UFO-MAC: A Unified Framework for Optimization of High-Performance Multipliers and Multiply-Accumulators

Dongsheng Zuo, Jiadongzhu, Chenglin Li, Yuzhe Ma

The Hong Kong University of Science and Technology (Guangzhou)

IEEE/ACM 2024 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN ICCAD 43rd Edition

Table of Contents

Introduction

Datapath Optimization in AI Field

- Multipliers and MACs are widely adopted in various circuits, especially in the AI field.

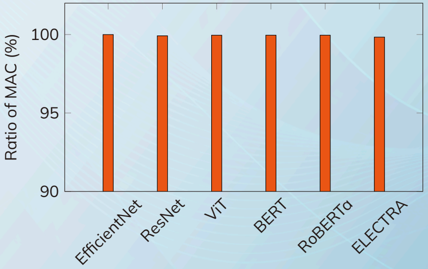
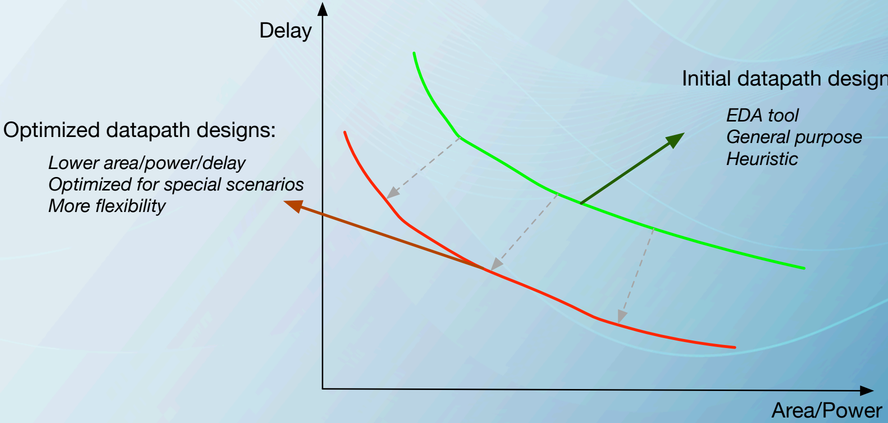


Figure 1: Ratios of MAC computations in various neural networks.

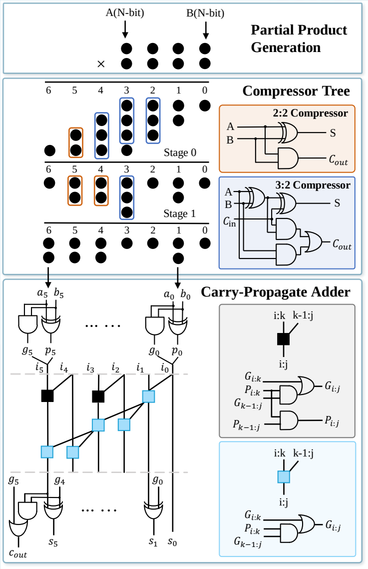
Datapath optimization is critical to improve performance of AI chips

Datapath Circuits Design Optimization

- Goal: Move the Pareto frontier for datapath circuits
  - Achieve better PPA.
  - Optimize for all scenarios.
- Optimizing datapath is non-trivial due to the huge design space.



Multiplier Architecture



Three part:

- Partial product generator (PPG)
- Compressor tree (CT)
  - Compress the partial products (PPs) into two rows.
- Carry propagate adder (CPA)
  - Addition of the two rows.

Previous Works on Multiplier Optimization

- Regular structures:
  - Wallace tree[1], Dadda tree[2]

Limitations

- PPA may not meet specifications.

[1] C. S. Wallace, **"A suggestion for a fast multiplier,"** *IEEE Transactions on Electronic Computers*, 1964.
[2] L. Dadda, **"Some schemes for fast serial input multipliers,"** in *1983 IEEE 6th Symposium on Computer Arithmetic (ARITH)*, 1983.

Previous Works on Multiplier Optimization

- Regular structures:
  - Wallace tree[1], Dadda tree[2]

Limitations

- PPA may not meet specifications.

- Manual custom designs

Limitations

- Large engineering efforts.
- Optimized only for specific process or scenario.

[1] C. S. Wallace, **"A suggestion for a fast multiplier,"** *IEEE Transactions on Electronic Computers*, 1964.
[2] L. Dadda, **"Some schemes for fast serial input multipliers,"** in *1983 IEEE 6th Symposium on Computer Arithmetic (ARITH)*, 1983.

Previous Works on Multiplier Optimization

- Mathematical Programming:
  - GOMIL[3]

Limitations

- Area-optimal may not guarantee timing-optimal.

[3]W. Xiao *et al.*, **"Gomil: Global optimization of multiplier by integer linear programming,"** in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2021.
[4]D. Zuo *et al.*, **"Rl-mul: Multiplier design optimization with deep reinforcement learning,"** in *ACM/IEEE Design Automation Conference (DAC)*, 2023.

Previous Works on Multiplier Optimization

- Mathematical Programming:
  - GOMIL[3]

Limitations

- Area-optimal may not guarantee timing-optimal.

- Reinforcement learning (RL) approach:
  - RL-MUL[4]

Limitations

- RL may lead to local-optimal.
- CPAs are not optimized.

[3]W. Xiao *et al.*, **“Gomil: Global optimization of multiplier by integer linear programming,”** in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2021.

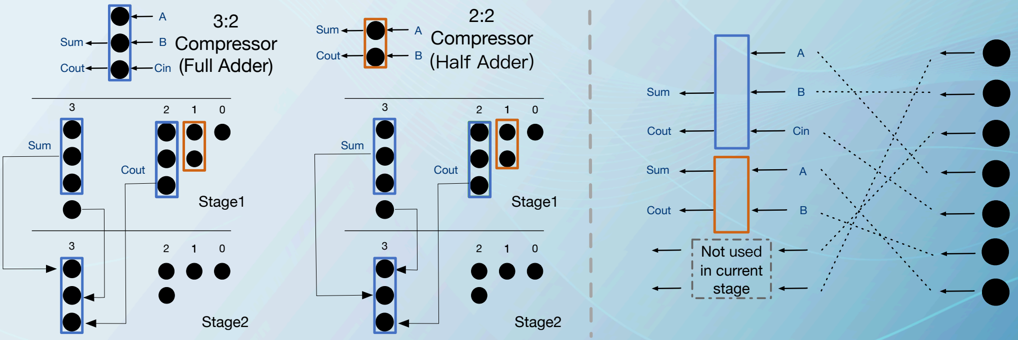
[4]D. Zuo *et al.*, **“Rl-mul: Multiplier design optimization with deep reinforcement learning,”** in *ACM/IEEE Design Automation Conference (DAC)*, 2023.

Motivation

Motivating Example 1: Neglected Design Space in Compressor Trees

Bijection mapping in CT

At each stage *i* column *j* in CT, we need to determine the bijection of partial products and compressor input ports.



(a) Different interconnection order

(b) Bijection mapping in interconnection

Motivating Example 1: Neglected Design Space in Compressor Trees (Cont')

- The interconnection between compressors may impact the delays.
  - There are fast input/output ports in compressors.
- We assign 10,000 random order for same CT structure.
  - The critical path delay varies over 10%.

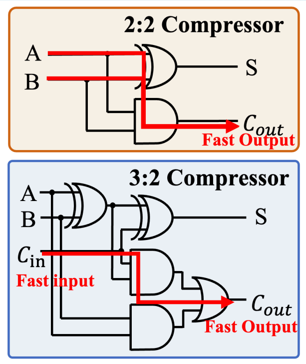


Figure 2: Fast input and out of compressors

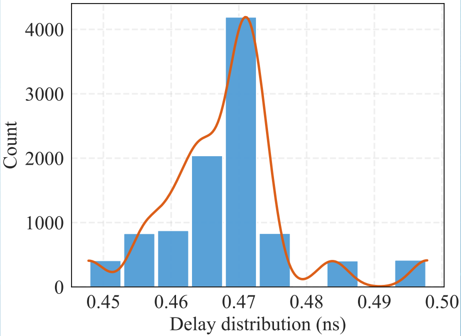
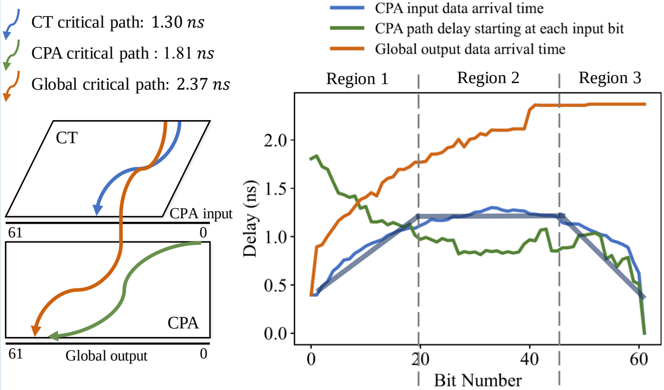


Figure 3: critical path delay distribution among different interconnection order

Motivating Example 2: Non-uniform Arrival Time of Final Adder

The input arrival time profile of CPA can be split into 3 regions:



- The optimization of the CT and the CPA are coupled.
- CPA exhibits a non-uniform arrival time profile.
- CPA requires a different optimization methodology from traditional adders.

Major Contributions of UFO-MAC

- A unified optimization framework for multipliers and MACs.
  - ILP for CT optimization.
  - Efficient heuristic for CPA optimization.

Major Contributions of UFO-MAC

- A unified optimization framework for multipliers and MACs.
  - ILP for CT optimization.
  - Efficient heuristic for CPA optimization.
- Extend the design space of compressor tree.
  - Compression stage assignment.
  - Interconnect between compressors.

Major Contributions of UFO-MAC

- A unified optimization framework for multipliers and MACs.
  - ILP for CT optimization.
  - Efficient heuristic for CPA optimization.
- Extend the design space of compressor tree.
  - Compression stage assignment.
  - Interconnect between compressors.
- Explicitly explore the non-uniform arrival profile of CPA.
  - Area-efficiency initial structure
  - FDC timing model
  - Timing driven optimization

UFO-MAC optimization framework

- Generate an optimized compressor tree.
- Final adder is optimized based on the non-uniform arrival time profile.

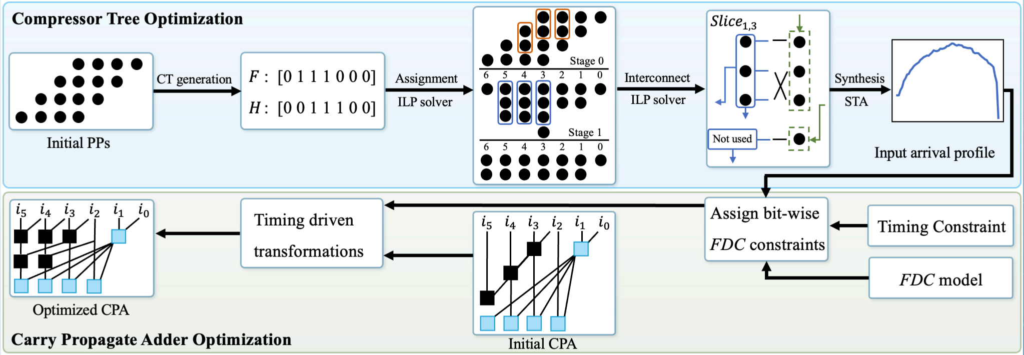
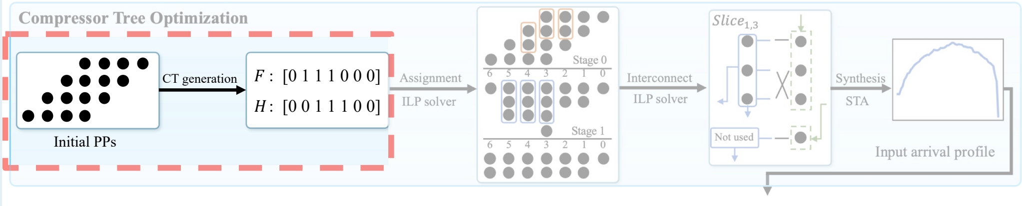


Figure 4: UFO-MAC framework

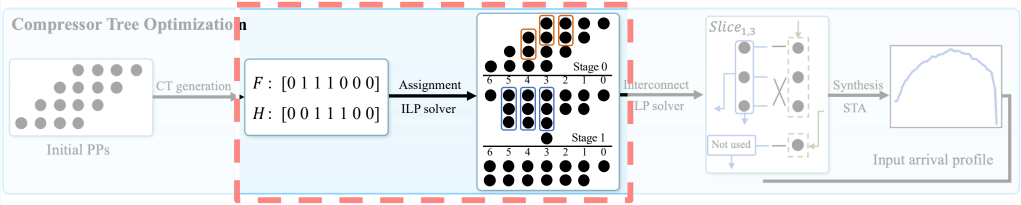
Optimization of Compressor Tree

Initial Compressor Tree Generation



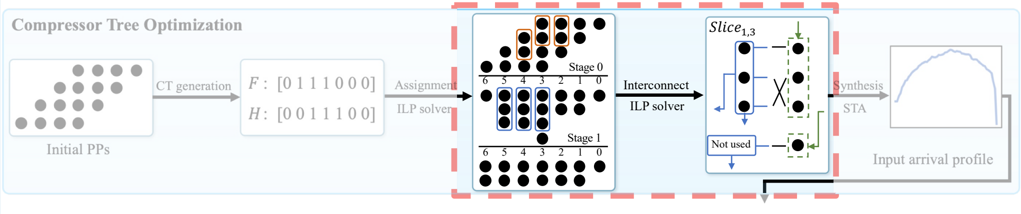
- In this stage we get F_j and H_j after initial CT generation
  - F_j, H_j: Total 3:3 and 2:2 compressor numbers at column j
- Basic strategy:
  - In each column, reduce the partial products to two remaining bits.
  - 2:2 compressor has low compression efficiency.
    - Maximize the use of 3:2 compressors.
    - Only use 2:2 compressors to adjust parity.

ILP for Stage Assignment



- We only get F_j and H_j after initial CT generation
  - F_j, H_j: Total 3:3 and 2:2 compressor numbers at column j
- We need to assign the compressors to corresponding stages
  - We apply ILP to assign compressors while minimizing the total stage count.

ILP for Interconnection Order Optimization



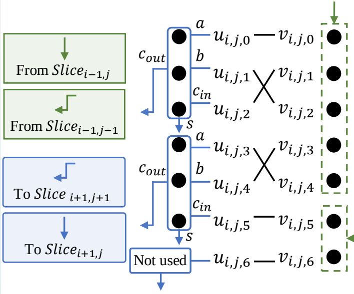
- Determine a bijective mapping at each stage *i* and column *j*.
  - Model data arrival time at each port.
  - Use ILP to optimize interconnection for minimum critical path delay.

ILP for Interconnection Order Optimization (Cont')

- At each stage i column j, we introduce matrix $z_{i,j}$ to indicate the bijection between sink vector $u_{i,j}$ and source vector $v_{i,j}$:

$$v = u \quad \text{if and only if} \quad z_{i,j,u,v} = 1$$

$$\sum_{v=0}^{m-1} z_{i,j,u,v} = 1 \quad \forall u, \quad \sum_{u=0}^{m-1} z_{i,j,u,v} = 1 \quad \forall v$$



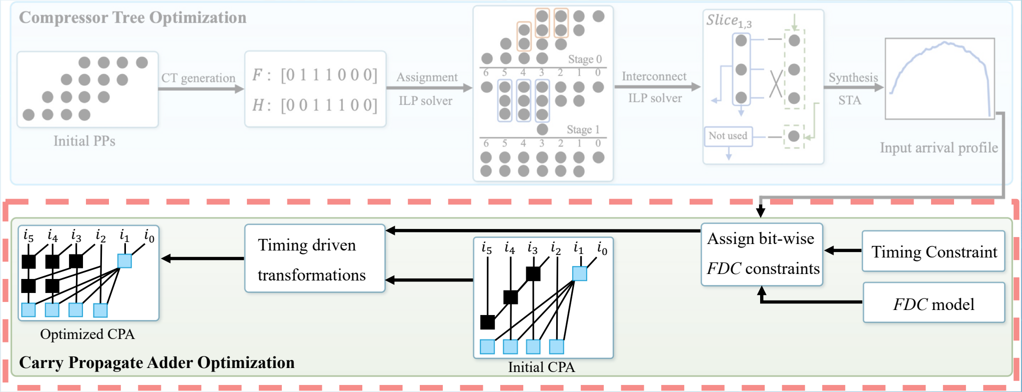
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Interconnect matrix $z_{i,j}$

- The ILP objective function is the critical path delay.

Optimization of Carry Propagate Adder

Optimization of Final Adder

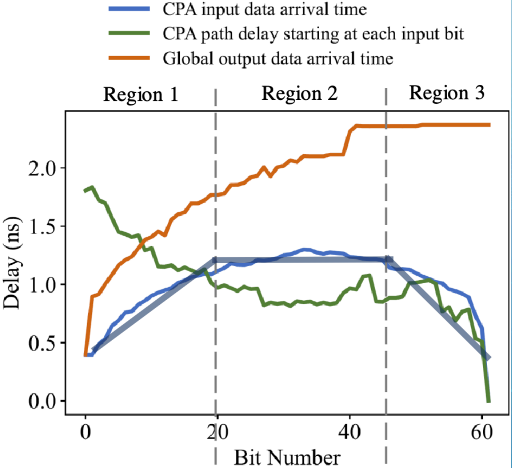


- Utilize the input arrival time profile.
- Initial prefix structure selection.
  - Area-efficient structures.
- Timing model to timing-driven transformation on prefix structure.

Initial Prefix Structure

Area-efficient structures are selected for the initial structure:

- *Region 1:* Ripple Carry Adder
- *Region 2:* Sklansky Adder
- *Region 3:* Carry Increment Adder



Timing Model for Prefix Adder

- A timing model with high fidelity is essential to guide timing-driven opt.
- Timing models in previous works:
  - Logic depth based[5]
  - Max-path-fanout (mpfo)[6]

However, both logic depth and fanout may affect the timing.

- We introduce **fanout depth combination (FDC)** model:

$$d_i = k_0 \times F_{black} + k_1 \times F_{blue} + k_2 \times N_{black} + k_3 \times N_{blue} + b \tag{1}$$

Here, k_0, k_1, k_2, k_3, and b are coefficients that can be determined to fit the model.

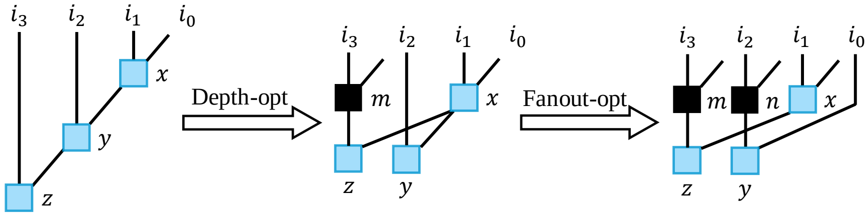
[5] R. Zimmermann, **"Non-heuristic optimization and synthesis of parallel-prefix adders,"** in *IWLAS*, 1996.

[6] Y. Ma *et al.*, **"Cross-layer optimization for high speed adders: A pareto driven machine learning approach,"** *TCAD*, 2019.

Timing-driven Prefix Graph Optimization

Timing-driven transformations to meet the timing constraints:

- Depth-optimization transformations.
- Fanout-optimization transformations.



Experimental Results

Experimental Setup

- Commercial synthesis tool
- Nangate 45nm Open Cell Library
- Comparison:
  - Compressor tree
  - Multiplier
  - MAC
- Implemented in larger functional modules for further validation
  - 5-stage FIR filter
  - Systolic array

Baselines

- ***GOMIL***: ILP based approach [Xiao+, DATE'21]
- ***RL-MUL***: state-of-the-art RL based approach [Zuo+, DAC'23]
- ***Commercial IPs***

Multiplier Comparison

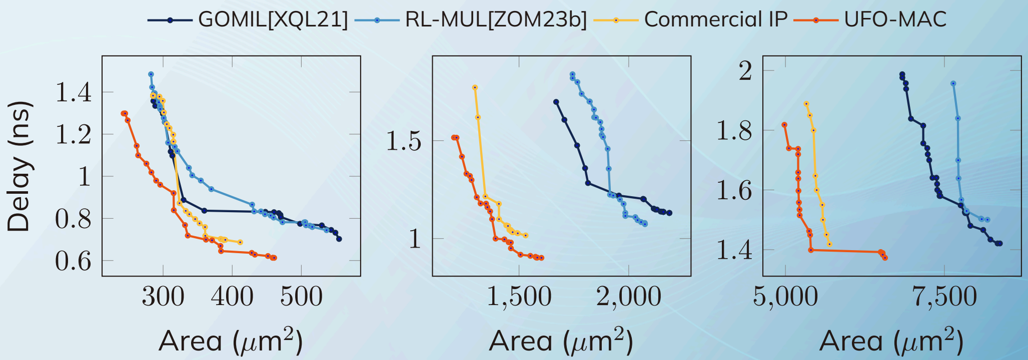


Figure 5: Pareto-frontiers of the synthesized results on multipliers. From left to right: 8-bit; 16-bit; 32-bit.

MAC Comparison

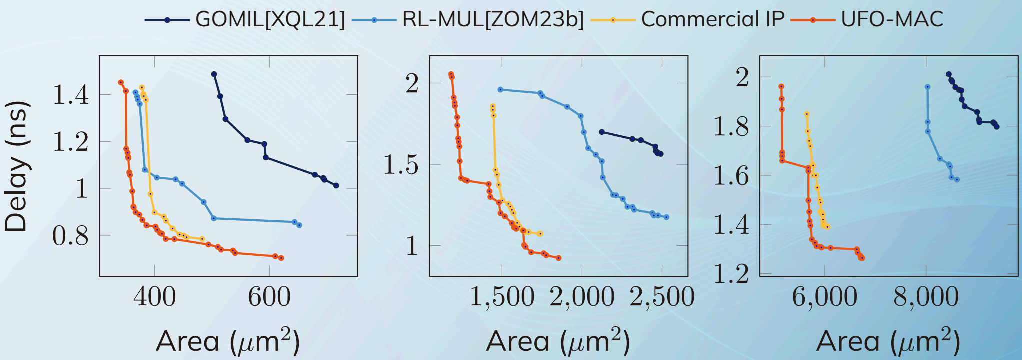


Figure 6: Pareto-frontiers of the synthesized results on MACs. From left to right: 8-bit; 16-bit; 32-bit.

FIR Filter Comparison

Table 1: FIR filter comparison.

Constraint	Method	8-bit				16-bit				32-bit			
		Freq (Hz)	WNS (ns)	Area (μm^2)	Power (mW)	Freq (Hz)	WNS (ns)	Area (μm^2)	Power (mW)	Freq (Hz)	WNS (ns)	Area (μm^2)	Power (mW)
Area-driven	GOMIL[XQL21]	660M	-0.4968	2354	1.5663	500M	-0.4990	9405	8.7474	400M	-0.4993	33804	36.584
	RL-MUL[ZOM23b]		-0.3525	2318	1.4298		**-0.4989**	8752	8.7020		**-0.5008**	38022	44.264
	Commercial IP		-0.1805	2358	1.3137		**-0.4989**	8397	6.9946		-0.6533	31900	35.302
	UFO-MAC		**-0.1188**	**1915**	**1.0934**		-0.5707	**6429**	**5.8867**		-0.5486	**29820**	**32.836**
Timing-driven	GOMIL[XQL21]	2G	-0.6287	3284	2.5342	1G	-0.6303	11112	12.004	660M	-0.5085	38167	46.405
	RL-MUL[ZOM23b]		-0.5115	3067	2.3223		-0.4992	10572	10.872		-0.4999	38898	45.361
	Commercial IP		-0.5205	2919	2.0671		-0.4477	8518	**7.3785**		-0.4994	32183	**35.715**
	UFO-MAC		**-0.4893**	**2733**	**1.7796**		**-0.4277**	**8394**	7.4621		**-0.4808**	**32127**	35.980
Trade-off	GOMIL[XQL21]	1G	-0.5468	2757	1.8771	660M	-0.4662	10373	10.615	500M	-0.4266	35372	40.126
	RL-MUL[ZOM23b]		-0.2998	2718	1.9156		-0.3976	10215	10.315		-0.5039	38245	44.211
	Commercial IP		-0.3486	2495	**1.4829**		-0.3493	8418	7.0109		-0.4360	31510	34.551
	UFO-MAC		**-0.2623**	**2349**	1.5419		**-0.3137**	**7658**	**6.4801**		**-0.3883**	**31366**	**34.217**

Systolic Array Comparison

Table 2: Systolic array comparison.

Constraint	Method	8-bit				16-bit			
		Freq (Hz)	WNS (ns)	Area (μm^2)	Power (mW)	Freq (Hz)	WNS (ns)	Area (μm^2)	Power (mW)
Area-driven	GOMIL[XQL21]	660M	-0.5102	168370	11.572	400M	-0.4976	559985	35.918
	RL-MUL[ZOM23b]		**-0.4239**	135659	10.207		-0.5102	436095	41.480
	Commercial IP		-0.4684	136529	10.393		-0.4828	438526	40.506
	UFO-MAC		-0.4974	**125334**	**9.2475**		**-0.4697**	**401782**	**35.762**
Timing-driven	GOMIL[XQL21]	2G	-0.9827	190381	12.193	1G	-0.9854	662801	44.912
	RL-MUL[ZOM23b]		-0.7077	172810	11.873		-0.5856	609563	44.275
	Commercial IP		-0.6053	144137	11.357		-0.3375	**467621**	45.221
	UFO-MAC		**-0.5946**	**138316**	**10.787**		**-0.1994**	533072	**40.164**
Trade-off	GOMIL[XQL21]	1G	-0.6842	178874	11.175	660M	-0.6611	611143	41.651
	RL-MUL[ZOM23b]		-0.6955	141754	10.892		-0.0981	564192	43.515
	Commercial IP		-0.6941	141905	10.831		-0.0999	458647	45.077
	UFO-MAC		**-0.6785**	**131083**	**9.5777**		**-0.0182**	**449184**	**36.205**

Conclusion

Conclusion

- A unified optimization framework for multipliers and MACs
- Extended design space
- Targeted optimization for CPA

Conclusion

- A unified optimization framework for multipliers and MACs
- Extended design space
- Targeted optimization for CPA

Future Works

- Extend to larger modules: AI Accelerator, FPU.
- Power Optimization.

Conclusion

- A unified optimization framework for multipliers and MACs
- Extended design space
- Targeted optimization for CPA

Future Works

- Extend to larger modules: AI Accelerator, FPU.
- Power Optimization.

Questions are welcomed!

Reference I

[1] C. S. Wallace, **"A suggestion for a fast multiplier,"** *IEEE Transactions on Electronic Computers*, 1964.

[2] L. Dadda, **"Some schemes for fast serial input multipliers,"** in *1983 IEEE 6th Symposium on Computer Arithmetic (ARITH)*, 1983.

[3] W. Xiao, W. Qian, and W. Liu, **"Gomil: Global optimization of multiplier by integer linear programming,"** in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2021.

[4] D. Zuo, Y. Ouyang, and Y. Ma, **"Rl-mul: Multiplier design optimization with deep reinforcement learning,"** in *ACM/IEEE Design Automation Conference (DAC)*, 2023.

[5] R. Zimmermann, **"Non-heuristic optimization and synthesis of parallel-prefix adders,"** in *IWLAS*, 1996.

Reference II

[6] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, **"Cross-layer optimization for high speed adders: A pareto driven machine learning approach,"** *TCAD*, 2019.

[7] D. Zuo, Y. Ouyang, and Y. Ma, **"Rl-mul: Multiplier design optimization with deep reinforcement learning,"** in *ACM/IEEE Design Automation Conference (DAC)*, 2023.