Energy-Aware Computation Duplication for Improving Reliability in Embedded Chip Multiprocessors

G. Chen, M. Kandemir, and F. Li
Department of Computer Science and Engineering
Pennsylvania State University
Introduction

• Advantages of chip multiprocessors (CMP)
  – Easy for verification
  – Appropriate for high-level code parallelism
  – Power efficient

• Transient errors
  – Cross-coupling, ground bounce, external terrestrial radiations…
  – Technology scaling and power-saving techniques increases embedded systems’ vulnerability to transient errors

• Our goal: Utilizing on-chip parallelism for best tradeoffs between performance, power, and reliability
Chip Multiprocessor Architecture

Off-chip memory
Loop Parallelization

Loop iteration space

Each processor gets a portion of iterations to execute
Adaptive Loop Parallelization

- Add more processors can degrade performance of a loop

4 Proc

- Assign the optimum number of processors to each loop

8 Proc

\[
\text{for } i=1..1024 \\
\text{for } j=1..1024 \\
\text{...}
\]
# Processor Number for Optimum Performance

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
<th>N5</th>
<th>N6</th>
<th>N7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3step-log</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adi</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>btrix</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>eflux</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>full-search</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n-real-updates</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tsf</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adaptive Loop Parallelization

Loop iteration space

How to utilize idle processors?
Optimizing Power Consumption

Loop iteration space

I₁ \rightarrow P₁ \quad I₂ \rightarrow P₂ \quad \ldots \ldots \quad Iₘ \rightarrow Pₘ \quad Pₘ₊₁ \quad Pₘ₊₂ \quad Pₘ₊₃ \quad \ldots \ldots \quad Pₙ

Idle processors are put into low-power mode to save leakage energy
Improving Reliability

Loop iteration space

\[ I_1 \quad I_2 \quad I_m \quad P_1 \quad P_2 \quad \ldots \quad P_m \quad P_{m+1} \quad P_{m+2} \quad P_{m+3} \quad \ldots \quad P_n \]
Improving Reliability

The local iteration space of r processors (P_1 through P_r) are duplicated and executed on processors P_{m+1} through P_{m+r}
Improving Reliability

Loop iteration space

\[ P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_m \rightarrow P_{m+1} \rightarrow \ldots \rightarrow P_{m+r} \rightarrow P_{m+r+1} \rightarrow P_n \]

\( r/m \) is called the duplication percentage. Different duplication percentages represent different tradeoff points between performance, energy, and reliability.
Computation Duplication

Primary execution  Duplicate execution

An important decision to make is how to compare the two executions
A lock-step approach can generate a lot of communication activities and it also requires many comparison instructions. Therefore, it is not desirable for embedded CMP.
Checksum-based Execution Comparison

- **write**
- **other**

**Primary execution**

**Duplicate execution**

![Diagram showing checksum-based execution comparison with time axis and arrows indicating checksums and execution paths.](image)
Checksum-based Execution Comparison

- write
- other

Primary execution

Duplicate execution

Time

sync & compare
Example Code

Original loop body

\[
A[i] = C[i+1] \times D[i] + E[i]; \\
B[i] = C[i-1] - D[i-1];
\]

Checksum for each processor

Primary execution

\[
A[i] = C[i+1] \times D[i] + E[i]; \\
CHECK[p] = A[i]; \\
B[i] = C[i-1] - D[i-1]; \\
CHECK[p] = B[i];
\]

Duplicate execution

\[
CHECK[p] = A[i]; \\
C[i+1] \times D[i] + E[i]; \\
CHECK[p] = B[i]; \\
C[i-1] - D[i-1];
\]
Shared Data Problem

Primary execution

Duplicate execution

Time

Desired execution order
Shared Data Problem

Primary execution

Duplicate execution

Time

Undesirable execution sequence

Race condition might happen if an array element is both read and written in the iteration space
Solution to Shared Data Problem

Primary execution

Duplicate execution

Time
Example Code

Original loop body

A[i]=B[i]+C[i];
B[i]=C[i]-10;

Primary execution

A[i]=B[i]+C[i];
CHECK[prid]+=A[i];
B[i]=C[i]-10;
CHECK[prid]+=B[i];

Duplicate execution

CHECK[prid]+=B'[i]+C[i];
CHECK[prid]+=C[i]-10;
Experimental Setup

• Simics for CMP simulator
• 8 processors
• 8KB L1 I-cache, 8KB L1 D-cache, 1MB L2 cache
• Seven benchmarks from Perfect Club, Livermore, DSPStone
Energy-Delay-Fallibility (EDF) Product

- EDF = energy * (execution cycles) / Fallibility
  - Fallibility = 1/reliability
  - Reliability is the percentage of primary processors that have duplicates
- EDF is a good metric for evaluating the tradeoffs between energy, performance, and reliability
  - We want EDF to be as small as possible
Performance Overhead

- Less than 2% overhead when averaged over all the benchmark codes
- Performance overhead breakdown
As percentage of duplicates increases, EDF increases since reliability increases.

As we use more processors for duplication, the benefits coming from increased reliability can be offset by increased energy consumption.
Conclusion

- On-chip parallelism of CMP can be used for improving reliability
- Single metric based compilation strategies are not sufficient for current embedded systems, where multiple constraints are important
- EDF can be used for evaluate the tradeoffs of power, performance, and reliability