Fast Trace-Driven Simulation of Programmable Heterogeneous Accelerators

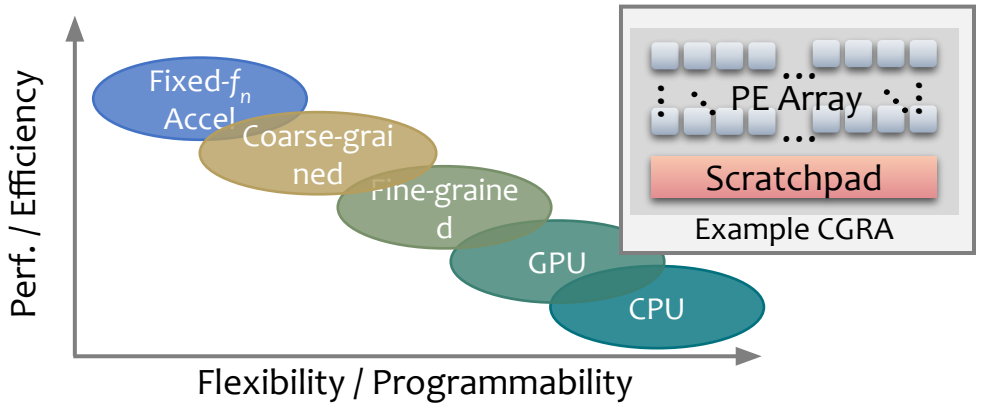
Subhankar Pal[^*], Kuba Kaszyk[†], Siying Feng[*], Björn Franke[†], Murray Cole[†], Michael O'Boyle[†], Trevor Mudge[*], Ronald Dreslinski[*]

[*]University of Michigan, USA [†]University of Edinburgh, UK

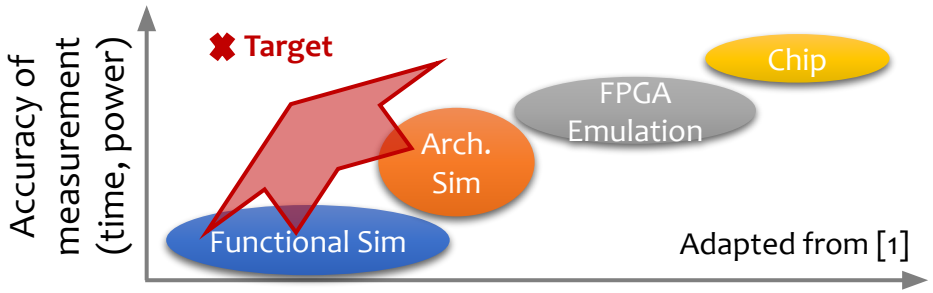
^Work done when author was a PhD student at Michigan. Author is now with IBM Research.

Programmable Accelerators and the Need for Fast Simulation

- End of Dennard scaling and Moore's law spurred heterogeneous architectures
  - Fixed function accelerators are the holy grail for the best performance and perf/Watt, but…
  - Programmable accelerators e.g. coarse-grained reconfigurable architectures (**CGRAs**) gaining popularity
- Early-stage evaluation critical to develop new architectures
  - Cycle-level architectural simulators often 3-5 orders of magnitude slower than silicon
  - Trace-driven simulation widely adopted to run much faster simulations at the cost of some accuracy
- Shift of research from accelerating compute-bound algorithms to **memory-bound** ones
  - We exploit this insight and thus retain fidelity for memory operations while squeezing speedup out of approximating simulation of compute operations



Trade-Offs in Modern Computing Paradigms



Trade-Offs between Accuracy and Resource Requirement

[1] Takamaeda-Yamazaki et al., "An FPGA-based scalable simulation accelerator for tile architectures", ACM SIGARCH Computer Architecture News, December 2011

HetSim – Features in a Nutshell

- Operation coarsening
- Selection of address space
- Instruction allow/denylisting

- Traces that captures additional information to improve flexibility
  - Tokens defining inter-PE communication
  - Annotations of dependent memory addresses
  - Virtual program counters, to support PC-based prefetching
- Support for hardware "primitives" that are common to modern heterogeneous systems
- Extendibility to support user-defined primitives
- High-level "specification" file that models the behavior of primitives, filtration criteria for instructions, etc.
- Scalable to heterogeneous targets w/ core counts of the order of 1,000

```
[...]
STALL 100 ( ) // startup latency
POP 0 2 // mgr ID on mgr<->wrkr bus
POP 0 2
POP 0 2
LD @10 0x2000 ( ) // A[0]
LD @11 0x3000 ( ) // B[0]
STALL 1 ( 0x2000 0x3000 )
                  // compute A[0]+B[0]
ST @12 0x4000 ( ) // C[0]=A[0]+B[0]
PUSH 0 0 // mgr ID on mgr<->wrkr bus
[...]
```

Example trace file

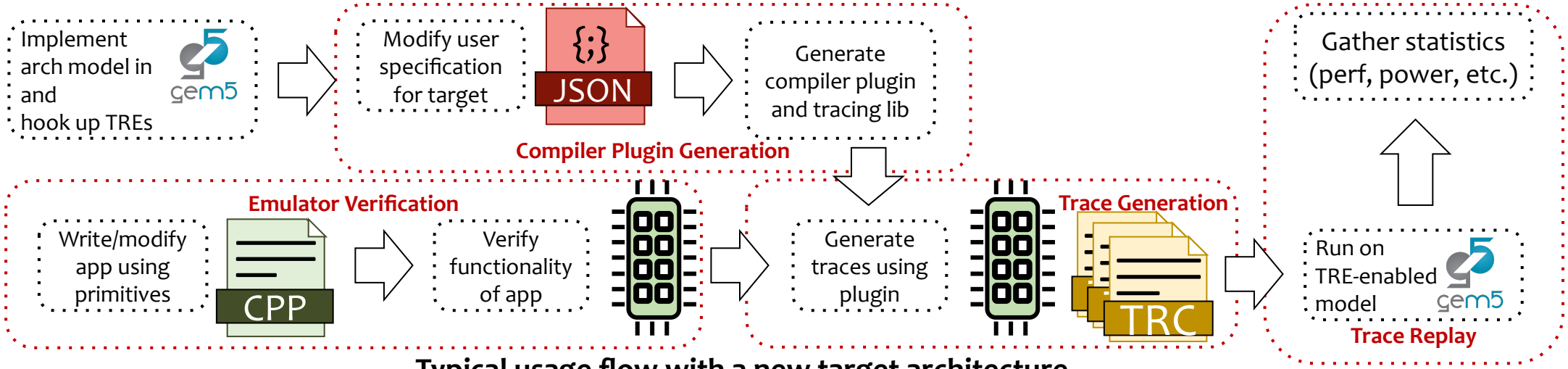
__store_uncache(&a,v)
__store_block_uncache (&a,v)
__barrier_init(&b,n)
__barrier_wait(&b)
__mutex_lock(&m)
__mutex_unlock(&m)
__sleep()
__signal(id)
__push(dir,v)
__pop(dir)

Examples of supported primitives

```
"addr_space": {
    "set1": {
        "start": "0x0000000000000000",
        "end": "0xFFFFFFFFFFFFFFFF"
    }
},
"llvm_instr": ["Add","FAdd","Mul","FMul","Br","FCmp","ICmp"],
"pe": {
    "mgr": {
        "id": [0],
        "__push(unsigned int, unsigned long)" : {
            "token": "PUSH #0",
            "cycles": 1,
            "enable": 1
        },
        "__push_mmap(unsigned int, unsigned long)" : {
            "token": "PUSHMMAP #0",
            "enable": 0
        },
        "__barrier_init(pthread_barrier_t*, unsigned int)": {
            "token": "BARINIT #0 #1",
            "cycles": 10,
            "enable": 1
        },
```

Example user specification file

Proposed Approach

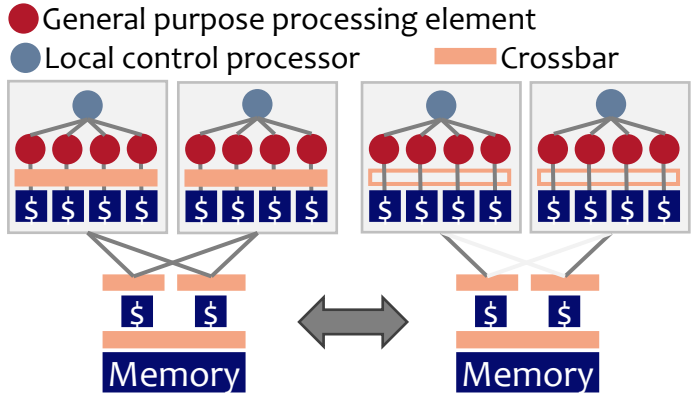


Typical usage flow with a new target architecture

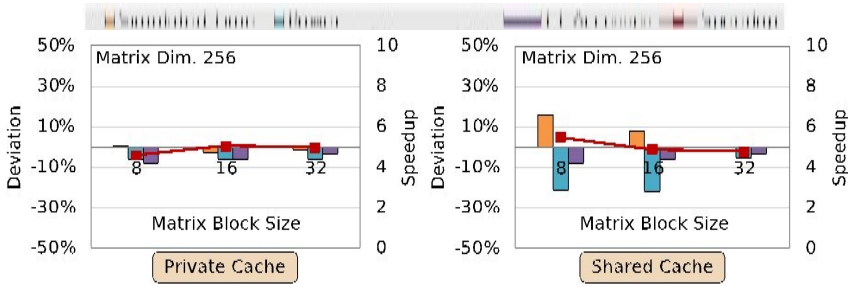
- **Compiler Plugin Generation.** generator script parses the user specification file and generates a compiler pass and tracing library specific to the target architecture
- **Emulator Verification.** user writes a multithreaded application for the target with calls to primitives, and verifies the functionality by running on a native machine
- **Trace Generation.** user generates instrumented version of application using the compiler pass and tracing library generated by HetSim, and runs it through the native machine to generate trace files
- **Trace Replay.** user runs the traces through the gem5 model to obtain performance and power estimates at faster timescales

Evaluation with Detailed Model and Chip

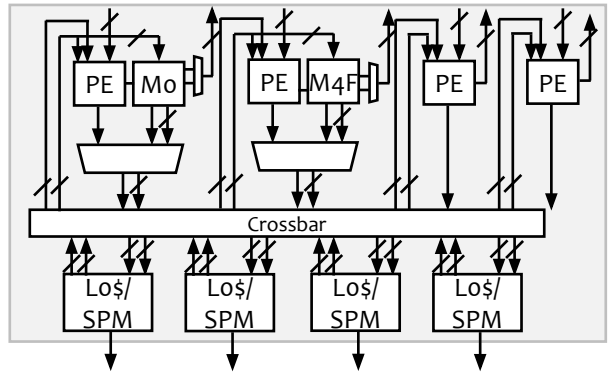
- Evaluation with architecture called Transmuter [2], a reconfigurable accelerator
  - Two hardware configurations: **shared cache** and **private cache**
  - Three workloads: **GeMM**, **GeMV**, and **SpMM**
- Average speedup of **5.0×** over detailed gem5 model, with timing and power deviations of **15.1%** and **10.9%** on average
- Validation against SpMM accelerator prototype chip
- Timing deviation: **32%** and **16%** for multiply and merge phases



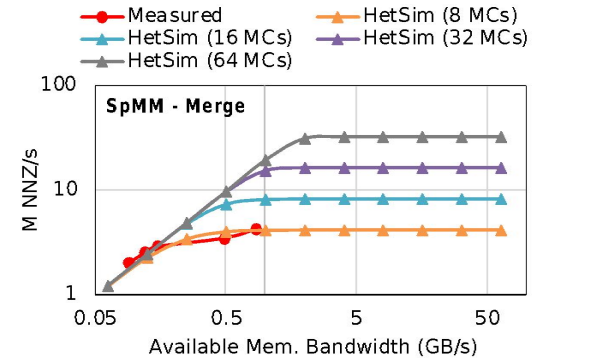
Overview of Transmuter



Speedup and deviations over detailed (GeMM)



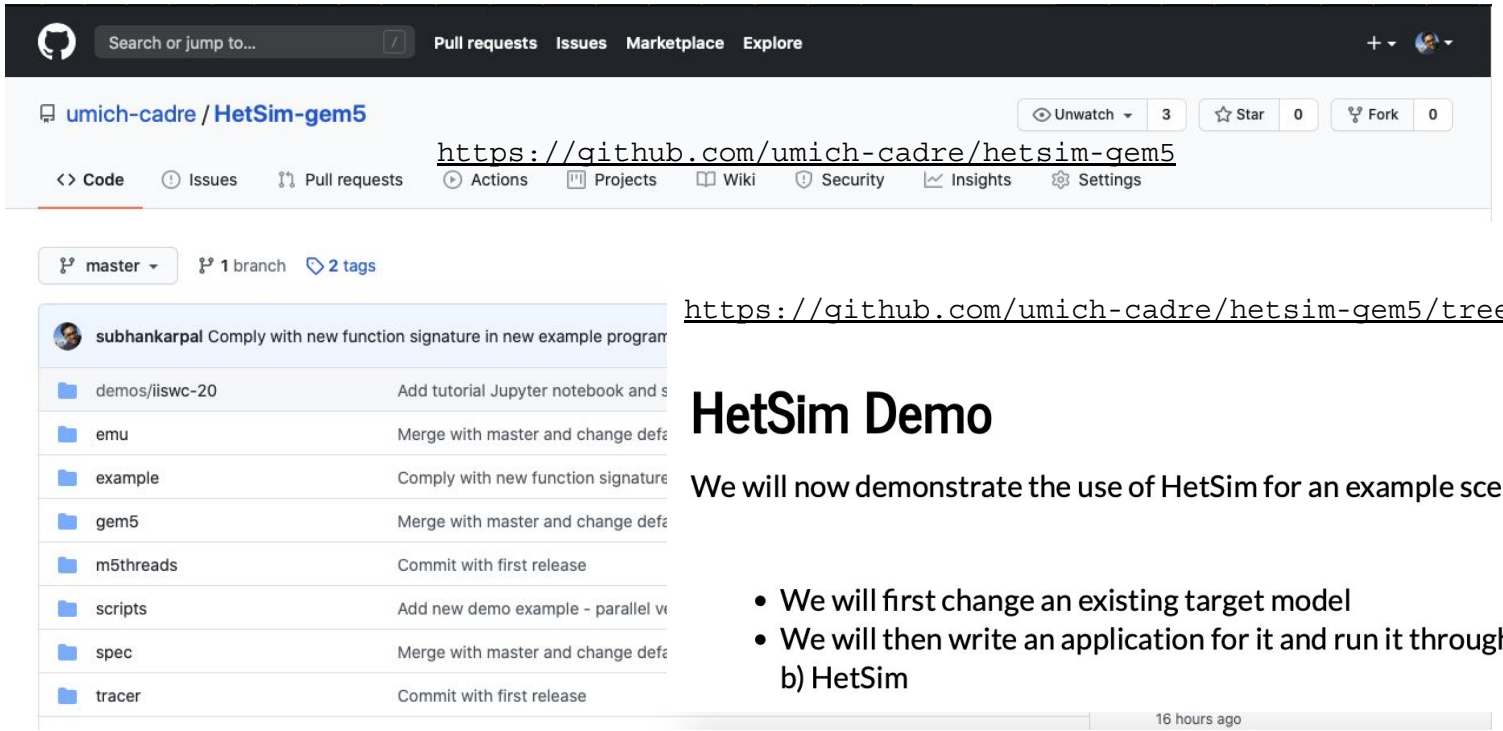
40 nm SpMM chip design (Tile View)



Deviation in Performance Estimation w.r.t. Chip

[2] Pal et al., "Transmuter: Bridging the Efficiency Gap using Memory and Dataflow Reconfiguration", ACM/IEEE PACT, October 2020

HetSim is Available on GitHub!



https://github.com/umich-cadre/hetsim-gem5

https://github.com/umich-cadre/hetsim-gem5/tree/master/demos/iiswc-20/tutorial.ipynb

HetSim Demo

We will now demonstrate the use of HetSim for an example scenarios.

- We will first change an existing target model
- We will then write an application for it and run it through a) detailed simulation, and b) HetSim

For details and preview of ongoing work, please join me in the breakout room!