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

This work was funded by the NSF/Intel Partnership on Foundational Microarchitecture Research (FoMR) (NSF grant no. CCF-1823517 and matching Intel grant) and other Intel grants.
Objective

• Delinquent branches and loads limit single-thread performance

• 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, and restarted on a wayward branch.

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

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Slipstream</th>
<th>DCE</th>
<th>DLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Leader-follower style pre-execution</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2. Fully automated using only hardware</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>3. Targets both branches and loads</td>
<td>no (branches)</td>
<td>no (loads)</td>
<td>yes</td>
</tr>
<tr>
<td>4. Effective at that which is targeted</td>
<td>no pre-exec. w/out conf. instr. removal</td>
<td>can’t tolerate miss -&gt; br. misp.</td>
<td>see others</td>
</tr>
</tbody>
</table>
Limitations of Prior Work

• **Slipstream**
  - Remove backward slices of confident branches in A-stream to pre-execute unconfident branches
  - *Ineffective for phases dominated by hard-to-predict branches, when branch pre-execution most needed*

• **DCE**
  - 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*
Slipstream Processor 2.0

- Remove *forward control-flow slices* of delinquent branches and loads
  - Control-dependent (CD) region of the delinquent branch
  - Other branches that are control-independent data-dependent (CIDD) with respect to the delinquent branch or load ("branch set"), and their CD regions

1. Leader-follower-style branch pre-execution without relying on confident instr. removal
2. Tolerate cache-missed loads that feed mispredicted branches

Delinq. Branch Pre-execution (DBP)

Delinq. Load Prefetching (DLP)
• Slipstream 2.0 (DBP+DLP) gives geomean speedups of 67%, 60%, and 12% over baseline, Slipstream 1.0, and DCE