Ultra-Elastic CGRAs for Irregular Loop Specialization

Christopher Torngr, Peitian Pan, Yanghui Ou, Cheng Tan, and Christopher Batten

Stanford University, Cornell University
HPCA-27 - March 2, 2021
Coarse-grain reconfigurable arrays (CGRAs)

- CGRAs remain **flexible after fabrication**, allowing algorithms to continue evolving
- More efficient than FPGAs due to coarse-grain PEs at the word level
- More flexible than dedicated ASICs
Recent CGRAs are increasingly adopting **elasticity**

- Traditional CGRAs **statically schedule** dataflows onto the compute fabric.
- An elastic CGRA includes latency insensitive handshakes.
- Elasticity allows a CGRA to **tolerate irregularity** to a limited degree:
  - Irregular memory accesses
  - Irregular control flow
  - Inter-iteration loop dependencies
Inter-iteration loop dependencies are important

(a) Code with Dep

out[0] = input

for i in 1 to N:
    out[i] = func(out[i-1])

e.g., pointer chasing

(b) Dataflow Graph

out[i-1] → A → B → C → D

func

out[i]

(c) Mapped CGRA

input bank → A → B → output bank

output bank ← D ← C

CGRA tile executing 'C'

(d) Pipeline Diagram

<table>
<thead>
<tr>
<th>Cycle</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter 0</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iter 1</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bottlenecks the throughput of the entire kernel

Inter-iteration dependency
How to architect a CGRA to mitigate this bottleneck?

Techniques which do not help

- Parallelism
- Control speculation
- Software pipelining

Key observation

- Even if cycle count is fixed
- We can change cycle time by adjusting voltage and frequency
Ultra-Elastic CGRAs for Irregular Loop Specialization

How to select voltage and frequency for PEs?

An architecture easily targeted by the compiler

Implications of ultra elasticity on fabric design?

Extremely fine-grain DVFS at the PE level

Our quantized approach to fine-grain DVFS, now fully verifiable with commercial STA tools
Intuition for the ultra-elastic computational model

At nominal voltage and frequency, throughput is limited to one iteration every three cycles.
Intuition for the ultra-elastic computational model

Resting nodes save energy, and throughput can remain at one iteration every three cycles.
Intuition for the ultra-elastic computational model

Both resting and sprinting can be combined, throughput is now **one iteration every two cycles**
An analytical performance model and energy model for arbitrary dataflow graphs

- **Discrete event simulator** models elastic nodes firing
  - Includes wire delays
  - Includes clock frequency

- **Detailed energy model** is validated against gate-level energy simulations

*More details in the paper and online*

We augmented a CGRA compiler flow

Code: \[ \text{for} \ (i=0; \ i<N; \ ++i) \{ \ b[i] = a[i] \times x \} \]

Inspired by Karunaratne et. al. DAC 2017
"HyCUBE: A CGRA with Reconfigurable Single-cycle Multi-hop Interconnect"
A heuristic three-phase power-mapping pass

Informal Problem Statement - Assign each PE a power mode from the set \{ V(\text{rest}), V(\text{nominal}), V(\text{sprint}) \}

Algorithm starts by \textit{heuristically} initializing all PEs to \textbf{V(sprint)} before:

1. Complexity-reduction phase
2. Energy-delay optimization phase
3. Constraint phase
Informal Problem Statement - Assign each PE a power mode from the set \{ V(\text{rest}), V(\text{nominal}), V(\text{sprint}) \}

Energy-Delay Optimization Phase

- Heuristic search over subset of space
- Step linearly through groups of PEs
  - Try to reduce voltage
  - Better energy-delay product?
  - Keep it and move along!

Variants

- Performance-optimized mapping
- Energy-optimized mapping
Ultra-Elastic CGRAs for Irregular Loop Specialization

A cross-stack approach for compiler-configurable per-PE fine-grain DVFS

**Compiler**

How to select voltage and frequency for PEs?

**Architecture**

An architecture easily targeted by the compiler

Implications of ultra elasticity on fabric design?

**VLSI**

Extremely fine-grain DVFS at the PE level
Overview of system and array architecture

Key characteristics of note

- Host core offloads compute through a tightly coupled accelerator command interface
- DMAs to host memory
- Array of homogeneous PEs
- Some PEs have memory
Overview of system and array architecture

Key characteristics of note for PE

- Elastic queues from each cardinal direction
- Operators include merge and branch
- Limited register file space
- Interconnection boxes
- Interconnect bypass muxes to route through a busy PE
Impact of inter-PE latency on performance

Takeaways

- Adding three extra cycles of latency degrades throughput by 4x or more!
- A performant architecture must reduce inter-PE sync latency as much as possible
- Asynchronous FIFOs reduce throughput by 3-4x
Which voltages and frequencies make sense?

A reasonable range of voltages in TSMC 28 ... \([0.6V - 1.3V]\)

- Higher sprinting frequency linearly improves throughput
- The lowest possible resting mode should be chosen
- Final selection of three modes
  - Resting = 0.6V
  - Nominal = 0.9V
  - Sprinting = 1.3V
Ultra-Elastic CGRAs for Irregular Loop Specialization

A cross-stack approach for compiler-configurable per-PE fine-grain DVFS

Compiler

How to select voltage and frequency for PEs?

Architecture

An architecture easily targeted by the compiler

Implications of ultra elasticity on fabric design?

VLSI

Extremely fine-grain DVFS at the PE level

Our quantized approach to fine-grain DVFS, now **fully verifiable** with commercial STA tools
Ratiochronous interfaces simplify fine-grain DVFS

Clock domains in all PEs are rationally related

- The divide-by-2 and divide-by-3 clocks have a periodic clocking relationship
- Edges can be uniquely labeled
- Some cycles are unsafe to send data, and the unsafe cycles repeat periodically

Tweak our voltages for rational clocks

- (0.60V, 0.9V, 1.30V) -> 1.9 : 3.0 : 8.5 ratio
- (0.61V, 0.9V, 1.23V) -> 2.0 : 3.0 : 9.0 ratio
Highlights in VLSI support for ultra elastic PEs

Augment PE with circuitry for rational crossing
- Detecting safe edges
- Novel elasticity-aware suppressor

Hierarchical clock network gating
- Three global clock networks... energy?
- Can be gated statically at compile time

Other considerations in the paper
- Multi-rail supply voltages -- three rails
- Bisynchronous queues
- Clock switchers and dividers at 50% duty
- Voltage level shifters and power switches
Ultra-Elastic CGRAs for Irregular Loop Specialization

A cross-stack approach for compiler-configurable per-PE fine-grain DVFS

Compiler

How to select voltage and frequency for PEs?

Architecture

An architecture easily targeted by the compiler

Implications of ultra elasticity on fabric design?

VLSI

Extremely fine-grain DVFS at the PE level

Methodology and Evaluation
Evaluation: Methodology

Benchmarks and compiler

- In-House: **llist** - linked-list search, **dither** - Floyd-Steinberg grayscale dithering
- From [21]: **fft** (fft), **susan** (automotive image smoothing), and **bf** (block-cipher)
- Basic compiler flow inspired by DAC 2017 "HyCUBE": LLVM pass (v3.8.0)

RTL and VLSI modeling

- Three 8x8 CGRAs: inelastic (IECGRA), elastic (ECGRA), ultra-elastic (UE-CGRA)
- Modeled in PyMTL, a Python-based HDL
- Commercial ASIC toolflow at 750 MHz using Synopsys and Cadence tools

Energy Modeling

- Detailed RTL/gate-level post-PnR SDF (Synopsys ASIC toolflow, TSMC 28nm 0.9V)
- SPICE modeling for voltage-frequency relationship in TSMC 28nm
CGRA energy profiles for llist and dither

Elastic CGRA

Christopher Torng - HPCA 2021 Session 4B - Ultra-Elastic CGRAs for Irregular Loop Specialization
CGRA energy profiles for llist and dither

Ultra-Elastic CGRA (Performance Optimized) - (0.61V, 0.90V, 1.23V)
CGRA energy profiles for llist and dither

Ultra-Elastic CGRA (Energy Optimized) - (0.61V, 0.90V)
Summary: Energy Efficiency versus Performance

Elastic CGRA
- DVFS applied to entire array (not per PE)
- Trades performance for energy
- Trades energy for performance
- Cannot achieve both (cannot enter the upper right quadrant)
Summary: Energy Efficiency versus Performance

Ultra-Elastic CGRA (Energy Opt)
- Up to 2.32x energy efficient
Summary: Energy Efficiency versus Performance

Ultra-Elastic CGRA (Energy Opt)
- Up to 2.32x energy efficient

Ultra-Elastic CGRA (Perf Opt)
- Up to 1.50x higher throughput
Summary: Energy Efficiency versus Performance

Ultra-Elastic CGRA (Energy Opt)
- Up to 2.32x energy efficient

Ultra-Elastic CGRA (Perf Opt)
- Up to 1.50x higher throughput

Compiler can control where in this space we target
Ultra-Elastic CGRAs: Takeaways

A cross-stack approach for compiler-configurable per-PE fine-grain DVFS

Inter-iteration loop dependencies are an important bottleneck for irregular loops.

Ultra-elastic CGRAs leverage elasticity and compiler-configurable fine-grain DVFS to specialize CGRAs for both regular and irregular loops.

Our quantized approach to fine-grain DVFS is fully verifiable with commercial STA tools.