HeteroSVD: Efficient SVD Accelerator on Versal ACAP with Algorithm-Hardware Co-Design

Xinya Luan[1], Zhe Lin[2], Kai Shi[1], Jianwang Zhai[1] and Kang Zhao[1]

[1] Beijing University of Posts and Telecommunications
[2] Sun Yat-sen University

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA sigda

Contents

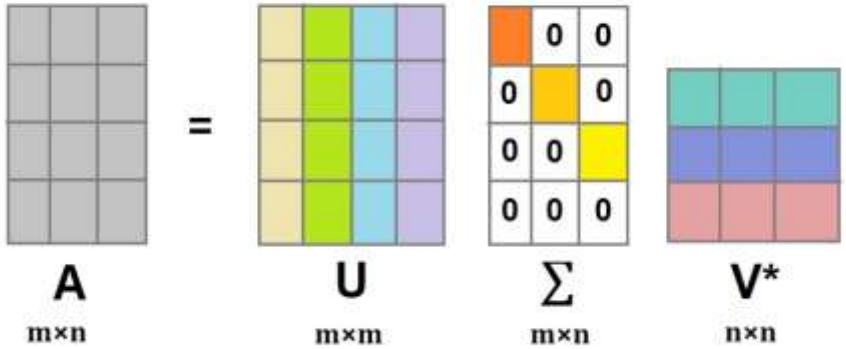
- Introduction
- Methodology
- Evaluation
- Conclution

Introduction

SPONSORED BY CEDA® sigda

Singular Value Decomposition

Hestenes-Jacobi Algorithm



Orthogonalization

$$[B_i, B_j] = [A_i, A_j] \cdot \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, where\ B_i^T \cdot B_j = 0,$$

$$c = \frac{1}{\sqrt{1+t^2}}, \text{ and } s = a_i^T a_j \frac{tc}{|a_i^T a_j|},$$

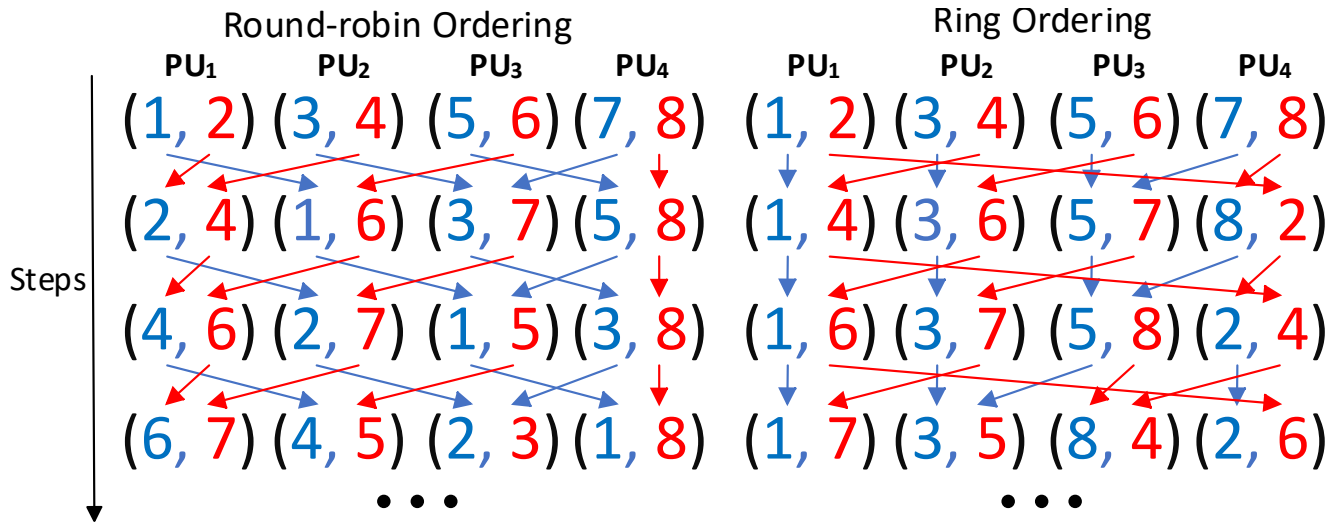
$$t = \frac{\text{sign}(\tau)}{|\tau| + \sqrt{1+\tau^2}}, \text{ and } \tau = \frac{a_j^T a_j - a_i^T a_i}{2|a_i^T a_j|}.$$

$$\frac{B_i^T B_j}{\sqrt{(B_i^T B_i)(B_j^T B_j)}} < precision.$$

Normalization

$$\Sigma = \sqrt{B^T B}, \quad U = B/\Sigma.$$

Orthogonalization in parallel



Singular Value Decomposition

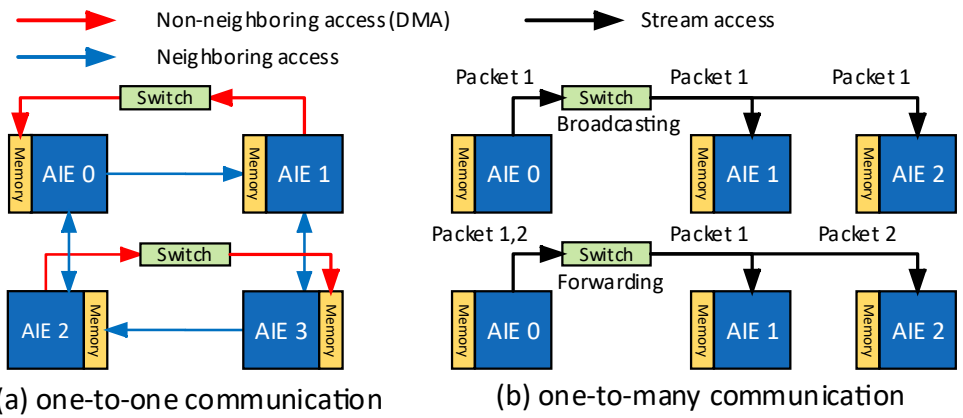
Exsiting Solutions

Hard to jointly optimize latency, throughput and power consumption

- **FPGA**
  - Low latency and power consumption
  - Low throughput due to limited memory
- **GPU**
  - High throughput
  - High power consumption
  - Poor performance for single and small-scale matrices

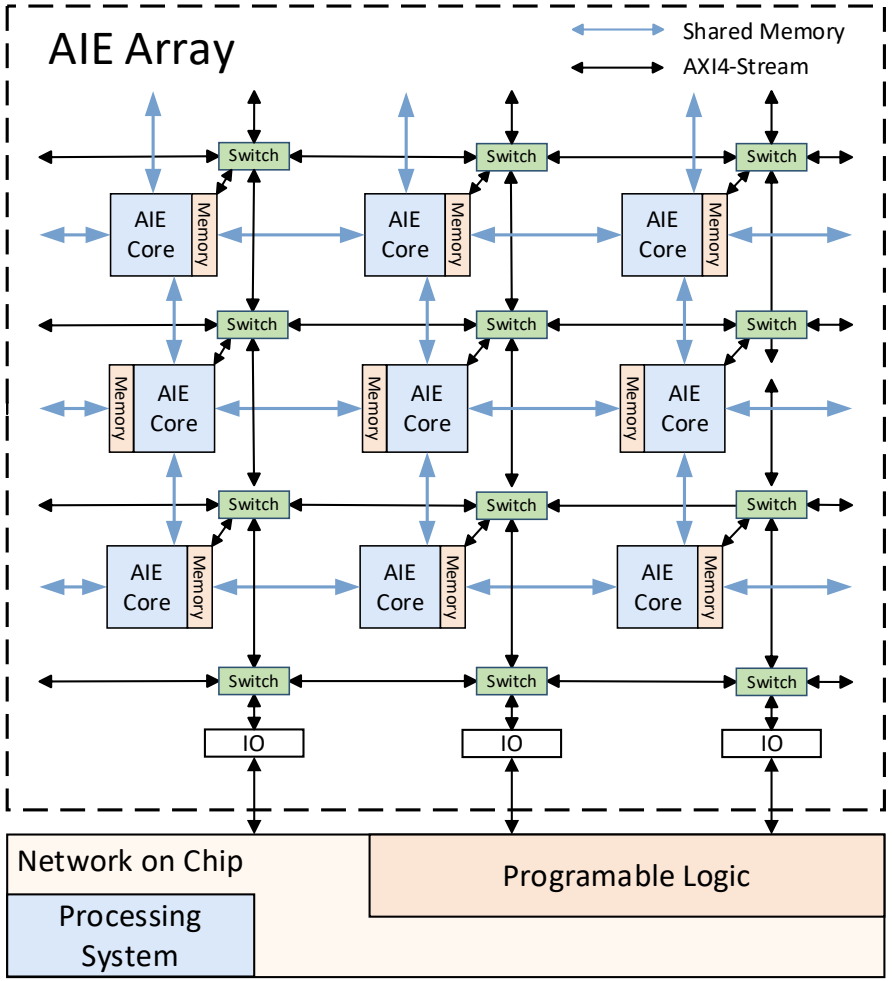
Versal ACAP

- AIE (AI Engines):
  - VLIW & SIMD processors
  - Various communication mechanism
- PL (Programmable Logic)
  - Larger on-chip memory
  - Multiple IOs between AIEs and PL
- PS (Processing System)
  - Arm processor



(a) one-to-one communication

(b) one-to-many communication



Challenges
SVD in Versal ACAP

- **Workload Assignment**
  - Map SVD into PL and AIE: orthogonalization, normalization
- **AIE/PL Interconnection**
  - DMA incurs double memory use and extra latency
  - Limited PLIO
- **Vast Design Space**
  - Workload Assignment
  - Rapid Performance Estimation

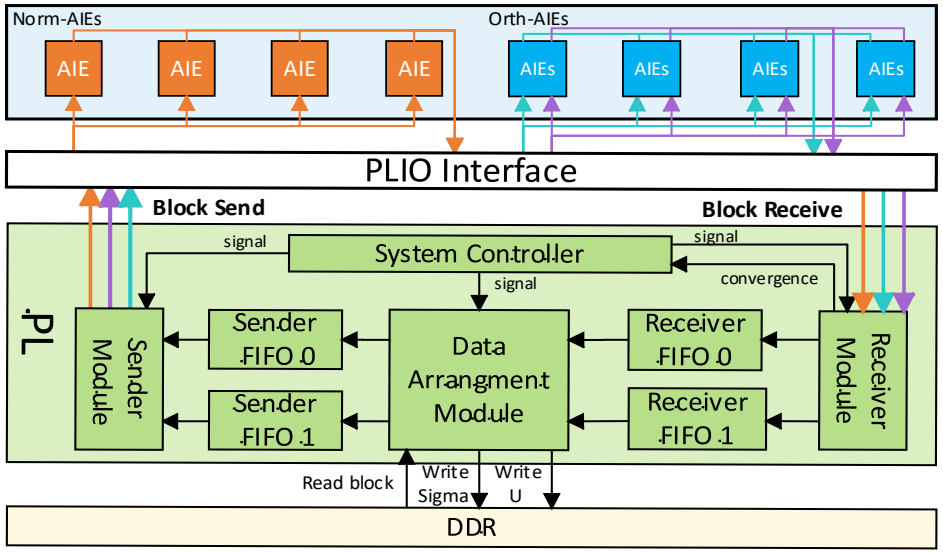
Methodology

SPONSORED BY CEDA SIGDA

Accelerator Framework

Map SVD to Versal ACAP

- HeteroSVD consists of several components:
  - AIEs for orthogonalization
  - AIEs for normalization
  - PL for SVD ordering and controller



Architecture of HeteroSVD

Algorithm 1: SVD Algorithm in HeteroSVD.

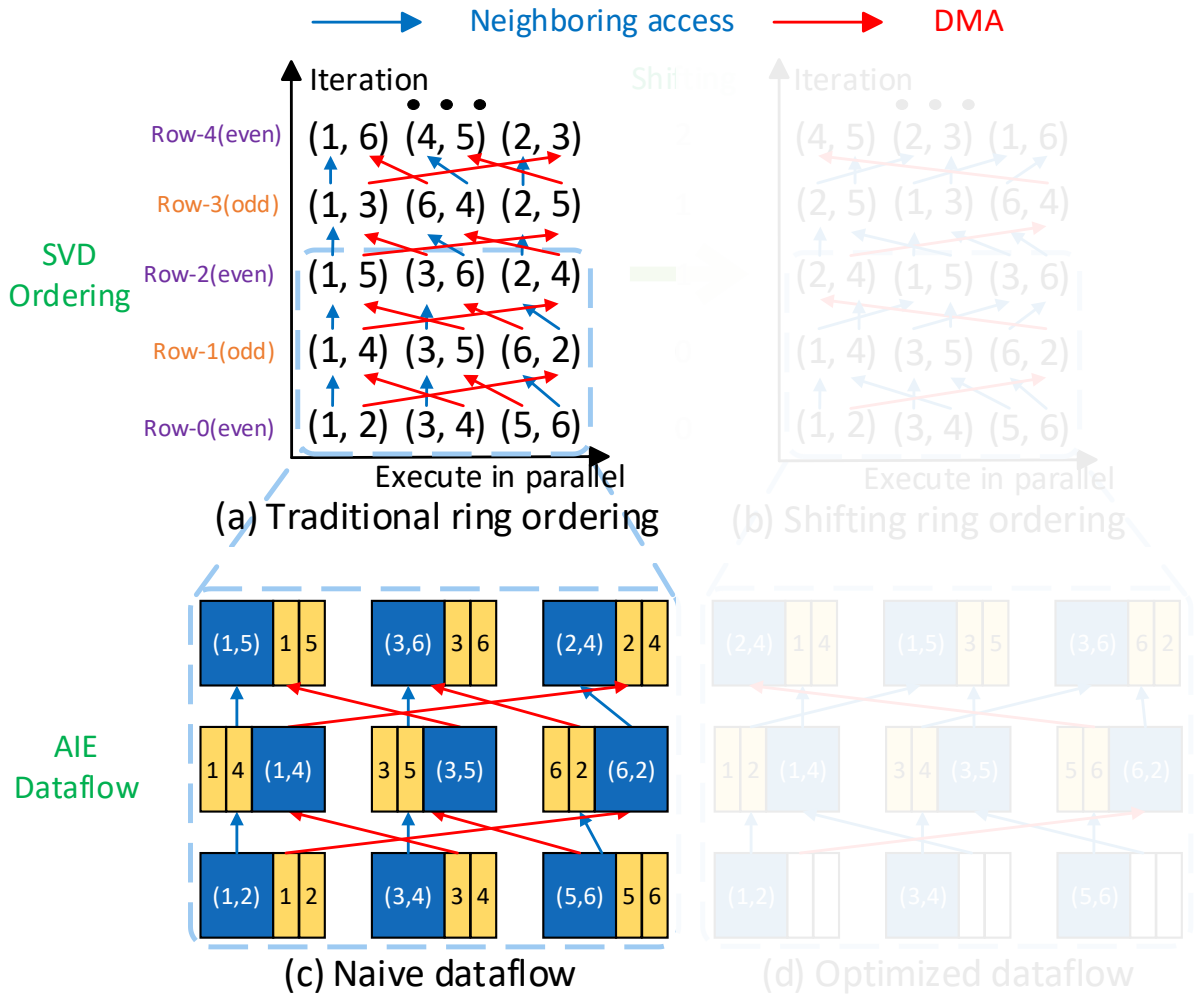
Input: $A_{m \times n}$
Output: U,Σ

1 $convergence\_rate = 0$;
2 **while** $convergence\_rate > precision$ **do**
3 // AIE-PL communication
4 **for** *each block pair* (A_u, A_v) *in* $[A_1, A_2, \cdots, A_p]$ **do**
5 send A_u, A_v to orth-AIEs;
6 // AIE interconnection
7 **for** *each column pair* (A_i, A_j) *in block pair* (A_u, A_v) **do**
8 // AIE computation of orthogonalization (orth-AIE)
9 calculate $conv(A_i, A_j)$;
10 update $conv(A_u, A_v)$;
11 calculate t, τ, c, s;
12 update(A_i, A_j);
13 **end**
14 receive A_u, A_v, $conv_(A_u, A_v)$ from orth-AIEs;
15 update convergence_rate;
16 **end**
17 **end**
18 // AIE-PL communication
19 **for** *each block*(A_i) *in* $[A_1, A_2, \cdots, A_p]$ **do**
20 send A_i to norm-AIEs;
21 **for** *each column*(A_j) *in block*(A_i) **do**
22 // AIE computation of normalization (norm-AIE)
23 calculate Σ_j, U_j;
24 **end**
25 receive Σ_i, U_i from norm-AIEs;
26 **end**

AIE Dataflow Design

Algorithm-Hardware Co-design

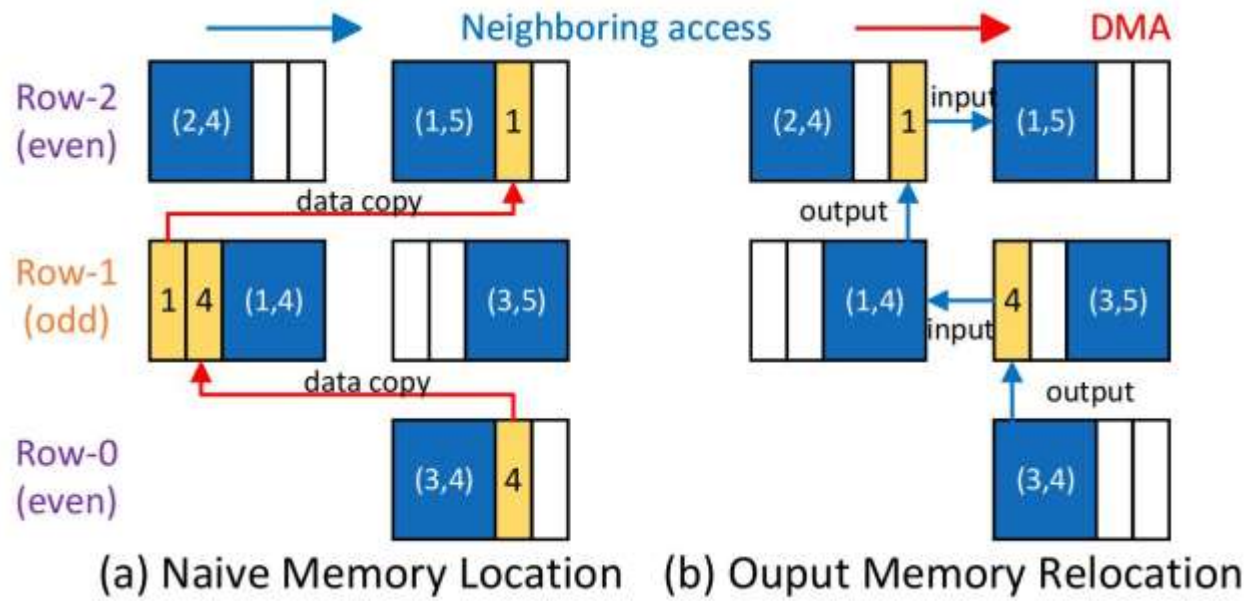
- **Traditional Ring Ordering**
  - Diagonal communication
  - Large DMA usage
  - Double memory consumption



AIE Dataflow Design

Algorithm-Hardware Co-design

- **Output Memory Relocation**
  - Eliminate DMA in diagonal communication

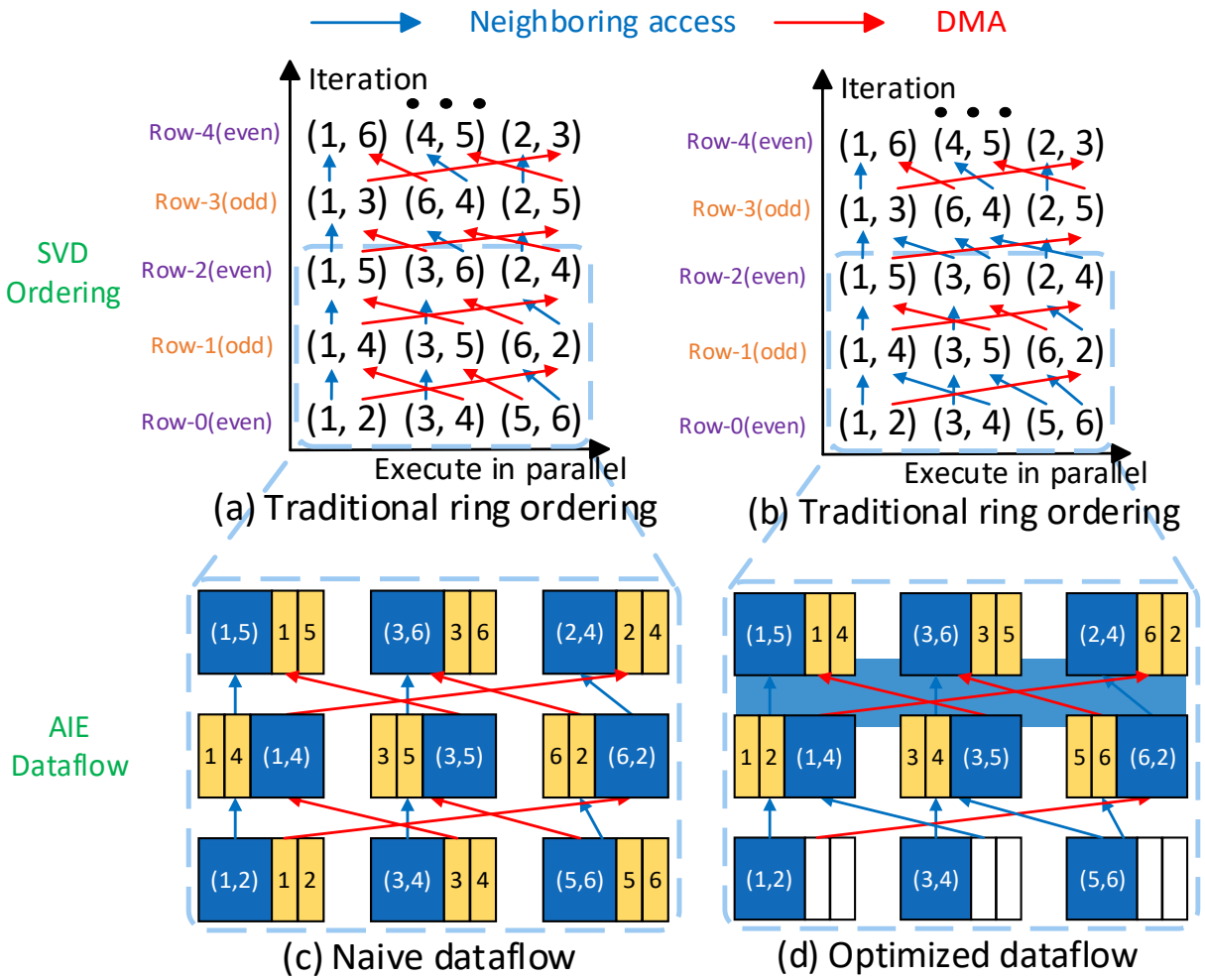


AIE Dataflow Design

Algorithm-Hardware Co-design

- **Traditional Ring Ordering**
  - Same computation pattern
  - N-1 Left, 1 right and 1 straight

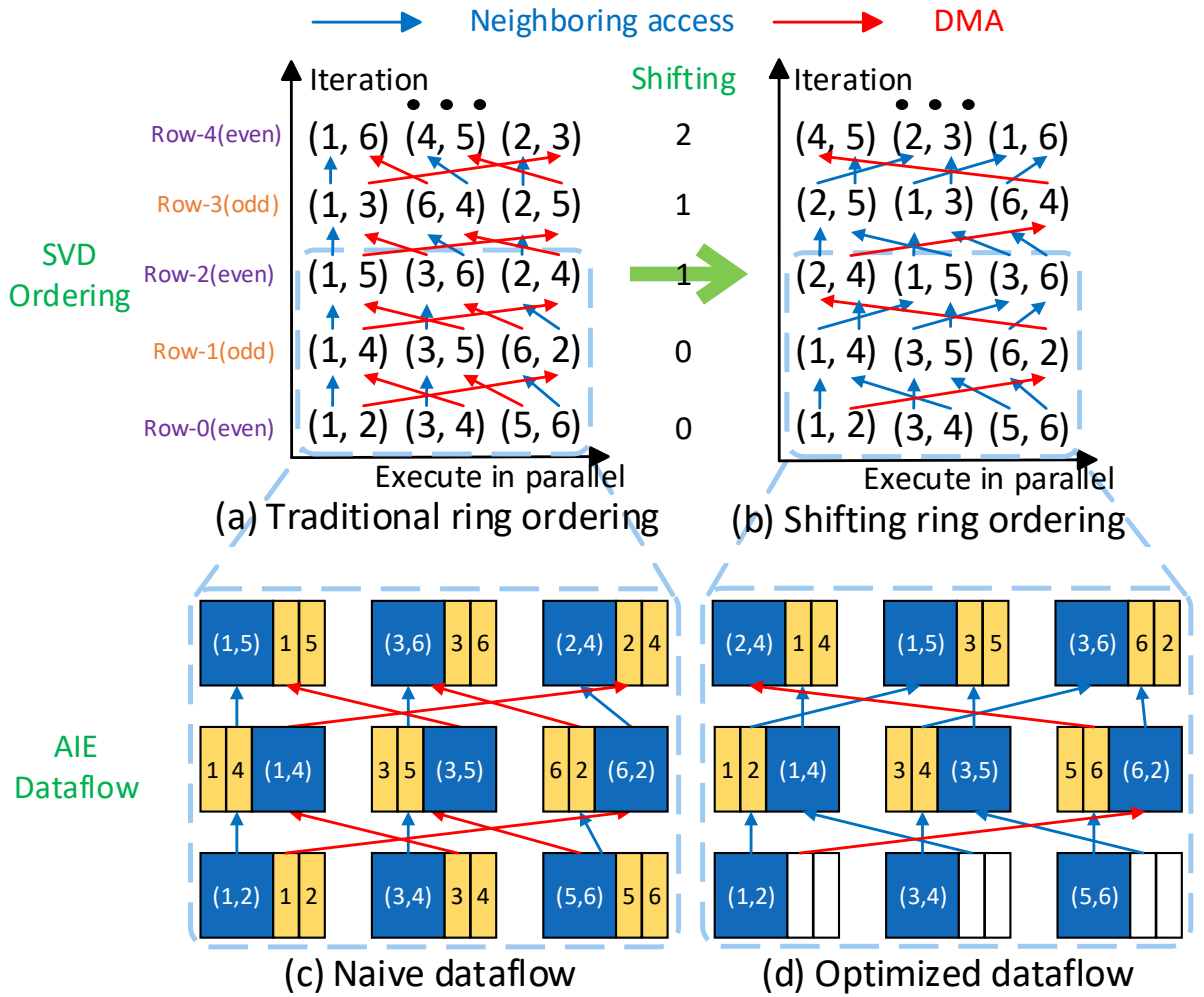
- **AIE Structure**
  - Changing structure based on the row position



AIE Dataflow Design

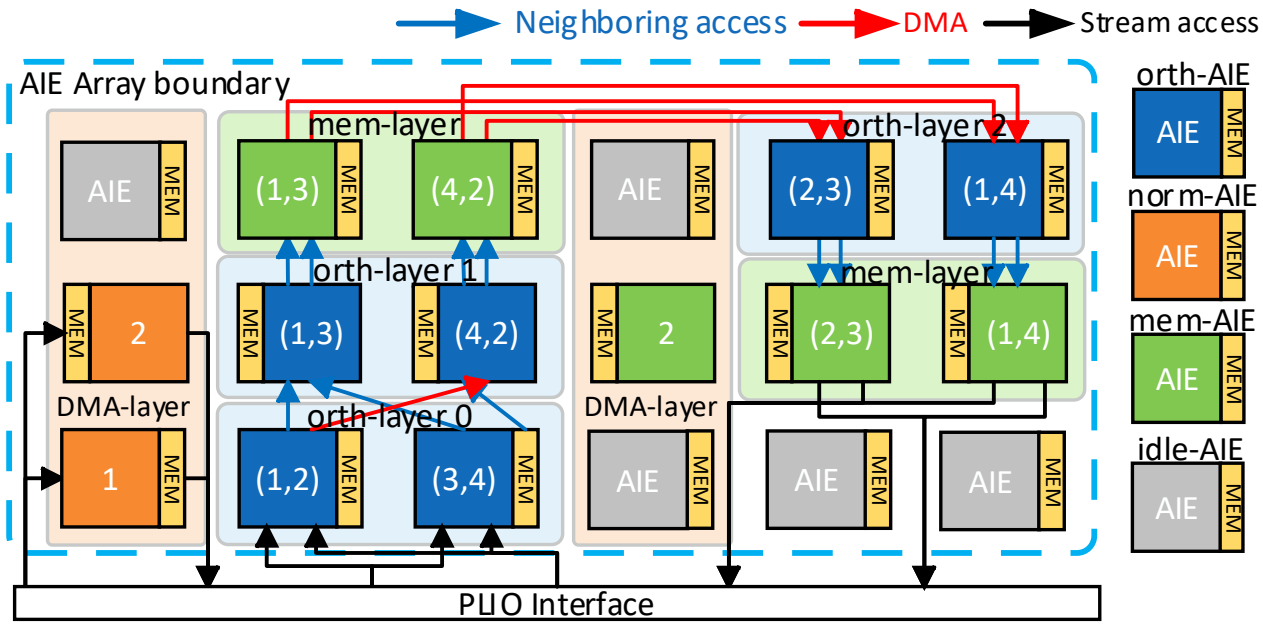
Algorithm-Hardware Co-design

- **Shifting Ring Ordering**
  - Shift ordering based on the row position
  - Lowest DMA usage



AIE Placement Design

- Location and Connection in AIE
  - Memory reservation for output memory relocation
  - Memory reservation for DMA
  - Forwarding for PLIO reuse

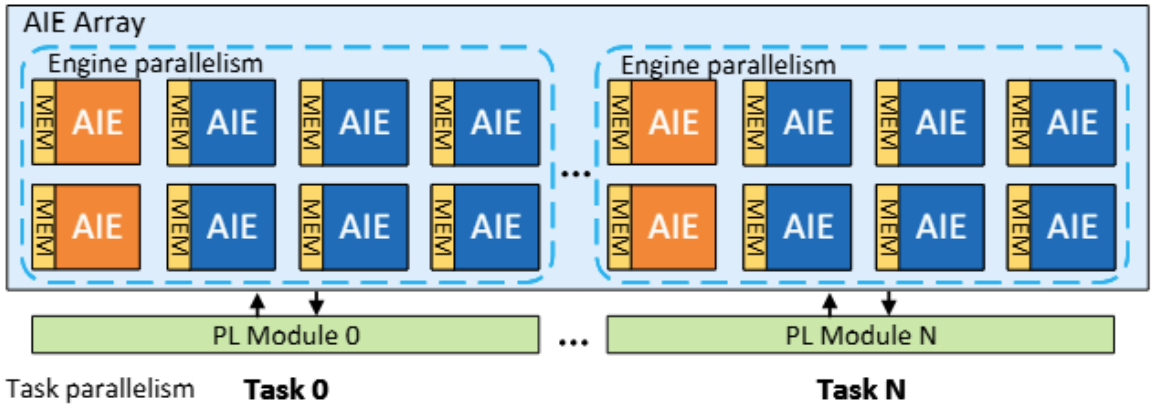


Automation Design Framework

Micro-Architecture Parameters

- AIE-level Parallelism
  - AIEs for single task
- Task-level Parallelism
  - Tasks processed simultaneously
- PL Frequency

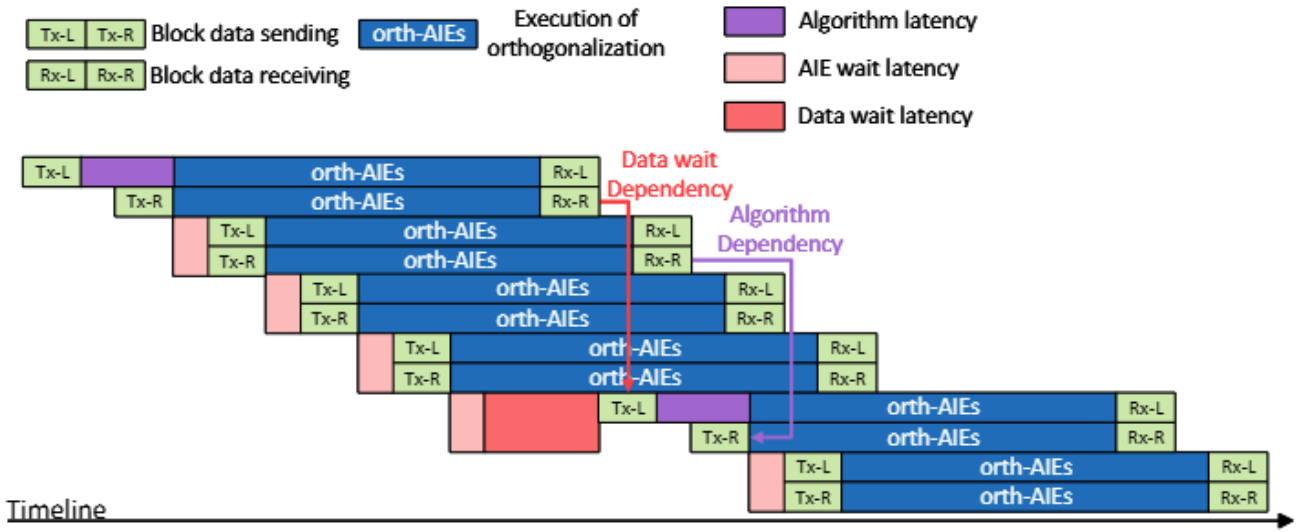
Hierarchy	Parameter	Range or Value
First order	P_{eng}	$n \in [1, 11]$
	P_{task}	$k \in [1, 26]$
	PL frequency (MHz)	-
Second order	the number of orth-AIE	$n(2n-1)k$
	the number of norm-AIE	nk
	the number of mem-AIE	-
	the number of PLIO	$6k$



Automation Design Framework

Performance Model

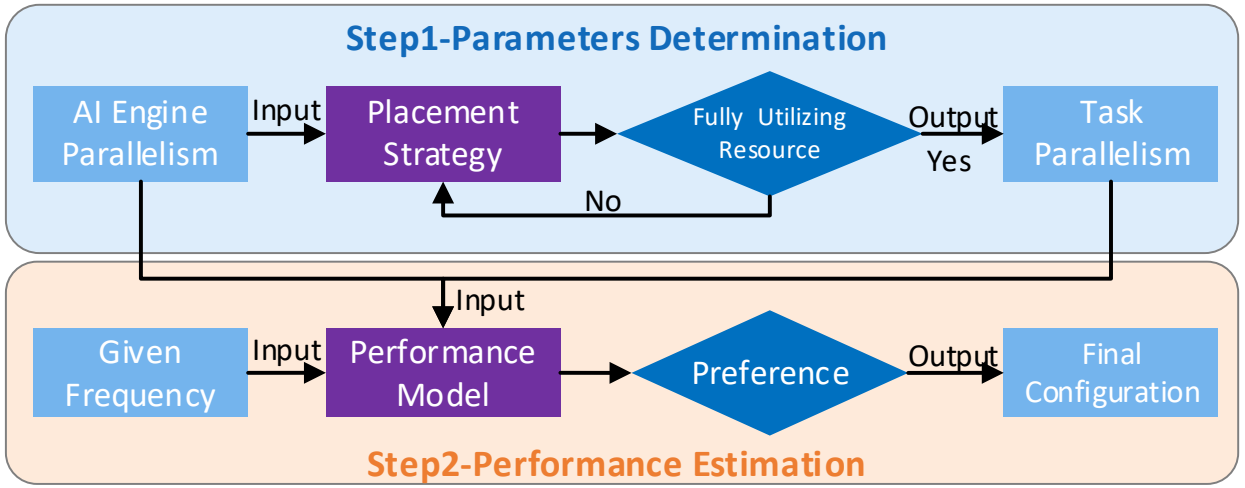
- Dataflow
  - PL data send/receive
  - AIE computation
- Extra Latency
  - Shifting ring ordering
  - Write-after-read
  - HLS loop switch



Automation Design Framework

Design Search Exploration

- DSE Flow
  - AI Engine Parallelism Enumeration
  - Maximize Task Parallelism



Evaluation

SPONSORED BY CEDA SIGDA

Evaluation

Experimental Settings

- Device
  - Versal ACAP: VCK190 Evaluation Kit
  - GPU: NVIDIA RTX 3090
- Baseline
  - FPGA solution (From TCAS-II 2022)
  - GPU solution (From SC 2022)

Evaluation

Performance results of HeteroSVD

TABLE II Latency comparison and resource utilization between HeteroSVD and FPGA.

Matrix Size	FPGA [6]				HeteroSVD				HeteroSVD
	Latency(s)	LUT	BRAM	DSP	Latency(s)	LUT	URAM	AIE	Speedup
128×128	0.0014	212K (30.6%)	519.5 (31.4%)	1602(44.5%)	0.0011	15.1K(1.68%)	4(0.86%)	128(32%)	1.27×
256×256	0.0113				0.0057	15.2K (1.69%)	20 (4.32%)		1.98×
512×512	0.0829				0.0435	15.5K (1.72%)	64 (13.82%)		1.90×
1024×1024	0.6119				0.3415	15.7K (1.75%)	244 (52.70%)		1.79×

TABLE III Latency, throughput and energy efficiency comparison between HeteroSVD and GPU.

Matrix Size	GPU [11] (270W)			HeteroSVD (<39W)			HeteroSVD vs GPU		
	Latency (s)	Throughput (Tasks/s)	Energy Efficiency (Tasks/s/Watt)	Latency (s)	Throughput (Tasks/s)	Energy Efficiency (Tasks/s/Watt)	Latency Speedup	Throughput Speedup	EE Gain
128×128	0.0166	1351.35	5.005	0.0023	2389.69	65.940	7.22×	1.77×	13.18×
256×256	0.0429	217.39	0.805	0.0130	239.48	6.251	3.30×	1.10×	7.76×
512×512	0.1237	27.55	0.102	0.1076	24.42	0.663	1.15×	0.89×	6.50×
1024×1024	0.6857	3.52	0.013	0.7937	1.27	0.057	0.86×	0.36×	4.36×

Evaluation

Model Evaluation

- Model Evaluation in single iteration
  - Maximum error: 3.03%
  - Average error: 1.78%
- Model Evaluation in DSE
  - Maximum error: 7.52%
  - Average error: 4.33%

TABLE IV The processing time (ms) of SVD single iteration from the performance model and on-board measurement under a frequency of 208.3 MHz.

Matrix Size	P_{eng}	On-board Meas.	Perf. Model	Error
128×128	2	0.993	1.022	2.92%
256×256		6.151	6.338	3.03%
512×512		43.229	42.020	2.80%
128×128	4	0.395	0.391	1.03%
256×256		2.853	2.806	1.66%
512×512		21.584	21.265	1.48%
128×128	8	0.214	0.219	2.57%
256×256		1.475	1.476	0.05%
512×512		10.965	10.903	0.56%

TABLE V The processing time (ms) of SVD with one iteration from the performance model and on-board measurement under various application scenarios.

Matrix Size	Batch	Freq.(MHz)	P_{eng}	P_{task}	On-board Meas.	Perf. Model	Error
128×128	1	450	8	1	0.357	0.384	7.52%
256×256	1	420	8	1	1.202	1.120	6.82%
512×512	1	350	8	1	7.815	7.510	3.90%
1024×1024	1	310	8	1	58.885	58.255	1.02%
128×128	100	330	4	9	6.099	6.412	5.12%
256×256	100	310	4	9	27.836	26.623	4.36%
512×512	100	310	4	7	238.002	224.301	5.76%
1024×1024	100	310	8	1	5872.181	5878.970	0.12%

Evaluation

Discussion on DSE Results

- High P_{eng} for low Latency
- High P_{task} for high Throughput

TABLE VI Latency(ms), throughput(task/s) and power(W) comparison between different design points. The matrix size is 256×256 and the PL frequency is reported in 208.3MHz

P_{eng}	P_{task}	AIE	URAM	Latency	Throughput	Power
2	26	293(73.25%)	416(89.85%)	35.689	707.501	44.16
4	9	357(89.25%)	144(31.10%)	19.303	508.436	34.63
6	4	366(91.50%)	120(25.92%)	13.117	306.876	30.79
8	2	322(80.50%)	32(6.91%)	9.247	219.257	26.06

Conclusion

SPONSORED BY CEDA® sigda

Conclusion

- We map SVD into Versal ACAP and design optimized AIE dataflow and placement
- We propose a performance model and design search exploration flow to find best architecture.
- Our work is open-source at https://github.com/zhaokang-lab/HeteroSVD

Thank you

Xinya Luan

Beijing University of Posts and Telecommunications