Proceedings of the ASME 2022
International Mechanical Engineering Congress and Exposition
IMECE2022
October 30 - November 3, 2022, Columbus, USA

IMECE2022-95034

DRAFT: A DOMAIN SPECIFIC LANGUAGE APPLIED TO PHONON BOLTZMANN TRANSPORT FOR HEAT CONDUCTION

Eric Heisler*
School of Computing
University of Utah
Salt Lake City, Utah 84112
Email: eric.heisler@utah.edu

Siddharth Saurav
Dept. of Mechanical and Aerospace Engineering
The Ohio State University
Columbus Ohio 43210
Email: siddharthsaurav.1@buckeyemail.osu.edu

Aadesh Deshmukh
School of Computing
University of Utah
Salt Lake City, Utah 84112
Email: u1369232@utah.edu

Sandip Mazumder
Dept. of Mechanical and Aerospace Engineering
The Ohio State University
Columbus Ohio 43210
Email: mazumder.2@osu.edu

Ponnuswamy Sadayappan
School of Computing
University of Utah
Salt Lake City, Utah 84112
Email: saday@cs.utah.edu

Hari Sundar
School of Computing
University of Utah
Salt Lake City, Utah 84112
Email: hari@cs.utah.edu

ABSTRACT

The phonon Boltzmann transport equation is a good model for heat transfer in nanometer scale structures such as semiconductor devices. Computational complexity is one of the main challenges in numerically solving this set of potentially thousands of nonlinearly coupled equations. Writing efficient code will involve careful optimization and choosing an effective parallelization strategy, requiring expertise in high performance computing, mathematical methods, and thermal physics. To address this challenge, we present the domain specific language and code generation software Finch. This language allows a domain scientist to enter the equations in a simple format, provide only basic mathematical functions used in the model, and generate efficient parallel code. Even very complex systems of equations such as phonon Boltzmann transport can be entered in a very simple, intuitive way. A feature of the framework is flexibility in numerical methods, computing environments, parallel strategies, and other aspects of the generated code. We demonstrate Finch on this problem using a variety of parallel strategies and model configurations to demonstrate the flexibility and ease of use.

*Address all correspondence to this author.

NOMENCLATURE

$I_{i,j}$	Phonon intensity for direction i, frequency band j.
v_g	Group velocity.
τ	Scattering time scale.
$\mathbf{s}_i$	Unit vector for direction i.
V	Volume of a cell.

INTRODUCTION

Heat conduction in sub-micron scale crystalline structures can be modeled by the phonon Boltzmann Transport Equation [1,2]. This seven-dimensional equation presents some challenges and some opportunities for efficient computation. In particular, the need to discretize the wavevector and frequency domains provides a good candidate for parallel computation because the phonon intensities in different frequency bands are not directly coupled. They simply need to be reduced when performing the intensity integration to find temperature. The directions are also loosely coupled at boundary interfaces providing an alternative strategy.

There are successful numerical codes for solving the BTE using different parallel strategies [3–5]. In this paper we will be

concerned with deterministic methods, though monte carlo techniques have also been tried [6]. Many of these provide efficient, scalable solutions, but there are some downsides to developing these types of programs. First, they are expensive to create in terms of programming effort and required skills. Beyond the mathematical complexities, the implementation of efficient parallel strategies and data structures for this large set of equations is a challenging task. Second, if they have been optimized for a specific computing environment, mesh, and problem setup, the code may lack flexibility to adapt to variations. Any modification to the scenario will again necessitate substantial effort. Third, their usefulness as a tool is essentially limited to the person or group that developed them. Unless the code was intentionally designed to be a shared tool with detailed documentation, it would be difficult for researchers outside of the development process to use, and even more difficult to adapt to their needs.

To address these challenges, we propose the Domain Specific Language(DSL) and code generation framework Finch as a tool for working with the phonon BTE to study heat conduction. This is a general purpose numerical PDE software that can take a high level description of an equation and generate efficient parallel code to solve it. One of the goals of Finch is to provide a highly configurable solution allowing a variety of code generation target options, mesh import or creation utilities, customizable symbolic operations, discretization options such as finite element(FEM), finite volume(FVM), or mixed methods, and direct access to generated code. Each of these topics will be addressed in more detail below.

Although the interface is designed for a high-level description of the equations, it does assume that the user has a fair understanding of the mathematics. For example, when using FEM the problem must be expressed in the weak form, and when using FVM it must be in integral form with the divergence theorem applied by the user when appropriate.

In this paper we will demonstrate Finch with the phonon BTE to simulate heat conduction. A variety of strategies will be compared and in particular the small programming effort involved in the variation will be shown. Although it would be difficult for this general purpose software to match the performance of a hand-optimized code taking full advantage of the specifics of the problem, some basic performance analysis will be given.

Related Work

Some of the relevant work on numerical BTE solutions has been mentioned above including some deterministic [3–5] and non-deterministic [6] efforts. There are also several other DSLs that have been developed for solving PDEs. The Unified Form Language(UFL) [7] and FreeFEM [8] include languages for representing variational forms of the equations in a way similar to Finch where variables, test functions and coefficients are combined in integral expressions. The Julia-based DSL MetaFEM [9] also involves variational form equations, but with a very different type of grammar.

A relevant FVM DSL is that of OpenFOAM [10], which again combines variable and coefficient components in a high-level expression. It uses a predefined set of operations designed specifically for common problems solved with FVM. There is no support for variational forms.

In contrast, Finch supports different types of expressions depending on the desired tools. It also allows a user to define custom symbolic operators to be included in the expressions. For example, when using FVM, specialized flux operators can be defined that either manipulate the symbolic equations or represent numerical callback functions.

Finch also includes some tools for combining FEM and FVM for situations where some different variables are better suited to different methods. Some modules of Dune [11], such as Dune-fem, can combine FEM and FVM, but these are low-level tools that are difficult to compare to a higher-level DSL.

MODEL

The BTE may be written in terms of the phonon intensity, I, as Eqn. (1) [1, 2]

$$\frac{\partial I}{\partial t} + v_g \cdot \nabla I = \frac{I_0 - I}{\tau} \tag{1}$$

including the group velocity v_g, scattering time scale τ and equilibrium intensity I_0. The intensity is discretized in terms of wavevector direction and frequency, and will be written as $I_{i,j}$ for direction i and frequency band j. The unit vector in direction i is denoted $\mathbf{s}_i$. A finite volume method will be used, so Eqn. (1) is integrated over a control volume V, and the divergence theorem is applied to the term involving spatial derivatives to give Eqn. (2)

$$\frac{\partial \overline{I_{i,j}}}{\partial t} V + |v_g|_j \int_{\partial V} I_{i,j} \mathbf{s}_i \cdot \mathbf{n} dA = \frac{\overline{I_{0,j}} - \overline{I_{i,j}}}{\overline{\tau}_j} V \tag{2}$$

where the overline indicates an average over the volume and the ∂V integral is over the surface area with normal vector $\mathbf{n}$. From here onward values will be assumed equal to their cell averages. Also note that the equilibrium intensity, group velocity, and time scale also depend on frequency.

Temperature is ultimately the quantity of interest, but we do not have a direct way to calculate temperature from non-equilibrium phonon intensity. Rather, the change in temperature can be approximated by considering the heat flux, which is the flux of the total integrated phonon intensity. The first law of thermodynamics then gives the change in internal energy. The relation between internal energy and temperature in the crystalline

material then completes the connection between intensity and temperature. This non-linear relationship allows us to iteratively approximate changes in temperature for each cell. For details on this procedure, please refer to [12].

The boundary conditions used here include isothermal and symmetric boundaries as in Eqn. (3) where the direction index r is for the reflected direction corresponding to i. At isothermal boundaries the intensity is equal to the equilibrium value for a specified temperature. Symmetric boundaries mimic a mirror reflection of the domain across the boundary. In the numerical procedure this effects the flux calculation at the boundary faces by effectively setting the intensity of ghost cells on the outside.

$$I_{i,j}^{outside} = \begin{cases} \overline{I_{0,j}} & \text{at isothermal boundary} \\ \overline{I_{r,j}} & \text{at symmetric boundary} \end{cases} \tag{3}$$

In the demonstration below, one side of the domain has an isothermal boundary representing a cold wall at the same temperature as the initial equilibrium. A small segment of the opposing side is a hot wall at a higher temperature representing a localized heater. The remaining boundaries are symmetric. See Fig. (1).

The initial condition is a thermal equilibrium at the same temperature as the cold wall of the domain.

DOMAIN SPECIFIC LANGUAGE

A goal of the DSL is to work with the most intuitive input possible. The expressions should closely resemble the types of mathematics that a domain scientist would be familiar with, but Finch does assume that the user is familiar enough with the mathematical methods to present the equations in a particular format. To use FEM, the weak form of the equations must be entered, and when using FVM as we are for the BTE, they must be transformed into conservative integral form with the divergence theorem applied as needed. We plan to ease this requirement in a future release to accept more general expressions.

The integral form used for FVM input includes volume and surface integrals for the control volumes(cells). Adopting the terminology of transport equations, the volume terms are labeled source and the surface terms are labeled flux. The time derivative is implied for this type of problem, so a general transport equation (4)

$$\int_V \frac{\partial u}{\partial t} dx = \int_V G(u) dx - \int_{\partial V} \mathbf{F}(u) \cdot \mathbf{n} ds \tag{4}$$

will be entered into Finch as

```
source(u, G_expression)
flux(u, F_expression)
```

where `G_expression` and `F_expression` are strings representing the mathematical expressions. For the BTE, Eqn. (2) would be entered as

```
source(I,"(Io[j] - I[i,j])/tau[j]")
flux(I,"vg[j]*upwind([Sx[i],Sy[i]],I[i,j])")
```

Note that the flux term includes an `upwind` operator. Since $I_{i,j}$ must be approximated at the surface between cells, this operator specifies that an upwind approximation shall be used depending on the vector $\mathbf{s}_i$ which in the two-dimensional case is entered as `[Sx[i],Sy[i]]`.

The symbols used for variables, coefficients, and indices in input expressions must first be defined. We refer to these as entities, and they are created with the following commands.

```
i= index("i",range=[1,ndirs])
j= index("j",range=[1,nbands])
I= variable("I",type=VAR_ARRAY,
        location=CELL,index=[i,j])
Io= variable("Io",type=VAR_ARRAY,
        location=CELL,index=[j])
tau= variable("tau",type=VAR_ARRAY,
        location=CELL,index=[j])
Sx= coefficient("Sx",sx_vals,type=VAR_ARRAY)
Sy= coefficient("Sy",sy_vals,type=VAR_ARRAY)
vg= coefficient("vg",vg_vals,type=VAR_ARRAY)
```

Although I_o and τ are assumed known in the equation for I, they are dependent on temperature, which is in turn dependent on I, so they are created as variables. On the other hand, $\mathbf{s}$ and v_g can be considered coefficients, which have values that are either pre-computed arrays or defined by a function of space-time coordinates.

The equations also include a set of boundary and initial conditions. The initial condition we use for I is the equilibrium intensity for a uniform temperature. These frequency dependent values are computed in `I_init` and set with

```
initial(I, [I_init[b]
            for d=1:ndirs, b=1:nbands])
```

The boundary conditions require a more complicated calculation that cannot be simply expressed as a function of coordinates. Instead of passing a mathematical expression to Finch to be turned into a generated function, we will write our own callback function that Finch will use when handling the boundary condition. Finch takes a string representing the function call, which can include parameters such as coordinates, other entities, and cell information keywords such as the boundary face normal vector.

```
boundary(I, 1, FLUX,
  "isothermal(I,vg,Sx,Sy,j,i,normal,305)")
```

This example will set the boundary condition for variable `I` in boundary region 1 to a fixed flux type condition with a value computed by the callback function `isothermal`. The relevant values for the parameters will be automatically determined by Finch when needed.

Configuration details that need to be specified include time stepping methods, and discretization type and order. A mesh must either be imported from a Gmsh or MEDIT formatted mesh file, or generated internally by Finch's simple generation utility. The relevant commands are illustrated in the documentation and example scripts available in the code repository [13].

CODE GENERATION

A simple FVM solver will consist of a loop over cells in which either the unknown quantities are evolved cell by cell or a linear system is assembled. Any high performance solver will do this loop in parallel, but since all cells are coupled to their neighbors for the flux calculation, this requires substantial communication between processes. In the case of the BTE there are a very large number of unknowns for each cell, and each iteration of the cell loop performs a similar calculation for each discrete direction and frequency band. In Finch the use of indexed variables such as $I_{i,j}$ results in nested loops for each index. This presents new options for parallelizing. Depending on the application these loops can be rearranged and different parallel strategies can be applied. The BTE will default to the following loop structure.

```
for cell = 1:Ncells
  for direction = 1:Ndirections
    for band = 1:Nbands
```

This may not be the ideal configuration. The frequency bands are only coupled through the temperature update which is performed after each time step. This means that in terms of the loop they are essentially independent and a good candidate to be done in parallel. A typical number of frequency bands is around 40 [5] to 80 [4], so the degree of parallelization is limited if only bands are considered. The discrete directions can also be loosely coupled depending on boundary conditions, so they may also be a good choice to do in parallel. Hybrid strategies that parallelize more than one of the loops are also possible as long as the number of processes is carefully chosen and communication is handled efficiently.

By default Finch will do the cell loop in parallel by partitioning the mesh according to the number of processes available. This is overridden by specifying the number of mesh partitions as one. The nested loop structure above can be permuted with the command

```
assemblyLoops(I,[band,"elements",direction])
```

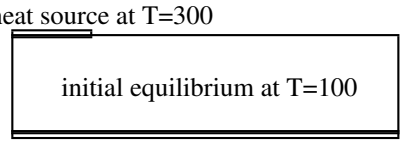

FIGURE 1. SCHEMATIC OF THE $4\mu m \times 1\mu m$ DOMAIN.

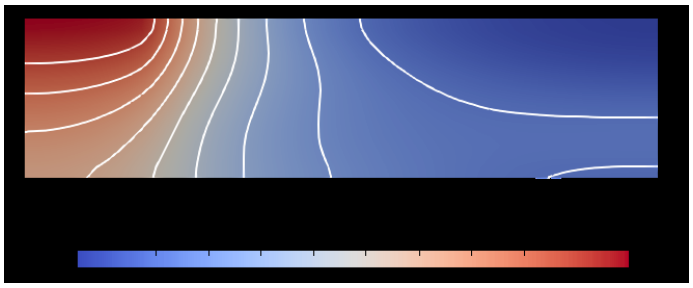

FIGURE 2. TEMPERATURE AFTER 5ns.

which makes the band loop outermost followed by cells(referred to as elements), and finally directions.

Since the communication details for these other parallel strategies are highly application dependent, and simply having Finch communicate all quantities for all cells could be very inefficient, it is better for the user to take control of this task when something other than the cell loop is done in parallel. For example, the band coupling for the temperature update only requires a single value reduction across the bands. Giving this programming burden to the user conflicts with the goals of Finch, so we are currently exploring intuitive interface ideas for communication. In the meantime, full access to MPI and the needed data structures are exposed.

Along with parallel strategies, there are sometimes application details that require a particular procedure or allow special optimizations which may not be performed automatically by Finch. A simple set of commands allow a user to export the generated code, modify it as desired, and import it to be used in the computation. Again, this programming burden does not align with the goals of Finch, so we are continually trying to improve automated solutions, while keeping this as a last resort.

RESULTS AND DISCUSSION

A test problem based on the description above was implemented. The domain is a two-dimensional thin layer with a small region of the boundary acting as a heat source and the opposing wall as a heat sink. See Fig. (1) The other walls are given a symmetric boundary condition to suggest that the scenario is mirrored in those directions. Initially the temperature of the whole domain is at an equilibrium value equal to the cold wall. Figure (2) shows the temperature after 2000 time steps, which is 5ns.

In addition to convenience, performance and scalability are

key goals of this software. We do not expect the generated code to surpass a hand-optimized, single purpose program in terms of performance. We do hope to achieve comparable performance and good scalability, which is the subject of ongoing development. This example problem was tested on the Notchpeak cluster at the University of Utah's Center for High Performance Computing using two-socket Intel XeonSP Cascadelake nodes with 40 cores and 192 GB of memory. The discretization includes 16 directions and 40 frequency bands on a 60×15 uniform mesh of rectangular cells.

Two different parallel strategies were used. A band-based strategy divides the frequency bands across processes requiring only a reduction of the integrated intensities at each time step. Figure (3.a) shows that the execution time scales well, but is limited to 40 processors because there are only 40 bands. Since no mesh partitioning is done and only minimal communication is needed, the overhead of parallelizing is small.

A cell-based strategy in Fig. (3.b) partitions the mesh. This allows a higher number of processors, but due to the increased communication overhead the scaling is not quite as good as the band-based strategy. Other options that could be tried are direction-based partitioning, which would require some communication for certain boundary conditions as well as the reduction at each time step. To do this efficiently would require communicating very specific values at some parts of the boundary. Although this is not challenging for a hand-written code, the details of this communication are highly application dependent and appropriate methods for supplying these details through the DSL are still under development.

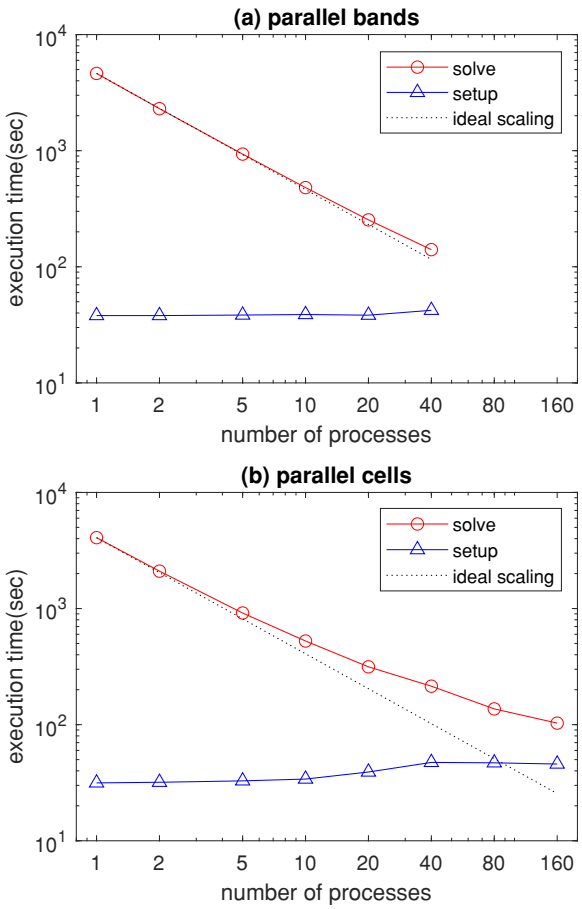

FIGURE 3. SCALING OF (a)BAND-BASED PARALLEL AND (b)CELL-BASED PARALLEL STRATEGIES.

CONCLUSION

We have demonstrated the DSL and code generation framework Finch as a convenient and highly customizable tool to investigate microscale heat conduction with the phonon BTE. This complex, seven-dimensional equation presents some challenges as well as opportunities for efficient computation that can be handled with relatively little programming effort.

Future developments will focus on improving performance through more adaptive code generation and providing an interface for more sophisticated parallel strategies without offloading too much of the programming burden on the user.

This work was funded by National Science Foundation grants 1808652 and 2008772.

REFERENCES

[1] Majumdar, A., Tien, C., and Gemer, F., 1997. "'microscale energy transport in solids". *Microscale Energy Transport*, pp. 3–93.

[2] Majumdar, A., 1993. "Microscale Heat Conduction in Dielectric Thin Films". *Journal of Heat Transfer,* ***115***(1), 02, pp. 7–16.

[3] Srinivasan, S., Miller, R., and Marotta, E., 2004. "Parallel computation of the boltzmann transport equation for microscale heat transfer in multilayered thin films". *Numerical Heat Transfer, Part B: Fundamentals,* ***46***(1), pp. 31–58.

[4] Ni, C., and Murthy, J. Y., 2009. "Parallel computation of the phonon boltzmann transport equation". *Numerical Heat Transfer, Part B: Fundamentals,* ***55***(6), pp. 435–456.

[5] Ali, S. A., Kollu, G., Mazumder, S., Sadayappan, P., and Mittal, A., 2014. "Large-scale parallel computation of the phonon boltzmann transport equation". *International Journal of Thermal Sciences,* ***86***, pp. 341–351.

[6] Péraud, J.-P. M., and Hadjiconstantinou, N. G., 2011. "Efficient simulation of multidimensional phonon transport using energy-based variance-reduced monte carlo formulations". *Physical Review B,* ***84***(20), p. 205331.

[7] Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., and

Wells, G. N., 2014. "Unified form language: A domain-specific language for weak formulations of partial differential equations". *ACM Trans. Math. Softw.,* ***40***(2), mar.

[8] Hecht, F., 2012. "New development in freefem++". *J. Numer. Math.,* ***20***(3-4), pp. 251–265.

[9] Xie, J., Ehmann, K., and Cao, J., 2021. Metafem: A generic fem solver by meta-expressions.

[10] Macià, S., Martínez-Ferrer, P. J., Mateo, S., Beltran, V., and Ayguadé, E., 2019. "Assembling a high-productivity dsl for computational fluid dynamics". In Proceedings of the Platform for Advanced Scientific Computing Conference, PASC '19, ACM Press, pp. 1–11.

[11] Dune, 2022. Dune.

[12] Ali, S. A., 2017. "Phonon boltzmann transport equation (bte) based modeling of heat conduction in semiconductor materials at sub-micron scales". PhD thesis, The Ohio State University.

[13] Heisler, E., Deshmukh, A., and Sundar, H., 2022. Finch code repository.