Slipstream Processor: Improving both Performance and Fault Tolerance

Karthik Sundaramoorthy Zach Purser Eric Rotenberg

North Carolina State University

Department of Electrical and Computer Engineering

Engineering Graduate Research Center, Campus Box 7914, Raleigh, NC 27695

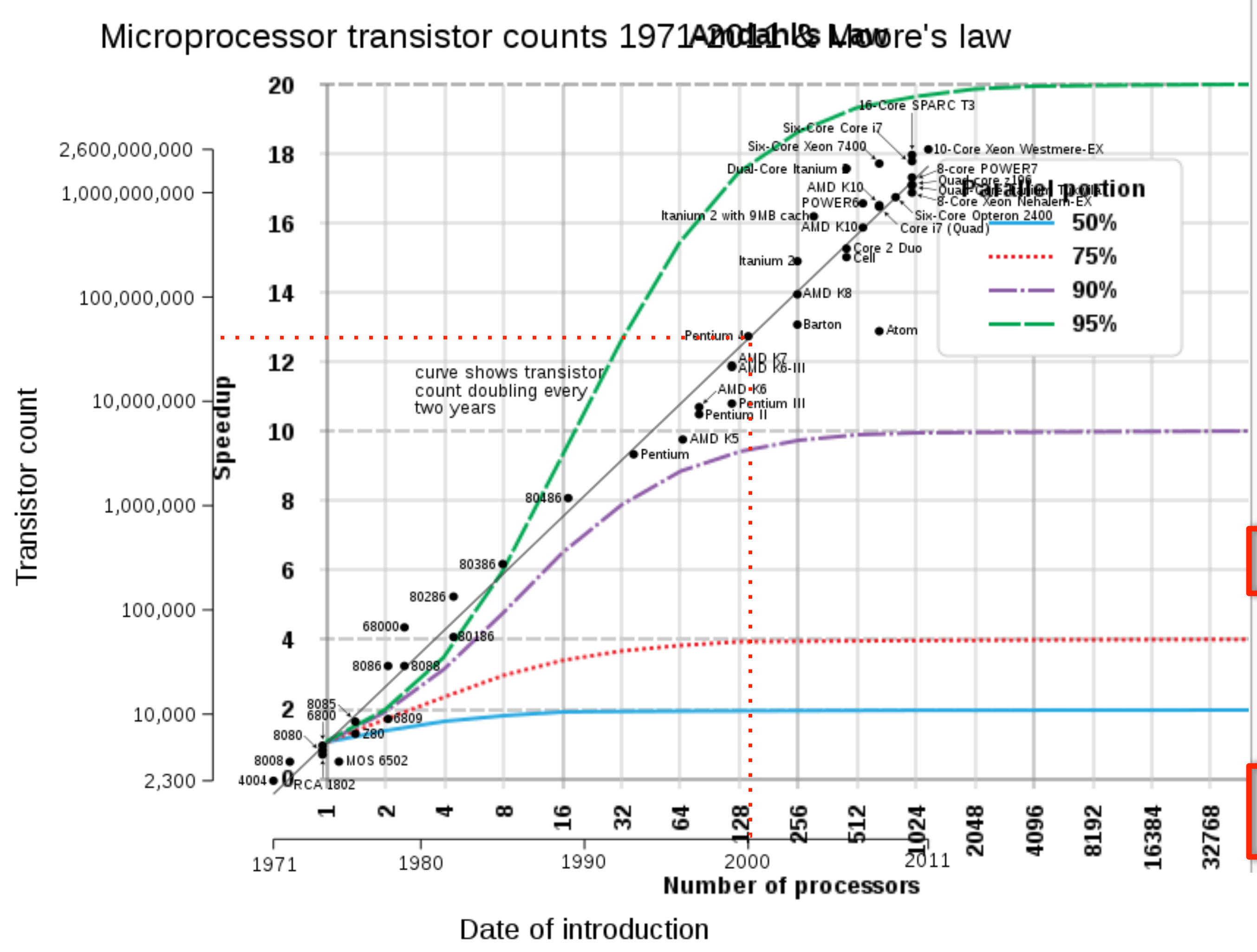
{ksundar,zrpurser,ericro}@ece.ncsu.edu, www.tinker.ncsu.edu/ericro/slipstream

ASPLOS 2000

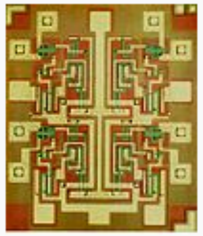
Presented by Michael Keller

BACKGROUND, PROBLEM & GOAL

Background



Semiconductor manufacturing processes



10 µm	– 1971
6 µm	– 1974
3 µm	– 1977
1.5 µm	– 1982
1 µm	– 1985
800 nm	– 1989
600 nm	– 1994
350 nm	– 1995
250 nm	– 1997
180 nm	– 1999
130 nm	– 2001
90 nm	– 2004
65 nm	– 2006
45 nm	– 2008
32 nm	– 2010
22 nm	– 2012
14 nm	– 2014
10 nm	– 2017
7 nm	– 2018
5 nm	– *~2020*

Daniels220 at English Wikipedia [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)]
Wgsimon [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], from Wikimedia Commons
https://en.wikipedia.org/wiki/Template:Semiconductor_manufacturing_processes

Problem

- Amdal’s law shows limitations for parallelizability
- Physical limitations for transistor count scalability
- Transient Hardware errors / Soft Errors / Bit flips due to smaller Hardware
- No error detection
    - This can slow down program execution, due to re-computation
- Limitations of branch prediction with compilers
- Today the problem is even worse than in 2000

Goal

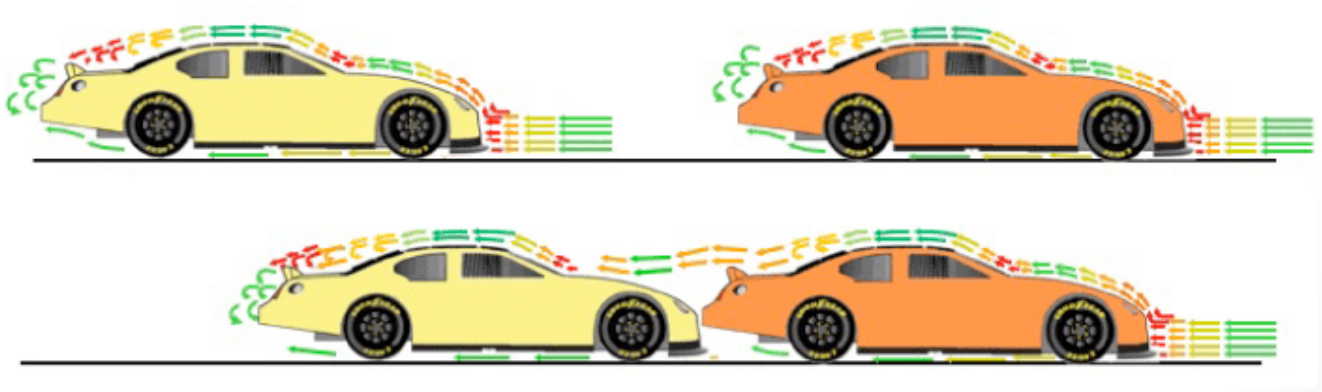
- Improve performance
  - Use multiple hardware context to speed up single thread execution by prefetching a perfect program prediction
- Improve fault tolerance
  - Comparison of different hardware context to detect and adjust incorrect executions

KEY IDEA

Slipstream - NASCAR

- At speed in excess of 190 m.p.h, high air pressure forms at the front and a partial vacuum at the rear of the car
- Second car can position itself behind a leading car
- Leading car has less drag, since vacuum is filled up by a car
- Car in the back has less air resistance

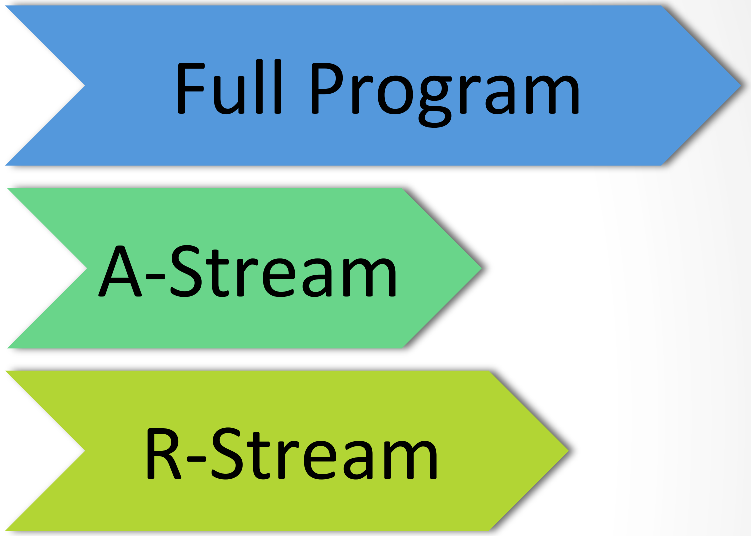
- Both cars together drive faster than either can alone



http://www.writeopinions.com/drafting-racing

Key Idea

Deploy program into two threads:



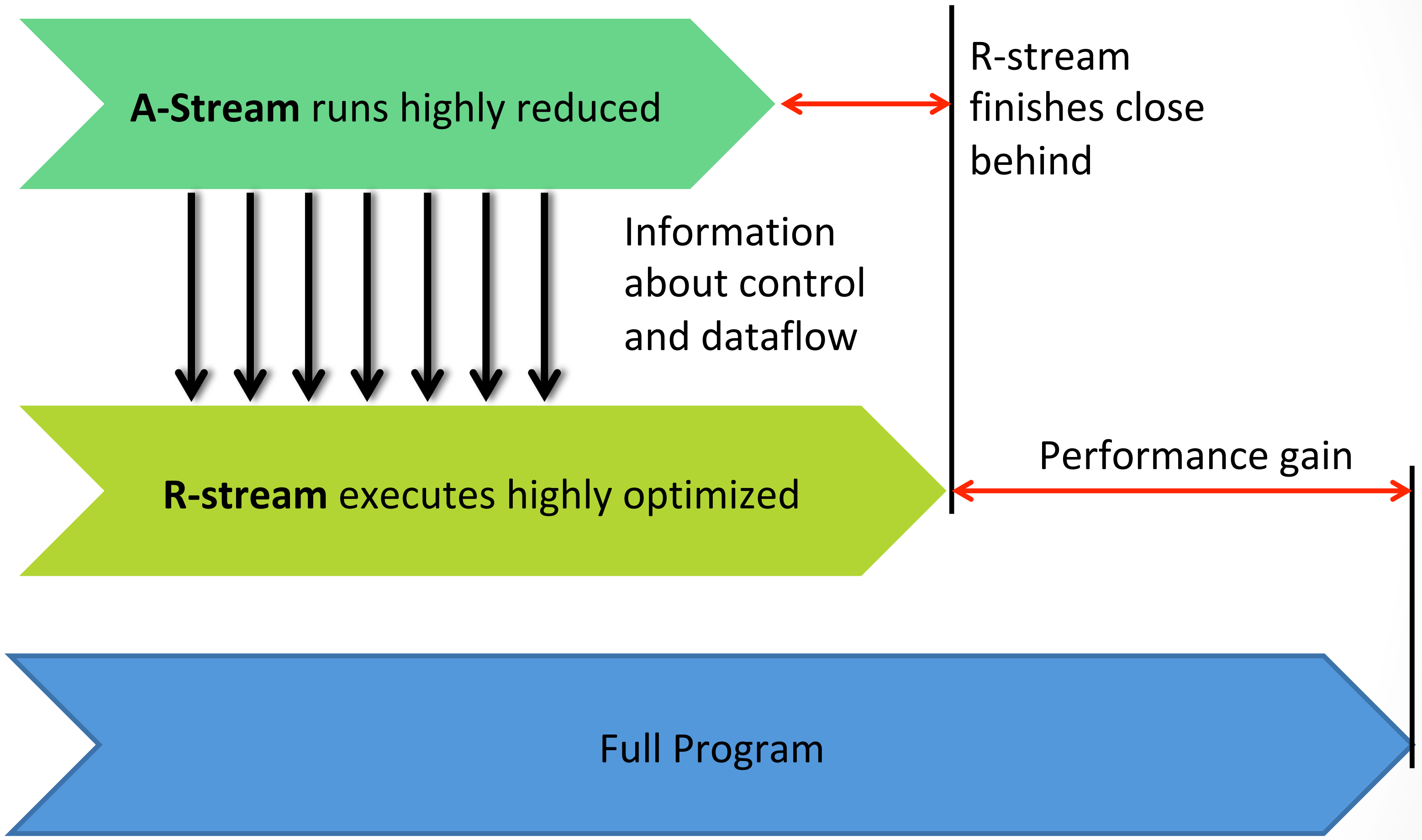
A-Stream

- Advanced thread executes a highly reduced instruction stream

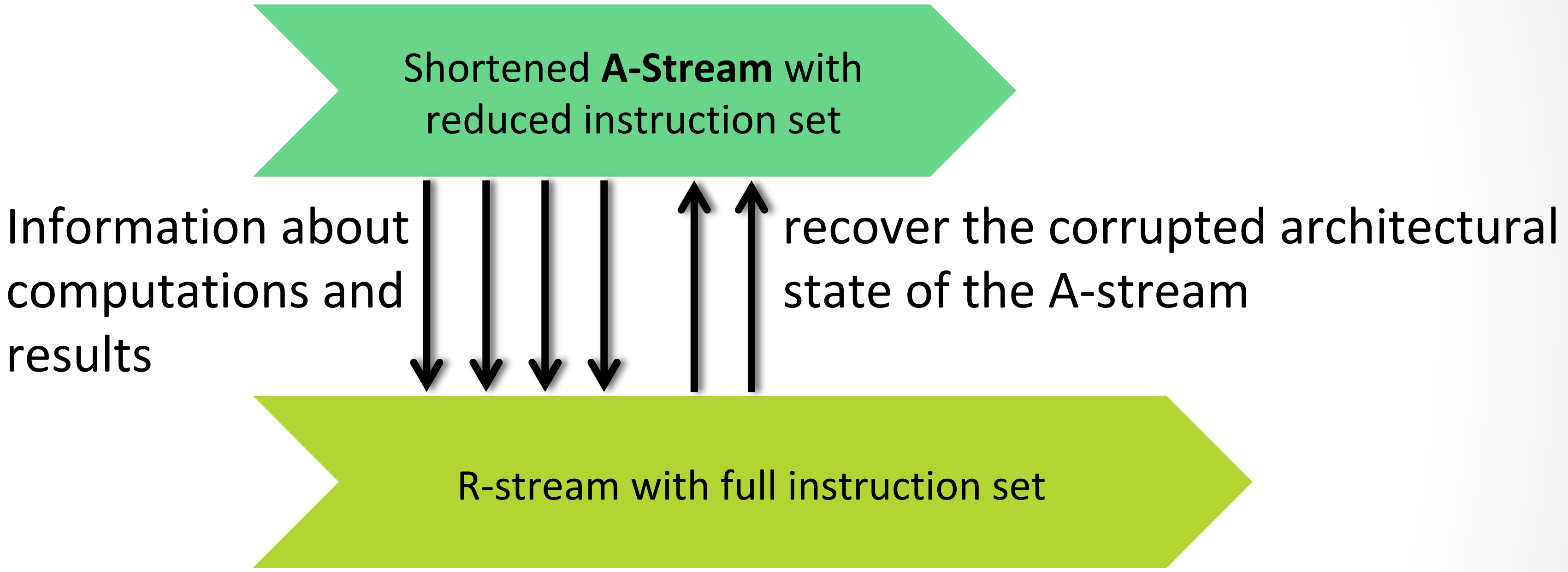
R-Stream

- Redundant thread uses results, prefetches predictions generated by advanced thread and ensures correctness

Improve Performance


A-Stream runs highly reduced
R-stream finishes close behind
Information about control and dataflow
R-stream executes highly optimized
Performance gain
Full Program

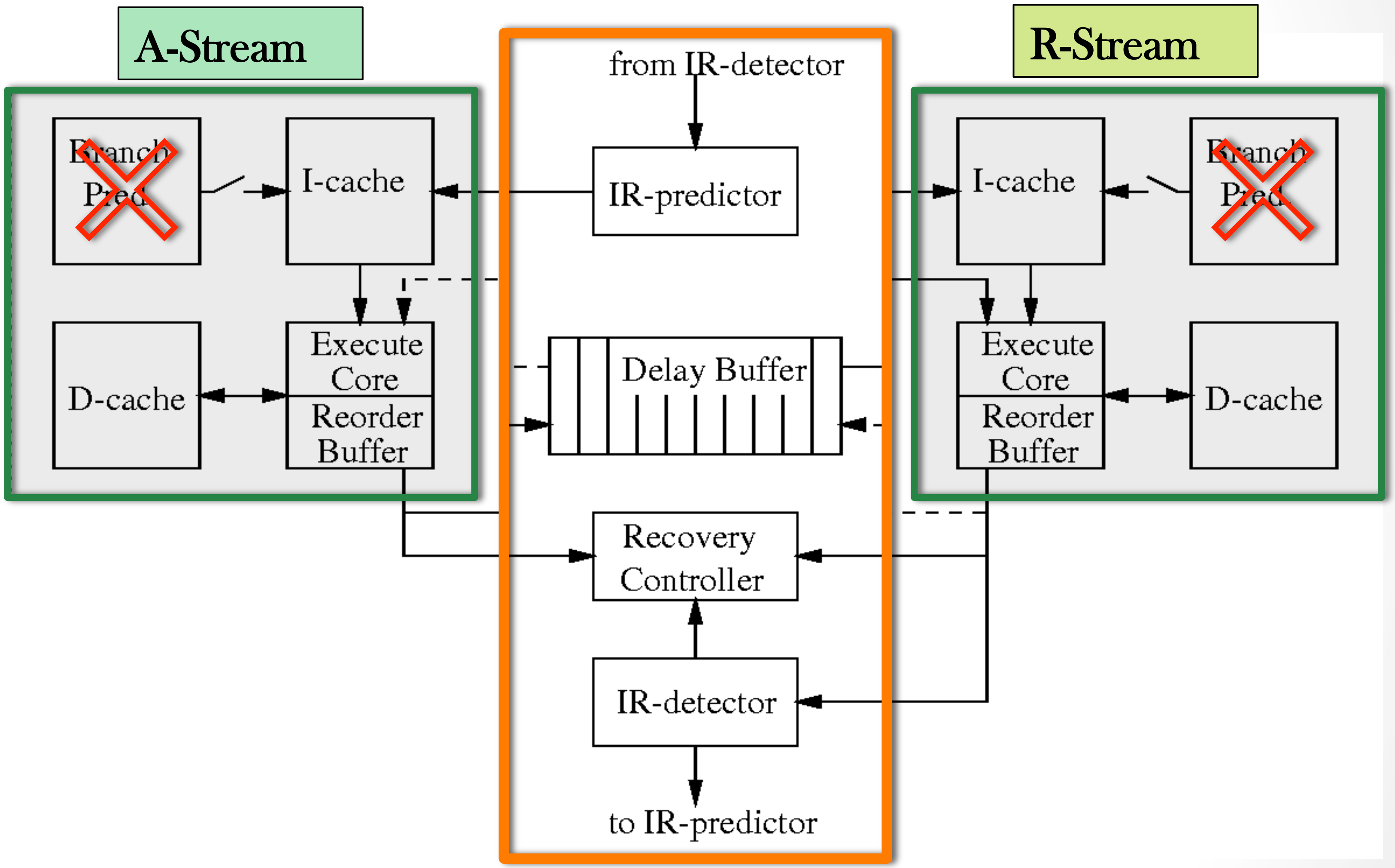
Improve Fault Tolerance


Shortened **A-Stream** with reduced instruction set
Information about computations and results
recover the corrupted architectural state of the A-stream
R-stream with full instruction set

If R-Stream detects a mismatch between results we can recover

MECHANISMS

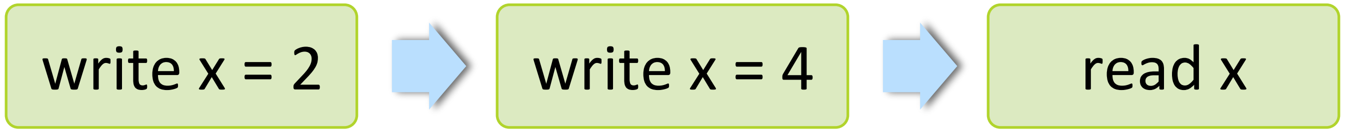
Hardware


A-Stream
R-Stream
from IR-detector
Branch Pred
I-cache
IR-predictor
I-cache
Branch Pred
Execute Core
Reorder Buffer
D-cache
Delay Buffer
Execute Core
Reorder Buffer
D-cache
Recovery Controller
IR-detector
to IR-predictor

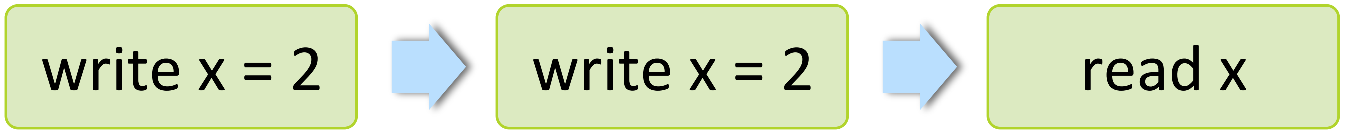
Removable Instructions

Distinguish three categories of ineffectual computation

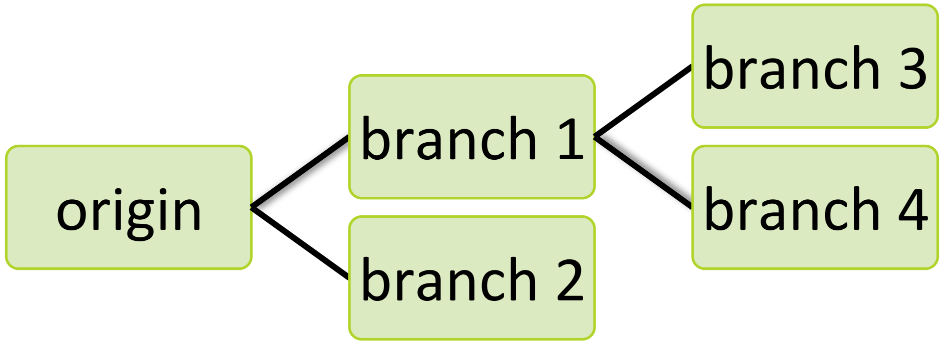
1. Unreferenced writes are values overwritten before use



2. Writes that do not modify the state of location

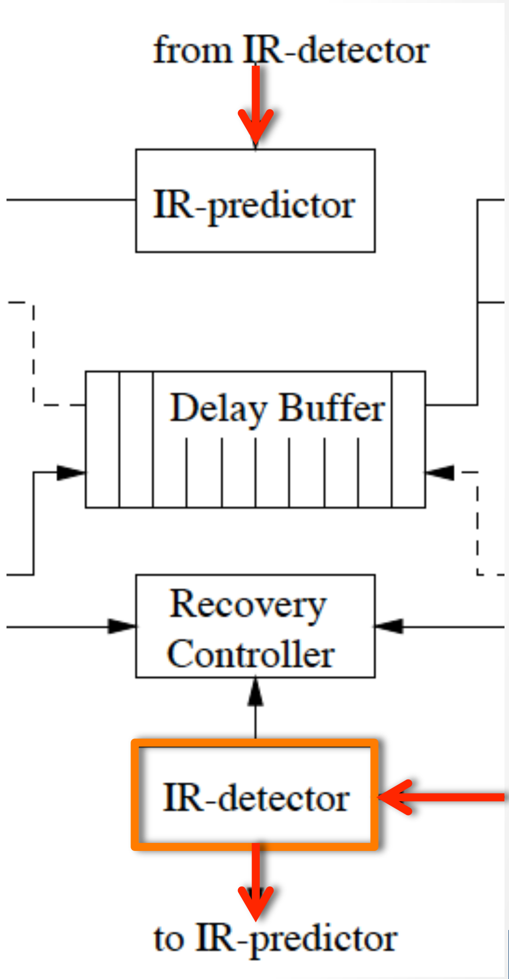


3. Dynamic branches whose outcomes are consistently predicted correctly.



Instruction Removal Detector

- Receives retired instructions from R-stream
- Detects and removes removable instructions
- Generates
  - trace id [Start PC, {branch outcomes}]
  - intermediate PC (Program counter)
  - ir-vec (instruction removal vector)
- Send information to IR-predictor

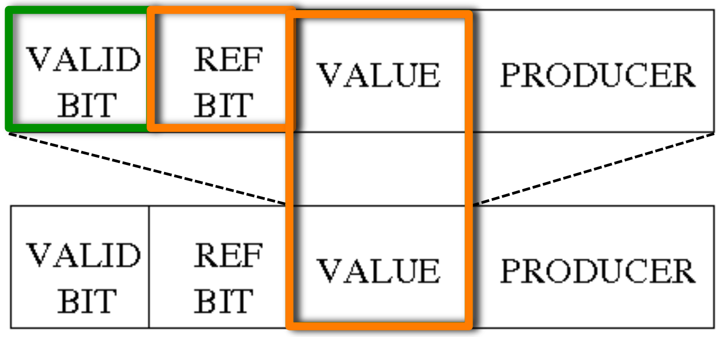


K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

Detect Removable Instructions

- Get the most recent producer of the value
- If new instruction is write
  - VALID BIT of producer is set and the new value is equal to its value we have a non modifying write
  - VALID BIT of producer is set and the new value does not match, we check the REF BIT, if it is not set we have an unreferenced write
- In both cases we select the instruction for removal by adding it to the ir-vec

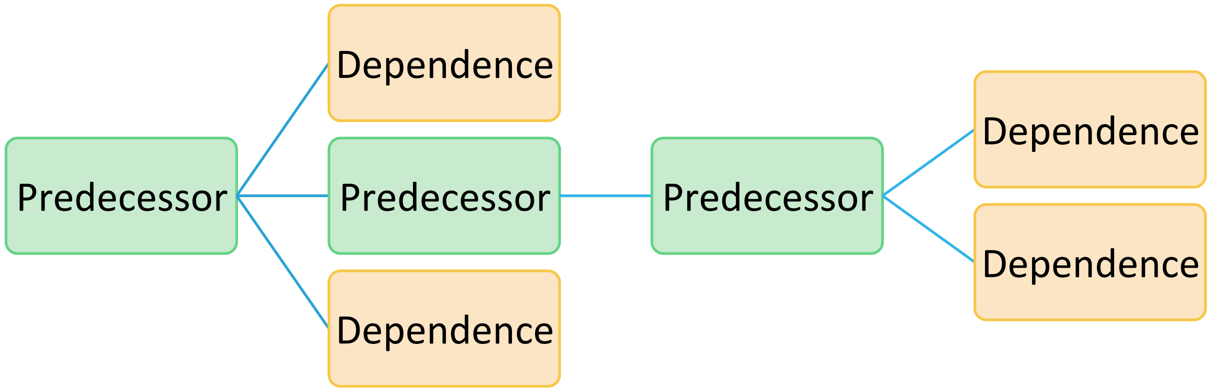
Producer instruction



New instruction

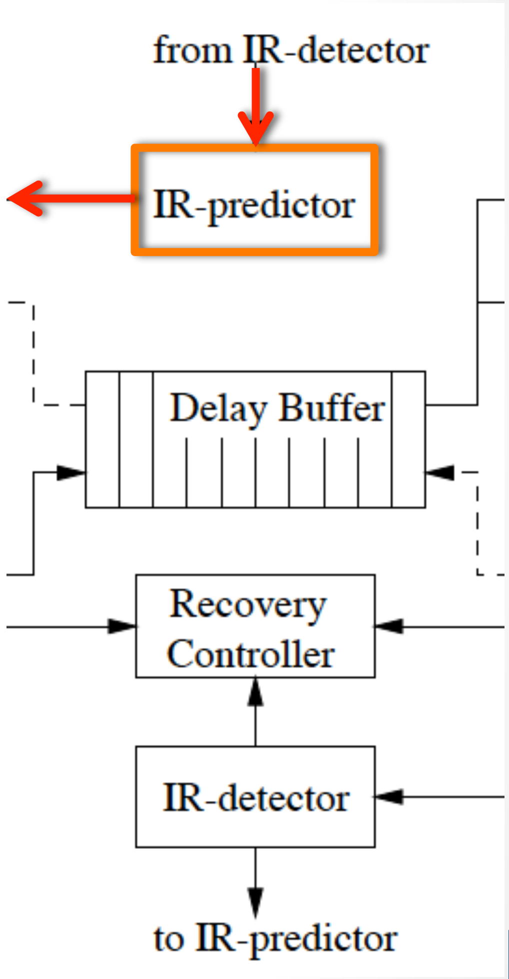
Detect Removable Branches

- Backpropagates selection status to predecessor instruction
- Predecessor is selected for removal if all dependent instructions are selected



Instruction Removal Predictor

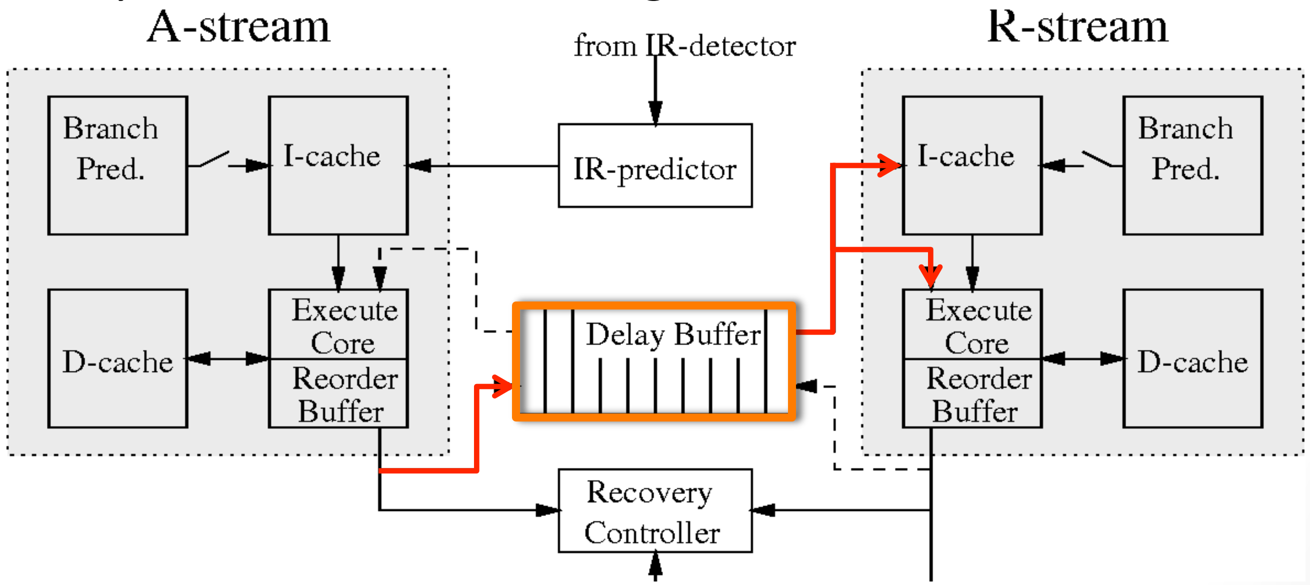
- Modified conventional trace predictor
  - Trace consist of 32 dynamic instructions
- Receives extended traces-ids
  - Trace-id consists of Start PC and its branch outcomes
  - Extended by an ir-vec and intermediate PC
- Confidence mechanism builds up confidence for removal
  - Counter
- Generates the Program Counter (PC) of next instructions to be fetched by A-stream



K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

Delay Buffer

- Simple FIFO queue
  - A-stream pushes operand outcomes and information about skipped instructions and branch outcomes
  - R-stream pops everything from the buffer
- Branch outcomes are loaded into the instruction cache for prefetching
- Operand outcomes are merged with their respective instructions before they enter the execution engine



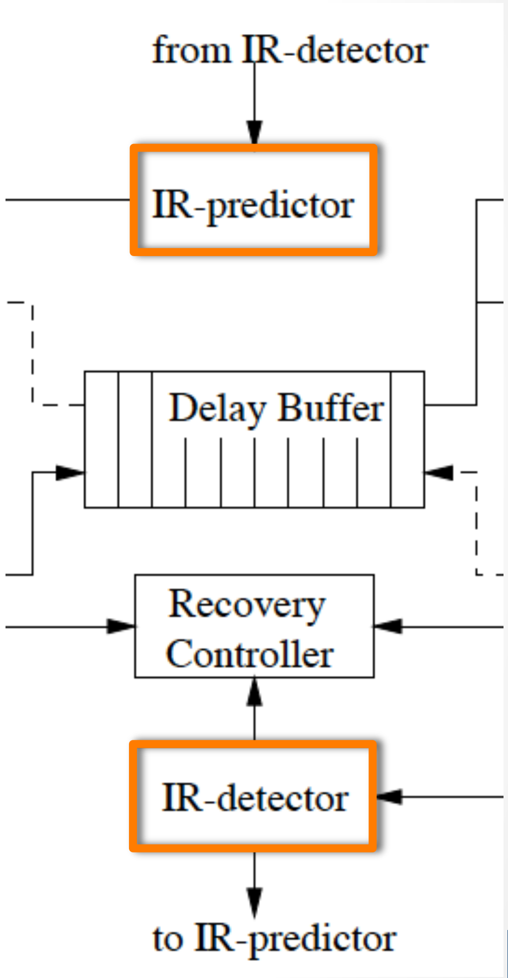
K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

Error Detection

Instructions IR-Predictor could remove by mistake

- Mispredicted branch
- Removal of effectual writes

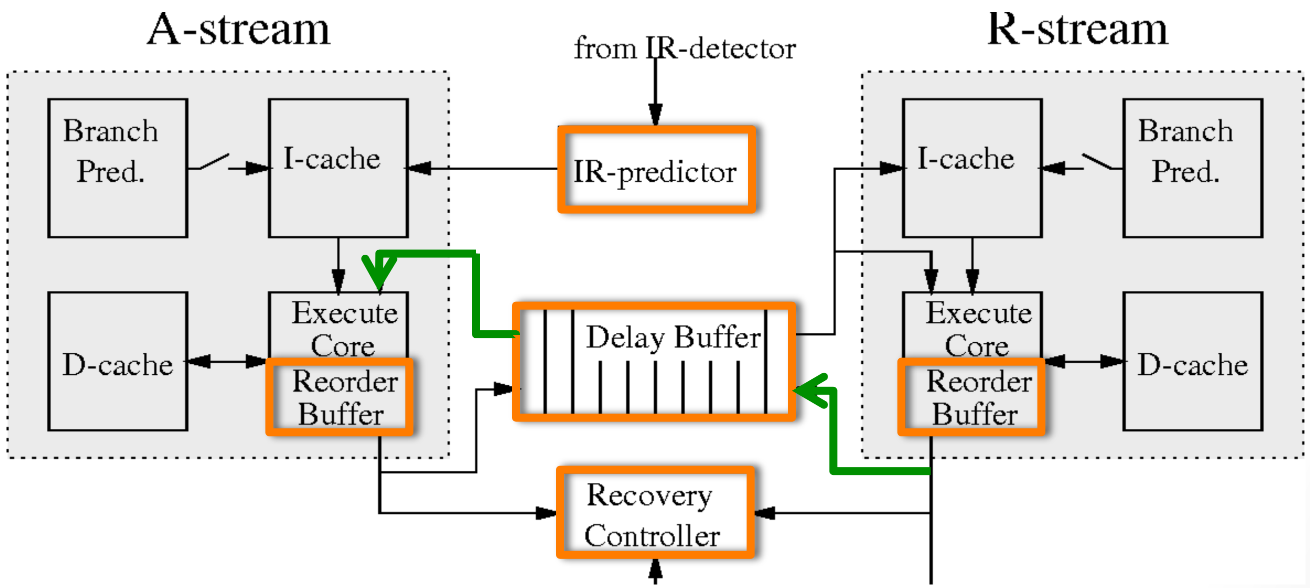
IR-detector can detect this early by comparing computed instructions from R-stream against predicted instructions used in A-stream



K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

Recovery

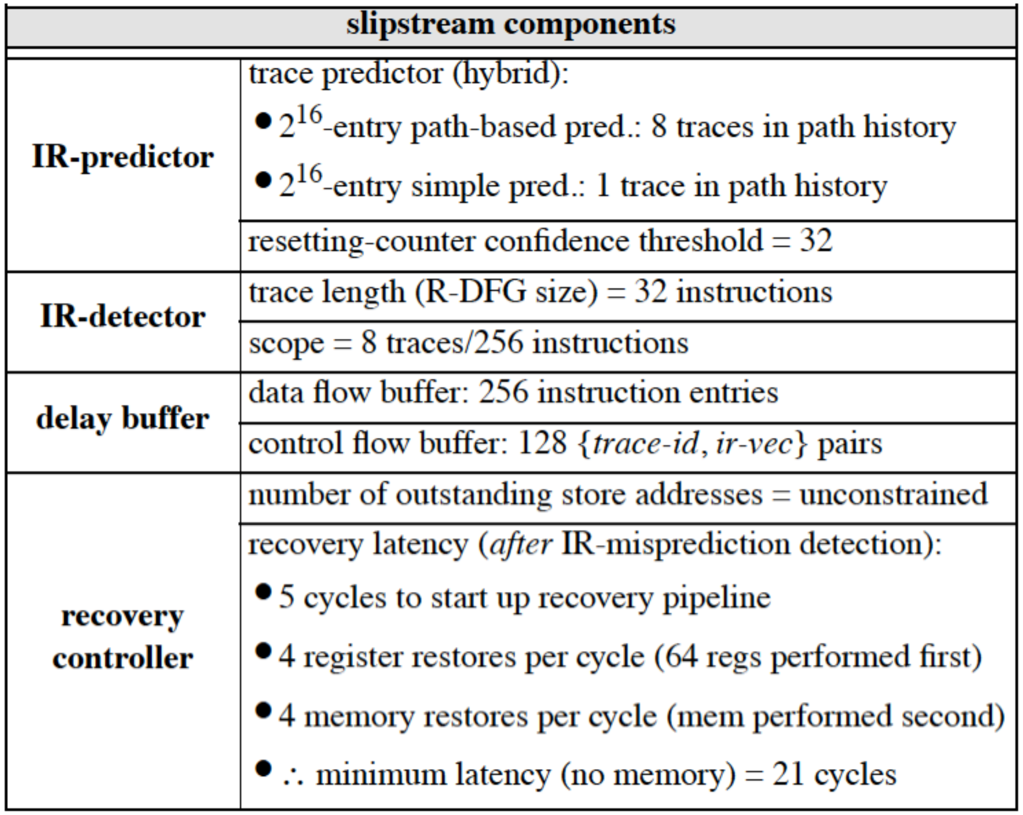
- Flush R-stream reorder buffer
- Flush delay buffer
- Backup IR-predictor to exact PC
- Flush A-stream reorder buffer
- Copy entire register file of R-stream to register file of A-stream via delay buffer
- The recovery controller correctly maps R-stream register to A-stream register



KEY RESULTS

Simulation Environment

- Execution driven simulation
- A Simplescalar, gcc-based compiler and MIPS-based ISA are used
- SPEC95 Benchmarks to compare compute-intensive integer performance



slipstream components	
IR-predictor	trace predictor (hybrid): ● 2^{16}-entry path-based pred.: 8 traces in path history ● 2^{16}-entry simple pred.: 1 trace in path history
	resetting-counter confidence threshold = 32
IR-detector	trace length (R-DFG size) = 32 instructions
	scope = 8 traces/256 instructions
delay buffer	data flow buffer: 256 instruction entries
	control flow buffer: 128 {*trace-id*, *ir-vec*} pairs
recovery controller	number of outstanding store addresses = unconstrained
	recovery latency (*after* IR-misprediction detection): ● 5 cycles to start up recovery pipeline ● 4 register restores per cycle (64 regs performed first) ● 4 memory restores per cycle (mem performed second) ● ∴ minimum latency (no memory) = 21 cycles

K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

Reference Models

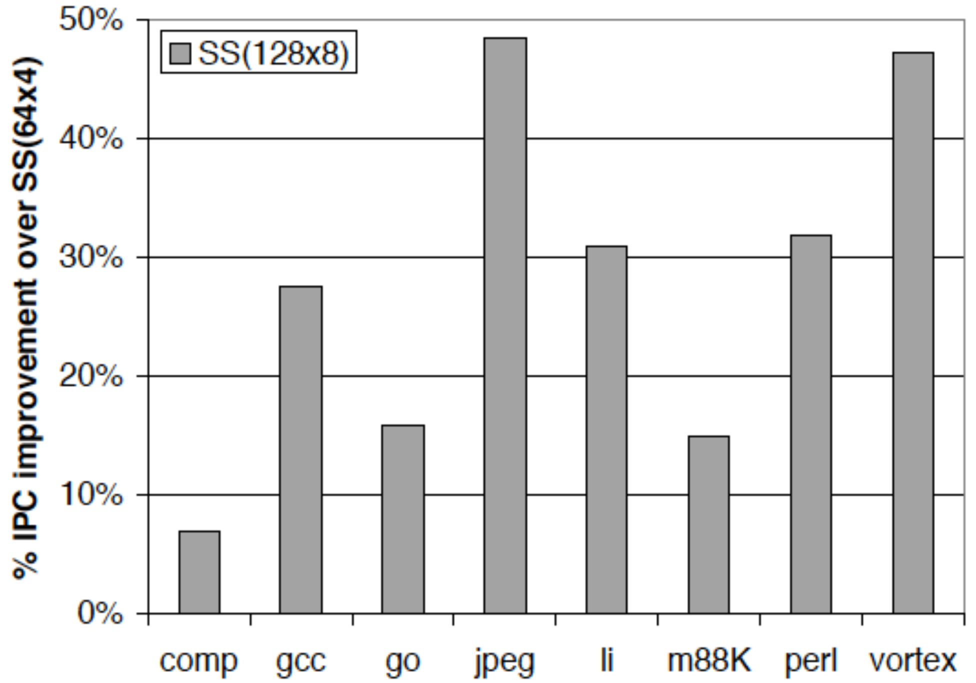
Performance of three models are compared

- SS(64x4) - A single copy of the program is run on one conventional 4-way superscalar processor with 64 ROB entries
- SS(128x8) - A single copy of the program is run on one conventional 8-way superscalar processor with 128 ROB entries
- CMP(2x64x4) - Slipstream processor using a CMP composed of two SS(64x4) cores

- Same trace predictor is used
- Performance measured in retired R-stream instructions divided by the number of cycles for both the A-stream and R-stream

Performance

- Results in comparison to SS(64x4) - 4-way superscalar processor
- Slipstream improves performance by 7% on average
- SS(128x8) - 8-way superscalar processor improves performance by 28% on average



K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

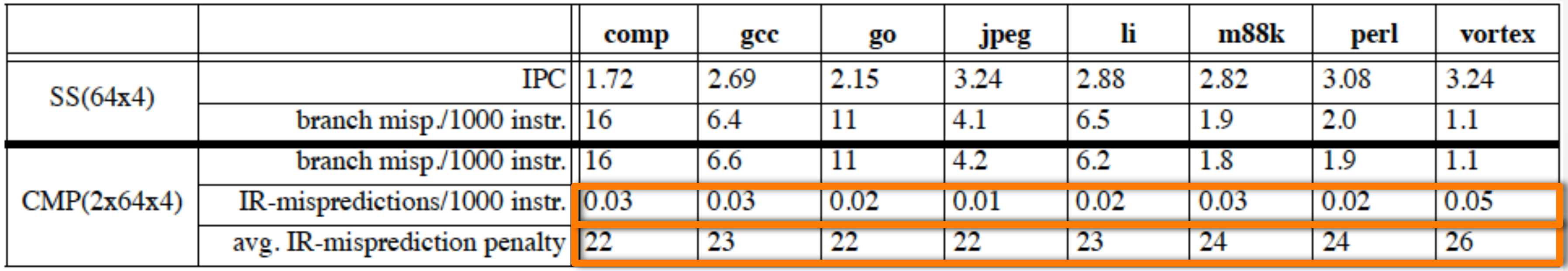
Performance

Even without surpassing the 8-way superscalar it has competitive potential

1. With much less complexity we achieved ¼ of the performance increase
2. More functionality and flexibility than a single super scalar processor
3. More potential if extended to an implementation on a 8-wide SMT, left for further work

Mispredictions

- Number of removed instructions correlates closely with performance improvement
- The confidence threshold of 32 results in less than 0.05 IR-mispredictions / 1000 instr.
- Average misprediction penalty is at most 26 with a minimum of 21 cycles



		comp	gcc	go	jpeg	li	m88k	perl	vortex
SS(64x4)	IPC	1.72	2.69	2.15	3.24	2.88	2.82	3.08	3.24
	branch misp./1000 instr.	16	6.4	11	4.1	6.5	1.9	2.0	1.1
CMP(2x64x4)	branch misp./1000 instr.	16	6.6	11	4.2	6.2	1.8	1.9	1.1
	IR-mispredictions/1000 instr.	0.03	0.03	0.02	0.01	0.02	0.03	0.02	0.05
	avg. IR-misprediction penalty	22	23	22	22	23	24	24	26

K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

Strengths

- The paper is ahead of its time, because it addresses multiple very fundamental problems
  - limits of parallelism
  - physical limits in processor design
  - ➢ Today, these problems are even more important, as they were in 2000

- Idea is different and innovative with the attempt of utilizing idle processors for improving single thread performance

- Looking for simplicity by not redesigning whole systems, instead it attempts to make minor changes to improve performance

Weaknesses

- Only touches surface of the underlying problem
  - Error detection is not solved with two redundant executions
  - Only little evaluation about performance
- No insight about power consumption
- Many informal speculations and conclusions
  - The whole evaluation of fault tolerance is informal
  - Speculations about other predictors and implementations with very little reasoning
- Hard to read and understand, of course also because it is outdated

Takeaways

- Idea deserves attention and might be a nice base for further improvements
- *“Two programs combined finish sooner than either would alone”*
- Problem of transient hardware faults not properly addressed

DISCUSSION

Discussion Starters

- What changes do you think have the most potential for additional performance increase?
- How can we really address the fault tolerance improvement?
- What happens if the A-stream is too fast?
- Can you think about improvements in todays heavily parallelized architectures with the slipstream paradigm

FURTHER/RELATED WORKS

Related Works

- E. Rotenberg. AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors. 29th Int'l Symp. on Fault-Tolerant Computing , June 1999
- E. Rotenberg. Exploiting Large Ineffectual Instruction Sequences. Technical Report, Department of Electrical and Computer Engineering, North Carolina State University,Nov. 1999
- A. Roth and G. Sohi. Speculative Data-Driven Multithreading. Technical Report CS-TR-2000-1414, Computer Sciences Department, University of Wisconsin - Madison, April 2000

Further Works

- Zach Purser, Karthik Sundaramoorthy, Eric Rotenberg: *A study of Slipstream processors*, MICRO 33 Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture Dec. 2000

- Nandita Vijaykumar, Gennady Pekhimenko, Adwait Jogt, Abhishek Bhowmick, Rachata Ausavarungnirun, Chita Dast, Mahmut Kandemirt, Todd C. Mowry, Onur Mutlu: *A case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling flexible data compression with assist warps,* 2015 ACM/IEEE International Symposium on Computer Architecture (ISCA)

BACKUP SLIDES

Compiler / Slipstream Example

```
function func(a, b) {
        var x;
        var i = 3000;
        while (i--) {
                // dead store
                x = a + b;
        }
}
```

Compiler can optimize

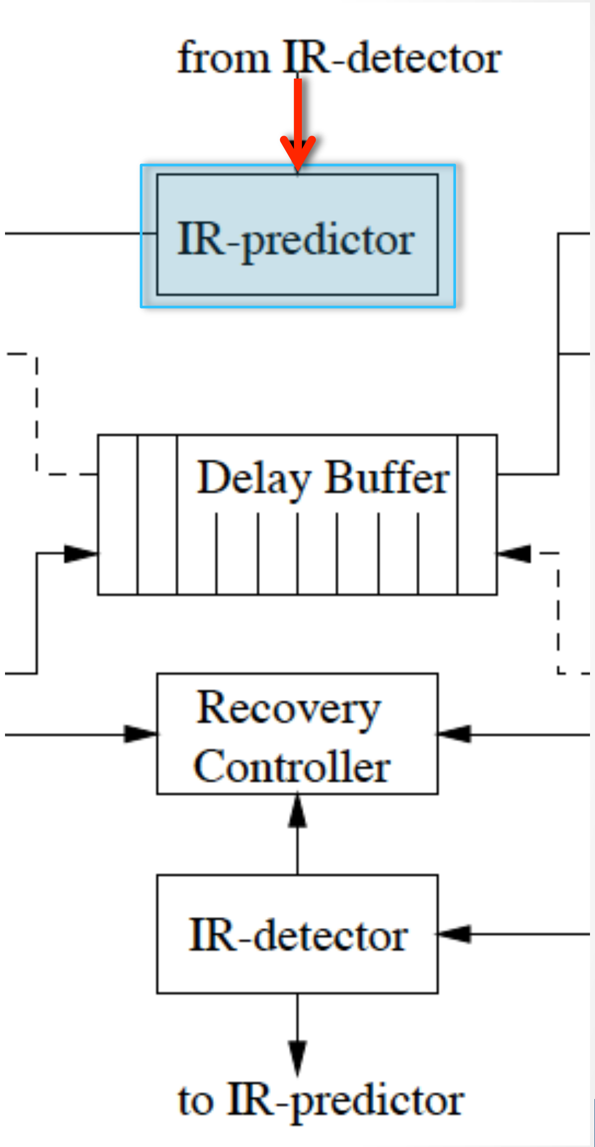
```
function func(a, b) {
        var y = random();
        var x;
        var i = 3000;
        while (i--) {
                x = a + b;
                if( i == y)
                        read x;
        }
}
```

Slipstream can predict

IR Predictor

Instruction Removal Predictor

- They present a modified trace predictor
- IR-detector suggests instruction to remove
- Confidence mechanism (counter) builds up confidence for removal
- Generates the PC of next instructions to be fetched by A-stream

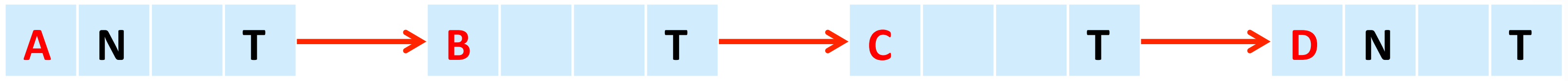


IR Predictor - Trace Predictor

- Set of dynamic instruction is a trace
- Start PC and branch outcome form a trace id

{A, N, T, T, T, N, T} - trace id

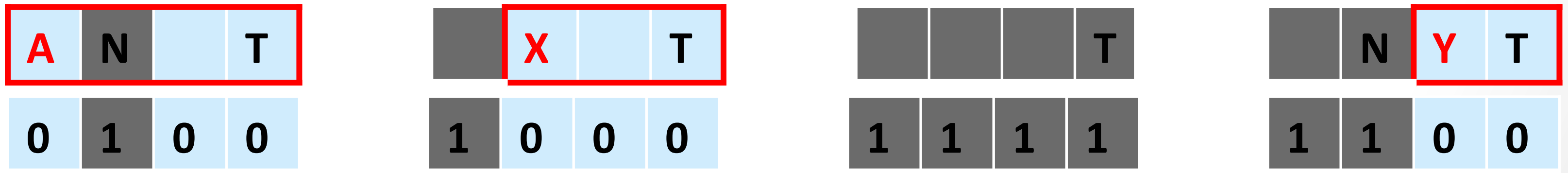
PC's B, C, D are produced using Branch Target Buffer (BTB)



Modification:

- Add instruction removal bit vector ir-vec, from IR detector
- Add intermediate PC values, from IR detector
- Add confidence mechanism

X, Y, intermediate PC's

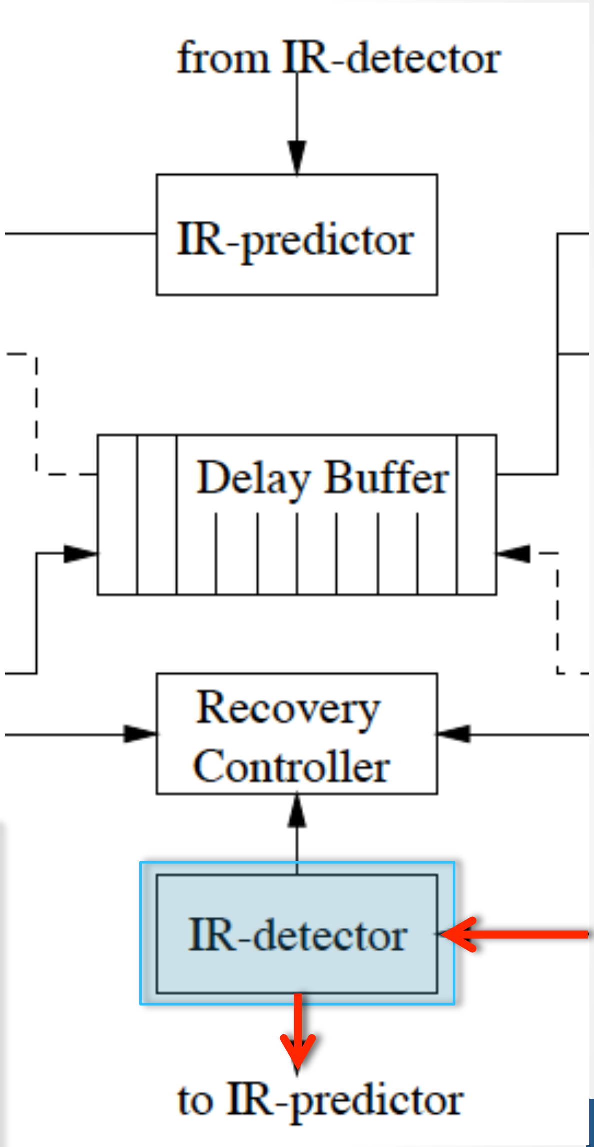


IR Detector

- Operand rename table performs dependence checking and detects removable instructions
- Construct a Reverse Data Flow Graph R-DFG

A R-DFG are 32 instructions as a single trace

Stores R-DFG for multiple traces



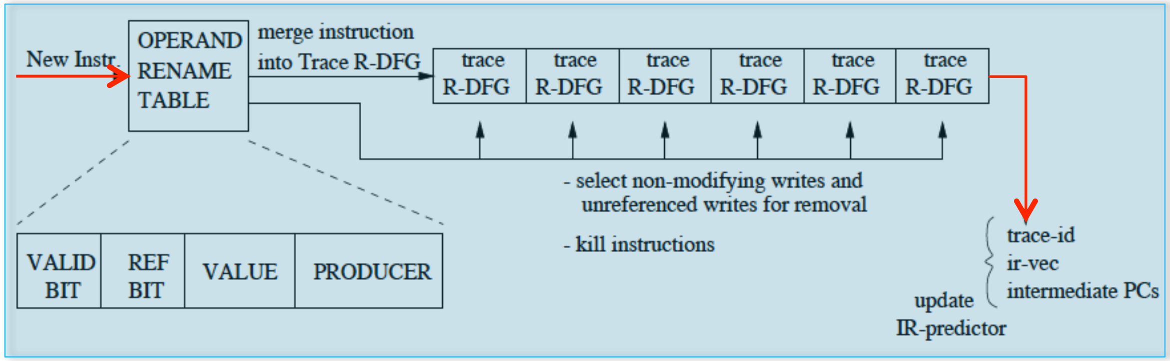


Figure 3: IR-detector.

K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

IR Detector

- All branches are candidates for removal

For the oldest trace

- Instruction removal bit vector ir-vec is formed
- Intermediate PCs are computed

Trace Id, ir-vec and intermediate PCs are loaded into the IR-predictor and the R-DFG circuitry is reclaimed for a new trace

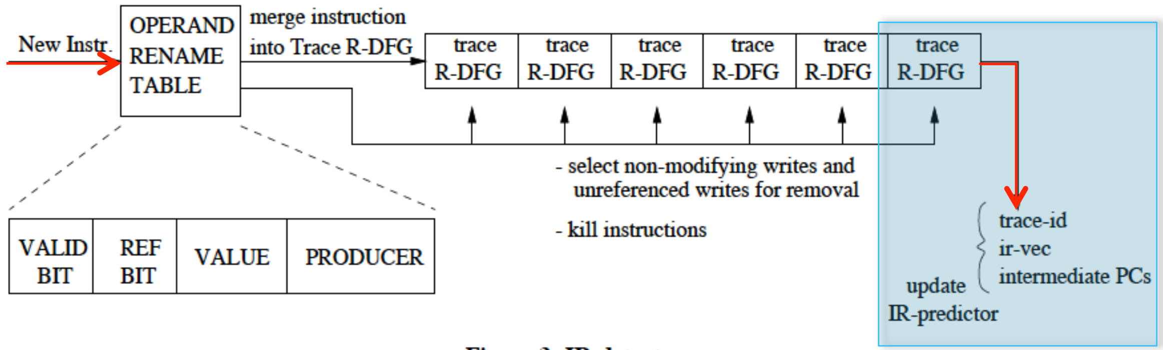
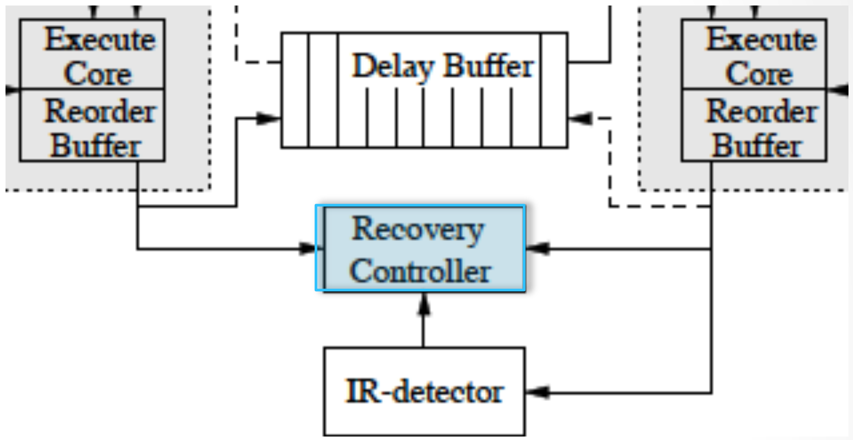


Figure 3: IR-detector.

K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.

Recovery Controller

In normal mode, the recovery controller receiver control signals and addresses of store instructions from A-stream, R-stream and IR-detector



- After a misprediction, the recovery controller correctly maps R-stream register to A-stream register
- Instructions must be undone or done in A-stream
  - Every instruction A-stream is ahead of R-stream are undone
  - Every predicted and unverified stores are done by copy data from R-stream
- With the comparison of ir-vec's by the IR detector, we can determine a bounding for the tracked addresses

K. Sundaramoorthy, Z. Purser, and E. Rotenberg. Slipstream Processors: Improving both Performance and Fault Tolerance. ACM SIGPLAN Notices Nov. 2000.