SwapCodes: Error Codes for Hardware-Software Cooperative GPU Pipeline Error Detection

Michael B. Sullivan, Siva Kumar Sastry Hari, Brian Zimmer, Timothy Tsai, Stephen W. Keckler

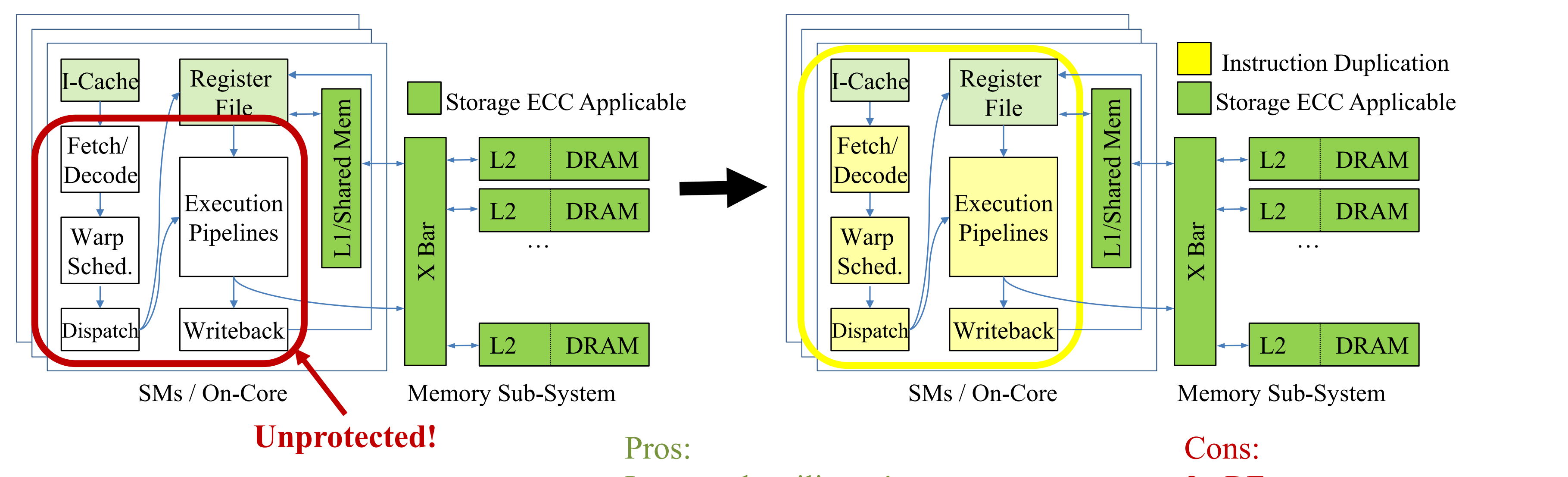
NVIDIA

Introduction

Compute-class GPUs use error-correcting codes (ECC) for storage and transmission, but leave the arithmetic pipelines unprotected.

Intra-thread instruction duplication is effective at lowering the overall GPU silent data corruption rate, but it comes with steep performance and power costs.

Intra-Thread Instruction Duplication:



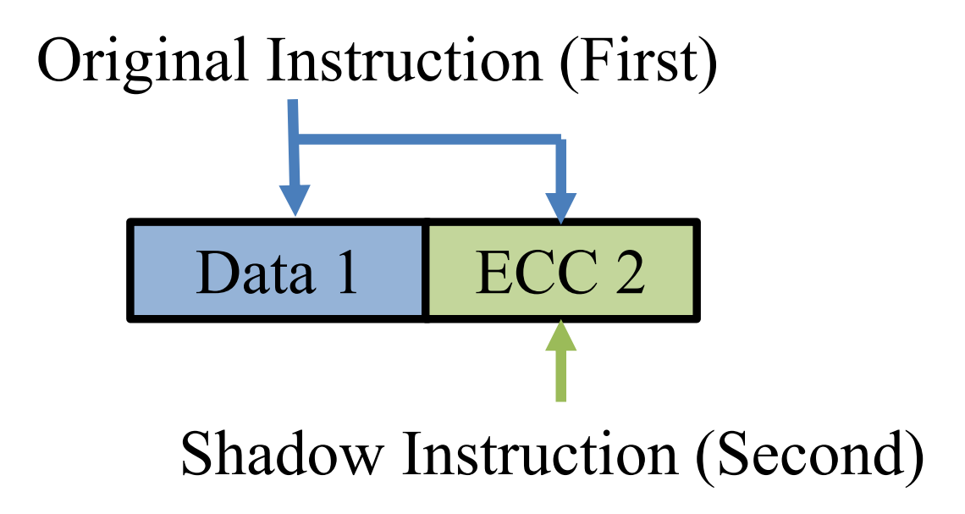
Pros:
Increased resilience!
(~10x SDC reduction in beam tests.)

Cons:
2x RF storage
> 2x instruction bloat
→ **49% average slowdown**

SwapCodes

SwapCodes is a family of hardware-software cooperative techniques to *accelerate intra-thread instruction duplication*.

SwapCodes duplicates instructions in the compiler, using a modified ISA to write the full output register with the first instruction but *only the ECC check-bits with the second*.



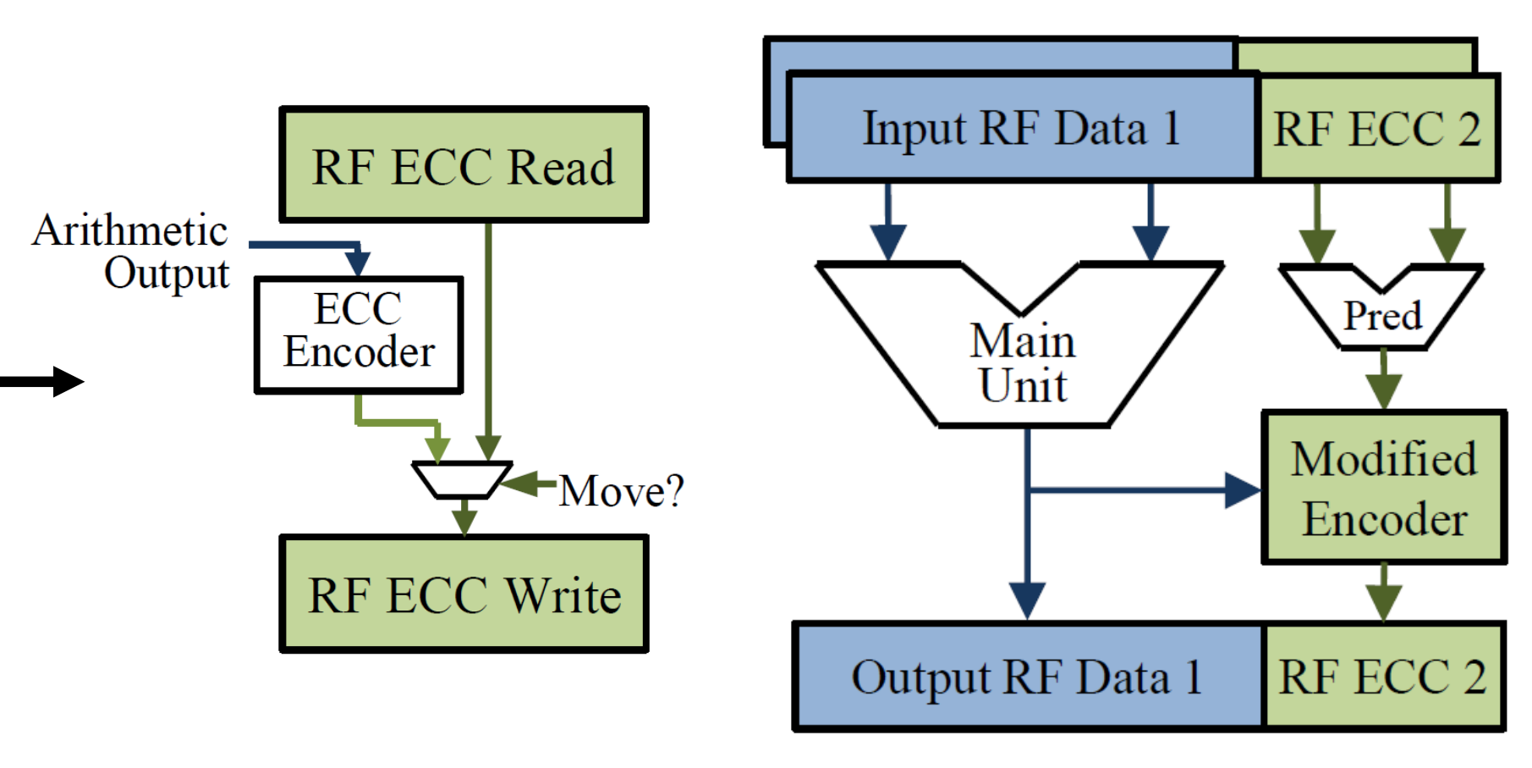
Thus, any single pipeline error will corrupt only the data-bits or only the check-bits of a register, allowing the ECC decoder to detect pipeline errors.

Swap-Predict

We also describe **Swap-Predict**, where operation-specific ECC prediction circuits can avoid the need to duplicate the most common operations.

Move Prediction:

Move operations are trivial to "predict". The ECC bits just need be propagated.



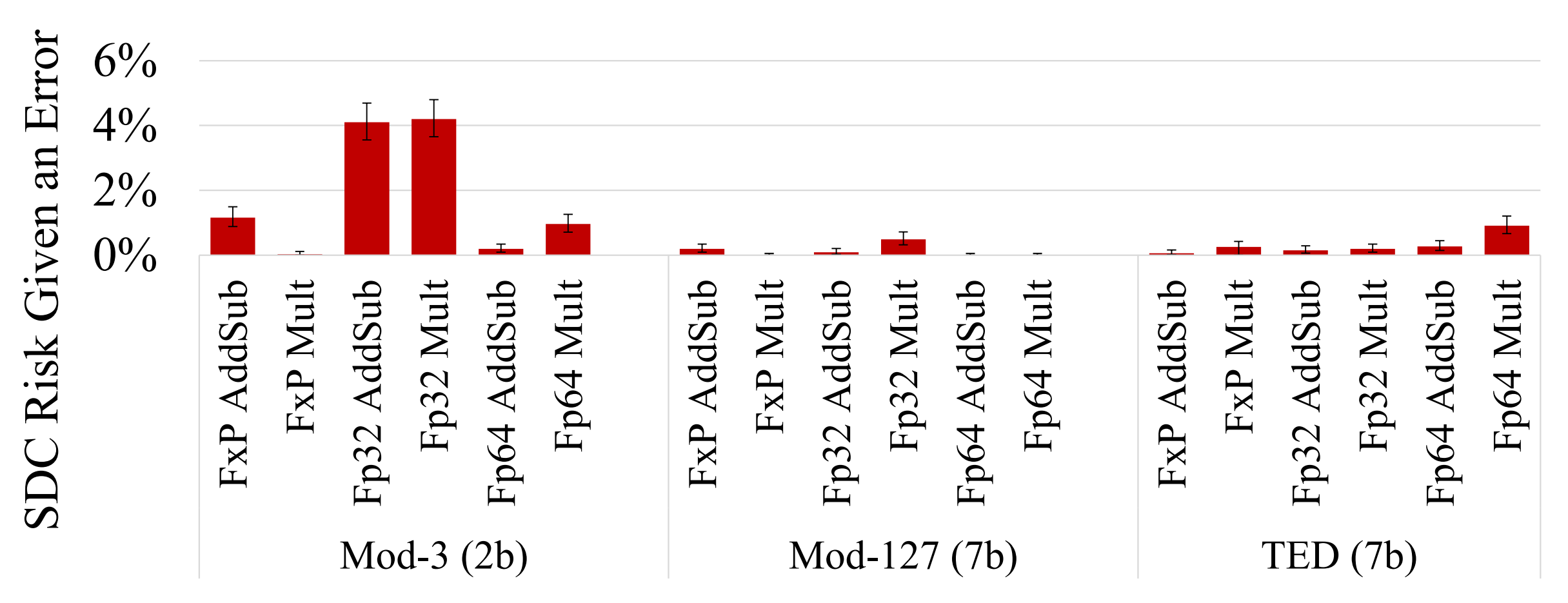
General Prediction:

Operation and ECC-specific prediction is also possible. We use residue codes for GPUs in the paper, and describe novel MAC prediction circuits.

Unlike concurrent checking, this is an optimization---we need not predict all operations!

Results: Resilience

SwapCodes works with any register file error detecting code.



SEC-DED / TED:

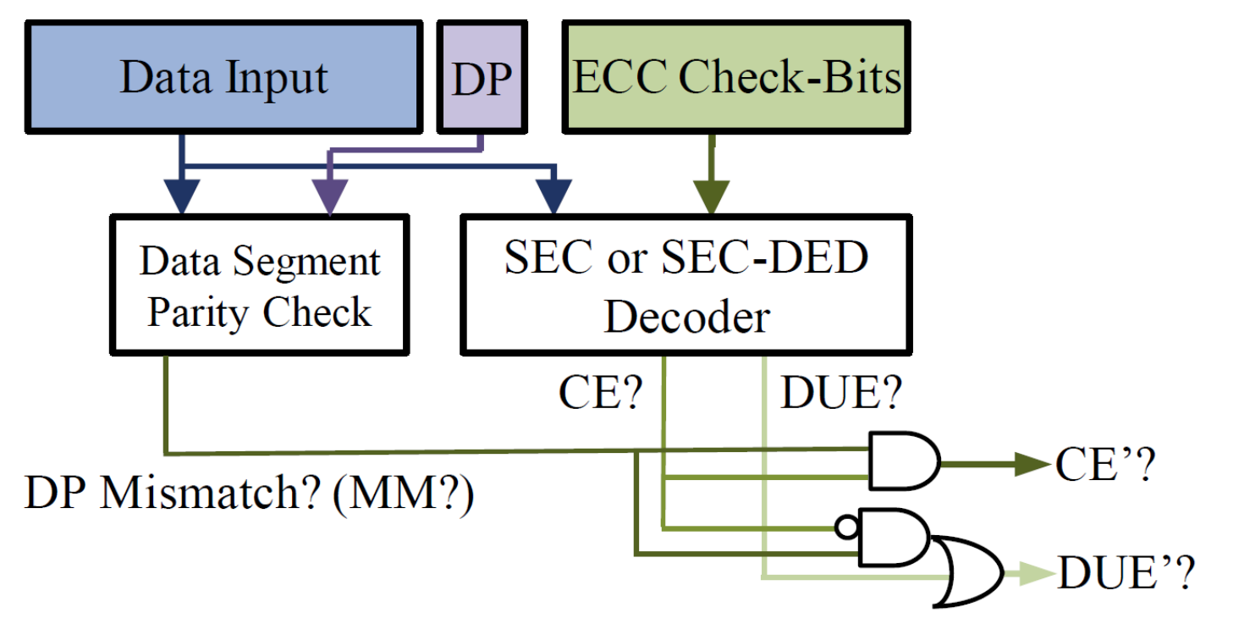
A SEC-DED code can correct single-bit storage errors while simultaneously providing triple-bit pipeline error detection.

Storage correction details:

Using gate-level fault injection, we see that SwapCodes detects more than **98.8%** of pipeline errors with a SEC-DED (TED) code and more than **99.3%** with an equal-redundancy residue code.

Preserving Single-Bit Error Correction

We describe two schemes to preserve storage correction without miscorrection risk for single-bit pipeline errors. These schemes use minor changes to the error reporting architecture, with no ECC decoder changes.

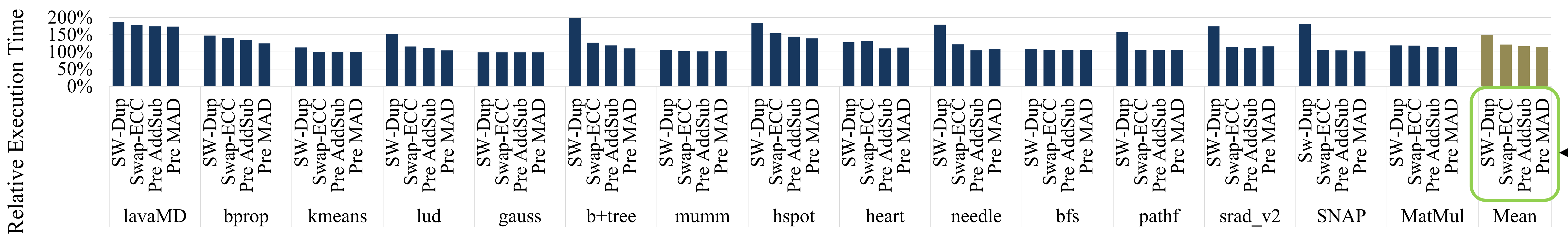


Results: Performance

SwapCodes roughly **halves the slowdown of instruction duplication** on average (49% vs 21% mean slowdown). The most performant Swap-Predict variant (Pre-MAD) incurs just **15% mean slowdown** over the un-duplicated program.

Methodology:

We use a modified compiler pass and run on an NVIDIA Tesla P100 compute-class GPU.

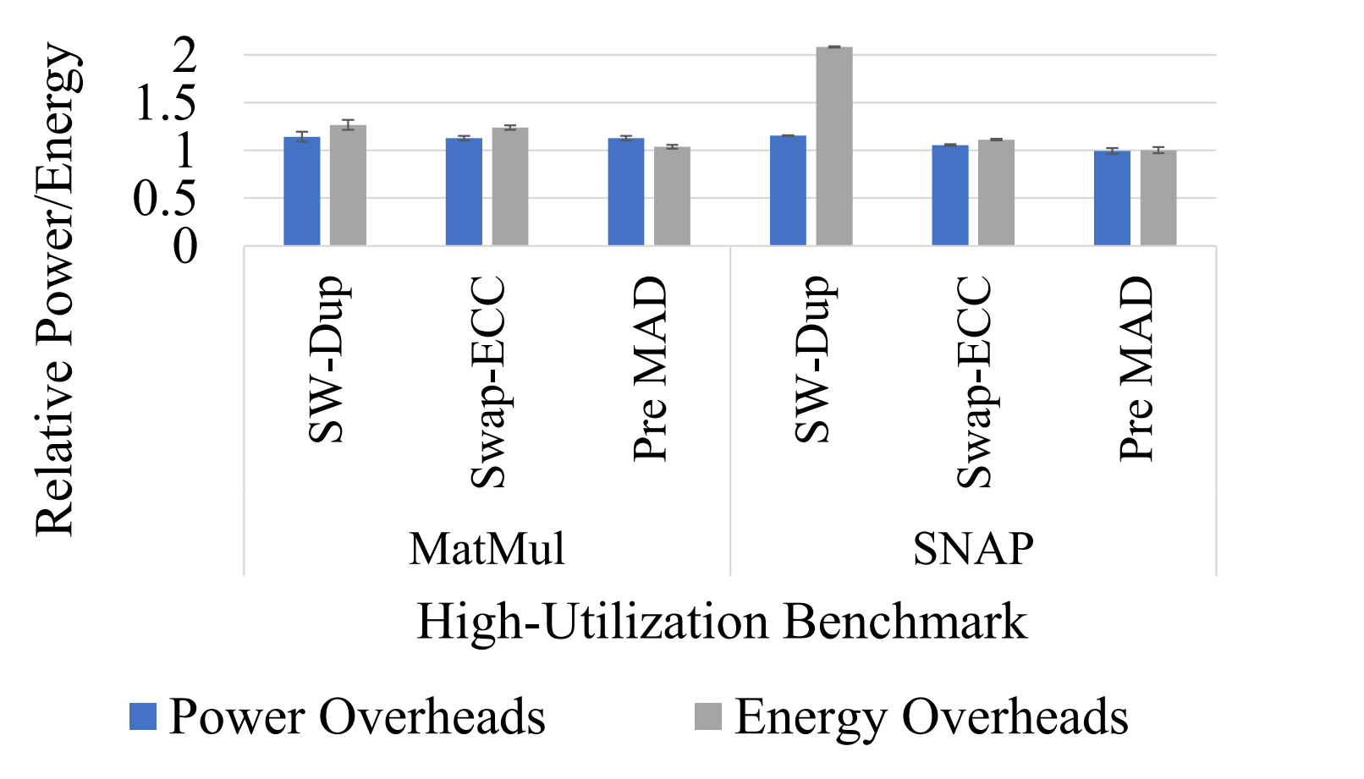


Legend:

SW-Dup: Instruction Duplication
Swap-ECC: SwapCodes Baseline
Pre-AddSub: Swap-Predict for ±
Pre-MAD: Swap-Predict for ±, ×

Results: Energy

SwapCodes' energy benefit is directly proportional to its performance improvement.



Hardware Changes

Hardware-software cooperative with modest hardware support:

TABLE: The Swap-ECC hardware and software changes.

Structure/Program	Swap-ECC Changes
Backend Compiler	Add an intra-thread duplication pass.
Backend Compiler	Swap-ECC-aware scheduling.
ISA Meta-Data	Add a 1b data write enable.
Register File	Add a data write enable and muxes for move propagation.
Error Reporting (Storage Correction)	Augmented error reporting to separate storage from pipeline errors.

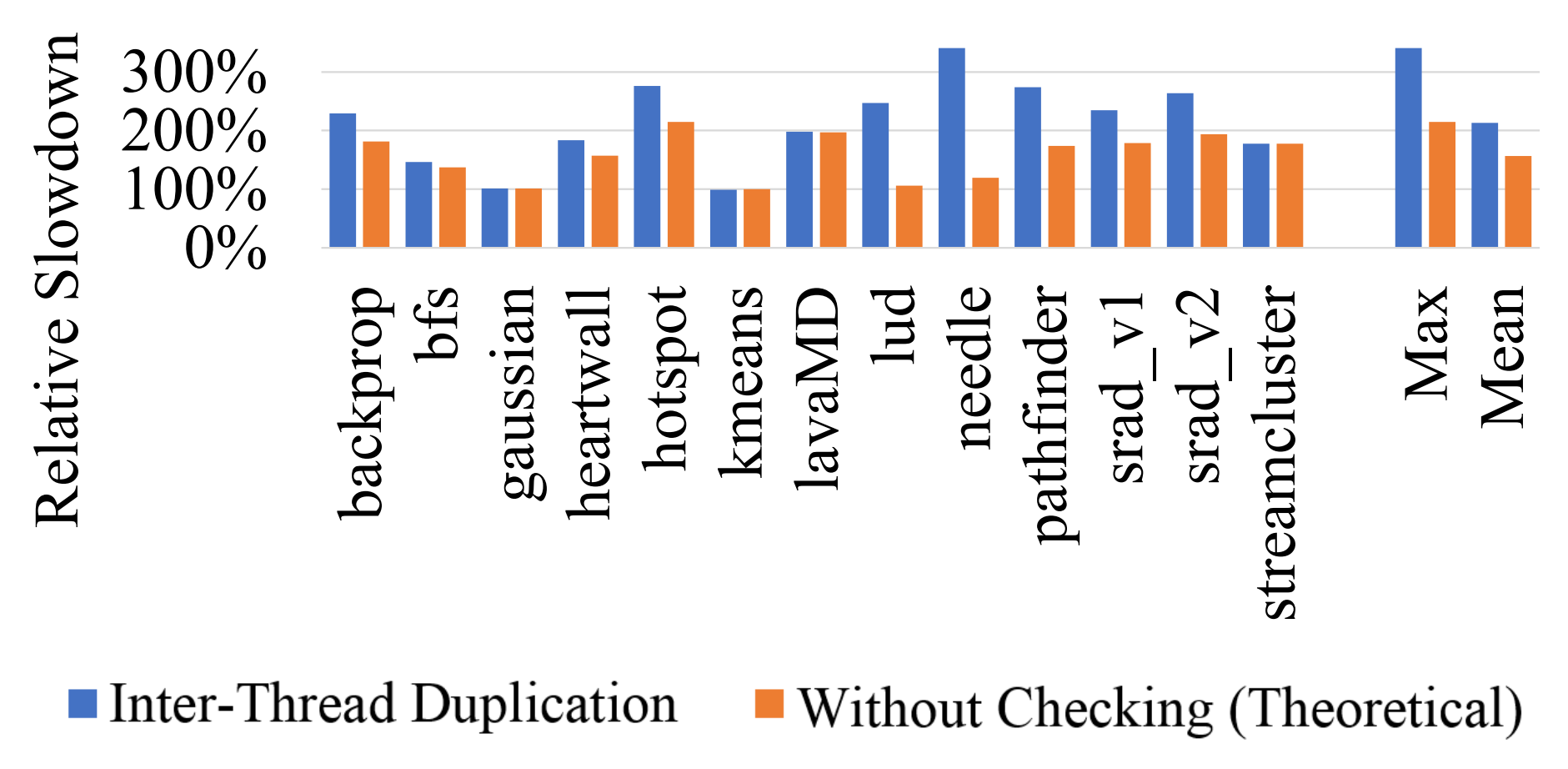
We evaluate these additions in a 16nm industrial library and show their area overheads to be nominal.

Alternative Techniques

SwapCodes improves intra-thread instruction duplication. There are other viable baselines, but none of them are as complete and programmer-transparent:

	High Level Duplication	Thread Duplication	Instruction Duplication	Concurrent Checking	SwapCodes
Transparent	No	No	Yes	Yes	**Yes**
H/W Changes	None	None	None	Many	**Few**
Perf. Overhead	High	High	Medium—High	None—Low	**Low**
Major Issue	Memory & Perf.	Threads & Perf.	Performance	Scope & Complexity	**None?**

In addition, our experiments indicate that thread duplication is not as performant as instruction duplication:



Advantages

SwapCodes reuses the existing register file ECC decoder to detect pipeline errors.

The design of SwapCodes is simple. It introduces:

- No new buffers or per-thread state
- No changes to the ECC decoders

The performance advantages of SwapCodes come from eliminating the need to duplicate the register space, and from eliminating any explicit checking instructions. For example:

(1) Un-Duplicated Code

```
ADD R1,R1,R2
```

(3) Swap-ECC

```
ADD     R3,R1,R2 //orig.
ADD.ECC R3,R1,R2 //shad.
```

(2) Intra-Thread Duplication

```
    ADD R1,R1,R2 //orig.
    ADD R3,R3,R4 //shad.
    ISETP.EQ P1,R1,R3
@P1 BRA,U `(.L_1) //check
    BPT.TRAP 0x1
.L_1:
```

Fig. 3: An example of intra-thread duplication and Swap-ECC.