Argus: Low-Cost, Comprehensive Error Detection in Simple Cores

Albert Meixner, Michael E. Bauer, Daniel J. Sorin

PRESENTER: YUNKAI ZHAO

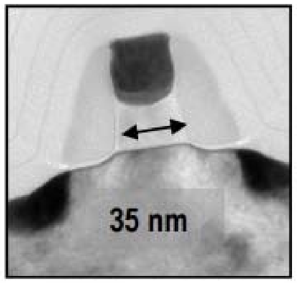
10/08/2015

Outline

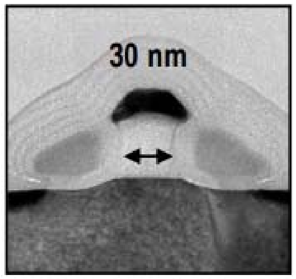
Introduction

❑ Problem

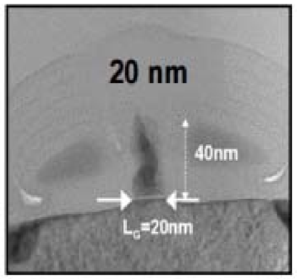
- More hardware errors (both transient and permanent) due to technology trends
- Simple cores remain popular and current self-checking methods are relatively expensive



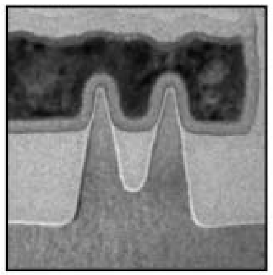
2006 - 65nm



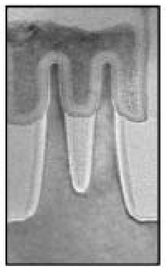
2008 - 45nm



2010 - 32nm



22nm - 2012



14nm - 2014

Argus Overview

Goal

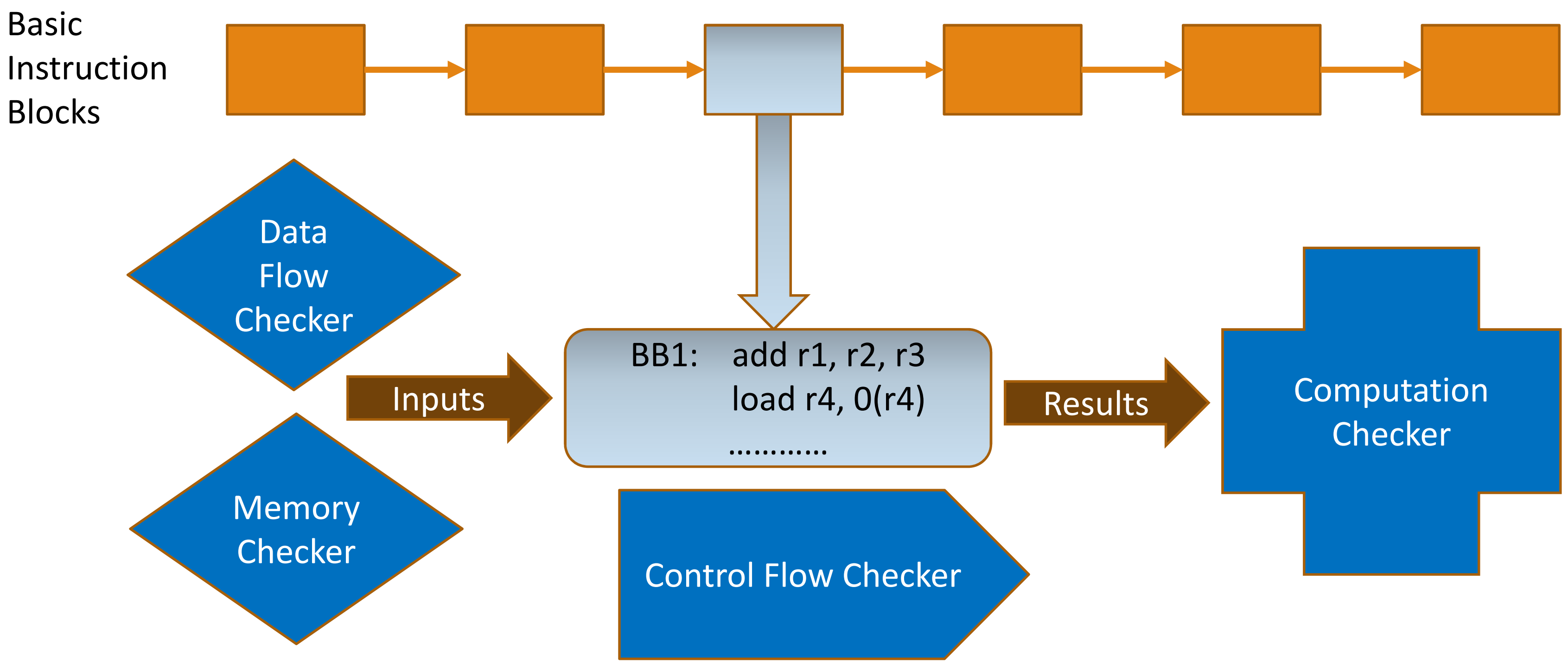
- Detect both transient and permanent errors in simple cores at low cost

Approach

- Divide and conquer four fundamental tasks (four invariants)
- Sufficient for detecting all possible single error
  that has no I/O, exceptions or interrupts

Control Flow | Data Flow | Computation | Memory

Divide and Conquer

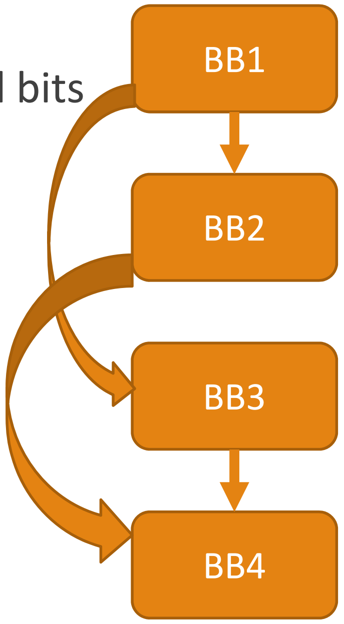


Control Flow and Dataflow Checkers

Dataflow and Control Signature (DCS)

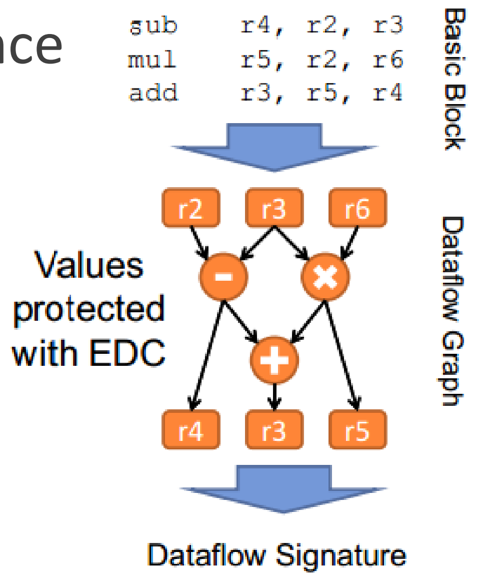
- Embeds signatures for legal successor blocks in unused instruction bits
- Use Signature instructions (NOPs) only when there is not sufficient unused bits
- State history signature (SHS) is maintained for flow tracking

```
BB1:    add r1, r2, r3
        sub r4, r1, r2
        Signature {BB2,BB3}
        beq BB3
BB2:    load r6, 0(r4)
        mul r7, r6, r6
        Signature {BB4}
        jmp BB4
BB3:    or r8, r6, r9
        Signature {BB4}
BB4:    and r10, r8, r6
        ...
```



Control Flow and Dataflow Checkers

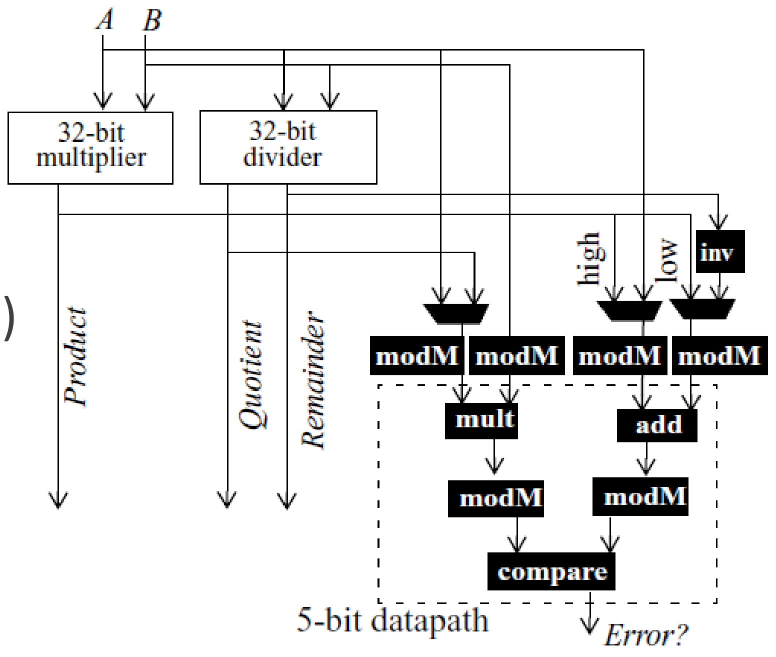
- ❑Compiler computes reference data flow signature for basic blocks
- ❑Data flow checker tracks and compares to reference
- ❑Also tracked by State History Signature (SHS)



courtesy to http://www.microarch.org/micro40/talks

Computation Checkers

- Multiple sub-checkers for different computation operations
- ALU Sub-Checker
  - Bitwise logical operation
  - Right-shift followed by a sign-extension (RSSE)
- Multiplier/Divider Sub-Checker
  - Modulo computation
  - (AB)mod M = ((A mod M)*(B mod M))mod M



Memory Checker

- Computation errors
  - Like the sub-checker in ALU
- Data corruption
  - Parity to each word in the data cache (not in instruction cache)
  - D_A = D XOR A; D’ = D_A’ XOR A
  - With a single-bit error, D’ ≠ D → parity different→ error
- Too expensive to detect “silently not performed” errors

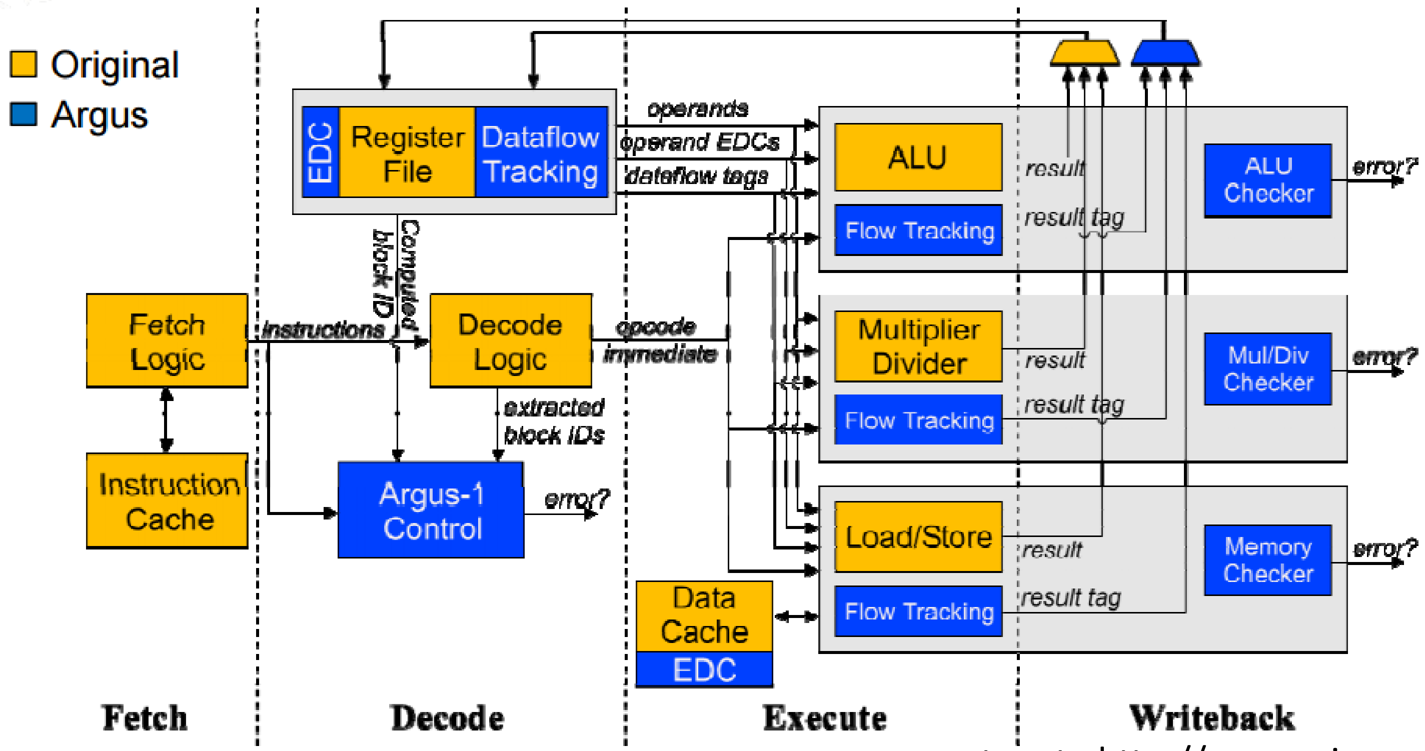
Evaluation and Analysis

- ❑Error Detection Coverage
- ❑Area Overhead
- ❑Performance Overhead

Argus-1

- Incorporated Argus error detection into the Open-RISC 1200 (OR1200) core
  - 4-stage, 32-bit scalar, in-order RISC, 32 general purpose registers
- Add Argus Components
  - Synthesized the Verilog, laid out the design
- Tradeoffs between cost and error coverage
  - Not perfect due to parity/signature aliasing, memory access errors and multiple-error scenarios

Argus-1 Implementation



courtesy to http://www.microarch.org/micro40/talks

Error Detection Coverage

Error Type	Unmasked, undetected	Unmasked detected	Masked, undetected	Masked, detected
transient	0.76%	37.4%	38.2%	23.7%
permanent	0.46%	37.6%	38.2%	23.7%

=> Detects the majority errors

Area Overhead

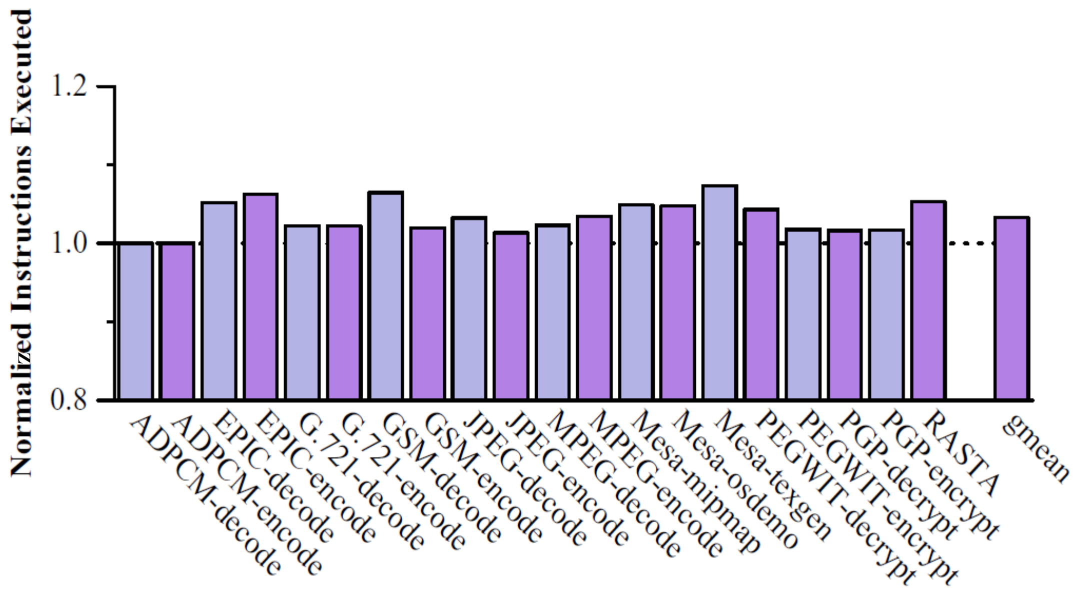
Unit in mm^2	OR1200 (Baseline)	With Argus-1	Overhead
core	6.58	7.67	16.6%
I-cache: 1-way	2.14	2.14	0%
2-way	2.42	2.42	
D-cache: 1-way	2.14	2.24	4.9%
2-way	2.42	2.54	5.1%
Total: 1-way	10.86	12.05	10.9%
2-way	11.42	12.63	10.6%

=> Low area overhead

=> Low power overhead

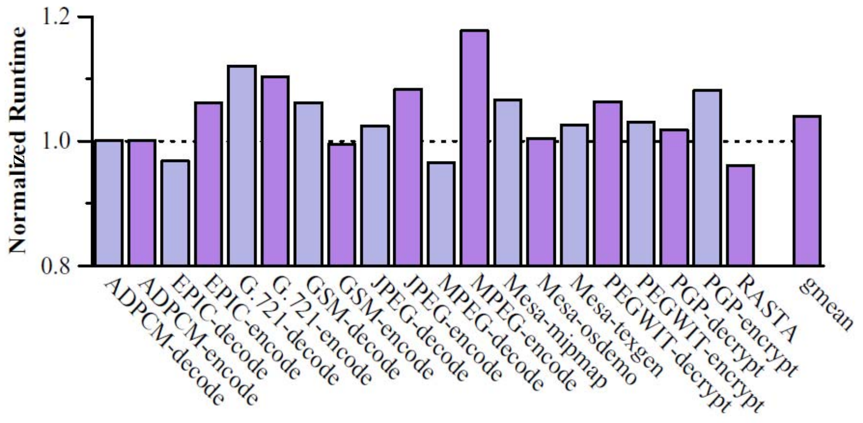
Performance Overhead

- No change in clock cycle since do not increase any critical path
- Increased instructions due to signature embedded (overhead is 3.5% on average)

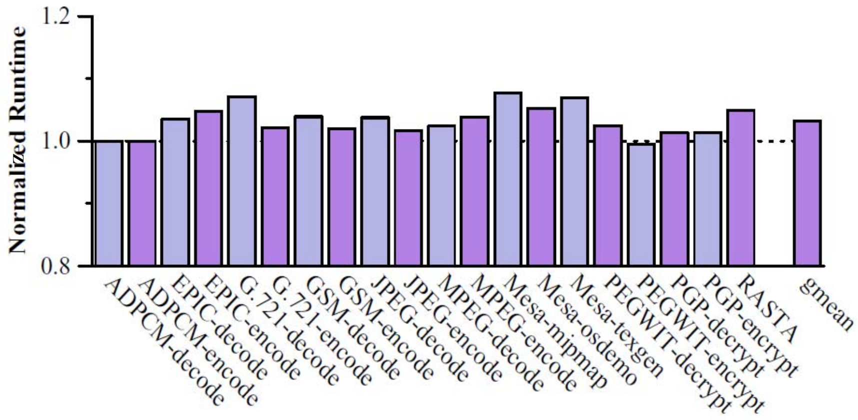


Performance Overhead

- Normalized performance impact for a direct-mapped and 2-way instruction cache
- On average, the runtime overheads are 3.9% and 3.2% respectively



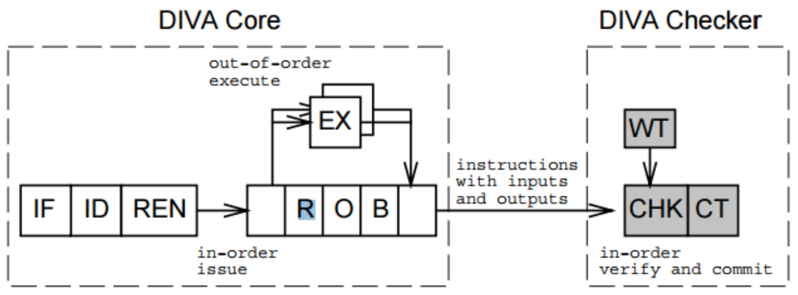
Runtime Overhead (1-way I-cache)



Runtime Overhead (2-way I-cache)

Related Work

- Dual Modular Redundancy (DMR)
- Redundant Multithreading (RMT)
  - S. K. Reinhardt and S. S. Mukherjee. Transient Fault Detection via Simultaneous Multithreading. In Proc. Of the 27th Annual Int'l Symp. on Computer Architecture, p. 25–36, June 2000.
- Dynamic Implementation Verification Architecture (DIVA)
  - T. M. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In Proc. of the 32nd Int'l Symp. on Microarchitecture, Nov. 1999.



Conclusion

- Efficient self-checking core solution
  - Check four invariants to detect transient and permanent errors in simple core
  - Detect majority errors
- Low area/performance cost
  - < 17% average chip area overhead
  - < 4% average performance overhead
- Incomplete solution
  - Exception and interrupt
  - Memory checker for CMP

Questions?

Discussion

- ❑Is Argus method useful in industry?
- ❑Can this solution be extended to many-core processors?

Thank you!