensions are 10 nm long), while the nanoribbon width ($W$) is varied from ~1 to ~4.4 nm. All SNRs belong to the same 3 family (i.e. number of dimer lines along the width is divisible by three), for the sake of consistency. An additional reason is that the 3m family offers the smallest effective transport mass, which should benefit device performance [26].

Figure 7 reports the transmission and density of states (DOS) for 0.96, 2.10, 3.25 and 4.39 nm-wide SNRs. As shown in Fig. 7a, the transmission is a series of step functions, depending on the number of conducting channels or modes. Narrower SNRs exhibit a larger bandgap and a lower transmission at a given energy than wider nanoribbons. The DOS results in Fig. 7b again illustrate the increasing bandgap when $W$ decreases. In addition, wider SNRs have higher DOS, indicating the possibility of an increased amount of inversion charge at the same voltage when compared to narrower SNRs. The energy gaps from Fig. 7a and b are identical, which means that the band- and transport-gap are the same, i.e. there are
As shown in Fig. 8, the bandgap increases from 0.12, over 0.16 and 0.24, to 0.44 eV when nanoribbon width scales down. Among the analyzed SNRs, it seems that only the narrowest one ($W = 0.96$ nm) could be a feasible transistor channel because other bandgap values are too small. Therefore, device characteristics are analyzed only for the 0.96 nm-wide and 30 nm-long SNR FET. Figure 9 reports transfer characteristics in log and lin scales, obtained for drain voltages of $V_D = 0.05$ and 0.5 V. The gate voltage is swept from $V_G = -0.3$ V to 0.5 V. From the results in Fig. 9 it is clear that the device suffers greatly from short-channel effects. Namely, subthreshold swing extracted for $V_D = 0.5$ V equals $S = 85$ mV/dec while the drain current at $V_G = 0.5$ V is greatly influenced by $V_D$; it increases from 1.2 to 2.1 $\mu$A when $V_D$ changes from 0.05 to 0.5 V, indicating poor saturation behavior. At the same time, off-state current (at $V_G = 0$ V, $V_D = 0.5$ V) is $\sim 15$ $\mu$A/$\mu$m. The only redeeming property is the high current drivability, i.e. on-state current (at $V_G = V_D = 0.5$ V) that equals 2.2 mA/$\mu$m when SNR width is taken into account, which is close to the required level given by ITRS [27]. Other parameters do not meet ITRS requirements, especially the off-state current which is $146\times$ higher than recommended.

In order to clarify SNR FET characteristic, in Fig. 10 we plot energy- and position-resolved (a) local density of states (LDOS), (b) electron density (ND) and (c) electron current density (CD), for the case of $V_G = 0.5$ V and $V_D = 0.05$ V. The LDOS shows the distribution of allowed states within the NEGF formalism, illustrating wave-like injection, reflection and transmission. The ND reports spatial and energy distribution of electrons in the device.
when Fermi functions in S/D are taken into account. Finally, the CD in Fig. 10c shows at which energies the current flow is the strongest. Clearly, even at high bias the majority of electrons tunnel directly through the barrier, in contrast to findings in [28]. The same is observed for \( V_D = 0.5 \) V (not shown in Fig. 10). The reported direct tunneling observed in this device with a 10 nm-long channel is caused by the low effective mass that equals only \(-0.1 m_0 \) (\( m_0 \) is electron rest mass) for the 0.96 nm-wide silicene nanoribbon. Therefore, low-effective-mass SNR FETs do not seem to be a feasible solution for the extremely scaled CMOS technology nodes, at least for digital/logic applications in the future due to strong direct tunneling, both in the off- and on-state.

VI. CONCLUSION

We have presented a C-based simulation software that combines atomistic non-equilibrium Green's function (NEGF) formalism (for transport) with the 2D Poisson equation solver for electrostatics. By using efficient numerical libraries, sparse matrix representations and hybrid CPU-GPU parallel programming in CUDA, we have significantly accelerated the time-to-solution in comparison to our old Matlab code. The obtained accelerations are up to 117× and 22× for the NEGF and Poisson module, respectively. This allows full device simulations, consisting of thousands of atoms, in the matter of hours on a single hybrid workstation. The capabilities of the simulation software are illustrated by analyzing 30 nm-long silicene nanoribbons of various widths in the range from ~1 to 4.4 nm, both in terms of their electronic/transport material properties and FET device characteristics. We have shown that low-effective-mass SNR FETs are not a feasible solution for digital/logic applications in the future due to strong direct source-to-drain tunneling, both in the off- and on-state.

ACKNOWLEDGMENT

M.P. acknowledges support of the NVIDIA Corporation through the donation of the Titan Xp GPU used in this work. The work is also partially sponsored by FER Starting Grant (MPSTART).

REFERENCES