A Reconfigurable Accelerator with Data Reordering Support for Low-Cost On-Chip Dataflow Switching

Jianming Tong, Anirudh Itagi, Tushar Krishna

jianming.tong@gatech.edu, tushar@ece.gatech.edu

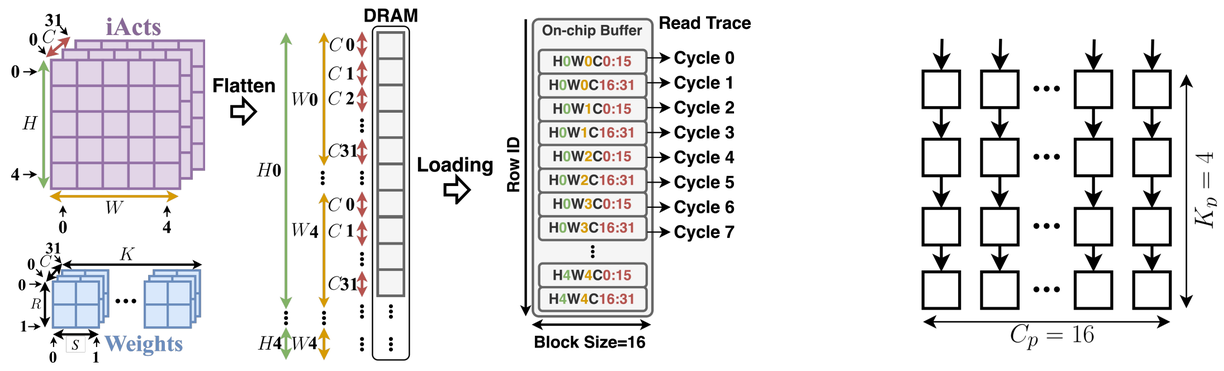
Georgia Institute of Technology

Paper

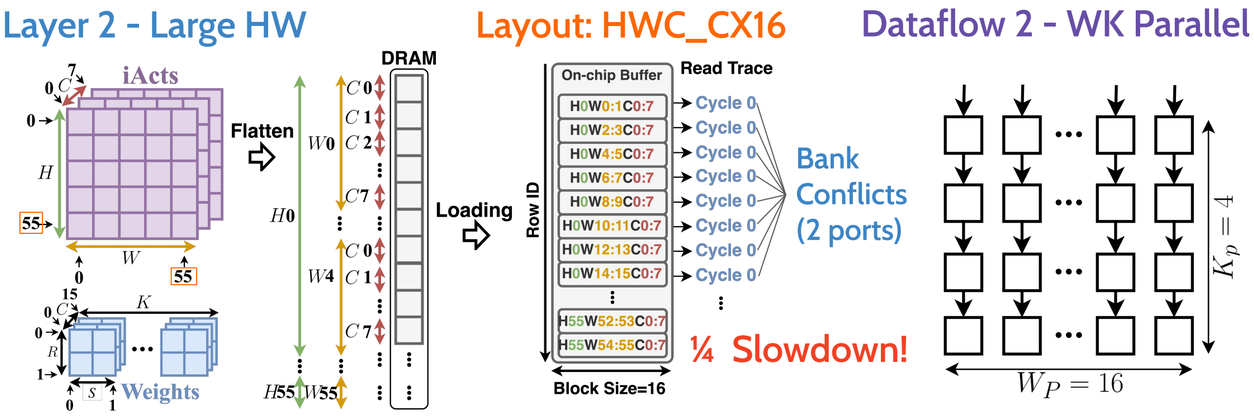
Motivation

1.ML inference boils down to processing different dataflows.
2.No global optimal dataflow. Different dataflows have different accessing patterns, requiring extra data reordering.
3. “Data Layout” - “Dataflow” mismatch leads to 30X slowdown.

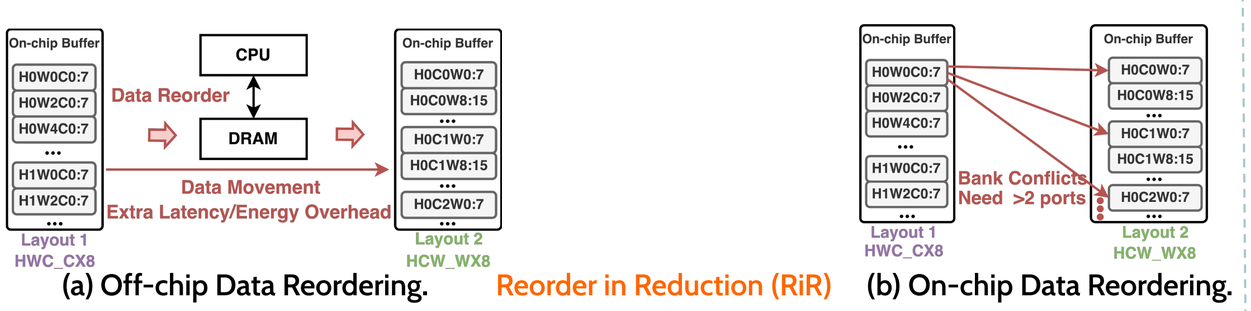
Layer 1 - Large C | Layout: HWC_CX16 | Dataflow 1 - CK Parallel



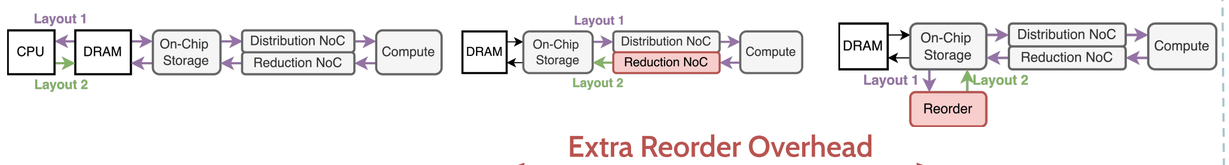
Layer 2 - Large HW | Layout: HWC_CX16 | Dataflow 2 - WK Parallel



3. change data layout requires data reordering, incurring overheads.



(a) Off-chip Data Reordering. Reorder in Reduction (RiR) (b) On-chip Data Reordering.



Extra Reorder Overhead

Read (layout 1)→Compute&Reduction→Write (layout 1)→Read & Reorder→Write (layout 2)

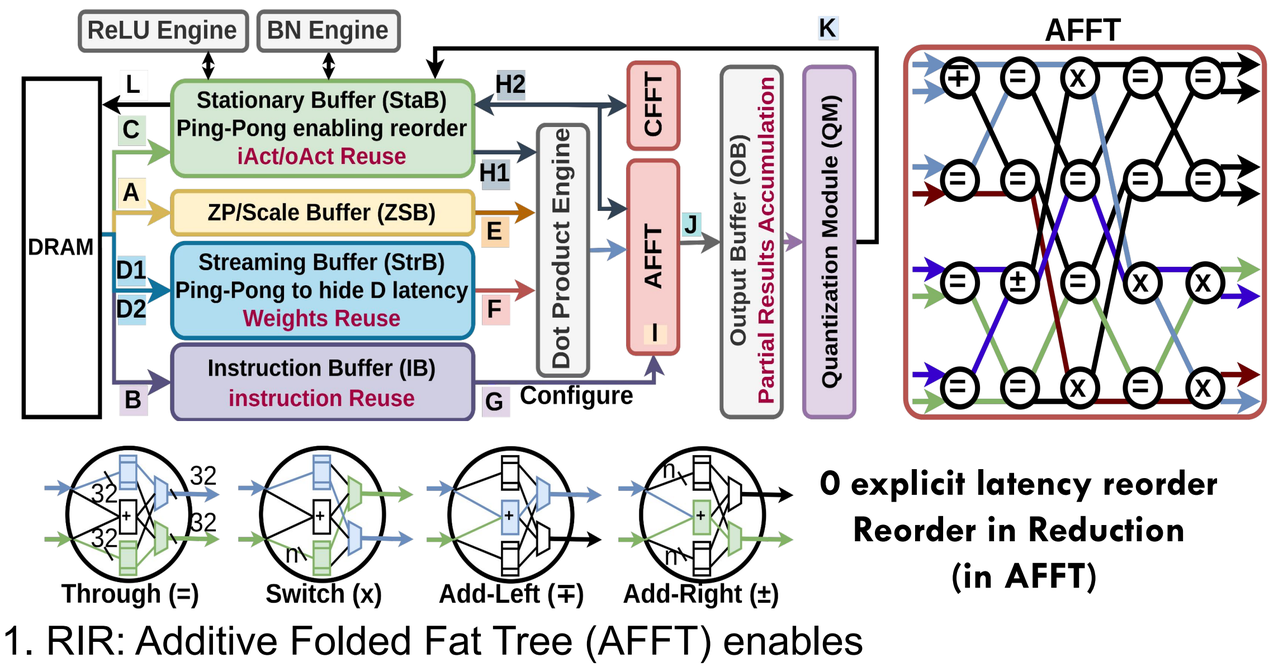
Read & Write On-chip Buffer Twice!

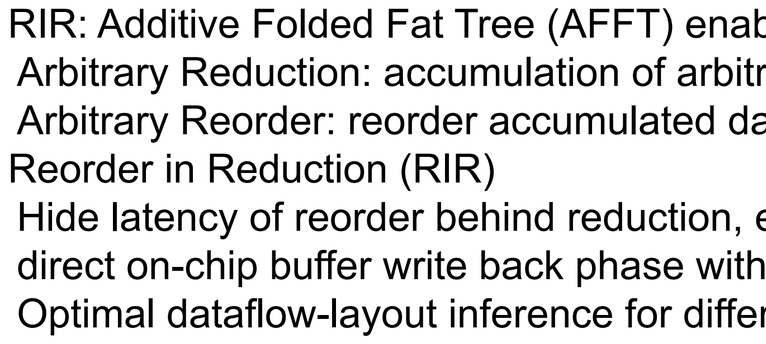
Read (layout 1)→Compute&Reorder in Reduction→Write (layout 2)

Challenges

1. **Hardware**: High reordering overheads restrict hw to compromise and adopt fixed suboptimal dataflows across different workloads.
2. **Layout Modeling:** No systematic data layout modeling method, and thus no dataflow exploration with layout consideration.

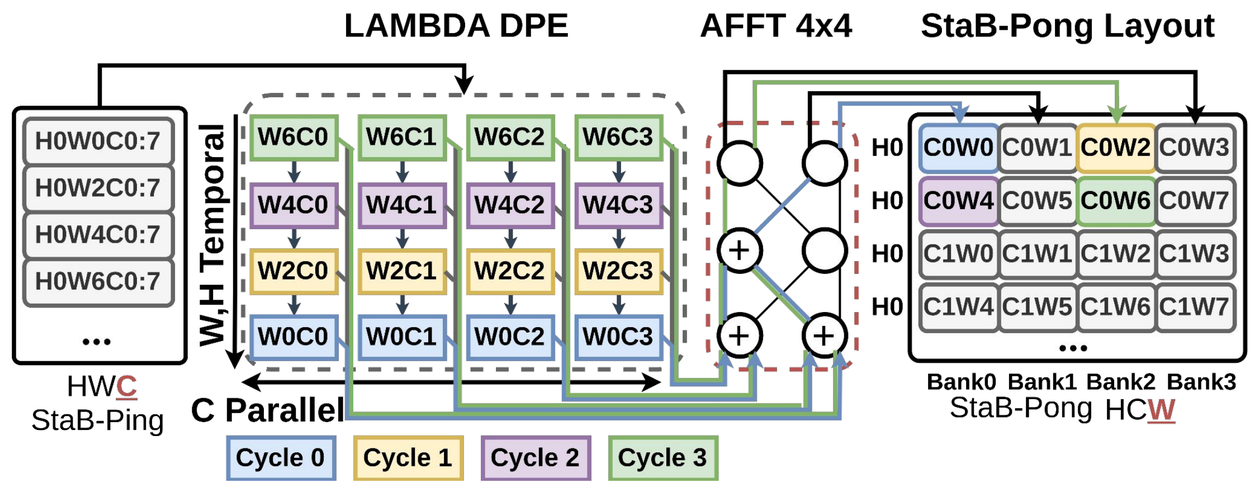
LAMBDA Solutions





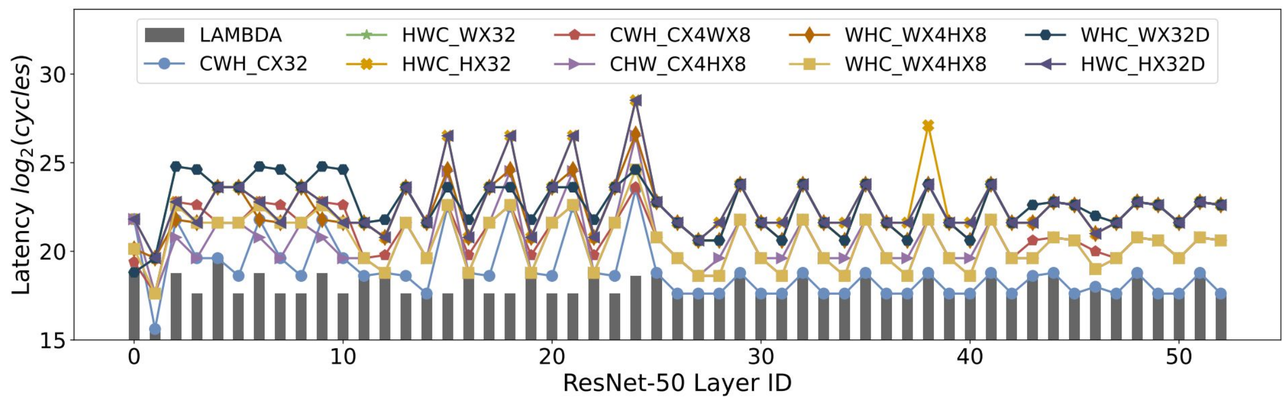
0 explicit latency reorder
Reorder in Reduction
(in AFFT)

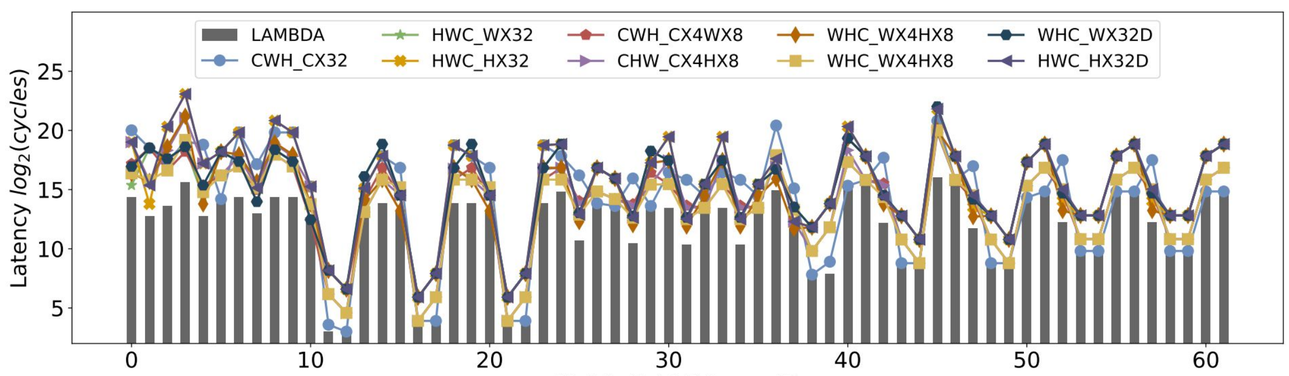
1. RIR: Additive Folded Fat Tree (AFFT) enables
   - Arbitrary Reduction: accumulation of arbitrarily selected inputs data.
   - Arbitrary Reorder: reorder accumulated data to arbitrary output ports.
2. Reorder in Reduction (RIR)
   - Hide latency of reorder behind reduction, enabling layout switching in direct on-chip buffer write back phase with negligible reorder costs.
   - Optimal dataflow-layout inference for different layers.



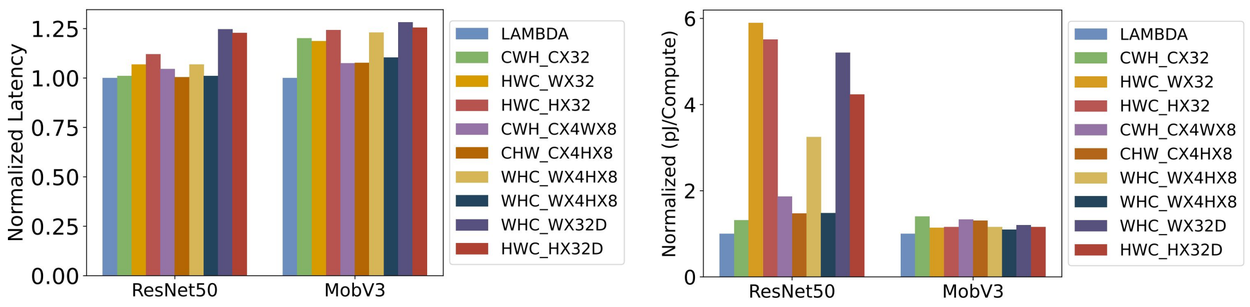
1. Only one DPE row sent output to AFFT per cycle.
   - AFFT resources overhead is amortized by all DPE rows.
2. Top-down store-and-forward enabling iActs/Weights Reuse.
3. Ping-poing StaB to enable layout-dataflow coswitching.

Experiments

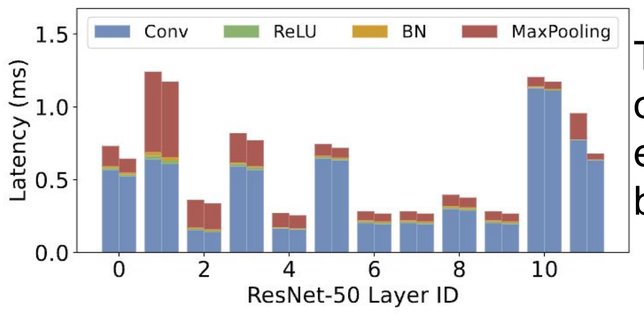




Takeaway 1: “Dataflow”-“layout” mismatch -> 2~30X slowdown.
Takeaway 2: No global optimal layout-dataflow combination.



Takeaway 3: Compared to (optimal dataflow under fixed layout) (optimal dataflow, optimal layout) speedup 1%~24.3%, and achieves 1.32~5.5X higher energy efficiency.



Takeaway 4: RIR saves 2.7~40.8% off-chip reordering latency for eliminated costs of moving data between on-chip buffers and DRAM,

	AFFT 16 × 16	CFFT16 × 16	LAMBDA DPE 16 × 9	ReLU Engine (64)	BN Engine (64)
LUT	17644 (24.5%)	7562 (10.5%)	37430 (57.8%)	4300 (6%)	5073(17988) (7%)
FF	21057 (17%)	9237(7.5%)	88592(71.6%)	1326 (1%)	3520(22029) (2.8%)
BRAM36E2	12 (6%)	12 (6%)	2(1%)	0	2(2) (1%)
BRAM18E2	1 (33.33%)	1 (33.33%)	1 (33.33%)	0	0(0)
URAM	24 (13.6%)	24(13.6%)	32(18.2%)	0	0(0)
DSP	0	0	1297(91%)	0	128(64) (9%)

Takeaway 5: AFFT consumes 24% resource from amortizing cost over row.

Conclusions

1. Switching dataflows should reorder data layout correspondingly.
   - Dataflow-layout mismatch leads to 2~30X slowdown in real system.
   - Reorder incurs extra overhead, overshadowing switching benefits.
2. Reorder in Reudction in LAMBDA hides reorder behind reduction, enabling 1~24.3% speedup and 1.32~5.5X higher energy efficiency
3. LAMBDA enable serving workloads using optimal dataflows-layout.
   - Proposed AFFT consumes 24% resources deliver 24.3% speedup.