HETSIM: Simulating Large-Scale Heterogeneous Systems using a Trace-driven, Synchronization and Dependency-Aware Framework

HETSIM: Simulating Large-Scale Heterogeneous Systems using a Trace-driven, Synchronization and Dependency-Aware Framework *(Advances in ModSim Implementation)*

Subhankar Pal*, Kuba Kaszyk†, Murray Cole†, Michael O'Boyle†, Ronald Dreslinski* — Universities of *Michigan and †Edinburgh

INTRODUCTION – Early-stage design-space exploration (DSE) and performance/power evaluation of large-scale heterogeneous systems, such as those composed of chip multiprocessors (CMPs) coupled with fixed-function logic, have gained importance in the dark silicon era. Prior work has explored trace-based simulation techniques, that offer good trade-offs between *simulation accuracy* and *speed*, to simulate CMPs with up to 100s of threads, and similar methods are used for accelerators. However, there is lack of a unified framework for fast simulation of large-scale *heterogeneous* systems. In this work, we propose HETSIM, a trace-driven framework for estimating performance and power of accelerators, CMPs and heterogeneous systems with 1000s of cores. We present results on (*i*) a CMP and (*ii*) a heterogeneous accelerator, demonstrating average speedups of **5.5**× and **16.1**× over detailed gem5 models with deviations in *simulated time* and *power consumption* of **4.6-28.1**% and **1.7-3.3**%, respectively.

PROPOSED APPROACH – Figure 1 summarizes the approach used in HETSIM. The first step involves executing a multithreaded version of the application on a native multiprocessing system to verify correctness. Next, the application code is instrumented with trace-generating function calls, and run through the native system to generate trace files – one per thread/core/processing element (PE) in the target architecture. Instrumentation is a one-time overhead for DSE of the target's memory subsystem. To further reduce the burden on the end-user, we provide an LLVM-based compiler pass, which automatically identifies target-specific intrinsics and memory accesses, and re-compiles the code with instrumentation. Lastly, the compute units in a gem5 model of the target are swapped with trace replay engines (TREs) that execute the "instructions" in their corresponding trace files, according to the rates at which the cores/PEs would issue them.

CAPTURED OPERATIONS – HETSIM captures broad classes of operations that appear in heterogeneous systems. Within a region-of-interest in the application, HETSIM captures:

Memory Operations: Memory accesses are captured with high fidelity using `LD`/`ST` tokens followed by their address. In addition, a *dependency list* for memory operations allows HETSIM to model flexible target architectures, such as complex in-order (InO) cores that support prefetch instructions and multiple outstanding loads.

Computation: HETSIM encodes all non-memory instructions, such as arithmetic ops, branches, etc. using the `STALL` token. Consecutive `STALL`s in a trace file are coalesced for faster trace replay and reduced trace storage. A TRE scales the number of `STALL`s based on the level of acceleration in the target.

Communication: Tokens such as `PUSH` and `POP` followed by the core ID are used to perform buffered pushes and pops of data. These are universal primitives for fast PE-to-PE communication in accelerators, such as systolic arrays. `SIGNAL` and `WAIT` are common primitives employed for handshaking between different hardware blocks. `BARRIER` is another useful primitive for synchronization across a set of PEs. Finally, `LOCK`/`UNLOCK`, in addition to the signaling and barrier synchronization primitives, are used to model Pthreads calls for CMP systems.

EVALUATION – We provide a summary of our evaluation of HETSIM for the DSE of two target systems.

Target 1: In-Order Manycore CMP – We evaluate a 32-128 core CMP system with shared 16 kB L1 (1 slice/core) and 256 kB L2 (4 slices) caches, executing matrix multiplication (GeMM). Figure 2 shows the execution-time deviation and speedup of HETSIM over a gem5 model that uses MinorCPU cores. We have also simulated this system with up to 4096 cores using HETSIM on a 64-core Threadripper 2990WX CPU with 128 GB of RAM (not shown).

Target 2: Heterogeneous Sparse Matrix Multiplication (SpMM) Accelerator – We deployed HETSIM for scalability studies on a heterogeneous SpMM accelerator prototype chip [1] that uses a tiled architecture – 8 tiles with 4 custom PEs and 2 Arm Cortex-M cores per tile. The algorithm is split into two phases – a *multiply* phase that uses caches and a *merge* phase that uses scratchpads.

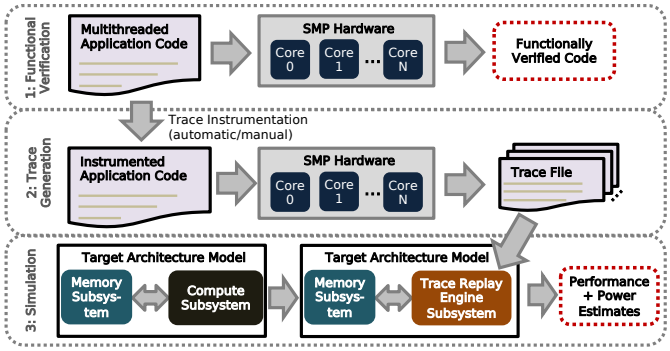

Fig. 1. Trace-based simulation approach deployed in the proposed HETSIM.

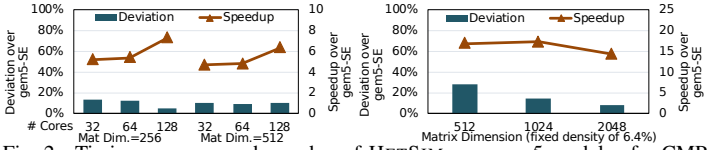

Fig. 2. Timing accuracy and speedup of HETSIM over gem5 models of a CMP system executing GeMM (left) and a heterogeneous SpMM accelerator (right).

Figure 2 (right) plots the accuracy and speedup of HETSIM over a gem5 core model of the accelerator that uses MinorCPUs for the PEs as well as the Arm cores. The average deviation of HETSIM from the measured chip performance is 34.6% for *multiply* and 4.0% for *merge* in our scalability studies (see Fig. 5 in [1]).

Overall, the *timing* and *power* deviations range from 4.6-28.1% to 1.7-3.3%, respectively, over detailed gem5 models, with a speedup of 4.7-17.3× (up to 8.6× for non-DSE experiments). Note that hand-annotated traces were used for Target 2, and automating trace instrumentation for accelerators is work-in-progress.

TABLE I
COMPARISON OF HETSIM WITH PRIOR TRACE-DRIVEN FRAMEWORKS.

Work	ISA	Thread-ing	Exec.	Sim. Limit	Synchro-nization	Target Platform	Trace Gen./ Replay
Elastic Traces [2]	Agnostic	Single	OoO	-	-	CMP	gem5/gem5
ElasticSim-MATE [3]	Armv7/8	Multi	OoO	128	OpenMP	CMP	gem5/gem5
Synchro-Trace [4]	Agnostic	Multi	InO	64	Pthreads/ OpenMP	CMP	Native/gem5
SST/ Macro [5]	N/A	Multi	OoO	1000	MPI	**Multi-CMP Sys.**	Native/Custom
HETSIM	**Agnostic**	**Multi**	**InO**	**4096**	**Pthreads/ Custom**	**CMP/accel. /hetero.**	**Native/gem5**

RELATED WORK – A few works have explored similar trace-driven methodologies as HETSIM, albeit only for simulating relatively small-scale CMP systems. We provide a qualitative comparison over these work in Table I. One close work, Synchro-Trace [4], uses dependency and synchronization aware traces for CMP systems with simple in-order cores. In contrast, HETSIM is applicable to CMPs, accelerators and heterogeneous targets, as well as offers flexibility to simulate complex in-order cores.

DISCUSSION – HETSIM addresses the issue of simulating heterogeneous systems with 1000s of cores within practical constraints. We have used HETSIM to evaluate the impact of bandwidth and clock speed scaling on a heterogeneous accelerator. We are in the process of using the same for a reconfigurable system in a multi-University project. The current effort is focused on automatic trace instrumentation, to make it more accurate and robust and support heterogeneous systems.

REFERENCES

[1] S. Pal *et al.*, "A 7.3 M Output Non-Zeros/J Sparse Matrix-Matrix Multiplication Accelerator using Memory Reconfiguration in 40 nm," *VLSI*, 2019.
[2] R. Jagtap *et al.*, "Exploring system performance using elastic traces: Fast, accurate and portable," *SAMOS*, 2017.
[3] A. Nocua *et al.*, "ElasticSimMATE: A fast and accurate gem5 trace-driven simulator for multicore systems," *ReCoSoC*, 2017.
[4] K. Sangaiah *et al.*, "SynchroTrace: Synchronization-Aware architecture-Agnostic traces for lightweight multicore simulation of CMP and HPC workloads," *TACO*, 2018.
[5] C. L. Janssen *et al.*, "A simulator for large-scale parallel computer architectures," *IJDST*, 2010.