20.4 A 28nm Physics Computing Unit Supporting Emerging Physics-Informed Neural Network and Finite Element Method for Real-Time Scientific Computing on Edge Devices

Yuhao Ju, Ganqi Xu, Jie Gu

Northwestern University, Evanston, IL

2024 IEEE International Solid-State Circuits Conference (ISSCC) | 979-8-3503-0620-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISSCC49657.2024.10454502

The demand for real-time computing on edge devices from emerging applications, e.g. AI, has exploded in recent years. Lately, physics-based scientific computing has also drawn significant interests driven by the growth of real-time applications, e.g., VR, IoT, robotics, etc. Fig. 20.4.1 shows examples of real-time physics-based computation including structural deformation in photorealistic VR/MR, robot dynamic control, temperature monitoring in additive manufacturing, and real-time leak-gas tracking. Unfortunately, hardware support for numerical scientific computing on edge devices is relatively poor, hindering the use of high-accuracy, high-resolution physics-based computing in real time. Figure 20.4.1 shows an example of beam deformation analysis in VR/MR falling short of a real-time latency target using classic solvers due to the large number of iterations for convergence. Recently, ASIC solvers have been designed to solve Poisson equation-related applications with a finite difference method (FDM), but have trouble handling more complex structures [1-3]. To overcome the real-time hurdle, physics-informed neural network (PINN) or physics-informed machine learning (PIML) solutions [4-7] are being developed by the scientific community, using a data-driven approach to boost the computing efficiency of physics solvers. Figure 20.4.1 shows PINN solutions can reach 1900-10000× speedup compared with classic solvers based on Nvidia Modulus with less than 1% accuracy loss [4]. However, if numerous physics equations are to be processed by a PINN, highly diversified dataflows are needed to support a variety of PINN models, making it unfriendly to an ASIC solution. In addition, a tradeoff of speed and accuracy needs to be made between a PINN and classic numerical solutions for a specific application. To overcome these challenges, this work presents a unified physics computing unit (PhyCU) architecture supporting both PINNs and classic finite element method (FEM) solution. The highlights of PhyCU are as follows: 1) This work delivers an ASIC solution supporting inference for most major PINN models with configurable dataflow; 2) The PhyCU architecture also natively supports the classic FEM through a conjugate gradient iterative method (CG) providing a high-accuracy alternative using the same hardware; 3) Sparsity and data compression techniques for both PINN and FEM computation are developed achieving orders of magnitude latency reduction compared with a classic solution on GPU and 19.5-35.9× energy savings compared with prior ASICs.

Figure 20.4.2 shows the supported algorithms, namely, PINN and FEM. The PINN takes coordinates and time steps as input data for a neural network (NN) model and generates the physical status for each mesh node, e.g., fluid velocity. As a PINN's loss function is confined by underlining physics principles, boundary conditions and initial conditions, PINN offers smaller and more accurate models compared with a plain NN. As for the FEM algorithm, after meshing the object with selected element shape, basic functions in cooperation with variational calculus and integrals are used to generate a symmetrical equation system. CG is the selected numerical method of PhyCU FEM mode due to its high convergence efficiency for complicated systems, e.g., 125× fewer iterations than some other iterative methods from prior works, e.g. Jacobi, and its high compatibility with the PINN architecture due to the use of matrix multiplication. As shown in Fig. 20.4.2, the PhyCU architecture contains an array of 9×16 2D physics processing elements (PHY-E) with top general purpose (TGP) SRAM banks, bottom general purpose (BGP) SRAM banks, input SRAM banks and a SRAM bank for special parameters. An Input Data Compression Module (ICDM) compresses coordinates with simple adders and control logic by utilizing the physics meshing characteristics for both PINN and FEM modes. An Offset-Based Sparsity Adders Scheduler (OBSAS) is designed to improve sparse matrix-vector multiplication (SpMV) in CG and PINN. The PHY-E supports output/weight stationary NN dataflows, with a multiplier and an ALU for various numerical operations in FEM and PINN. PHY-E supports 8b,16b, 32b precision for latency and accuracy tradeoff.

Figure 20.4.3 shows the supported highly diversified PINN inference models with 7 dedicated dataflows. Except the common NN dataflows, such as fully connected (FC) and convolutional NN (CNN), many PINN models need cos/sin activations such as Fourier Network (FN), SiReNs [5], etc. To realize cos/sin in the integer domain, polynomial approximation is implemented in PhyCU by approximating cos/sin functions as piecewise functions with the PHY-E array used for range selection and MAC operations. Another specially built dataflow is for the Discrete Fourier Transform (DFT) of the Fourier Neural Operator (FNO) [6]. Mathematical transformation with trigonometric function is used to replace DFT with matrix multiplications with a small matrix size by eliminating the repeated calculations in the original DFT, which provides a 26× run cycle saving for an application with a 32×32 elements mesh. For the dataflow in the Deep Galerkin Method (DGM) network, which is similar to LSTM, PhyCU reuses input SRAM as the final output SRAM avoiding the data transfer for later iterations in the DGM network. Figure 20.4.3 also shows the details of the input mesh data compression (IDCM) operation used in this work. Different elements have the same space within a specific segment as in the example of the bottom slice from a beam mesh. For each segment with the same grid space, only initial coordinate and grid space numbers need to be stored in input SRAM. ICDM utilizes adder chains to accumulate space numbers from the initial coordinates for generating a complete input dataset automatically, eliminating the coordinate information for the segments of the object. By implementing ICDM, input data size is reduced by 74% for PINN and 81% for FEM for a 3D sink heat-transfer analysis. By gating the input SRAM during computing using the compressed data from ICDM, a 27-32% power saving is achieved for the first layer inference of PINN or FEM integral operation.

Figure 20.4.4 describes details and optimizations for FEM mode of the PhyCU. The PHY-Es transfer the triple integral of 3D objects and structures to MAC and ALU operations with coordinates from IDCM as input. Among the three major operations in the CG algorithm, SpMV takes 87% of CG workload in each iteration. To optimize SpMV, an Offset-Based Sparsity Adders Scheduler (OBSAS) is implemented exploiting the sparsity of the coefficient matrix of the equation system (matrix A) which is the integral result. In FEM, each node of mesh only interacts with its neighbor nodes. Hence, the non-zero values of matrix A are only located along the diagonal groups with three consecutive elements, as in the beam mesh example shown in Fig. 20.4.4. Utilizing fixed offsets, e.g. length offset, layer offset on sparse matrix A and the reload offset from PHY-E array size, a significant compression is achieved leveraging the repetitive pattern of meshing. As shown in Fig. 20.4.4, the indices of each row of compressed matrix A are continuous and can be generated by shifting the indices from other rows in the same group. By utilizing self-accumulating adders and 3 offsets above, the OBSAS can generate the required address for the parameter vector Pk for SpMV of A*Pk in CG without any index record of the compressed matrix A. Pk can be directly sent to the PHY-E array to be multiplied by a group of compressed matrix A after generating all Pk values by 2 shifters in the OBSAS. The compression through the OBSAS leads to a 460× CG speedup on a 3D 12500-element heat-sink application with FEM.

A 28nm PhyCU test chip has been fabricated. Figure 20.4.5 shows three detailed real-time test cases. The first case is a beam deformation from a hand push using the dynamic equilibrium equation in a VR/MR environment with a 25fps requirement. PhyCU finished the deformation analysis in only 8ms by using a GNN-based PINN operator vs. 9s on RTX3080 GPU using conventional solver rendering a 1125× speedup with a 1.9% accuracy degradation. The second case is fluid pressure analysis with an aneurysm during medical imaging. PhyCU finished the analysis in 22ms achieving 2590× speedup vs. conventional solvers on GPU with 2.6% accuracy loss. In the third case, thermodynamics and fluid dynamics are combined for heat-transfer and fluid-velocity analysis. PhyCU finished the analysis in 40ms with 1839× speedup over GPU and 3.39% accuracy loss. Ten additional test cases from Nvidia Modulus [4] are also shown in Fig. 20.4.5. PINN in PhyCU achieves 434-to-2457× speedup over GPU with 1-to-5.7% (average 2.4%) accuracy loss.

Figure 20.4.6 shows the measured power, frequency and energy efficiency with a supply voltage scaling from 0.9V to 0.55V. A 1.14-to-2.67TOPS/W energy efficiency and a 1.01-to-2.05TOPS/W energy efficiency are achieved for 16b PINN and FEM, respectively. A comparison table with prior physics solvers is shown in Fig. 20.4.6. The PhyCU in this design enables physics scientific computing for both PINN and FEM and solves a larger space of applications beyond Poisson equations with lower latency and energy consumption compared with prior work [1-3]. For the example of a 512-element 3D Poisson equation test case, PhyCU achieves 382-to-779× speedup and 19.5-to-35.9× energy saving compared with prior ASICs. The latency and energy savings from PhyCU FEM comes from the faster convergence of CG method, sparsity techniques and bit-parallel computing, while the savings from PhyCU PINN are attributed to the lower computing cost of PINN inference. Figure 20.4.7 shows the die photo and chip specifications.

Acknowledgement:
This work is supported in part by NSF grant CCF-2008906.

References:
[1] J. Mu et al., "29.2 A 21×21 Dynamic-Precision Bit-Serial Computing Graph Accelerator for Solving Partial Differential Equations Using Finite Difference Method," *ISSCC*, pp. 406-408, 2021.
[2] T. Chen et al., "A 1.87-mm2 56.9-GOPS Accelerator for Solving Partial Differential Equations," *IEEE JSSC*, vol. 55, no. 6, pp. 1709-1718, 2020.
[3] J. Mu et al., "A Scalable Bit-Serial Computing Hardware Accelerator for Solving 2D/3D Partial Differential Equations Using Finite Difference Method," *ESSCIRC*, pp. 353-356, 2022.
[4] Nvidia Modulus Document, <https://docs.nvidia.com/modulus/index.html>, accessed: November 2023.
[5] V. Sitzmann et al., "Implicit Neural Representations with Periodic Activation Functions," *NeurIPS*, pp. 7462-7473, 2020.
[6] Z. Li et al., "Fourier Neural Operator for Parametric Partial Differential Equations," *ICLR*, 2021.
[7] Q. Hernandez et al., "Thermodynamics-Informed Neural Networks for Physically Realistic Mixed Reality," *arXiv: 2210.13414,* 2022.

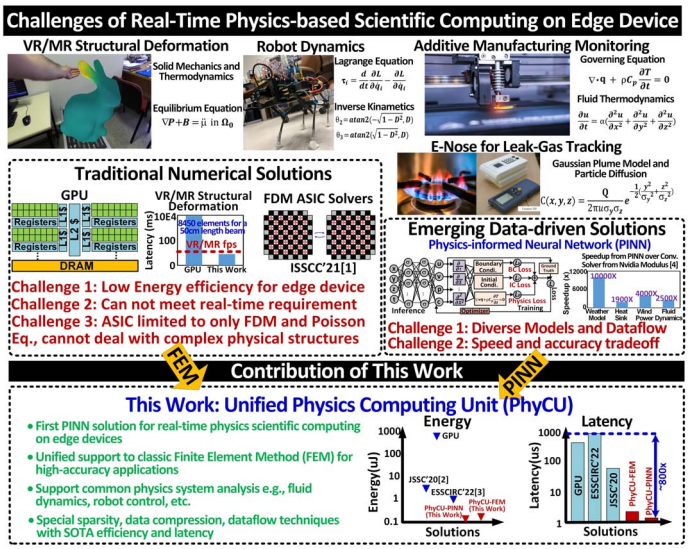


Figure 20.4.1: Applications and the challenges of existing numerical and PINN solutions for real-time physics scientific computing on edge devices, as well as the contributions of this work.

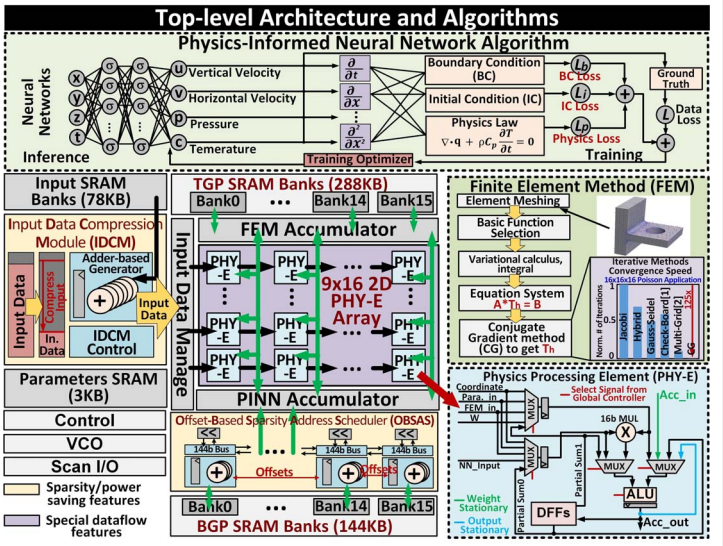


Figure 20.4.2: PINN algorithm and finite element method (FEM) using conjugate gradient (CG) iterative method. Top-level chip architecture with sparsity, data compression and dataflow features.

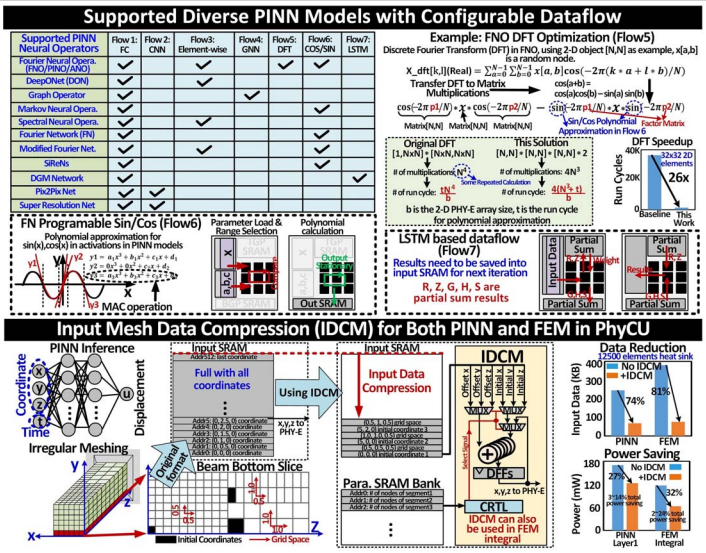


Figure 20.4.3: Examples of the configurable dataflows optimized for PINN supporting diversified models and the input mesh data compression (IDCM) technique for both FEM and PINN.

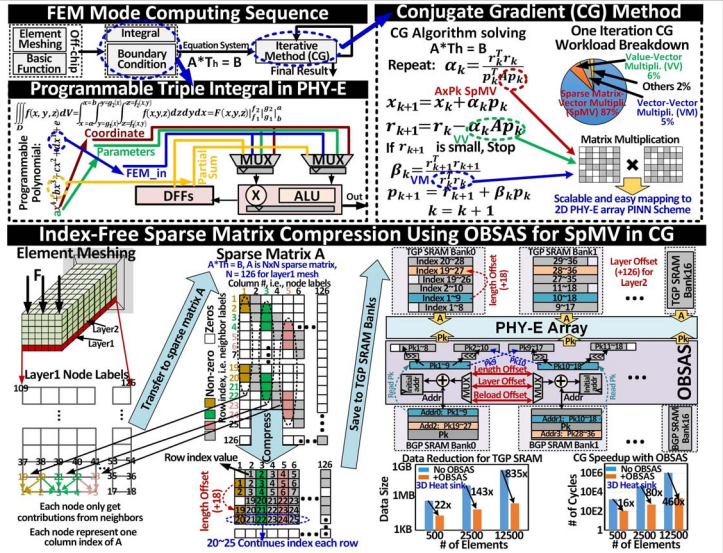


Figure 20.4.4: Implementation details of FEM mode with programmable integral, optimized CG iterative method, and index-free data compression technique using Offset-Based Sparsity Adders Scheduler (OBSAS).

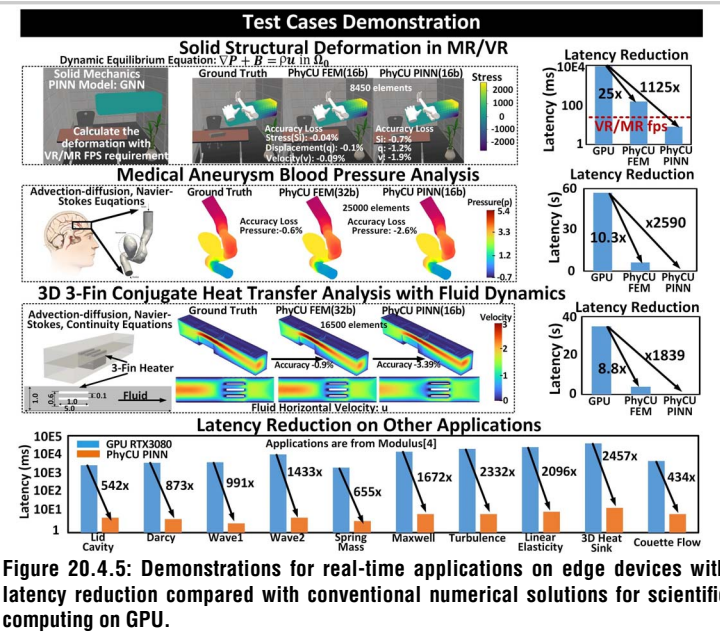
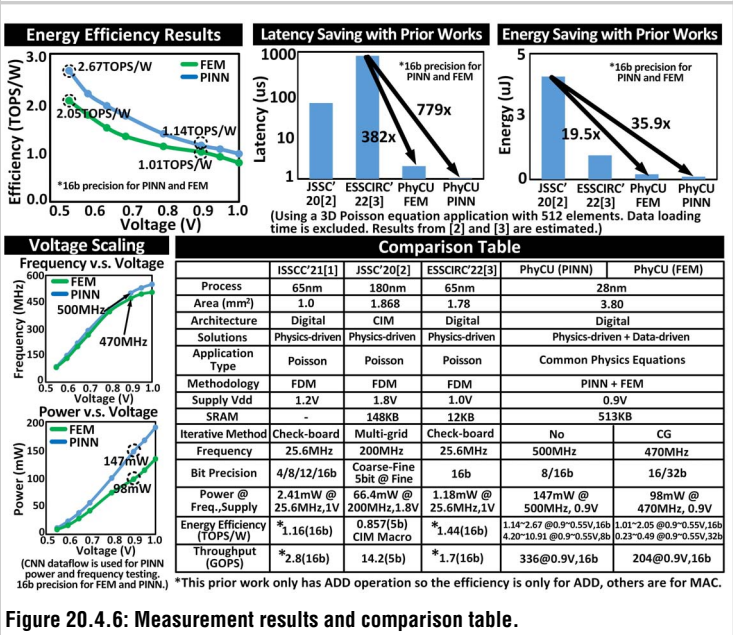


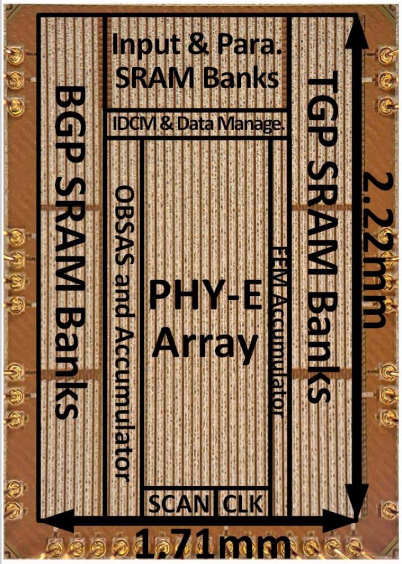
Figure 20.4.5: Demonstrations for real-time applications on edge devices with latency reduction compared with conventional numerical solutions for scientific computing on GPU.



	ISSCC'21[1]	JSSC'20[2]	ESSCIRC'22[3]	PhyCU (PINN)	PhyCU (FEM)
Process	65nm	180nm	65nm	28nm	
Area (mm²)	1.0	1.868	1.78	3.80	
Architecture	Digital	CIM	Digital	Digital	
Solutions	Physics-driven	Physics-driven	Physics-driven	Physics-driven + Data-driven	
Application Type	Poisson	Poisson	Poisson	Common Physics Equations	
Methodology	FDM	FDM	FDM	PINN + FEM	
Supply Vdd	1.2V	1.8V	1.0V	0.9V	
SRAM	-	148KB	12KB	513KB	
Iterative Method	Check-board	Multi-grid	Check-board	No	CG
Frequency	25.6MHz	200MHz	25.6MHz	500MHz	470MHz
Bit Precision	4/8/12/16b	Coarse-Fine 5bit @ Fine	16b	8/16b	16/32b
Power @ Freq.,Supply	2.41mW @ 25.6MHz,1V	66.4mW @ 200MHz,1.8V	1.18mW @ 25.6MHz,1V	147mW @ 500MHz, 0.9V	98mW @ 470MHz, 0.9V
Energy Efficiency (TOPS/W)	*1.16(16b)	0.857(5b) CIM Macro	*1.44(16b)	1.14~2.67 @0.9~0.55V,16b 4.20~10.91 @0.9~0.55V,8b	1.01~2.05 @0.9~0.55V,16b 0.23~0.49 @0.9~0.55V,32b
Throughput (GOPS)	*2.8(16b)	14.2(5b)	*1.7(16b)	336@0.9V,16b	204@0.9V,16b

*This prior work only has ADD operation so the efficiency is only for ADD, others are for MAC.

Figure 20.4.6: Measurement results and comparison table.



Technology	28nm CMOS
Chip Area	3.80mm²
SRAM Size	513KB
# of PE	144
Bit Precision	INT8, 16, 32
Max Freq.	500MHz
Supply Voltage	0.55~0.9V
PINN Power	147mW@0.9V
FEM Power	98mW@0.9V
Peak Perform. (GOPS)	336 (0.9V, 16b)
PINN Efficiency (TOPS/W)	1.14 (0.9V, 16b)
FEM Efficiency (TOPS/W)	1.01 (0.9V, 16b)

Figure 20.4.7: Die micrograph and details.

979-8-3503-0620-0/24/$31.00 ©2024 IEEE