A Unified Microrobotic Visual-Perception Processor with 62.2-FPS/mm^2 and 103-uJ/frame Navigation in 28nm

Shuyuan Zhang[1], Yujin Wang[1], Yifan He[1], Yuchen Su[2], Huazhong Yang[1,3], Yongpan Liu[1,3], Hongyang Jia[1,3]

[1]Department of Electronic Engineering, [2]Xingjian College, [3]BNRist, Tsinghua University, China.

2024 IEEE Asian Solid-State Circuits Conference (A-SSCC) | 979-8-3503-7632-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/A-SSCC60305.2024.10848990

Microrobotic technology has made significant advancements recently, while the payload and power limitations restrict its development toward greater intelligence, for example, the constraint of ~100mg and ~100mW in an insect-sized robot [1]. As an essential role in robotic navigation, visual perception usually has two critical components: localization and neural-network (NN) inference. The former has unique operators and dataflow featuring small-kernel filtering in the vision front-end (VFE) and, more importantly, the linear solver in the optimization back-end involves heavy Cholesky decomposition which is hard to cast into matrix-vector multiplication (MVM) [2]. These distinct dataflows usually lead to heterogeneous systems with separated fixed-function accelerators [3-7]. The visual-inertial odometry in [3] focuses on localization and exploits fixed-pattern sparsity in the back-end, which tradeoffs efficiency with restrictions on applications' data distributions. The visual-perception SoCs in [5,6] handle localization front-end and NN inference in robots efficiently, with the optimization back-end left external. However, severe system throughput degradation occurs when the compute-demand ratio deviates from the native supply of the accelerators in diverse application scenarios [8], as shown in Fig. 1.

This motivates the unified visual-perception processor (VPP) where diverse computational tasks can be flexibly mapped to the processor, greatly alleviating performance loss. However, the efficient unified VPP still faces key challenges: (1) The dominant operator in localization back-end, i.e., Cholesky decomposition, which prepares triangular matrices for solving linear equations, is unfriendly for SIMD acceleration due to its complex data indexing and dependency. (2) Drastically different parallelisms for MVM-based operators in the visual front-end and NN inference with various compute precision require careful utilization enhancement in the MAC engine array.

This work proposes a unified processor featuring architecture- and circuit-level hardware fusion for localization and NN inference to enable efficient processing of multi-task robotic visual perception. Fig.2 shows an architecture overview with three major insights: (1) A reconfigurable matrix-multiply-decompose array (ReMMDA) enables fast linear solving with in-situ triangular-stationary updating. (2) Reconfigurable vector engines (RVEs) with reduced multi-precision datapaths and the multi-granularity operand buffer (MGOB) perform flexible data reuse for diverse granularities. (3) A hybrid-precision-localization dataflow (HPLD) reduces local buffering using NN-compatible bit width, further enabling unified coarse-grain ISA with merged post-processing datapath (MPPD). The proposed VPP includes a 512kB unified local memory and a 4kB instruction memory holding 32b VLIW-style instructions for heterogeneous operations. The ReMMDA comprises 32 RVEs, each performing 16 BF16 MACs by default. The processor also includes a specialized Rodrigues engine and a 32b APB-style test-only off-chip interface.

Figure 3 illustrates the array-level fusion of MVM and fast linear solver in the ReMMDA. To overcome the high latency and complexity of Cholesky decomposition, the spatial array and triangular-stationary scheme are co-designed to embed a linear solver in the NN MAC array, enabling 2D data broadcasting and in-situ data reuse for Cholesky decomposition with minimal area overhead. The ReMMDA is inherently configured as a 16x32 BF16 NN MAC array. Two adjacent BF16 MAC datapaths in the RVE can form a solver-processing element (SPE) with adders multiplexed for local updating when the ReMMDA is reconfigured to a Cholesky decomposition engine. It reuses the stage registers in adder trees as local accumulation buffers for intermediate values. The ReMMDA performs BF16 (sub-)matrix decomposition with a maximum dimension of 16×32 (2 elements per SPE) and supports larger matrices by tiling and processing sequentially. Fig. 3 also shows the triangular-stationary scheme: after loading the Hessian matrix from the linearization step to the registers in SPEs, the first-row vector moves to MPPD for *sqrt* and *div*, then pushed back to vertical and horizontal solver buffers for broadcasting in the next step. Then, the reused BF16 multipliers and adders in SPE perform coefficient subtraction for later rows and columns through the multiplexed adders with broadcasted data. The computed row is pushed back to the unified memory when the triangular-stationary scheme iterates to the next row. As a result, the ReMMDA reduces the complexity of Cholesky decomposition from O(n^3) to O(n) within its hardware dimensionalities and achieves ~18× speedup than the sparsity-aware processing in [3] for 300×300 matrix in terms of cycles.

Figure 4 shows the logic-sharing design of the multi-precision RVE and the mapping of MVMs with different sizes. In RVEs, both the exponent and mantissa datapaths of BF16 MAC can perform INT8/FP8 multiplication. For high-dimensional MVMs, such as Conv-BN layers, weights will be loaded to the MGOB and kept stationary for up to 128 cycles thanks to the 16kB accumulation buffer. For low-dimensional MVMs, such as small kernel filtering and depth-wise convolution, the kernel is broadcasted to the whole array as input and the image is pushed to the top rows of the MGOB one line per cycle. Then, the MGOB groups the image lines and pushes them to the ReMMDA with automatic parallel vectorization of sliding windows, enhancing effective input bandwidth and exploring more data reuse. Particularly, for 3×3 filter kernels, an RVE will split into two smaller adder trees by early outputting the results, achieving higher utilization. Gating of idle logic and careful selection of transistor threshold voltage help further energy reduction.

Figure 4 also illustrates the HPLD. Instead of using INT16 or FP32 throughout the dataflow, it uses hybrid INT8/BF16 operands for localization with high bit-width MAC results to maintain precision, reducing intermediate buffering with negligible accuracy loss. In this way, the bit-width of operators for localization aligns with typical NN inference, enabling the sharing of memory blocks, interconnect fabrics, MAC hardware, and post-processing logic. These lead to unified interconnect fabrics (UIF) for programmable dataflow and MPPD with 16-way BF16 SIMD, supporting a combination of linear and nonlinear operations for both NN (e.g., *ReLU*) and localization. Besides the unified design, the HPLD and the UIF enable a 66% reduction in local memory, as shown in Fig. 5, balancing the area of computation logic to more than 50% of the unified processor.

The prototype is fabricated in 28nm CMOS with a 2.1mm^2 core area. As shown in Fig.5, it achieves high energy efficiency and throughput with high accuracy in both NN inference (YOLOv3-tiny) and localization (VINS-Mono [9] on EuRoC). The processor achieves an average relative error of 0.147% for localization, significantly smaller than 0.28% of the leading prior work [3]. The comparison table in Fig.6 shows that the proposed VPP achieves up to 30.9× speedup for Cholesky decomposition. More specifically, for a typical visual navigation shown in [5] including localization and Dronet-based collision avoidance, the prototype achieves a peak area efficiency of 62.2 FPS/mm^2 and a peak energy efficiency of 103 uJ/frame under energy-efficient mode at 0.6V, whose latency and energy numbers include the control overheads measured from a light-weight RISC-V CPU [10] implemented in another chip with the same technology, with up to 6.9x navigation area efficiency than priors works. Fig.7 provides the die photo and chip summary. Compared to a combination of typical localization [3] and NN inference [7] accelerators, the unified design of VPP achieves up to 8.6× system-level area efficiency as plotted in Fig. 7.

Acknowledgement:

This work was supported in part by NSFC Grant 62374101. The corresponding author is Hongyang Jia.

References:

[1] N. T. Jafferis *et al*.. "Untethered flight of an insect-sized flapping-wing microscale aerial vehicle," Nature, 2019.

[2] H. Mehta *et al*., "ALX: Large Scale Matrix Factorization on TPUs," arXiv:2112.02194.

[3] A. Suleiman *et al*., "Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones," JSSC, 2019.

[4] G. Park *et al*., "20.8 Space-Mate: A 303.5mW Real-Time Sparse Mixture-of-Experts-Based NeRF-SLAM Processor for Mobile Spatial Computing," ISSCC, 2024.

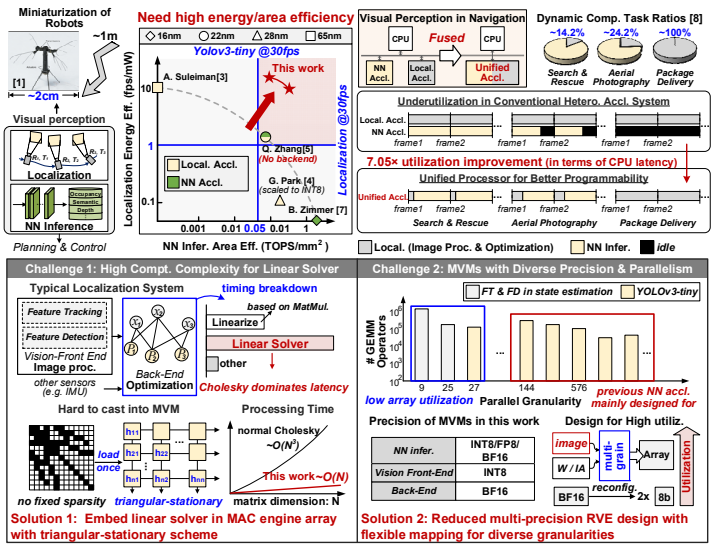
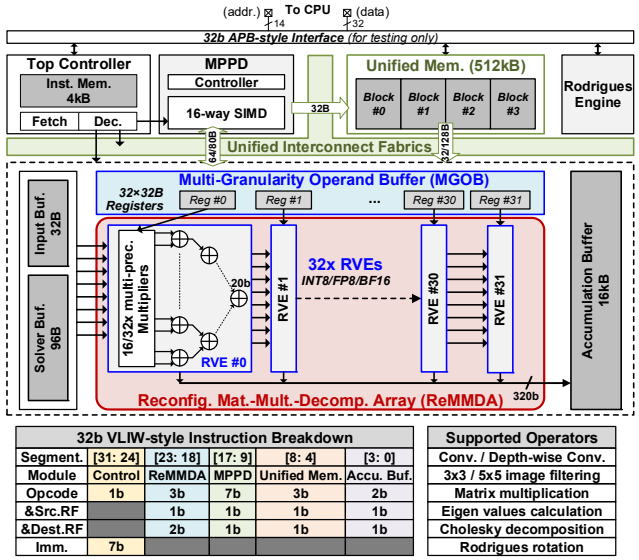


Fig. 1. Limitations of the heterogeneous system and challenges for efficient unified visual-perception processors for microrobots.



32b VLIW-style Instruction Breakdown					
Segment.	[31: 24]	[23: 18]	[17: 9]	[8: 4]	[3: 0]
Module	Control	ReMMDA	MPPD	Unified Mem.	Accu. Buf.
Opcode	1b	3b	7b	3b	2b
&Src.RF		1b	1b	1b	1b
&Dest.RF		2b	1b	1b	1b
Imm.	7b				

Supported Operators
Conv. / Depth-wise Conv.
3x3 / 5x5 image filtering
Matrix multiplication
Eigen values calculation
Cholesky decomposition
Rodrigues rotation

Fig. 2. Overall architecture of the proposed localization-NN-inference-unified processor and instruction set overview.

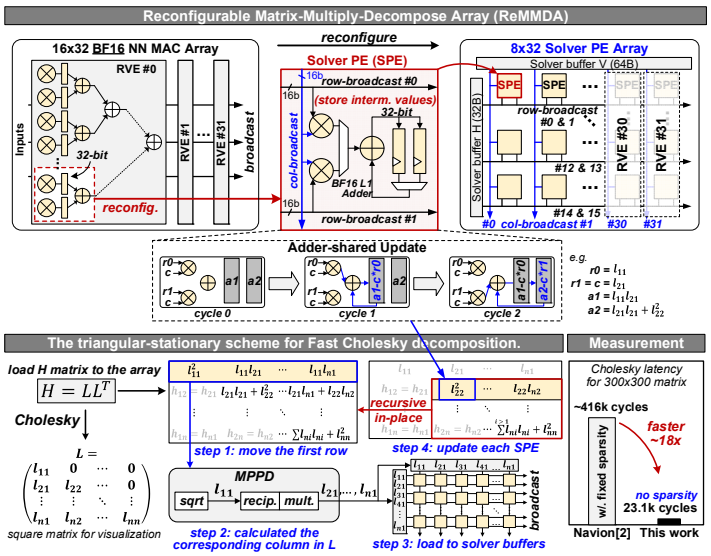


Fig. 3. Array-level fusion of MVM and fast linear solver with the triangular-stationary scheme for Cholesky decomposition.

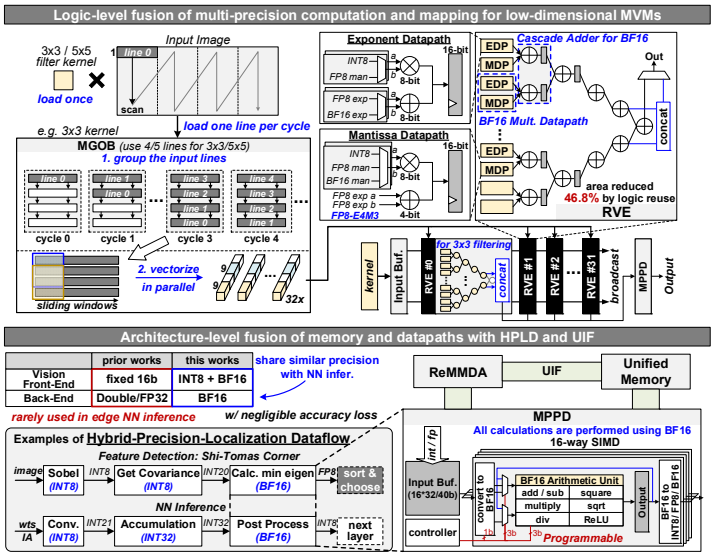
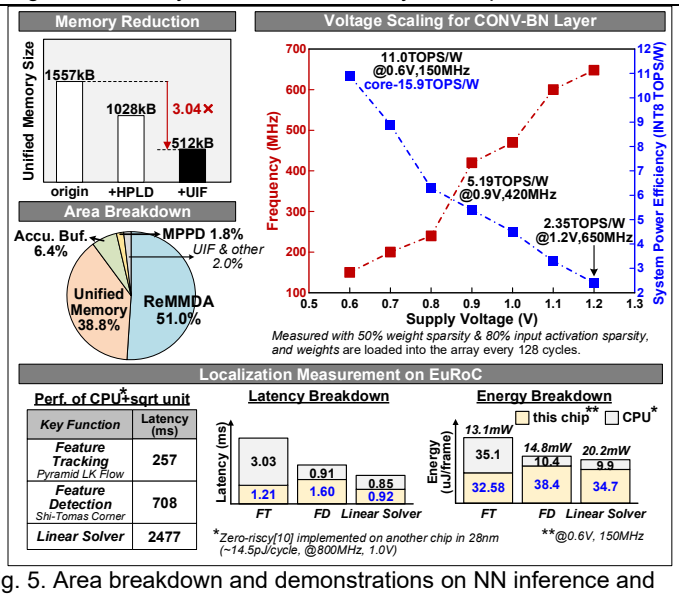


Fig. 4. Logic- and architecture-level fusion with HPLD and UIF for high hardware utilization and area efficiency.



Key Function	Latency (ms)
Feature Tracking Pyramid LK Flow	257
Feature Detection Shi-Tomas Corner	708
Linear Solver	2477

Fig. 5. Area breakdown and demonstrations on NN inference and localization.

Evaluation Results of NN Inference

Model	YOLOv3-tiny	Dataset	COCO2014
mAP50 (official FP32)	36.1%	mAP50 (Ours INT8)	35.8%
Highest Throughput	219fps, 560mW @1.2V, 650MHz	Best Efficiency	33fps, 18.6mW @0.6V,100MHz

Related Trajectory Error on EuRoC

Room #	This Work	Intel i7-12700K
MH_01	0.166%	0.159%
MH_02	0.121%	0.114%
MH_03	0.071%	0.101%
MH_04	0.147%	0.138%
MH_05	0.146%	0.142%
Average	0.147%	0.145%

		JSSC'19[3]	JSSC'20[7]	VLSI'22[5]	ISSCC'24[4]	ISSCC'24[6]	This Work
Basic Information	Technology	65nm	16nm	22nm	28nm	40nm	28nm
	Achitecture	Digital	Digital	Digital	Digital	Digital	Digital
	Application	Visual-Inertial Odometry	NN Inference	MVM-based vision navigation	Nerf-SLAM	Micro Surveillance	Visual Localization & NN Inference for Navigation
	Chip/Core Area (mm²)	20/16.07	6/3.1	8.76/5.89	20.25/18.49	20.25/18.5	3.425/2.1
	On-chip Memory	854kB SRAM	640kB SRAM	1428kB SRAM & 2MB MRAM	1348kB SRAM	760kB SRAM & 5MB RRAM	532kB SRAM
	Frequency (MHz)	62.5/83.3	161-2001	0.059-190	50-200	80-210	150-650
	Supply Voltage (V)	1.0	0.41-1.2	0.5-1.0	0.7-0.9	0.8-1.1	0.6-1.2
	Power (mW)	24	30-4160	0.468-158	303.5	0.11-609.3	24-560[1]
Local.	Tracking Rate (FPS)	171 (752x480)	Not Supported	217 (640x480)	52.6 (640x480)	-	235-302 (752x480)[2]
	Latency of 300x300 Cholesky (ms)	6.7 w/ fixed sparsity	-	No Backend	-	-	0.21-1.36 no sparsity required
NN Inference	NN Data Format	Not Supported	INT8	INT8, INT16	FP16	INT8	INT8, FP8, BF16
	Peak Dense NN Infer. Thru. (GOPS)	-	4.01k INT8	511 INT8 146 INT16	1330 FP16	268.8	307-1331 INT8/FP8 153-665 BF16
	Peak Area Eff. (GOPS/mm²)	-	1.29k INT8	86.7 INT8	71.9 FP16	14.58 INT8	146-633 INT8/FP8 72.8-316 BF16
	Energy Efficiency (TOPS/W)	-	9.52 INT8	12.1 INT8	3.24 FP16	0.84 INT8	3.71-15.9[3](INT8 Core) 2.35-11.0 (INT8 System)
Nav	System Throughput Density (FPS/mm²)	8.95		5.77-19.9 (No Backend)	-	-	62.2-90.9[2]
	System Energy Efficiency (uJ/frame)	149		249 @0.65V (No Backend)	-	-	103-376[2]

1 24mW is measured for feature detection @0.6V, 150MHz, the power of NN inference is 28.7mW @0.6V, 150MHz.
2 Includes control operations running on a zero-riscy CPU [10] @800MHz (1V, ~14.5pJ/cycle) implemented on another 28nm chip w/o parallelism.
3 With 50% weight sparsity & 80% input activation sparsity @0.6V,100MHz, and weights are loaded into the array every 128 cycles.

Fig. 6. Comparison table and accuracy measurement results.

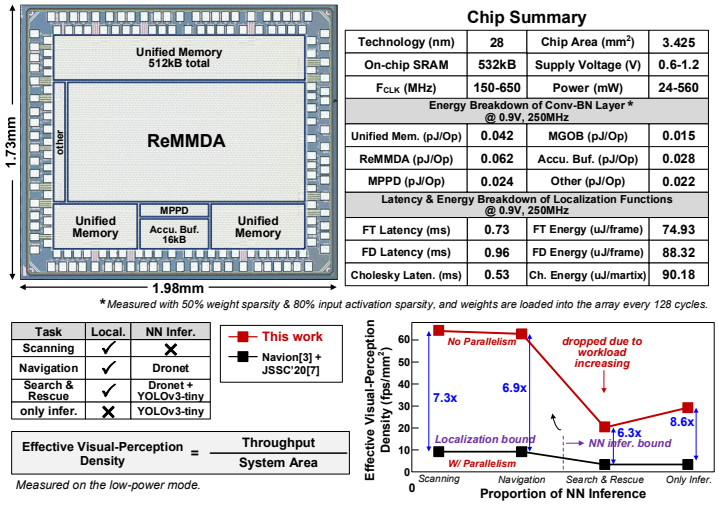


Fig. 7. Die photo, chip summary, and system measurement on multi-tasks.

Additional References:

[5] Q. Zhang *et al.*, "A 22nm 3.5TOPS/W Flexible Micro-Robotic Vision SoC with 2MB eMRAM for Fully-on-Chip Intelligence," VLSI Symp., 2022.

[6] S. D. Spetalnick et al., "30.1 A 40nm VLIW Edge Accelerator with 5MB of 0.256pJ/b RRAM and a Localization Solver for Bristle Robot Surveillance," ISSCC, 2024.

[7] B. Zimmer et al., "A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm," JSSC, 2020.

[8] B. Boroujerdian et al., "The Role of Compute in Autonomous Micro Aerial Vehicles: Optimizing for Mission Time and Energy Efficiency," ACM TOCS, 2022.

[9] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," IEEE T-RO, 2018.

[10] P. D. Schiavone et al., "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," PATMOS, 2017.

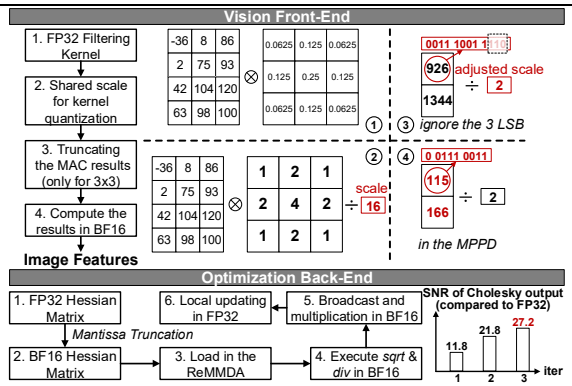


Fig. S1 Quantization flow for the localization, enabling the reported HPLD design in this work. For the vision front-end, the full-precision values of kernels are clamped to fit INT8 range, and the MAC results are truncated to reduce the increased output bandwidth caused by early outputting. For the optimization back-end, the Hessian matrix in FP32 is converted to BF16 format with direct mantissa truncation, and most operations are executed in BF16 (*sqrt*, *div*, broadcast, multiplication). Then, the SPEs maintain and update the intermediate results in FP32 format to preserve the accuracy.

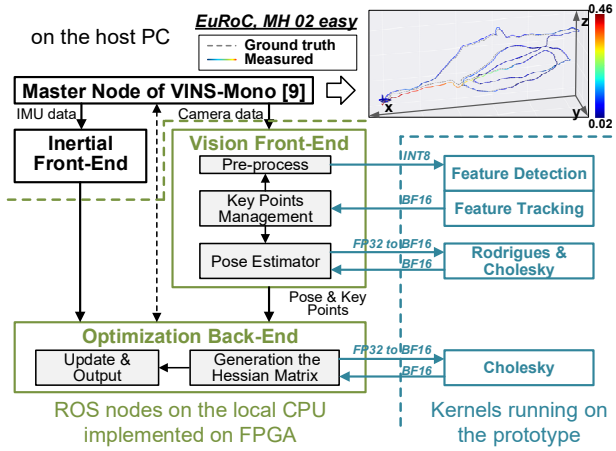


Fig. S2 Tasks mapping for localization demonstration. Computational function kernels run on the proposed VPP, and the front-end and back-end are implemented as two robot operating system (ROS) nodes running on the CPU of a Zynq FPGA. The host PC is in charge of data input and results visualization. The whole system is modified from the official repository of the VINS-Mono project.

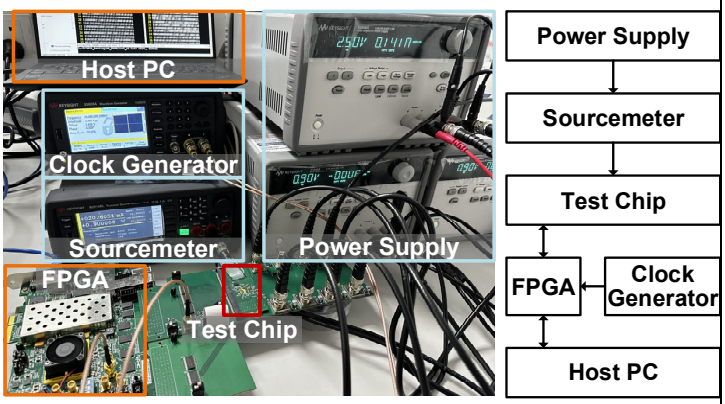


Fig. S3 Illustration of the test setup, showing the prototype is wire-bonded and connected to the FPGA through an adapter board. A customized compiler on the host PC generates instructions and data for chip configuration and execution, and the clock generator provides the clock signal for the FPGA and the test chip. The host PC is connected to the FPGA by Ethernet for data communication. A sourcemeter is inserted for energy measurements.