A Software Controllable Variable Line Size Cache
Exploiting High On-Chip Memory Bandwidth for Low Power Embedded SoCs

Takatsugu Ono†, Koji Inoue †, Kazuaki Murakami†
and Kenji Yoshida‡

†Kyushu University
‡ System Fabrications Technologies, Inc.
System-in-Silicon Architecture (1/2)

• Features
  – An application ASIC and a SiS-DRAM are mounted onto a silicon interposer
  – Each chip is connected to the silicon interposer by eutectic solder bumps

• Advantages
  – High memory bandwidth is easily achieved by increasing the # of on-silicon communication wires
  – We can reduce the energy consumption for DRAM access because no activations of high capacitive I/O pads are required
System-in-Silicon Architecture (2/2)
Motivation (1/2)

• Large access latency of SiS-DRAM is still an important issue

• To solve this problem, we employ a small cache memory, called *SiS-cache*
  – This is not cache memory included in the microprocessor core

• It is still required to achieve more energy reduction, particularly the energy for SiS-DRAM accesses
  – The load capacitance of SiS-DRAM is large
  – It may dominate the total energy consumption
Motivation (2/2)

• SiS-cache can employ larger cache line size in order to expect the effect of prefetching

• But... the larger cache line size also requires a number of SiS-DRAM bank activations

A large amount of energy is required!
Software-Controllable Variable Line-Size cache (1/3)

• We propose *SC-VLS cache*
  – It attempts to optimize the amount of data to be transferred between SiS-cache and SiS-DRAM

• When a target application program does not require high memory bandwidth
  \( \Rightarrow \) SC-VLS cache reduces the cache line size
Software-Controllable Variable Line-Size cache (2/3)

• Features
  – SC-VLS cache does not requires any hardware monitor to decide the line size

• Advantages
  – SC-VLS cache reduces energy consumption with trivial hardware overhead
Software-Controllable Variable Line-Size cache (3/3)

• *Adequate line size analysis*
  – Before an application program is executed, we analyze an adequate line size of each function

• *Code generation*
  – Line size change instructions are inserted into start of functions in original program code
  – The instruction sets status register to indicate an adequate line size
## SC-VLS Cache Architecture

### Diagram Description

- **Processor**: The input address is sent to the Processor block.
- **Address**: The address is then split into **Tag**, **Index**, and **Offset**.
- **Status Reg.**: A Status Register is used to determine the hit/miss status.
- **Valid bit** and **Tag** are inputs to the **MUX** block.
- **Minimum line size** is used to set the adequate line size.
- **Hit / Miss** output from the MUX block.
- **32B** blocks represent the cache lines.
- **Main Memory**: The output is directed to the Main Memory block.

### Table

<table>
<thead>
<tr>
<th>Tag</th>
<th>Index</th>
<th>Offset</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Set an adequate line size*
Adequate Line Size Analysis

• We execute cache simulation with each line size independently to determine an adequate line size

1. An average cache miss rate of each function is calculated
2. We compare the average cache miss rates with all line size candidates
3. A line size which the cache miss rate is the smallest is determined as an adequate line size
Adequate Line Size Analysis
- Example -

**Line size = 64B**

<table>
<thead>
<tr>
<th></th>
<th>Misses</th>
<th>Accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>100</td>
<td>2100</td>
</tr>
<tr>
<td>6</td>
<td>200</td>
<td>5100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>2200</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>2100</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>5100</td>
</tr>
</tbody>
</table>

**foo1()**

ave. \( MR_{64Bfoo1} = \frac{10}{200} = 5.0\% \)

**foo2()**

ave. \( MR_{64Bfoo2} = \frac{8}{400} = 2.0\% \)

**foo3()**

ave. \( MR_{64Bfoo3} = \frac{2}{100} = 2.0\% \)

\( MR_{64B} \approx 2.9\% \)

**Line size = 32B**

<table>
<thead>
<tr>
<th></th>
<th>Misses</th>
<th>Accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>1100</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>2200</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>1100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>14200</td>
</tr>
</tbody>
</table>

**foo1()**

ave. \( MR_{32Bfoo1} = \frac{4}{200} = 2.0\% \)

**foo2()**

ave. \( MR_{32Bfoo2} = \frac{16}{400} = 4.0\% \)

**foo3()**

ave. \( MR_{32Bfoo3} = \frac{1}{100} = 1.0\% \)

\( MR_{32B} = 3.0\% \)

**Line size = adequate**

<table>
<thead>
<tr>
<th></th>
<th>Misses</th>
<th>Accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>32B</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>64B</td>
<td>6</td>
<td>200</td>
</tr>
<tr>
<td>32B</td>
<td>32B</td>
<td>32B</td>
</tr>
<tr>
<td>64B</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>5100</td>
</tr>
</tbody>
</table>

**foo1()**

adequate line size \( foo1() = 32B \)

**foo2()**

adequate line size \( foo2() = 64B \)

**foo3()**

adequate line size \( foo3() = 32B \)

\( MR_{adequate} \approx 1.9\% \)
Energy Model

• Total SiS-DRAM energy consumption ($E_{mem}$)

The total number of accesses to SiS-DRAM

$$E_{mem} = \sum_{i=1}^{\text{access}} \left( E_{Bacnk} \times \left[ \frac{\text{LineSize}_{SCVLS_i}}{\text{LineSize}_{Bank}} \right] \right)$$

Energy consumption per a bank access

SiS-cache line size

The bandwidth of SiS-DRAM bank
Experimental Setup

• Measuring energy consumption and SiS-cache miss rates

• The SC-VLS cache can dynamically choose four line sizes;
  – 32B, 64B, 128B and 256B

• Benchmark programs and input data sets

<table>
<thead>
<tr>
<th></th>
<th>Analysis phase</th>
<th>Execution phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiBench</td>
<td>small</td>
<td>large</td>
</tr>
<tr>
<td>H.264</td>
<td>akiyo (QCIF)</td>
<td>salesman (QCIF)</td>
</tr>
</tbody>
</table>
Energy & Miss Rate

![Graph showing normalized energy and miss rate for various benchmark programs]

- Benchmark programs:
  - jpeg
  - qsort
  - rijndael
  - sha
  - susan
  - H.264

- Normalized energy and miss rate are plotted against each benchmark program.
Performance Overhead

overhead = \frac{\text{change instructions}}{\text{all instructions}} \times 100

<table>
<thead>
<tr>
<th>Benchmark Programs</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpeg</td>
<td>0.180</td>
</tr>
<tr>
<td>qsort</td>
<td>0.911</td>
</tr>
<tr>
<td>rijndael</td>
<td>0.208</td>
</tr>
<tr>
<td>sha</td>
<td>0.038</td>
</tr>
<tr>
<td>susan</td>
<td>0.244</td>
</tr>
<tr>
<td>H.264</td>
<td>1.67</td>
</tr>
</tbody>
</table>
Conclusions

• Summary
  – We have proposed SC-VLS cache to reduce on-chip DRAM energy consumption
  – The SiS-cache line sizes are dynamically changed during a program execution
  – The SiS-cache reduces on-chip DRAM energy consumptions up to 50%

• Future work
  – We plant to reduce an adequate line size analysis costs