Time Redundancy in Trace Processors

Eric Rotenberg

Computer Sciences Department
University of Wisconsin — Madison

Outline

- Problem statement: time redundancy spectrum
- AR-SMT high level
- Applying SMT concepts to AR-SMT
- AR-SMT detail
- Trace processors as a platform for AR-SMT
- Results
- Conclusions

Time Redundancy

- Pure time redundancy
  - Run program twice (or more times) back to back
  - Covers transient faults cheaply

  Pros/Cons

  + No hardware redundancy
  + Good transient fault coverage
  + Good hardware coverage
  - Slow: 2x latency
  - Error latency

Time Redundancy

- Instruction re-execution

[Sohi/Franklin/Saluja], [753: Abid/Faisal]

  - Introduced to improve performance of time redundancy
  - Leverages highly parallel uniprocessors
  - "Duplicate" instruction at issue buffer or functional unit

Pros/Cons

+ High performance -- fine control over idle cycles and idle resources
+ Small error latency (recovery simple)
- Poor hardware coverage
- Poor transient fault coverage: **long latency transients**

Time Redundancy: Granularity

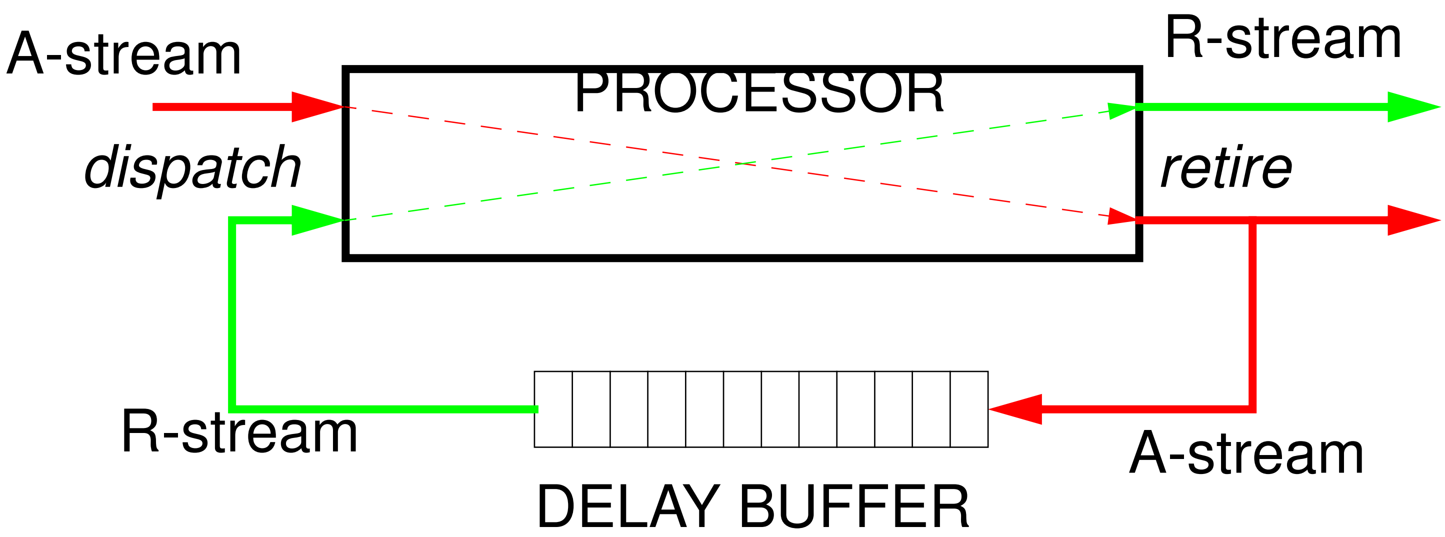
- Observation: models differ in granularity of interleaving two streams

program granularity — AR-SMT — instruction granularity

good coverage
poor performance

poor coverage
good performance

AR-SMT High Level



- “A” => “**A**ctive stream”
- “R” => “**R**edundant stream”
- “SMT” => “**S**imultaneous **M**ulti**T**hreading”

SMT

- Simultaneous multithreading

  [Tullsen, Eggers, Emer, Levy, Lo, Stamm]

  - Multiple contexts space-share a wide-issue superscalar processor
  - Key ideas
    - Performance: better overall utilization of highly parallel uniprocessor
    - Complexity: leverage register renaming to make multiple contexts transparent to dynamic scheduler
  - Prophecy: SMT will happen
  - SMT is a good match for implementing AR-SMT

AR-SMT Detail: dispatch/retirement

- Critical resources are time-shared (multiplexed)
  - Frontend: fetch/dispatch pipeline
  - Retirement (committing register/memory state)
  - Simple scheduling heuristic
    - Dispatch R-stream only when Delay Buffer is full
    - Retire A-stream only if Delay Buffer not full
    - i.e. less choice than in general SMT because A-R streams are coupled

AR-SMT Detail: registers

- Handling register values
  - Register renaming
    - Rename mechanism separates the streams -- **issue mechanism is unaware of multiple streams**
    - Streams must space-share the register rename maps
  - 32 more physical registers (GPRs) to support R-stream architectural register state

AR-SMT Detail: memory

- Handling memory values
  - Context ids (single bit) in load/store buffers to separate memory dependences
  - New problem over conventional SMT:
    - R-stream context is created by *hardware*
    - But A-/R-stream need separate memory spaces -- normally done by the O/S!

AR-SMT Detail: memory

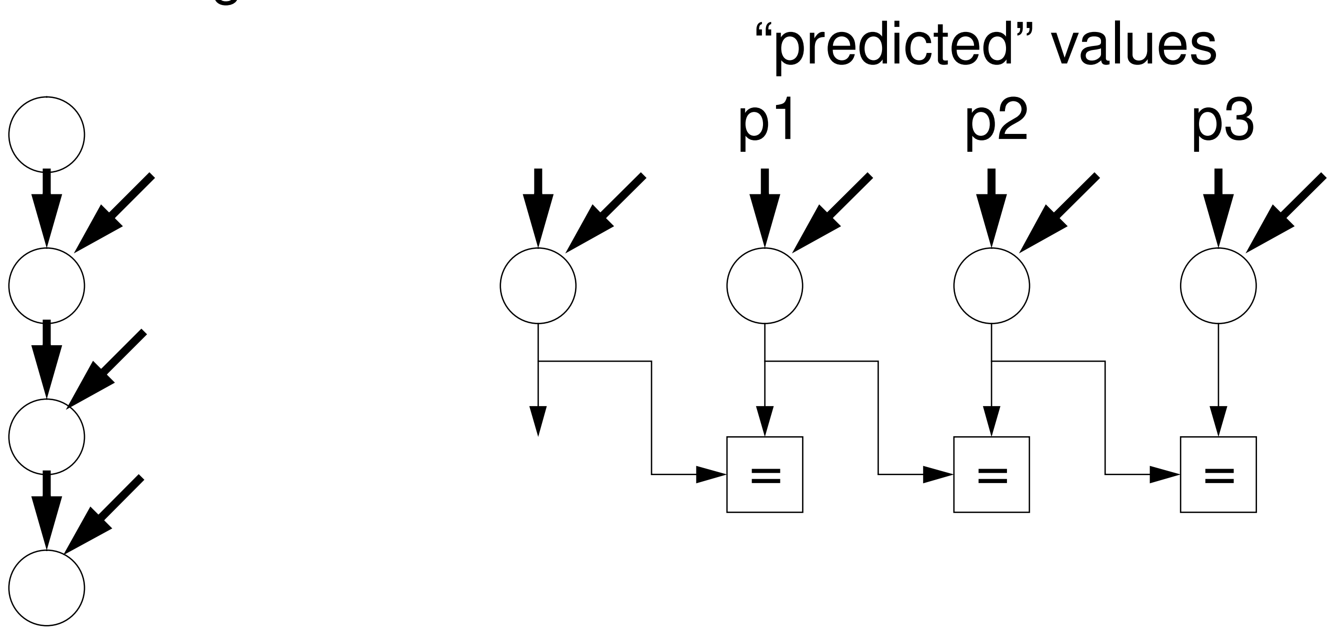
- O/S support
  - Allocate pages in contiguous pairs
  - One of the pages is for the R-stream
  - Single address space with redundancy
- H/W translation support for R-stream
  - Pass PTE information in the Delay Buffer
  - Translation is simple: page + 1
  - Delay Buffer PTE information must remain consistent

AR-SMT Detail: Delay Buffer

- Delay Buffer contents
  - At a minimum: control flow information
    - PCs in a conventional processor
    - Trace IDs in a trace processor (later...)
  - May also include data flow information
    - Register values (source, destination, or both)
    - Memory addresses
  - PTEs for translation

AR-SMT Detail: Delay Buffer

- Benefits of storing values/addresses in Delay Buffer
  - Error latency: can detect faults and recover while running (like instruction re-execution)
  - Performance: perfect value “prediction” and memory disambiguation!



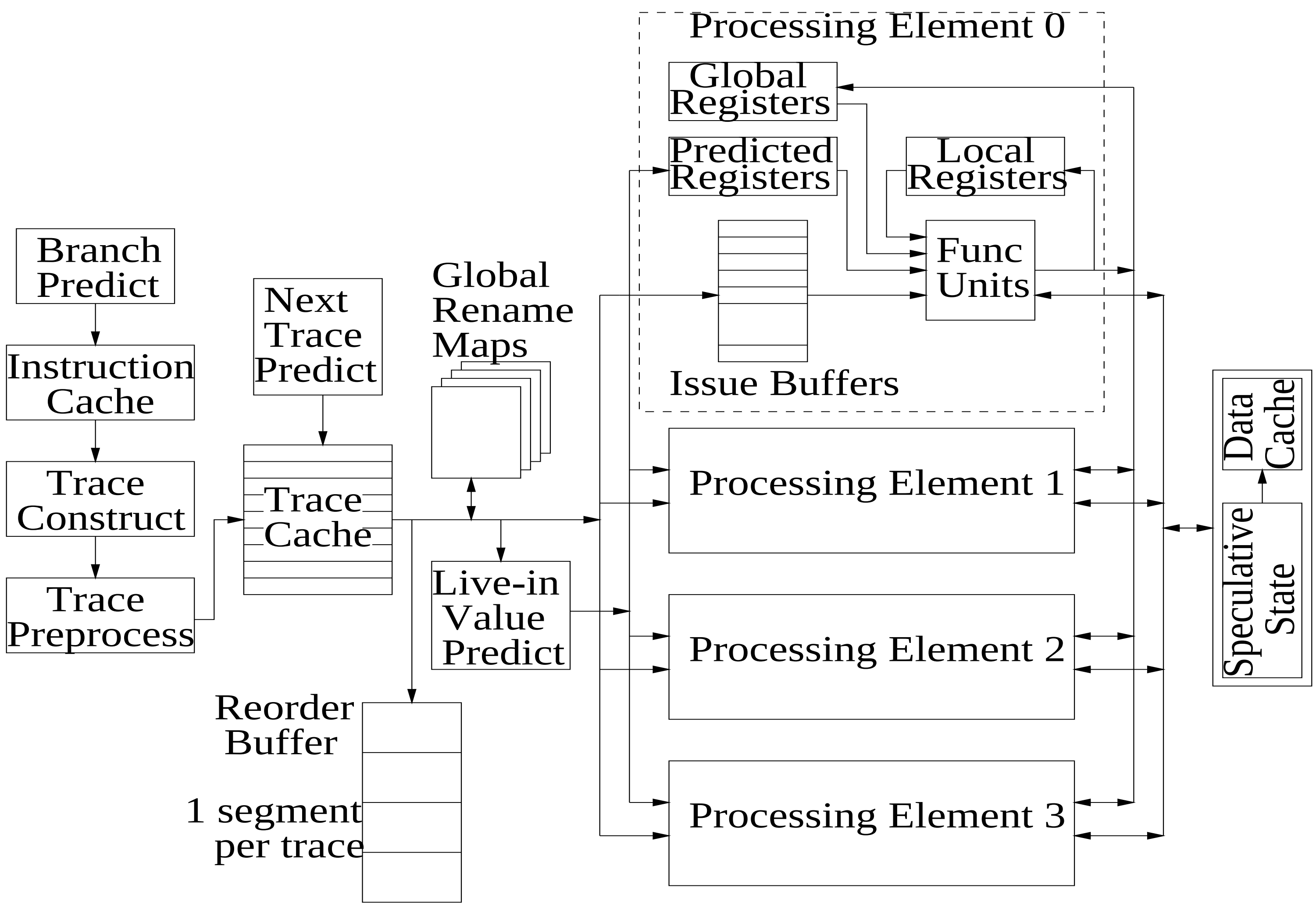
AR-SMT Detail: recovery

- Recovery from faults
  - Values in Delay Buffer
    - Faults can be detected with a relatively small delay
    - R-stream committed state serves as “checkpoint” which we can restart from
  - No values in Delay Buffer
    - Compare two program outputs
    - Re-run program if they differ

Grading AR-SMT

- Good coverage
  - Delay adds coverage of longer latency transients
  - Covers caches and most of the processor
- Good performance
  1. SMT utilization
  2. Perfect control flow and data flow knowledge
- Good error latency and fast recovery
- Reasonable cost
  - Registers for R-stream, added complexity for SMT
  - Delay Buffer (dense, specialized shift-register cells)

Trace Processor



Conclusions