A Low Energy Set-Associative I-Cache with Extended BTB

K. Inoue, V. Moshnyaga, and K. Murakami

Introduction

Increase in cache size

Power consumed in on-chip caches

DEC 21164 CPU*  StrongARM SA-110 CPU*  Bipolar ECL CPU**

25%  43%  50%

* Kamble et. al., “Analytical energy Dissipation Models for Low Power Caches”, ISLPED ’97
**Problem of Conventional Caches**

<table>
<thead>
<tr>
<th>Cycle 1</th>
<th>Cycle 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conventional Cache</strong></td>
<td><strong>Phased Cache</strong></td>
</tr>
<tr>
<td><img src="image1" alt="Diagram of Conventional Cache" /></td>
<td><img src="image2" alt="Diagram of Phased Cache" /></td>
</tr>
<tr>
<td>Parallel search strategy produces unnecessary way activation!</td>
<td>Low energy consumption but Slow access</td>
</tr>
<tr>
<td>First access but High energy consumption</td>
<td></td>
</tr>
</tbody>
</table>

**Our Proposal**

**History-Based Tag-Comparison I-Cache**

😊 Attempts to reduce cache-access energy without performance degradation
😊 Reuses tag-check results to eliminate unnecessary way activation
😊 Can achieve 62% of energy reduction with only 0.2% of performance degradation
**Conventional Tag-Check Scheme**

Cache is updated only when replacement occurs!

- A loaded data stays at the same location at least until the next cache-miss takes place

Programs include a lot of loops!

- A number of instructions are executed repetitively

**History-Based Tag-Comparison (HBTC) Scheme**

Attempts to reuse tag-check results produced before during a cache-miss interval!

- The target instruction has been referenced before, and
- No cache miss has occurred since the previous reference.
Concept of the HBTC Cache

1. Execute an instruction \( A \) at time \( T \)
   - Perform tag check
   - Save the tag-check result into extended BTB

   ![Index Diagram]

   [way2] is the Hit-way!

2. If a cache miss occurs, then we invalidate all the stored tag-check results

3. Execute the instruction \( A \) at time \( T+X \)
   - Reuse the tag-check result to activate only the hit-way's data sub-array

   ![Index Diagram]

   [way2] is the Hit-way!

Conventional VS. Phased VS. HBTC

<table>
<thead>
<tr>
<th></th>
<th>Conventional</th>
<th>Phased</th>
<th>HBTC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cache Hit</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reuse</td>
</tr>
<tr>
<td><strong>Cache Miss</strong></td>
<td></td>
<td></td>
<td>No Reuse</td>
</tr>
</tbody>
</table>
**HBTC SA I-$ Architecture**

- **PBAreg**: Branch Inst. Addr., Pred. Result
- **BTB (Branch Target Buffer)**: Branch-Inst. Addr., Target Addr.
- **WP Table**
  - Taken
  - Not Taken

**Mode Controller**

**I-Cache**

**WP Recode Reg.**: Address for writing

**Tag Check Result**

**WP Reg.**: WP Reg., WP Table

**Entry of the WP Table**
- valid flag
- n of way pointers

**HBTC I-$ Operation**

**Normal Mode (NM)**: w/ Tag checks
**Omitting Mode (OM)**: w/o Tag checks (Reuse)
**Tracing Mode (TM)**: w/ Tag checks
  - (tag-check results are preserved into the WPRreg, and are stored into the WP-table on the next BTB hit)

**Mode Transition**

- **Valid**
  - BTB Hit
  - OM
  - NM
  - TM

- **Invalid**
  - PC and Pred.-result
  - PBAreg

**GOtoNM**
- I-Cache miss or
- BTB replacement or
- RAS access or
- Branch misprediction

**All WPs are invalidated!**
**HBTC I-$ Operation Example**

Mode Transition:
- Valid
- BTB Hit
- GOtoNM
- Invalid

**PC**
- PBAreg
- Inst. Addr. A
- Branch Target Buffer
- Inst. Addr. B
- Target Addr.
- Pred. (T or N)

**WP Table**
- 0 1 2 3
- 4-way I-Cache

**Mode Controller**
- PC
- From I-Cache

**WB reg**
- 0 1 2 3
- 4-way I-Cache

**Mode**
- Controller

**Taken**
- TN

**From**
- I-Cache

**WB reg**
- 0 1 2 3
- 4-way I-Cache

**Mode**
- Controller

**Taken**
- TN

**From**
- I-Cache
**HBTC I-$ Operation Example**

Mode Transition

- **OM**: Valid
- **BTB Hit**: From I-Cache
- **GotoNM**: GOtoNM
- **TM**: Invalid
- **NM**: GOtoNM

### Branch Target Buffer


### WP Table


### Target Addr.

- **Valid**: Taken
- **Invalid**: Taken

### Mode Controller

- **Pred. (T or N)**: Taken
- **From I-Cache**: Taken

### Register

- **PBAreg**: Taken
- **WPReg**: Taken
- **WPRreg**: Taken
- **WPreg**: Taken

### PC and Branch Prediction

- **PC**: Taken
- **Branch Prediction Result**: Taken

**NO valid WPs are detected!**

---

**HBTC I-$ Operation Example**

Mode Transition

- **OM**: Valid
- **BTB Hit**: From I-Cache
- **GotoNM**: GOtoNM
- **TM**: Invalid
- **NM**: GOtoNM

### Branch Target Buffer


### WP Table


### Target Addr.

- **Valid**: Taken
- **Invalid**: Taken

### Mode Controller

- **Pred. (T or N)**: Taken
- **From I-Cache**: Taken

### Register

- **PBAreg**: Taken
- **WPReg**: Taken
- **WPRreg**: Taken
- **WPreg**: Taken

### PC and Branch Prediction

- **PC**: Taken
- **Branch Prediction Result**: Taken

**NO valid WPs are detected!**
**HBTC I-$ Operation Example**

Tag-Comparison result is stored into the WPRreg!

**Mode Transition**
- OM Valid \(\rightarrow\) BTB Hit (NM \(\rightarrow\) GOtoNM)
- Invalid \(\rightarrow\) TM (NM \(\rightarrow\) GOtoNM)

**Conventional Accesses!**
- 4-way I-Cache

**Mode Controller**
- WPRreg
- WP Table
- PBAreg
- Inst. Addr. A
- Target Addr.
- Branch Target Buffer
- Pred. (T or N) → T
- Inst. Addr. B
- Target Addr.
- WPreg
- WPRreg
- Inst. Addr. A
- Target Addr.
- Inst. Addr. B
- Target Addr.
- Pred. (T or N) → T

**HBTC I-$ Operation Example**

Tag-Comparison result is stored into the WPRreg!

**Mode Transition**
- OM Valid \(\rightarrow\) BTB Hit (NM \(\rightarrow\) GOtoNM)
- Invalid \(\rightarrow\) TM (NM \(\rightarrow\) GOtoNM)

**Conventional Accesses!**
- 4-way I-Cache

**Mode Controller**
- WPRreg
- WP Table
- PBAreg
- Inst. Addr. A
- Target Addr.
- Branch Target Buffer
- Pred. (T or N) → T
- Inst. Addr. B
- Target Addr.
- WPreg
- WPRreg
- Inst. Addr. A
- Target Addr.
- Inst. Addr. B
- Target Addr.
- Pred. (T or N) → T
HBTC I-$ Operation Example

Tag-Comparison result is stored into the WPRreg!

The WPRreg is stored into the WP-Table entry pointed by the PBAreg!

Mode Transition

OM  Valid  BTB Hit

GotoNM

Invalid

NM  GotoNM

TM

Conventional Accesses!

4-way I-Cache

0 1 2 3

Pred. (T or N) →

Branch Target Buffer

PBAreg

A  T

WP Table

T  N

WPRreg

From

I-Cache

Branch Target Buffer

WPRreg

A  T

From

I-Cache

Branch Target Buffer

WPRreg

B  T

From

I-Cache

Branch Target Buffer

WPRreg

B  T

From

I-Cache

Branch Target Buffer

WPRreg

B  T

From

I-Cache

Branch Target Buffer

WPRreg

B  T

From

I-Cache
**HBTC I-$ Operation Example**

Mode Transition

- **OM**: Valid
  - BTB Hit
  - GotoNM
  - TM

- **NM**: GotoNM

- **Valid Target Addr.**
- **Target Addr.**
- **Branch Target Buffer**
- **PBAreg**
- **WP Table**
- **WPRreg**

**Controller**
- **TNA**
- **From I-Cache**
- **4-way I-Cache**

**Taken**
- **0 1 2 3**

**Mode Controller**

**Valid WPs are detected!**
**HBTC I-$$ Operation Example**

**Mode Transition**
- **OM**: Valid
- **BTB Hit**: 
  - **NM**: GOtoNM
  - **TM**: GOtoNM

**Tag-Comparison Reuse**
- 4-way I-Cache
- **From I-Cache**: WPreg
- **Reusable**: 0 1 2 3

**PC**
- **Inst. Addr. A**
- **Inst. Addr. B**
- **Pred. (T or N)**
- **Mode Controller**
- **WP Table**
- **WPRreg**
- **PBAreg**
**HBTC I-$ Operation Example**

**Mode Transition**
- **Valid**: BTB Hit
- **Invalid**: GOtoNM

**Tag-Comparison**
- **Reuse**: 4-way I-Cache

**Mode Controller**
- **Pred. (T or N)**
- **From**: I-Cache
- **WPreg**
- **WP Table**
- **Inst. Addr. A**: T
- **Inst. Addr. B**: N

**Re-use**
- **0 1 2 3**

No valid WPs in the WPreg!

**PC**
- **PBAreg**
- **WPRreg**
- **Target Addr.**
**HBTC I-$ Operation Example**

**Mode Transition**
- **OM** → **Valid**
- **BTB Hit**
- **OM** → **Invalid**
- **GotoNM**
- **GotoNM**

**Target Addr.**
- **PBAreg**
- **WP Table**
- **Conventional Accesses!**
- **4-way I-Cache**
- **PC**
- **Advantages and Disadvantages**

**Advantages and Disadvantages**

- **Eliminate unnecessary energy consumption w/o performance degradation (during OM)!**
- **BTB energy overhead due to WP-table read-accesses**
- **BTB access conflict for invalidating all WPs (causes 1 stall cycle)**
- **BTB access conflict to record WPs (causes 1 stall cycle)**
Evaluation – Environment –

- OOO simulation by SimpleScalar
  - 16 KB 4-way I-cache (32 B line size)
  - For others, default parameters were used
- Cache Energy Model based on [Kamble97]
  (including the WP-table read-energy overhead)
- Assume that the BTB is accessed only when branch or jump instructions are executed (instructions are pre-decoded)

Benchmark Programs

- SPECint95
  - 099.go, 124.m88ksim, 126.gcc, 129.compress, 130.li, 132.ijpeg
- SPECfp95
  - 102.swim
- Mediabench
  - mpeg2encode, mpeg2decode, adpcm_enc, adpcm_dec


Evaluation – Energy and Performance –

- # of WPs = 4

- 62% of Ecache reduction with 0.2% of Exe. Time increase
- Even if in the worst case, about 20% of Ecache reduction
If the penalty is equal to or smaller than 4 clock cycles, the performance overhead is trivial.

The performance overhead grows after the penalty is more than 4 clock cycles.

- Increasing the number of WPs makes it possible to reuse many tag-check results
- But, it produces BTB access energy overhead
**Evaluation**

– Effect of Cache Associativity –

**Conclusions**

**History-Based Tag-Comparison Instruction Cache**

1. Recodes tag-check results generated by the I-cache into the extended BTB
2. Attempts to reuse them in order to eliminate unnecessary way activation
3. Achieves 62% of I-cache energy reduction with only 0.2% of performance degradation!

**Future work**

- Analyze energy consumption based on real chip design.
Buck Up Slides
(History-based Tag-Comparison Cache)

Evaluation
– Comparison with IS Approach –

Interline Sequential approach
History-Based Look-up Cache
Combination of IS and HBL

Normalized Tag-Compare Count

<table>
<thead>
<tr>
<th>Program</th>
<th>IS Approach</th>
<th>HBL Approach</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>099.go</td>
<td>0.8</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>126.gcc</td>
<td>0.6</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>130.li</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>102.swim</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>adpcm(d)</td>
<td>0.1</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>mpeg2(d)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>124.m88ksim</td>
<td>0.8</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>129.comp.</td>
<td>0.6</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>132.jpg</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>adpcm(e)</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>mpeg2(e)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
**Evaluation**

-- Effects of Cache Associativity --

<table>
<thead>
<tr>
<th>Associativity</th>
<th>099.go</th>
<th>126.gcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Eothers</td>
<td>Eothers</td>
</tr>
<tr>
<td>2</td>
<td>Etag</td>
<td>Etag</td>
</tr>
<tr>
<td>4</td>
<td>Edata,bl</td>
<td>Edata,bl</td>
</tr>
<tr>
<td>8</td>
<td>Edata,prectl</td>
<td>Edata,prectl</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* M.B. Kamble and K. Ghose, "Energy-Efficiency of VLSI Caches: A Comparative Study," 10th Int. Conf. on VLSI Design
**Evaluation**

– Effects of Cache Associativity –

![Bar chart comparing conventional and HBL Cache energy consumption](132.ijpeg)

**Evaluation**

– Effects of Cache Associativity –

![Bar chart comparing conventional and HBL Cache energy consumption](mpeg2decode)

---


---
Evaluation
– Effects of # of WPs –

w/ Pre-Decoding
(BTB access occurs only at branch, or jump, executions)

Evaluation
– Effects of # of WPs –

w/o Pre-Decoding
(BTB access occurs for all instructions)
Evaluation

– Effect of WP invalidation penalty –

- If the penalty is equal to or smaller than 4 clock cycles, the performance overhead is trivial.
- The performance overhead grows after the penalty is more than 4 clock cycles.