Adaptive Execution on 3D Microprocessors

Koji Inoue
Kyushu University
Outline

• Why 3D?
• Will 3D always work well?
• Support Adaptive Execution!
  – Memory Hierarchy Run-time Optimization
• Conclusions
Outline

• Why 3D?
• Will 3D always work well?
• Support Adaptive Execution!
  – Memory Hierarchy Run-time Optimization
• Conclusions
From 2D to 3D! (not only TV)

- Stack Multiple Dies
- Connect Dies with Through Silicon Vias
Chip Implementation Examples from ISSCC’09

- Image Sensors
- SRAM for SoCs
- DRAM
- Multi-core + SRAM connected with wireless TSVs

K. Niitsu et al., “An Inductive-Coupling Link for 3D Integration of a 90nm CMOS Processor and a 65nm CMOS SRAM,” ISSCC’09.
Why 3D? (1/2)

- Wire Length Reduction
  - Replace long, high capacitance wires by TSVs
  - Low Latency, Low Energy
- Small footprint
Why 3D? (2/2)

• Integration
  – From “Off-Chip” to “On-Chip”
  – Improved Communication
    • Low Latency, High Bandwidth, and Low Energy
  – Heterogeneous Integration
    • E.g. Emerging Devices
Outline

• Why 3D?
• Will 3D always work well?
• Support Adaptive Execution!
  – Memory Hierarchy Run-time Optimization
• Conclusions
Importance of On-Chip Caches

• Memory-Wall Problem
  – Memory bandwidth does not scale with the # of cores
  – Growing speed gap between processor cores and DRAMs
  – So, Becomes more serious

• Let’s increase on-chip cache capacity, but...
  – Requires large chip area
Will 3D always work well?  
“Stacking a DRAM Cache”

\[
AMAT = HT_{L1} + MR_{L1} \times (HT_{L2} + MR_{L2} \times MMAT)
\]

**Ave. Memory Acc. Time**

| Impact of DRAM Stacking | → | → | ↑ | ↓ | → |

<table>
<thead>
<tr>
<th>Core</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>2MB Cache</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>32MB DRAM Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
</tr>
<tr>
<td>Tag RAM</td>
</tr>
</tbody>
</table>

**Will 3D always work well?**

“Stacking a DRAM Cache”

\[
AMAT = HT_{L1} + MR_{L1} \times (HT_{L2} + MR_{L2} \times MMAT)
\]
Cache-Size Sensitivity Varies among Programs!
Profit = \frac{\text{MRL2}_2 \_REDUCTION \times \text{MMAT}}{\text{HTL2}_2 \_OVERHEAD}
Appropriate Cache Size Varies within Programs!

The lower, the better

- 2MB(12cc)
- 32MB(60cc)

Time Interval (100K L2 Accesses / Interval)
Outline

• Why 3D?
• Will 3D always work well?
• Adaptive Execution!
  – Memory Hierarchy Run-time Optimization
• Conclusions
Will 3D always work well?
“Stacking a DRAM Cache”

\[
AMAT = HT_{L1} + MR_{L1} \times (HT_{L2} + MR_{L2} \times MMAT)
\]

Ave. Memory Acc. Time

| Impact of DRAM Stacking | → | → | ↑ | ? | → |

32MB DRAM Cache
SRAM/DRAM Hybrid Cache Architecture

• Support Two Operation Modes
  – High-Speed, Small Cache Mode (or SRAM Cache Mode)
  – Low-Speed, Large Cache Mode (or DRAM Cache Mode)

• Adapt to variation of application behavior
Microarchitecture (1/2)
Microarchitecture (2/2)

SARM (Size: Cs, Block: Ls, Asso. Ws)

DARM (Size: d Block: Ld, Asso. Wd)

Data (SRAM)

Hit/Miss (SRAM)

Data (DRAM)

Hit/Miss (DRAM)

\[ I_s = \log \frac{C_s}{L_s \cdot W_s} \]

\[ I_d = \log \frac{C_d}{L_d \cdot W_d} \]

Assume Ld == Ls == 64B
How to Adapt

• Static Approach
  – Optimizes at program level
  – Does not change it during execution
  – Needs a static analysis

• Dynamic Approach
  – Optimizes at interval level (or phase level)
  – Needs a run-time profiling
Run-Time Mode Selection

- Divide Program Execution into “epochs”, e.g. 200K L2 Misses
- Predict an Appropriate Operation Mode for Next Epoch
- On SRAM mode, a small tag RAM which stores sampled tags is used to predict DRAM mode miss rates

\[
\text{if } MR_{L2SRAM} - MR_{L2DRAM} > \frac{HT_{L2DRAM} - HT_{L2SRAM} + AveOverhead}{MMAT} \\
\text{then transit from SRAM mode to DRAM mode!}
\]
Experimental Set Up

- Processor: In-Order
- Benchmarks: SPEC CPU 2000, Splash2

**SRAM Cache Mode**

- **Core**
- **L1 Cache**
- **L2 Cache**

**DRAM Cache Mode**

- **Core**
- **L1 Cache**
- **L2 Cache**

**Main Memory**

- **Size:** 32KB
- **Access Time:** 2 clock cycles
- **Size:** 2MB
- **Access Time:** 6 clock cycles
- **Size:** 32MB
- **Access Time:** 28 clock cycles

**Access Time:** 181 clock cycles
Results

Benchmark Program

Normalized AMAT

Accuracy of Mode Selection

2D-SRAM DRAM-STACK

ammp art bzip2 mcf mgrid swim twolf

mgrid swim twolf Cholesky FFT FMM LU Ocean
Results

The chart shows the normalized AMAT (acceleration metric) for various benchmark programs. The programs include ammp, art, bzip2, mcf, mgrid, swim, twolf, Cholesky, FFT, FMM, LU, and Ocean. The X-axis represents the benchmark program, while the Y-axis shows the normalized AMAT.

The chart highlights three different memory architectures: 2D-SRAM, DRAM-STACK, and HYBRID. Each memory architecture is represented by a different color, allowing for easy comparison.

The bar chart indicates that the HYBRID memory architecture generally yields a lower normalized AMAT compared to the other two architectures, suggesting improved performance or efficiency. The specific metrics for each program are not clearly visible due to the resolution of the image, but the chart provides a clear visual comparison of performance across different memory configurations.
Conclusions

• The 3D solution is one of the most promising ways to achieve...
  – High performance
  – Low energy
• It does not ALWAYS work well!
• Run-time adaptive execution by considering memory access behavior
Acknowledgement

• This research was supported in part by New Energy and Industrial Technology Development Organization