RipTide: *A Programmable, Energy-Minimal Dataflow Compiler and Architecture*

Graham Gobieski, **Souradip Ghosh**, Marijn Heule, Todd C. Mowry, Tony Nowatzki*, Nathan Beckmann, Brandon Lucia

*Carnegie Mellon University, *UCLA | (SRC Task 3019.001)*

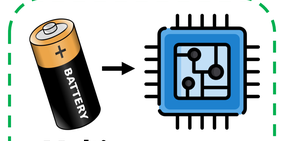
Smart sensor devices at the extreme edge are emerging with huge industrial impact

Trillions *of devices[1]* + ***sophisticated*** *apps*

Must sustainably & efficiently compute at the edge. How?

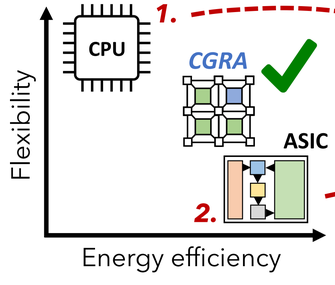
1. Run apps on ***ultra low power*** (*ULP*), µWs

2. More **compute on-device**, less communication[2]



Multi-year sensor deployment

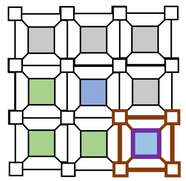
Goal: build a highly flexible & energy efficient compute



1. Wastes up to **90% energy** on non-compute[3]

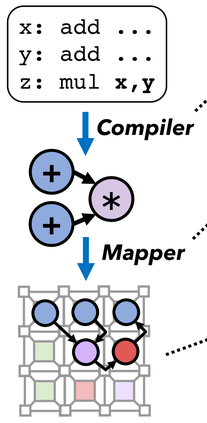
2. Limited to **single app**

Coarse-grained reconfigurable arrays are flexible & efficient!



Grid of processing elements (**PE**) connected by a **NoC**.

Eliminates **fetch/decode & reg. file usage!**



1. Extract a *dataflow graph* from code

2. *Map* ops to a PE mix and links on the NoC

3. Execute ops w/ "*dataflow firing*" or a *static schedule*

Prior ULP CGRAs are limited

```
void foo (...):
 for (i = 0..n):
   vlh v1, a + i
   vadd v3, v1, v2
   vsh b + i, v3
```

CGRA code in assembly[4]

Runs ***only*** **affine inner loops.** *No* irregularity or operation ordering.

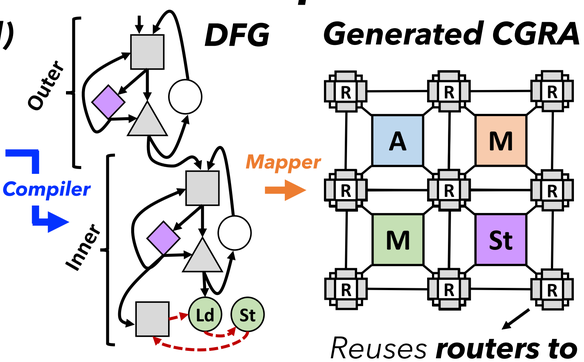
Insight: To improve efficiency, CGRAs need to run entire apps & support common PL idioms

RipTide is a new ULP CGRA compiler & arch.

C code (lightly annotated)

```
#riptide void foo
(int * restrict a, b) {
  while (!q.empty()) {
    for (i in 0..q.pop())
      if (b[a[i]]) ...
  }
}
```

Handles arbitrary code via
1) *Complex* ***control-flow***
2) ***Irregular mem.*** *access*
3) *Operation* ***ordering***



Optimizes *DFG, fuses subgraphs*

Reuses ***routers to run control-flow.*** *Frees PEs for ops.*

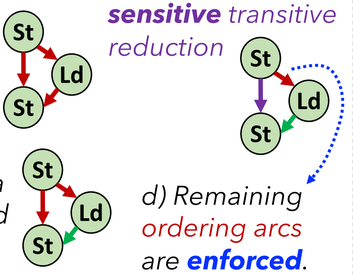
Memory ordering:

a) Build an ***ordering graph*** *for mem. deps*

b) ***Prune*** *arcs via existing data and control deps*

c) Perform a ***path-sensitive*** *transitive reduction*

d) Remaining ordering arcs are ***enforced****.*



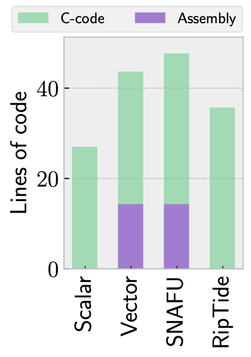
Mapping DFGs to the CGRA:

RipTide *formulates PE/link place & route as* **SAT** or **ILP problems**.

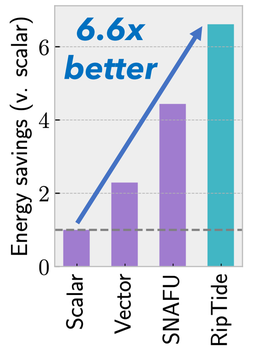
SAT: *Adequate soln. Quickly solves.*

ILP: *Better soln. Slowly solves.*

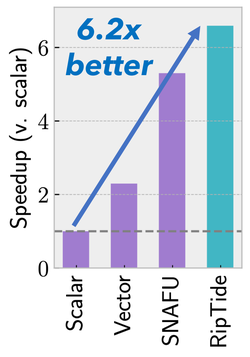
Programmability



Energy-Efficiency



Performance



10 apps from linalg, graph processing, and signal processing

+ Full compiler built with **LLVM**

+ Full RTL design and **synthesized**

+ Ran an ***entire DNN*** *on* ***RipTide****!*

[1]Arm, "How to build a trillion connected things."
[2]Gobieski et al., "Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems." (ASPLOS '19).
[3]Horowitz, "Computing's energy problem (and what we can do about it)." (ISSCC '14).
[4]Gobieski et al., "SNAFU: An Ultra-Low-Power, Energy-Minimal CGRA-Generation Framework and Architecture." (ISCA '21).

CMU UCLA