DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design

Or, how I learned to stop worrying and love complexity.

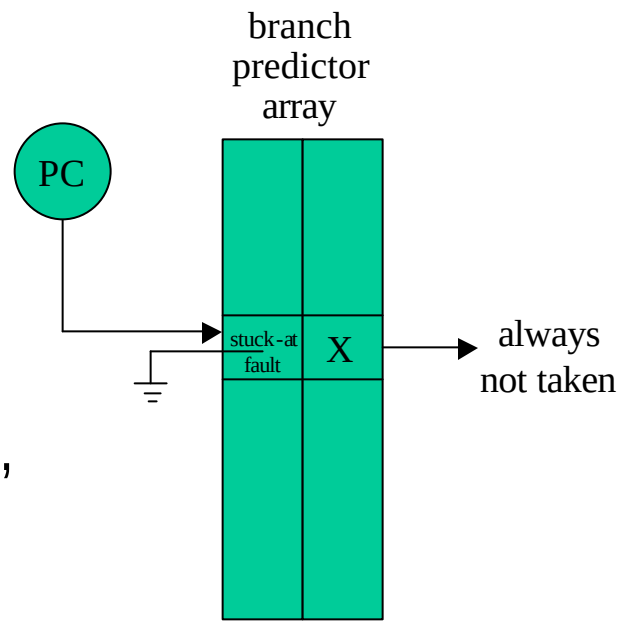
Todd Austin
Advanced Computer Architecture Lab
University of Michigan
http://www.eecs.umich.edu/~taustin

Microprocessor Verification

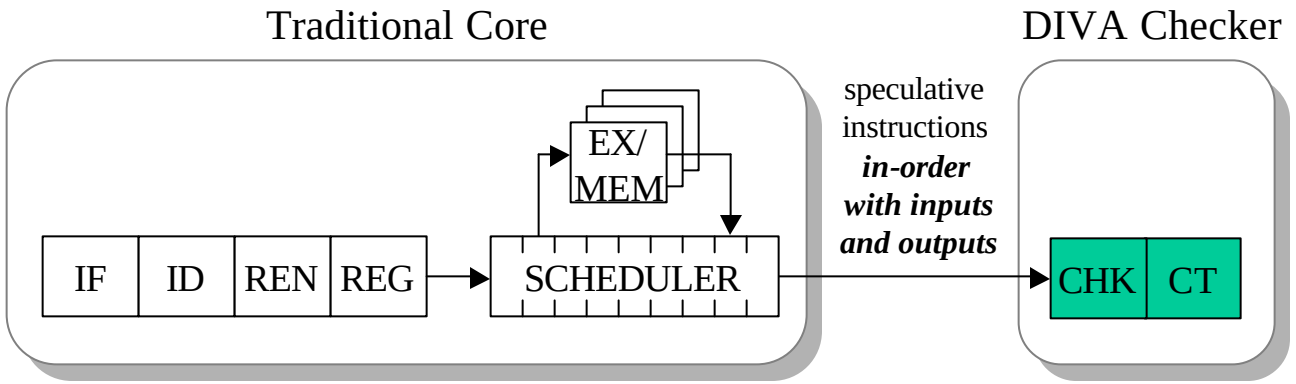
- Task of determining if a design is correct
  - " *starting states ($state_i$, $inputs_i$), next state ($state_{i+1}$) is correct*
  - Implemented with functional and electrical verification
- Huge burden on design teams
  - Immense test space
  - Done with respect to ill-defined reference, *what is correct?*
  - Expensive and time-consuming process, typically 1-2 years after tape-out
  - High-risk, only one chance to "get it right" or else…
- New reliability challenges in deep submicron silicon
  - Increased complexity
  - Degraded signal quality
  - Increased exposure to single event radiation (SER) upsets

Motivating Observations

- Speculative execution is fault tolerant
  - Design errors and electrical faults indistinguishable from bad predictions
  - Predictor faults only manifest as performance divots
  - Correct checking mechanism will fix any incorrect speculative operation
- What if all computation, communication, and control were speculative?
  - *Any* fault outside the checking mechanism would be detected and corrected



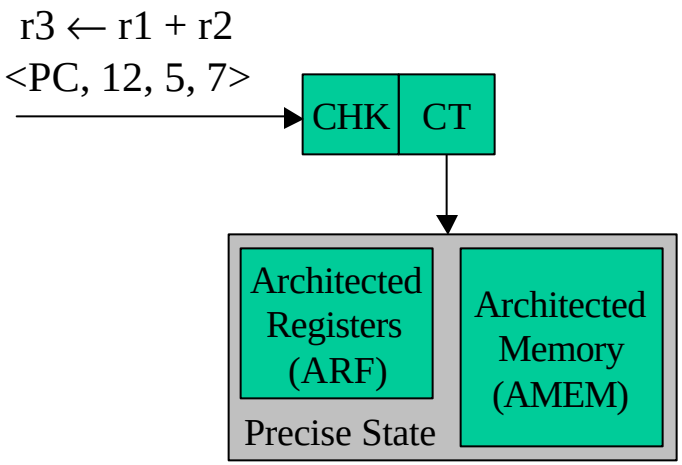
Dynamic Verification: Seatbelts for Your CPU



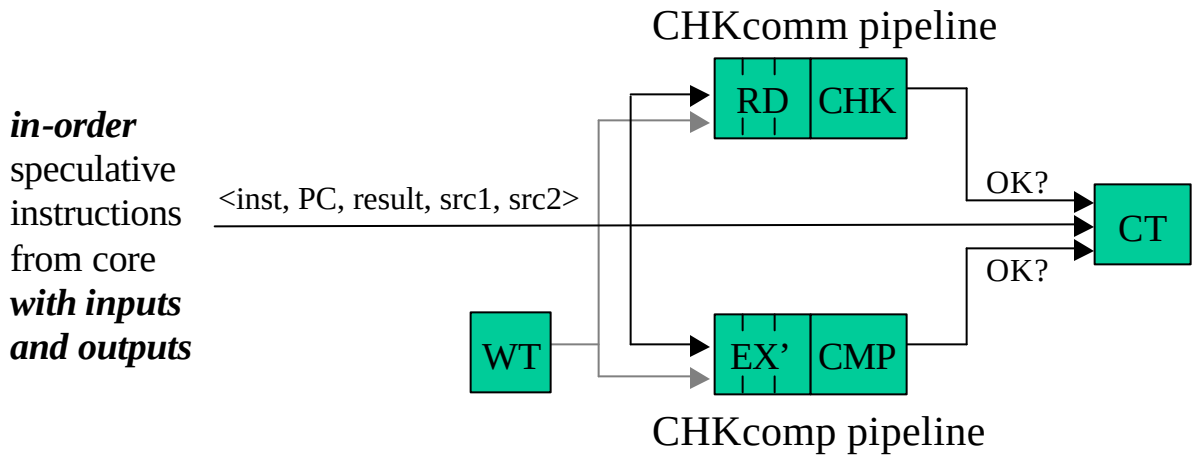
- Dynamic implementation verification architecture (DIVA)
  - Instructions verified by checker before retirements
  - Checker *detects and corrects* any faulty result, restarts core
  - Existing speculation infrastructure protects architected state
- *Lifts the burden of correctness from core processor*
  - All core computation, communication, control is speculative
  - Tolerates design errors, electrical faults, silicon defects, and failures
  - Core has burden of high accuracy

Architectural-Level Instruction Checking

- Checker enforces serial semantics
  - Correct control
    - $PC_i == NPC_{i-1}$
  - Correct computation
    - $5 + 7 == 12$
  - Correct communication (inputs)
    - $ARF[r1] == 5, ARF[r2] == 7$
  - Forward progress
    - *Use timeout mechanism*



DIVA Checker Architecture



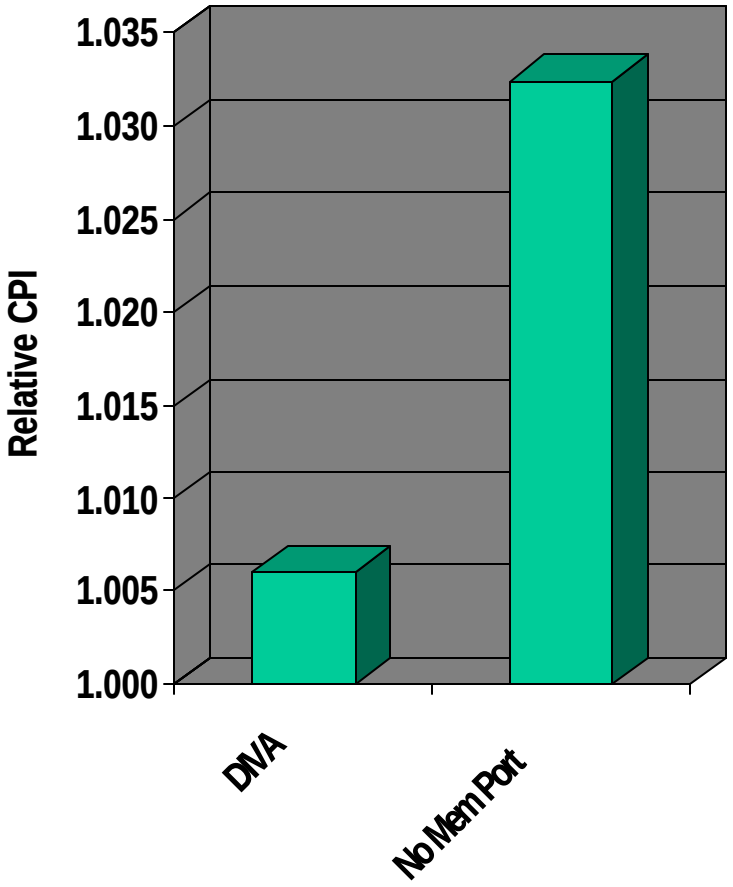
- *CHKcomm* pipe verifies reg/mem inputs (with in-order accesses)
- *CHKcomp* pipe verifies results (with simple, robust algorithm)
- Watchdog timer detects deadlocks, livelocks, and lockups
- *Availability of correct results ensures forward progress*
- **Key design issue**: checker must be simple, reliable and fast

Verifying the Checker

- Simplicity should ensure high-quality functional verification
  - In-order blocking pipelines (trivial scheduler, no rename/reorder)
  - Design lends itself to formal verification (in-order, precise state)
- Latency insensitive design should ensure robust implementation
  - Deeply pipeline the design for large timing margins, high noise immunity
- Effort to better quantify verification costs is underway
  - Including other costs such as area and power

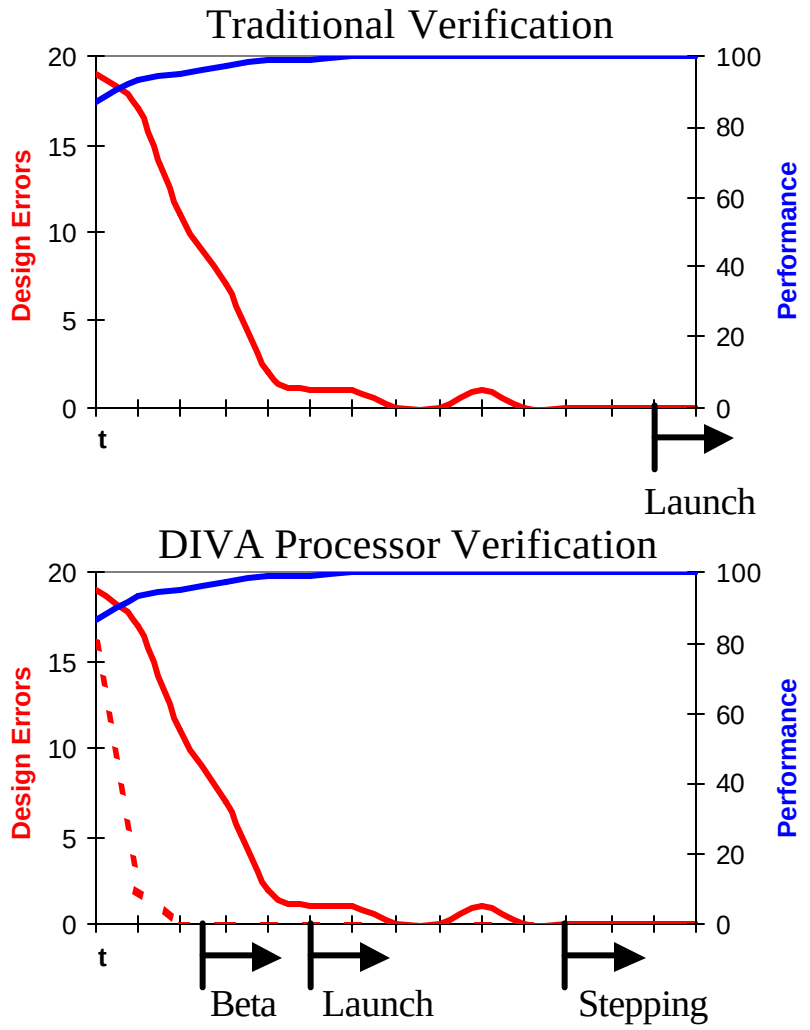
Minimal Impact on Core Performance

- Checker *throughput* bounds IPC
  - No control hazards (simply check PC)
  - Computation embarrassingly parallel
  - Few cache stalls (core warms cache)
  - Plus, retirement B/W typically low
- Core performance fairly insensitive to checker design parameters

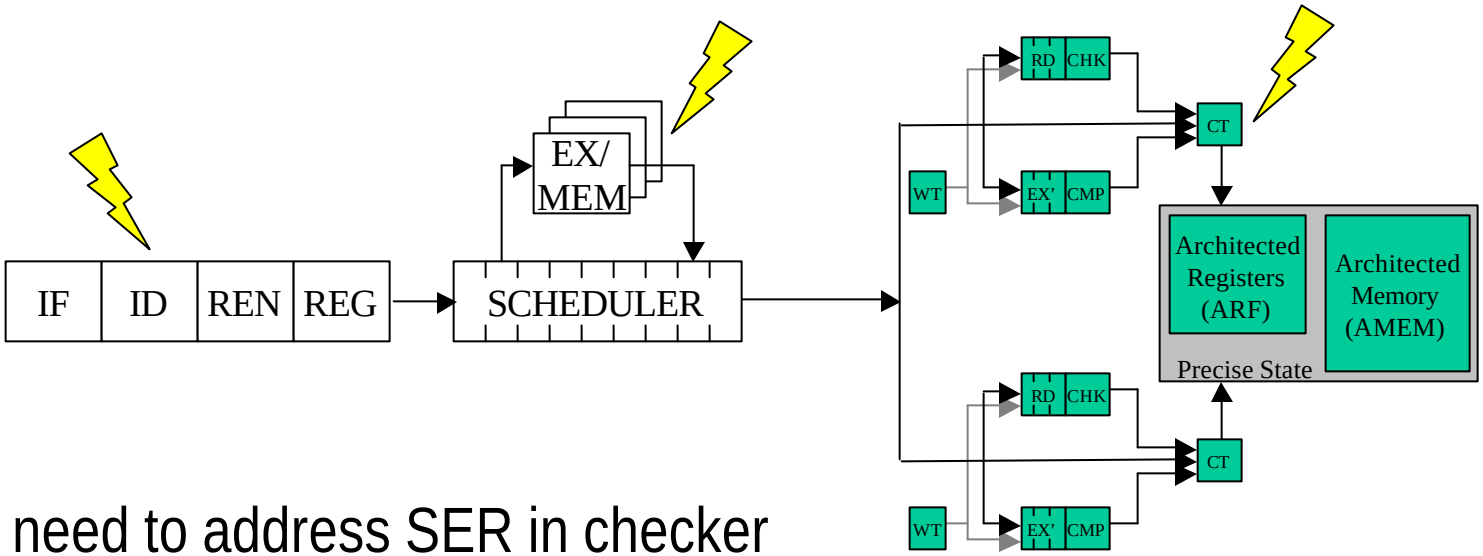


Future Work: Beta-Release Processors

- Traditional verification stalls launch until debug complete
- DIVA processor verification could overlap with launch
  - Beta-release when checker works
  - Launch when performance stable
  - Step for good karma

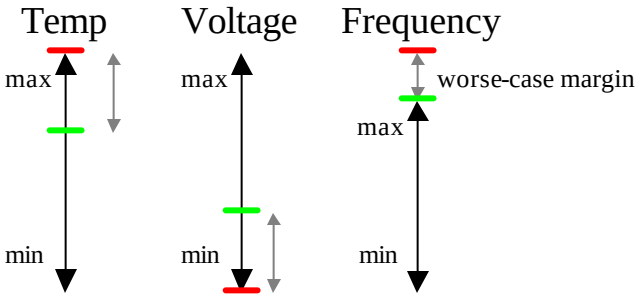


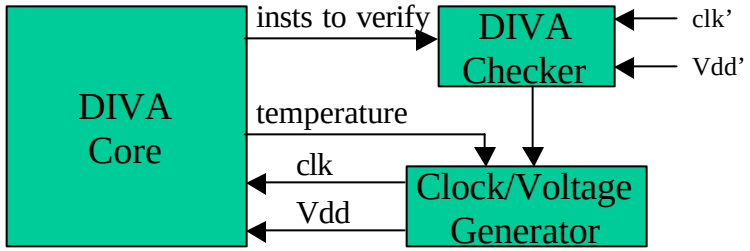
Future Work: Scalable SER Protection



- Only need to address SER in checker
  - Sparse strikes manifest as functional errors
  - Rad-hard checker detects and corrects faults
  - Core designed without regard to particle strikes (e.g., no ECC…)
- Rad-hard checker designs
  - Small checker will provide natural resistance to SER (small target!)
  - Or, replicate the checker logic, restart pipes on disagreement

Future Work: Self-Tuned Systems





- Traditional logic implementations way too conservative for DIVA
  - Unnecessary design margins consume power and performance
  - System may not be operating at slow corner
- DIVA checker enables a self-tuned clock/voltage strategy
  - Push clock, drop voltage until desired power-performance characteristics
  - If system fails, reliable checker will correct error, notify control system
  - Reclaims design margins plus any temperature and voltage margins

Conclusions

- Design quality and reliability are uncompromising tasks
  - Verification costs and risks are very high and growing
- Speculative execution can reduce the burden of verification
  - Dynamic verification makes core processor fully speculative
  - Core tolerates design errors, electrical faults, silicon defects, and failures
  - Architectural-level checking keeps checker design simple
  - Core processor eliminates hazards that could slow checker pipeline
- Pushing speculation to the limit may yield more benefits
  - Beta-release processors could overlap verification with launch
  - Rad-hard checker provides single event radiation (SER) protection
  - Fault-tolerant core can leverage aggressive circuits (self-tuned systems)