Ab-initio Quantum Transport with the *GW* Approximation, 42,240 Atoms, and Sustained Exascale Performance

Nicolas Vetsch*, Alexander Maeder*, Vincent Maillou*, Anders Winka*, Jiang Cao*, Grzegorz Kwasniewski†, Leonard Deuschle*, Torsten Hoefler†, Alexandros N. Ziogas*, and Mathieu Luisier*
*Integrated Systems Laboratory, ETH Zurich, Switzerland
†Scalable Parallel Computing Laboratory, ETH Zurich, Switzerland

Abstract

Designing nanoscale electronic devices such as the currently manufactured nanoribbon field-effect transistors (NRFETs) requires advanced modeling tools capturing all relevant quantum mechanical effects. State-of-the-art approaches combine the non-equilibrium Green's function (NEGF) formalism and density functional theory (DFT). However, as device dimensions do not exceed a few nanometers anymore, electrons are confined in ultra-small volumes, giving rise to strong electron-electron interactions. To account for these critical effects, DFT+NEGF solvers should be extended with the *GW* approximation, which massively increases their computational intensity. Here, we present the first implementation of the NEGF+*GW* scheme capable of handling NRFET geometries with dimensions comparable to experiments. This package, called *QuaTrEx*, makes use of a novel spatial domain decomposition scheme, can treat devices made of up to 84,480 atoms, scales very well on the Alps and Frontier supercomputers (>80% weak scaling efficiency), and sustains an exascale FP64 performance on 42,240 atoms (1.15 Eflop/s).

1 Justification for ACM Gordon Bell Prize

We report *ab-initio* transistor simulations of unprecedented scale (up to 84,480 atoms) including electron-electron interactions within the self-consistent *GW* approximation. Key achievements are simulations of 42,240 atoms on 37,600 GPUs with (i) ultra-short iteration time (~42 seconds per iteration), (ii) excellent parallel efficiency (82% in weak scaling), and (iii) high computational performance (1.15 Eflop/s — 85% of Rmax — 56% of Rpeak).

2 Performance Attributes

Performance attributes	Our submission
Category of achievement	Peak performance, time-to-solution
Type of method used	Non-linear system of equations
Results reported on basis of	Full application except I/O
Precision reported	Double precision
System scale	Full-scale
Measurements	Timers, FLOP count

3 Overview of the Problem

Density functional theory (DFT) [27] is the method of choice to determine the quantum mechanical properties of metals, oxides, and semiconductors, which build the core of all (opto-)electronic devices, e.g., transistors, light-emitting diodes, or memory cells. Physical quantities such as their band gap, effective mass, defect forming energy, structure stability, or absorption coefficient can be readily obtained with DFT tools, for example, VASP [28], GPAW [37], Quantum Espresso [21], Abinit [22], CP2K [30], or Siesta [2]. Although DFT is a powerful *ab-initio* framework with the capability of predicting several material properties, it remains a ground-state theory and, as such, does not describe excited electronic states very accurately [40]. In particular, it underestimates the band gap of most common semiconductors (Si, Ge, or GaAs) and oxides (SiO_2, HfO_2, or Al_2O_3) by a factor up to 2, band alignments between different compounds are not correctly reproduced, while important effects like Auger processes or excitons are missing [39].

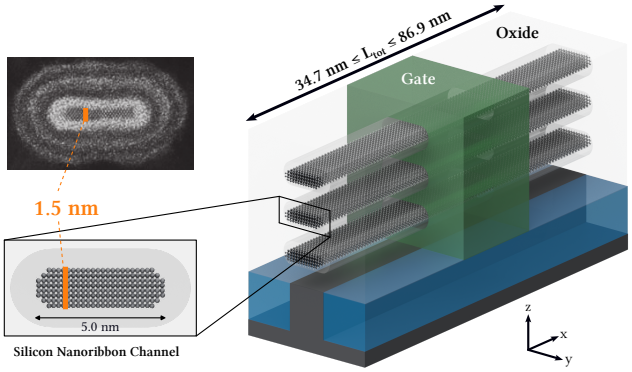


Figure 1: Schematic view of a silicon nanoribbon field-effect transistor where three nanoribbon layers are stacked on top of each other. The central ribbon is simulated in this work. Its cross section is shown on the left, highlighting its shape and size resemblance with a device structure that was recently fabricated by Intel. Reprinted with permission from [1].

Modeling approaches going beyond DFT have, therefore, been developed to overcome these limitations. The *GW* approximation is one of them. It relies on Hedin's equations, which establish a relationship between electrons, whose behavior is described by a variable called G, and the (screened) Coulomb interactions they induce (attraction/repulsion forces) labeled as W [24]. The product of G and W does not only give the *GW* method its name, but also forms a self-energy Σ that corrects DFT results, in particular band gaps [25], thus providing optical absorption spectra [44] or defect levels [12] in excellent agreement with experimental data.

While the extension of DFT with *GW* enables studying the properties of materials with high accuracy, it fails at capturing non-equilibrium phenomena such as the voltage-driven electronic current flowing through atomic systems, i.e., their "current vs. voltage" characteristics. Combining DFT and the *GW* approximation with the non-equilibrium Green's function (NEGF) formalism addresses

Table 1: Summary of the state-of-the-art *GW* performance. The tool name, type of *GW* approximation, inclusion of transport properties, basis set into which the Hamiltonian matrices are expanded, computational scalability, number of atoms achieved on a specific hardware, and simulation time are reported. All symbols are defined in the text and in Table 3.

Tool	Type	Transport	Basis	Scalability	No. Atoms	Performance [Pflop/s]	Hardware	Time* [s]
VASP [33]	G_0W_0	No	PAW	$O(N_A^3)$	54	–	64 CPUs	720
CP2K [49]	G_0W_0	No	GTO	$O(N_A^{2.13})$	1,734	–	14,400 CPUs	18,000
BerkeleyGW [4]	G_0W_0	No	PW	$O(N_A^4)$	2,742	105.9	27,648 GPUs	>1,000
WEST [51]	G_0W_0	No	PW	$O(N_A^4)$	1,728	36-59	25,960 GPUs	>1,500
NanoGW [20]	G_0W_0	No	RSG	$O(N_A^4)$	2,551	–	1,280 CPUs + 80 GPUs	33,120
PWDFT [50]	G_0W_0	No	PW	$O(N_A^3)$	13,824	–	449,280 cores	285
FlapwMBPT [29]	sc$GW^\dagger$	No	RSG	$O(N_A)$ to $O(N_A^2)$	72	–	288 CPUs	18,000
QuaTrEx$_{24}$ [15]	sc$GW^\dagger$	NEGF	MLWF	$O(N_E N_B N_{BS}^3)$	10,560	69.3	7,200 GPUs	31
This work	**sc$GW^\dagger$**	**NEGF**	**MLWF**	$O(N_E N_B N_{BS}^3)$	**46,464** **42,240**	**342.64** **1,146.04**	**9,400 GPUs** **37,600 GPUs**	**25** **42**

*: Estimates based on data provided, except for QuaTrEx$_{24}$, PWDFT, and this work (measurements). †: Scalability and time are given per iteration.

this issue and, therefore, allows for the exploration of the quantum transport properties of devices at the *ab-initio* level, with an atomistic resolution, and in the presence of electron-electron interactions [46]. The latter might play a critical role in future transistors with ultra-short gate lengths. Contrary to expectations, these devices will probably not deliver electronic currents that are closer to their theoretical (ballistic) limit than previous generations [18]. Although electrons (or holes) have a lower probability to interact with crystal vibrations (phonons), rough surfaces, or defects when gate lengths do not exceed 10-15 nm, the densely populated source and drain electrodes of these transistors become so close to each other that the carriers located there start to strongly interact. Additional scattering that does not exist in longer-channel devices is thus induced. It might severely limit the drive current of, for example, nanoribbon field-effect transistors (NRFET), as illustrated in Fig. 1, where the carrier population is highly confined. Hence, the inclusion of electron-electron interactions through the *GW* approximation is indispensable to predict the performance of not-yet-fabricated devices and to guide the design of next-generation transistors with sufficiently high output currents.

The major bottleneck of NEGF+*GW* approaches is their computational intensity. When DFT+NEGF *ab-initio* quantum transport solvers were pioneered in the early 2000s, without *GW* corrections, they were limited to tens of atoms [7]. Algorithmic and hardware developments have pushed back this limit to 10,000s of atoms, still in the ballistic limit of transport (no electron interactions with other particles) [9]. More recently, by leveraging data-centric programming, Joule heating could be introduced into a DFT+NEGF tool to treat realistic transistor geometries on GPUs [52]. However, self-consistently solving for the electron and phonon (crystal vibration) populations increases the workload by almost two orders of magnitude as compared to ballistic transport. Climbing up the ladder of physical and computational complexity, the inclusion of electron-electron interactions within the *GW* approximation is the next step. This relevant feature was demonstrated in 2024 for nanowire test structures made of up to 10,000 atoms [15] but not for devices comparable to those produced in semiconductor fabs.

Here, we present an optimized implementation of the NEGF+*GW* scheme where each computational kernel is carefully optimized to take advantage of the Alps [19] and Frontier [3] supercomputers. Key improvements have been made in the following areas:

- Spatial domain decomposition: Development and integration of a distributed linear solver to compute selected entries of the Green's function and screened Coulomb interaction matrices, allowing for the first simulation of devices with up to 84,480 atoms, in the presence of electron-electron interactions;
- Memory management: Exploitation of the symmetry of all involved physical quantities to reduce the memory footprint, and introduction of a memoization technique to boost specific, otherwise time-consuming calculations;
- Sustained performance: Python-orchestrated code running on Alps and Frontier up to 9,400 and 37,600 GPUs, reaching a FP64 performance of 343 Pflop/s and 1.146 Eflop/s, respectively;
- Time-to-solution: Investigation of realistic NRFETs with dimensions comparable to experimental devices and generating 16× the simulation workload of the state of the art, with only a 35% increase in computational time (42.1 sec per iteration vs. 31.3 sec).

3.1 The *GW* Approximation

Although Hedin's equations have been known for 60 years, successful implementations of the *GW* approximation have not been realized before 1986 and the seminal work by Hybertsen and Louie [25]. The general approach consists of first computing a Green's function G from DFT wavefunctions ψ, then a screened Coulomb interaction W, and finally a self-energy $\Sigma = GW$ that modifies the original DFT results. Although the procedure is self-consistent (G depends on the ψ, which are in turn altered by Σ), it is usually restricted to a one-shot calculation due to its high computational ($O(N_A^4)$) and memory ($O(N_A^3)$) requirements, N_A being the total number of atoms in the system of interest. This is known as the G_0W_0 approximation, where G and W are only evaluated once.

To treat large structures, attempts have been made to improve the scalability of the G_0W_0 method. The developers of VASP [28], a package relying on a plane-wave (PW) basis, managed to decrease the computational complexity from $O(N_A^4)$ to $O(N_A^3)$ [33], while the CP2K team took advantage of the high localization of their Gaussian-type orbitals (GTO) to reach $O(N_A^{2.13})$ and simulate 1,734 atoms [49]. Alternatively, optimizations in the implementation of

the G_0W_0 approximation have been reported, without changing its scalability. BerkeleyGW (2,742 silicon atoms in a PW basis) [4], WEST (1,728 silicon atoms with PW) [51], and NanoGW (silicon quantum dot comprising 2,551 atoms in a real-space grid (RSG)) [20] are excellent examples of recent progress in this area. Finally, combinations of algorithmic innovations and software developments have led to the calculation of systems with up to 13,824 atoms using the PWDFT tool [50].

Going beyond the G_0W_0 approximation in terms of physical accuracy requires solving G and W self-consistently, which can take hundreds of iterations to converge. The resulting self-consistent GW (scGW) schemes scale with $O(N_{iter}N_A^4)$, where N_{iter} is the number of iterations needed to attain convergence. Despite significant scalability improvements, e.g., $O(N_A)$ to $O(N_A^2)$ for FlapwMBPT [29], the capabilities of scGW solvers rarely exceed tens of atoms. At the next level of complexity, the non-equilibrium Green's function formalism is combined with the self-consistent GW approximation to capture correlation effects in nano-devices driven out of equilibrium by external voltages, as the NRFET in Fig. 1. In 2007, it was shown that this can be done in small molecules [46]. In 2024, we investigated a silicon nanowire composed of 10,560 atoms with NEGF+scGW and the first prototype of our quantum transport tool QuaTrEx [15]. Here, we refer to this version of the code from 2024 as QuaTrEx$_{24}$ and to the current one simply as QuaTrEx. Major achievements in the GW approximation are summarized in Table 1.

3.2 The NEGF+sc*GW* Scheme

Contrary to equilibrium G_0W_0 or scGW approaches, NEGF+scGW relies on two particle descriptors (G / W) and two interaction terms (Σ / P) of four different types; retarded (R), advanced (A), lesser ($<$), and greater ($>$). All of them are expressed in a basis of localized atomic orbitals. Their governing matrix equations can be written in the following compact form [14]

$$\left[\mathbf{M}(E) - \mathbf{B}^R(E)\right] \mathbf{X}^{\lessgtr}(E) \left[\mathbf{M}(E) - \mathbf{B}^R(E)\right]^{\dagger} = \mathbf{B}^{\lessgtr}(E), \tag{1}$$

$$\mathbf{B}(E) = \mathbf{B}_{OBC}(E) + \mathbf{B}_{scatt}(E), \tag{2}$$

$$\mathbf{B}_{scatt}(E) \propto \int dE' \mathbf{X}_1(E - E')\mathbf{X}_2(E'). \tag{3}$$

The $\mathbf{X}_{(i)}$ matrices refer to the particle descriptors $\mathbf{G}$ or $\mathbf{W}$, while $\mathbf{B}$ corresponds to the interaction terms $\mathbf{P}$ or Σ. All these quantities are functions of energy E, and the above set of equations typically has to be solved for $10,000 \leq N_E \leq 100,000$ energy points. Furthermore, since $\mathbf{G}$, $\mathbf{P}$, $\mathbf{W}$, and Σ are interdependent, their equations must be repeatedly evaluated until reaching self-consistency. This iterative approach is known as the self-consistent Born approximation (SCBA). Up to N_{iter}=500 iterations are needed to achieve convergence with it. It should be noted that $\mathbf{B}$ contains two parts, one labeled $\mathbf{B}_{OBC}$ that accounts for the so-called open boundary conditions (OBCs) and connects the device with its contacts, and one denoted $\mathbf{B}_{scatt}$ to model electron-electron scattering.

The link between Eqs. (1) to (3) and all underlying physical quantities is provided in Table 2. As NEGF+scGW is an extension to DFT, it takes inputs from it, namely the Hamiltonian ($\mathbf{H}_{\mathrm{DFT}}$), overlap ($\mathbf{S}_{\mathrm{DFT}}$), and bare Coulomb ($\mathbf{V}$) matrices.

Table 2: Definition of the physical quantities in Eqs. (1) to (3).

Quantity	Electrons	Screened Coulomb
$\mathbf{M}(E)$	$E\mathbf{S}_{\mathrm{DFT}} - \mathbf{H}_{\mathrm{DFT}}$	$\mathbf{I}$
$\mathbf{B}^R_{scatt}(E)$	$\Sigma^R(E)$	$\mathbf{V}\mathbf{P}^R(E)$
$\mathbf{B}^{\lessgtr}_{scatt}(E)$	$\Sigma^{\lessgtr}(E)$	$\mathbf{V}\mathbf{P}^{\lessgtr}(E)\mathbf{V}^{\dagger}$
$\widetilde{\mathbf{M}}(E)$	$\mathbf{M}(E) - \mathbf{B}^R_{scatt}(E) - \mathbf{B}^R_{OBC}(E)$	

4 Current State of the Art

Here, we describe the main computational tasks of an NEGF+scGW solver and discuss state-of-the-art algorithms to accomplish them.

As already mentioned, two mutually interacting subsystems are coupled in NEGF+scGW, the electron population and the screened Coulomb interaction. For each of them, three main tasks are sequentially executed after gathering input data from DFT. First, to inject particles into the simulation domain and collect them when they exit, suitable OBCs must be constructed. The second step consists of assembling the system's governing matrix equation and computing its solution for all energies that the particle descriptor $\mathbf{G}(E)$ or $\mathbf{W}(E)$ can occupy. In a last step, the interactions between them, i.e., $\mathbf{P}(E)$ and $\Sigma(E)$, are determined via energy convolution.

Before describing these three tasks and the associated algorithms in detail, we elaborate on the structure of the data we are dealing with in the NEGF+scGW method.

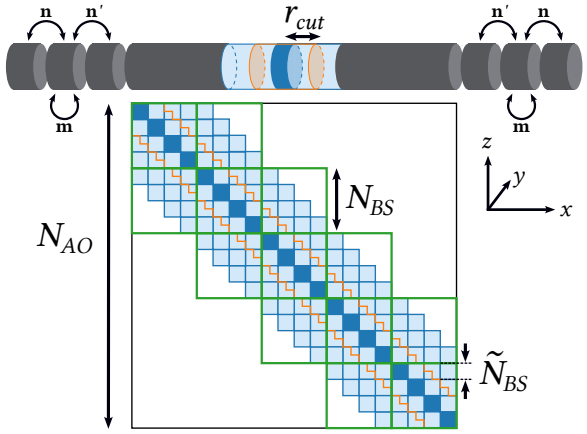
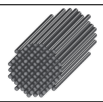
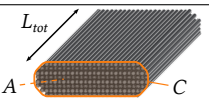


Figure 2: Schematic mapping of a general nanowire structure (top) onto a block-banded matrix of size $N_{AO} \times N_{AO}$. The sparsity pattern can be tiled as a block-tridiagonal matrix with N_B diagonal blocks of size N_{BS} (green boxes). Each of them can be further decomposed into smaller blocks of size $\widetilde{N}_{BS}$ corresponding to the primitive unit cell of the nanostructure. The effect of introducing a cut-off radius r_{cut} for the inter-atomic interactions is shown in orange, both in the matrix and device structure. The periodic contact layers are marked on the left and right of the nanowire.

4.1 Matrix Representations of Nano-devices

When simulating nano-devices out of equilibrium, the structure's periodicity is broken along at least one dimension by the externally

Table 3: Physical dimensions of the nano-device structures considered in this work (NW: nanowire, NR: nanoribbon). The parameters characterizing their numerical representation within the NEGF formalism are listed in the bottom part of the table.

Variable	Description	Formulation	NW-1	NW-2	NR-16	NR-24	NR-40	NR-N_B
								
L_{tot}	Device length (nm)		39.1	34.7	34.7	52.1	86.9	$2.172\times N_B$
A	Device cross section (nm^2)		0.8	4.3	7.5			
C	Device circumference (nm)		3.1	6.9	13.0			
r_{cut}	Interaction cutoff distance (Å)		10.95	7.15	7.5			
N_A	Number of atoms	$O\left((A+C)\,L_{tot}\right)$	2,952	10,560	16,896	25,344	42,240	$1{,}056\times N_B$
N_{AO}	Number of atomic orbitals	$O\left((A+C)\,L_{tot}\right)$	7,488	32,256	54,528	81,792	136,320	$3{,}408\times N_B$
$\widetilde{N}_{BS}$	Primitive unit cell size	$O\left(A\right)$	104	504	852			
N_U	Number of primitive unit cells per transport cell ($\mathbf{G}$/$\mathbf{W}$)	MLWF-dependent	4/8		4			
N_{BS}	Transport cell size for $\mathbf{G}$/$\mathbf{W}$	$\widetilde{N}_{BS}N_U$	416/832	2,016	3,408			
N_B	Number of transport cells ($\mathbf{G}$/$\mathbf{W}$)	N_{AO}/N_{BS}	18/9	16	16	24	40	N_B
H_{NNZ}	Number of non-zeros* in $\mathbf{H}$	$O\left(N_U\widetilde{N}_{BS}N_{AO}\right)$	0.5×10^7	14.1×10^7	40.4×10^7	61.3×10^7	103.1×10^7	$\approx 2.6N_B\times10^7$
G_{NNZ}	Number of non-zeros* in the $\mathbf{G}$, $\mathbf{P}$, $\mathbf{W}$, and $\mathbf{\Sigma}$ matrices	$O\left(r_{cut}\widetilde{N}_{BS}N_{AO}\right)$	0.3×10^7	4.3×10^7	12.6×10^7	19.0×10^7	31.8×10^7	$\approx 0.8N_B\times10^7$

*Number of non-zeros are given when no symmetry applies.

applied voltage. This defines the transport axis, e.g., x, along which carriers (electrons and/or holes) propagate. As a consequence, spatially extended basis functions such as the plane-waves commonly used in DFT and GW (see Table 1) are not suitable for NEGF. A sufficiently localized basis set is needed to construct the $\mathbf{H_{DFT}}$, $\mathbf{S_{DFT}}$, and $\mathbf{V}$ matrices. CP2K [30] or Siesta [2] natively provide such localized functions. Alternatively, maximally localized Wannier functions (MLWF) [36] can be created from a PW basis with Wannier90 [42].

In this work, we consider eight nano-devices whose structure models a silicon channel passivated with hydrogen atoms (see Table 3). Transport occurs along the x axis, while y and z are directions of confinement. We take advantage of MLWF to build their $\mathbf{H_{DFT}}$ and $\mathbf{V}$ matrices ($\mathbf{S_{DFT}} = \mathbf{I}$ because of the orthogonality of our MLWF basis). Using the PW DFT code VASP [28], we first compute the electronic structure of a primitive unit cell (PUC) that can be repeated along the transport axis to obtain the desired device length L_{tot}. The results are converted into MLWFs (4 per Si, 1 per H).

The outcome is a square Hamiltonian matrix $\mathbf{h}_{ii}$ ($\mathbf{h}_{ij}$) of size $\widetilde{N}_{BS}\times\widetilde{N}_{BS}$ that represents the coupling between all MLWFs located within the original PUC with themselves (with MLWFs situated in neighboring cells), as illustrated in Fig. 2. In this example, it can be seen that one PUC (dark blue) is connected to four others along the $+x$ axis ($\mathbf{h}_{ii+1}$ to $\mathbf{h}_{ii+4}$, light blue). Since the magnitude of the interactions between two MLWF situated further apart rapidly decreases, the $\mathbf{h}_{ij}$ blocks with $j > 4$ can be safely ignored.

Because (1) $\mathbf{H_{DFT}}$ is per definition Hermitian and (2) our test structures are made of the repetition of a single PUC along x, the knowledge of the five aforementioned $\mathbf{h}_{ij}$ blocks is sufficient to construct the Hamiltonian matrix of any device with length L_{tot} and total number of atoms N_A. The size of $\mathbf{H_{DFT}}$ is $N_{AO}\times N_{AO}$, where N_{AO} denotes the number of MLWFs in the system. We finally arrive at a block-banded (BB) sparsity pattern for the full Hamiltonian matrix $\mathbf{H_{DFT}}$, as depicted in Fig. 2.

The bare Coulomb matrix $\mathbf{V}$ can be directly computed in the MLWF basis [46]. It is full, as electron-electron interactions typically extend over long ranges. Nevertheless, the magnitude of the V_{ij} entries between two basis functions i and j decays as the distance between them, $R_{ij} = \left|\mathbf{R}_i - \mathbf{R}_j\right|$, increases. If a cut-off radius r_{cut} is applied to the elements of $\mathbf{V}$, i.e., if we keep only the V_{ij} satisfying $R_{ij} \leq r_{cut}$, then $\mathbf{V}$ becomes BB as well. The same simplification can be extended to all other quantities in Eqs. (1) to (3) so that only BB matrices similar to the one in Fig. 2 are involved. The accuracy of this truncation scheme has been validated by Deuschle et al. [14].

4.2 Open Boundary Conditions

An overview of NEGF+scGW calculations and the SCBA loop is visualized in Fig. 3. In the first step, the OBC matrices are computed. It is assumed that the device of interest is contacted by two leads in thermodynamic equilibrium, placed at both extremities of the structure along the transport axis x. In case of transistors, these leads play the role of the source and drain electrodes. The OBC are then cast into the $\mathbf{B}_{OBC}(E)$ matrices in Eq. (1). They can be of retarded or lesser/greater type. Regardless, they occupy the upper left and lower right blocks of $\mathbf{B}_{OBC}(E)$ with size $N_{BS}\times N_{BS}$. All the other blocks are equal to zero.

4.2.1 Retarded OBC. The physical properties of the leads give rise to a recursion relation with the following form [6]

$$\mathbf{x}^R = \left(\mathbf{m} - \mathbf{n}\mathbf{x}^R\mathbf{n}'\right)^{-1}. \tag{4}$$

Here $\mathbf{x}^R$ denotes a surface matrix of size $N_{BS}\times N_{BS}$, whereas $\mathbf{m}$, $\mathbf{n}$ and $\mathbf{n}'$ refer to blocks of the $\mathbf{M}(E) - \mathbf{B}^R_{scatt}(E)$ matrices (see Fig. 2).

Several methods have been proposed to solve the non-linear Eq. (4). They can be generally divided into two classes, *iterative* and *direct* techniques. A straightforward iterative approach is to perform fixed-point iterations as

$$\mathbf{x}^R_{i+1} = \left(\mathbf{m} - \mathbf{n}\mathbf{x}^R_i\mathbf{n}'\right)^{-1} \tag{5}$$

until $\left\|\mathbf{x}^R_{i+1} - \mathbf{x}^R_i\right\|$ falls below an imposed convergence criterion. Every iteration comes at the cost of multiplying three matrices and

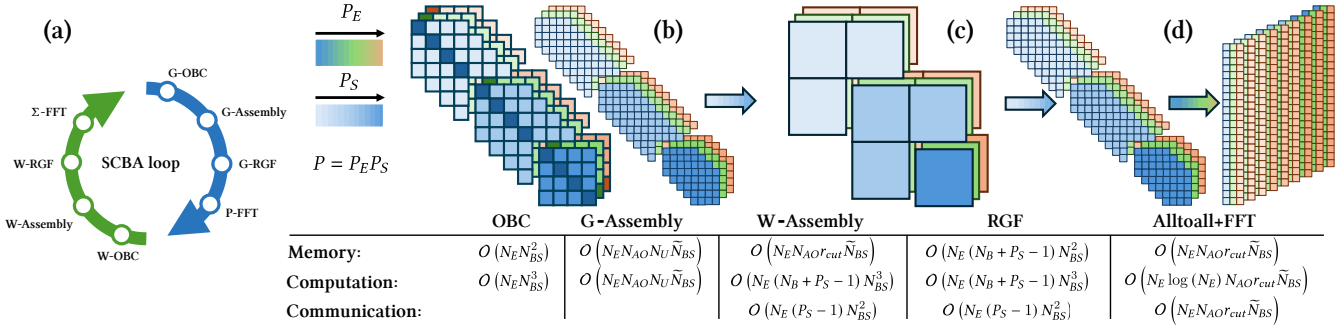


Figure 3: Evolution of the data distribution throughout an SCBA iteration. (a) Illustration of the self-consistent G → P → W → Σ cycle. (b) Data distribution through the energy stack. (c) Selected solution of the quadratic matrix problem in Eq. (1) with RGF. (d) Data distribution through the non-zero elements of the matrices and FFT kernel.

inverting one. Since the convergence (if any) of Eq. (5) can take 100s of iterations, alternatives have been developed to accelerate the process. The well-established Sancho-Rubio method [45] significantly reduces the number of required iterations, but it still remains in the order of 10s. Also, as the convergence rate of Eq. (5) can be different for each energy and we want to treat the N_E points in parallel, iterative methods potentially lead to load imbalance.

Another approach is to determine the $\mathbf{x}^R$ matrix in Eq. (4) *directly* by solving a polynomial eigenvalue problem (PEVP) [8, 34]

$$\left[\sum_{i=-N_U}^{N_U} \lambda^i \widetilde{\mathbf{m}}_i\right] \phi = 0, \tag{6}$$

where the $\widetilde{\mathbf{m}}_i$ matrices are of size $\widetilde{N}_{BS} \times \widetilde{N}_{BS}$. They are also extracted from $\mathbf{M}(E) - \mathbf{B}^R_{scatt}(E)$. The surface function $\mathbf{x}^R$ can be reconstructed from the eigenvectors ϕ and eigenvalues λ.

Contour integral methods such as the Beyn algorithm can be harnessed to tackle Eq. (6) [5]. A contour is first defined in the complex plane, two projectors $\mathbf{Q}_0$ and $\mathbf{Q}_1$ are created by solving sparse linear systems of equations at predefined points on this contour, the results are summed up, a singular value decomposition (SVD) of $\mathbf{Q}_0$ is performed, and the produced data is used to assemble a regular, non-symmetric eigenvalue problem (EVP) of significantly reduced dimensions. The EVP is solved to obtain the desired ϕ, λ, and finally $\mathbf{x}^R$ [8]. Once $\mathbf{x}^R$ is available, the non-zero blocks of $\mathbf{B}^R_{OBC}$ can be computed as $\mathbf{n}\mathbf{x}^R\mathbf{n}'$. While the Beyn algorithm returns accurate surface functions, it involves SVD and non-symmetric EVP operations that do not perform well on GPUs [15].

4.2.2 Lesser/Greater OBC. The lesser/greater boundary self-energy, $\Sigma^{\lessgtr}_{OBC}$, can be derived analytically from the retarded surface Green's function $\mathbf{g}^R = \mathbf{x}^R$ in Eq. (4) through the fluctuation-dissipation theorem [10]. In case of the screened Coulomb interaction, the calculation of the required surface function $\mathbf{w}^{\lessgtr}$ is more complex as it involves solving a discrete-time Lyapunov equation [14]

$$\mathbf{w}^{\lessgtr} = \mathbf{q}^{\lessgtr} - \mathbf{a}\mathbf{w}^{\lessgtr}\mathbf{a}^\dagger, \tag{7}$$

where $\mathbf{q}^{\lessgtr}$ and $\mathbf{a}$ are blocks of size $N_{BS} \times N_{BS}$ that can be extracted from the $\mathbf{P}$, $\mathbf{V}$, and $\mathbf{w}^{\mathbf{R}}$ matrices. Equation (7) is standard in control systems, but not yet in quantum transport. As in the retarded case, it can be solved iteratively [43] or directly [26]. None of these techniques is ideal: The former converges slowly, and a matrix of size slightly smaller than N_{BS} must be diagonalized in the latter.

4.3 Selected System Solver

The second step of the NEGF+sc*GW* method is concerned with the "selected" solution of the quadratic matrix problem in Eq. (1) where both the $\mathbf{M}(E)$ (left-hand-side, (LHS)) and $\mathbf{B}^{\lessgtr}(E)$ (right-hand-side, (RHS)) matrices are block-banded as in Fig. 2. "Selected," in this context, means that we are not interested in all entries of the solution $\mathbf{X}^{\lessgtr}(E)$, but only in those that lie within the r_{cut} boundaries in Fig. 2. The so-called recursive Green's function (RGF) solver provides the desired elements of $\mathbf{X}^{\lessgtr}(E)$ with little fill-in [31]. However, it does not apply to general BB matrices, but only to block-tridiagonal (BT) ones. To address this issue, N_U primitive blocks $\mathbf{h}_{ij}$ in Fig. 2 are grouped together to form transport cells with green-colored borders and size $N_{BS} = \widetilde{N}_{BS} N_U$. This partitioning leads to a BT sparsity pattern with N_B diagonal blocks.

4.3.1 Assembly of the Linear System. Before solving Eq. (1) for a given energy point E, its $\widetilde{\mathbf{M}}$ and $\mathbf{B}^{\lessgtr}$ matrices must first be assembled (see Table 2). More concretely, in the G-solver, the matrix

$$\widetilde{\mathbf{M}}(E) = E\mathbf{S}_{\mathrm{DFT}} - \mathbf{H}_{\mathrm{DFT}} - \Sigma^R_{scatt}(E) - \Sigma^R_{OBC}(E) \tag{8}$$

has to be generated, which requires performing an element-wise subtraction of BB matrices and two OBC blocks. To assemble the RHS ($\Sigma^{\lessgtr}$), $\Sigma^{\lessgtr}_{scatt}$ and $\Sigma^{\lessgtr}_{OBC}$ must be added element-wise. The resulting computational cost is in the same order as the largest number of non-zeros among these matrices, i.e., $O(N_{AO}\max(H_{BW}, \Sigma_{BW}))$, where H_{BW} (Σ_{BW}) is the bandwidth of $\mathbf{H}_{\mathrm{DFT}}$ (Σ).

In the W-solver, the construction of both $\widetilde{\mathbf{M}}$ and $\mathbf{B}^{\lessgtr}$ involves matrix multiplications, $\mathbf{V}\mathbf{P}^R$ and $\mathbf{V}\mathbf{P}^{\lessgtr}\mathbf{V}^\dagger$, respectively. Since $\mathbf{P}^{R,\lessgtr}$ and $\mathbf{V}$ have the same r_{cut}-dependent bandwidth $P_{BW} = V_{BW}$, the computational cost of assembling $\widetilde{\mathbf{M}}$ and $\mathbf{B}^{\lessgtr}$ is equal to $O(N_{AO}P^2_{BW})$. These operations can be performed either in a BB or even BT representation to increase the computational efficiency. In the latter case, the block-bandwidth of the matrices is three ($\mathbf{V}$, $\mathbf{P}$), five ($\mathbf{V}\mathbf{P}^R$), and seven ($\mathbf{V}\mathbf{P}^{\lessgtr}\mathbf{V}^\dagger$). The overall complexity increases to $O(N_B N^3_{BS})$, but the multiplication of larger blocks is ideally suited to GPUs.

4.3.2 Recursive Green's Function (RGF) Solver. In the device modeling community, the RGF algorithm is widely seen as the most

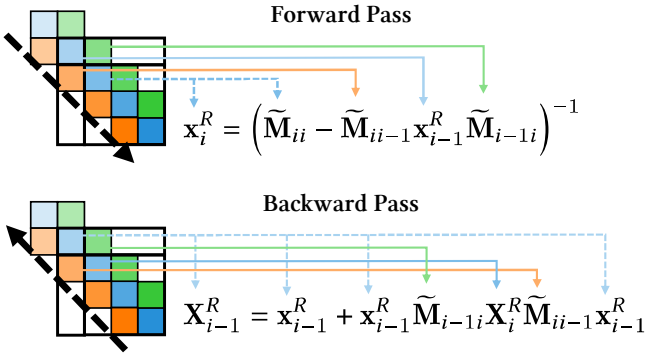


Figure 4: Visualization of RGF's recursive Schur complement approach to obtain the main diagonal blocks of the selected inverse of $\widetilde{\mathbf{M}}$, $\mathbf{X}^R$. The forward pass iterates from the top left to the bottom right, while the backward pass iterates in the opposite direction. In every i-th forward pass iteration, we consider the $\widetilde{\mathbf{M}}_{i:,i:}$ submatrix and compute $\mathbf{x}_i^R$, as in Eq. (9). Similarly, in the backward pass, the computation of $\mathbf{X}_{i-1}^R$ depends on the value from the previous iteration, $\mathbf{X}_i^R$ (Eq. (11)).

powerful method to solve Eq. (1) for Green's functions [31] and for screened Coulomb interactions [14]. It relies on a recursive calculation of the Schur complement of the system matrix $\widetilde{\mathbf{M}}$ and exploits the BT sparsity pattern of this matrix and of $\mathbf{B}^\lessgtr$. The RGF method articulates itself around a forward pass during which intermediate quantities, $\mathbf{x}^R$ and $\mathbf{x}^\lessgtr$, are computed, followed by a backward pass that produces the targeted blocks of $\mathbf{X}^\lessgtr$ (diagonal and first off-diagonal). One iteration of the forward and backward passes is shown in Fig. 4 to compute the selected inverse of $\widetilde{\mathbf{M}}$, $\mathbf{X}^R$.

The recursion starts, for example, at index $i = 1$ by determining $\mathbf{x}_1^R = \widetilde{\mathbf{M}}_{11}^{-1}$ and $\mathbf{x}_1^\lessgtr = \mathbf{x}_1^R\mathbf{B}_{11}^\lessgtr\mathbf{x}_1^{R\dagger}$, where all matrices are of size $N_{BS} \times N_{BS}$, i.e., they correspond to the green blocks in Fig. 2. The RGF algorithm provides then a recursion for the indices $2 \leq i \leq N_B$

$$\mathbf{x}_i^R = \left(\widetilde{\mathbf{M}}_{ii} - \widetilde{\mathbf{M}}_{ii-1}\mathbf{x}_{i-1}^R\widetilde{\mathbf{M}}_{i-1i}\right)^{-1}, \tag{9}$$

$$\mathbf{x}_i^\lessgtr = \mathbf{x}_i^R\left(\mathbf{B}_{ii}^\lessgtr + \widetilde{\mathbf{M}}_{ii-1}\mathbf{x}_{i-1}^\lessgtr\widetilde{\mathbf{M}}_{i-1i}^\dagger - \left[\mathbf{y}_i^\lessgtr - \mathbf{y}_i^{\lessgtr\dagger}\right]\right)\mathbf{x}_i^{R\dagger}, \tag{10}$$

with $\mathbf{y}_i^\lessgtr = \widetilde{\mathbf{M}}_{ii-1}\mathbf{x}_{i-1}^\lessgtr\mathbf{B}_{i-1i}^\lessgtr$. At the end of the forward pass, $\mathbf{X}_{N_B}^\lessgtr = \mathbf{x}_{N_B}^\lessgtr$ and $\mathbf{X}_{N_B}^R = \mathbf{x}_{N_B}^R$. This marks the beginning of the backward pass during which the index i goes from $N_B - 1$ to 1

$$\mathbf{X}_i^R = \mathbf{x}_i^R + \mathbf{x}_i^R\widetilde{\mathbf{M}}_{ii+1}\mathbf{X}_{i+1}^R\widetilde{\mathbf{M}}_{i+1i}\mathbf{x}_i^R, \tag{11}$$

$$\mathbf{X}_i^\lessgtr = \mathbf{x}_i^\lessgtr + \mathbf{x}_i^R\widetilde{\mathbf{M}}_{ii+1}\mathbf{X}_{i+1}^\lessgtr\widetilde{\mathbf{M}}_{i+1i}^\dagger\mathbf{x}_i^{R\dagger} - \left(\mathbf{Y}_i^\lessgtr - \mathbf{Y}_i^{\lessgtr\dagger}\right) + \left(\mathbf{Z}_i^\lessgtr - \mathbf{Z}_i^{\lessgtr\dagger}\right). \tag{12}$$

Here, $\mathbf{Y}_i^\lessgtr$ and $\mathbf{Z}_i^\lessgtr$ are defined as $\mathbf{Y}_i^\lessgtr = \mathbf{x}_i^R\mathbf{B}_{ii+1}^\lessgtr(\mathbf{x}_i^R\widetilde{\mathbf{M}}_{ii+1}\mathbf{X}_{i+1}^R)^\dagger$ and $\mathbf{Z}_i^\lessgtr = \mathbf{x}_i^R\widetilde{\mathbf{M}}_{ii+1}\mathbf{X}_{i+1}^R\widetilde{\mathbf{M}}_{i+1i}\mathbf{x}_i^\lessgtr$, respectively. The first off-diagonal blocks of $\mathbf{X}^\lessgtr$ ($\mathbf{X}_{ii+1}^\lessgtr$) can also be computed with RGF [14].

There are two main challenges associated with the RGF algorithm. First, owing to its recursive nature, the method is inherently sequential and thus limited to moderately large device structures. Secondly, because of the accumulation of small numerical errors over multiple SCBA iterations, the $\mathbf{X}^\lessgtr$ matrices tend to lose their symmetry ($X_{ij}^\lessgtr = -X_{ji}^{\lessgtr *}$), which deteriorates the convergence of NEGF+scGW. Regular symmetrizations of $\mathbf{X}^\lessgtr$ are therefore needed.

4.3.3 Computational Complexity. The RGF algorithm consists of N_B steps in the forward and backward pass. For each of this step, matrices of size $N_{BS}\times N_{BS}$ are either multiplied or inverted such that its computational complexity is equal to $O(N_B N_{BS}^3)$, or equivalently $O(N_{AO}N_{BS}^2)$. However, N_E energy points must be computed for each SCBA iteration so that the cost increases to $O(N_E N_B N_{BS}^3)$ (see Table 1). Without RGF, it would be $O(N_E N_{AO}^3)$.

4.4 Energy Convolutions

The Green's functions and screened Coulomb interactions enter Eq. (3) to calculate either the scattering self-energy $\Sigma(E)$ (element-wise multiplication of $\mathbf{G}(E')$ and $\mathbf{W}(E-E')$) or the polarization function $\mathbf{P}(E)$ (element-wise multiplication of two $\mathbf{G}$ at E' and $E - E'$). The required energy convolutions can be replaced by fast Fourier transforms (FFTs), which provide efficient conversions between the energy and time domains (and vice versa), thereby reducing the complexity of Eq. (3) from $O(N_E^2)$ to $O(N_E \log N_E)$.

The computational complexity is not the only challenge here. It can be seen that the solution of Eq. (1) returns the desired (i, j) entries of the $\mathbf{X}^\lessgtr$ matrix for one given energy E. All these energies are independent of each other and can be treated in parallel. However, in Eq. (3), obtaining $\mathbf{B}_{scatt}(E)$ requires gathering all energies of two matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ that are multiplied element-wise. Hence, the FFTs can be performed for specific (i, j) entries, independently from the others. To address this data distribution duality, a solution was proposed in [52]: data transposition (see Fig. 3). When dealing with Eq. (1), each rank stores all (i, j) entries of $\mathbf{X}^\lessgtr$ for one (or a few) energy point(s). After data transposition, they store all energies of $\mathbf{X}_1$ and $\mathbf{X}_2$ for a few (i, j) entries, and can then process Eq. (3).

4.5 Observables

The simulation of nano-transistors such as the NRFET from Fig. 1 is expected to produce relevant observables, e.g., the local density-of-states (DOS), charge density $\rho(r)$, or electronic (energy) current I_d (I_{dE}). All can be derived from the diagonal and first off-diagonal blocks of the lesser and greater Green's functions. The GW-renormalized band structure can also be obtained from the DFT Hamiltonian $\mathbf{H}_{\mathrm{DFT}}$ and the retarded scattering self-energy Σ_{scatt}^R.

5 Innovations Realized

To address all the challenges arising in the NEGF+scGW method, we present algorithmic and programming developments that leverage both the physics at play and the underlying mathematical structure of the computational problem. The innovations and optimizations concern the programming model, symmetry exploitation, dynamic OBC memoization, and a distributed solver to enable unprecedented spatial domain decomposition of Eq. (1).

5.1 Programming Model

We implemented a new, modular NEGF+scGW simulator in Python, making use of the NumPy [23], CuPy [38], mpi4py [13], Numba [32],

and Scipy[48] packages, which provide a high-level interface to low-level, high-performing vendor-specific BLAS, LAPACK, FFT, and communication libraries. By harnessing the abstractions of these packages, our software is completely hardware-agnostic and runs on any CPU- and GPU-based system with different communication libraries (MPI / *CCL). Where package-provided implementations do not suffice, we implement and dispatch custom CPU and GPU kernels, using the just-in-time compilation functionality of CuPy and Numba. We also develop wrappers for certain linear algebra kernels (SVD, QR, EVP, etc.), to facilitate dispatching to the CPU. This is needed in case GPU implementations do not exist (non-symmetric EVP) or are outperformed by CPU ones.

5.2 Exploiting Symmetry

As mentioned in Section 4.3.2, the lesser/greater quantities of the NEGF+sc*GW* model must fulfill a very specific symmetry relationship ($X^{\lessgtr}_{ij} = -X^{\lessgtr *}_{ji}$). At each execution of the RGF algorithm, slight deviations from this condition are induced, for example, due to the floating-point arithmetic. These errors propagate to the $\mathbf{B}^{\lessgtr}_{scatt}$ matrices, which should also satisfy $B^{\lessgtr}_{scatt,ij} = -B^{\lessgtr *}_{scatt,ji}$. At the next SCBA iteration, they are injected back into the RGF solver. Overall, the errors accumulate over time, slowing down or even preventing the convergence of the $\mathbf{G} \rightarrow \mathbf{P} \rightarrow \mathbf{W} \rightarrow \Sigma$ cycle.

By enforcing the desired symmetry properties at each SCBA iteration, convergence is accelerated, or restored in the worst-case scenario. The symmetrization needed is applied by computing $X^{\lessgtr}_{ij} = (X^{\lessgtr}_{ij} - X^{\lessgtr *}_{ji})/2$, which necessitates that both the upper and lower off-diagonal blocks of the matrix $\mathbf{X}^{\lessgtr}$ are determined and stored explicitly. The same must be done for $\mathbf{B}^{\lessgtr}$. Such an approach incurs significant computational and memory overhead.

In this work, we improve the situation by absorbing the symmetrization into the data structure and the main computational kernels. Hence, symmetries are enforced on-the-fly, without sacrificing the convergence stability offered by computing all values. The computational workload remains mostly unchanged, but the memory cost is significantly lowered by storing only the upper triangular part of the quantities obeying symmetry properties. Furthermore, the communication volume during data transposition and the time to calculate the matrices $\mathbf{B}^{\lessgtr}_{scatt}$ with Eq. (3) are halved.

5.3 OBC Memoization

In Section 4.2, we highlighted that direct solutions of the retarded and lesser/greater OBCs with Eqs. (4) and (7), respectively, always provide accurate results, contrary to iterative solutions of the same equations, whose convergence is slow, especially if the initial guess for the quantity searched is set to zero. By closely analyzing the convergence behavior of the NEGF+sc*GW* scheme for different device examples and through many SCBA cycles, we found that, already after relatively few iterations, the retarded and lesser/greater OBC blocks tend to stabilize, i.e., they stop varying significantly from one SCBA iteration to the next. This observation motivated us to switch from a direct to an iterative approach as soon as the OBC blocks become stable. Indeed, the time to iteratively compute these blocks drastically decreases when the solution from the previous SCBA iteration is very close to the new one and is used as an initial guess. Moreover, our code identifies the moment to dynamically transition from one mode of calculation to the other.

The switching from a direct to an iterative method requires storing the OBC blocks at each SCBA iteration. This technique, therefore, trades computational speed-up for increased memory usage, which lends itself well after having previously "freed up" memory by exploiting data symmetry. We thus implemented a memoization scheme for the retarded and lesser/greater OBC blocks. More concretely, we initially cache the $\mathbf{x}^{R\lessgtr}(E)$ obtained from a call to the corresponding *direct* OBC solver for each contact (left and right) in each subsystem ($\mathbf{G}$ / $\mathbf{W}$). In the following SCBA iteration, we retrieve the cached results and compute the update $\|\mathbf{x}^{R\lessgtr}_{i+1} - \mathbf{x}^{R\lessgtr}_{i}\|$. From this update, the memoizer estimates whether it can reach convergence in a predetermined number of fixed-point iterations N_{FPI}. If a predefined condition is satisfied, the N_{FPI} iterations are performed, the result is returned, and the cache is updated. If the allotted N_{FPI} is deemed to be insufficient to achieve convergence, the direct solver is called instead. In this way, the memoizer decides at runtime whether to use the more robust direct method or the significantly more performant iterative one.

Since we solve the OBC on all ranks, the speed-up only truly manifests if all ranks decide to memoize. Allotting a fixed number of iterations N_{FPI} prevents load imbalance, should the memoizers on all ranks succeed. In our analysis, we found that the lesser/greater recursion relation in Eq. (7) typically stabilizes within fewer than 10 iterations. The retarded recursion relation in Eq. (4) also tends to stop varying much after 20 iterations. Since typically 100s of SCBA iterations are needed to reach convergence, this means we usually can expect all memoizers to succeed in >90% of iterations.

5.4 Spatial Domain Decomposition

The computation of OBCs and the subsequent assembly of each subsystem's matrices are embarrassingly parallel through the energy grid. Decomposition of the spatial domain, on the other hand, is significantly less straightforward due to the sequential dependence that the RGF algorithm introduces between spatial subdomains (see Section 4.3.2). To overcome this limitation, we developed and integrated a novel distributed algorithm for the production of selected elements of the solution to Eq. (1) [35]. Similarly to RGF, this method is based on the Schur complement. Our algorithm extends prior works [11, 41], employing a nested-dissection scheme to permute the system matrix and the RHS $\mathbf{B}^{\lessgtr}$, enabling the concurrent solution of the equation in different sections of the spatial domain.

The nested-dissection scheme is visualized in Fig. 5. A BT matrix $\mathbf{A}$ ($\widetilde{\mathbf{M}}$ and $\mathbf{B}^{\lessgtr}$) is split into P_S disjoint partitions, as shown with different shades of blue. We explicitly select the visualized *arrow* partitioning so that every $\mathbf{A}_{i,j}$ off-diagonal block lies in the same partition as its symmetric $\mathbf{A}_{j,i}$. The symmetrization techniques discussed in Section 5.2 naturally extend to this spatial decomposition scheme without inducing extra communication across the partitions' boundaries. Multiplying $\mathbf{A}$ with a suitable permutation matrix $\mathbf{P}$ per partition from the left and right exposes parallel sections, where RGF's forward and backward passes can be executed concurrently. However, this parallelization comes at a cost: At the end of the forward pass, a *reduced system* (in green color) must be constructed and selectively solved before proceeding with the

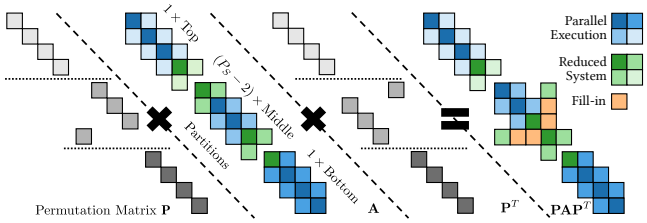


Figure 5: Visualization of the nested-dissection scheme for RGF. From left to right: permutation matrix P, BT matrix A (applies to both $\widetilde{\mathbf{M}}$ and $\mathbf{B}^{\lessgtr}$), $\mathbf{P}^T$, and $\mathbf{PAP}^T$. The scheme splits the matrices into P_S partitions, 1 top, 1 bottom, and $P_S - 2$ middle ones. The permutation applies separately to each partition, breaking the sequential dependencies among them in RGF's forward and backward passes, thus enabling parallel execution. Between the passes, a reduced system must be solved (green), inducing additional workload.

backward pass. Furthermore, each *middle* partition has to compute additional $O(N_B/P_S)$ blocks as fill-in (in orange color), causing load imbalance. The reduced system increases the total computational workload by $O(P_S N_{BS}^3)$ and the overall memory requirements by $O(P_S N_{BS}^2)$, while adding a $O(P_S N_{BS}^2)$ communication cost to gather the system, compared to sequential RGF. These additional costs can nevertheless be distributed over multiple ranks by applying the nested-dissection scheme to the reduced system recursively.

To take advantage of this nested-dissection solver, the domain decomposition must be applied to all other tasks as well. While this is trivial for the most part, the BT matrix multiplications in the assembly of $\widetilde{\mathbf{M}}$ and $\mathbf{B}^{\lessgtr}$ for $\mathbf{W}$ (see Section 4.3.1) requires the implementation of a specific halo communication scheme to exchange blocks at the partition boundaries.

6 How Performance was Measured

6.1 Hardware and Measurement Setup

Our measurements were done on the Alps and Frontier supercomputers. Alps is made of HPE Cray EX254n blades, each housing two nodes, for a total of 2,600 [19]. Every node contains four NVIDIA GH200 superchips (72-core Grace CPU and one Hopper GPU), with 96 GB of high-bandwidth memory (HBM) each. Intranode communication uses NVLink interconnect with 150 GB/s bidirectional bandwidth per link. Every GH200 is connected to a Slingshot network through a separate network interface card (NIC), with 25 GB/s bidirectional bandwidth. Hopper's theoretical maximum double-precision floating-point (tensor-core) performance (FP64) is 67 Tflop/s. Alps' Rpeak and Rmax [47] per GH200 superchip are 55.3 and 41.8 Tflop/s, respectively. Frontier consists of 9,604 Cray EX 235a nodes [3]. Every node has four AMD Instinct MI250X GPUs and an AMD EPYC 64-core Trento CPU. Each MI250X GPU is made of two separate graphics compute dies (GCD) with 64 GB HBM memory. Intranode communication uses Infinity Fabric interconnect with up to 50 GB/s bidirectional bandwidth. Similarly to Alps, each GPU is connected to a Slingshot network through a separate NIC with 25 GB/s bidirectional bandwidth. Each MI250X GPU (GCD) has a theoretical maximum FP64 (tensor-core) performance of 95.7 (47.9) Tflop/s. Frontier's Rpeak and Rmax per GPU (GCD) are 53.5 (26.8) and 35.2 (17.6) Tflop/s, respectively.

6.2 Test Structures

We use, in total, eight different nanowire and nanoribbon devices, as presented in Table 3, to benchmark the performance of our NEGF+sc*GW* solver, QuaTrEx. All structures model a silicon transistor, the surface of which is passivated with hydrogen atoms. The first two (NW-1 and NW-2) are nanowires that are identical to the configurations labeled "medium" and "large" in [15]. The six other devices, NR-{16, 24, 40} on Frontier and NR-{23, 44, 80} on Alps, are NRFETs with the same cross section (~1.5×5 nm^2) as in Fig. 1. They correspond to devices fabricated by Intel and reported in December 2024 at the International Electron Devices Meeting (IEDM) [1]. Each transport cell of length L_{TC}=2.172 nm contains 1,056 atoms (N_{BS}=3,408). The digits following "NR-" indicate the total number of transport cells. Hence, structure NR-40, measures L_{tot}=86.9 nm comprising 42,240 atoms.

We use as many energy points per compute unit as possible, saturating the available GPU memory when running experiments at scale. Due to GH200's larger HBM (96 GB) compared to the MI250X GCD (64 GB), the NW-X devices can be run with more energy points per compute device. Similarly, without domain decomposition, the largest possible structure on Alps is NR-23, while Frontier is limited to the smaller NR-16. When turning on domain decomposition onto two compute units ($P_S = 2$), the maximum size increases to NR-24 on Frontier and NR-44 on Alps. Finally, with four compute units ($P_S = 4$), we can run NR-40 on Frontier and NR-80 on Alps.

6.3 Testing Methodology

In our benchmarks, we report the median of at least 10 measurements obtained by executing consecutive SCBA iterations. We set the self-energy to zero at the beginning of each iteration to ensure stability with a small number of nodes/energy points. The first iteration is always discarded since it includes the just-in-time (JIT) compilation of compute kernels and other initial overhead.

Workload measurements (all in FP64) are performed with the vendor-provided tools, AMD ROCProfiler (rocprof) and NVIDIA Nsight Compute (NCU). NCU could only be used for NW-1 since its runtime drastically increases with the device size. On NW-2, we interpolate the rocprof's measurements by scaling the relevant workload with the increase in the energy points. On the NR-X configurations, we use rocprof's measurements for the kernels that are independent of the device length (e.g., OBCs). On the other hand, the workload of the kernels that scale with device length (e.g., RGF) is dominated by BLAS level 3 calls (mainly GEMM). Therefore, we individually profile those calls with rocprof and NCU, and subsequently multiply the result by their exact count.

7 Performance Results

7.1 Micro-benchmarks

To evaluate our software, QuaTrEx, and compare it with its predecessor QuaTrEx$_{24}$ [15] (see Table 1), we first discuss single-device (GPU Alps / GCD Frontier) performance of the main computational

Table 4: Workload, time, and performance per SCBA iteration on a single GCD (Frontier) / GPU (Alps) of the main kernels for the NW-1, NW-2, NR-16, and NR-23 device structures. Median values of at least 10 measurements are reported.

Software	QuaTrEx$_{24}$	This Work				QuaTrEx$_{24}$	This Work				This Work			
Device	NW-1					NW-2					NR-16		NR-23	
Machine	LUMI	Frontier		Alps		LUMI	Frontier		Alps		Frontier		Alps	
Energies	32	50		80		1	4		6		1		1	
Memoizer	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗	✓
Devices & Machines														
Workload [Tflop]														
G: OBC	1.242	1.635	0.619	1.796	0.811	–	8.554	4.878	12.831	7.317	9.686	5.809	9.686	5.809
G: RGF	16.035	21.071		28.821		–	149.930		224.895		167.704		244.077	
W: Assembly														
Beyn	–	6.585	4.729	8.613	6.461	–	6.556	4.878	9.834	7.317	7.629	5.809	7.629	5.809
Lyapunov	–	8.613	5.900	11.114	10.964	–	12.490	8.126	18.735	12.189	8.486	5.875	8.486	5.875
LHS-$\mathbf{B}^{R}_{scatt}$	–	5.170		7.114		–	36.292		54.438		44.287		64.504	
RHS-$\mathbf{B}^{\lessgtr}_{scatt}$	–	21.076		38.051		–	149.518		224.277		181.056		261.904	
Total	12.680	41.444	36.875	64.891	62.590	–	204.856	198.814	307.284	298.221	241.458	237.027	342.523	338.092
W: RGF	55.424	77.011		109.384		–	149.90		224.895		167.704		244.077	
Other	–	4.643	6.706	10.611	13.898	–	39.264	36.65	58.896	54.975	3.345	1.338	3.345	1.338
Total Work	85.381	145.804	142.283	215.500	209.603	151.069	552.534	540.202	828.801	810.303	589.897	579.582	843.707	833.392
Time [s]														
G: OBC	1.298	0.901	0.161	0.611	0.176	4.178	2.505	0.649	1.987	0.470	2.402	0.650	2.371	0.250
G: RGF	1.050	1.621		1.396		2.445	8.402		7.367		7.873		5.963	
W: Assembly														
Beyn	1.630	1.193	0.421	1.311	0.745	1.885	2.376	0.581	1.890	0.423	1.992	0.533	2.237	0.208
Lyapunov	1.098	4.468	0.372	2.211	0.264	2.628	11.713	0.457	13.008	0.378	19.658	0.501	22.578	0.279
LHS-$\mathbf{B}^{R}_{scatt}$	3.547	0.384		0.299		2.362	1.539		1.113		1.672		1.257	
RHS-$\mathbf{B}^{\lessgtr}_{scatt}$		1.333		0.916			5.678		4.3715		6.077		4.792	
Total	6.230	7.378	2.51	4.7365	2.224	10.593	21.306	8.255	20.383	6.286	29.399	8.783	30.864	6.536
W: RGF	2.866	3.705		3.753		2.357	7.2105		6.052		7.800		5.956	
Other	1.104	1.747		1.1		6.244	2.883		1.088		4.556		1.021	
Total Time	12.458	15.376	9.744	11.769	8.643	25.817	42.097	27.404	36.943	21.263	52.695	29.662	46.175	19.726
Time/Energy	0.389	0.308	**0.195**	0.147	**0.108**	25.817	10.524	**6.851**	6.157	**3.544**	52.695	**29.662**	46.175	**19.726**
Performance [Tflop/s]														
Performance	6.854	9.483	14.602	18.311	24.251	5.852	13.125	19.713	22.435	38.109	11.195	19.540	18.272	42.248

kernels, as shown in Table 4. QuaTrEx$_{24}$'s measurements were obtained on LUMI [16], which has the same architecture as Frontier, and are also reported per GCD. Thanks to computational optimizations and memory savings attained by exploiting data symmetries, our software achieves a higher throughput of energy points per time. We fit 1.56× (NW-1) / 4.00× (NW-2) the number of energies onto one MI250X GCD while simultaneously increasing the total runtime by only 1.23× (NW-1) and 1.63× (NW-2), for an overall 1.27× (NW-1) and 2.45× (NW-2) speed-up per energy. The performance is further improved when enabling memoization, resulting in 2.00× (NW-1) and 3.77× (NW-2) speed-up per energy. When simulating the NR-16 (Frontier) and NR-23 (Alps), our software achieves high performance, especially when OBC memoization is enabled; approximately 72.9% and 76.4% of the respective machine's Rpeak.

We proceed with the last four device configurations, NR-{24, 40} (Frontier) and NR-{44, 80} (Alps), containing an unprecedented number of atoms, 25,344, 42,240, 44,464, and 84,480, respectively. To simulate such large devices, we employ spatial-domain decomposition with $P_S = 2$ or 4 processes treating a single energy. Table 5 presents our software's performance with the minimum number of compute devices required to run a simulation (approximately 2-4 energies). As mentioned in Section 5.4, distributed RGF induces an increased workload for the middle partitions. In this work, we do not employ any load-balancing techniques yet, so that the boundary partitions perform about 60% of the middle partitions' workload.

Table 5: Workload, time, and performance measurements for one energy point of the NR-{24, 40, 44, 80} devices, with spatial domain decomposition and $P_S = 2$ or 4. Median values of at least 10 measurements are reported.

Software	This Work			
Machine	Frontier		Alps	
Device	NR-24	NR-40	NR-44	NR-80
P_S	2	4	2	4
Energies	1			
Memoizer	✓	✓	✓	✓
Workload [Tflop]				
Top partition	483.547	490.711	899.501	906.632
Middle partitions (per rank)	—	771.766	—	1,536.419
Bottom partition	526.531	532.392	948.770	954.605
Total	1,010.078	2,566.635	1,848.271	4,934.075
Total Time [s]				
All partitions	25.452	34.543	21.742	36.722
Performance [Tflop/s]				
Top partition	18.998	14.206	41.372	24.689
Middle partitions (per rank)	—	22.342	—	41.839
Bottom partition	20.687	15.412	43.638	25.995
Total	39.686	74.303	85.009	134.363

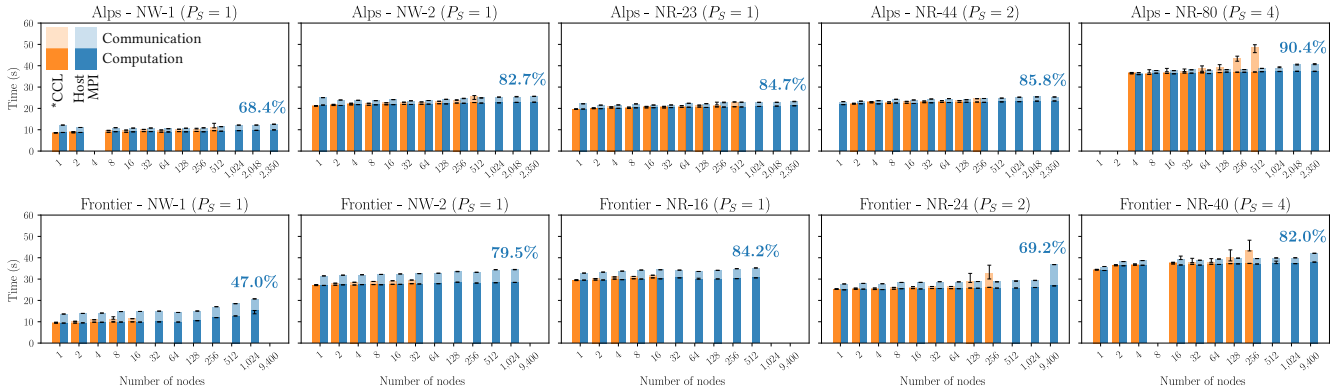


Figure 6: Weak scaling as a function of the number of energy points N_E. Reporting median runtimes for two different communication backends: *CCL (orange) and Host MPI (blue). The runtimes are further divided into two parts: communication (lighter shade) and computation (darker shade). We compute median values of ~10 measurements and 95% CI (error bars) for a single SCBA iteration of the NW-1 and NW-2 nanowires (Alps and Frontier), as well as for the NR-{23, 44, 80} (Alps) and NR-{16, 24, 40} (Frontier) configurations.

7.2 Weak Scaling

We perform weak scaling experiments on the number of energy points N_E. Except for the FFT, the workload scales linearly with N_E. We use one (MPI) rank per GPU (Alps) or GCD (Frontier). The number of energy points per rank for NW-1, NW-2, NR-16, and NR-23 is shown in Table 4. Spatial-domain decomposition is used only for NR-{24, 44}, with $P_S = 2$, and NR-{40, 80}, with $P_S = 4$ ranks treating a single energy point. The computation, communication, and total runtimes per SCBA iteration with the memoizer enabled are presented in Fig. 6. The data points are median runtimes, and the error bars are 95% CI computed with bootstrapping [17]. We annotate the achieved parallel efficiency at the maximum number of nodes for every machine/device combination.

Our software supports different communication libraries: *CCL, GPU-aware MPI, and "host MPI" (copying data to the host). We show only *CCL and host MPI, as they performed the best. On Alps, the best network performance up to 256-512 nodes is achieved with NCCL, after which it exhibits instability and slowdowns, necessitating a switch to host MPI. On Frontier, we observe similar performance with RCCL, but it starts to become unstable earlier, at about 32 nodes. Communication induced by spatial-domain decomposition is intranode, and therefore handled with *CCL.

In all experiments, we achieve a flat scaling until 128 nodes. Afterward, due to FFT's computational cost and other delays resulting in compute time increases, the smaller NW-1 device loses scaling efficiency. The rest of the devices continue to scale well up to 1024 nodes or even the full scale. When running almost at the full scale of Alps, we observe significant spurious delays in compute kernels, more precisely in the system solvers, but only for a subset of the executed iterations. We attribute those to faulty GPUs, but we were unable to repeat the experiments due to limited resources.

In Table 6, we summarize our software's performance at almost the full scale of Frontier and Alps for the NR-{24, 40} and NR-{44-80} NRFETs, respectively. On the larger machine, Frontier, we surpass 1 Eflop/s FP64 sustained performance on 9,400 nodes, achieving with NR-40 57.0% and 86.5% of Frontier's Rpeak and Rmax (scaled for number of nodes).

Table 6: Large-scale simulations on Alps and Frontier.

Machine	Frontier		Alps	
Nodes	9,604		2,600	
Rmax [Pflop/s]	1,353.00		434.90	
Rpeak [Pflop/s]	2,055.72		574.84	
Device	NR-24	NR-40	NR-23	NR-44
P_S	2	4	1	2
Memoizer	✓	✓	✓	✓
Atoms	25,344	**42,240**	24,288	46,464
Total Energies	37,600	18,800	9,400	4,700
Nodes (#N)	9,400		2,350	
GCDs / GPUs	75,200		9,400	
Workload [Pflop]	37,978.933	48,252.738	7,833.885	8,686.874
Time per iteration [s]	36.789	**42.104**	23.286	25.353
Performance [Pflop/s]	1,032.345	**1,146.037**	336.420	342.637
Scaling efficiency [%]	69.2	82.0	84.7	85.8
Rmax (#N scaled) [%]	76.3 (80.0)	**84.7** (86.5)	77.4 (85.6)	78.8 (87.2)
Rpeak (#N scaled) [%]	50.2 (51.3)	55.7 (57.0)	58.5 (64.8)	59.6 (65.9)

8 Implications

We presented a new *ab-initio* quantum transport solver called QuaTrEx based on the atomistic non-equilibrium Green's function formalism and capable of modeling electron-electron interactions within the *GW* approximation in transistor structures with dimensions comparable to experimental devices [1]. Compared to the state of the art from 2024, we have increased the maximum achievable simulation workload, proportional to $O(N_E N_B N_{BS}^3)$, by a factor of almost 16 (18,800 vs. 14,400 for N_E, 40 vs. 16 for N_B, 3,408 vs. 2,016 for N_{BS}), while increasing the walltime by only 35% (42.1 s vs. 31.3 s), for a total improvement by a factor of almost 12. Thanks to computational, algorithmic, and programming innovations, our tool can handle unprecedentedly large atomic systems (up to 84,480 atoms) and achieve a sustained Eflop/s performance in FP64. In particular, it accounts for dissipative mechanisms that are expected to drastically impact the behavior of nano-transistors, e.g., NRFETs.

Although the current implementation includes electron-electron interactions only, other types of scattering, such as electron-phonon or electron-photon, can be readily integrated, making our package the reference in the technology computer-aided design (TCAD) of ultra-scaled components.

Nevertheless, some limitations persist. Handling realistically-sized devices consisting of more than 25,000 atoms requires the use of distributed-memory algorithms to solve for the Green's functions and screened Coulomb interactions. Consequently, the size of the energy grid cannot be as large as in the case of structure, e.g., NW-1 and NW-2, where multiple energies fit into a single GPU. For example, even at the full scale of Frontier, we could "only" simulate ~19-38×10^3 energies, which is at the lower bound of the necessary resolution. A number closer to 100,000 would be better.

We believe there are further memory-related innovations to be exploited that can bring us closer towards the goal of combining both realistically-sized devices and energy grids on the current generation of supercomputers. In this work, we focused on keeping data on the GPU memory to minimize data movement and maximize the compute potential. Our memory-handling scheme can, therefore, be further improved by making better use of the host memory. Also, the data that has to be stored and communicated to the energy convolutions can potentially be reduced by appropriate compression or lower-precision schemes that retain the required computational accuracy. Overall, we are planning to explore these optimization opportunities and bring our "nano-TCAD" tool QuaTrEx to the next level of size and accuracy.

Acknowledgment

This work was supported by the Swiss National Science Foundation (SNSF) under grant n° 209358 (QuaTrEx) and grant n° 205602 (NCCR MARVEL), and by the Platform for Advanced Scientific Computing in Switzerland (BoostQT). We acknowledge support from CSCS (projects c33, g34, g186, lp16, lp82). This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725 (project NEL107). We especially thank Maria Grazia Giuffreda, Verónica G. Melesse Vergara, and Rocco Meli.

References

[1] A. Agrawal et al. 2024. Silicon RibbonFET CMOS at 6nm Gate Length. In *2024 IEEE International Electron Devices Meeting (IEDM)*.
[2] Emilio Artacho et al. 2008. The SIESTA method; developments and applicability. *Journal of Physics: Condensed Matter* (Jan. 2008).
[3] Scott Atchley et al. 2023. Frontier: Exploring Exascale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23)*. ACM.
[4] Mauro Del Ben et al. 2020. Accelerating Large-Scale Excited-State GW Calculations on Leadership HPC Systems. In *Proc. of SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*.
[5] Wolf-Jürgen Beyn. 2012. An Integral Method for Solving Nonlinear Eigenvalue Problems. *Linear Algebra Appl.* (May 2012).
[6] R. Chris Bowen et al. 1995. Transmission resonances and zeros in multiband models. *Physical Review B* (July 1995).
[7] Mads Brandbyge et al. 2002. Density-functional method for nonequilibrium electron transport. *Phys. Rev. B* (March 2002).
[8] Sascha Brück et al. 2017. Efficient Algorithms for Large-Scale Quantum Transport Calculations. *The Journal of Chemical Physics* (Aug. 2017).
[9] Mauro Calderara et al. 2015. Pushing back the limit of ab-initio quantum transport simulations on hybrid supercomputers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Austin, Texas) *(SC '15)*. Association for Computing Machinery.
[10] Herbert B. Callen and Theodore A. Welton. 1951. Irreversibility and Generalized Noise. *Phys. Rev.* (July 1951).
[11] Stephen Cauley et al. 2011. Distributed non-equilibrium Green's function algorithms for the simulation of nanoelectronic devices with scattering. *Journal of Applied Physics* (Aug. 2011).
[12] Wei Chen and Alfredo Pasquarello. 2017. Accuracy of GW for calculating defect energy levels in solids. *Physical Review B* (July 2017).
[13] Lisandro Dalcin and Yao-Lung L. Fang. 2021. Mpi4py: Status Update After 12 Years of Development. *Computing in Science & Engineering* (July 2021).
[14] Leonard Deuschle et al. 2025. Electron-electron interactions in device simulation via nonequilibrium Green's functions and the GW approximation. *Phys. Rev. B* (May 2025).
[15] Leonard Deuschle et al. 2024. Towards Exascale Simulations of Nanoelectronic Devices in the GW Approximation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis* (Atlanta, GA, USA) *(SC '24)*. IEEE Press.
[16] docs.lumi supercomputer.eu. 2025. GPU nodes - LUMI-G.
[17] Bradley Efron. 1992. Bootstrap Methods: Another Look at the Jackknife. In *Breakthroughs in Statistics: Methodology and Distribution*, Samuel Kotz and Norman L. Johnson (Eds.). Springer.
[18] M. V. Fischetti. 2001. Long-range Coulomb interactions in small Si devices. Part II. Effective electron mobility in thin-oxide structures. *Journal of Applied Physics* (Jan. 2001).
[19] Luigi Fusco et al. 2024. Understanding Data Movement in Tightly Coupled Heterogeneous Systems: A Case Study with the Grace Hopper Superchip.
[20] Weiwei Gao et al. 2024. Efficient Full-Frequency GW Calculations Using a Lanczos Method. *Phys. Rev. Lett.* (March 2024).
[21] P Giannozzi et al. 2017. Advanced capabilities for materials modelling with QUANTUM ESPRESSO. *Journal of Physics: Condensed Matter* (Oct. 2017).
[22] Xavier Gonze et al. 2020. The Abinit project: Impact, environment and recent developments. *Comput. Phys. Commun.* (March 2020).
[23] Charles R. Harris et al. 2020. Array programming with NumPy. *Nature* (Sept. 2020).
[24] Lars Hedin. 1965. New Method for Calculating the One-Particle Green's Function with Application to the Electron-Gas Problem. *Physical Review* (Sept. 1965).
[25] Mark S. Hybertsen and Steven G. Louie. 1986. Electron correlation in semiconductors and insulators: Band gaps and quasiparticle energies. *Physical Review B* (Oct. 1986).
[26] Genshiro Kitagawa. 1977. An algorithm for solving the matrix equation $X = FXF^T + S$. *Internat. J. Control* (Jan. 1977).
[27] W. Kohn and L. J. Sham. 1965. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.* (Nov. 1965).
[28] G. Kresse and J. Hafner. 1993. Ab initio molecular dynamics for liquid metals. *Phys. Rev. B* (Jan. 1993).
[29] A. L. Kutepov. 2020. Self-Consistent GW Method: O(N) Algorithm for Polarizability and Self Energy. *Comput. Phys. Commun.* (Dec. 2020).
[30] Thomas D. Kühne et al. 2020. CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations. *The Journal of Chemical Physics* (May 2020).
[31] Roger Lake et al. 1997. Single and multiband modeling of quantum electron transport through layered semiconductor devices. *Journal of Applied Physics* (June 1997).
[32] Siu Kwan Lam, Antoine Pitrou and Stanley Seibert. 2015. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*.
[33] Peitao Liu et al. 2016. Cubic Scaling GW: Towards Fast Quasiparticle Calculations. *Phys. Rev. B* (Oct. 2016).
[34] Mathieu Luisier et al. 2006. Atomistic Simulation of Nanowires in the $sp^3d^5s^*$ Tight-Binding Formalism: From Boundary Conditions to Strain Calculations. *Physical Review B* (Nov. 2006).
[35] Vincent Maillou et al. 2025. Serinv: A Scalable Library for the Selected Inversion of Block-Tridiagonal with Arrowhead Matrices.
[36] Nicola Marzari and David Vanderbilt. 1997. Maximally localized generalized Wannier functions for composite energy bands. *Phys. Rev. B* (Nov. 1997).
[37] Jens Jørgen Mortensen et al. 2024. GPAW: An open Python package for electronic structure calculations. *The Journal of Chemical Physics* (March 2024).
[38] Ryosuke Okuta et al. 2017. CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*.
[39] G. Onida, L. Reining and A. Rubio. 2002. Many-body perturbation theory approaches to electronic excitations: density functional versus Green function methods. *Reviews of Modern Physics* (June 2002).
[40] John P. Perdew et al. 2009. Some Fundamental Issues in Ground-State Density Functional Theory: A Guide for the Perplexed. *Journal of Chemical Theory and*

Computation (April 2009).
[41] Dan Erik Petersen et al. 2009. A hybrid method for the parallel computation of Green's functions. *J. Comput. Phys.* (Aug. 2009).
[42] Giovanni Pizzi et al. 2020. Wannier90 as a Community Code: New Features and Applications. *Journal of Physics: Condensed Matter* (April 2020).
[43] Federico Poloni. 2020. Iterative and doubling algorithms for Riccati-type matrix equations: A comparative introduction. *GAMM-Mitteilungen* (Oct. 2020).
[44] Diana Y. Qiu, Felipe H. da Jornada and Steven G. Louie. 2013. Optical Spectrum of MoS_2: Many-Body Effects and Diversity of Exciton States. *Physical Review Letters* (Nov. 2013).
[45] M P Lopez Sancho et al. 1985. Highly Convergent Schemes for the Calculation of Bulk and Surface Green Functions. *Journal of Physics F: Metal Physics* (April 1985).
[46] Kristian S. Thygesen and Angel Rubio. 2007. Nonequilibrium GW approach to quantum transport in nano-scale contacts. *The Journal of Chemical Physics* (March 2007).
[47] top500.org. 2024. TOP500 November 2024.
[48] Pauli Virtanen et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* (Feb. 2020).
[49] Jan Wilhelm et al. 2018. Toward GW Calculations on Thousands of Atoms. *J. Phys. Chem. Lett.* (Jan. 2018).
[50] Wentiao Wu et al. 2024. Enabling 13K-Atom Excited-State GW Calculations via Low-Rank Approximations and HPC on the New Sunway Supercomputer. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*.
[51] Victor Wen-zhe Yu and Marco Govoni. 2022. GPU Acceleration of Large-Scale Full-Frequency GW Calculations. *Journal of Chemical Theory and Computation* (Aug. 2022).
[52] Alexandros Nikolaos Ziogas et al. 2019. A data-centric approach to extreme-scale ab initio dissipative quantum transport simulations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, Colorado) *(SC '19)*. Association for Computing Machinery.