SplitSolve: a Fast Solver for Wave Function Based Quantum Transport Simulations on Accelerators

M. Calderara, S. Brück, and M. Luisier
Integrated Systems Laboratory, ETH Zürich, Switzerland

***Abstract*—We present SplitSolve, a novel sparse solver dedicated to linear systems in ballistic quantum transport calculations. The proposed algorithm specifically addresses the need for higher performance in the innermost loop of the energy integration in *ab-initio* simulations on hybrid architectures. The computation of the open boundary condition is deserialized from the most time consuming preprocessing step of the linear solver. The implementation of the algorithm itself is based on algebraic primitives that perform close to peak performance on current accelerator platforms. Using SplitSolve it is possible to efficiently harness the computational performance of these devices for a specific class of sparse systems. Combined with an efficient eigenvalue solver for the open boundary conditions, CPUs and accelerators can be used in parallel, resulting in significant speedups and much increased resource usage compared to traditional CPU-based methods. We compare here our implementation of SplitSolve with MUMPS, a state-of-the-art sparse linear solver for CPUs, and demonstrate an overall speedup of 7.7x for a nanowire simulation.**

I. Introduction

The general trend to ever smaller and smaller semiconductor devices has highlighted the shortcoming of classical simulation methods, which miss important quantum mechanical effects. Even full-band and atomistic approaches based on empirical models might not be accurate enough in nanostructures with large surface-to-volume ratios or in heterostructures. *Ab-initio* methods such as those based on density-functional theory (DFT) are therefore becoming more and more popular to study transport phenomena at the nanoscale. However, they suffer from well-known band gap underestimations that advanced hybrid functionals can partly resolve and they are computationally much more demanding than their empirical counterparts.

Meanwhile the quest for higher and higher performance to be fitted within a restrictive power budget has led to the emergence of hybrid computing platforms that combine traditional CPU-based resources with accelerator cards such as `nvidia` GPUs or `intel` MICs. These hybrid computing systems provide high performance within a comparably small power envelope, but they also necessitate the adaptation of established algorithms or their replacement by new ones to be fully effective. Unlocking the accelerators' full performance potential, particularly for sparse problems, has proven to be a formidable challenge in both algorithm design and implementation. The development of novel parallel numerical methods that can take advantage of all available computational resources on hybrid platforms is however the key step to make the computationally intensive *ab-initio* methods a viable alternative to empirical quantum transport simulations.

II. Method

Quantum transport within the framework of the Quantum Transmitting Boundary Method [1] or Wave Function formalism [2] gives rise to linear systems of equations $Tx = b$ that exhibit a specific structure: the T matrix is usually block-tridiagonal, while the right hand side (RHS) b (injection vector) has non-zero entries only in the top and bottom block rows. The atomic structure and the corresponding Hamiltonian matrix of a Si nanowire are given in Figs. 1 and 2, respectively, as an example. General-purpose sparse linear solvers cannot leverage the specific features of the systems they solve. We have therefor developed SplitSolve, an algorithm that is optimized for the matrices occurring in quantum transport simulations. It clearly outperforms general sparse system packages at the task of solving the linear systems it was optimized for. The key properties of SplitSolve are:

1) Efficient preprocessing of the sparse system using only accelerators such as graphics processing units.

2) Interleaving the preprocessing on the accelerators with the computationally demanding calculation of the open boundary conditions on the now idle CPUs.

3) Use of a specific low-rank representation of the boundary self-energy Σ in combination with the Sherman-Morrison-Woodbury formula to reconstruct the full-system wave function.

A. The SplitSolve Algorithm

Given the previously mentioned specific structure of the right-hand-side b it is straightforward to realize that solving $Tx = b$ for x is equivalent to computing the first and last block columns of T^{-1}. The desired solution x can then be obtained by multiplying these columns with the non-zero entries of b. As shown in Fig. 3(a) and mentioned above the matrix $T = (E \cdot S - H - \Sigma^{RB})$ is block tridiagonal. Observe that Σ^{RB} is a term of lower rank. It can thus be expressed as the non-unique product of two lower rank matrices so that $\Sigma^{RB} = B \cdot C$. Using the Sherman-Morrison-Woodbury formula

$$(A + BC)^{-1} = A^{-1} - A^{-1}B(\mathbb{1} + CA^{-1}B)^{-1}CA^{-1}$$

we can reformulate the problem of solving for x as

$$\begin{aligned} x &= T^{-1}b = (A - BC)^{-1}b \\ &= (A^{-1} + A^{-1}B(\mathbb{1} - CA^{-1}B)^{-1}CA^{-1})b \\ &= A^{-1}b + A^{-1}B(\mathbb{1} - CA^{-1}B)^{-1}CA^{-1}b \end{aligned}$$

Let m be the number of rows in $A = E \cdot S - H$, n_1 the size of the upper left most diagonal block in A, and n_n the

978-1-4673-7860-4/15/$31.00 ©2015 IEEE

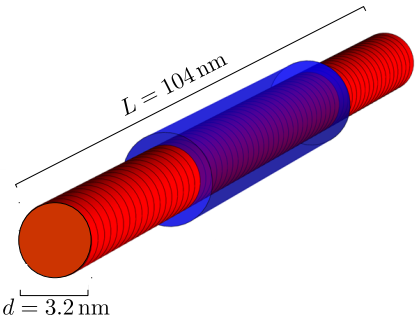

Fig. 1. Schematic view of the Silicon nanowire used to evaluate the performance of SplitSolve. At the given length and diameter the structure consists of 55488 atoms.

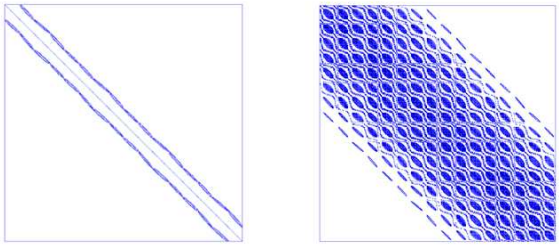

Fig. 2. Hamiltonian of the Si nanowire in Fig. 1 in a tight binding model (left) and in an *ab-initio* Gaussian orbital basis. In DFT each atom is represented by 12 orbitals in a 3SP basis set.

size of its lower right most diagonal block. We now choose a specific representation of $\Sigma^{RB} = B \cdot C$: B shall consist of the first n_1 and last n_n columns of the unity matrix of size m while C shall consist of the first n_1 and last n_n rows of Σ^{RB}, as illustrated in Fig. 3(a). If Q is the first and last block columns of A^{-1}, i.e. $Q := A^{-1}B$ and y the solution to the system without the boundary conditions, $y := A^{-1}b$, the previous equations reduce to

$$\begin{aligned} x &= T^{-1}b \\ &= y + Q(\mathbb{1} - CQ)^{-1}Cy \\ &= y + Qz. \end{aligned} \tag{1}$$

Equation 1 gives rise to the following algorithm to solve the full system $Tx = b$:

- Step 1: Solve $AQ = B$ for Q.
- Step 2: Solve $Ay = b$ for y.
- Step 3: Solve $Rz = (\mathbb{1} - CQ)z = Cy$ for z.
- Step 4: Compute $x = y + Qz$

B. Preprocessing Phase

Step 1 of the algorithm is independent of Σ^{RB} and b and can thus run in prefect parallelism to the solution of the eigenvalue problem required for the open boundary conditions. Therefore SplitSolve can be seen as consisting of a preprocessing phase (Step 1) and a postprocessing phase (Steps 2, 3, and 4). In order to efficiently obtain Q Algorithm 1 shown below can be used. It is a simplified variant of the LU decomposition for block tridiagonal systems. Let $A_{i,j}$ denote the i^{th} block row and j^{th} block column of A and let N be the number of diagonal blocks in A. Then Algorithm 1 gives $Q_{i,1:n_1} = A^{-1}_{i,1}$, the first block column of A^{-1}. The last block column of A can be computed with a similar recursion starting from the upper left diagonal block and proceeding downwards along the diagonal. Since they do not share any data dependency, the calculations of the first and last block columns of Q can be executed in parallel and make use of two accelerators. To leverage a higher number of accelerators and reduce the computational time a modified version of the SPIKE algorithm [3] has been implemented in SplitSolve. The concept is schematized in Fig. 4: a 1-D spatial decomposition breaks down the device under consideration into several partitions. Forming a binary tree these are subsequently merged together recursively at the

Algorithm 1: Block column inversion

$X_{n+1} \leftarrow 0$
$Q_0 \leftarrow -\mathbb{1}$
$i \leftarrow n$
▷ P_1 & P_2 in Fig. 4
while $i \geq 1$ **do**
  solve $(A_{i,i} - A_{i,i+1} \cdot X_{i+1}) \cdot X_i = A_{i,i-1}$ for X_i
  $i \leftarrow i - 1$
end while
▷ P_3 & P_4 in Fig. 4
while $i \leq N$ **do**
  $Q_i \leftarrow -X_i \cdot Q_{i-1}$
  $i \leftarrow i + 1$
end while

parent level when traversing the tree upwards. Each partition consists of two accelerators computing the first and last block columns of the inverse of the locally stored diagonal system. Upon completion siblings in the tree exchange the adjacent blocks of the inverse to form the inverse on the next upper level. The merging steps involve solving a linear system whose size can be reduced to the same order as the exchanged blocks, making this part computationally very efficient. The first and last block columns of the inverse of two merged partitions are obtained by multiplying the previous level's block inverse with the solution of the system. This approach incurs an overhead of one small system to be solved and four matrices to be multiplied for every merging step. Note that the number of merging steps logarithmically grows with the number of partitions: past experiences have shown that the proposed scheme is mostly practical for parallelization up to 16 accelerators. This is enough energy level parallelism to enable most realistic QT simulations to be run on the largest available hybrid super computing systems.

C. Postprocessing Phase

Since y can be computed as $y = Q \cdot b$, Step 2 is integrated into Step 4 by evaluating $x = Q \cdot (b' + z)$ where b' denotes the nonzero rows of b. In Step 3 C's sparsity pattern allows R to be constructed in an efficient way. The final system to solve is of comparably small size $n_1 + n_n$ due to the choice of C being

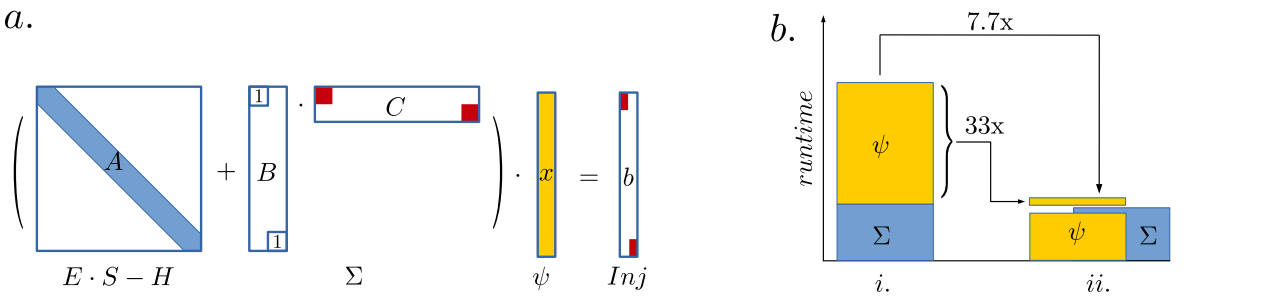

Fig. 3. (a) Sparsity pattern of the linear system resulting from the Wave Function formalism [2]. The boundary condition self-energy Σ can be represented as a non-unique product of low-rank matrices. The decomposition used in SplitSolve is shown here. (b) Comparison of runtimes of MUMPS ($i.$) and SplitSolve ($ii.$) (y-axis not to scale). Most of the time of SplitSolve is spent in its preprocessing phase. The speedup over MUMPS is typically 3-10× and 35× for the preprocessing and postprocessing phase, respectively. For the Si nanowire of Fig. 1 the overall speedup is 7.7×.

a $(n_1 + n_n) \times m$ matrix. The unknown x vector is obtained in Step 4 with one multiplication of a tall $m \times (n_1 + n_n)$ matrix with a thin $(n_1 + n_n) \times k$ one where k denotes the number of columns in b or the number of states injected into the simulation domain at the left and right contacts.

D. Numerical Stability

The stability of the block column inversion method in Algorithm 1 can be demonstrated to be equivalent to that of the well-known recursive Green's Function (RGF) technique [4]. The stability of the postprocessing phase depends on the stability of the Sherman-Morrison-Woodbury relation and the specific selection of representation for the update. Given our choice of B consisting of columns of the unity matrix, the proposed application of the Sherman-Morrison-Woodbury formula can be shown to be numerically stable. A detailed proof can be found in Ref. [5].

E. Roofline Analysis

Current as well as already announced future supercomputers featuring accelerators will favor computational power over execution flow flexibility and memory bandwidth. The evolution of accelerators seem to follow the same trend as the one of CPUs. Thus the "memory wall" already observed in conventional systems will over time deteriorate the performance of algorithms bound by memory bandwidth on accelerators as well.

SplitSolve involves only algebraic operations on dense matrices. These operations are well suited to accelerators given the outlined design compromises and the availability of highly efficient numerical libraries such as `MAGMA`, `CUBLAS`, or `intel MKL`. SplitSolve thus manages to employ a large fraction of the computational power provided by accelerators and succeeds in outperforming traditional sparse solvers even though a larger number of algebraic operations are required for its execution.

It can be shown that SplitSolve has a very high algebraic intensity (ratio of floating point operations performed to the number of bytes of operational data) and is therefore bound by compute power and not memory bandwidth. This is best illustrated by Fig. 5, where a roofline analysis of the algorithm is presented [6]. Based on the current position of SplitSolve in the graph, it can be expected that it will not be limited by

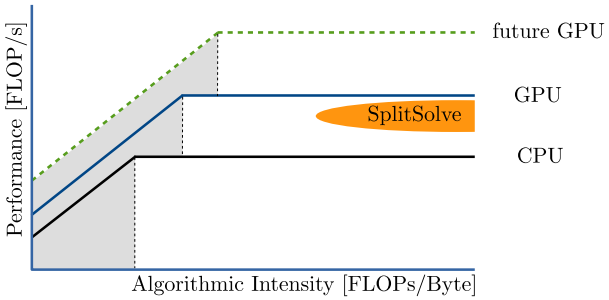

Fig. 5. Schematic presentation of a roofline analysis for SplitSolve. Shaded areas below the curve mark the domain where performance is limited by the memory bandwidth, algorithms in white areas are computationally bound. SplitSolve's performance and algorithmic intensity is shown for instances of realistic quantum transport simulations.

memory accesses but computational power for the foreseeable future. This property makes it a good candidate to benefit from the more even powerful next generation of accelerators having a higher ratio of compute power to memory bandwidth.

III. Results

To measure the performance of our implementation of SplitSolve we have prepared a Si nanowire structure with a diameter d=3.2 nm and a total length L=104 nm, as shown in Fig. 1. The structure is composed of 55488 atoms represented in a localized DFT basis set with 12 contracted Gaussian orbitals per atom. This results in a system size m=665856. The required Hamiltonian H and overlap S matrices are generated with the CP2K package [7]. For our measurements, SplitSolve has been implemented in OMEN [8], a general-purpose framework for atomistic quantum transport calculations. OMEN subsequently loads the Hamiltonian and overlap matrices from CP2K and performs device simulations based on them. As a testing platform the Cray-XK7 Titan at Oak Ridge National Laboratory has been chosen. For the eigenvalue problem involved in computing the open boundary conditions (OBCs), the FEAST technique [9] has been applied.
Utilizing 16 hybrid nodes for a single energy point the Schödinger equation with open boundary conditions is solved

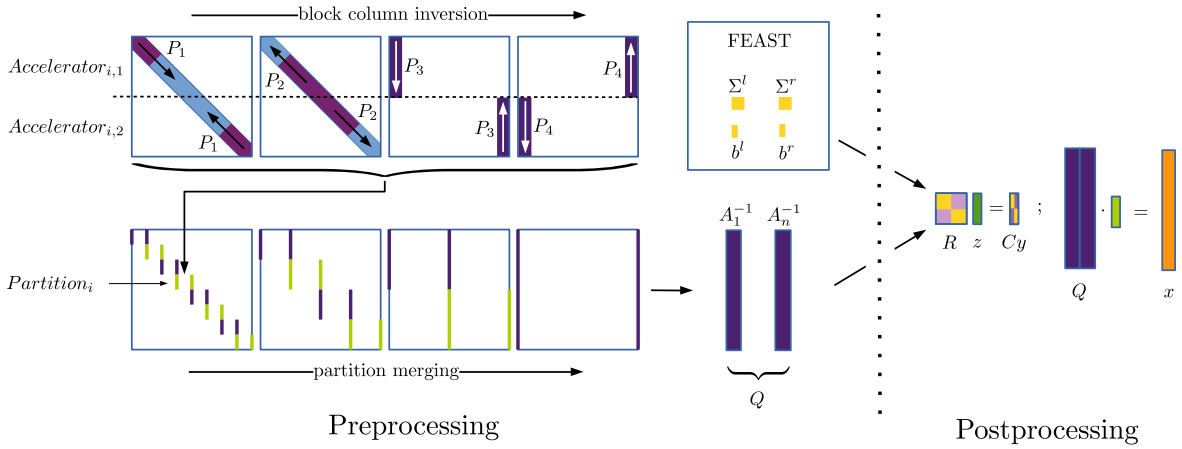

Fig. 4. Graphical overview of the SplitSolve algorithm. The system to solve is partitioned into 2^n horizontal partitions. Each partition is processed by two accelerators with perfect parallelism. To obtain $Q = [A_1^{-1}, A_n^{-1}]$, the partitions are merged recursively. Upon availability of the boundary conditions Σ and injection vectors b, postprocessing begins. The postprocessing phase mainly consists of solving a small system R and performing a matrix multiplication. It typically consumes one order of magnitude less time than the preprocessing phase.

in 128 sec with SplitSolve, while 883 sec are necessary with the MUMPS solver [10] (only CPUs). Hence, the obtained speed up is equal to 7.7×, as reported in Fig. 3 (b). The interleaving of the OBC calculation and of the Schrödinger equation solution as well as the efficiency of the SplitSolve algorithm both significantly contribute to the drastic reduction of the simulation time.

IV. Conclusion

We have developed an algorithm called SplitSolve targeting the linear systems of equations encountered in quantum transport simulations. By leveraging the special structure of the involved matrices and optimizing the algorithm to be a good fit for accelerators a significant speedup over MUMPS, a state-of-the-art sparse solver, has been achieved. It has been demonstrated that SplitSolve scales to multiple accelerators, enabling the investigation of very large device structures and allowing to make full use of tens of thousands of nodes in *ab-initio* quantum transport simulations. With SplitSolve, a deeper insight into the transport properties of nanostructures is expected. This holds true not only for the current generation of supercomputers, but also for the next ones with higher ratios of computational power to memory bandwidth.

Acknowledgment

This work was supported by SNF Grant No. PP00P2_133591, by the Hartmann Müller-Fonds on ETH-Research Grant ETH-34 12-1, by the Platform for Advanced Scientific Computing in Switzerland (ANSWERS), and by a grant from the Swiss National Supercomputing Centre under Project No. s579. This research also 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.

References

[1] Craig S. Lent and David J. Kirkner, The quantum transmitting boundary method," *Journal of Applied Physics*, vol. 67, no. 10, pp. 6353–6359, May 1990.

[2] Mathieu Luisier, Andreas Schenk, Wolfgang Fichtner, and Gerhard Klimeck, Atomistic simulation of nanowires in the $sp^3d^5s^*$ tight-binding formalism: From boundary conditions to strain calculations," *Phys. Rev. B*, vol. 74, pp. 205323, Nov 2006.

[3] Eric Polizzi and Ahmed H. Sameh, A parallel hybrid banded system solver: the SPIKE algorithm," *Parallel Computing*, vol. 32, no. 2, pp. 177 – 194, 2006, Parallel Matrix Algorithms and Applications (PMAA04).

[4] A. Svizhenko, A. M. Anantram, Govindan T. R., Biegel B., and Venugopal R., Two Dimensional Quantum Mechanical Modeling of Nanotransistors," *J. of Appl. Phys*, p. 2343, 2002.

[5] E. L. Yip, A Note on the Stability of Solving a Rank-p Modification of a Linear System by the Sherman-Morrison-Woodbury Formula," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 2, pp. 507–513, 1986.

[6] Samuel Williams, Andrew Waterman, and David Patterson, Roofline: An Insightful Visual Performance Model for Multicore Architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009.

[7] Joost VandeVondele and Juerg Hutter, Gaussian basis sets for accurate calculations on molecular systems in gas and condensed phases," *The Journal of Chemical Physics*, vol. 127, no. 11, pp. –, 2007.

[8] Mathieu Luisier, Timothy B. Boykin, Gerhard Klimeck, and Wolfgang Fichtner, Atomistic Nanoelectronic Device Engineering with Sustained Performances Up to 1.44 PFlop/s," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, 2011, SC '11, pp. 2:1–2:11, ACM.

[9] Eric Polizzi, Density-matrix-based algorithm for solving eigenvalue problems," *Phys. Rev. B*, vol. 79, pp. 115112, Mar 2009.

[10] P.R. Amestoy, I.S. Duff, and J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers," *Computer Methods in Applied Mechanics and Engineering*, vol. 184, no. 24, pp. 501 – 520, 2000.