Journal of Computational Electronics (2018) 17:698–706
https://doi.org/10.1007/s10825-018-1138-4

Fast, energy-efficient electronic structure simulations for multi-million atomic systems with GPU devices

Hoon Ryu[1] · **Oh-Kyoung Kwon**[1]

Published online: 26 February 2018

Abstract
We report that speed and energy efficiency of large-scale electronic structure simulations, which target realistically sized systems consisting of multi-million atoms, can be hugely improved with offload-computing using graphics processing unit (GPU) devices. For simulations of quantum dot devices that have ∼1.5 million atoms and are described by a $sp^3d^5s^*$ tight-binding (TB) model, a remarkable performance enhancement is obtained with asynchronous sharing of computing load in host CPUs and Tesla K40 devices. Compared to the case when only host CPUs are used, sparse matrix-vector multiplications, the core operation needed to solve Schrödinger equations, become remarkably faster leading ∼1.5× speed-up of end-to-end simulations with GPU devices. Asynchronous streams accelerate data-transfer reducing the associated overhead below ∼15% of the total wall-time. The speed and energy efficiency of TB simulations also turn out to be better with Tesla GPU devices than those obtained with Intel Xeon Phi Knights Corner (KNC) coprocessors, such that Tesla K40 GPU devices save ∼10% of the wall-time and ∼40% of the total energy consumed with KNC 7120 coprocessors for the target simulation. With technical details of offload-computing that can be also applied to accelerate other numerical problems involving large-scale sparse matrix operations, this work delivers practical information regarding the efficiency of GPU computing that has not been well covered for empirical modeling of large-scale electronic structures.

Keywords Electronic structure simulations · Tight-binding approach · Heterogeneous computing · High-performance computers

1 Introduction

Low-dimensional semiconductor structures have become of critical importance with recent revolutionary progress in lithographical technologies, [1–3] since they not only present the fundamental framework for designs of advanced electronic devices in the nanoscale regime, [4,5] but also open the possibility to find novel materials that may not be feasible with traditional bulk structures [6,7]. Solid understanding of electronic and material properties of nanoscale structures must involve a precise prediction with computer-aided simulations coupled to quantum physics, because otherwise there would be a huge loss in time and cost stemming from trial-and-error processes for determination of optimal conditions in sizes, shapes, material species, and so on. In particular, electronic structures have been a popular target of quantum simulations since they reveal the key information of low-dimensional structures, e.g., the band-structure which provides clues for material and electronic properties of nanoscale structures such as electrostatic profiles and carrier transport [8–10].

As semiconductor structures are downscaled in sizes, their characteristics become more sensitive to atomistic fluctuations such as interface roughness, non-ideal or unintended doping, alloy composition and strain relaxation, and associated quantum effects [11–14]. Also, experimentally realizable modern nanoscale structures usually consist of several million or more atoms though the core regions (e.g., nanowire channels or core-cells of quantum dot structures) have sizes of just a few nanometers (nms), because, in many

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10825-018-1138-4) contains supplementary material, which is available to authorized users.

✉ Hoon Ryu
elec1020@kisti.re.kr

[1] National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Daejeon 34141, Republic of Korea

cases, these regions are surrounded by or connected to large external layers that affect bias-dependent potential profiles and energy-level splitting in core parts [14–16]. The nearest-neighbor $sp^3d^5s^*$ tight-binding (TB) model, [17] which represents a single atom with a set of 10 (20 with spin-orbit couplings) bases and parameters that are fitted to reproduce band-structures of bulk materials, has been popularly used to study electronic structures of multi-million atomic systems, and has been verified with the capability to present strong connections to experimentally observed behaviors [5,8,18–21] or guidelines for advanced designs of nanoscale devices [22–24].

High-performance computing (HPC) clusters are essential to simulate electronic structures of multi-million atomic systems with empirical methods like a TB approach, because the size of Hamiltonian matrices, which needs to be diagonalized to solve associated Schrödinger equations, is proportional to the number of atoms residing in simulation domains. Nano-Electronic Modeling tool (NEMO), which is a well-known package of TB simulations [25,26], has established the framework of large-scale electronic structure simulations with traditional HPCs of multicore processors (CPUs), overcoming the structural size-limit ($< 10^3$ atoms) of simulations with density functional theory (DFT) [27]. However, general-purpose graphics processing unit (GPU) devices, which have attracted attention of HPC communities being utilized to accelerate expensive scientific computations, [28–31] have not been fully exploited yet for empirical modeling of large-scale electronic structures that involves multi-million atomic systems, although several pioneer works have reported remarkable performance enhancement with GPU devices for DFT simulations of electronic structures [30,31].

This work examines the utility of GPU devices for simulations of extremely large-scale electronic structures with a TB approach. Using our in-house code as a baseline that has been recently introduced as quantum simulation tool for advanced nanoscale devices (Q-AND), [32] we perform extensive code-refactoring with CUDA and benchmark the performance using Si:P quantum dots (QDs) as target devices, which are defined to be huge silicon (Si) layers encapsulating a phosphorus (P) atom and have been studied with a 10-band $sp^3d^5s^*$ TB model for designs of Si-based quantum computers [20,21]. In particular, we justify the utility of GPU devices by elaborating the following items: (1) strategical details of offload-computing and asynchronous data-transfer scheme with descriptions of major numerical approaches for solving large-scale electronic structures, (2) excellence of speed and scalability of end-to-end simulations with Nvidia Tesla K40 devices, compared to the performance measured in CPU-only nodes, and (3) benefits of simulations with Tesla K40 devices (a single K40 device has 2880 CUDA cores (745 MHz) and 12 GB memory [33]) in terms of computing time, data-transfer overhead and energy consumption, particularly against the case with their Intel counterpart, Xeon Phi Knights Corner (KNC) 7120 family (a single KNC 7120 coprocessor has 61 cores (1.24 GHz) and 16 GB memory [34]). Extending our latest study with KNC coprocessors [32] to the area of GPU devices, this work delivers practical information for the efficiency of offload-computing that has been rarely covered for empirical modeling of large-scale electronic structures and would be thus beneficial to researchers in the field of nanoelectronics who consider a code-migration toward heterogeneous computing systems supporting manycore devices via PCI-express (PCI-E) communications, which take $\sim$30% of top 100 HPC clusters in the world [35].

2 Methods

All the simulations of Si:P QD electronic structures considered in this work employ a 10-band $sp^3d^5s^*$ TB model, [27] which describes a single atom with a set of 10 orthogonal orbital bases (s, $3 \times p$, $5 \times d$, s^*) ignoring spin–orbit couplings. The size of a Hamiltonian matrix associated with a specific atomic structure, therefore, becomes 10 times larger than the number of atoms in the structure. As the TB approach we employed assumes nonzero coupling energies only among nearest-neighbor atoms, Hamiltonian matrices become sparse and thus are constructed with a Compressed Sparse Row (CSR) format [36]. Simulation domains are parallelized with a hybrid utilization of Massage Passing Interface (MPI) and OpenMP, so they are decomposed along the x-direction with MPI processes where the y-z plane allocated in a single MPI process is further decomposed with multiple threads. (We note Ref. [32] presents a detailed illustration of the domain decompositions with multicore CPUs.) Hamiltonian matrices are then decomposed in a row-wise manner as Fig. 1a shows. The Schrödinger equation solver, which numerically tackles normal eigenvalue problems in our case, is implemented with the well-known Lanczos method [37].

The two core mathematical operations, which involve MPI communications for parallel processing of Lanczos iterations, are multiplication of a sparse matrix and a vector, and dot-product of two vectors [37]. In particular, matrix-vector multiplications take a significant portion of the end-to-end computing time [32], becoming a hot spot that must be tackled to accelerate simulations with GPU devices. To achieve this mission with offload-computing, we decompose a block of the matrix belonging to a single MPI rank into two sub-blocks in a row-wise manner, where the ratio of decomposition is set as an input parameter in the unit of percentages. As illustrated in Fig. 1a, we then place one sub-block of the matrix in a single GPU device to process multiplications simultaneously in host and GPU devices. Additional over-

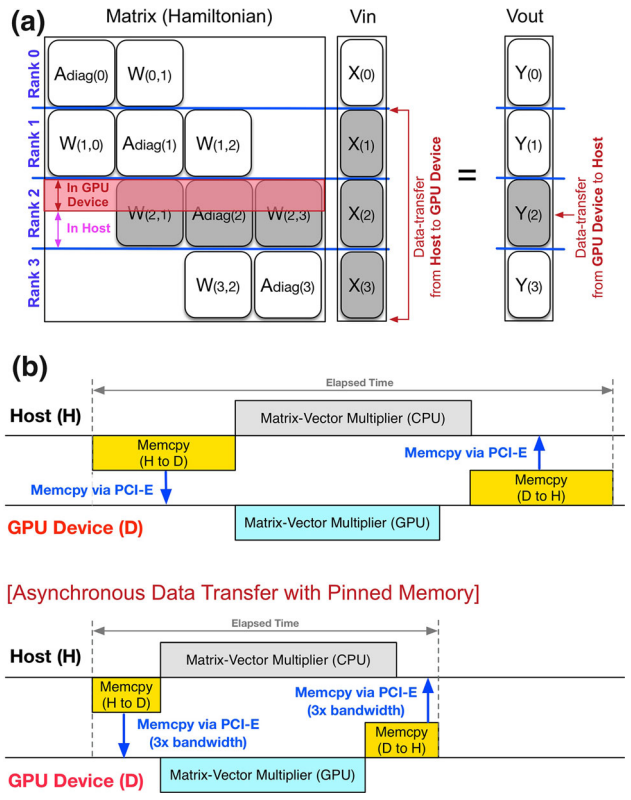

Fig. 1 A scheme of offload-computing for matrix-vector multiplications. **a** Each GPU device has a part of the block-matrix belonging to a single MPI process in host, which sends/receives input/output vectors to/from the associated GPU device. Each MPI process does not need to send the whole input vector (Vin), since multiplications in a MPI process can be done with only three block-vectors of Vin (one in itself, two in its neighbor processes) as our TB model assumes nearest-neighbor couplings. Upon the completion of multiplications, a GPU device just needs to send one block of the output vector (Vout) back to its associated MPI process. Host CPUs and GPU devices can thus perform multiplications simultaneously with no heavy overhead of data-transfer. **b** Utilization of pinned memory can further reduce the overhead of data-transfer, as it not only boosts the speed of data-transfer ($\sim 3\times$ speed-up against the case with pageable memory), but enables asynchronous data-transfer by which Vout can be transferred back to host while host is performing multiplications.

head, caused by transfer of input (Vin) and output (Vout) vectors between host and GPU devices, must be paid. The cost, however, may not be huge since we assume nearest-neighbor couplings, where each MPI process needs to send only three blocks of Vin (one in itself, two in adjacent MPI processes) to the associated GPU device, and each GPU device just needs to send one block of Vout back to the associated MPI process after multiplications are completed. This is clearly illustrated in Fig. 1a, where three blocks of Vin and one block of Vout are the targets of data-transfer between a MPI process (rank 2 of 4 processes) and its associated GPU device. The cost of data-transfer can be further reduced with aids of CUDA pinned (page-locked) memory, [38] which leads $\sim 3\times$ speed-up of data-transfer against the case with regular (pageable) memory. Utilization of CUDA pinned memory also enables data to be transferred via asynchronous streams, so Vout can be transferred from GPU devices to host while host is still performing multiplications as Fig. 1b shows.

Performance of sparse matrix-vector multiplications in GPU kernels is known to be limited by global memory access [39], which cannot be circumvented in our case as nonzero elements of large sparse matrices (Hamiltonian) have to be stored in global memory of GPU devices. We thus adopt a single-instruction, multiple-thread (SIMT) model [40] to increase the efficiency of global memory access. Figure 2 conceptually describes the advantage of performance that can be achieved with a SIMT model. Figure 2a shows first 4 rows of the Hamiltonian matrix for a [100] Si unitcell, where $h(i,j)$ is the nonzero element at (row i, column j), and (NZ k) denotes the element is the kth nonzero value of the matrix. Figure 2b gives a fundamental view of how a GPU kernel can access nonzero elements in first 4 rows with multiple threads. Here, a single thread takes a single row of the matrix, so the speed of global memory access for reading nonzero elements will be determined by the thread which takes the row that has the largest number of nonzero values. The optimized version of Fig. 2b is shown in Fig. 2c, which we use in the code. Here, multiple threads simultaneously access multiple nonzero elements, where each thread accesses a single element. A group of these threads, called as a CUDA WARP, [40] consists of contiguous threads, and a single WARP in Tesla K40 devices consists of 32 threads [41].

One of goals this work pursues is to assess the energy efficiency of GPU (Tesla K40) devices for TB simulations of electronic structures, via a comparison to data obtained with Intel Xeon Phi (KNC 7120) coprocessors. For this purpose, the real-time power-usage of a single computing node is monitored while simulations are being performed. The power used by host (CPUs and memory) is measured with Intel RAPL library, [42] where power used by KNC and Tesla devices are retrieved with Intel MICSMC utility and NVidia Management Library (NVML), [43,44], respectively.

3 Results and discussion

Performance of TB simulations is carefully investigated against a Si:P QD system that includes a single P atom at the center of a cuboid [100] Si layer. The Si layer consists of $30 \times 80 \times 80$ [100] unitcells that spatially correspond to a dimension of $\sim 16\,\text{nm} \times 43\,\text{nm} \times 43\,\text{nm}$. The target device has a total of 1.536 million (M) atoms and involves a 15.36×15.36 M Hamiltonian system matrix. With a maximal iteration of 10^4 and a convergence criterion of 10^{-8} electron-volt, the calculations are continued until either they reach the maximal iteration or find 10 lowest energy-levels in conduc-

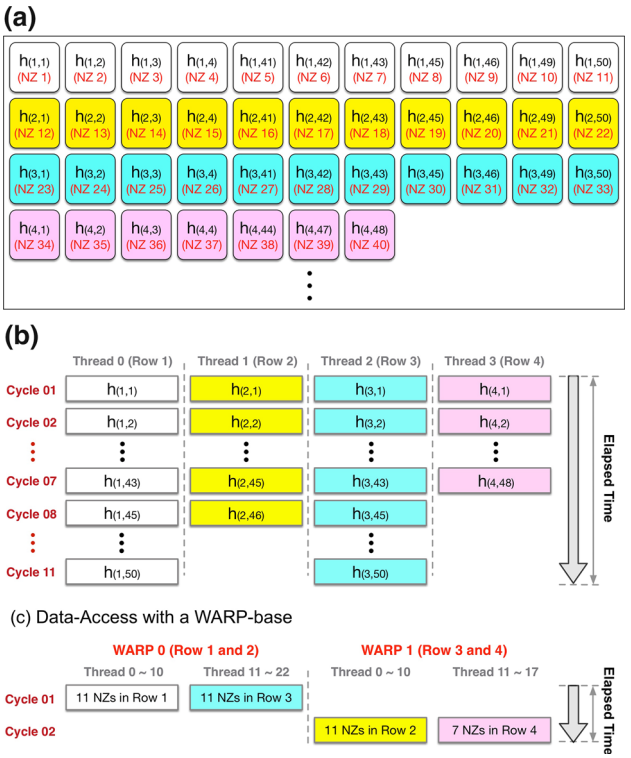

Fig. 2 A conceptual illustration of performance benefits that can be obtained with WARP-based parallelization of matrix-vector multiplications in GPU kernels. **a** First 4 rows of a 10-band TB Hamiltonian that describes a single [100] silicon unitcell, being stored in a CSR format. h(i,j) represents the nonzero element located in (row i, column j) of the matrix, where (NZ k) denotes the matrix element is the kth nonzero element of the matrix. **b** The pattern of access to global memory when purely thread-based parallelization is used. Here, a single thread accesses a single row of the matrix, so the speed of accessing the matrix stored in global memory is determined by the thread which takes the row that has the largest number of nonzero values. **c** With WARPs, the speed of multiplications can be improved since multiple threads (32 threads in this work) in a single WARP can access multiple nonzero values simultaneously.

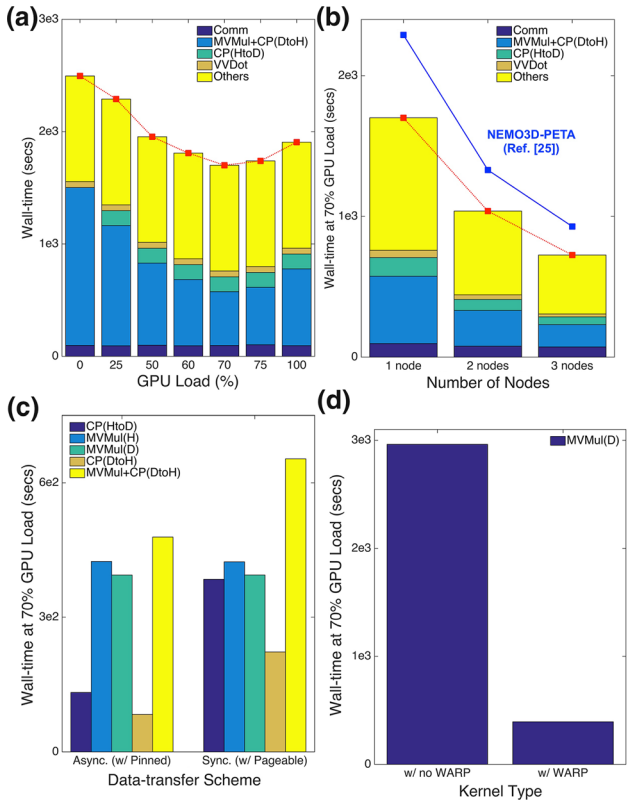

Fig. 3 Performance measured for the simulation of a 16 nm×43 nm×43 nm cuboid [100] Si:P QD that involves a ∼15 M × 15 M Hamiltonian matrix. **a** The wall-time of end-to-end simulations measured in a single node, shown as a function of computing load of matrix-vector multiplications imposed upon GPU devices. The wall-time is minimized when GPU devices perform 70% of multiplications. With a 70% load, the overall speed-up becomes ∼1.46× compared to the case when host performs all the multiplications, mainly due to ∼2.93× speed-up of the process of multiplications that includes data-transfer from GPU devices to host. **b** The strong scalability measured up to 3 nodes is nice regardless of the computing load of GPU devices (only the results at a 70% load are shown). Note that the performance here is also better than that of NEMO3D-PETA (Ref. [25]). **c** Utilization of pinned memory accelerates data-transfer and enables a simultaneous execution of data-transfer (GPUs → host) and multiplications (host). With a 70% load, the speed of multiplications (MVMul+CP(DtoH)) and data-transfer from host to GPU devices (CP(HtoD)) become ∼1.4× and ∼3×, respectively, compared to the case when pageable memory is used. **d** WARP-based parallelization (Fig. 2c) dramatically improves the speed of multiplications in GPU kernels, leading ∼7.5× speed-up against the case with purely thread-based parallelization (Fig. 2b)

tion band. All the workloads are tested with up to 3 computing nodes connected with an infinite-band 4 × FDR (56 Gbps) network, where each node has 2 Intel Xeon E5-2670 v2 (2.5 GHz) processors (10 cores per processor), 128 GB DDR3 SDRAM (1866 MHz), and 2 PCI-E (16×) devices (Tesla K40 or KNC 7120). Since one MPI process is mapped to a single PCI-E device as described in the previous section, all the simulations are performed with 2 MPI processes per node, where a single MPI process has 10 threads to maximally utilize the host computing resource. In each PCI-E device, multiplication is performed with a maximum number of threads that the device can support.

Figure 3a summarizes the performance of calculations measured in a single computing node with Tesla K40 devices. The wall-time for end-to-end simulations is shown as a function of computing load for multiplications imposed upon GPU devices (GPU Load) and is decomposed into the following 5 components: the time taken for (1) MPI Communication (Comm), (2) data-transfer from host to PCI-E devices (CP(HtoD)), (3) multiplication including data-transfer from PCI-E devices to host (MVMul+CP(DtoH)), (4) dot-product (VVDot), and (5) other operations (Others). As briefly mentioned in the Methodology section, the two operations involving MPI communications, i.e., matrix-vector multiplication and vector dot-product, take a

significant portion of the wall-time. In particular, the process of multiplications, which includes data-transfer from PCI-E devices to host, consumes ∼56% of the wall-time when only host CPUs are utilized. However, the time needed to complete the process of multiplications reduces as the GPU Load increases, and finally reaches its minimum when the GPU Load is ∼70%. A 70% GPU Load, consequently, minimizes the wall-time, causing ∼1.46× speed-up of end-to-end simulations compared to the case when host CPUs perform all the multiplications (GPU Load is zero), which is driven by ∼2.93× speed-up of the process of multiplications. Though other operations (Others) also take a non-negligible portion of the wall-time, they are performed in host CPUs and do not belong to the targets of GPU computing in this work. We note that more detailed discussion about the Others portion can be found in the supplementary document.

It is not easy to clarify with exact numbers why the wall-time is minimized at a ∼70% GPU Load, because the speed of sparse matrix operations is affected by many factors such as computing performance, memory bandwidth, and latency stemming from data-transfer via PCI-E lanes. The ideal roofline of the optimal GPU Load, however, can be roughly estimated with the known theoretical (peak) performance of host CPUs and GPU devices, by ignoring effects of memory access and data-transfer. Let us say that the peak performance (in the unit of FLoating point Operations Per Second (FLOPS)) of host CPUs and PCI-E devices are P_H and P_D, respectively. The ceiling of the optimal GPU Load (x) then can be calculated with a simple equation as follows:

$$\frac{x}{100-x} \simeq \frac{P_{\mathrm{D}}}{P_{\mathrm{H}}}, \tag{1}$$

which can be justified as long as we ignore effects of memory access and data-transfer, because the speed of overall multiplications then would be maximized when host CPUs and GPU devices complete computing operations at the same time. For double-precision floating point operations, a single computing node used in this work has P_{H} of $\sim 4 \times 10^{11}$ FLOPS (for 20 Xeon E5-2670 v2 cores), [45] and P_{D} of $\sim 2.86 \times 10^{12}$ FLOPS (for 2 Tesla K40 devices) [33]. x is therefore estimated as ∼87.7%, which is a bit larger than what we observe (∼70%) due to the assumed ignorance. Although the answer may not be strictly precise, Eq. (1) can still explain why the optimal load in this work is observed to be slightly larger than the one observed with Intel Xeon Phi KNC 7120 coprocessors (∼65%) for the workload and host computing environment identical to the ones of this work, [32], because the peak performance of a single Tesla K40 device ($\sim 1.43 \times 10^{12}$ FLOPS) is a bit larger than that of a single KNC 7120 coprocessor ($\sim 1.21 \times 10^{12}$ FLOPS) [34] for double-precision operations.

Figure 3b shows the strong scalability at a 70% GPU Load that is measured in up to 3 computing nodes (2 MPI processes per node). Here, we show that the scalability in multiple nodes is quite nice even though the workload involves offload-computing. The speed-up of end-to-end simulations obtained with 3 computing nodes becomes ∼ 2.34x against the case with a single node, showing a ∼78% scaling-ratio ($= 2.34 \div 3$) that is not very different to the one obtained with host CPUs only ($\sim 85\% = 2.6 \div 3$) in both this work and Ref. [32]. It is straightforward that utilization of more nodes reduces the computing load of both multiplication and dot-product that a single MPI process has. It is, however, worthwhile to emphasize the overhead of data-transfer is also mitigated with an increasing number of computing nodes (or MPI processes), because of the size reduction of Vin and Vout (See Fig. 1a) that have to be transferred between host and GPU devices during the process of multiplications. The performance at a 70% GPU Load also turns out to be generally better than that of NEMO3D-PETA package, [25] which adopts a MPI-based 3D domain decomposition for parallel processing of TB simulations for large-scale electronic structures. We note that, for simulations with NEMO3D-PETA, 2/4/6 MPI processes are used to decompose the simulation domain along the x-direction, as we did with our code. The y-z plane belonging to each MPI process, however, is decomposed with 10 MPI processes instead of 10 threads since NEMO3D-PETA only supports a MPI-based parallelization.

WARP-based parallelization of matrix-vector multiplications (Fig. 2c) and asynchronous data-transfer with pinned memory (Fig. 1b) can drive remarkable speed-up of end-to-end simulations. Figure 3c, which shows the time spent for the process of multiplications (MVMul+CP(DtoH)) and data-transfer at a 70% GPU Load, delivers the following messages: (1) the time spent for CP(HtoD) and CP(DtoH) supports the speed of data-transfer itself increases with pinned memory, showing ∼3× speed-up against the case when pageable memory is used. (2) The time spent for MVMul+CP(DtoH) shows the benefit of data-transfer via asynchronous streams. When pinned memory is used for data-transfer, we find ∼174 s are saved for MVMul+CP (DtoH) compared to the case when pageable memory is used. This time-saving is a bit larger than the time saved for CP(DtoH) (∼139 s), and the additional saving of 35 s is thus caused by the overlap of the two processes, i.e., multiplications in host and data-transfer from host to GPU devices, which is enabled by data-transfer via asynchronous streams. Note the speed of multiplications itself in host (MVMul(H)) and GPU kernels (MVMul(D)) is not much affected by the type of memory used for data-transfer. As Fig. 3d shows, multiplications in GPU kernels become drastically faster with WARP-based parallelization, where we find ∼7.5× speed-up at a 70% GPU Load, compared to the speed measured with thread-based parallelization (Fig. 2b).

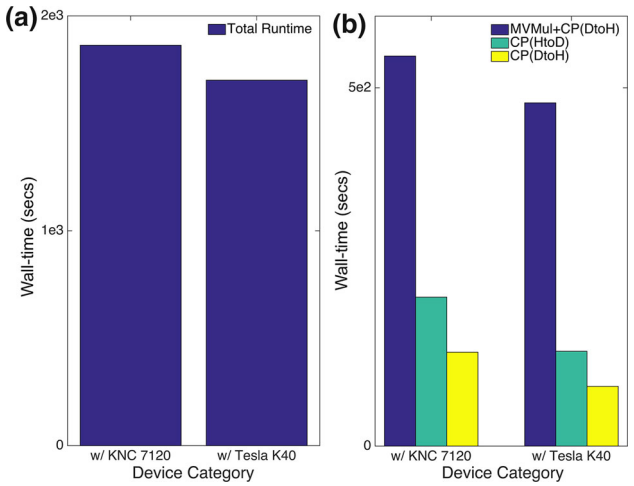

Fig. 4 Performance measured in a single computing node equipped with GPU devices or Xeon Phi coprocessors. 70 and 65% of multiplications are performed in Tesla K40 devices and KNC 7120 coprocessors, respectively, where it is known from Ref. [32] that a 65% load gives the best performance with KNC 7120 coprocessors in the same condition as the one used in this work. **a** The end-to-end simulation with Tesla K40 devices takes ∼10% less time (163 s) than the one with KNC 7120 coprocessors. **b** Most of the time-saving (163 s) driven with Tesla K40 devices comes from the process of multiplication (MVMul+CP(DtoH)) and transfer of input vectors (CP(HtoD)), which take less by 65 and 76 s, respectively, than the ones with KNC 7120 coprocessors. **b** Transfer of output vectors (CP(DtoH)) with Tesla devices takes less by 48 s than the one with KNC coprocessors, explaining ∼75% of the time saved for MVMul+CP(DtoH) (65 s). Remaining ∼25% of the time-saving (17 s) means the speed of multiplications itself is also improved with Tesla K40 devices

Figure 4a shows the wall-time of end-to-end simulations measured in a single computing node that has either Tesla K40 devices or KNC 7120 coprocessors. 70 and 65% of multiplications are performed in GPU devices and Xeon Phi coprocessors, respectively, where 65% is known to be the optimal load for KNC 7120 coprocessors in the same environment as the one used in this work [32]. With Tesla K40 devices, the wall-time is measured to be ∼1700 s, which turns out to be ∼10% smaller than the wall-time measured in a single node with KNC coprocessors (∼1863 s). Figure 4b shows the times that are spent for the process of multiplications (including data-transfer from PCI-E devices to host) and data-transfer, where the labels of MVMul+CP(DtoH), CP(HtoD) and CP(DtoH), are identical to the ones defined in Fig. 3. Here, the time-saving driven by Tesla K40 devices (163 s) mostly comes from CP(HtoD) and MVMul+CP(DtoH), which take less by 76 and 65 s, respectively, than the times taken with KNC coprocessors. Time-saving of CP(HtoD) (48 s) and CP(DtoH) (76 s) is due to the increased speed of data-transfer that comes from utilization of the asynchronous stream through pinned memory (Fig. 1b). The time saved for MVMul (17 s) implies the speed of multiplications itself improves with Tesla K40 devices, although the multiplication processes is also optimized in KNC coprocessors with Initial Many Core Instructions (IMCI), [46] which support single-instruction, multiple-data (SIMD) vectorization with 512-bit registers. It should be noted our code also transfers data through the asynchronous stream in KNC coprocessors [47]. Another important point that must be clarified is that the pattern of convergence and the accuracy of eigenvalues do not prominently depend on the type of PCI-E devices. The fairness in the performance comparison with different PCI-E devices can be supported by Table S1 in the supplementary document, which shows the total number of converged eigenvalues, magnitudes of converged eigenvalues, and iteration numbers at which eigenvalues are converged in computing environments with CPU-only, CPU+K40 devices, and CPU+KNC coprocessors.

So far, we have used the speed (wall-time) as the only indicator to assess the performance of TB simulations for large-scale electronic structures. Another indicator that has been widely agreed to be important to discuss the performance of scientific computing in HPC clusters, however, is the energy efficiency that is defined as the rate of computation performed for every watt (W) of power consumed (in the unit of FLOPS/W). As Tesla GPU devices are considered to be good solutions for energy-efficient computing, they are popularly adopted by clusters in Green500 sites, [48] where computing systems in Top500 sites citeR33 are ranked in terms of energy efficiency that is normally quantified with LINPACK benchmark [49]. The official energy efficiency known for Tesla K40 devices ($\sim 6 \times 10^9$ FLOPS/W) is nice, [33] particularly compared to the one of KNC 7120 coprocessors ($\sim 4 \times 10^9$ FLOPS/W) [34]. But this official comparison may not be directly applicable to our target problem, since LINPACK benchmark consists of computing-bound problems involving full matrix operations, while the core of TB electronic structure simulations involve large-scale sparse matrix operations whose performance may not be computing-bound [39].

To investigate the energy efficiency of TB simulations, the power-usage of host and the two PCI-E devices is retrieved as a function of elapsed time. Figure 5a, b shows the results measured, while the simulation is being performed in a single computing node with KNC 7120 coprocessors (with a 65% load) and Tesla K40 devices (with a 70% load), respectively. Here, the pattern of power-usage is similar in both cases such that it roughly consists of the following 3 steps: (1) power-usage increases during the setup of the domain and associated Hamiltonian matrix, (2) oscillates rapidly during the process of Lanczos iterations, and (3) reduces back as the workload is finished. Our results indicate the power-usage of host (CPUs and memory) does not show clear dependency on PCI-E devices during the whole runtime, but the power-usage of KNC 7120 coprocessors is much larger than that

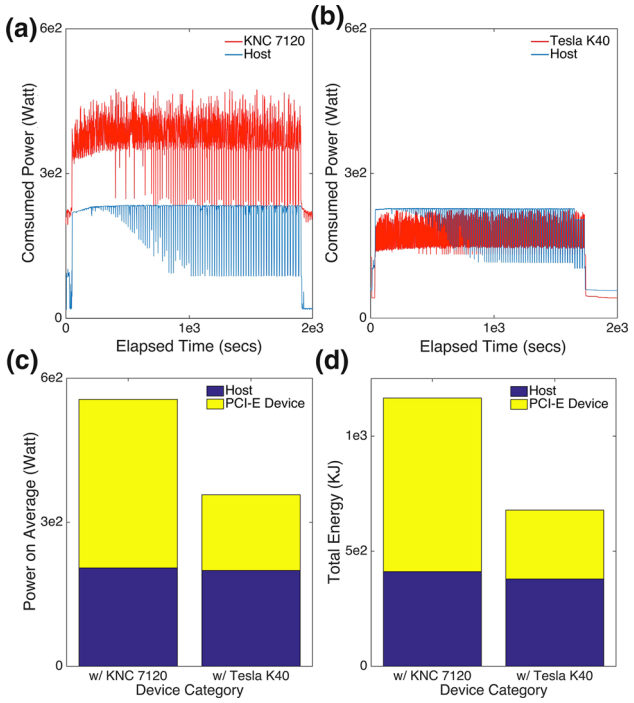

Fig. 5 Power-usage and energy consumption associated with the target simulation. **a** Power-usage of a single computing node with KNC 7120 coprocessors at a 65% load, and **b** with Tesla K40 devices at a 70% load of multiplications are shown as a function of elapsed time. During the runtime of simulations, power-usage of host is not much affected by PCI-E devices. Tesla K40 devices, however, use much less power than KNC 7120 coprocessors. **c** Real-time power-usage is time-averaged, where we find Tesla K40 devices use ∼158 W to perform 70% of multiplications, while KNC 7120 coprocessors use ∼351 W for 65% of multiplications. Host uses ∼200 W in both cases. **d** With Tesla K40 devices, a computing node consumes ∼58% of the total energy consumed for the end-to-end simulation with KNC 7120 coprocessors. Focusing on the energy consumption of PCI-E devices, we find KNC coprocessors consume ∼754 KJ to perform 65% of multiplications, while Tesla devices consume ∼300 KJ to perform 70%

of Tesla K40 devices. Figure 5c, which shows time-averaged power-usage of host and PCI-E devices, reveals KNC 7120 coprocessors and Tesla K40 devices use ∼351 and ∼158 W, respectively, while host uses ∼200 W in both cases. The energy consumed by the simulation, which can be obtained by multiplying the time-averaged power-usage by the wall-time, is shown in Fig. 5d. When the simulation runs in a single computing node with KNC 7120 coprocessors, host and two PCI-E devices consume ∼411 kilojoule (KJ) and ∼754 KJ, respectively, while corresponding values become ∼380 and ∼300 KJ with Tesla K40 devices.

As addressed, the energy efficiency is defined as the rate of computation performed for power consumption of 1 W. If we introduce a new quantity, the total number of floating point operations that a single job processes (NF), the energy efficiency (FLOPS/W) can be extracted since FLOPS is effectively equivalent to NF divided by the time taken until the job is finished (wall-time). In particular, the energy efficiency of TB simulations (η) can be approximated with the following equation:

$$\eta = \frac{NP}{T_{\text{total}}} \times \frac{1}{W} = \frac{NP}{E_{\text{total}}}, \tag{2}$$

where E_{total} and T_{total} represent the total energy and wall-time consumed by a single job (or simulation), respectively. As discussed with the results shown in Fig. 5, the total energy consumed to simulate the electronic structure of a 16 nm ×43 nm ×43 nm cuboid [100] Si:P QD, is ∼1165 KJ in a single computing node with KNC 7120 coprocessors and ∼680 KJ in a node with Tesla K40 devices. As shown in Eq. (2), for the same workload, the energy efficiency is inversely proportional to the consumed energy. Consequently, by assuming the energy consumed by CPUs and memory can reasonably approximate the energy consumed by host computing resource, we find Tesla K40 devices increase the energy efficiency by a factor of ∼1.7 compared to the case when KNC 7120 coprocessors are used. As Tesla K40 devices and KNC 7120 coprocessors perform 70 and 65 % of total multiplication consuming 754 and 300 KJ, respectively, the energy efficiency of Tesla K40 devices ($\propto 70 \div 300$) becomes ∼2.7× against that of KNC 7120 coprocessors ($\propto 65 \div 711$). We note that the superiority of Tesla K40 devices retrieved here (2.7×) becomes more remarkable that the one (1.5×) obtained with the official efficiencies known for the two PCI-E devices [33,34].

4 Conclusion

We have investigated the practicality of general-purpose graphics processing unit (GPU) devices for empirical tight-binding (TB) simulations of extremely large-scale electronic structures, which target multi-million atomic systems and involve sparse Hamiltonian system matrices of 10^7 or larger degrees of freedom (DOFs). Major technical strategies used to exploit the strength of GPU-based offload-computing, which are data-transfer via asynchronous streams and WARP-based parallelization, have been explained in detail with short but clear descriptions of the numerical method employed to solve large-scale Schrödinger equations in parallel. The gain of performance obtained by offload-computing with Tesla K40 devices has been carefully analyzed for simulations of a phosphorus quantum dot encapsulated by large silicon (Si) layers (Si:P QD), which has 1.536 million (M) atoms involving a Hamiltonian matrix of 15.36 M DOFs.

The wall-time of end-to-end simulations fluctuates as the GPU portion of matrix-vector multiplications, which is

the core numerical operation of electronic structure calculations, is varied, and reaches its minimum when Tesla K40 devices perform ∼70% of multiplications so ∼1.46× speed-up is observed with respect to the wall-time measured in CPU-only nodes, which is mainly due to ∼2.93× speed-up of multiplications. Compared to the case when lntel Xeon Phi Knights Corner (KNC) 7120 coprocessors are used to offload matrix-vector multiplications similarly as what is done in this work, [32] Tesla K40 devices save ∼10% of the wall-time due to the speed-up of data-transfer, consuming ∼58% of the total energy to complete the target simulation. Although the purpose of this work is to present the details of technical approaches for the performance improvement of large-scale electronic structure simulations with GPU computing, it should be noted that Tesla GPU devices are not cost-competitive. We thus encourage readers to carefully examine the benefit that can be obtained even at the expense of additional costs, particularly before writing codes for GPU computing. Readers, who are interested in the related analysis for this work, may want to refer to Table S2 in the supplementary document.

Acknowledgements This work has been carried out as Intel Parallel Computing Centre (IPCC) project under the financial support from Intel Corporation, USA, and with the extensive utilization of KISTI Accelerator Testbed (KAT) computing resources supported by Korea Institute of Science and Technology Information. Hoon Ryu would like to appreciate Jeehye Sohn for all the invaluable support and encouragement for this research.

References

1. Ruess, F.J., Oberbeck, L., Simmons, M.Y., Goh, K.E.J., Hamilton, A.R., Hallam, T., Schofield, S.R., Curson, N.J., Clark, R.G.: Toward atomic-scale device fabrication in silicon using scanning probe microscopy. Nano Lett. **4**(10), 1969 (2004)
2. Cooil, S.P., Mazzola, F., Klemm, H.W., Peschel, G., Niu, Y.R., Zakharov, A.A., Simmons, M.Y., Schmidt, T., Evans, D.A., Miwa, J.A., Wells, J.W.: In situ patterning of ultrasharp dopant profiles in silicon. ACS Nano **11**(1683–1688), 2 (2017)
3. Manfrinato, V.R., Zhang, L., Su, D., Duan, H., Hobbs, R.G., Stach, E.A., Berggren, K.K.: Resolution limits of electron-beam lithography toward the atomic scale. Nano Lett. **13**(4), 1555 (2013)
4. Fiori, G., Bonaccorso, F., Iannaccone, G., Palacios, T., Neumaier, D., Seabaugh, A., Banerjee, S.K., Colombo, L.: Electronics based on two-dimensional materials. Nature Nanotechnol. **9**, 768 (2014)
5. Weber, B., Tan, Y.H.M., Mahapatra, S., Watson, T.F., Ryu, H., Rahman, R., Hollenberg, L.C.L., Klimeck, G., Simmons, M.Y.: Spin blockade and exchange in Coulomb-confined silicon double quantum dots. Nature Nanotechnol. **9**, 430 (2014)
6. Eaves, L.: Semiconductors: an empire of many dimensions. Nat. Mater. **5**, 775 (2006)
7. Teng, Z., Liu, C., Yana, X.: A CO monolayer: first-principles design of a new direct band-gap semiconductor with excellent mechanical properties. Nanoscale **9**, 5445 (2017)
8. Weber, B., Mahapatra, S., Ryu, H., Lee, S., Fuhrer, A., Reusch, T.C.G., Thompson, D.L., Lee, W.C.T., Klimeck, G., Hollenberg, L.C.L., Simmons, M.Y.: Ohmõs law survives to the atomic scale. Science **335**, 64 (2012)
9. Paul, A., Mehrotra, S., Klimeck, G., Luisier, M.: On the validity of the top of the barrier quantum transport model for ballistic nanowire MOSFETs. In: Proceedings of IEEE international workshop on computational electronics (IWCE), pp. 173–176 (2009). https://doi.org/10.1109/IWCE.2009.5091134
10. Ryu, H.: A multi-subband Monte Carlo study on dominance of scattering mechanisms over carrier transport in sub-10-nm Si nanowire FETs. Nanoscale Res. Lett. **11**(11), 36 (2016)
11. Shinada, T., Okamoto, S., Kobayashi, T., Ohdomari, I.: Enhancing semiconductor device performance using ordered dopant arrays. Nature **437**, 1128 (2005)
12. Mlinar, V., Zunger, A.: Effect of atomic-scale randomness on the optical polarization of semiconductor quantum dots. Phys. Rev. B **79**, 115416 (2009)
13. Ahmed, S., Sundaresan, S., Ryu, H., Usman, M.: Multimillion-atom modeling of InAs/GaAs quantum dots: interplay of geometry, quantization, atomicity, strain, and linear and quadratic polarization fields. J. Comput. Electr. **14**, 543 (2015)
14. Tatebayashi, J., Nuntawong, N., Wong, P.S., Xin, Y.C., Lester, L.F., Huffaker, D.L.: Strain compensation technique in self-assembled InAs/GaAs quantum dots for applications to photonic devices. J. Phys. D Appl. Phys. **42**, 073002 (2009)
15. Schwarzenbach, W., Nguyen, B., Allibert, F., Girard, C., Maleville, C.: Ultra-thin body & buried oxide SOI substrate development and qualification for fully depleted SOI device with back bias capability. Solid-State Electr. **117**, 2 (2016)
16. Fuechsle, M., Mahapatra, S., Zwanenburg, F.A., Friesen, M., Eriksson, M.A., Simmons, M.: Spectroscopy of few-electron single-crystal silicon quantum dots. Nat. Nanotechnol. **5**, 502 (2010)
17. Jancu, J.M., Scholz, R., Beltram, F., Bassani, F.: Empirical $spds*$ tight-binding calculation for cubic semiconductors: general method and material parameters. Phys. Rev. B **57**, 6493 (1998)
18. Usman, M., Ryu, H., Woo, I., Ebert, D.S., Klimeck, G.: Moving toward nano-TCAD through multimillion-atom quantum-dot simulations matching experimental data. IEEE Trans. Nanotechnol. **8**(3), 330 (2009)
19. Ryu, H., Kim, J., Hong, K.H.: Atomistic study on dopant-distributions in relistically sized, highly P-doped Si nanowires. Nano Lett. **1**, 450 (2015)
20. Ryu, H., Lee, S., Fuechsle, M., Miwa, J.A., Mahapatra, S., Hollenberg, L.C.L., Simmons, M.Y., Klimeck, G.: A tight-binding study of single-atom transistors. Small **11**(3), 374 (2015)
21. Fuechsle, M., Miwa, J.A., Mahapatra, S., Ryu, H., Lee, S., Warschkow, O., Hollenberg, L.C.L., Klimeck, G., Simmons, M.Y.: A single-atom transistor. Nat. Nanotechonol. **7**, 242 (2012)
22. Ilatikhameneh, H., Klimeck, G., Appenzeller, J., Rahman, R.: Design rules for high performance tunnel transistors from 2D materials. IEEE J. Electron Device Soc. **5**, 260 (2016)
23. Mohiyaddin, F.A., Kalra, R., Laucht, A., Rahman, R., Klimeck, G., Morello, A.: Transport of spin qubits with donor chains under realistic experimental conditions. Phys. Rev. B **94**, 045314 (2016)
24. Agarwal, S., Klimeck, G., Luisier, M.: Leakage-reduction design concepts for low-power vertical tunneling field-effect transistors. IEEE Electron Device Lett. **31**(6), 621 (2010)
25. Lee, S., Ryu, H., Jiang, Z., Klimeck, G.: Million atom electronic structure and device calculations on peta-scale computers. In: Proceedings of IEEE international workshop on computational electronics (IWCE), pp. 1–4 (2009) https://doi.org/10.1109/IWCE.2009.5091117
26. Steiger, S., Povolotskyi, M., Park, H.H., Kubis, T., Klimeck, G.: NEMO5: a parallel multiscale nanoelectronics modeling tool. IEEE Trans. Nanotechnol. **10**(6), 1464 (2011)

27. Hasnip, P.J., Refson, K., Probert, M.I.J., Yates, J.R., Clark, S.J., Pickard, C.J.: Density functional theory in the solid state. Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. **372**(2011), 1 (2014)
28. Navarroa, C.A., Huangb, W., Dengb, Y.: Adaptive multi-GPU exchange Monte Carlo for the 3D random field Ising model. Comput. Phys. Commun. **205**, 48 (2016)
29. Hou, Q., Li, M., Zhou, Y., Cui, J., Cui, Z., Wang, J.: Molecular dynamics simulations with many-body potentials on multiple GPUs—the implementation, package and performance. Comput. Phys. Commun. **184**, 2091 (2013)
30. Maintz, S., Eck, B., Dronskowski, R.: Speeding up plane-wave electronic-structure calculations using graphics-processing units. Comput. Phys. Commun. **182**, 1421 (2011)
31. Harju, A., Siro, T., Canova, F.F., Hakala, S., Rantalaiho, T.: Computational physics on graphics processing units. Lect. Note Comput. Sci. **7782**, 3 (2012)
32. Ryu, H., Jeong, Y., Kang, J., Cho, K.: Time-efficient simulations of tight-binding electronic structures with Intel Xeon Phi^{TM} many-core processors. Comput. Phys. Commun. **209**, 79 (2016)
33. Nvidia Tesla K40 GPU accelerator. See http://www.nvidia.com/content/PDF/kepler/Tesla-K40-PCIe-Passive-Board-Spec-BD-06902-001_v05.pdf. Accessed 13 Feb 2018
34. Intel Xeon Phi Coprocessor 7120. See https://ark.intel.com/products/80555/Intel-Xeon-Phi-Coprocessor-7120A-16GB-1_238-GHz-61-core
35. Top500 HPC Sites (2016) See https://www.top500.org/lists/2016/11. Accessed 13 Feb 2018
36. Buluç, A., Fineman, J.T., Frigo M., Gilbert, J.R., Leiserson, C.E.: Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In: Proceedings of the annual symposium on parallelism in algorithms and architectures (SPAA), pp. 233–244 (2009) https://doi.org/10.1145/1583991.1584053
37. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Natl. Bur. Stand. **45**(4), 255(1950)
38. How to optimize data transfers in CUDA C/C++. See https://devblogs.nvidia.com/parallelforall/how-optimize-data-transfers-cuda-cc. Accessed 13 Feb 2018
39. Xu, S., Xue, W., Lin, H.: Performance modeling and optimization of sparse matrix-vector multiplication on NVIDIA CUDA platform. J. Supercomput. **63**, 710 (2011)
40. How to access global memory efficiently in CUDA C/C++ kernels. See https://devblogs.nvidia.com/parallelforall/how-access-global-memory-efficiently-cuda-c-kernels. Accessed 13 Feb 2018
41. Tuning CUDA applications for Kepler. See http://docs.nvidia.com/cuda/kepler-tuning-guide/#axzz4aiV2QGvP. Accessed 13 Feb 2018
42. Rountree, B., Ahn, D., de Supinski, B., Lowenthal, D., Schulz, M.: Beyond DVFS: A first look at performance under a hardware-enforced power bound. In: Proceedings of IEEE international parallel and distributed processing symposium workshops & PHD forum (IPDPSW), pp. 947–953 (2012) https://doi.org/10.1109/IPDPSW.2012.116
43. Reinders, J.: High Performance Parallelism Pearls, vol. 1, 1st edn. Morgan Kaufmann, Burlington (2014)
44. NVIDIA management library (NVML). See https://developer.nvidia.com/nvidia-management-library-nvml. Accessed 13 Feb 2018
45. Intel Xeon Processor E5-2670 v2. See http://ark.intel.com/products/75275. Accessed 13 Feb 2018
46. Details about intrinsic functions supporting intel IMCI. See https://software.intel.com/en-us/node/523389. Accessed 13 Feb 2018
47. Introduction to the heterogeneous streams library. See https://software.intel.com/en-us/articles/introduction-to-heterogeneous-streams-library. Accessed 13 Feb 2018
48. Green500 List (2016) See https://www.top500.org/green500/list/2016/11. Accessed 13 Feb 2018
49. The Linpack Benchmark. See https://www.top500.org/project/linpack. Accessed 13 Feb 2018