

Towards Efficient Matrix Operations on CGRAs with Linear Computation Coding Dataflows

FLORIAN ROKOHL, Alex Lehnert, Marc Reichenbach

Chair of Integrated Systems

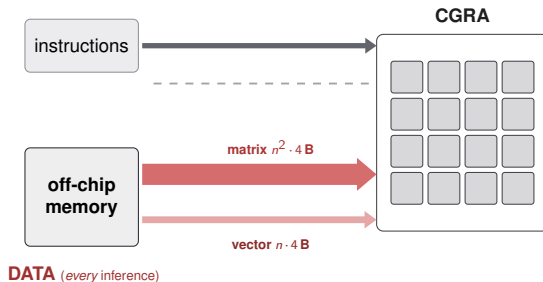
Institute of Applied Microelectronics and Computer Engineering

Where the matrix sits today

For DNN inference it ends up on the wrong side of the configuration boundary

- Mappers start from **control flow**
- So the matrix stays a **runtime input**
- Its structure goes **unused**

CONFIGURATION (loaded once)



Where the matrix sits today

For DNN inference it ends up on the wrong side of the configuration boundary

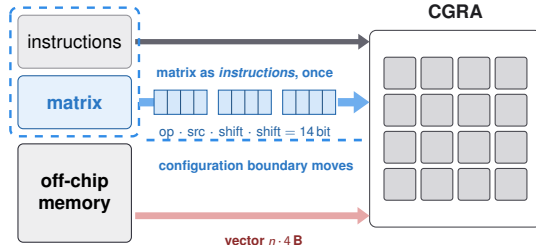
- Mappers start from **control flow**
- So the matrix stays a **runtime input**
- Its structure goes **unused**

> **90 %**

of *Mul+Add* baseline time is DMA, not compute

at $n=32$, grid 16×16 and larger

CONFIGURATION (loaded once)



Move the matrix across the boundary

LCC rewrites a constant matrix as shifts and adds

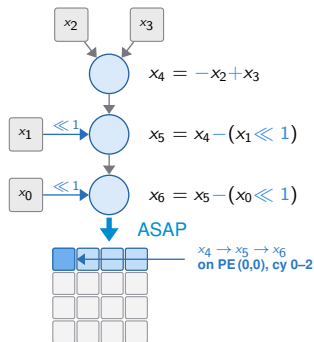
$$x_k = (x_p \ll s_0) \pm (x_q \ll s_1)$$

- Shift widths and signs are constants, so they are **configuration**
- Approximate: **48 dB** SQNR $\approx$ 8-bit quantization [Wei et al., Appl. Sci. 2024]

LZD shares subexpressions across rows [Müller, ITW'22]

real model, 4×4 matrix:

29 SAcc ops • 10 cycles • 4 of 16 PEs used



What the move costs, and what it buys

The matrix leaves the data path and enters the configuration

Runtime traffic

$$\mathcal{O}(n^2) \rightarrow \mathcal{O}(n)$$

Their coefficients are **data**.

Our shifts are **configuration**.

The baselines cannot simply copy this.

The cost: the bitstream is **1.0–1.4**× the raw matrix, paid *once per matrix*, not per inference.

Mul+Add and ML-CGRA [Luo et al., DAC'23]

matrix + vector streamed

4224 B every inference ($n=32$)

What the move costs, and what it buys

The matrix leaves the data path and enters the configuration

Runtime traffic

$$\mathcal{O}(n^2) \rightarrow \mathcal{O}(n)$$

Their coefficients are **data**.

Our shifts are **configuration**.

The baselines cannot simply copy this.

Mul+Add and ML-CGRA [Luo et al., DAC'23]

matrix + vector streamed

4224 B every inference ($n=32$)



SAcc (this work)

vector streamed, matrix configured

128 B every inference ($n=32$)

The cost: the bitstream is **1.0–1.4**× the raw matrix, paid *once per matrix*, not per inference.

Results: $4.4\times$ faster, dominated by DMA

solid = compute, hatched = DMA

at $n=32$ on the 4×4 CGRA

$4.4\times$

over Mul+Add

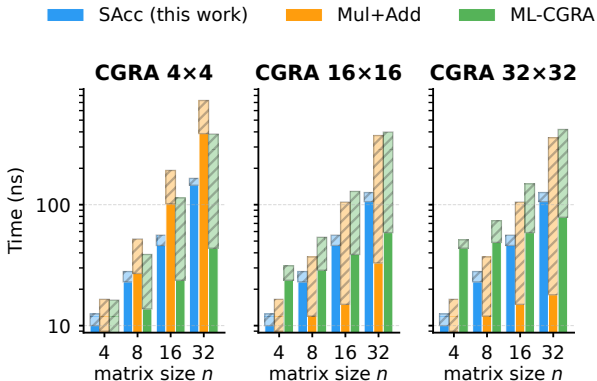
$2.3\times$

over ML-CGRA (a lower bound)

$3.3\times$ on larger grids

$2.8\times$

from DMA elimination alone



Model: SAcc/Add 1 ns, Mul 3 ns: illustrative, not silicon (SKY130: parity to $\sim 1.5\times$) • DMA 400 MHz, 32 B/cyc

Take-away

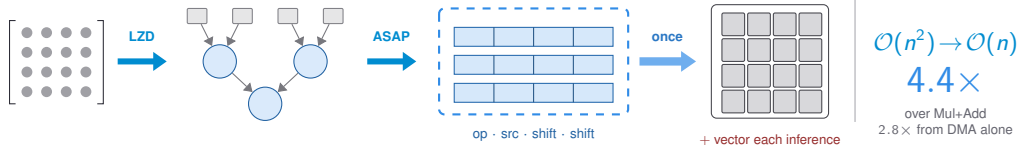
Algorithm vs. data is a design decision, not a given

constant matrix

shift-add graph

configuration

CGRA



Thank you

florian.rokohl@uni-rostock.de

Supported in part by the German Research Foundation (DFG) through the project *Berechnungskodierung*, grant RE 4182/4-1.



Institute



LinkedIn
Florian Rokohl



LinkedIn
Alex Lehnert