MPSoC 2011

FUJITSU

shaping tomorrow with you

A software centric system design for OS scheduling scheme in the upstream phase

Koichiro YAMASHITA
Fujitsu Laboratories LTD
Platform technologies laboratories

Challenging

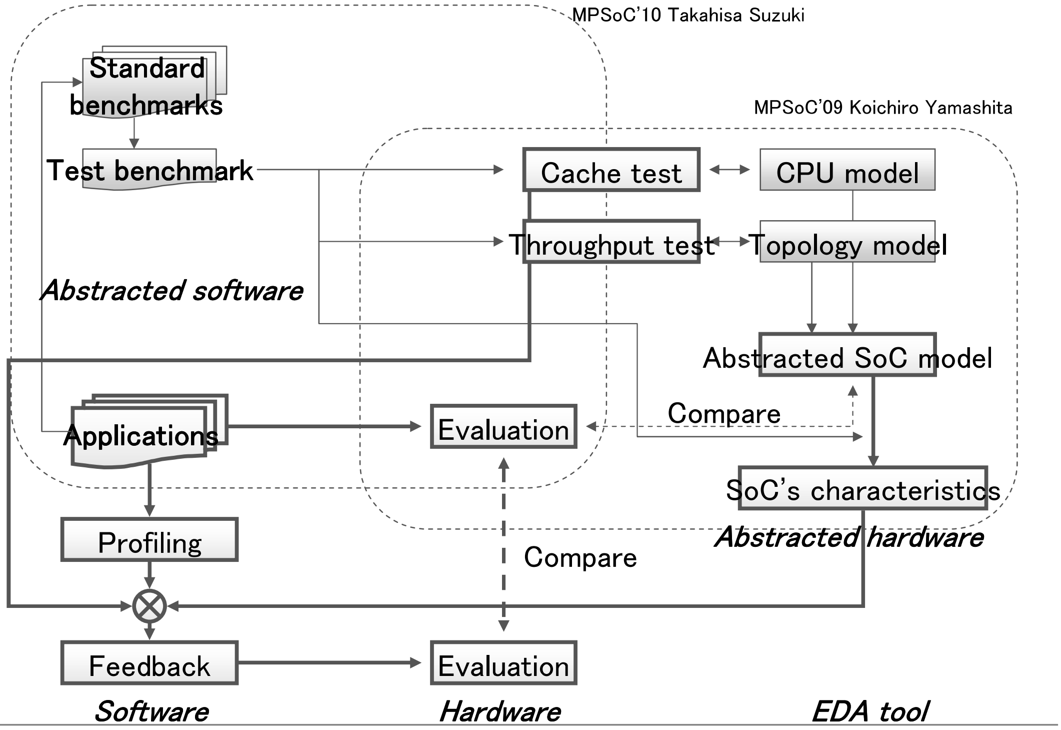
FUJITSU

End−goal: A software that maximum uses potential of hardware

- Conditions:
  1. Multifunctional and multi-core based SoC
     (complex transaction)
  2. Multiple-applications, multiple-threads on the OS
     (complex software configuration)
  3. Enhancement of application is minimum
     (Dilemma of consumer application)

Milestone of this project

- SW, HW harmonized system design



MPSoC'10 Takahisa Suzuki

MPSoC'09 Koichiro Yamashita

Standard benchmarks

Test benchmark

Cache test

CPU model

Throughput test

Topology model

Abstracted software

Abstracted SoC model

Applications

Evaluation

Compare

SoC's characteristics

Profiling

Abstracted hardware

Compare

Feedback

Evaluation

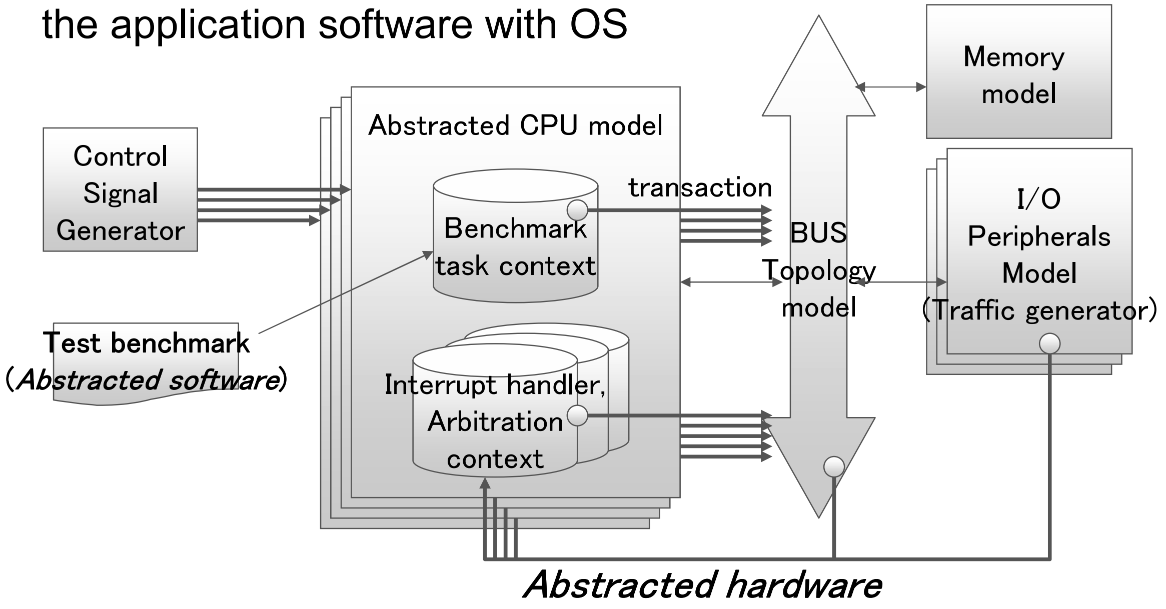
Software

Hardware

EDA tool

Abstracted system model

- By using highly abstracted model, it is still difficult to evaluate the application software with OS



Memory model

Abstracted CPU model

Control Signal Generator

transaction

Benchmark task context

BUS Topology model

I/O Peripherals Model (Traffic generator)

Test benchmark (*Abstracted software*)

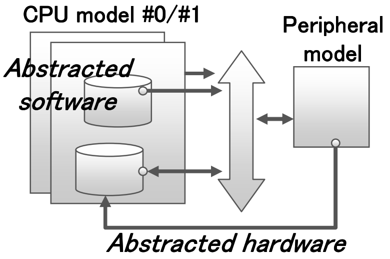
Interrupt handler, Arbitration context

Abstracted hardware

The abstracted modeling is effective to know the characteristic of object hardware. And also the software that operate many threads should be abstracted to simplify the system.

Evaluation of characteristics of target system

- A “characteristics of collision” between threads



Abstracted software

– CPU #0 model creates 2M access transaction (constant).

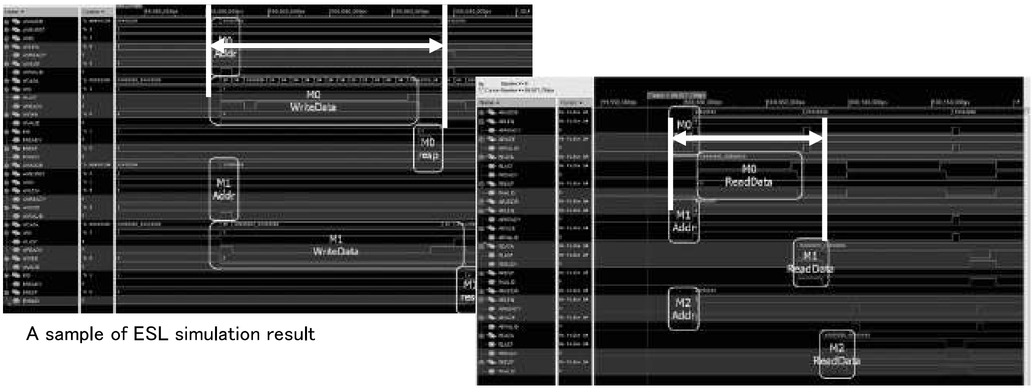
– CPU #1 model creates 32K to 2M access transaction (variable) .

Abstracted hardware

– Peripheral model creates system’s whole access transaction (LCD, file access etc.)
(The average of transaction is a constant though generated in random)

– The parameter of bus model is configurable (priority, outstand buffer depth etc)

– Measure performance of CPU #1

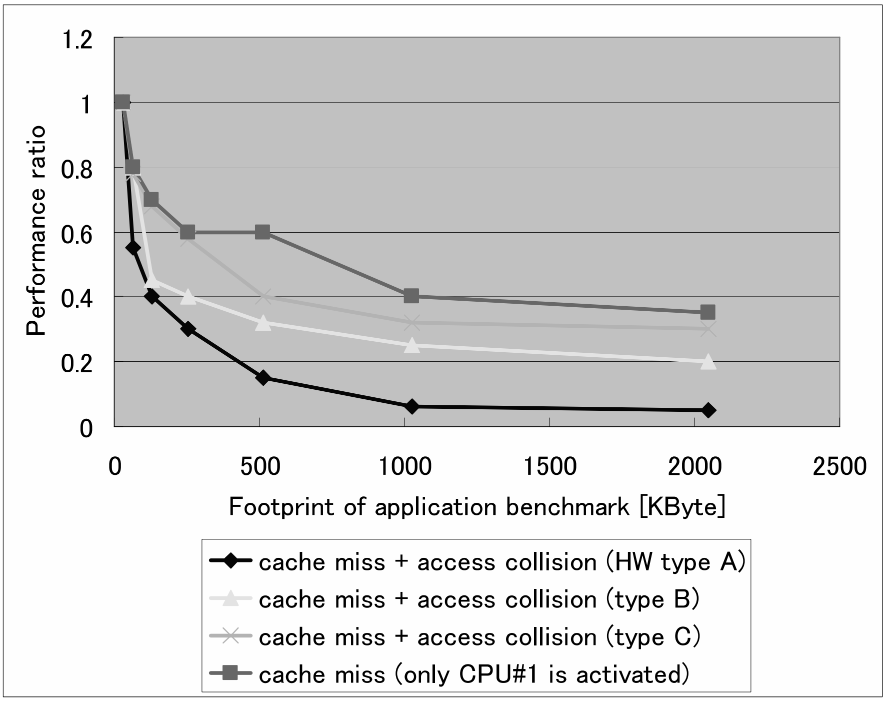


A sample of ESL simulation result

The bus collision has been happened also between independent threads of neighbor CPU

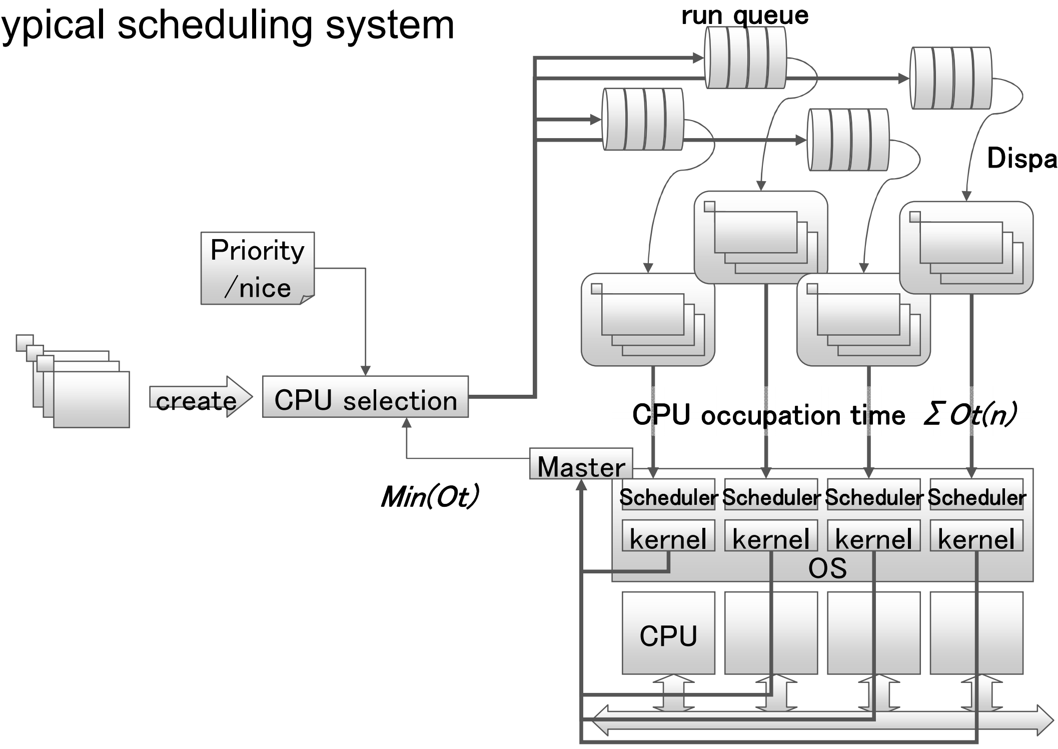
Result of performance

- A “characteristics of collision” performance of target system



Feedback to the scheduler of OS

- Typical scheduling system



Point of consideration

- Simple and traditional approach

Give some feedback from the hardware overhead to CP length of target thread... (CP=Critical Path, an elapsed time of target software module)

a) The scheduling accuracy is improved by the updated software cost.

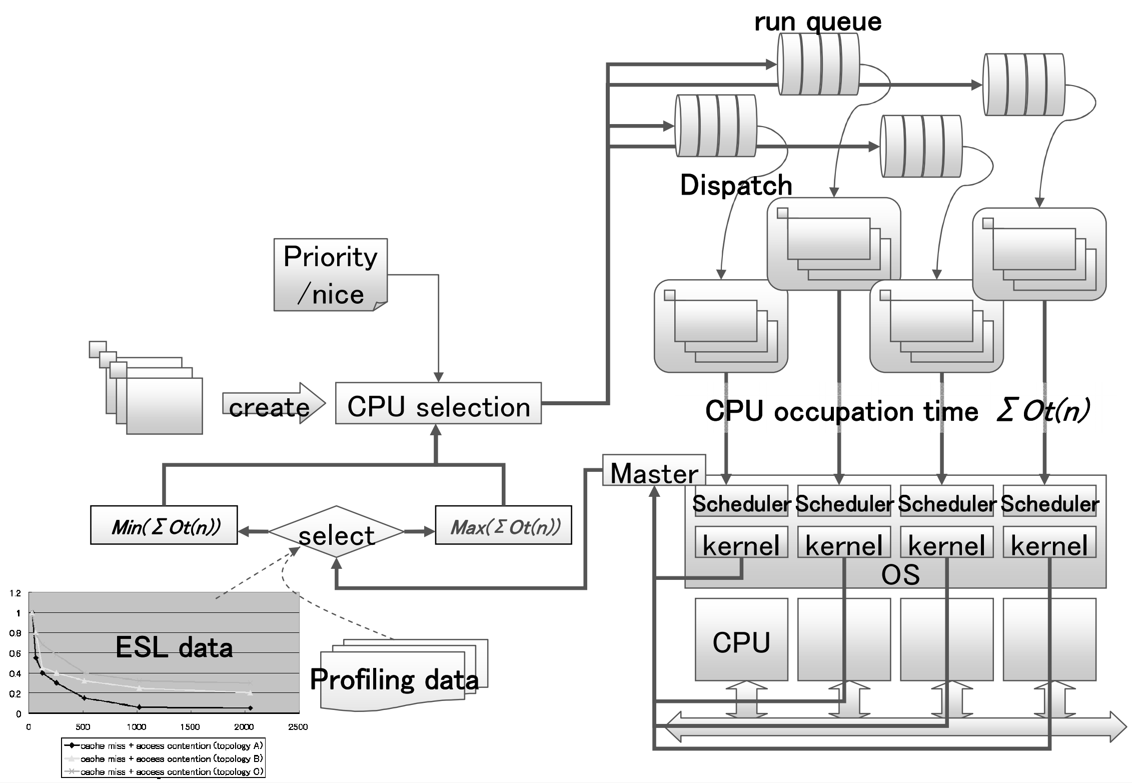
Useless energy at the collision period is not solved.

b) The execution order is replaced for the collision avoidance.

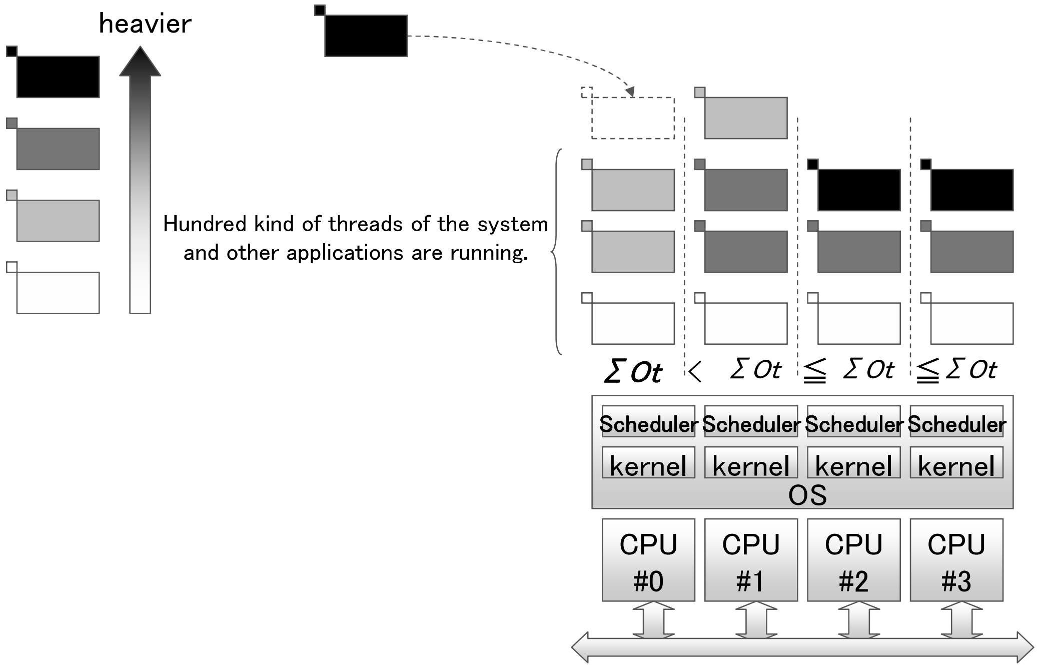
If there is a dependency in the execution order between threads…

→ Save energy, keep execution order and get good performance. How to solve it.

Proposed scheduling system (diagram)

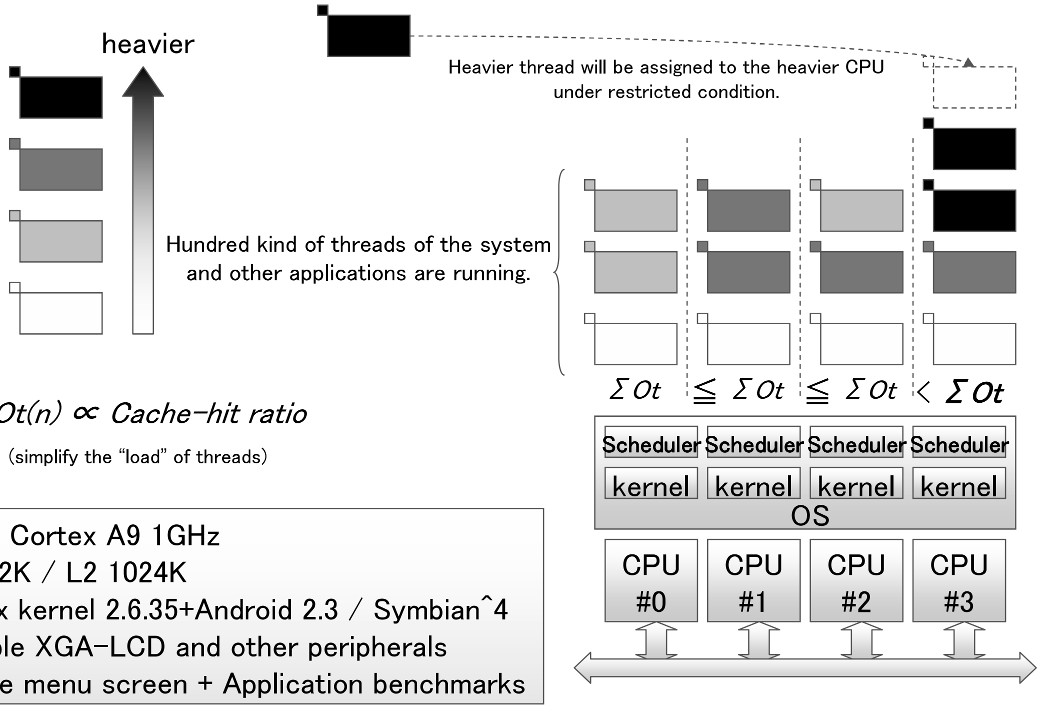


Typical scheduler (an image of behavior)


heavier
Hundred kind of threads of the system
and other applications are running.
Σ Ot < Σ Ot ≦ Σ Ot ≦ Σ Ot
Scheduler
Scheduler
Scheduler
Scheduler
kernel
kernel
kernel
kernel
OS
CPU #0
CPU #1
CPU #2
CPU #3

Proposed scheduling system (an image of behavior)

- An image of proposed scheduler


heavier
Heavier thread will be assigned to the heavier CPU under restricted condition.
Hundred kind of threads of the system and other applications are running.
Σ Ot ≦ Σ Ot ≦ Σ Ot < Σ Ot
Scheduler
Scheduler
Scheduler
Scheduler
kernel
kernel
kernel
kernel
OS
CPU #0
CPU #1
CPU #2
CPU #3

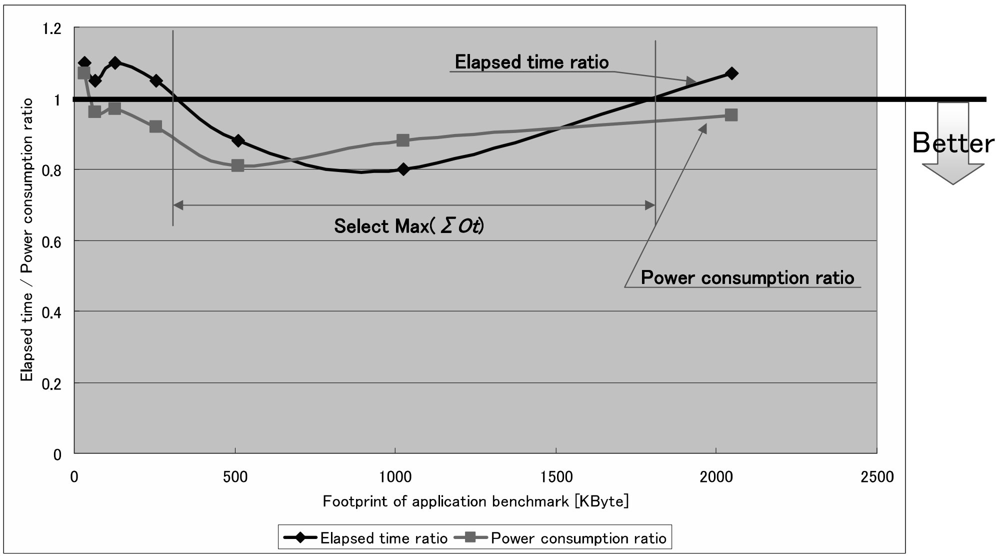
Ot(n) ∝ Cache-hit ratio

(simplify the "load" of threads)

CPU Cortex A9 1GHz
L1 32K / L2 1024K
Linux kernel 2.6.35+Android 2.3 / Symbian^4
Enable XGA-LCD and other peripherals
Home menu screen + Application benchmarks

Result of performance / power consumption

- Measurement result on the existing hardware device (the past scheduler= 1).



The figure is dividing of the elapsed time and the power consumption of the proposal scheme by the result of original scheme.

Performance and power improved 20% by a simple change of scheduling method.

Conclusion

- It is difficult to evaluate a complex whole system by using a simulation model.
- Abstract model's characteristic gives some feedback to OS scheduler.
- In the viewpoint of the performance and power consumption, the effect was able to be confirmed without large-scale enhancement with an existing system.

FUJITSU

shaping tomorrow with you