Transmuter: A Reconfigurable Computer

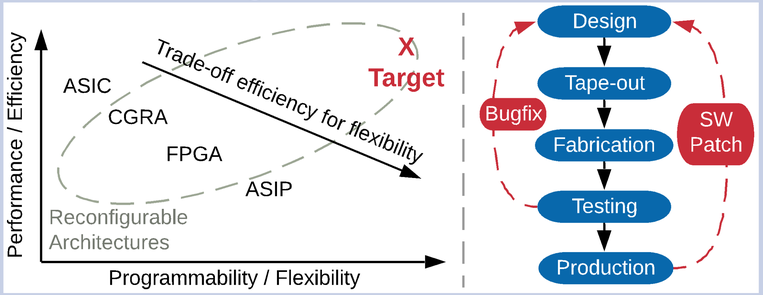
Ronald Dreslinski (PI), Trevor Mudge, Hun-Seok Kim, David Blaauw (University of Michigan)
Chaitali Chakrabarti (Arizona State University), Michael O'Boyle, Murray Cole (University of Edinburgh)

ELECTRONICS RESURGENCE INITIATIVE SUMMIT & MTO Symposium 2020 | Seattle, WA | August 18-20

Software-Defined Hardware

Artificial Intelligence

Background and SoTA



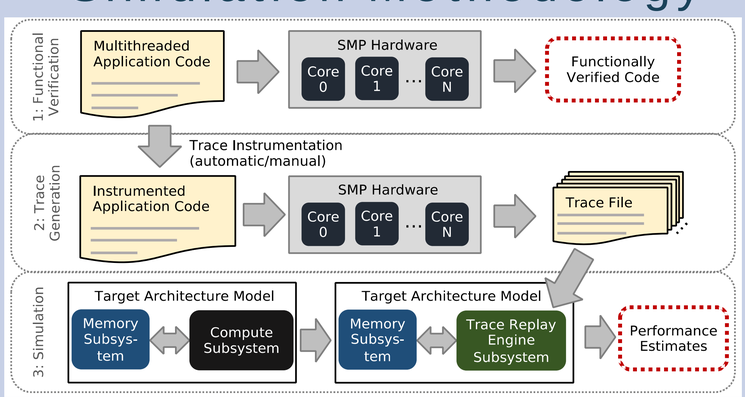
- Existing platforms are historically bounded by three conflicting constraints: **programmability**, **specificity** & **efficiency**
- ASICs compromise on generality and limit their functionality to one algorithm
- FPGAs enable fast prototyping but suffer from large cost, power and area overheads. They also have high reconfiguration overheads: O(ms)-O(µs)
- CGRAs overcome some of inefficiencies by reconfiguring at a coarser granularity, but require customized software stacks and tool support
- CPUs and GPUs are the de facto choice for programmers as they provide high flexibility and programmer-friendly semantics, but lack specialization
- **SDH Goal: to build hardware and software that enable application reconfigurable high performance without sacrificing programmability**

Targets for SDH

Goals	TA1 vs. CPU*	ASIC vs. TA1	ASIC vs. TA1 (sparse)	TA2 Programmability
Phase 1	100-300x	Within 50x	~1x	within 10x
Phase 2	100-300x	Within 10x	2x better	within 10x
Phase 3	500-1000x	Within 5x	8-10x better	~1x

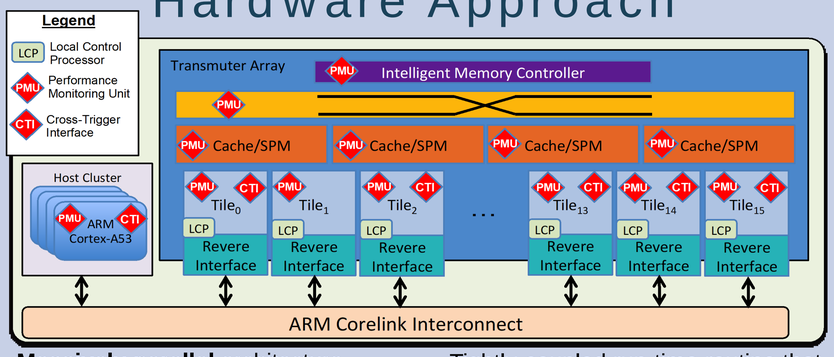
- Target algorithms in the domains of **machine learning** and **data science** that process large volumes of data and involve intense linear algebra
- Develop a system that allows data-intensive algorithms to run at near-ASIC efficiency without the cost, development time or single application limitations associated with ASICs **(TA1 – HW architecture; TA2 – SW environment)**

Simulation Methodology



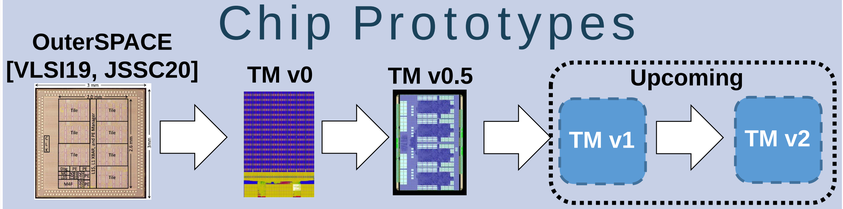
- Development of a robust simulation methodology enables rapid design space exploration. Our trace based approach improves simulation speed by **~12x**.

Hardware Approach



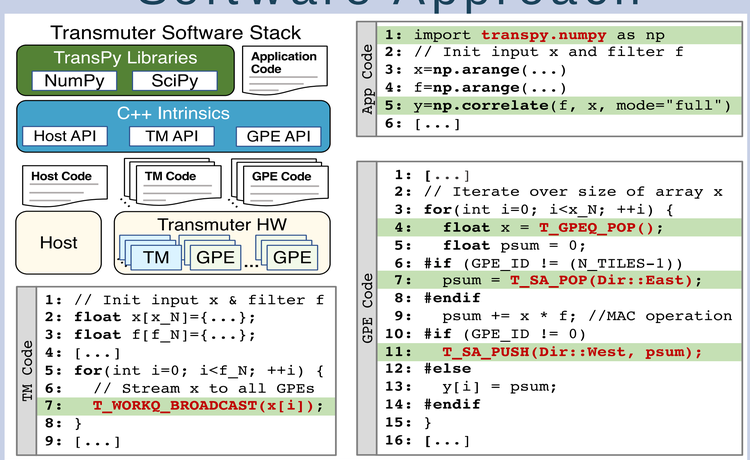
- **Massively parallel** architecture employing a SIMD/MIMD paradigm and in-order Arm cores
- **Fast reconfigurable** within O(10) cycles supporting modes such as cache/scratchpad/queue, systolic, etc.
- Tightly-coupled run-time routine that monitors hardware via **introspection**
- **Intelligent memory controller** for better data marshaling to maximize off-chip bandwidth utilization
- High-Speed **XBAR interconnect**

Chip Prototypes



- Prototyping done in **5-phase progression**
- **OuterSPACE**: Sparse Matrix Multiplication
  - Small substrate
  - Cache/scratchpad/xbar reconfiguration
  - Decoupled access/execute
- **TM v0** and **v0.5**: multiple applications
  - Streaming crossbar
  - Register-to-register tunneling
  - DRAM emulation
- **TM v1**: multiple applications
  - Medium substrate + host
  - TM<->host interface
  - Software stack
  - Higher bandwidth off-chip
- **TM v2**: multiple applications
  - Large substrate + host
  - Malleable prefetcher
  - Data packing/restructure

Software Approach



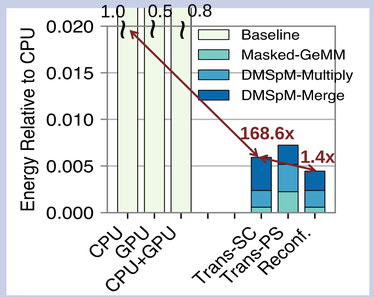
- User writes Python programs and adds "transpy" to the import call
- Code is generated & compiled under-the-hood to run on Transmuter hardware
- Achieved Phase 3 goal of **~1x programmability**

Key Takeaways

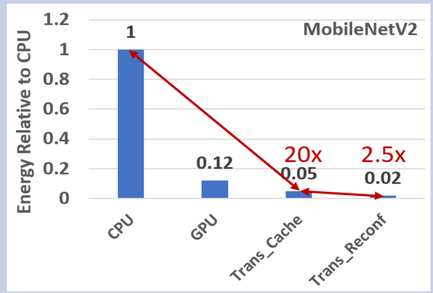
Phase 1	TA1 vs. CPU*	ASIC vs. TA1	ASIC vs. TA1 (sparse)	TA2 Programmability
Goals	100-300x	Within 50x	~1x	within 10x
Achieved	**503x**	**Within 33.05x**	**0.75x**	**2.6% Overhead**

- Significant energy reduction using reconfiguration
  - **236x** for Sinkhorn (dense/sparse OT app); **50x** for MobileNetV2 (dense NN)
- **1x** programmability using drop-in replacements for NumPy, SciPy, etc.
- **6.2x-26.3x** improvement using delayed execution

Comparison Against SoTA



- Evaluation on Sinkhorn, a workload with two sparse kernels: masked-GeMM and Dense*Sparse MM
- **168.6x** energy reduction vs. i7 CPU
- Additional **1.4x** reduction using cache mode & sharing reconfiguration
- Combined reduction of **236x**



- Evaluation on MobileNetV2, which is a deep neural network using point-wise convolution and depth-wise convolution
- **20x** energy reduction vs. i9 CPU
- Additional **2.5x** reduction using R2R systolic array mode
- Combined reduction of **50x**

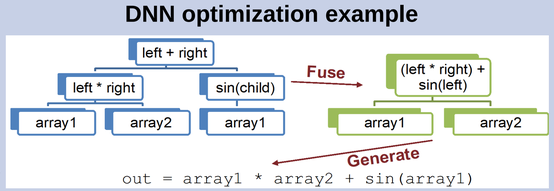
Software Evaluation

- Lift: using rewrite rules and HW constraints to generate optimized code
- **10x** faster than direct convolution (hand-written)
- Perf. on par with hand-written GeMM version with **3x** less memory consumption

```
map2d(slideWin: [float]inChs*kerW*kerH =>
  map(singleK =>
    reduce(+, 0, map(*,
      zip(join(join(slideWin)),
          join(join(singleK))))),
    ks)
  slide2D(kerH, kerStepY,
          kerW, kerStepX, in))
```

DNN optimization example

- DelayReplay: avoiding repeated host-to-device transfers thru delayed execution
- **26.3x** speedup vs NumPy
- **6.2x** speedup vs CuPy



- **2.5x** memory usage improvement

Transition Effort

- Arm will transition the technology developed on our program by making hardware IP and software drivers available to commercial and DoD users
- If you are interested in being a potential user of the IP, please contact us at **transmuter-ip@umich.edu**

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA).
The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.
Distribution Statement A – Approved for Public Release, Distribution Unlimited