An Evaluation of an Energy Efficient Many-Core SoC with Parallelized Face Detection

Hiroyuki Usui, Jun Tanabe, Toru Sano, Hui Xu, and Takashi Miyamori

Toshiba Corporation, Kawasaki, Japan
Executive Summary

- Future architecture will have many cores
- **A key challenge**: How to efficiently use them?
- We evaluated techniques to accelerate one type of important application (face detection)
- Performance scales up to 64 cores
- Energy efficiency is 20x better than desktop CPU
Outline

• Introduction
• Face Detection using Joint Haar-Like Features
• Architecture of Energy Efficient Many-Core SoC
• Issues in Implementing Parallelized Face Detection
• Implementation and Evaluation of Parallelized Face Detection
  – On the Single Cluster
  – On the Dual Cluster
• Conclusion
Outline

• Introduction
• Face Detection using Joint Haar-Like Features
• Architecture of Energy Efficient Many-Core SoC
• Issues in Implementing Parallelized Face Detection
• Implementation and Evaluation of Parallelized Face Detection
  – On the Single Cluster
  – On the Dual Cluster
• Conclusion
Two Key Trends in Embedded Systems

• Trend 1: New applications (e.g. image recognition) need more computing power while keeping low power
• Trend 2: New architecture can enable much more parallelism than before
Two Key Trends in Embedded Systems

• Trend 1: New applications (e.g. image recognition) need more computing power while keeping low power

• Trend 2: New architecture can enable much more parallelism than before
  
  Now: 500GOPS

Heterogeneous Multi-Core

Visconti™2 [ISSCC’12]
Two Key Trends in Embedded Systems

• Trend 1: New applications (e.g. image recognition) need more computing power while keeping low power

• Trend 2: New architecture can enable much more parallelism than before

Now: 500GOPS

Heterogeneous Multi-Core

Future: More than 1TOPS

Heterogeneous Many-Core

Visconti™2 [ISSCC’12]

Toshiba Many-Core [VLSI Sympo. ’12]
Two Key Trends in Embedded Systems

- **Trend 1**: New applications (e.g., image recognition) need more computing power while keeping low power.
- **Trend 2**: New architecture can enable much more parallelism than before.

**Now**: 500GOPS

**Future**: More than 1TOPS

Heterogeneous **Multi-Core**

**Result**: A need for efficient and scalable application performance on many-core.
Power and Performance Target of our Many-Core
Power and Performance Target of our Many-Core

- **Power Consumption [W]**
  - 0.1
  - 1
  - 10
  - 100
  - 1000

- **Performance [GOPS]**
  - GHz cores
  - High Performance
  - Multi & Many-Cores for HPC

- **Intel® 80-Tile**
- **Intel® SOC**
- **Cell Broadband Engine™**
- **Tilera® Tile64**

---

High Performance Multi & Many-Cores for HPC
Power and Performance Target of our Many-Core

Less than 3W is needed for embedded applications

High Performance Multi & Many-Cores for HPC

GHz cores

Intel® SOC

Cell Broadband Engine™

Tilera® Tile64

Intel® 80-Tile

Less than 3W is needed for embedded applications

Power Consumption [W]

Performance [GOPS]
Power and Performance Target of our Many-Core

High Performance Multi & Many-Cores for HPC

Intel® 80-Tile

GHz cores

Intel® SOC

Cell Broadband Engine™

Tilera® Tile64

Less than 3W is needed for embedded applications

Energy Efficient Embedded Multi-Cores

Renesas RP-X

ARM® Cortex™-A5

Toshiba Multi-Core (Visconti™2)
Power and Performance Target of our Many-Core

Less than 3W is needed for embedded applications

Our Target Area

Energy Efficient Embedded Multi-Cores

Toshiba Multi-Core (Visconti™2)

Renesas RP-X

ARM® Cortex™-A5

High Performance Multi & Many-Cores for HPC

Intel® 80-Tile

Cell Broadband Engine™

Intel® SOC

Tilera® Tile64
Power and Performance Target of our Many-Core

Power Consumption [W] vs Performance [GOPS]

- Less than 3W is needed for embedded applications
- Toshiba Energy Efficient Many-Core (64Core)
- High Performance Multi & Many-Cores for HPC
- GHz cores
- Intel® SCC
- Intel® 80-Tile
- Cell Broadband Engine
- Tilera® Tile64
- Toshiba Multi-Core (Visconti™2)
- ARM® Cortex™-A5
- Renesas RP-X
- Energy Efficient Embedded Multi-Cores

Our Target Area
Many-Core Scalability

Performance

The Number of Cores
Many-Core Scalability

Performance

The Number of Cores

Ideal
Many-Core Scalability

Ideal

Actual?

The Number of Cores

Performance
Many-Core Scalability

Can we achieve good performance scaling-up on face detection?
Outline

• Introduction
• **Face Detection using Joint Haar-Like Features**
• **Architecture of Energy Efficient Many-Core SoC**
• **Issues in Implementing Parallelized Face Detection**
• Implementation and Evaluation of Parallelized Face Detection
  – On the Single Cluster
  – On the Dual Cluster
• **Conclusion**
Face Detection
Face Detection

Check if a face exists or not

ROI : Region of Interest

25 pixels

1st ROI

25 pixels
Face Detection

Check if a face exists or not

ROI : Region of Interest
Face Detection

ROI: Region of Interest

Check if a face exists or not

3rd ROI

25 pixels

25 pixels

4 pixels
Face Detection

ROI : Region of Interest

Check if a face exists or not
Face Detection

25 pixels

(N+1)th ROI

2 pixels

ROI: Region of Interest

Check if a face exists or not
Face Detection

ROI: Region of Interest
Joint Haar-Like Features [ICCV ‘05]

- Extension to widely-used Viola and John’s Method [CVPR ‘01] (using Haar-like features)

Haar-like feature : Difference of image intensities between blue and red rectangles.

ROI (Region Of Interest)
Joint Haar-Like Features [ICCV ‘05]

• Extension to widely-used Viola and John’s Method [CVPR ‘01] (using Haar-like features)

Haar-like feature: Difference of image intensities between blue and red rectangles.

Eye is darker than cheek

Compared to each threshold
If greater than: 1, otherwise: 0

1 1 0
Positions of features and tables are learned in advance and stored in the dictionary.
Characteristics of Face Detection

- Face detection for each ROI can be executed in parallel
- There are a lot of ROIs in an image
  - 3M ROIs when image size is 4000x3200

- A lot of coarse grain thread parallelism based on ROIs
  - Overhead of thread scheduling can be minimized

Many-core is good for face detection!
Outline

• Introduction
• Face Detection using Joint Haar-Like Features
• Architecture of Energy Efficient Many-Core SoC
• Issues in Implementing Parallelized Face Detection
• Implementation and Evaluation of Parallelized Face Detection
  – On the Single Cluster
  – On the Dual Cluster
• Conclusion
Chip Micrograph and Features

- **Technology**: 40nm LP Process
- **Interconnect**: 8 metal (Cu)
- **Chip Size**: 15.0mm x 14.0mm
- **Cluster Size**: 7.4mm x 5.7mm
- **Transistors**: 87.5 Million
- **Cluster Frequency**: 333MHz, 1.1V
- **Package**: 1369-pin FCBGA

**Cluster 0**
- 2MB L2 Cache
- DDR3 I/F
- Core

**Cluster 1**
- 2MB L2 Cache
- DDR3 I/F

**Reconfigurable Engines**

**DDR3 I/F**

**L2 Cache**
- Bank 0
- Bank 1
- Bank 2
- Bank 3
Chip Micrograph and Features

- Technology: 40nm LP Process
- Interconnect: 8 metal (Cu)
- Chip Size: 15.0mm x 14.0mm
- Cluster Size: 7.4mm x 5.7mm
- Transistors: 87.5 Million
- Cluster Frequency: 333MHz, 1.1V
- Package: 1369-pin FCBGA
Structure of Many-Core Cluster

- **Tree-based NoC**
  - Leaf nodes: Core
  - Root nodes: L2 cache banks

![Diagram showing tree-based NoC structure with core, L2 cache banks, and cluster control module.](image-url)
Many-Core SoC Architecture

- Image Recognition & Processing Accelerators
  - SRAM 512KB x 4
  - ARM Cortex-A9 X 2
- Reconfigurable Engine x 2
  - Core x 32
  - L2 Cache 2MB
- Many-Core Cluster 0
  - Core x 32
  - L2 Cache 2MB
- Many-Core Cluster 1
  - Core x 32
  - L2 Cache 2MB

- External Interface
- PCIe X 1
- DDR3 Controller X 2
- PCIe X 1

- Video In x 4
- Video Out
- Peripherals

10.7GB/s
Core : Media Processing Block (MPB)

- 3-Way VLIW Processor
- L1 Instruction Cache: 32KB
- L1 Data Cache: 16KB
- 333 MHz

Exploits multi-grain parallelism
- Thread level by many cores
- Instruction level by VLIW architecture
- Data level by SIMD instructions
Outline

• Introduction
• Face Detection using Joint Haar-Like Features
• Architecture of Energy Efficient Many-Core SoC
• **Issues in Implementing Parallelized Face Detection**
• Implementation and Evaluation of Parallelized Face Detection
  – On the Single Cluster
  – On the Dual Cluster
• Conclusion
Issues in implementing parallelized face detection

- **High coarse-grain parallelism:** Good for Parallelization
  - There are enough ROIs to exploit by many cores

- **Imbalanced workload:** Bad for Processor Utilization
  - The workload of an ROI where a face exists is higher than that of an ROI without a face

**Implementation of parallelized face-detection**
- Minimize the number of threads in order to reduce synchronization cost
  - Allocate one thread to one core
  - Find a good thread partitioning with balancing workload of threads
  - Reduce data bandwidth (L1$-L2$ and L2$-DDR3$)
Outline

• Introduction
• Face Detection using Joint Haar-Like Features
• Architecture of Energy Efficient Many-Core SoC
• Issues in Implementing Parallelized Face Detection
• Implementation and Evaluation of Parallelized Face Detection
  – On the Single Cluster
  – On the Dual Cluster
• Conclusion
Implementation on the Single Cluster

• We implemented the face detection with two methods to allocate image to cores
  – Allocating Cyclically
  – Splitting Equally
(1) Allocating Cyclically

This way allocates lines to each core cyclically. Effective in balancing workload.
(2) Splitting Equally

This way divides the image evenly. Effective to reduce data size read by each core.

Image

Height

Height/32

Core0

Core1

Core2

Core31
Images for Evaluation

- High Resolution Images (5.76-12.7Mp) including many faces

<table>
<thead>
<tr>
<th>No.</th>
<th>Resolution</th>
<th>Number of Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4000x1440</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>3000x4082</td>
<td>37</td>
</tr>
<tr>
<td>2</td>
<td>4083x3062</td>
<td>78</td>
</tr>
<tr>
<td>3</td>
<td>4094x3107</td>
<td>148</td>
</tr>
<tr>
<td>4</td>
<td>3568x2568</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>3568x2568</td>
<td>10</td>
</tr>
</tbody>
</table>

Face Detection Result of Image 4
Evaluation Board

I/O and switches for evaluation

Many-Core SoC (Fan-less Cooling)
Relative Performance on Single Cluster

Allocating Cyclically

Splitting Equally

Number of Cores

Relative Performance

1x

2x

4x

8x

16x

32x

1

2

4

8

16

32

img.0

img.1

img.2

img.3

img.4

img.5

ideal
Average Relative Performance on Single Cluster

With Allocating Cyclically, performance scales up to 32 cores
Execution Time of the Fastest and Slowest Cores

Allocating Cyclically

Splitting Equally

Image Number

Time (sec)

Fastest Core

Slowest Core

1.1x

11x
Processor Utilization

- Allocating Cyclically: 90 ~ 95%
- Splitting Equally: 55 ~ 75%

Low processor utilization deteriorates the performance of Splitting Equally
Bandwidth of L2 Cache and DDR3

- **L1-L2 bandwidth is nearly the same**
  - L1 cache is not enough to store ROI line
- **About L2-DDR3, Allocating Cyclically is better**
  - All cores access the small area at the same
Outline

- Introduction
- Face Detection using Joint Haar-Like Features
- Architecture of Energy Efficient Many-Core SoC
- Issues in Implementing Parallelized Face Detection
- Implementation and Evaluation of Parallelized Face Detection
  - On the Single Cluster
  - On the Dual Cluster
- Conclusion
Implementation on Dual Cluster

• Each cluster has its own L2 cache and shares DDR3
  – Because bandwidth is narrower than L1 and L2 cache, reducing bandwidth between L2 cache and DDR3 is important

• We implemented the two ways
  – Allocating Cyclically
  – Bisection
(1) Allocating Cyclically

This way is the same as that of a single cluster

Effective in balancing workload

Core0

Core1

Core2

Core31

Cluster0

ROI

Image

Core0

Core1

Core2

Core31

Cluster1
(2) Bisection

This way divides the image into two blocks

Cluster0

Image

Height/2

Cluster1

Each cluster processes each block
(2) Bisection

This way divides the image into two blocks

Effective to reduce data size read by each cluster

In each block, Allocating Cyclically is used
Performance of Dual Cluster (64 Cores)

- Relative Performance of Dual Cluster (64 Cores)
- Allocation Cyclically
- Bisection
- Ideal (64x) Performance

Image Number: 0, 1, 2, 3, 4, 5

- Image 0: 61x
- Image 1: 42x

55
Performance of Dual Cluster (64 Cores)

By Allocating Cyclically, performance scales up to 64 cores.
Execution Time of the Fastest and Slowest Cores

- Allocating Cyclically
- Bisection

Image Number

- Fastest Core
- Slowest Core

Time (sec)

Execution Time of the Fastest and Slowest Cores

- 1.3x
- 2.8x
Processor Utilization

- Allocating Cyclically: 87 ~ 95%
- Bisection: 66 ~ 91%

Low processor utilization deteriorates the performance of Bisection
Utilized bandwidth is 750MB/s (only 7% of maximum (10.7GB/s))
Memory bandwidth is not bottleneck even when two clusters operate.
Power Consumption

Our many-core SoC achieves less than 3W

SoC : 2.21W

2Clusters
1.18W

Typical Process, Room Temperature, using Allocating Cyclically
Comparison with Desk-Top CPU

- Compared with Desk-Top CPU
  (Core™-i7-3820: 3.6GHz, 4 Cores, 8 Threads)

TDP of Core™-i7-3820 (130W) is used for calculating energy
Conclusion

• Future architecture will have many cores
  – A key challenge: How to efficiently use them?

• We evaluated the many-core SoC with parallelized face detection
  – Many-core is suited for the face detection because it exploits ROI based coarse-grained parallelism efficiently
    • Scale up by 30x (32 cores) to 60x (64 cores)
    • Balancing workload is important

• Power consumption is only 2.21W under actual workload: enables fan-less cooling
  – Our many-core SoC is remarkably energy efficient in image recognition applications
    • 20x better than the desk-top CPU
Thank you!