2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits) | 979-8-3503-6146-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/VLSITECHNOLOGYANDCIR46783.2024.10631383

# Onyx: A 12nm 756 GOPS/W Coarse-Grained Reconfigurable Array for Accelerating Dense and Sparse Applications

Kalhan Koul[1†], Maxwell Strange[1], Jackson Melchert[1], Alex Carsello[1], Yuchen Mei[1], Olivia Hsu[1], Taeyoung Kong[1], Po-Han Chen[1], Huifeng Ke[1], Keyi Zhang[1], Qiaoyi Liu[1], Gedeon Nyengele[1], Akhilesh Balasingam[1], Jayashree Adivarahan[1], Ritvik Sharma[1], Zhouhua Xie[1], Christopher Torng[2], Joel Emer[3], Fredrik Kjolstad[1], Mark Horowitz[1], Priyanka Raina[1]

[1]Stanford University, CA, USA; [2]University of Southern California, CA, USA; [3]MIT, MA, USA; [†]Email: kkoul@stanford.edu

## Abstract

Onyx is the first fully programmable accelerator for arbitrary sparse tensor algebra kernels. Unlike prior work, it supports higher-order tensors, multiple inputs, and fusion. It achieves this with a coarse-grained reconfigurable array (CGRA) that has composable memory primitives for storing compressed any-order tensors and compute primitives that eliminate ineffectual computations in sparse expressions. Further, Onyx improves dense image processing and machine learning (ML) with application-specialized compute tiles, memory tiles optimized for affine access patterns, and hybrid clock gating in the global buffer. We achieve up to 565x better energy-delay product (EDP) for sparse kernels vs. CPUs with sparse libraries, and up to 76% and 85% lower EDP for image processing and ML, respectively, vs. Amber [1].

## Introduction

CGRAs are widely-used programmable fabrics for building accelerators, but prior work has focused on dense applications [1, 2]. Given that important applications ranging from machine learning to scientific computing leverage sparsity for efficiency, we present Onyx, the first fabricated CGRA accelerating both dense and sparse applications (Fig. 1). The Onyx SoC (Fig. 2) has a CGRA with 384 processing element tiles (PE) and 128 memory tiles (MEM) on an interconnect that supports static (dense) and dynamic (sparse) data movement. The CGRA connects to a 4 MB global buffer (GLB) with 16 tiles, each with load and store engines. An Arm M3 processor orchestrates kernels on the accelerator.

## Sparse Acceleration Hardware

Prior sparse accelerators have focused on sparsity in specific kernels [3, 4], mostly matrix multiplication, leaving out support for other kernels, higher-order tensors, multiple inputs, and fusion. Onyx addresses this by building on a streaming representation of fibertrees [5] (Fig. 3), enabling us to store and process any-order tensors and eliminate unnecessary memory accesses. We convert stored tensors into coordinate, reference (pointer to fiber), and value streams that move between CGRA tiles using level writer, level buffer, and level scanner primitives, which we implement in the memory tile (Fig. 3). We partition a single-port 512x64b SRAM in the level buffer into segment and coordinate arrays. The level writer consumes a coordinate stream and generates segment and coordinate addresses to write to the partitioned memory. The level scanner takes a reference pointing to a segment pair, requests that pair from the level buffer to determine the coordinate array length, and then generates addresses to read the coordinate stream from the level buffer. It also produces the corresponding reference stream for downstream tiles. We add cache lines in the level buffer to avoid redundant reads of the same 4-element word in consecutive accesses.

We eliminate all ineffectual computation by adding intersecter/unioner and coordinate dropper primitives to the PEs (Fig. 4). The intersecter/unioner computes on the input coordinate streams to determine the output coordinates that have non-zero values. Since this can produce empty fibers, the coordinate dropper removes these fibers by dropping adjacent stop tokens in the inner-coordinate stream and the matching outer coordinate. To support tensor algebra, we also add a reducer, which sums a stream, and a repeater which duplicates streams for tensor broadcasting. Fig. 5 shows a sparse matrix multiplication mapped on Onyx to demonstrate the composability of our primitives. Our generality supports higher-order tensors, multiple inputs, and, importantly, fusion for expressions such as MTTKRP ($X_{ij}=\sum_{kl}B_{ikl}C_{jk}D_{jl}$), whose fused schedule eliminates ineffectual compute across all inputs and is up to 6.6x faster than two unfused kernels.

To support non-deterministic latency, we add ready-valid capability to the interconnect in [1] by replacing pipeline registers with FIFOs. However, we avoid doubling the number of registers by designing a "SplitFIFO" (Fig. 5) composed of pipeline registers in adjacent tiles sharing control signals, saving us 13% in interconnect area.

## Dense Acceleration Hardware Optimizations

In contrast to Amber [1], which performs a single Int16 or BFloat16 operation per PE, Onyx improves compute density by performing single or multiple (multiply-add, add-add, multiply-shift, min-max) Int16 operations per PE. This allows higher parallelization, decreasing execution cycles by up to 50%. Compared to the memory controllers in [1], we eliminate a write address generator by reusing the addresses generated by the read address generator and adding a delay block for read-modify-write operations (Fig. 6). We also reduce the depth of the serial-in, parallel-out buffer and bit widths of counters and configuration registers. These optimizations reduce the area of the memory tile by 24%. Finally, since the number of active GLB tiles varies between kernels, we only clock the active memories to save power. Clock gating the GLB poses a challenge since although CGRA data transfers are static, off-chip data transfers are only known at runtime. We implement hybrid static/dynamic gating for each GLB tile, that for typical double-buffering workloads, reduces GLB power by 24%.

## Results

We use the Halide [1] and Custard [5] compilers to map dense and sparse expressions, respectively. For image processing, we achieve up to a 76% EDP reduction vs. Amber [1] (Fig. 7). ResNet layers achieve a 78–85% lower EDP vs. Amber. On sparse kernels, we achieve a 4.4x–565x geomean improvement in EDP versus a CPU with sparse libraries (MKL and TACO). Fig. 8 shows comparisons against other fabricated chips. Onyx is fabricated in GF 12 nm and achieves a 756 INT16 GOPS/W peak energy efficiency, 41% better than Amber. Unlike [3, 4], which put together specialized accelerators for different applications on the same chip, Onyx is the first programmable accelerator for arbitrary dense and sparse tensor algebra, demonstrating a promising approach for accelerating fast evolving application domains.

**Acknowledgements:** DARPA DSSoC, AHA, SystemX, SRC PRISM Center, NSF Award 2238006, Intel HIP and Samsung.

**References:** [1] K. Feng et al., JSSC'2023. [2] Q. Zhang et al., VLSI'2022. [3] S. Song, et al., VLSI'2023. [4] W. -C. Huang et al., VLSI'2022. [5] O. Hsu et al., ASPLOS'2023.

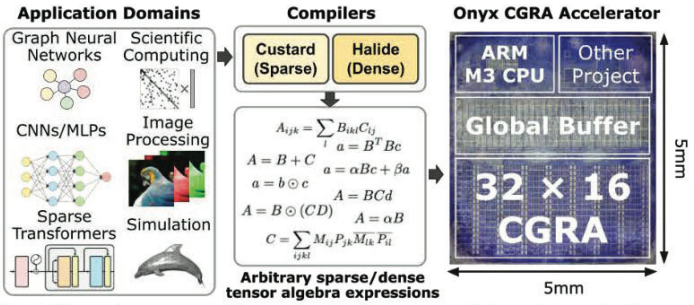


Fig. 1 Onyx: A programmable accelerator for sparse and dense tensor algebra.

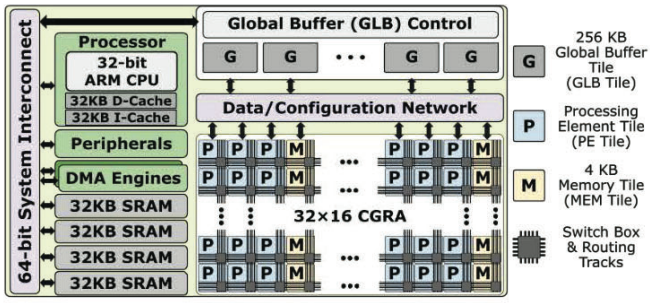


Fig. 2 Onyx system-on-chip architecture with 384 PE and 128 MEM tiles.

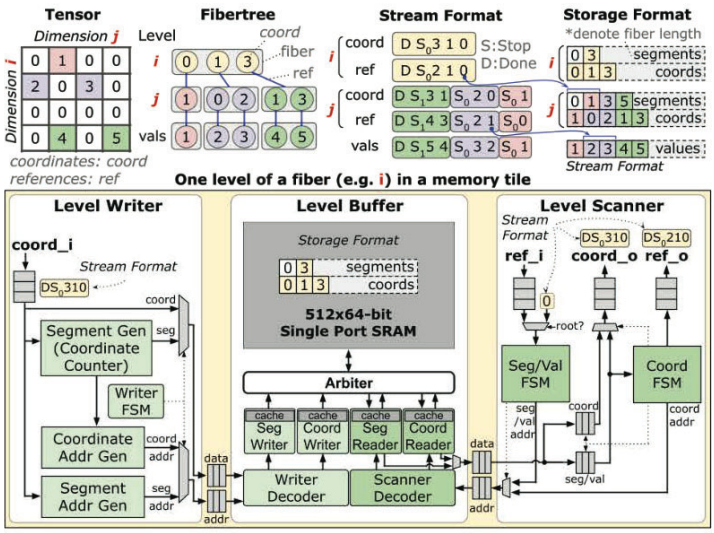


Fig. 3 MEM tile primitives that write, store, and read fibertrees.

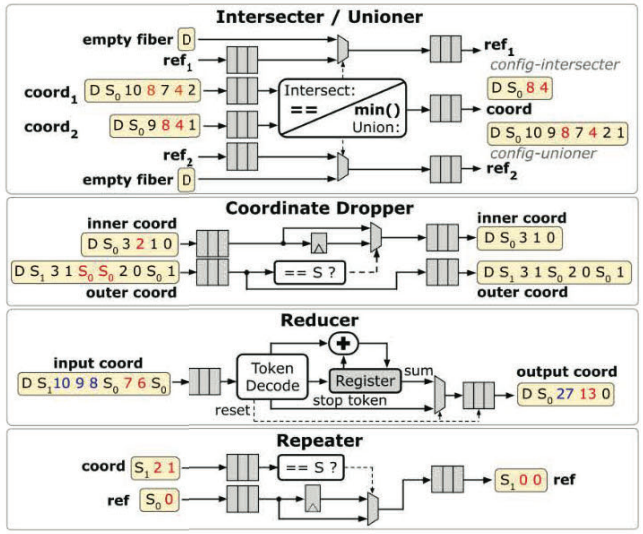


Fig. 4 PE tile primitives that compute on tensor indices and addresses.

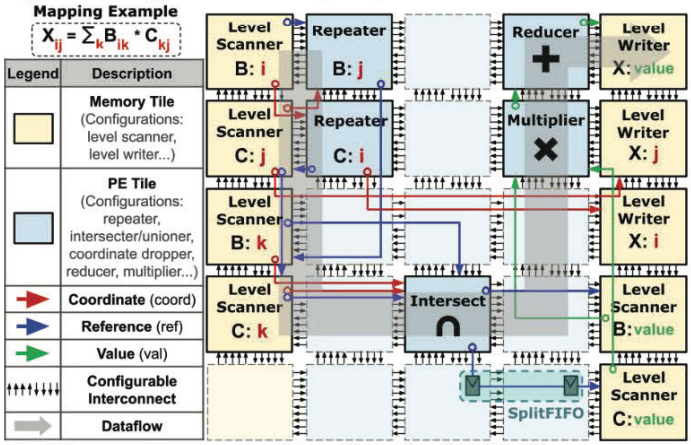


Fig. 5 Sparse matrix - sparse matrix multiplication (SpM*SpM) mapped onto the Onyx CGRA demonstrating composability for our hardware primitives.

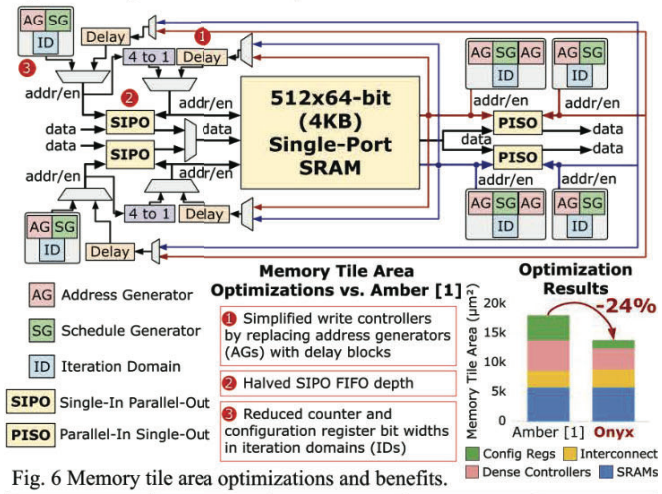


Fig. 6 Memory tile area optimizations and benefits.

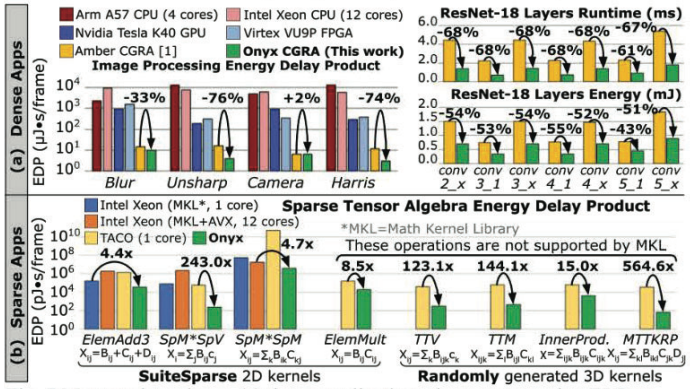


Fig. 7 Measured results on (a) dense applications: image processing EDP vs. other platforms, and machine learning (ResNet-18) runtime and energy vs. Amber (b) sparse applications: EDP vs. CPU with MKL/TACO sparse library.

| | This Work | Amber VLSI 22 | Zhang VLSI 22 | Huang VLSI 22 |
|---|---|---|---|---|
| Architecture | SoC w/CGRA | SoC w/CGRA | CNN/ Image Processing PE Array | Sparsity-Aware CNN-GCN |
| Programmability | ✔ | ✔ | ✔ | ✘ |
| Sparse/Dense | Sparse/Dense | Dense Only | Dense Only | Sparse/Dense |
| Technology/Area | 12 nm/23 mm² | 16 nm/20.1 mm² | 22 nm/8.8 mm² | 28 nm/8.3 mm² |
| Formats Supported | BFloat16, INT16 | BFloat16, INT16 | INT16 | INT8 |
| # of Cores | 384 PEs, 1 M3 | 384 PEs, 1 M3 | 576 PEs, RISC Core, Custom M33 | 1024 8-bit MACs |
| Total SRAM | 4.5 MB | 4.5 MB | 1428 KB, 2MB MRAM | 292 KB |
| Voltage & Frequency | 0.78 V @970 MHz | 0.84 V @580 MHz 1.29 V @955 MHz | 0.5 - 1.0 V 56 KHz - 190 MHz | 10 - 200 MHz |
| Peak Performance (Normalized to MAC=2OPs for all) | 571 INT16 GOPS @(0.78 V, 850 MHz) | 367 INT16 GOPS @(1.29 V, 955 MHz) | 414 INT16 GOPS @(1.0 V, 180 MHz) | Dense: 410 GOPS Sparse: 3.3 TOPS |
| Peak Energy Efficiency (Normalized to MAC=2OPs for all) | 756 INT16 GOPS/W @(0.66 V, 500 MHz) | 538 INT16 GOPS/W @(0.84 V, 580 MHz) | 7.0 INT16 TOPS/W @(0.5 V, 16 MHz) | Dense: 3.1 TOPS/W Sparse: 25.1 TOPS/W |

Fig. 8 Comparison with state-of-the art SoCs.