Towards a Verifiable Domain-Specific Language for Hardware-Accelerated Stencils

Damian Rouson[1] and Xaokun Yang[1,2]

[1]Lawrence Berkeley National Laboratory, [2]University of Houston - Clear Lake

Abstract

Defining a domain-specific language (DSL) that supports vector-calculus abstractions eases the porting of partial differential equation (PDE) solvers to specialized architectures. Sufficiently high-level abstractions empower users to express universal laws with sufficient generality that the laws must always hold true within their domain of validity. A broad class of PDE solvers employs stencil-based algorithms, the target domain of Berkeley Lab's stencil accelerator chip co-design project. First released as open-source in January 2026, the Formal software framework lays a foundation for defining an embedded DSL based on composable operators that implement mimetic numerical methods -- stencil algorithms that guarantee satisfaction of discrete versions of important vector calculus theorems.

The Formal DSL will be the front-end to a new class of stencil-PDE accelerators developed jointly by LBNL, UHCL, and UC Berkeley through the DOE Competitive Portfolios for Computer Science Project. This offers the potential of an order of magnitude acceleration for this important category of computational methods to serve the DOE mission.

Future work on the Formal DSL will facilitate software verification via type-safe templates that enable problem-specific correctness proofs relying upon generic function theory and carefully crafted unit tests.

Stencil Taxonomy

We classify stencil algorithms according to the following properties:

1. **By Dimensionality:**
   a. 1D Stencils, e.g., $\{u_{i-1}, u_i, u_{i+1}\}$;
   b. 2D Stencils, e.g., 5-point or 9-point stencils on a Cartesian grid.
   c. 3D Stencils: e.g., 7-point, 19-point, or 27-point stencils.
   d. nD Stencils, e.g., higher-dimensional or phase-space methods.
2. **By Radius and Size:** Radius *r*: How far the stencil reaches from the central point; Radius-1: only immediate neighbors; Radius-2: includes next-nearest neighbors. Size: Total number of points used. E.g., a 2D 5-point stencil has size 5, radius 1.
3. **By Shape/Geometry:** Cross/Star Stencil (e.g., 5-point in 2D): center + cardinal directions. Box/Full Grid Stencil (e.g., 9-point in 2D, 27-point in 3D): includes diagonals; Diamond or Hexagonal Stencil: used on non-Cartesian or hexagonal meshes; Irregular/Adaptive Stencils: used in adaptive mesh refinement (AMR), unstructured grids, or meshless methods
4. **By Order of Accuracy:** First-Order: low accuracy, but often more stable. Second-Order: e.g., classic central difference. Higher-Order: fourth-order, sixth-order, compact stencils require more neighbors and sophisticated coefficients.
5. **By Temporal Behavior:** Explicit Stencils: value depends only on the current or past time step. Implicit Stencils: require solving linear systems (e.g., Crank–Nicolson). Multistep/Multistage Schemes: e.g., Runge-Kutta with spatial stencils.
6. **By Application Domain**

By Application Domain	Common Stencil Type
Heat Equation	5-point or 7-point diffusion stencils
Wave Equation	3-point (in time) + 5/7/9-point spatial
Fluid Dynamics (e.g., Navier–Stokes)	cell grid stencils, staggered stencils
Compressible Euler	Godunov method, PPM reconstruction, approximate Riemann solver
Electromagnetics (FDTD)	Yee stencil (curl-based)
Quantum Mechanics	High-order Laplacians
Image Processing	Sobel, Gaussian, Laplacian filters

7. **By Grid Topology:** Block-Structured Grid Stencils: regular, including patch-based adaptively refined; Unstructured Grid Stencils: require indirect addressing. Meshless Methods: use kernel-based or point cloud stencils.

Software Framework

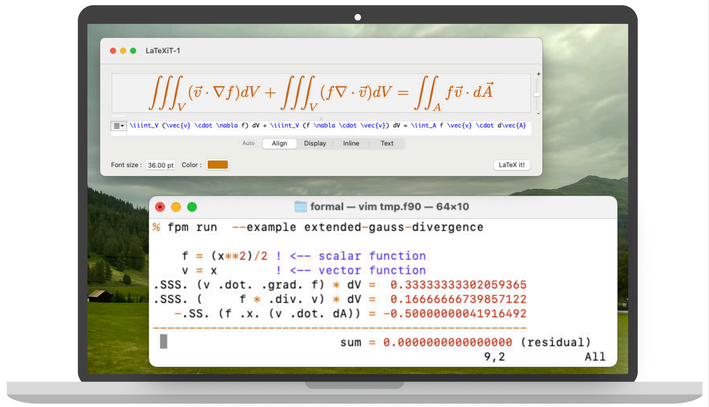
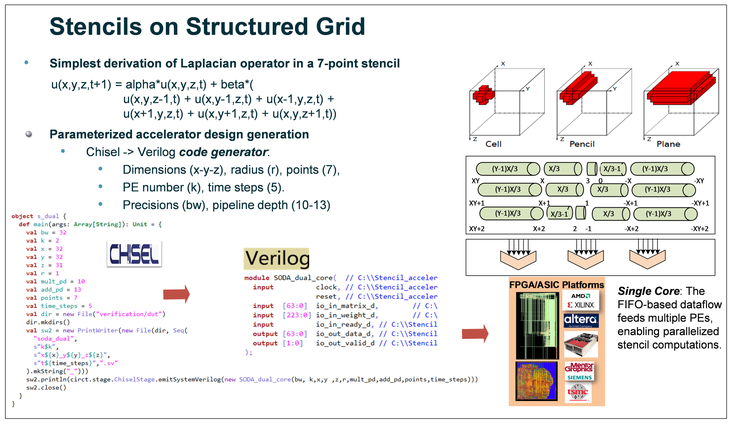


Figure 1. The Formal framework (https://go.lbl.gov/formal) defines mimetic software abstractions, i.e., types and operators that mimic the continuous integral and differential operators of vector calculus. Formal supports these abstractions with the discrete calculus of Corbino & Castillo [1] and Dumett & Castillo [2]

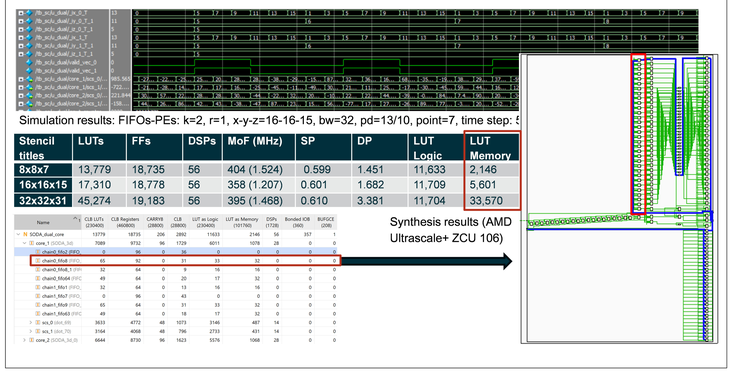
[1] Corbino, J., & Castillo, J. (2020). *Journal of Computational and Applied Mathematics*, *364*, 112326.

[2] Dumett, M. & Castillo, J. (2022) Computational Science Research Center, San Diego State University, *Tech. Rep. CSRCR2022-2*.

Hardware Acceleration



Results



Stencil titles	LUTs	FFs	DSPs	MoF (MHz)	SP	DP	LUT Logic	LUT Memory
8x8x7	13,779	18,735	56	404 (1.524)	0.599	1.451	11,633	2,146
16x16x15	17,310	18,778	56	358 (1.207)	0.601	1.682	11,709	5,601
32x32x31	45,274	19,183	56	395 (1.468)	0.610	3.381	11,704	33,570

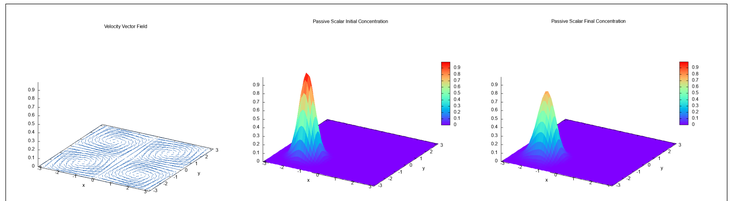


Figure 2. A Taylor-Green vortex velocity field (left) and passive scalar concentration initial (middle) and final (right) states computed with Formal mimetic methods of 4th-order accuracy in space and time to approximate the scalar advection/diffusion PDE:

$$\partial s/\partial t = \nabla\cdot(D\nabla s) - \nabla\cdot(\vec{v}s)$$

which a Formal repository example expresses with Formal types and operators as:

```
ds_dt = .div. (D * .grad. s) - .div. (v * s)
```

Impact

- Finalizing LightSolver/LBNL CRADA to study accelerating LBNL's Formal, Chombo, and AMREx on LightSolver's Light Processing Unit
- Collaborating with San Diego State University (SDSU) on positioning Formal as an alternative to SDSU's MOLE C++/MATLAB library

Future Work

Generic programming with enforceable type-safety requirements will facilitate

- Defining an embedded DSL via Fortran 2028 templates already supported in the DOE-funded LFortran compiler (https://lfortran.org)
- Problem-specific formal correctness proofs

Algorithm-Driven Codesign of Specialized Architectures for Energy Efficient HPC

This research is funded by the Competitive Portfolios for Advanced Scientific Computing Research Program of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under contract DE-AC02-05CH11231.