CrossingFlow: Minimizing Domain Crossings in Heterogeneous Systems Under Error Constraints

Anonymous Author(s)

Working Paper — February 2026

Abstract

Heterogeneous compute systems—combining analog, digital, near-memory, multi-chiplet, and multi-voltage-domain substrates—promise order-of-magnitude efficiency gains. However, these gains are systematically eroded by **domain crossings**: the overhead incurred when data transitions between domains. We formalize a cost decomposition separating intra-domain compute from crossing cost, introduce elasticity ε as a dominance indicator, and verify the resulting Domain Crossing Law across five boundary classes and five technology nodes (3–28nm). Crossing-to-compute cost ratios range from 5× to 32,000× (geometric mean: 73× in pJ/byte). Elasticity reaches $\varepsilon \approx 0.86$–1.00 in crossing-dominated regimes, confirming near-linear energy reduction from crossing elimination. We present CrossingFlow, a compiler that treats crossings as a scarce resource and minimizes crossing volume under error and latency constraints.

1. Introduction

The end of Dennard scaling has driven architecture toward heterogeneous systems combining multiple compute substrates. Analog compute-in-memory (CIM) promises sub-femtojoule MACs; chiplet architectures enable specialized die integration; near-memory processing reduces data movement. Each achieves impressive intra-domain efficiency. Yet system-level gains consistently fall short of component-level projections.

We argue this gap is explained by a single phenomenon: the cost of **domain crossings**. When data crosses any boundary—ADC/DAC, DRAM fetch, inter-die transfer, level-shifter—the system pays 5× to 32,000× more per byte than intra-domain compute. This observation, the **Domain Crossing Law**, has three corollaries: (1) improving intra-domain efficiency yields diminishing returns once crossings dominate; (2) architectural crossing reduction gives near-linear gains; (3) the correct compiler target is crossings, not operations.

Contributions: (i) formal cost decomposition in pJ/byte; (ii) taxonomy of five boundary classes; (iii) elasticity ε as dominance thermometer; (iv) three invariance tests plus ablation; (v) CrossingFlow compiler with multi-domain IR and crossing-minimizing optimizer.

2. The Domain Crossing Law

2.1 Definitions

Domain boundary: any interface where data changes representation, location, or operating constraints. Each boundary b has crossing cost c_b (pJ/byte). **Crossing volume V_b:** total bytes traversing boundary b during execution. **Intra-domain cost C_intra:** energy of operations within a single domain.

2.2 Decomposition

$$C(P,H) = C\_intra(P,H) + \Sigma\_b\ V\_b(P,H) \cdot c\_b(H)$$

This is exact when crossing costs are independent and per-byte cost is constant (burst-amortizable). Both hold within 10–15% for all five boundary types studied.

2.3 Elasticity

$$\varepsilon\_b = \partial\log(C) / \partial\log(V\_b)$$

When crossing cost dominates, ε → 1: every 1% crossing reduction yields ~1% energy reduction. When compute dominates, ε → 0. The elasticity is the decision criterion: optimize V_b when ε is high; optimize c_b when ε is low.

3. Experimental Setup

We study five boundary types with costs from published sources, normalized to pJ/byte at 7nm.

Boundary	c_compute	c_crossing	Ratio	Source
Analog↔Digital	0.0001	**3.20**	**32,000×**	ISSCC 2023, Walden FoM
Memory↔Compute	0.25	**1.25**	**5×**	Horowitz 2014, 7nm scaled
Chiplet↔Chiplet	0.25	**5.00**	**20×**	UCIe 1.0 spec
Near↔Far Memory	0.25	**10.00**	**40×**	Samsung HBM3
Voltage↔Voltage	0.05	**0.80**	**16×**	Multi-Vdd literature

Table 1. Crossing-to-compute cost ratios at 7nm (pJ/byte). Geometric mean: 73×.

Analog↔Digital: 0.1 fJ/MAC (ISSCC 2023 best CIM). One MAC ≈ 1 byte. ADC: Walden FoM 50 fJ/conv-step, 6-bit → 50×2^6 = 3,200 fJ/value = 3.2 pJ/byte. **Memory↔Compute:** Horowitz (2014) scaled to 7nm: ~1.25 pJ/byte DRAM access vs 0.25 pJ/byte compute. **Chiplet:** UCIe short-reach ~0.5 pJ/bit + overhead = ~5 pJ/byte. **HBM:** Samsung HBM3 ~3.9 pJ/bit + controller ≈ 10 pJ/byte. **Level shifter:** ~0.1 pJ/bit = 0.8 pJ/byte.

Canonical workload: 256 KB intra-domain compute, crossing volume swept 256 B to 512 KB.

4. Results

4.1 Test A: Technology Invariance

We repeat across 3nm–28nm. Compute scales with transistor density; crossing scales more slowly (physical phenomena: charge redistribution, signal propagation).

Boundary	**3nm**	**5nm**	**7nm**	**14nm**	**28nm**
Analog↔Digital	56,000×	42,667×	32,000×	20,800×	12,000×
Memory↔Compute	9×	7×	5×	3×	2×
Chiplet↔Chiplet	35×	27×	20×	13×	8×
Near↔Far Memory	70×	53×	40×	26×	15×
Voltage↔Voltage	28×	21×	16×	10×	6×

Table 2. Ratios across technology nodes. Ratios decrease but remain >> 1 at all nodes.

4.2 Test B: Scale Invariance

The tipping point (crossing fraction where crossing energy > 50%) ranges from 0.1% (analog, extreme sensitivity) to 50% (memory, low sensitivity). All fall within realistic workload parameters.

4.3 Test C: Elasticity

Boundary	**Ratio**	**ε crossing**	**ε transition**	**ε compute**
Analog↔Digital	32,000×	**0.996**	—	—
Memory↔Compute	5×	0.763	0.31–0.47	0.01–0.05
Chiplet↔Chiplet	20×	0.838	0.31–0.47	0.03–0.10
Near↔Far Memory	40×	**0.862**	0.31–0.47	0.05–0.10
Voltage↔Voltage	16×	**0.865**	0.26–0.42	0.02–0.08

Table 3. Elasticity ε by regime. In crossing-dominated regimes, ε ≈ 0.86–1.00.

The analog↔digital boundary achieves ε = 0.996: a 50% reduction in conversions yields 49.8% energy reduction. Even memory↔compute, with modest 5× ratio, reaches ε = 0.76 in its crossing regime.

4.4 Ablation: Hardware vs Architecture

Comparing 10× better crossing technology vs 10× fewer crossings: at 25% crossing fraction, both yield comparable gains. The critical insight is that architectural gains compose multiplicatively across boundaries, while technological gains are bounded by physics. In fully crossing-dominated regimes, architecture strictly wins.

5. CrossingFlow Compiler

CrossingFlow operationalizes the Domain Crossing Law. Its IR annotates every tensor with its current domain (analog-voltage, charge, time, digital). Domain transitions are explicit cast operations with costs. A noise-budget type system tracks cumulative error through the graph. The optimizer solves:

$$\min \Sigma_b V_b \cdot c_b \quad \text{s.t.} \quad \text{error} \le \tau, \quad \text{latency} \le L$$

using three strategies: fusion (eliminate inter-layer crossings), domain extension (stay in cheaper domain longer), and batched conversion (amortize cost over accumulated results).

6. Discussion

When the law breaks down. In compute-bound workloads with high arithmetic intensity and minimal data movement ($\varepsilon \approx 0$), crossing reduction is ineffective. The law correctly identifies this regime via the elasticity metric. Such workloads are increasingly rare in modern AI applications.

Post-Moore implications. The law reframes the research agenda: instead of better components (ADCs, compute dies, memory cells), the highest-leverage investments are in reducing crossing frequency through architectural co-design and novel interface technologies (time-domain encoding, charge-domain accumulation).

Limitations. Our model assumes linear per-byte costs. In practice, crossings have setup costs and non-linear burst scaling. Contention effects are not modeled. All results are simulation-based; silicon validation remains future work.

7. Related Work

The energy of data movement was quantified by Horowitz (2014), showing DRAM access dominates compute. This drove near-memory and processing-in-memory architectures. We generalize from one boundary type to all domain crossings. In analog CIM, the ADC bottleneck is well documented (60–80% of accelerator energy/area). ADC-less architectures (rTD-CiM, HCiM) address this at the hardware level; CrossingFlow provides the software counterpart. Heterogeneous compilers (MLIR, TVM, Halide) focus on scheduling and layout; we extend this with crossing-first optimization and noise-budget types.

8. Conclusion

We have presented the Domain Crossing Law: heterogeneous system efficiency is dominated by crossing count under error constraints. Across five boundaries and five nodes, crossing costs exceed compute by 5–32,000×. Architectural crossing elimination yields near-linear gains ($\varepsilon \approx 0.86$–1.00). CrossingFlow operationalizes this as a compiler that minimizes crossing volume.

The correct unit of optimization in heterogeneous computing is not the operation—it is the crossing.

References

[1] M. Horowitz, “Computing’s Energy Problem,” ISSCC, 2014.

[2] A. Shafiee et al., “ISAAC: A CNN Accelerator with In-Situ Analog Arithmetic,” ISCA, 2016.

[3] UCIe Consortium, “UCIe 1.0 Specification,” 2022.
[4] Samsung, “HBM3 Product Specification,” 2023.
[5] B. Murmann, “ADC Performance Survey 1997–2024,” Stanford.
[6] W. Wan et al., “Compute-in-Memory Chip Based on RRAM,” Nature, 2022.
[7] X. Peng et al., “DNN+NeuroSim V2.0,” IEEE TCAD, 2021.