

FourPhonon_GPU: A GPU-accelerated framework for calculating phonon scattering rates and thermal conductivity

Ziqi Guo,¹ Xiulin Ruan,^{1, a)} and Guang Lin^{2, b)}

¹⁾*School of Mechanical Engineering and the Birck Nanotechnology Center, Purdue University, West Lafayette, 47907, IN, USA*

²⁾*School of Mathematics, Purdue University, West Lafayette, 47907, IN, USA*

(Dated: 2 October 2025)

Accurately predicting phonon scattering is crucial for understanding thermal transport properties. However, the computational cost of such calculations, especially for four-phonon scattering, can often be more prohibitive when large number of phonon branches and scattering processes are involved. In this work, we present FourPhonon_GPU, a GPU-accelerated framework for three-phonon and four-phonon scattering rate calculations based on the FourPhonon package. By leveraging OpenACC and adopting a heterogeneous CPU–GPU computing strategy, we efficiently offload massive, parallelizable tasks to the GPU while using the CPU for process enumeration and control-heavy operations. Our approach achieves over $25\times$ acceleration for the scattering rate computation step and over $10\times$ total runtime speedup without sacrificing accuracy. Benchmarking on various GPU architectures confirms the method’s scalability and highlights the importance of aligning parallelization strategies with hardware capabilities. This work provides an efficient and accurate computational tool for phonon transport modeling and opens pathways for accelerated materials discovery.

^{a)}Electronic mail: ruan@purdue.edu

^{b)}Electronic mail: guanglin@purdue.edu

I. INTRODUCTION

Thermal conductivity is a fundamental material property that plays a critical role in a wide range of applications, including thermal management in electronic devices^{1,2}, thermoelectric energy conversion^{3,4}, etc. Understanding and accurately predicting thermal conductivity is essential for optimizing material performance. The primary heat carriers in dielectrics and semiconductors are phonons. Phonon scattering rates, which limit the phonon mean free paths, are the central mechanism for determining thermal conductivity⁵. Accurately predicting phonon scattering rates is therefore key to predicting thermal transport behavior in materials.

Recent advances in first-principles calculations of phonon scattering coupled with Boltzmann transport equation (BTE) have enabled accurate, parameter-free predictions of lattice thermal conductivity. The theoretical foundation for phonon BTE was first laid by Peierls⁶. Later, Maradudin et al.⁷ developed a comprehensive framework for three-phonon (3ph) scattering. Broido et al.⁸ further combined *ab initio* force constants with these approaches, leading to robust first-principles predictions of thermal conductivity. It was believed for decades that 3ph scattering is adequate for describing thermal transport except at very high temperatures. Moreover, the general theory and computational method for four-phonon (4ph) scattering were lacking, preventing quantitative evaluation of its role. Feng and Ruan^{9,10} developed the general theory and computational method for 4ph scattering, demonstrating that 4ph processes can significantly impact the thermal conductivity of many materials, at elevated temperatures or even room temperature. Their theoretical predictions for boron arsenide (BAs) were later confirmed experimentally^{11–13}, proving that four-phonon scattering is a critical factor in thermal transport.

While first-principles calculations have greatly improved the fundamental understanding of thermal transport, they are extremely computationally expensive when including 4ph scattering. The high cost arises from the need to enumerate and compute a large number of phonon scattering processes, which scale as N^3 and N^4 for 3ph and 4ph scattering, respectively (N is the number of \mathbf{q} -points in the Brillouin zone). This leads to a dramatic increase in computational complexity. For example, for a silicon calculation using a $16 \times 16 \times 16$ \mathbf{q} -mesh (discretized grid in the reciprocal space), there are approximately 9.0×10^5 and 7.6×10^9 processes for 3ph and 4ph scattering, respectively, which could take over 7000 CPU hours to calculate.

To address the high or even prohibitive computational cost, several approaches have been explored, including machine learning surrogates^{14,15} and statistical sampling methods¹⁶. These tech-

niques have been successfully applied to estimate thermal and radiative properties of solids^{17–24}. Although these methods dramatically reduce computational cost, they achieve this by introducing approximations that sacrifice some degree of accuracy. As a result, they are particularly useful for applications where a certain level of error is acceptable. However, in scenarios that require rigorous and fully resolved calculations of every scattering process, such approximate methods are inadequate.

An alternative path for improving computational efficiency is leveraging hardware acceleration through Graphics Processing Units (GPUs). Originally developed for rendering computer graphics, GPUs have evolved as a powerful tool for scientific computing²⁵, machine learning^{26–28}, and high-performance simulations²⁹ due to their massively parallel architecture. With thousands of cores capable of executing millions of lightweight threads simultaneously, GPUs are ideally suited for workloads that involve a large number of independent operations³⁰. The advantage of GPUs has already been demonstrated across various fields of atomistic simulations. Popular first-principles simulation packages such as Abinit³¹ and VASP^{32,33} have incorporated GPU support, achieving orders-of-magnitude acceleration over traditional CPU implementations. In the context of phonon scattering calculations, Wei et al.³⁴ first identified the performance bottlenecks in ShengBTE³⁵ and offloaded the scattering rate calculations onto GPUs. Building on this foundation, Zhang et al.³⁶ developed the GPU_PBTE package, which employed a two-kernel strategy to further accelerate phonon scattering calculations. It is shown that under the relaxation time approximation (RTA), each phonon scattering process is fully decoupled and can be independently evaluated, making the problem highly suitable for GPU parallelization.

In this work, we develop a GPU-accelerated framework for both 3ph and 4ph phonon scattering calculations by combining CPU-based preprocessing with GPU-based large-scale parallel computing. Starting from the original FourPhonon³⁷ package, we adopt OpenACC to enable GPU acceleration with minimal code restructuring. However, due to the challenges of divergent branching, direct GPU offloading alone is insufficient to achieve optimal performance. To overcome these limitations, we propose a heterogeneous CPU–GPU computing scheme, in which the CPU enumerates momentum- and energy-conserving scattering processes, and the GPU efficiently evaluates the corresponding scattering rates in parallel (Fig. 1). In addition, we implement several optimization techniques to enhance parallelism and further improve computational efficiency. Through systematic comparisons, we demonstrate that our method preserves the full accuracy of the original calculations while achieving over $25\times$ acceleration for the scattering rate computa-

tion step and over $10\times$ total speedup. Our work establishes an efficient and scalable pathway for rigorous phonon scattering calculations on modern high-performance computing architectures.

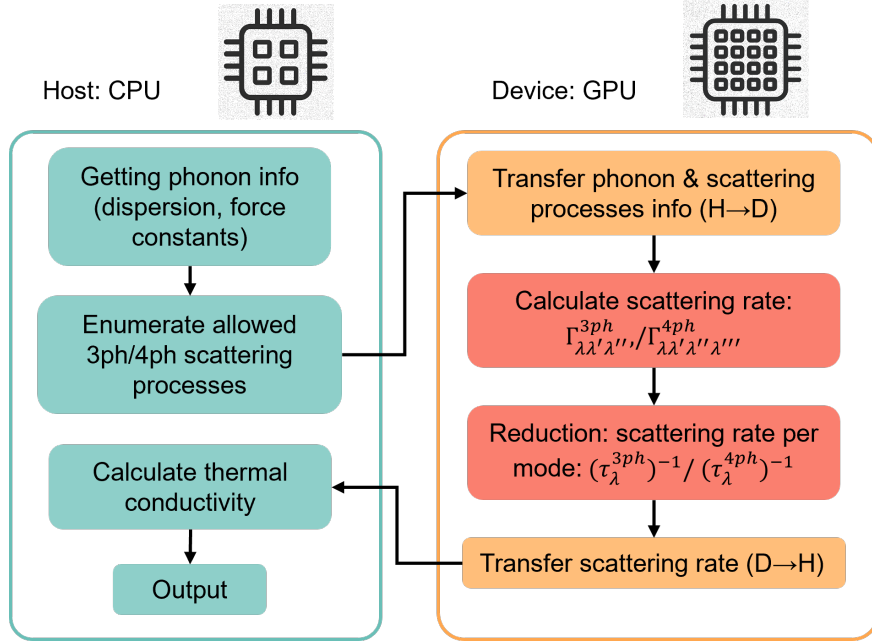


FIG. 1. **Workflow of CPU-GPU heterogeneous computing.**

II. METHODOLOGY

The original ShengBTE³⁵ and FourPhonon³⁷ packages are written in Fortran and parallelized using MPI, optimized for CPU-based high-performance computing environments. To enable GPU acceleration with minimal code restructuring and maximize cross-platform portability, we adopt OpenACC, a directive-based programming model designed to simplify the development of heterogeneous applications targeting both CPUs and GPUs. Since the phonon scattering processes are independent of each other, there are no loop-carried dependencies and the task is highly suitable for GPU calculation. This computational pattern aligns well with the Single Instruction, Multiple Threads (SIMT) execution model of GPUs, where thousands of lightweight threads execute the same instruction on different data. As a result, the calculation of scattering rates across different scattering channels can be efficiently mapped onto parallel GPU threads.

To illustrate our GPU acceleration strategy, we use the 3ph absorption process as an example, with detailed pseudocode provided in the Appendix. In the original CPU implementation (Algorithm 1 in the Appendix), the code loops over all phonon modes in a driver function, where each

iteration calls a subroutine to evaluate the scattering rate for an individual mode. This subroutine performs nested loops over all possible combinations of phonon wavevectors and branches. Energy conservation checks are conducted to filter out forbidden scattering processes before computing the weighted phase space (WP) and scattering rate (Γ). Due to the vast number of phonon combinations, this calculation is computationally expensive. Our first strategy is directly offloading the entire computation to the GPU (see Algorithm 2 in the Appendix). All necessary data is preloaded into GPU memory to avoid host-device data transfer overhead during runtime. After GPU computation, the results are transferred back to CPU memory, and GPU memory is released. We parallelize over all possible scattering processes and, beyond that, across multiple phonon modes simultaneously to enhance concurrency. This all-modes parallelization approach outperforms the mode-by-mode parallelization used in prior GPU implementations^{34,36} (see Algorithm 5), offering higher levels of parallelism and improved speedup while having higher GPU memory cost. Both of these methods are implemented, and a comparison is provided in the Results section.

However, directly applying OpenACC directives to existing MPI-based Fortran code is insufficient due to architectural differences between CPUs and GPUs. Several optimizations have been implemented to make the code GPU-compatible and efficient. First, we apply loop flattening using the collapse clause to combine nested loops and expose more parallelism within each phonon mode. Second, the loop order is rearranged to achieve memory coalescing, allowing consecutive threads to access contiguous memory locations, which improves memory access efficiency. Third, we inline the computations for the broadening factor (σ) and the matrix element (V_p) to avoid the overhead of function and subroutine calls on the GPU. Finally, the accumulation of the scattering rate and weighted phase space is handled using the reduction clause, which efficiently manages parallel updates and avoids race conditions, outperforming the use of atomic operations.

While this direct GPU-offloading approach preserves the original CPU workflow and achieves acceleration, it shows performance degradation due to divergent branching. Specifically, conditional statements used to check energy conservation introduce warp divergence, which undermines the efficiency of the SIMT execution. In our scenario, threads associated with forbidden scattering processes stall while others continue, leading to serialization and reduced throughput. Given the sparsity of allowed scattering processes, this divergence leads to significant underutilization of GPU resources (Fig. 2 upper figure). This problem has also been reported in prior work³⁶. Moreover, for 4ph scattering calculations, symmetry considerations restrict the iteration domain to a triangular region (see Fig. 3). However, OpenACC’s collapse directive requires rectangu-

lar iteration domains, preventing the use of symmetry conditions on GPUs and further impacting performance.

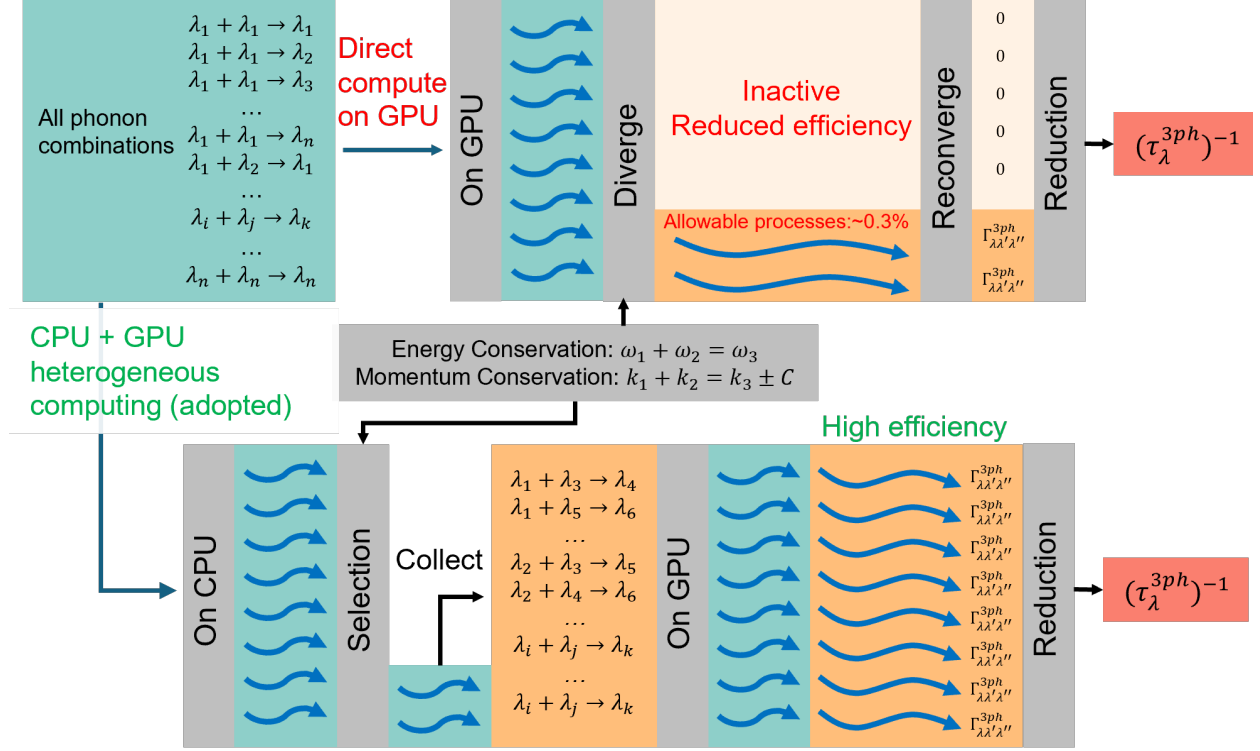


FIG. 2. **GPU-only vs. CPU+GPU heterogeneous computing.** The GPU-only approach suffers from warp divergence due to conditional branching when filtering forbidden phonon scattering processes, leading to reduced computational efficiency. In contrast, the CPU+GPU heterogeneous strategy first filters and prepares valid scattering processes on the CPU, allowing the GPU to execute the computation with higher parallel efficiency.

To address these limitations, we further propose a hybrid CPU–GPU computing framework that partitions the workload between CPU and GPU (Fig. 2 lower figure). In this scheme, the CPU is responsible for enumerating all symmetry-allowed and energy-conserving scattering processes, as described in Algorithm 3. Once the allowable scattering process indices are generated, they are transferred to the GPU. The GPU then performs the expensive scattering rate calculations in parallel, following Algorithm 4. By transferring only the relevant scattering events, we eliminate divergent branching on the GPU and ensure maximum occupancy of GPU resources during computation. Although this hybrid approach introduces additional CPU overhead for process enumeration, it significantly reduces the computational cost associated with scattering rate evaluations

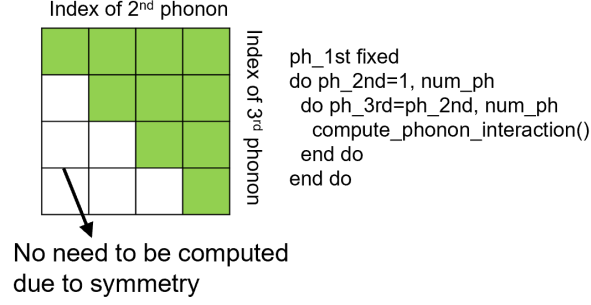


FIG. 3. **Triangular iteration region due to symmetry.**

and improves overall efficiency. In the Results section, we provide a detailed comparison between the CPU-only, GPU-only, and hybrid schemes, demonstrating the performance advantages of the proposed heterogeneous workflow. In our open source code, we adopted the hybrid approach for phonon scattering calculation.

III. RESULTS

The performance of both the original and GPU-accelerated codes is evaluated using silicon as a benchmark material. All calculations are performed at 300 K under RTA for both 3ph and 3ph+4ph scattering processes. The broadening factor is chosen as 0.1 for 3ph+4ph calculation following our previous work³⁷. Simulations are carried out on the Gilbreth cluster at Purdue University’s Rosen Center for Advanced Computing (RCAC). The software and hardware configurations are summarized in Table I.

TABLE I. Experimental hardware and software configurations

Item	Description
CPU	AMD EPYC 7543 32-Core Processor
GPU	Primary: NVIDIA A100 (80GB) GPU. Tests: NVIDIA A10, A30 GPUs
Compiler (GPU)	NVIDIA HPC Compiler (nvc 23.5-0)
CUDA Version	CUDA 12.6.0
Compiler (CPU)	Intel OneAPI Compilers 2024.2.1
MPI Version	Intel MPI 2021.13

For CPU-only calculations, parallel computing is performed with 32 CPU cores since the wall

time for serial computing is impractical. The reported time is CPU time, which represents the summed computational time over all cores. For GPU-accelerated calculations, the simulations are performed using one CPU core and one GPU, and the reported runtime is the sum of the CPU and GPU computation times. The relative difference in the calculated thermal conductivity between CPU-only and GPU-accelerated runs is less than 0.1%, which is primarily attributed to minor numerical variations between different compilers. These results confirm that GPU acceleration does not compromise the accuracy of the original code.

We first analyze the acceleration achieved by our CPU-GPU heterogeneous computing implementation. Figure 4 shows a comparison of the total computational cost between the original CPU-based method and the CPU-GPU hybrid version across different \mathbf{q} -mesh densities. For all tested \mathbf{q} -meshes, we observe a consistent acceleration of over $10\times$ for both 3ph and 3ph+4ph scattering calculations. Note that this computational time includes not only the phonon scattering step but also other computational overheads that are inherently performed on the CPU, such as the calculation of harmonic properties, phonon phase space, and the post-processing steps. If we only consider the computational cost of the CPU+GPU phonon scattering calculation step, we observe even more substantial speedups of over $18\times$ and over $25\times$ for 3ph and 4ph scattering rate calculation, which is shown in the insets of Fig 4. We are expecting to see an even higher acceleration rate if we set the broadening factor to unity for the 3ph+4ph calculation. These results clearly demonstrate the effectiveness of our GPU acceleration strategy.

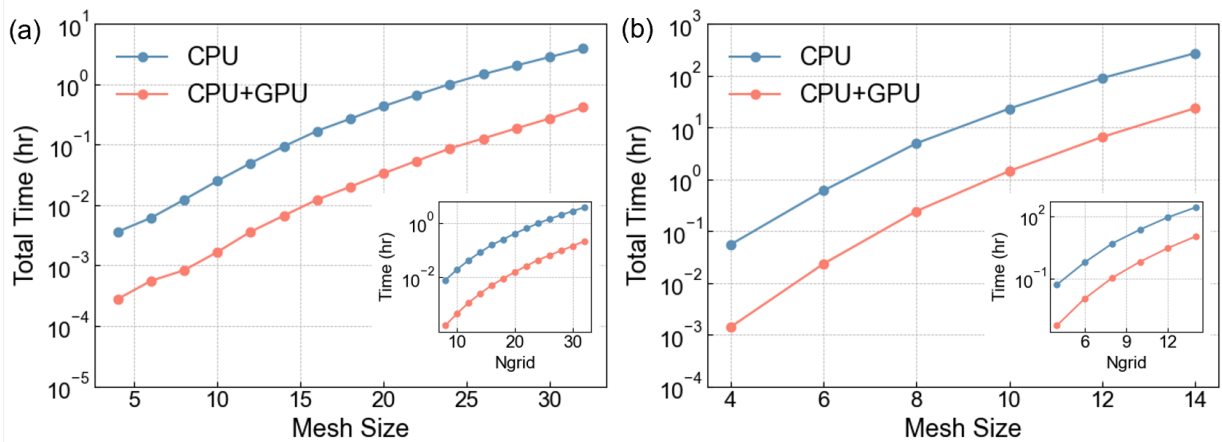


FIG. 4. **Comparison of total computational cost between CPU-only and CPU-GPU hybrid implementations across different \mathbf{q} -mesh sizes.** (a) 3ph scattering, (b) 3ph+4ph scattering. The insets show the isolated computational cost of the 3ph and 4ph scattering step alone.

We then compare two GPU parallelization strategies for phonon scattering calculations: mode-by-mode parallelization (where each phonon mode is processed independently) and all-modes parallelization (where all scattering processes are computed collectively). Figure 5(a) shows the comparison for representative cases. Note that for 4ph scattering, we choose a smaller \mathbf{q} -mesh size to illustrate the large computational cost, but the trend would be similar for large mesh size. We observe that all-modes parallelization leads to additional acceleration of 21% and 9% for the 3ph scattering case with a $32 \times 32 \times 32$ \mathbf{q} -mesh and 3ph+4ph with $10 \times 10 \times 10$ \mathbf{q} -mesh, respectively. This performance gain is due to two factors: (1) enhanced parallelism enabled by concurrent execution of all scattering processes, and (2) reduced overhead in transferring data between CPU and GPU memory by avoiding frequent host-device memory traffic. However, this gain comes with a trade-off. As shown in Fig. 5(b), preloading all scattering processes into GPU memory significantly increases the GPU memory cost. For dense \mathbf{q} -meshes, this demand may exceed available GPU memory. This is the case for the 3ph+4ph calculation at a $16 \times 16 \times 16$ mesh, which could not be completed using the all-modes strategy due to excessive memory usage (>80 GB). To address this, we fall back to the mode-by-mode parallelization approach for the $16 \times 16 \times 16$ case. While the computational efficiency is reduced, it still enables a notable acceleration of approximately $7\times$ compared to the original CPU-based implementation (Fig. 5(c)). This trade-off between memory usage and speed highlights the importance of selecting an appropriate parallelization strategy based on problem size and hardware constraints.

We further compared our CPU-GPU hybrid method with a GPU-only implementation, as illustrated in Fig. 6. For the 3ph scattering calculation with a $32 \times 32 \times 32$ \mathbf{q} -mesh, while both approaches are faster than the CPU-only baseline, the CPU-GPU hybrid approach achieves a $5\times$ speedup over the GPU-only method. Moreover, for the 3ph+4ph calculation using a \mathbf{q} -mesh of $10 \times 10 \times 10$, the GPU-only implementation is about twice as slow as the CPU-only baseline. This degradation is primarily due to the sparsity of the 4ph scattering matrix, which results in severe warp divergence and significantly reduces parallel efficiency. These results further demonstrate the effectiveness of our heterogeneous computing strategy by using the GPU for highly parallel workloads while using CPU for irregular, branching-heavy operations.

Finally, we evaluated the performance of our method on different GPU architectures. Since FourPhonon relies on double-precision arithmetic (FP64), the performance is strongly influenced by the available FP64 floating-point operations per second (FLOPS) on the GPU. Specifically, we tested three NVIDIA GPUs: A100, A30, and A10. The key specifications of these GPUs, includ-

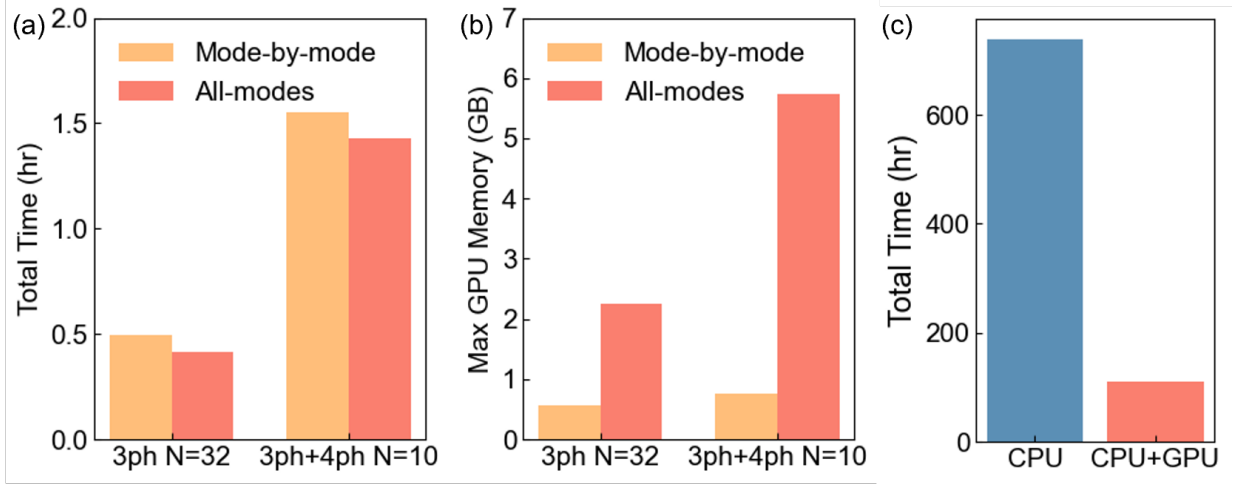


FIG. 5. **Comparison between all-modes and mode-by-mode parallelization strategies.** (a) Computational cost and (b) GPU memory usage for 3ph and 3ph+4ph scattering calculations using a \mathbf{q} -mesh of $32 \times 32 \times 32$ and $10 \times 10 \times 10$, respectively. (c) Computational cost for 3ph+4ph scattering with a $16 \times 16 \times 16$ \mathbf{q} -mesh, comparing CPU-only and CPU-GPU with mode-by-mode parallelization implementations.

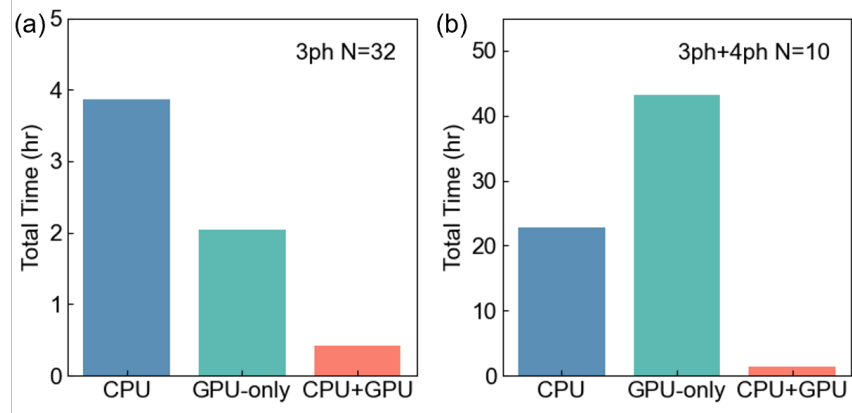


FIG. 6. **Comparison between CPU-GPU hybrid and GPU-only implementations.** (a) 3ph scattering with \mathbf{q} -mesh of $32 \times 32 \times 32$, (b) 3ph+4ph scattering with \mathbf{q} -mesh of $10 \times 10 \times 10$.

ing memory capacity and peak FLOPS for both single-precision (FP32) and FP64 operations, are summarized in Table II. To isolate GPU performance, we measured only the execution time of the GPU kernel, excluding CPU-side overheads. We observe that A100 > A30 > A10 in terms of computational speed for our double-precision workloads (Fig. 7), which is consistent with the FP64 FLOPS ranking. Since the A10 is optimized for FP32 workloads and has significantly lower FP64 performance, it is less suitable for scientific computing applications and shows a substan-

tial performance drop in our task. Additionally, the A100’s larger memory capacity provides an advantage in dense \mathbf{q} -mesh scenarios. For example, in the case of a $14 \times 14 \times 14$ \mathbf{q} -mesh with all-modes parallelization strategy, the total GPU memory requirement will be approximately 63 GB. This exceeds the memory capacity of the A10 and A30, which therefore will have to fall back to mode-by-mode parallelization, sacrificing performance to stay within hardware limitations. These results highlight the importance of selecting GPU hardware that matches the computational precision and memory demands of the targeted simulation workload.

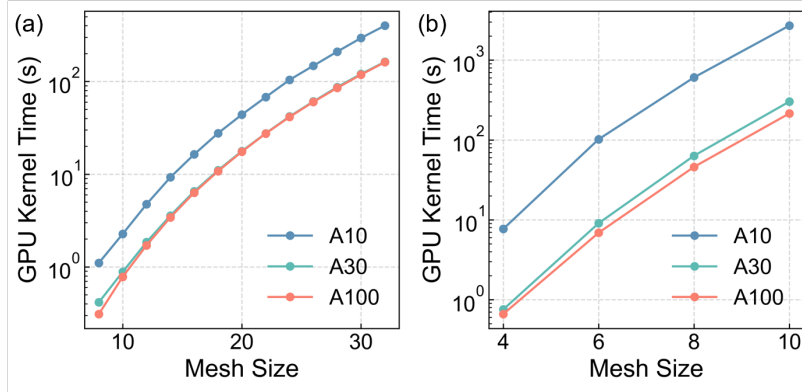


FIG. 7. **Comparison of GPU kernel time on different NVIDIA GPUS (A10, A30, A100).** (a) 3ph scattering, (b) 4ph scattering.

TABLE II. Specifications of selected NVIDIA GPUs used in this work.

GPU Model	FP32 Performance (TFLOPS)	FP64 Performance (TFLOPS)	Memory (GB)
A100	19.5	9.7	80
A30	10.3	5.2	24
A10	31.2	0.39	24

There are several directions for further improving the performance. Currently, our method is limited to the RTA due to the high memory cost and the overhead associated with data transfer between CPU and GPU memory. Future work could involve extending the framework to support iterative solvers. Besides, combining the GPU with the sampling-estimation-based approaches¹⁶ would further reduce both time and memory cost. In addition, the enumeration step on the CPU and the parallel computing step on the GPU are executed sequentially. Introducing asynchronous operations to overlap the CPU and GPU calculations could further reduce total runtime. Lastly,

while our implementation focuses on double-precision arithmetic for accuracy, it is worth noting that many modern GPUs, particularly those optimized for machine learning workloads, have significantly higher performance in single-precision computations. Exploring the use of mixed-precision or single-precision approaches, where appropriate, could yield additional performance gains without compromising accuracy for certain tasks.

In conclusion, we developed `FourPhonon_GPU`, a GPU-accelerated framework for phonon scattering calculations using a CPU–GPU heterogeneous computing strategy. By offloading highly parallel tasks to the GPU and retaining enumeration and control-heavy operations on the CPU, our approach achieves substantial speedups, with over $10\times$ improvement in total runtime and over $25\times$ acceleration in the scattering rate calculation step. With comprehensive benchmarking tests, we demonstrated the effectiveness of this framework and highlighted the importance of aligning algorithm design with hardware capabilities. This work provides an efficient computational tool for evaluating materials’ thermal properties and paves the way for accelerating materials discovery.

IV. APPENDIX

Algorithm1 Original CPU-based computation of three-phonon scattering process (absorption process)

```

1: Initialize global arrays: rate_scatt_plus( $N_{bands}, N_{list}$ ), WP3_plus_array( $N_{bands}, N_{list}$ )

2: for  $mm = 1$  to  $N_{bands} * N_{list}$  do

3:   // Subroutine: compute scattering for mode  $mm$ 

4:   Get  $i, ll$  from  $mm$ 

5:   Get  $q, \omega, v$  from  $i, ll$ 

6:   Initialize  $WP3_{plus} \leftarrow 0, \Gamma_{plus} \leftarrow 0$ 

7:   for  $j = 1$  to  $N_{bands}$  do

8:     for  $ii = 1$  to  $nptk$  do

9:       Get  $q', \omega', v'$  from  $j, ii$ 

10:       $f' \leftarrow BE(\omega')$ 

11:      for  $k = 1$  to  $N_{bands}$  do

12:         $q'' \leftarrow \text{modulo}(q + q', N_{grid})$ 

13:        Get  $ss$  from  $q''$ , get  $\omega'', v''$  from  $ss, k$ 

14:         $f'' \leftarrow BE(\omega'')$ 

15:         $\sigma \leftarrow \text{compute\_sigma}(v' - v'')$ 

16:        if  $|\omega + \omega' - \omega''| \leq 2\sigma$  then

17:           $WP \leftarrow \frac{(f' - f'') \cdot \exp\left[-\frac{(\omega + \omega' - \omega'')^2}{\sigma^2}\right]}{\sigma \sqrt{\pi} \cdot \omega \omega' \omega''}$ 

18:           $WP3_{plus} \leftarrow WP3_{plus} + WP$ 

19:           $V_p \leftarrow \text{compute\_Vp}(\dots)$ 

20:           $\Gamma_{plus} \leftarrow \Gamma_{plus} + WP \cdot |V_p|^2$ 

21:        end if

22:      end for

23:    end for

24:  end for

25:  rate_scatt_plus( $i, ll$ )  $\leftarrow \Gamma_{plus}$ 

26:  WP3_plus_array( $i, ll$ )  $\leftarrow WP3_{plus}$ 

27:  // End subroutine

28: end for

29: WP3_plus_array  $\leftarrow WP3\_plus\_array / nptk$ 

30: rate_scatt_plus  $\leftarrow \text{rate\_scatt\_plus} \cdot \text{const} / nptk$ 

```

Algorithm2 GPU-only-based computation of three-phonon scattering process (absorption process)

```
1: Initialize global arrays: rate_scatt_plus( $N_{bands}, N_{list}$ ), WP3_plus_array( $N_{bands}, N_{list}$ )

2: GPU: Copy data from host to device memory

3: // GPU: Launch parallel loop

4: for  $mm = 1$  to  $N_{bands} \times N_{list}$  do

5:   Get  $i, ll$  from  $mm$  ▷ Inlined original subroutine for higher efficiency

6:   Get  $q, \omega, v$  from  $i, ll$ 

7:   Initialize  $WP3_{plus} \leftarrow 0, \Gamma_{plus} \leftarrow 0$ 

8:   // GPU: Launch parallel loop, collapse(3), reduction over  $\Gamma_{plus}, WP3_{plus}$ 

9:   for  $ii = 1$  to  $nptk$  do

10:    for  $j = 1$  to  $N_{bands}$  do ▷ Reordered for memory coalescing

11:     for  $k = 1$  to  $N_{bands}$  do ▷ Rearranged for GPU loop collapsing

12:      Get  $q', \omega', v'$  from  $j, ii$ 

13:       $q'' \leftarrow \text{modulo}(q + q', N_{grid})$ 

14:      Get  $ss$  from  $q''$ , get  $\omega'', v''$  from  $ss, k$ 

15:       $f' \leftarrow \text{BE}(\omega'), f'' \leftarrow \text{BE}(\omega'')$ 

16:      // GPU: parallel loop, reduction over  $\sigma$ 

17:       $\sigma \leftarrow \text{inline\_compute\_sigma}(v' - v'')$  ▷ Avoid function call for GPU efficiency

18:      if  $|\omega + \omega' - \omega''| \leq 2\sigma$  then ▷ Divergent branching on GPU

19:        
$$WP \leftarrow \frac{(f' - f'') \cdot \exp\left[-\frac{(\omega + \omega' - \omega'')^2}{\sigma^2}\right]}{\sigma \sqrt{\pi} \cdot \omega \omega' \omega''}$$


20:         $WP3_{plus} \leftarrow WP3_{plus} + WP$ 

21:         $V_p \leftarrow \text{inline\_compute\_Vp}(\dots)$  ▷ Avoid function call for GPU efficiency

22:         $\Gamma_{plus} \leftarrow \Gamma_{plus} + WP \cdot |V_p|^2$ 

23:      end if

24:    end for

25:  end for

26: end for

27:  $\text{rate\_scatt\_plus}(i, ll) \leftarrow \Gamma_{plus}, WP3\_plus\_array(i, ll) \leftarrow WP3_{plus}$ 

28: end for

29:  $WP3\_plus\_array \leftarrow WP3\_plus\_array / nptk$ 

30:  $\text{rate\_scatt\_plus} \leftarrow \text{rate\_scatt\_plus} \cdot \text{const} / nptk$  ▷ Unit conversion

31: GPU: Copy data from device to host memory
```

Algorithm3 CPU-based precomputing of three-phonon scattering process indices (absorption process)

```
1: // CPU: Enumerate of all allowed scattering processes

2: Compute cumulative offset array  $N_{\text{accum\_plus}}$  from  $N_{\text{plus}}$ 

3: Initialize global arrays:  $\text{Ind2}_{\text{all}}(\text{sum}(N_{\text{plus}}))$ ,  $\text{Ind3}_{\text{all}}(\text{sum}(N_{\text{plus}}))$ 

4: for  $mm = 1$  to  $N_{\text{bands}} \times N_{\text{list}}$  do

5:   Get  $N_{\text{plus}}$  from  $mm$ 

6:   Get  $i, ll$  from  $mm$ 

7:   Get  $q, \omega, v$  from  $i, ll$ 

8:   Initialize  $N_{\text{plus\_count}} \leftarrow 0$ 

9:   Initialize arrays  $\text{Ind2}(N_{\text{plus}})$ ,  $\text{Ind3}(N_{\text{plus}})$ 

10:  for  $ii = 1$  to  $n_{\text{ptk}}$  do

11:    for  $j = 1$  to  $N_{\text{bands}}$  do ▷ Reordered for memory coalescing

12:      Get  $q', \omega', v'$  from  $j, ii$ 

13:      for  $k = 1$  to  $N_{\text{bands}}$  do

14:         $q'' \leftarrow \text{modulo}(q + q', N_{\text{grid}})$ 

15:        Get  $ss$  from  $q''$ , get  $\omega'', v''$  from  $ss, k$ 

16:         $\sigma \leftarrow \text{compute\_sigma}(v' - v'')$ 

17:        if  $|\omega + \omega' - \omega''| \leq 2\sigma$  then

18:           $N_{\text{plus\_count}} \leftarrow N_{\text{plus\_count}} + 1$ 

19:           $\text{Ind2}(N_{\text{plus\_count}}) \leftarrow (ii - 1) \cdot N_{\text{bands}} + j$  ▷ Detect possible scattering processes

20:           $\text{Ind3}(N_{\text{plus\_count}}) \leftarrow (ss - 1) \cdot N_{\text{bands}} + k$  ▷ Detect possible scattering processes

21:        end if

22:      end for

23:    end for

24:  end for

25:  Copy  $\text{Ind2}$  to  $\text{Ind2}_{\text{all}}$  at  $N_{\text{accum\_plus}}(mm) + 1$ 

26:  Copy  $\text{Ind3}$  to  $\text{Ind3}_{\text{all}}$  at  $N_{\text{accum\_plus}}(mm) + 1$ 

27: end for
```

Algorithm4 GPU-based computation of three-phonon scattering process using precomputed indices (absorption process)

```

1: GPU: Copy data from host to device memory

2: Compute cumulative offset array  $N_{\text{accum\_plus}}$  from  $N_{\text{plus}}$ 

3: // GPU: Launch parallel loop

4: for  $mm = 1$  to  $N_{\text{bands}} \times N_{\text{list}}$  do

5:   Get  $i, ll$  from  $mm$ 

6:   Get  $q, \omega$  from  $i, ll$ 

7:   Initialize  $\Gamma_{\text{plus}} \leftarrow 0, \text{WP3}_{\text{plus}} \leftarrow 0$ 

8:   // GPU: Launch parallel loop, reduction over  $\Gamma_{\text{plus}}, \text{WP3}_{\text{plus}}$ 

9:   for  $ind$  from  $N_{\text{accum\_plus}}(mm) + 1$  to  $N_{\text{accum\_plus}}(mm) + N_{\text{plus}}(mm)$  do

10:    Get phonon indices:  $(j, ii), (k, ss)$  from Ind2, Ind3

11:    Get  $q', \omega'$  from  $j, ii$ 

12:     $q'' \leftarrow \text{modulo}(q + q', N_{\text{grid}})$ 

13:    Get  $\omega''$  from  $k, ss$ 

14:    // GPU: parallel loop, reduction over  $\sigma$ 

15:    Compute  $\sigma \leftarrow \text{inline\_compute\_sigma}(v' - v'')$  ▷ Avoid function call for GPU efficiency

16:    Compute  $f' \leftarrow \text{BE}(\omega'), f'' \leftarrow \text{BE}(\omega'')$ 

17:     $\text{WP} \leftarrow \frac{(f' - f'') \cdot \exp\left[-\frac{(\omega + \omega' - \omega'')^2}{\sigma^2}\right]}{\sigma \sqrt{\pi} \cdot \omega \omega' \omega''}$ 

18:     $\text{WP3}_{\text{plus}} \leftarrow \text{WP3}_{\text{plus}} + \text{WP}$ 

19:     $V_p \leftarrow \text{inline\_compute\_Vp}(\dots)$  ▷ Avoid function call for GPU efficiency

20:     $\Gamma_{\text{plus}} \leftarrow \Gamma_{\text{plus}} + \text{WP} \cdot |V_p|^2$ 

21:  end for

22:   $\text{rate\_scatt\_plus}(i, ll) \leftarrow \Gamma_{\text{plus}}$ 

23:   $\text{WP3\_plus\_array}(i, ll) \leftarrow \text{WP3}_{\text{plus}}$ 

24: end for

25:  $\text{WP3\_plus\_array} \leftarrow \text{WP3\_plus\_array} / n_{\text{ptk}}$ 

26:  $\text{rate\_scatt\_plus} \leftarrow \text{rate\_scatt\_plus} \cdot \text{const} / n_{\text{ptk}}$  ▷ Unit conversion

27: GPU: Copy data from device to host memory

```

Algorithm5 GPU-based, mode-wise computation of three-phonon scattering process using precomputed indices (absorption process)

```

1: Compute cumulative offset array  $N_{\text{accum\_plus}}$  from  $N_{\text{plus}}$ 

2: for  $mm = 1$  to  $N_{\text{bands}} \times N_{\text{list}}$  do

3:   Get  $i, ll$  from  $mm$ 

4:   Get  $q, \omega$  from  $i, ll$ 

5:   Initialize  $\Gamma_{\text{plus}} \leftarrow 0, \text{WP3}_{\text{plus}} \leftarrow 0$ 

6:   Copy mode- $mm$  data in Ind2, Ind3 from host to device memory    ▷ Only copy mode- $mm$  for memory saving

7:   // GPU: Launch parallel loop, reduction over  $\Gamma_{\text{plus}}, \text{WP3}_{\text{plus}}$ 

8:   for  $ind$  from  $N_{\text{accum\_plus}}(mm) + 1$  to  $N_{\text{accum\_plus}}(mm) + N_{\text{plus}}(mm)$  do

9:     Get phonon indices:  $(j, ii), (k, ss)$  from Ind2, Ind3

10:    Get  $q', \omega'$  from  $j, ii$ 

11:     $q'' \leftarrow \text{modulo}(q + q', N_{\text{grid}})$ 

12:    Get  $\omega''$  from  $k, ss$ 

13:    // GPU: parallel loop, reduction over  $\sigma$ 

14:    Compute  $\sigma \leftarrow \text{inline\_compute\_sigma}(v' - v'')$     ▷ Avoid function call for GPU efficiency

15:    Compute  $f' \leftarrow \text{BE}(\omega'), f'' \leftarrow \text{BE}(\omega'')$ 

16:     $\text{WP} \leftarrow \frac{(f' - f'') \cdot \exp\left[-\frac{(\omega + \omega' - \omega'')^2}{\sigma^2}\right]}{\sigma \sqrt{\pi} \cdot \omega \omega' \omega''}$ 

17:     $\text{WP3}_{\text{plus}} \leftarrow \text{WP3}_{\text{plus}} + \text{WP}$ 

18:     $V_p \leftarrow \text{inline\_compute\_Vp}(\dots)$     ▷ Avoid function call for GPU efficiency

19:     $\Gamma_{\text{plus}} \leftarrow \Gamma_{\text{plus}} + \text{WP} \cdot |V_p|^2$ 

20:  end for

21:  Copy  $\Gamma_{\text{plus}}$  and  $\text{WP3}_{\text{plus}}$  of mode- $mm$  from device to host memory

22:   $\text{rate\_scatt\_plus}(i, ll) \leftarrow \Gamma_{\text{plus}}$ 

23:   $\text{WP3\_plus\_array}(i, ll) \leftarrow \text{WP3}_{\text{plus}}$ 

24: end for

25:  $\text{WP3\_plus\_array} \leftarrow \text{WP3\_plus\_array} / n_{\text{ptk}}$ 

26:  $\text{rate\_scatt\_plus} \leftarrow \text{rate\_scatt\_plus} \cdot \text{const} / n_{\text{ptk}}$     ▷ Unit conversion

```

DATA AVAILABILITY

The original results of the study are available from the corresponding authors upon reasonable request.

CODE AVAILABILITY

The work is incorporated as a new feature of the FourPhonon package and is available at <https://github.com/FourPhonon/FourPhonon>.

COMPETING INTERESTS

The authors declare no competing interests.

AUTHOR CONTRIBUTIONS

G.L., X.R. and Z.G. conceived the study. Z.G. designed and implemented the software, did the simulations, analyzed the results, and wrote the manuscript. G.L. and X.R. supervised the project. All authors contributed to discussions and revisions of the manuscript.

ACKNOWLEDGMENTS

Z.G. and X.R. acknowledge partial support from NSF Awards 2311848 and 2321301. Z.G is partly supported by the System Fellows 2024 Doctoral Fellowship provided by the Purdue Systems Collaboratory at Purdue University. Simulations were performed at the Rosen Center for Advanced Computing (RCAC) of Purdue University. G.L. acknowledges the National Science Foundation under grants DMS-2053746, DMS-2134209, ECCS-2328241, CBET-2347401, and OAC-2311848. The U.S. Department of Energy also supports this work through the Office of Science Advanced Scientific Computing Research program (DE-SC0023161) and the Office of Fusion Energy Sciences (DE-SC0024583).

REFERENCES

- ¹A. L. Moore and L. Shi, “Emerging challenges and materials for thermal management of electronics,” *Materials today* **17**, 163–174 (2014).
- ²S. He, Z. Ma, W. Deng, Z. Zhang, Z. Guo, W. Liu, and Z. Liu, “Experimental investigation on the start-up performance of a novel flat loop heat pipe with dual evaporators,” *Energy Reports* **8**, 7500–7507 (2022).
- ³M. S. Dresselhaus, G. Chen, M. Y. Tang, R. Yang, H. Lee, D. Wang, Z. Ren, J.-P. Fleurial, and P. Gogna, “New directions for low-dimensional thermoelectric materials,” *Advanced materials* **19**, 1043–1053 (2007).
- ⁴G. Chen, M. Dresselhaus, G. Dresselhaus, J.-P. Fleurial, and T. Caillat, “Recent developments in thermoelectric materials,” *International materials reviews* **48**, 45–66 (2003).
- ⁵J. Ziman, *Electrons and Phonons: The Theory of Transport Phenomena in Solids* (Oxford University Press, Northamptonshire, 2001).
- ⁶R. Peierls, “Zur kinetischen theorie der wärmeleitung in kristallen,” *Annalen der Physik* **395**, 1055–1101 (1929).
- ⁷A. Maradudin and A. Fein, “Scattering of neutrons by an anharmonic crystal,” *Phys. Rev.* **128**, 2589 (1962).
- ⁸D. A. Broido, M. Malorny, G. Birner, N. Mingo, and D. Stewart, “Intrinsic lattice thermal conductivity of semiconductors from first principles,” *Appl. Phys. Lett.* **91**, 231922 (2007).
- ⁹T. Feng and X. Ruan, “Quantum mechanical prediction of four-phonon scattering rates and reduced thermal conductivity of solids,” *Phys. Rev. B* **93**, 045202 (2016).
- ¹⁰T. Feng, L. Lindsay, and X. Ruan, “Four-phonon scattering significantly reduces intrinsic thermal conductivity of solids,” *Phys. Rev. B* **96**, 161201 (2017).
- ¹¹J. S. Kang, M. Li, H. Wu, H. Nguyen, and Y. Hu, “Experimental observation of high thermal conductivity in boron arsenide,” *Science* **361**, 575–578 (2018).
- ¹²F. Tian, B. Song, X. Chen, N. K. Ravichandran, Y. Lv, K. Chen, S. Sullivan, J. Kim, Y. Zhou, T.-H. Liu, *et al.*, “Unusual high thermal conductivity in boron arsenide bulk crystals,” *Science* **361**, 582–585 (2018).
- ¹³S. Li, Q. Zheng, Y. Lv, X. Liu, X. Wang, P. Y. Huang, D. G. Cahill, and B. Lv, “High thermal conductivity in cubic boron arsenide crystals,” *Science* **361**, 579–581 (2018).

- ¹⁴Z. Guo, P. Roy Chowdhury, Z. Han, Y. Sun, D. Feng, G. Lin, and X. Ruan, “Fast and accurate machine learning prediction of phonon scattering rates and lattice thermal conductivity,” *Npj Comput. Mater.* **9**, 95 (2023).
- ¹⁵Y. Srivastava and A. Jain, “Accelerating thermal conductivity prediction through machine-learning: Two orders of magnitude reduction in phonon-phonon scattering rates calculation,” *Materials Today Physics* **41**, 101345 (2024).
- ¹⁶Z. Guo, Z. Han, D. Feng, G. Lin, and X. Ruan, “Sampling-accelerated prediction of phonon scattering rates for converged thermal conductivity and radiative properties,” *Npj Comput. Mater.* **10**, 31 (2024).
- ¹⁷Z. Guo, Z. Han, A. Alkandari, K. Khot, and X. Ruan, “First-principles prediction of thermal conductivity of bulk hexagonal boron nitride,” *Applied Physics Letters* **124** (2024).
- ¹⁸X. Zhang, C. Shao, and H. Bao, “Cryogenic thermal transport properties from accelerated first-principles calculations: Role of boundary and isotope scattering,” *Physical Review B* **110**, 224301 (2024).
- ¹⁹Z. Guo, P. Sokalski, Z. Han, Y. Cheng, L. Shi, T. Taniguchi, K. Watanabe, and X. Ruan, “First-principles prediction of zone-center optical phonon linewidths and ir spectra of hexagonal boron nitride,” *Applied Physics Letters* **125** (2024).
- ²⁰Z. Tang, X. Wang, C. He, J. Li, M. Chen, C. Tang, and T. Ouyang, “Effects of thermal expansion and four-phonon interactions on the lattice thermal conductivity of the negative thermal expansion material scf 3,” *Physical Review B* **110**, 134320 (2024).
- ²¹A. Alkandari, Z. Han, Z. Guo, T. E. Beechem, and X. Ruan, “Anisotropic anharmonicity dictates the thermal conductivity of β -ga 2 o 3,” *Physical Review B* **111**, 094308 (2025).
- ²²J. Wei, Z. Xia, Y. Xia, and J. He, “Hierarchy of exchange-correlation functionals in computing lattice thermal conductivities of rocksalt and zinc-blende semiconductors,” *Physical Review B* **110**, 035205 (2024).
- ²³Z. Guo, I. Katsamba, D. Carne, D. Feng, K. Moss, E. Barber, Z. Fang, A. Felicelli, and X. Ruan, “Electronic and phononic characteristics of high-performance radiative cooling pigments h-bn: A comparative study to baso4,” *Materials Today Physics* , 101721 (2025).
- ²⁴K. Khot, B. Xiao, Z. Han, Z. Guo, Z. Xiong, and X. Ruan, “Phonon local non-equilibrium at al/si interface from machine learning molecular dynamics,” *Journal of Applied Physics* **137**, 115301 (2025), https://pubs.aip.org/aip/jap/article-pdf/doi/10.1063/5.0243641/20444055/115301_1_5.0243641.pdf.

- ²⁵J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, “A survey of general-purpose computation on graphics hardware,” in *Computer graphics forum*, Vol. 26 (Wiley Online Library, 2007) pp. 80–113.
- ²⁶R. Raina, A. Madhavan, A. Y. Ng, *et al.*, “Large-scale deep unsupervised learning using graphics processors,” in *Icml*, Vol. 9 (2009) pp. 873–880.
- ²⁷A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems* **25** (2012).
- ²⁸D. Carne, Z. Guo, and X. Ruan, “Overcoming the curse of dimensionality: Enabling multi-layer photon transport with recurrent neural network,” (2025), arXiv:2509.22890 [physics.optics].
- ²⁹J. E. Stone, J. C. Phillips, L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten, “Accelerating molecular modeling applications with graphics processors,” *Journal of computational chemistry* **28**, 2618–2640 (2007).
- ³⁰D. B. Kirk and W. H. Wen-Mei, *Programming massively parallel processors: a hands-on approach* (Morgan kaufmann, 2016).
- ³¹X. Gonze, F. Jollet, F. A. Araujo, D. Adams, B. Amadon, T. Applencourt, C. Audouze, J.-M. Beuken, J. Bieder, A. Bokhanchuk, *et al.*, “Recent developments in the abinit software package,” *Computer physics communications* **205**, 106–131 (2016).
- ³²M. Hacene, A. Anciaux-Sedrakian, X. Rozanska, D. Klahr, T. Guignon, and P. Fleurat-Lessard, “Accelerating vasp electronic structure calculations using graphic processing units,” *Journal of computational chemistry* **33**, 2581–2589 (2012).
- ³³M. Hutchinson and M. Widom, “Vasp on a gpu: Application to exact-exchange calculations of the stability of elemental boron,” *Computer Physics Communications* **183**, 1422–1426 (2012).
- ³⁴Y. Wei, X. You, H. Yang, Z. Luan, and D. Qian, “Towards gpu acceleration of phonon computation with shengbte,” in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, HPCAsia2020* (Association for Computing Machinery, New York, NY, USA, 2020) p. 32–42.
- ³⁵W. Li, J. Carrete, N. A. Katcho, and N. Mingo, “Shengbte: A solver of the boltzmann transport equation for phonons,” *Computer Physics Communications* **185**, 1747–1758 (2014).
- ³⁶B. Zhang, Z. Fan, C. Zhao, and X. Gu, “Gpu_pbte: an efficient solver for three and four phonon scattering rates on graphics processing units,” *J. Phys. Condens. Matter* **33**, 495901 (2021).
- ³⁷Z. Han, X. Yang, W. Li, T. Feng, and X. Ruan, “Fourphonon: An extension module to shengbte for computing four-phonon scattering rates and thermal conductivity,” *Computer Physics*

Communications **270**, 108179 (2022).