Slipstream Processors Revisited: Exploiting Branch Sets

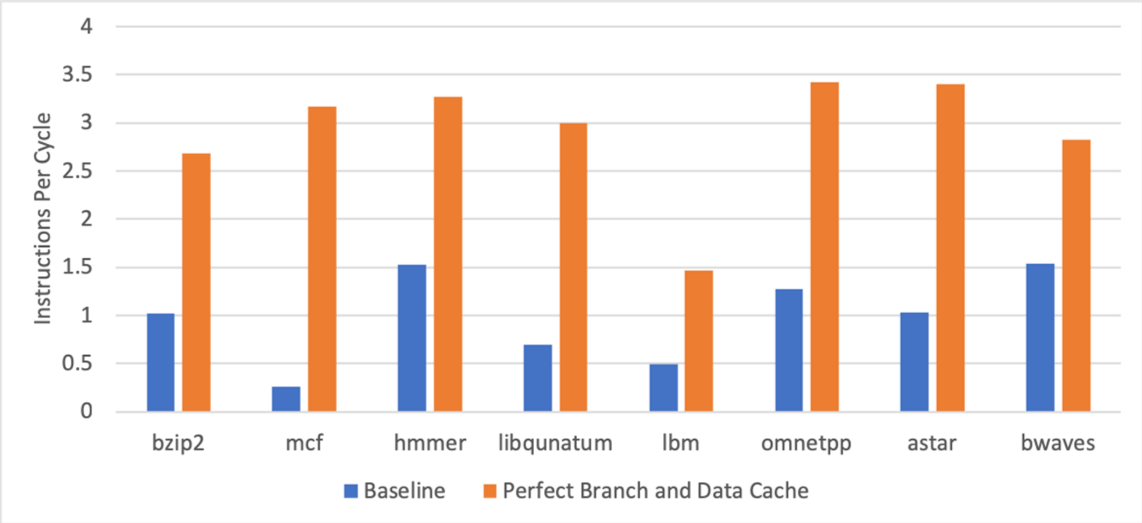
Vinesh Srinivasan
Dep't of Elec. and Comp. Eng.
North Carolina State University
vsriniv3@ncsu.edu

Rangeen Basu Roy Chowdhury
Intel Corporation
rangeen.basu.roy.chowdhury@intel.com

Eric Rotenberg
Dep't of Elec. and Comp. Eng.
North Carolina State University
ericro@ncsu.edu

Single-Thread Performance Limiters

- Delinquent branches and loads limit single-thread performance
- Individually they are bad
- Even worse when they coincide
  - Cache-missed load that feeds a mispredicted branch
  - Large-window processor loses latency tolerance in this case
  - Squash many instructions after the cache-missed load



← Peak IPC=4 for this 4-wide fetch/retire core.

Pre-execution via Helper Threads

- Resolve hard-to-predict branches and initiate delinquent loads before these instructions are fetched by the main thread
- Two classes of pre-execution
  - Per-dynamic-instance helper threads: Each helper thread is the backward slice of instructions leading to a single dynamic instance of a branch or load.
  - Two redundant threads in a leader-follower arrangement: Leader thread is speculatively reduced by pruning instructions.

Objective

Design a new pre-execution microarchitecture that meets four criteria:

1. Leader-follower style pre-execution
2. Fully automated using only hardware
3. Targets both branches and loads
4. Effective at that which is targeted

Avoid tricky issues of per-dynamic-instance helper threads:

- Timing of forking, accounting for live-in values of helper thread
- Lining up pre-executed branch outcomes with corresponding instances in the main thread

Objective

Design a new pre-execution microarchitecture that meets four criteria:

1. Leader-follower style pre-execution
2. Fully automated using only hardware
3. Targets both branches and loads
4. Effective at that which is targeted

- Not opposed to compiler support. S/w and h/w co-dependency introduces risk.

Objective

Design a new pre-execution microarchitecture that meets four criteria:

1. Leader-follower style pre-execution
2. Fully automated using only hardware
3. Targets both branches and loads
4. Effective at that which is targeted

- Unified solution for performance-degrading instructions

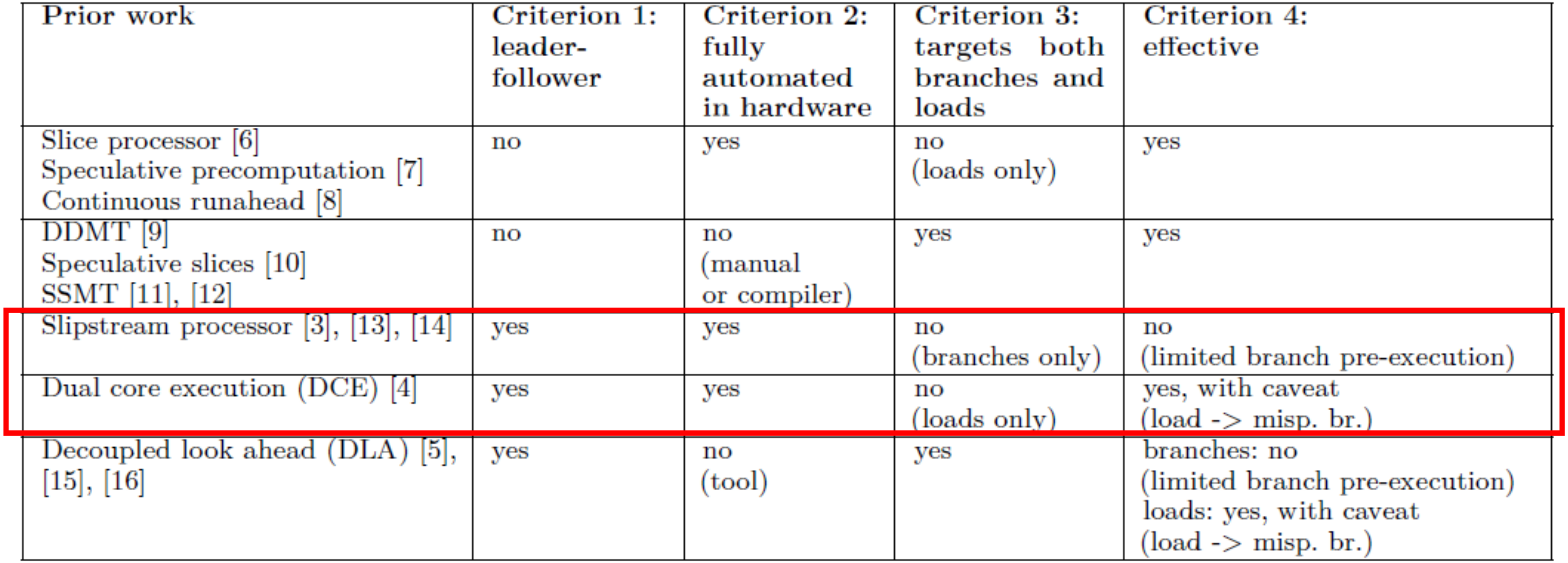
Objective

Design a new pre-execution microarchitecture that meets four criteria:

1. Leader-follower style pre-execution
2. Fully automated using only hardware
3. Targets both branches and loads
4. Effective at that which is targeted

- Overcome performance limitations of other leader-follower microarchitectures

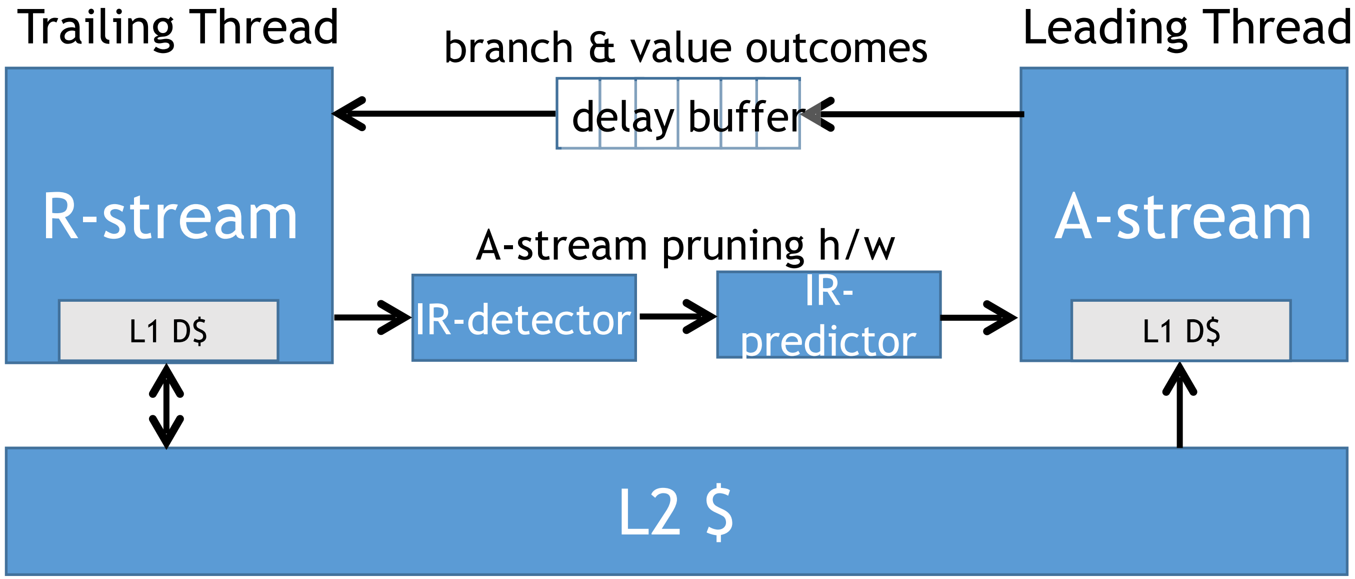
No Prior Work Meets All Four Criteria



Prior work	Criterion 1: leader-follower	Criterion 2: fully automated in hardware	Criterion 3: targets both branches and loads	Criterion 4: effective
Slice processor [6] Speculative precomputation [7] Continuous runahead [8]	no	yes	no (loads only)	yes
DDMT [9] Speculative slices [10] SSMT [11], [12]	no	no (manual or compiler)	yes	yes
Slipstream processor [3], [13], [14]	yes	yes	no (branches only)	no (limited branch pre-execution)
Dual core execution (DCE) [4]	yes	yes	no (loads only)	yes, with caveat (load -> misp. br.)
Decoupled look ahead (DLA) [5], [15], [16]	yes	no (tool)	yes	branches: no (limited branch pre-execution) loads: yes, with caveat (load -> misp. br.)

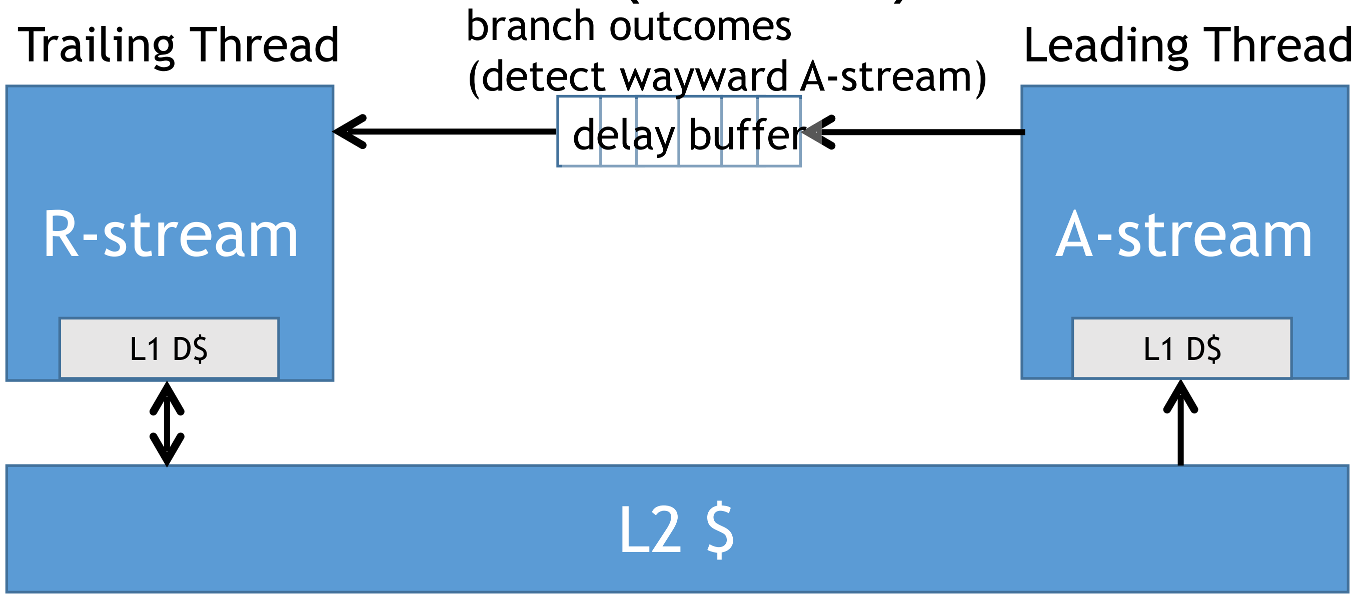
Table I: Related work analysis.

Slipstream Processor



- Remove backward slices of confident branches in the A-stream to pre-execute unconfident branches
  - *Ineffective for phases dominated by hard-to-predict branches, when branch pre-execution is most needed*
- W.r.t. loads: Backward slice removal does not stop short at delinquent loads, failing to convert removed loads into non-binding prefetches

Dual Core Execution (DCE)



Load converted to non-binding prefetc
if blocks retire stage, silence executio
of dependent instructions.

- Convert cache-missed loads that block A-stream’s retire stage to non-binding prefetches, and silence execution of their dependent instructions
  - Very good at tolerating cache-missed loads, *except when their dependent branches are mispredicted*
- W.r.t. branches: No A-stream pruning *per se*, so no branch pre-execution

Slipstream Processor 2.0

- Remove *forward control-flow slices* of delinquent branches and loads
- Overcomes performance limitations of Slipstream and DCE
- Two firsts:
  - Leader-follower-style branch pre-execution without relying on confident instr. removal
  - Tolerate cache-missed loads that feed mispredicted branches

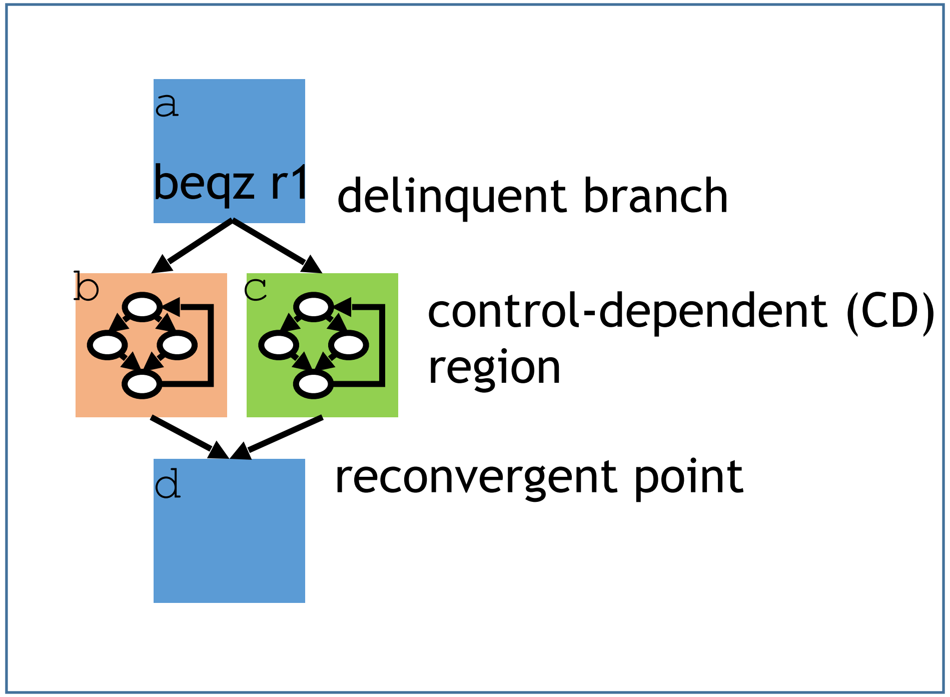
- Meets all four criteria
  1. Leader-follower style pre-execution
  2. Fully automated using only hardware
  3. Targets both branches and loads
  4. Effective at that which is targeted (improves upon Slipstream and DCE)

- Microarch. turbo-boost: Auto-enable/disable A-stream based on profitability

Delinquent Branch Pre-execution (DBP)

- Forward control-flow slice of a delinquent branch
  - Control-dependent (CD) region of the delinquent branch
  - Other branches that are control-independent data-dependent (CIDD) with respect to the delinquent branch, and their CD regions

DBP (cont.)



a

beqz r1

delinquent branch

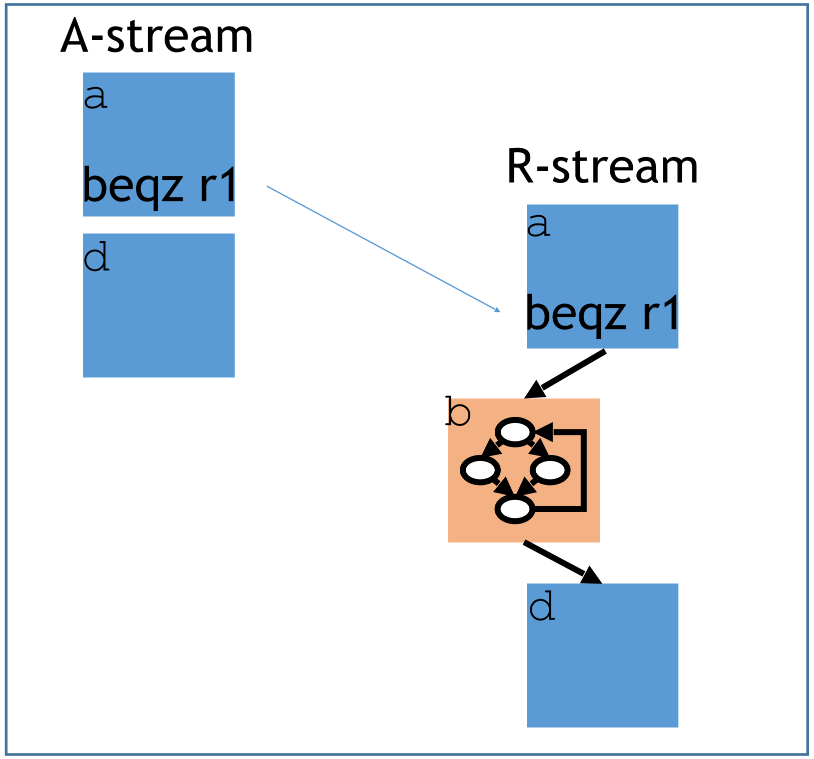
b

c

control-dependent (CD) region

d

reconvergent point



A-stream

a

beqz r1

d

R-stream

a

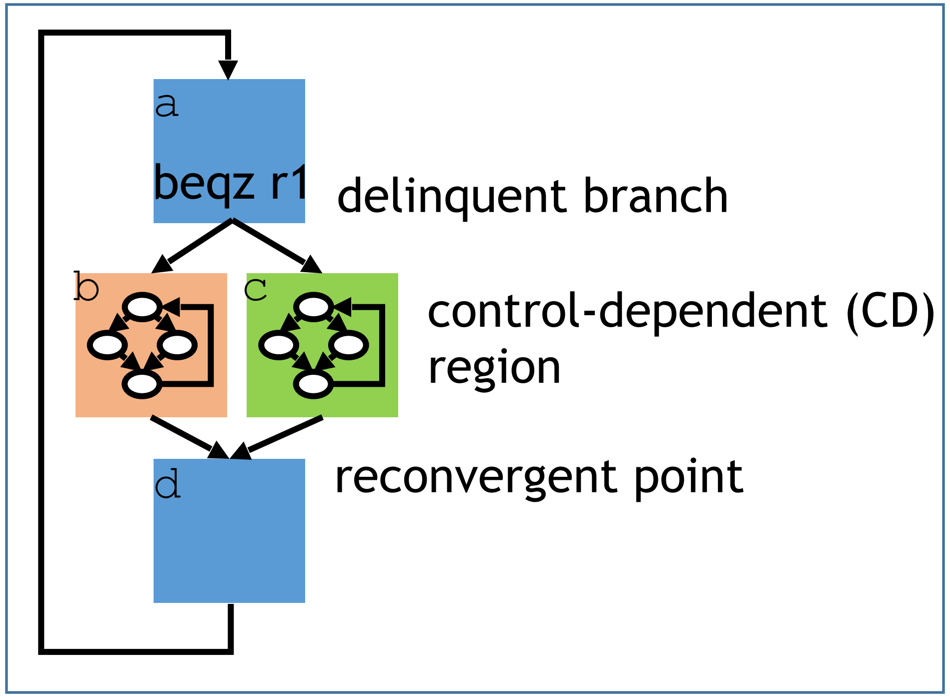
beqz r1

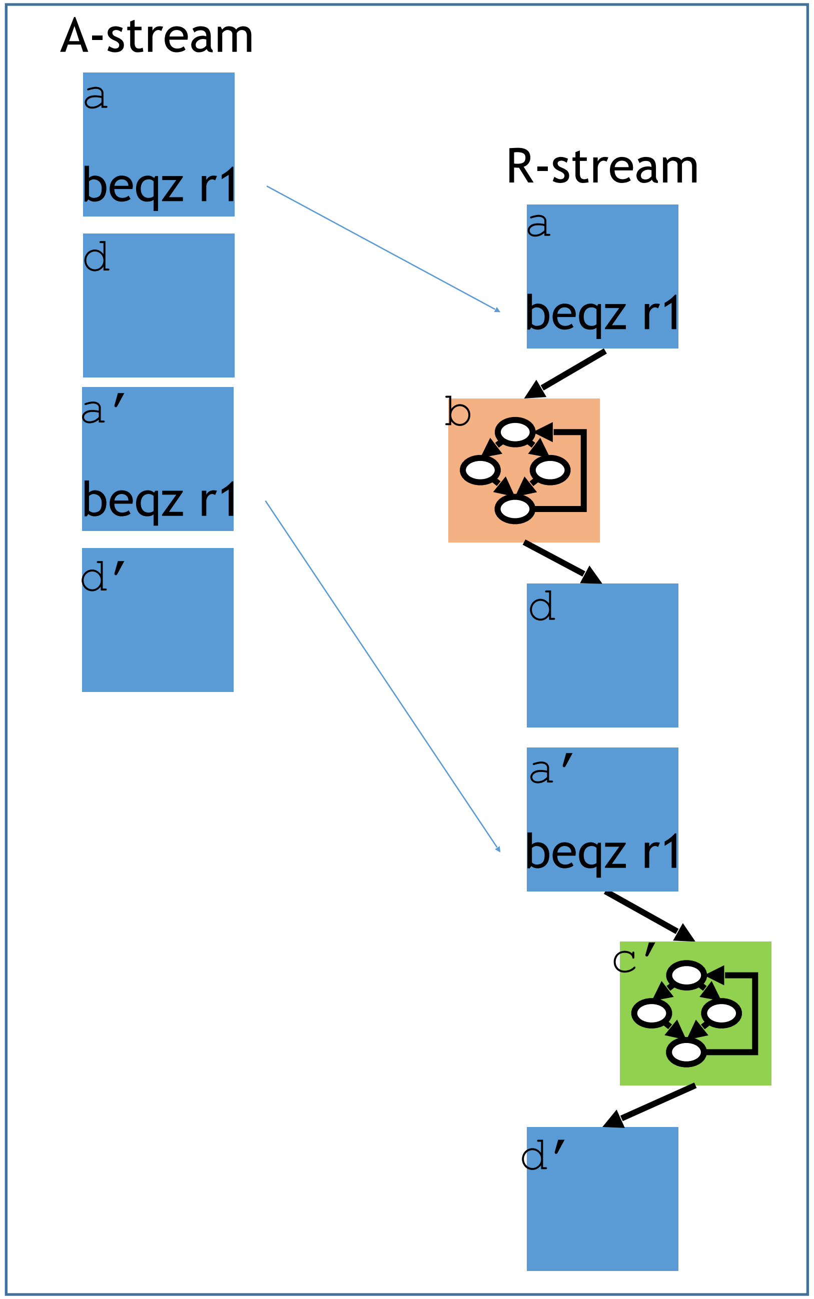
b

d

- A-stream
  - Delinquent branch converted to unconditional “branch-to-reconvergent-point”
  - Resolves delinquent branch’s predicate
  - Not slowed by would-be mispredictions of delinquent branch
- R-stream
  - Receives accurate prediction for delinquent branch from A-stream
  - Locally predicts and resolves any branches nested within skipped CD region: A-stream is insulated from any R-stream-local mispredictions within the CD regio

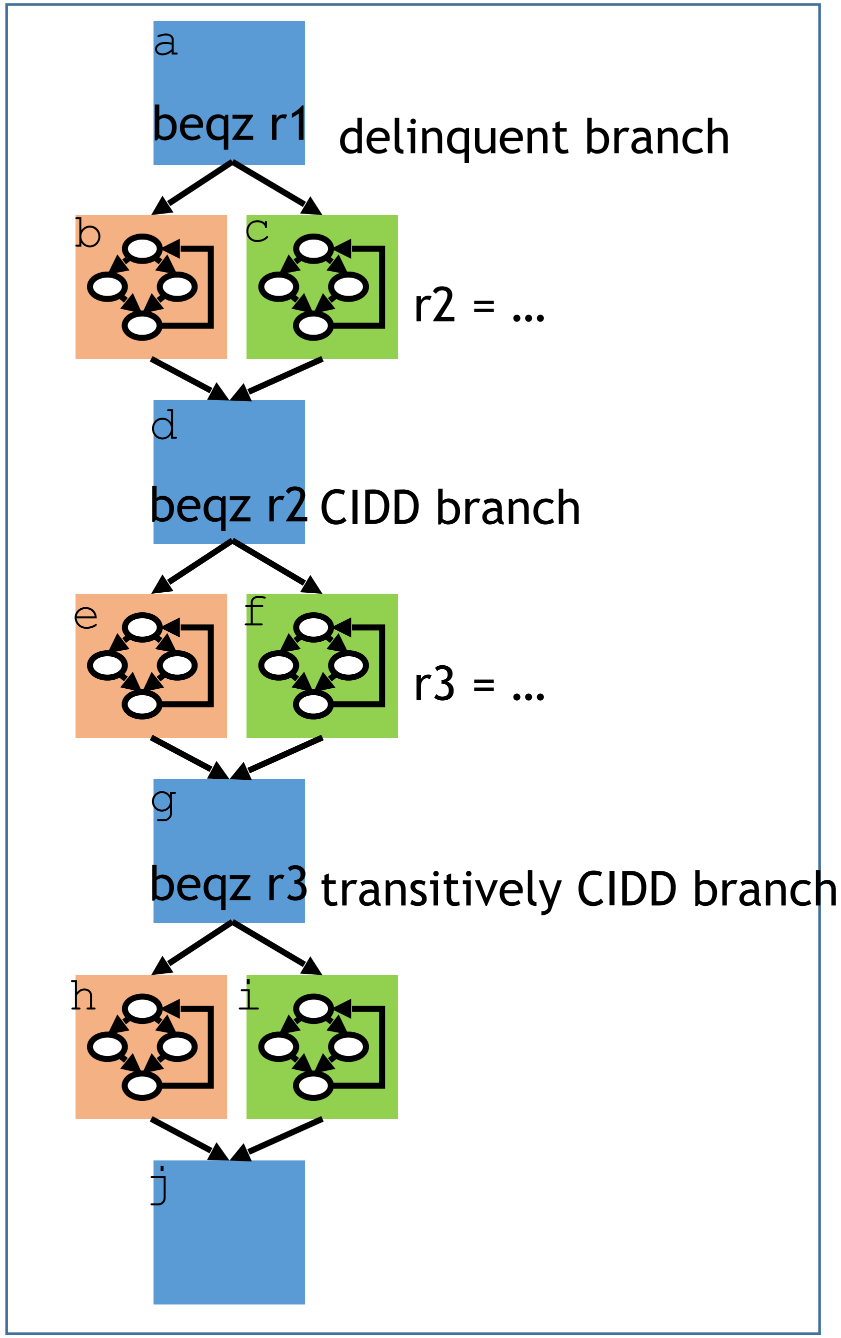
DBP (cont.)

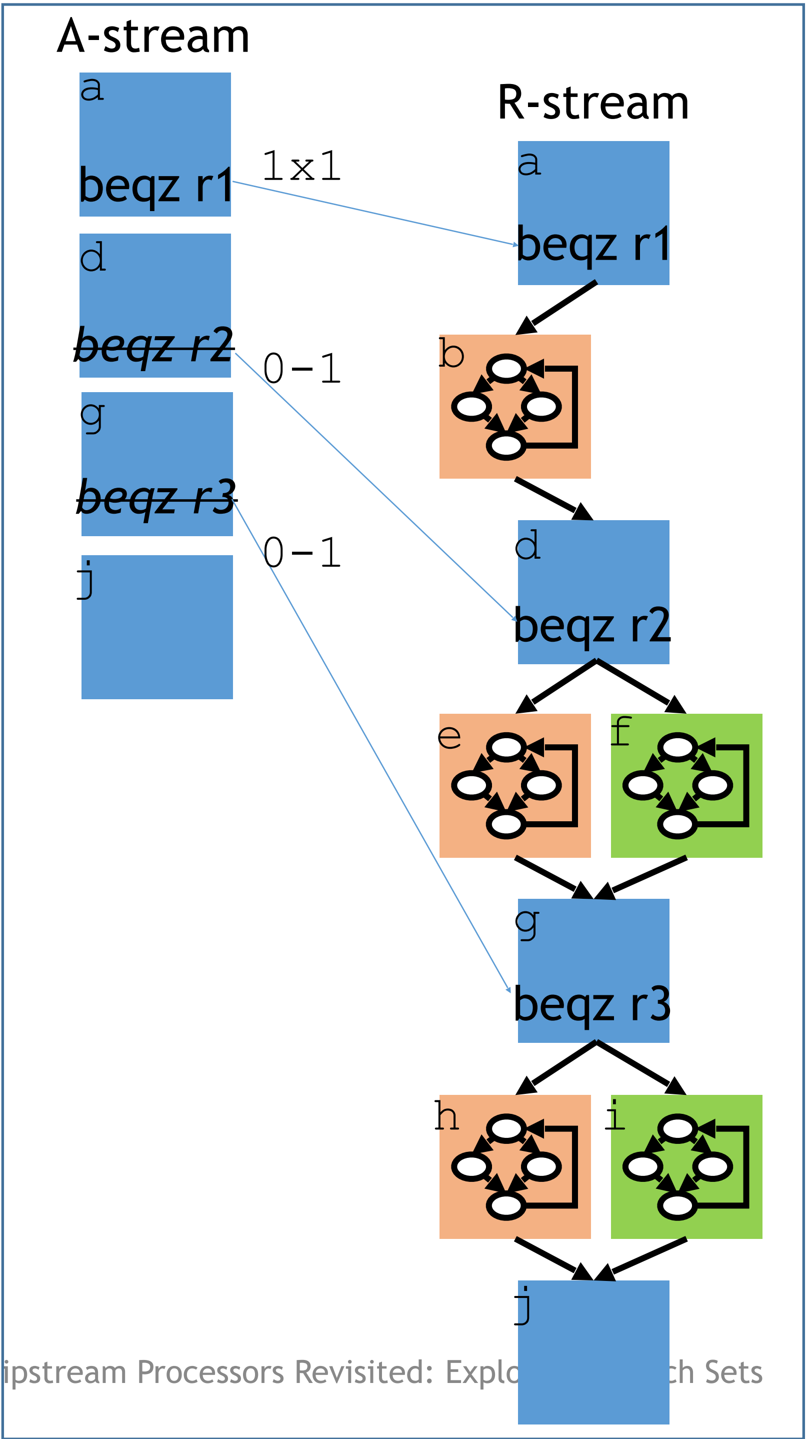

a
beqz r1
delinquent branch
b
c
control-dependent (CD) region
d
reconvergent point


A-stream
a
beqz r1
d
a'
beqz r1
d'
R-stream
a
beqz r1
b
d
a'
beqz r1
c'
d'

DBP (cont.)

- Forward control-flow slice of a delinquent branch
  - Control-dependent (CD) region of the delinquent branch
  - Other branches that are control-independent data-dependent (CIDD) with respect to the delinquent branch, and their CD regions





Branch encodings supplied by A-stream to R-stream:

`1x0`: executed branch

`1x1`: pre-executed branch

`0-1`: CIDD branch

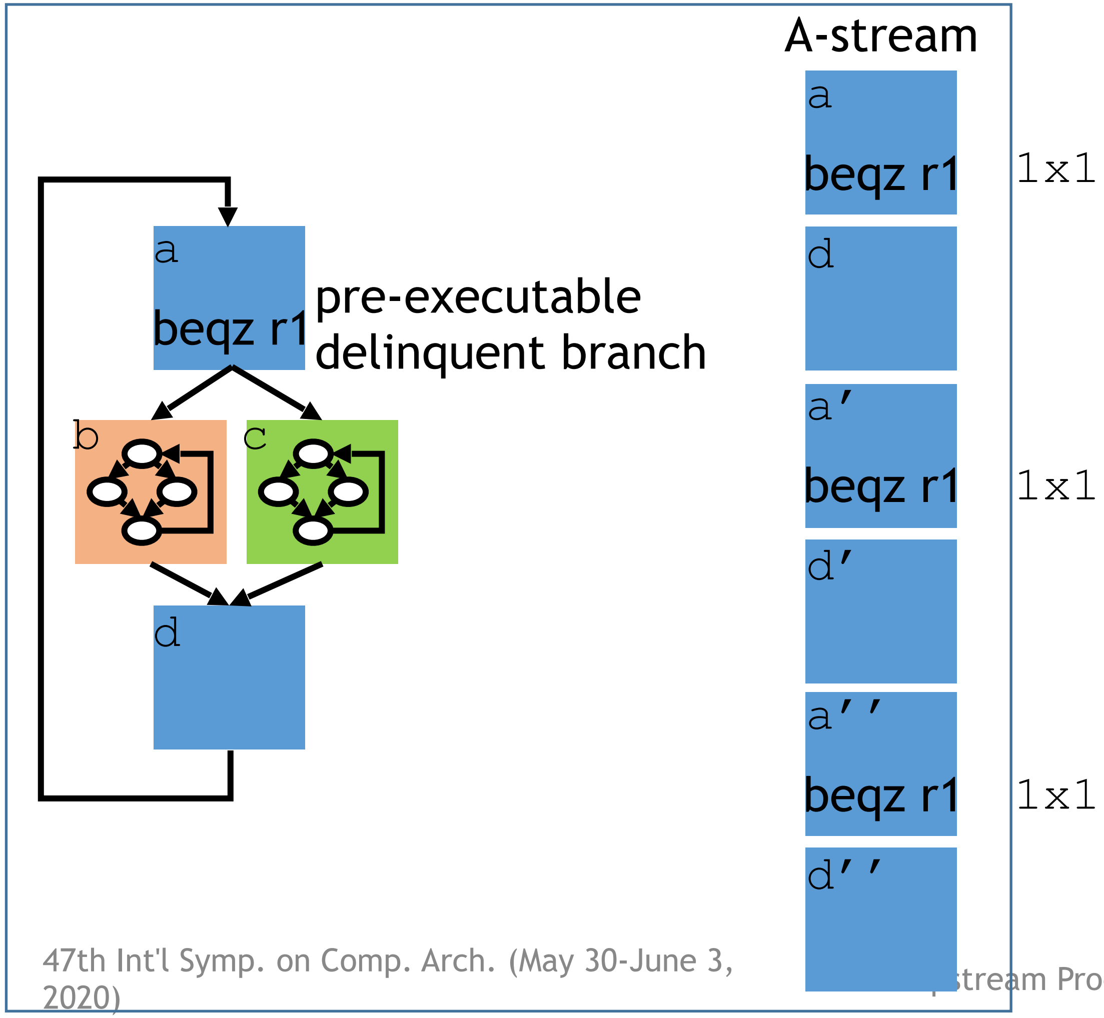
CD region skipped (1) or not

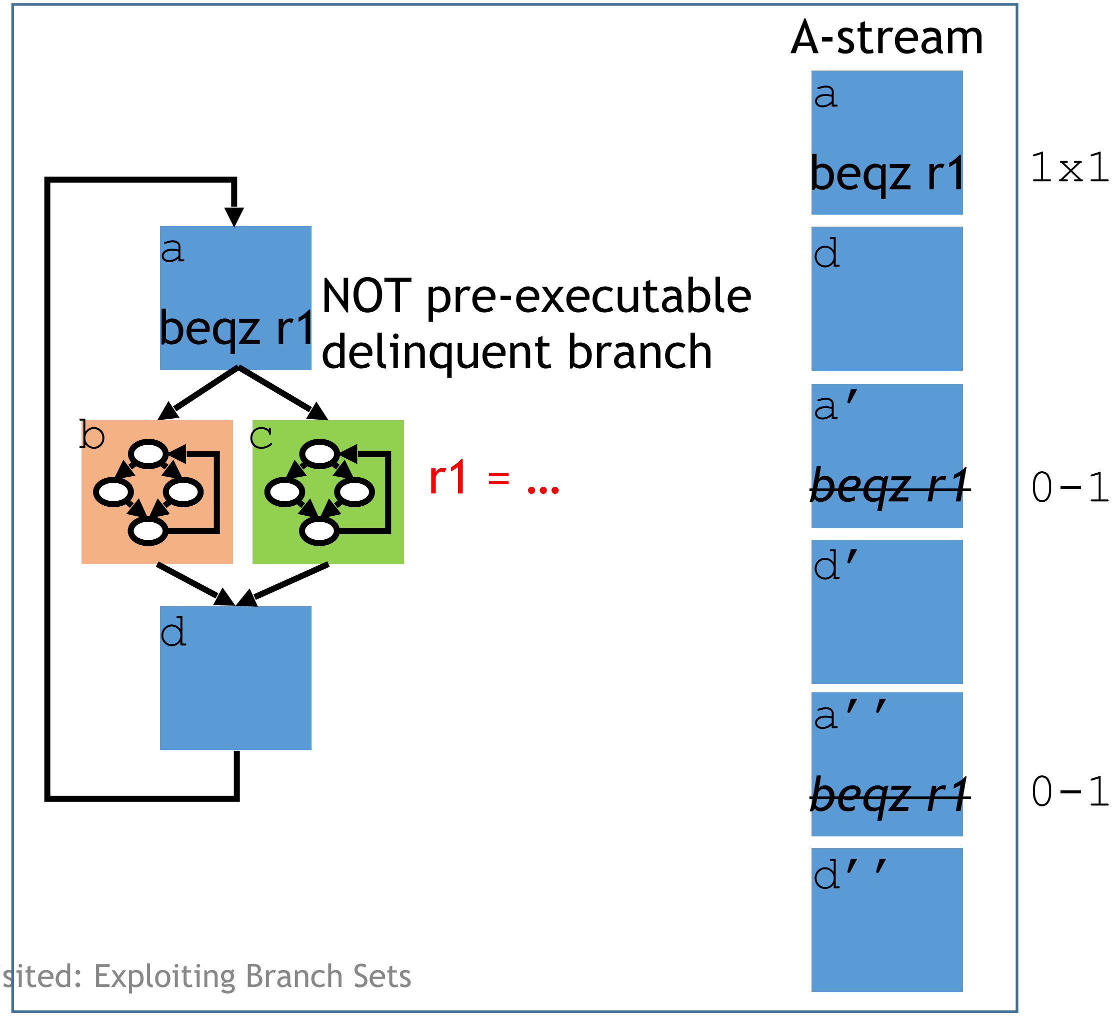
outcome if executed (x=0 or 1)

executed (1) or not (0) by A-strea

Pre-executable vs. not pre-executable branches

- A delinquent branch is pre-executable only if it is not in its own forward control-flow slice, *i.e.*, not self-dependent

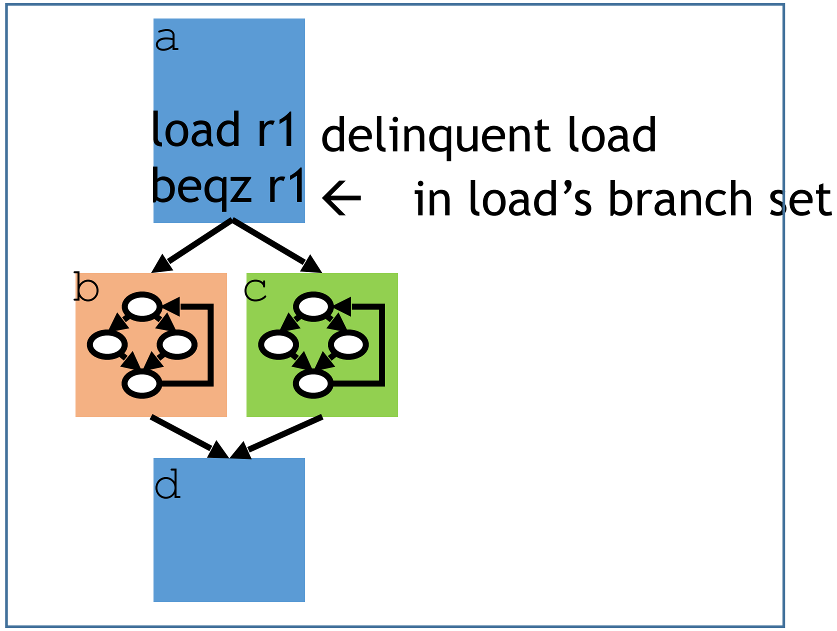


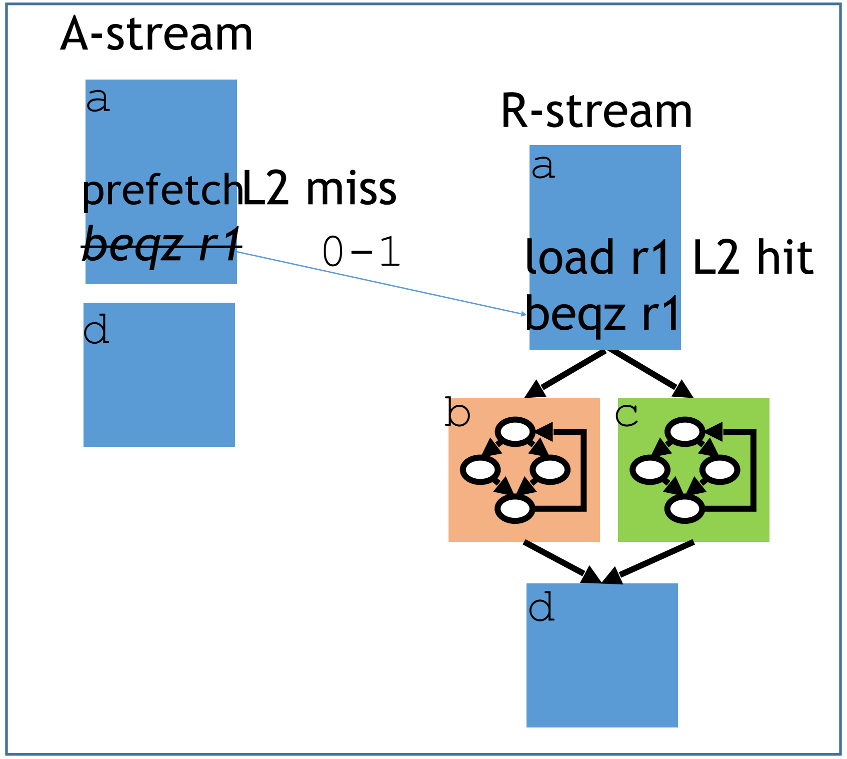


Branch Sets

- Branch set of a delinquent branch or load
  - CIDD branches with respect to the branch or load
- Concept of branch sets is important for two reasons:
  1. Branch set constitutes the forward control-flow slice to be removed (in addition to the delinquent branch’s own CD region)
  2. A delinquent branch is pre-executable (eligible for DBP) if it is not in its own branch set

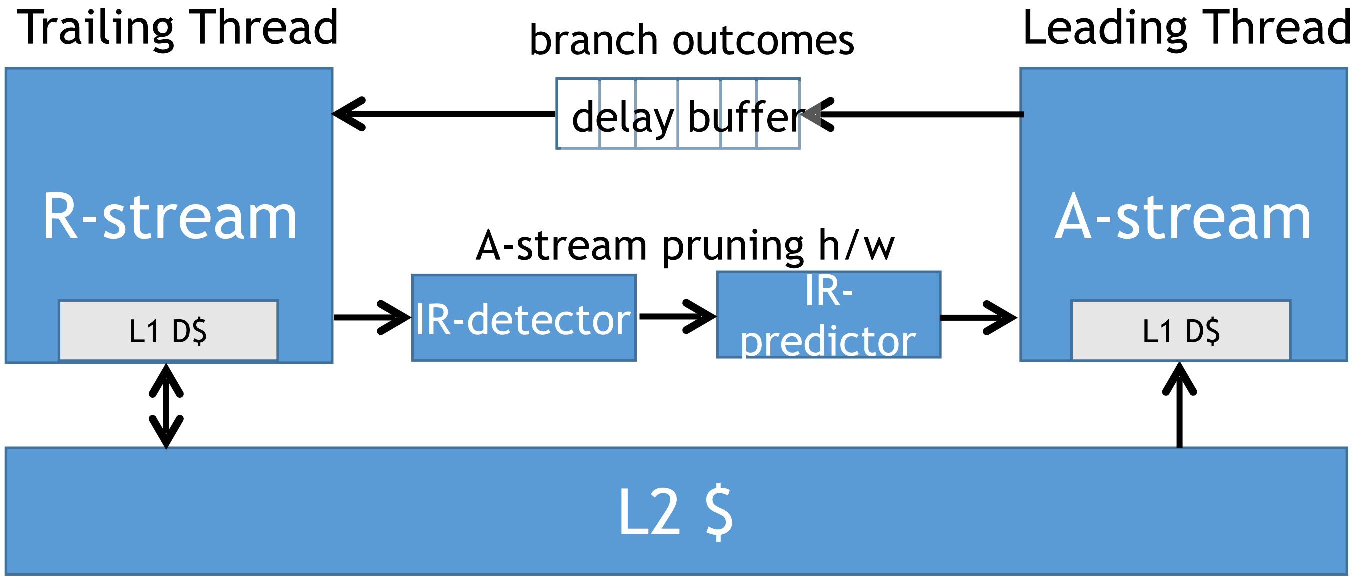
Delinquent Load Prefetching (DLP)





- A-stream
  - Delinquent load converted to non-binding prefetch
  - Branches in its branch set are silenced and their CD regions skipped
- R-stream
  - Delinquent load hits in L2$
  - Locally predict and resolve any missing control-flow (branches in the load's branch set, and branches nested in their CD regions):
    A-stream is insulated from any R-stream-local mispredictions in load's forward control-flow slice

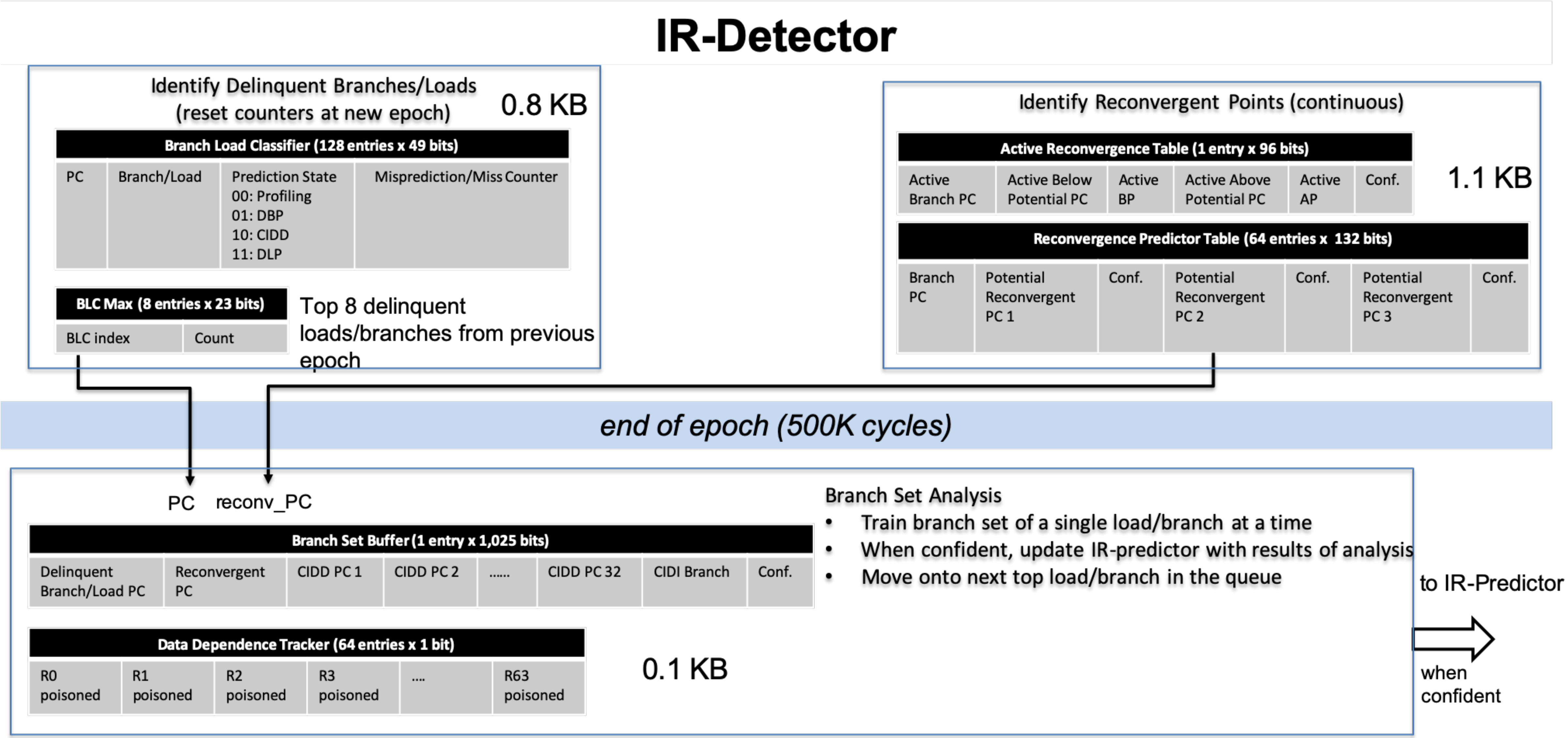
Slipstream 2.0 Microarchitecture



- Follows Slipstream template, but components implemented differently
  - Delay Buffer: 3-bit branch encodings
  - IR-predictor: entries for DBP branches ("1x1"), DLP loads, and CIDD branches ("0-1")
  - Instruction-Removal Detector (IR-detector): delinquent load/branch classifier, reconvergent PC detector, and branch set analysis

4.2 KB

Slipstream 2.0’s IR-detector



IR-Detector

Identify Delinquent Branches/Loads
(reset counters at new epoch)

0.8 KB

Branch Load Classifier (128 entries x 49 bits)

PC	Branch/Load	Prediction State 00: Profiling 01: DBP 10: CIDD 11: DLP	Misprediction/Miss Counter

BLC Max (8 entries x 23 bits)

BLC index	Count

Top 8 delinquent loads/branches from previous epoch

Identify Reconvergent Points (continuous)

Active Reconvergence Table (1 entry x 96 bits)

Active Branch PC	Active Below Potential PC	Active BP	Active Above Potential PC	Active AP	Conf.

1.1 KB

Reconvergence Predictor Table (64 entries x 132 bits)

Branch PC	Potential Reconvergent PC 1	Conf.	Potential Reconvergent PC 2	Conf.	Potential Reconvergent PC 3	Conf.

end of epoch (500K cycles)

PC reconv_PC

Branch Set Buffer (1 entry x 1,025 bits)

Delinquent Branch/Load PC	Reconvergent PC	CIDD PC 1	CIDD PC 2	......	CIDD PC 32	CIDI Branch	Conf.

Data Dependence Tracker (64 entries x 1 bit)

R0 poisoned	R1 poisoned	R2 poisoned	R3 poisoned	....	R63 poisoned

0.1 KB

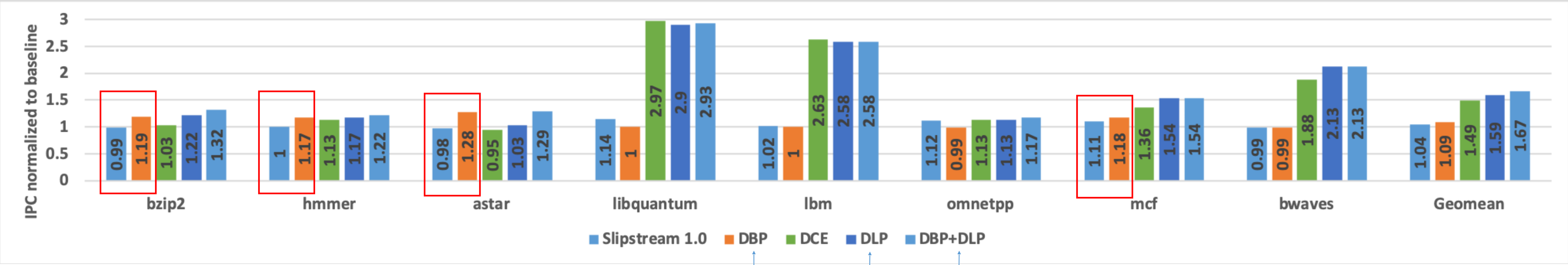
Branch Set Analysis

- Train branch set of a single load/branch at a time
- When confident, update IR-predictor with results of analysis
- Move onto next top load/branch in the queue

to IR-Predictor

when confident

Results: DBP vs. Slipstream 1.0

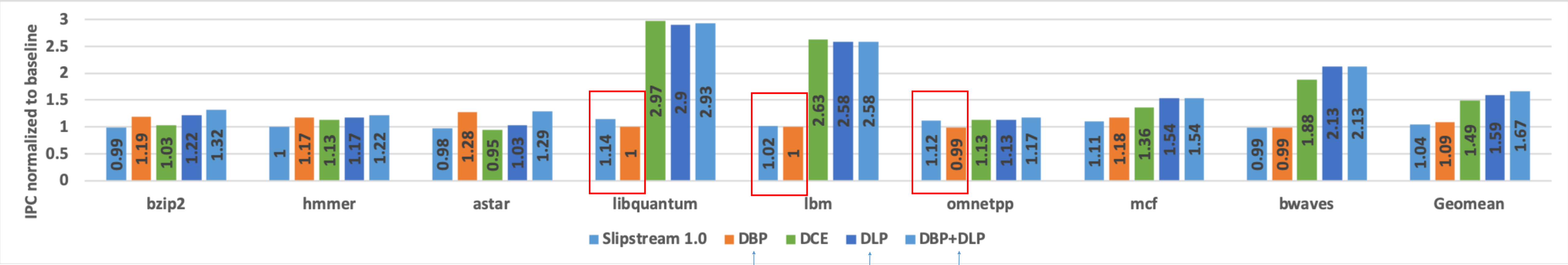


Slipstream 2.0, DBP only

Slipstream 2.0, DLP only

Slipstream 2.0, DBP+DLP

Results: DBP vs. Slipstream 1.0 (cont.)

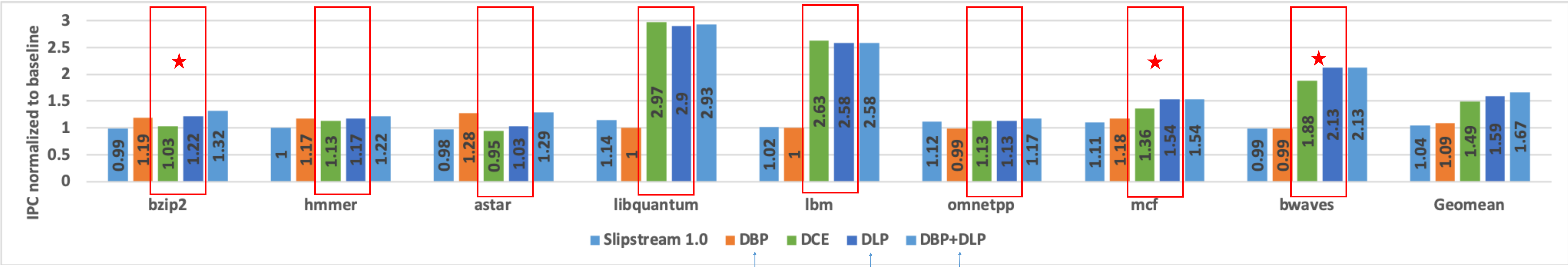


Slipstream 2.0, DBP only

Slipstream 2.0, DLP only

Slipstream 2.0, DBP+DLP

Results: DLP vs. DCE

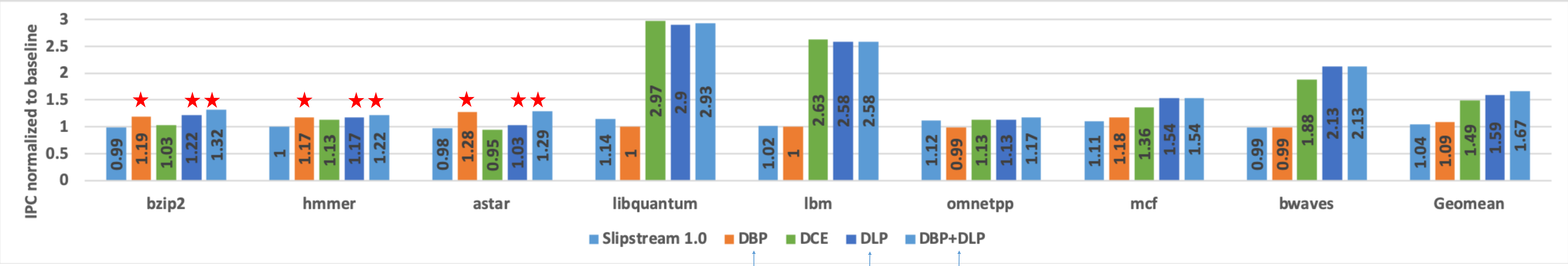


Slipstream 2.0, DBP only

Slipstream 2.0, DLP only

Slipstream 2.0, DBP+DLP

Results: DBP+DLP

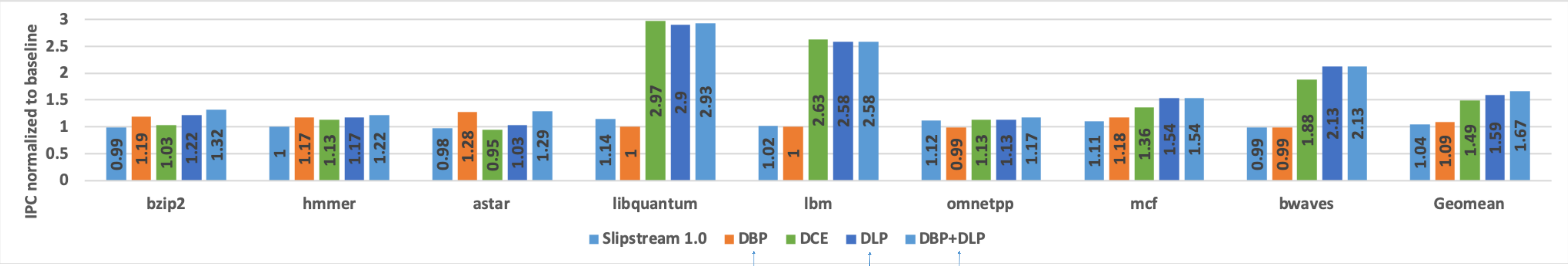


Slipstream 2.0, DBP only

Slipstream 2.0, DLP only

Slipstream 2.0, DBP+DLP

Results: summary



Slipstream 2.0, DBP only

Slipstream 2.0, DLP only

Slipstream 2.0, DBP+DLP

- Slipstream 2.0 (DBP+DLP) gives geomean speedups of 67%, 60%, and 12% over baseline, Slipstream 1.0, and DCE

Summary: Slipstream Processor 2.0

- Remove *forward control-flow slices* of delinquent branches and loads
- Overcomes performance limitations of Slipstream and DCE
- Two firsts:
  - Leader-follower-style branch pre-execution without relying on confident instr. removal
  - Tolerate cache-missed loads that feed mispredicted branches

- Meets all four criteria
  1. Leader-follower style pre-execution
  2. Fully automated using only hardware
  3. Targets both branches and loads
  4. Effective at that which is targeted (improves upon Slipstream and DCE)

- Microarch. turbo-boost: Auto-enable/disable A-stream based on profitability

Future Work

- Need solutions for non-pre-executable delinquent branches
    1. Self-dependent delinquent branches are very serializing
    2. Delinquent branches that are individually pre-executable, but not actually pre-executed due to being in the forward control-flow slice of another delinquent branch