THE POSSIBILITIES ARE INFINITE

ASP-DAC 2010 9D-1

–Possibility of ESL–
A software centric system design for multicore SoC in the upstream phase

Koichiro YAMASHITA

SENIOR RESEARCHER : Fujitsu Laboratories Platform Technologies Labs.

MANAGER : Fujitsu LTD. Mobile Phones Dept.

CHAIRMAN : Symbian Foundation SMP Working Group

Contents

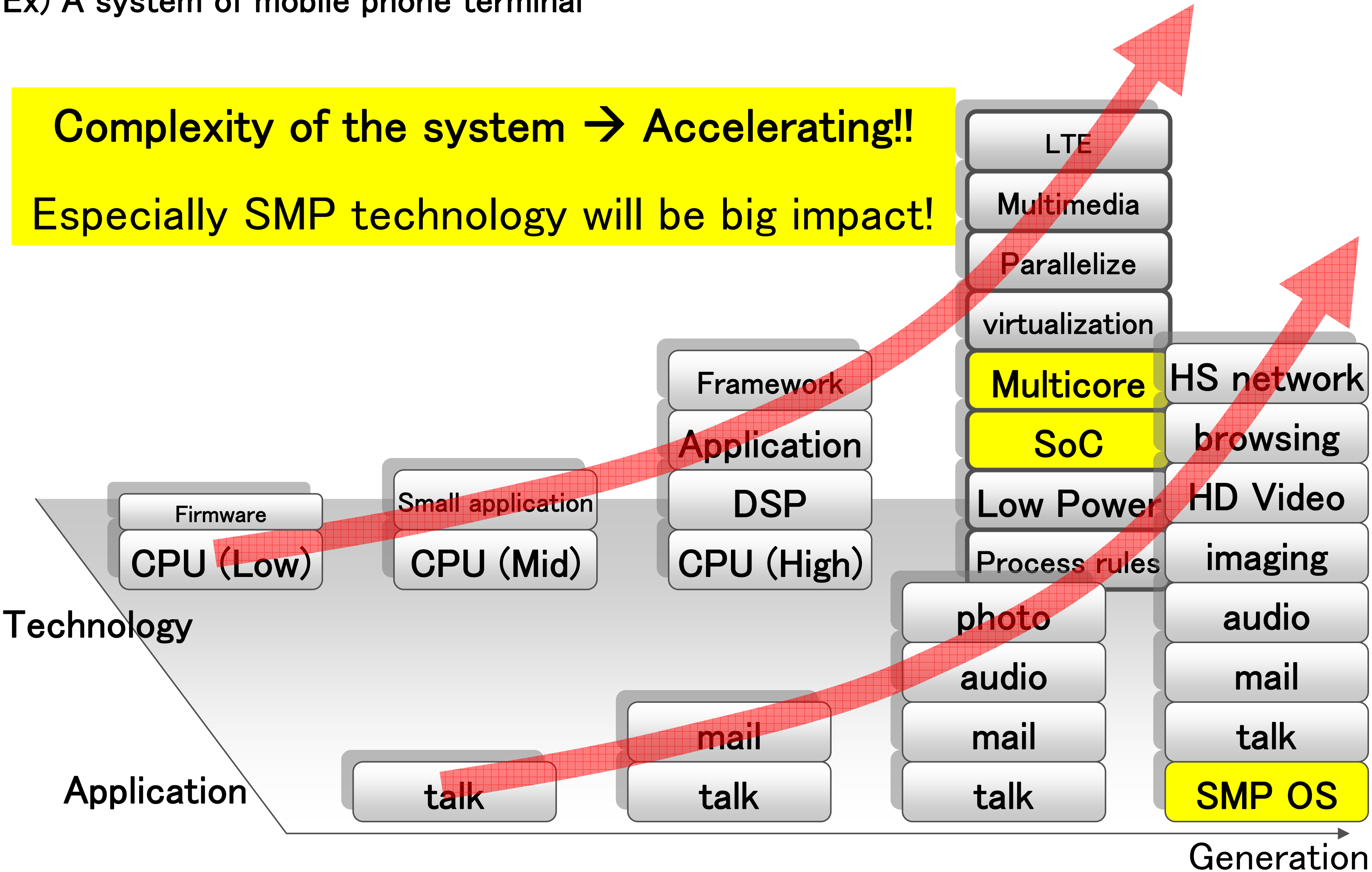
Why “Software centric” !?

1.Why "Software Centric" – background–

Ex) A system of mobile phone terminal

Complexity of the system → Accelerating!!

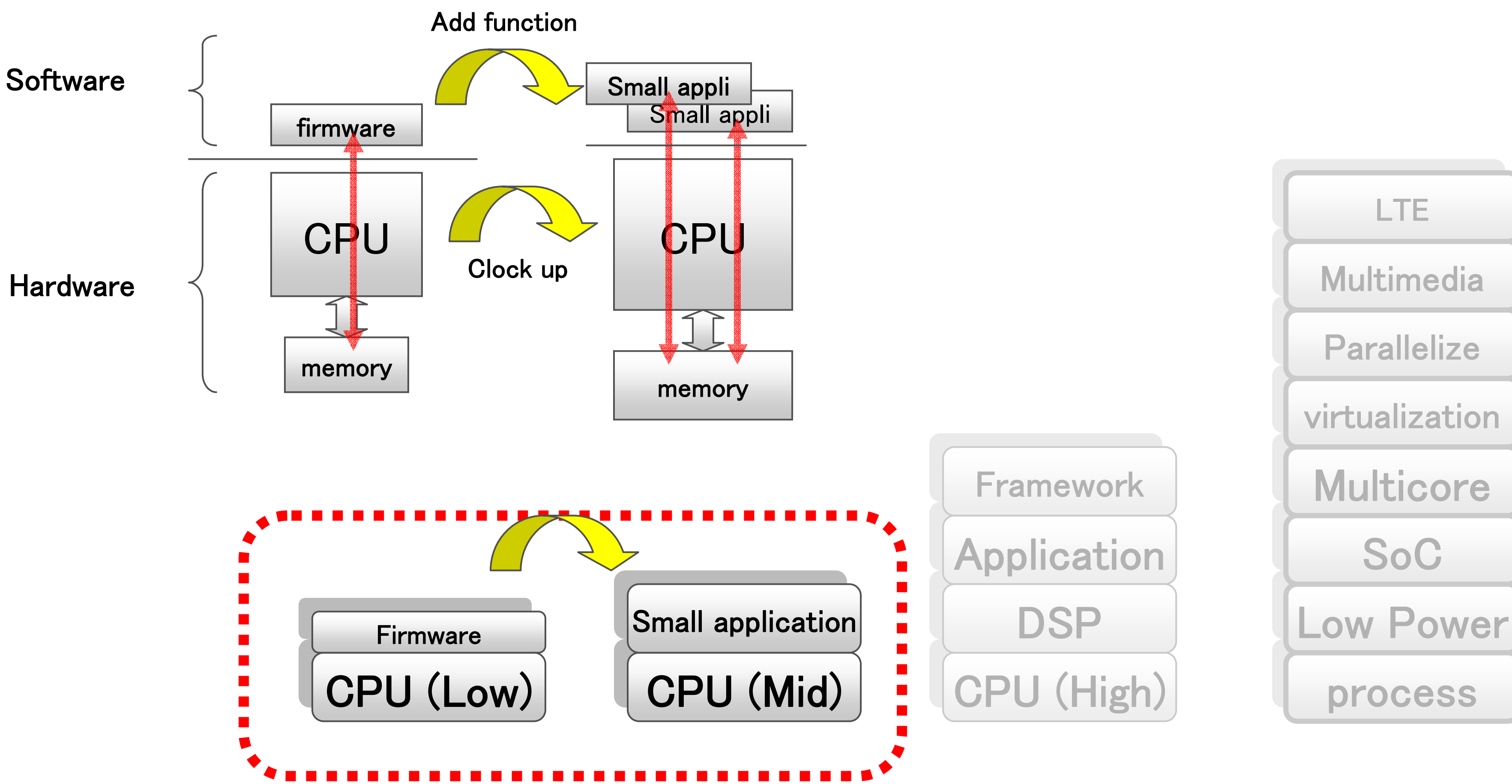
Especially SMP technology will be big impact!



1.Why "Software Centric" – background–

Ex) A system of mobile phone terminal

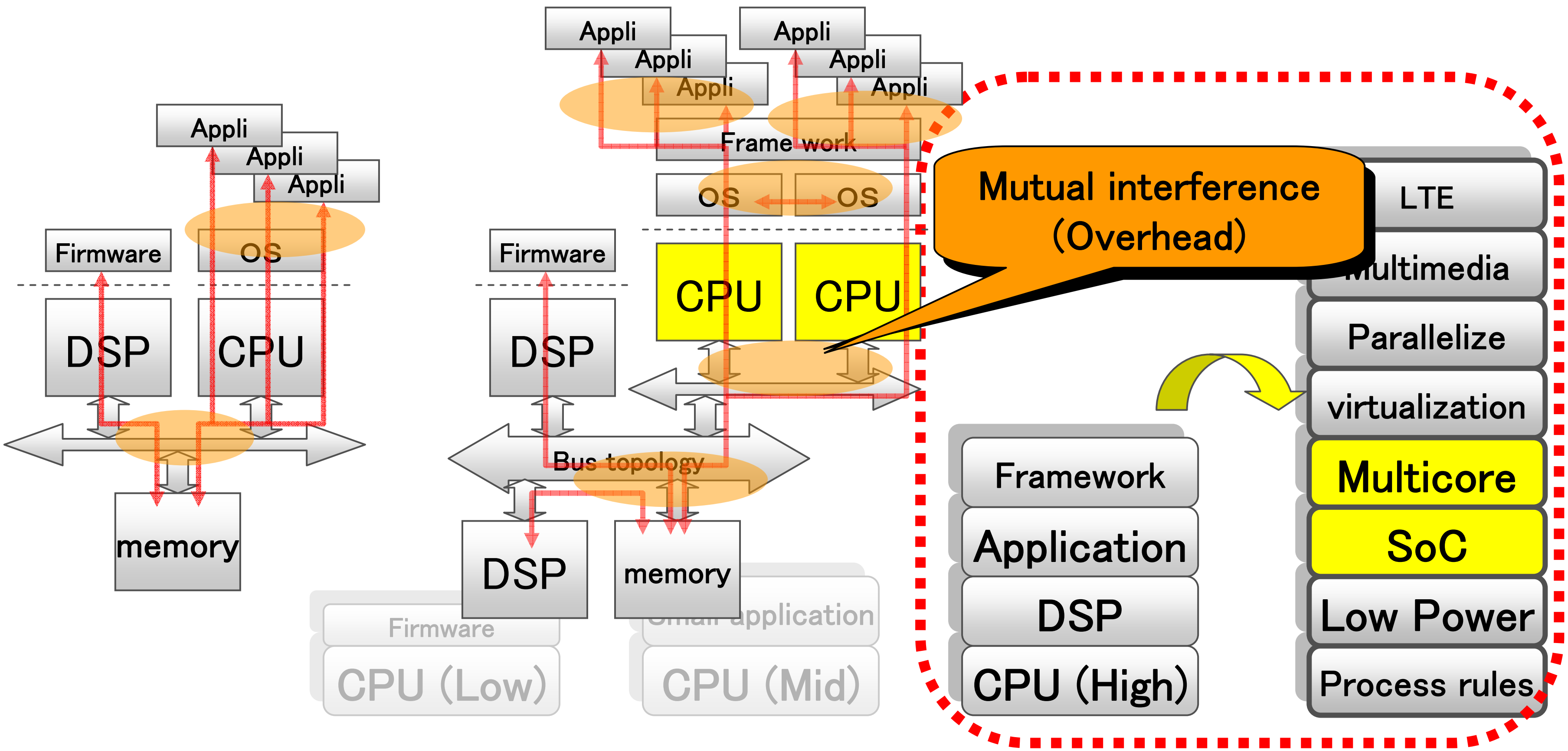
Small enhancement : Easy to estimate by hand estimation



1.Why "Software Centric" – background–

Ex) System of mobile phone terminal

Large enhancement : Hard to estimate by hand estimation



Difficulty of Software & Hardware co-design

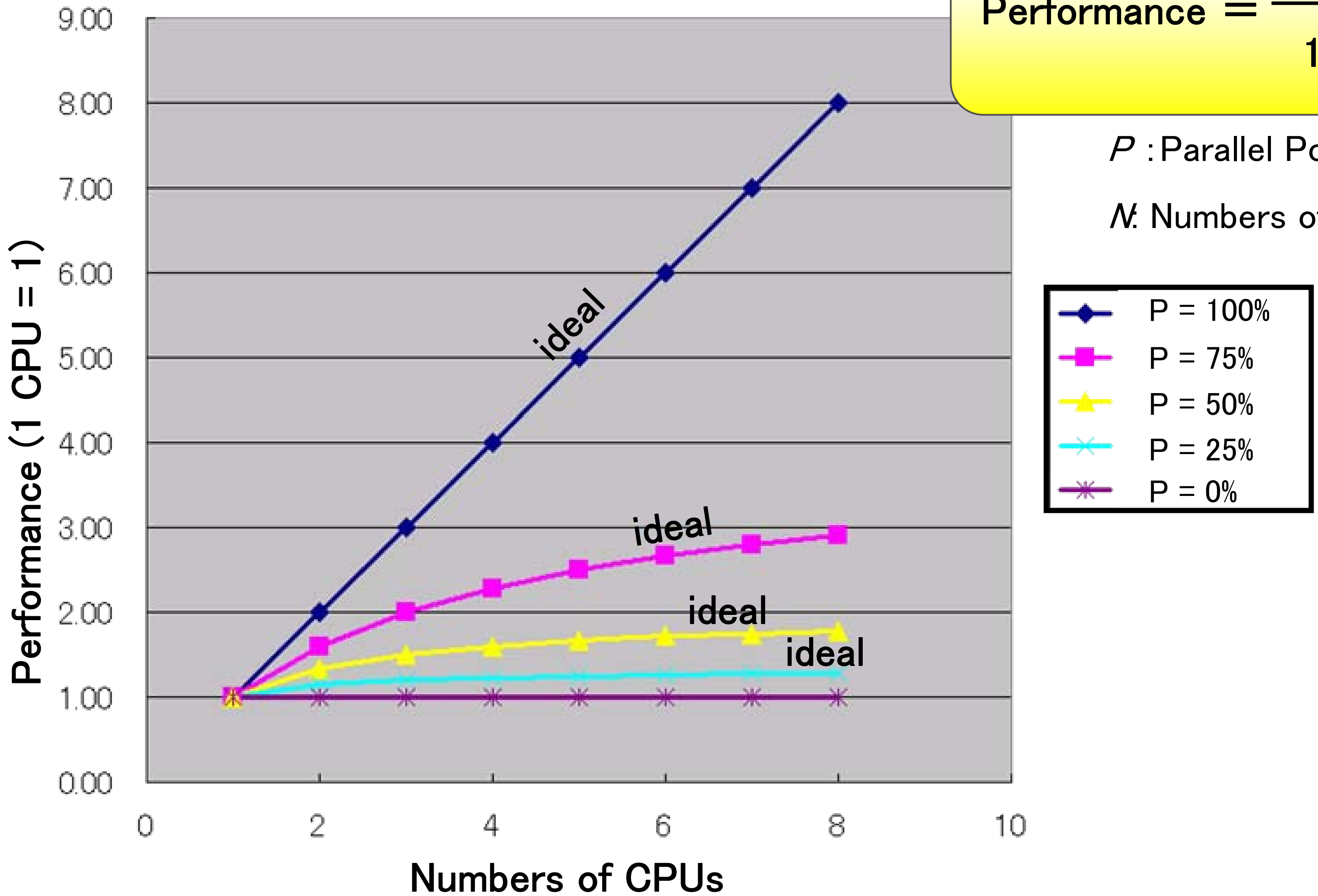
2.Difficulty of Software & Hardware co-design

Amdahl's law

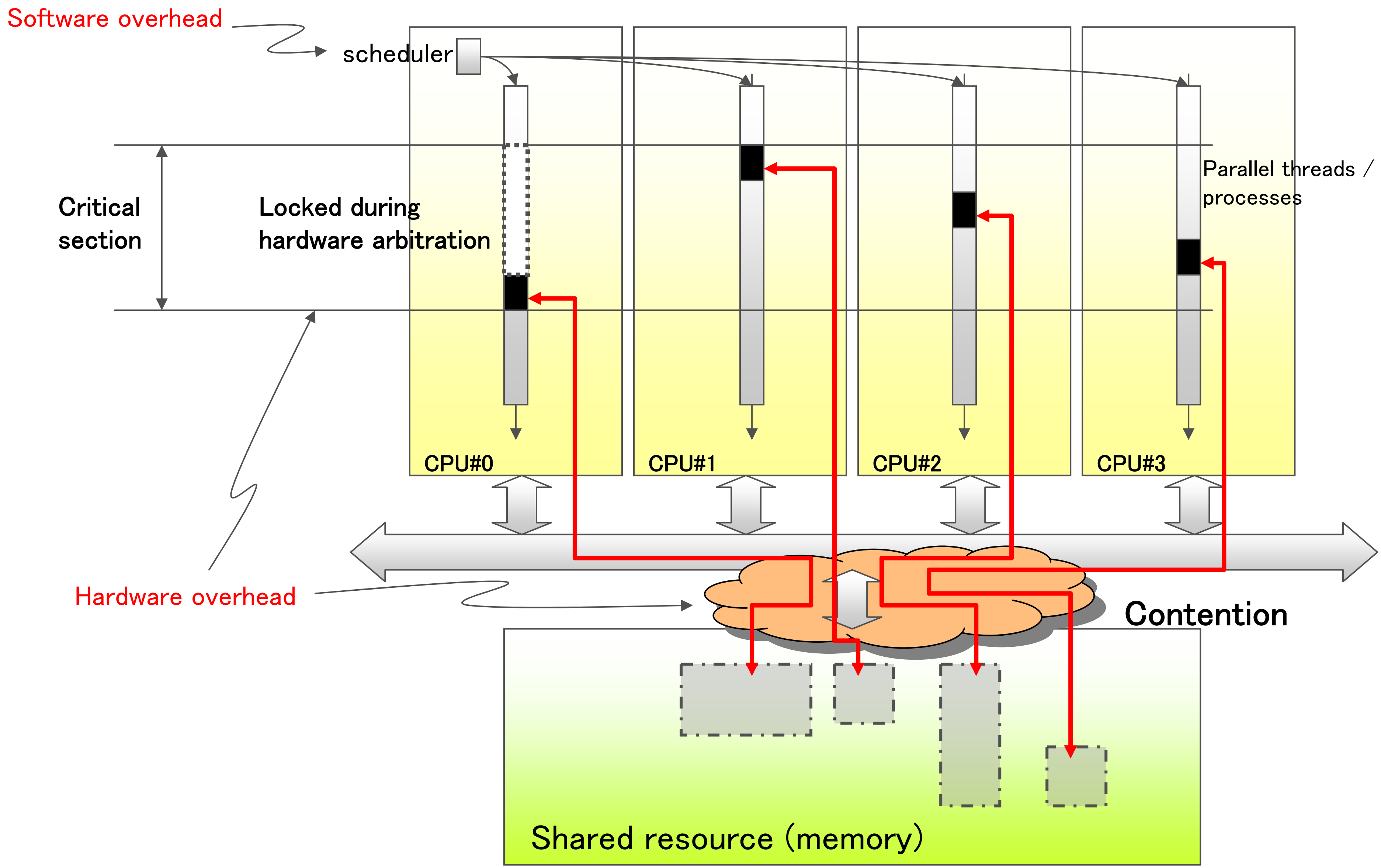
$$\text{Performance} = \frac{1}{1 - P + \frac{P}{N}}$$

P : Parallel Portion

N: Numbers of CPUs



2.Difficulty of Software & Hardware co-design

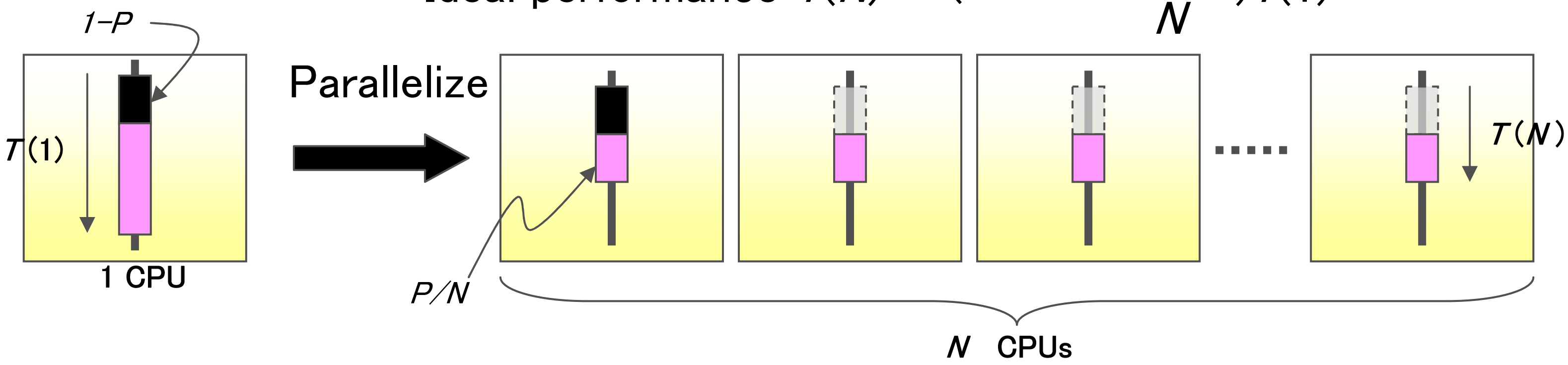


Overhead of both software and hardware must be considered.

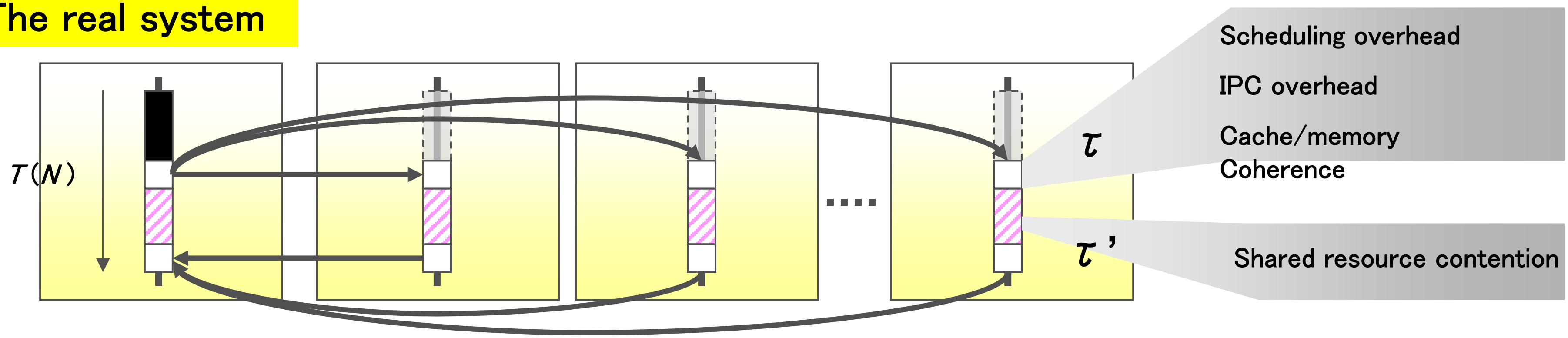
2.Difficulty of Software & Hardware co-design

The ideal system

Ideal performance $T(N) = (1 - P + \frac{P}{N})\,T(1)$



The real system

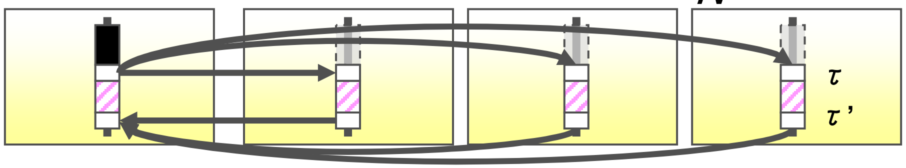


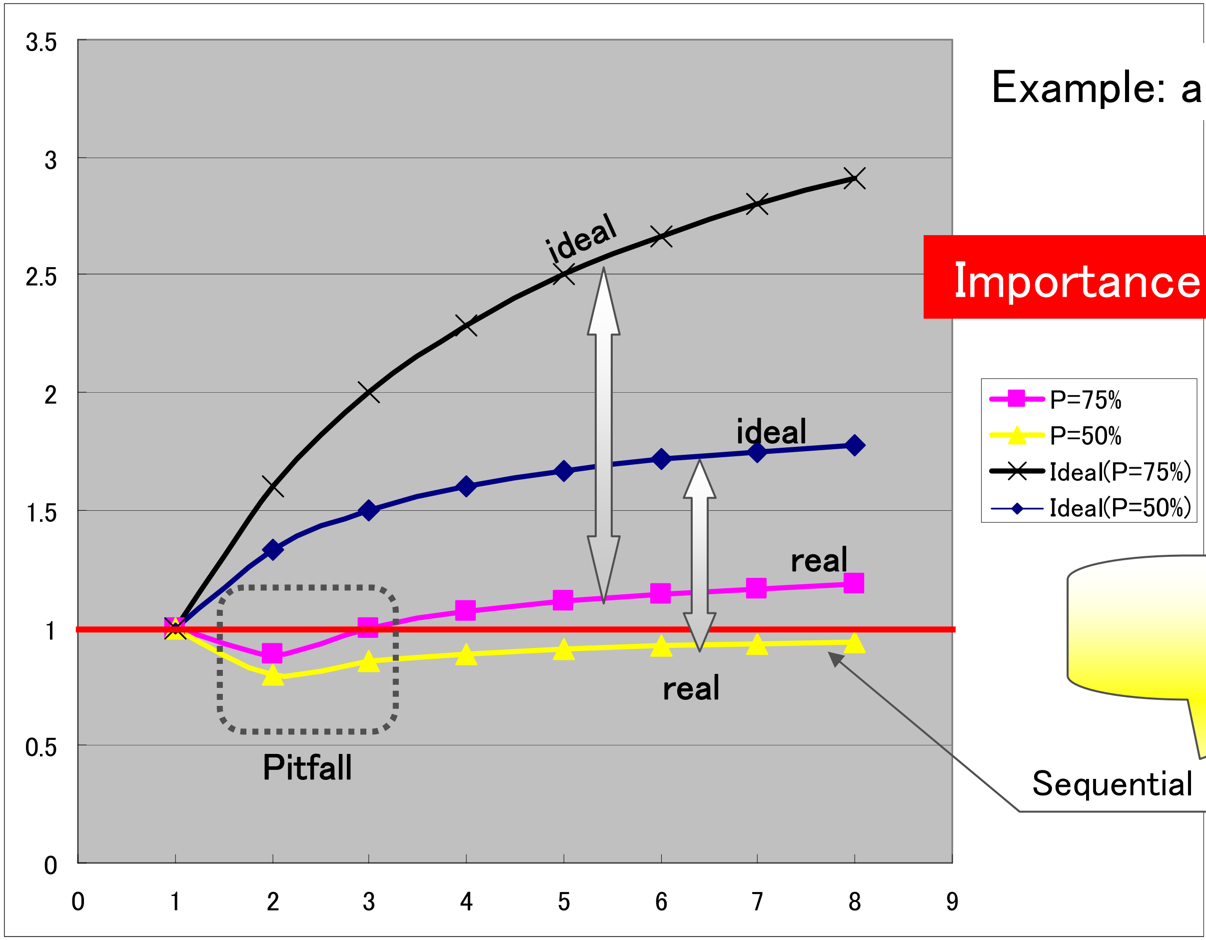
Amdahl's low with mutual interference

Real performance $T(N) = (1 - P + \frac{P}{N})\,T(1) + \tau + \tau'$

2.Difficulty of Software & Hardware co-design

Real performance $T(N) = (1 - P + \frac{P}{N})T(1) + \tau + \tau'$





Example: a case of T(1) =50, τ =10

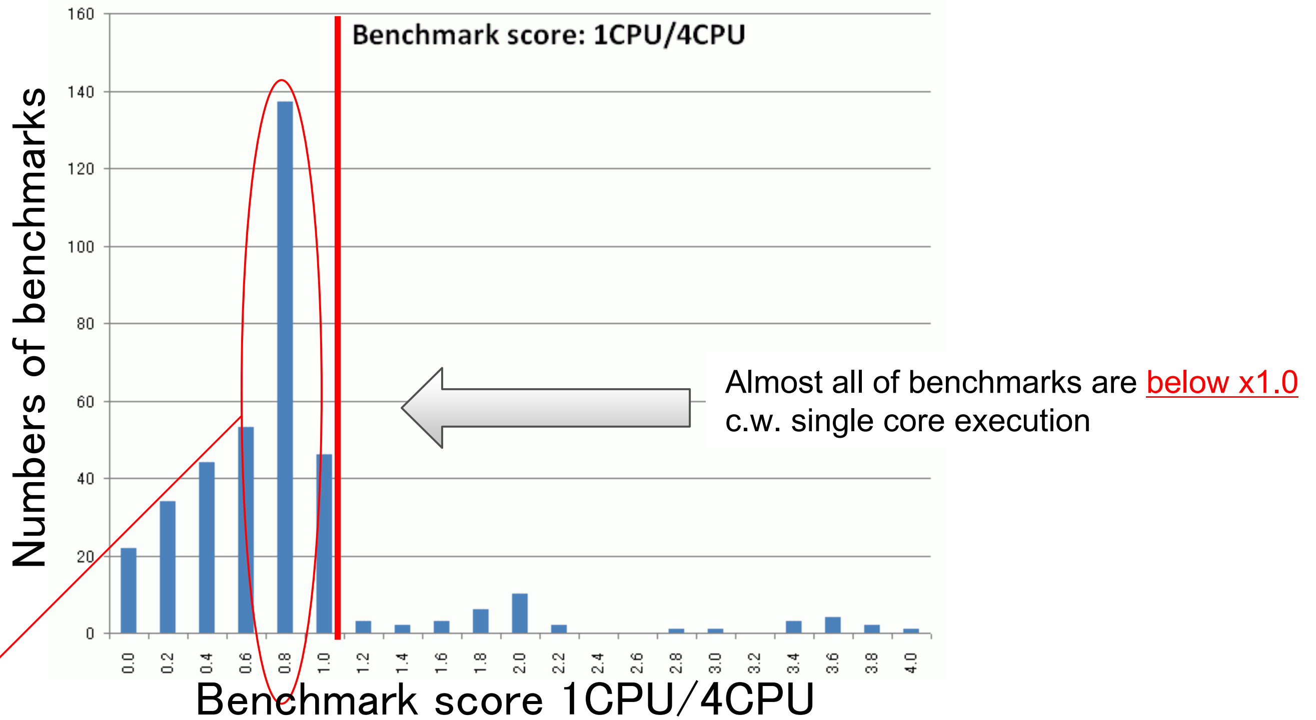
Importance of "Overhead" estimation

How to plan
Software development

Sequential execution on the single CPU is better

2.Difficulty of Software & Hardware co-design

Example of result of benchmarks



Benchmark's peak : x0.6-1.0

It means, most of software achieve only 10-20% existing parallelism (sequential part is 80-90%) without any optimization.

> 1.0 : application must be **implemented** in parallel (coarse grain execution)

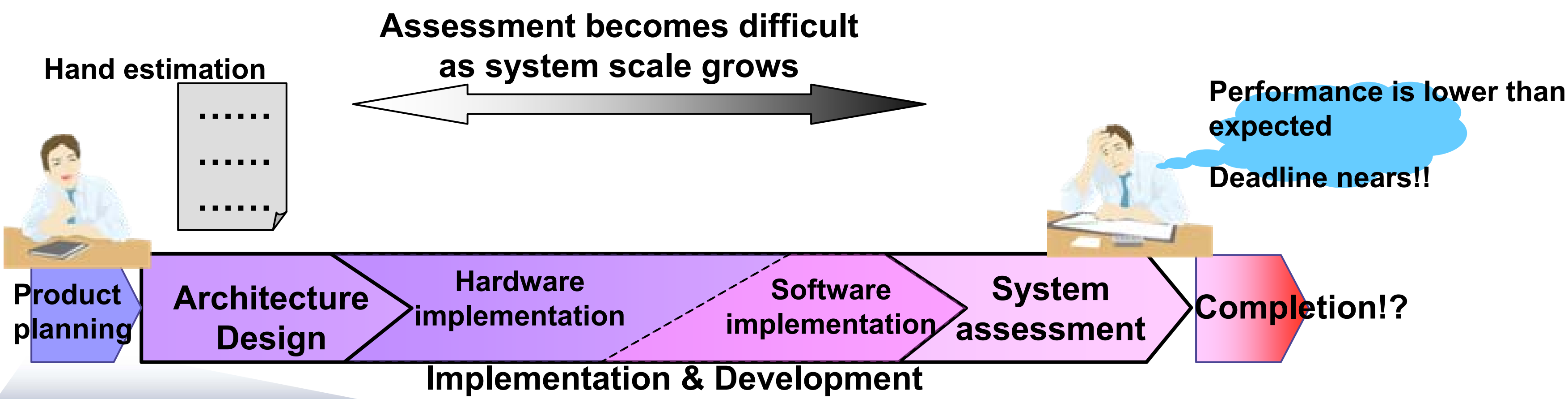
<= 1.0 : applications must be **executed** in parallel (distributed execution)

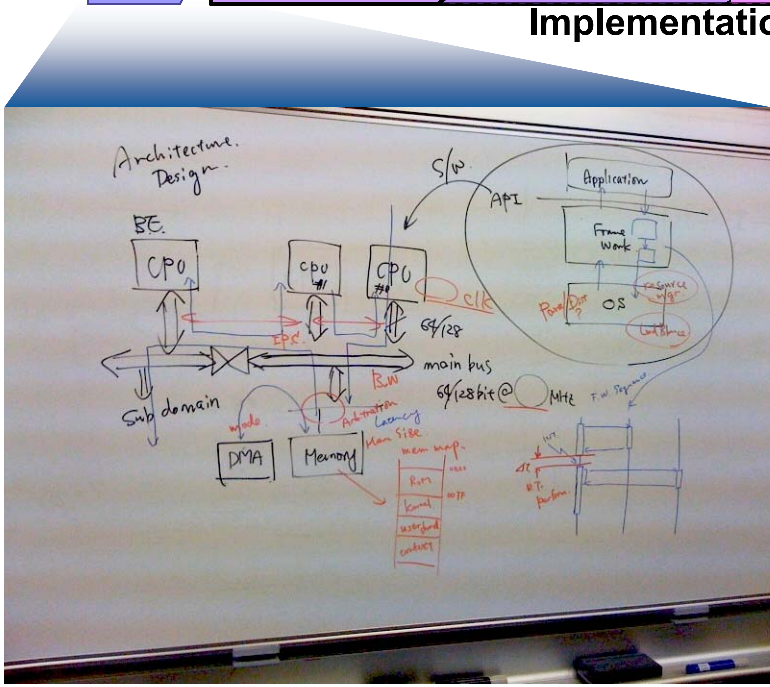
→ A plan of Software implementation

*Linux(2.6.30)-SMP configuration+ARM based multicore processors with open source benchmarks (N-bench/Himeno-bench/...)

2.Difficulty of Software & Hardware co-design

Conventional design flow



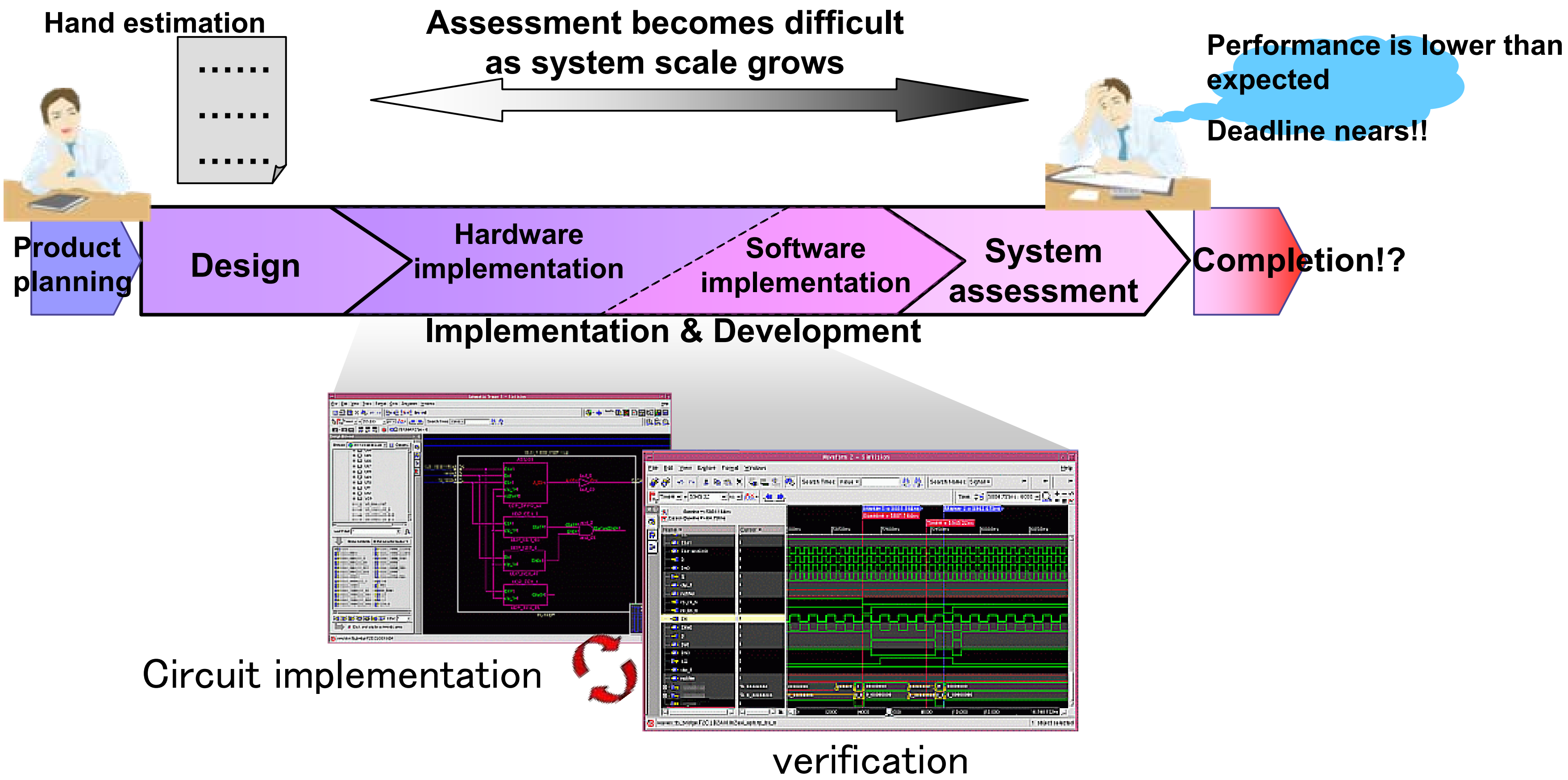


Heuristically and manually design,

but not quantitative.

2.Difficulty of Software & Hardware co-design

Conventional design flow

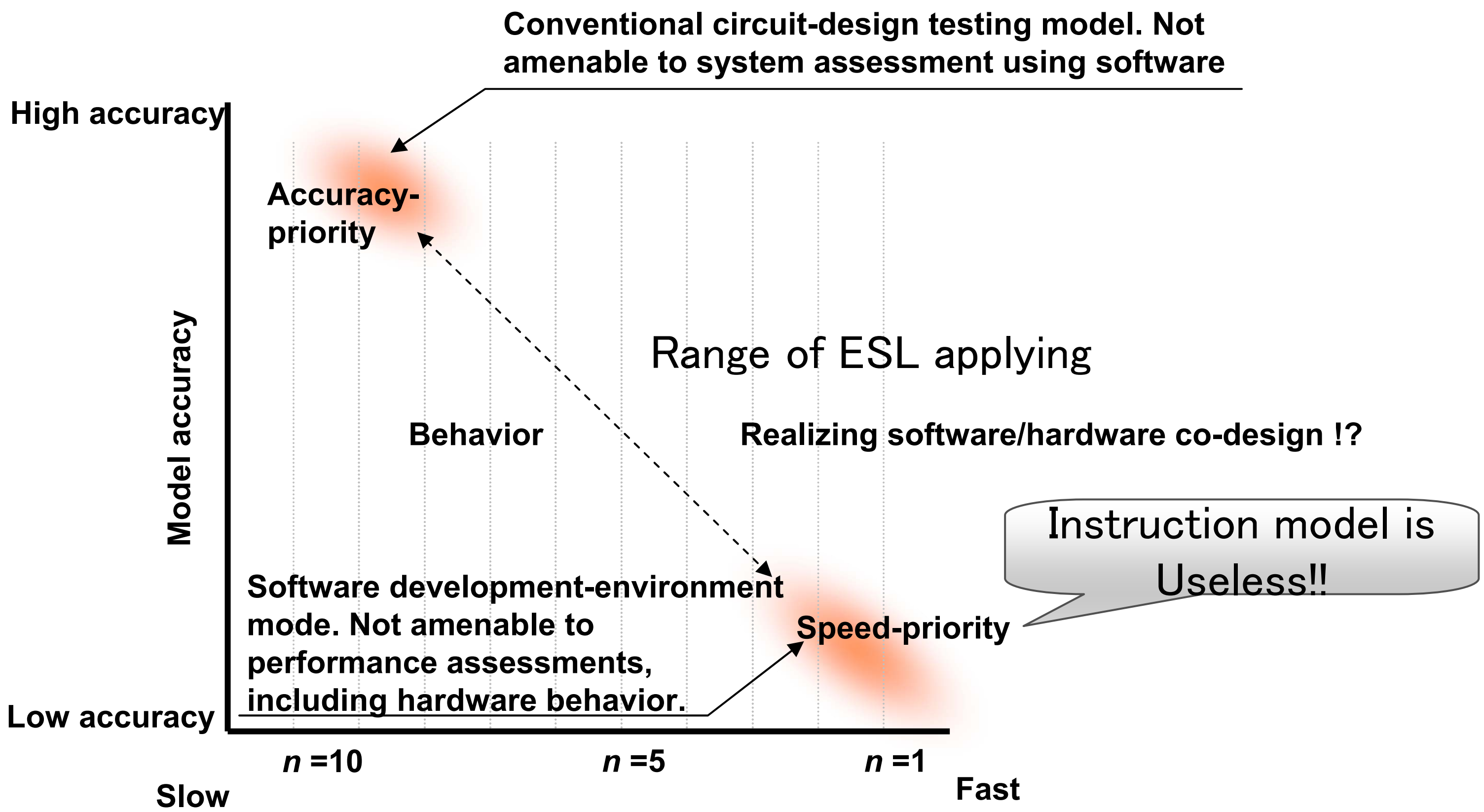


Cannot check the fundamental design policy of the system architecture

Software centric system design methodology

3.Software centric system design methodology

Possibility of ESL Simulation



Model speed n (inverse-logarithmic scale, $1/10^n$ actual device speed)

3.Software centric system design methodology

What we want !?

- Boot up with SMP-OS with framework as a software evaluation platform.
- Execute benchmarks
- Measure software overhead and hardware overhead
- Check effect of target parallelized method, cache or memory usage… and so on
- Feedback to the design

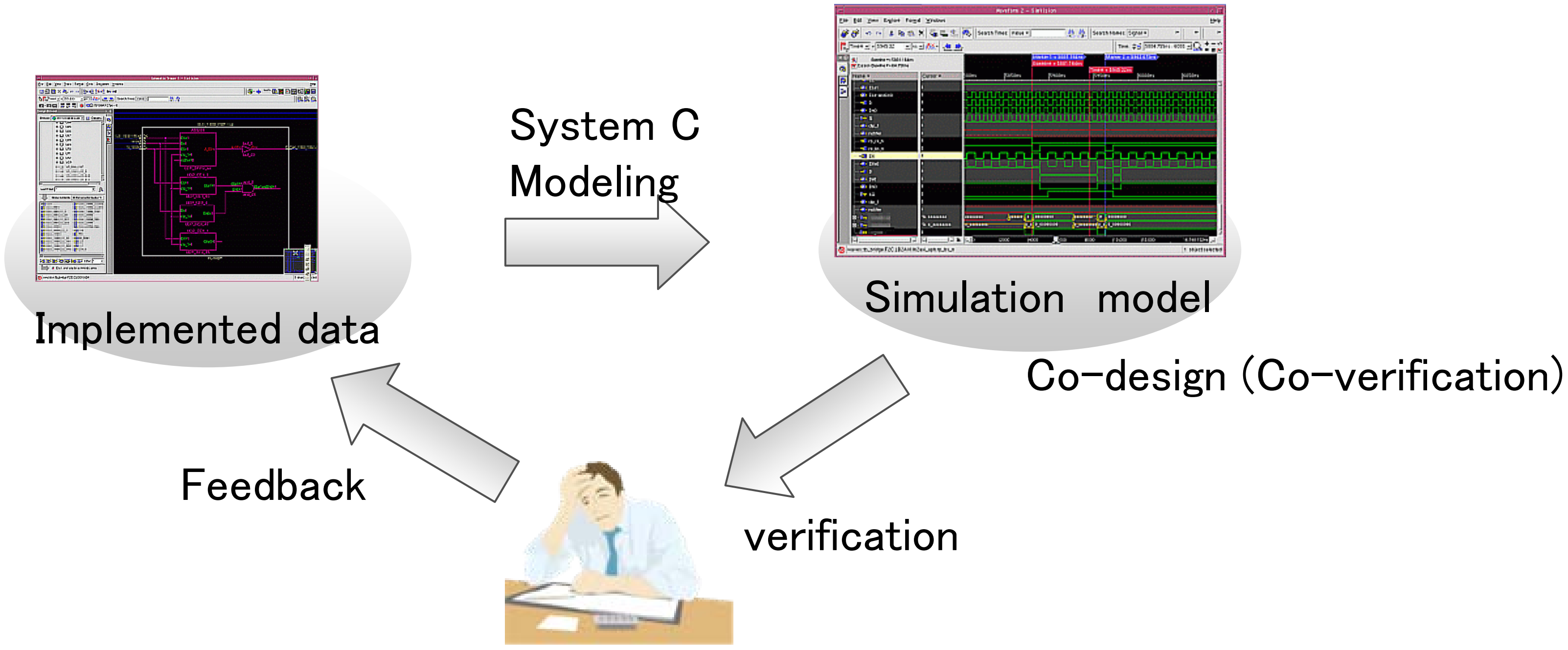
BY USING ESL MODEL in the upstream phase!!

3.Software centric system design methodology

Possibility of ESL Simulation

Conventional usage of ESL Simulation

- Verification for logical activation of implemented circuit of architecture.
- Verification for instructions of software **step by step** without overhead.

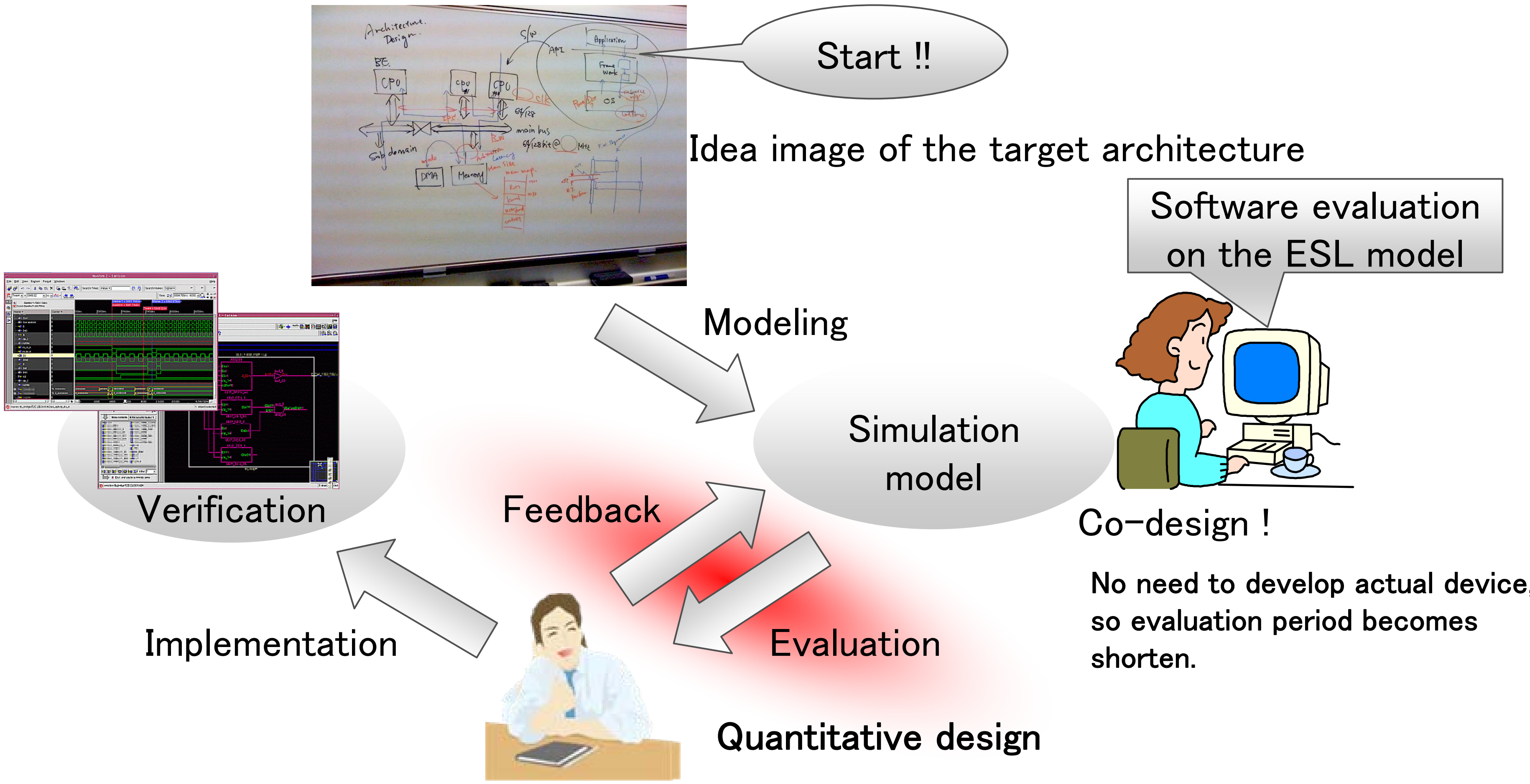


Cannot evaluate the target architecture itself

No Glory!!!

3.Software centric system design methodology

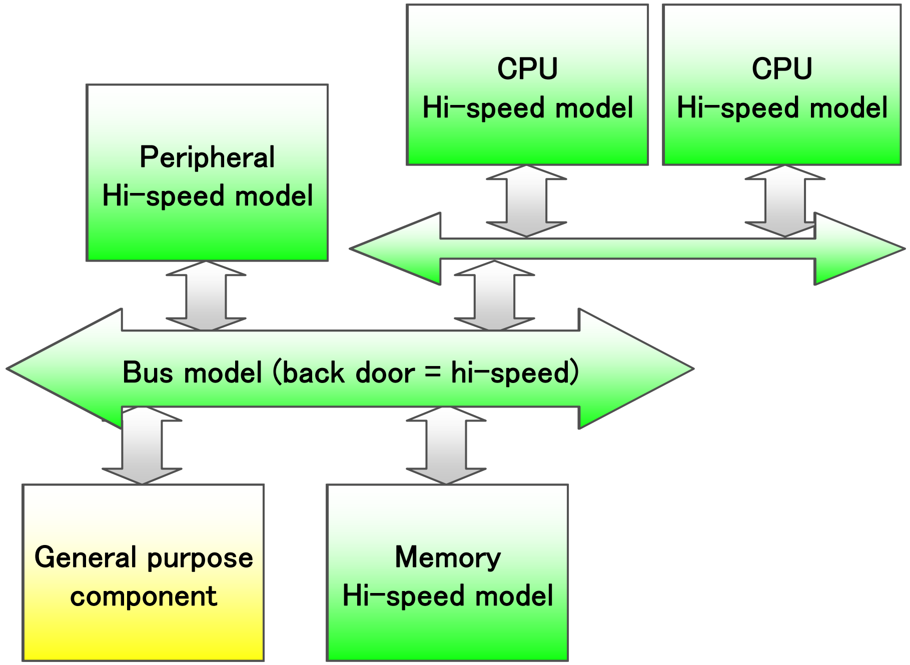
■ Software centric approach



Evaluate the target architecture itself

3.Software centric system design methodology

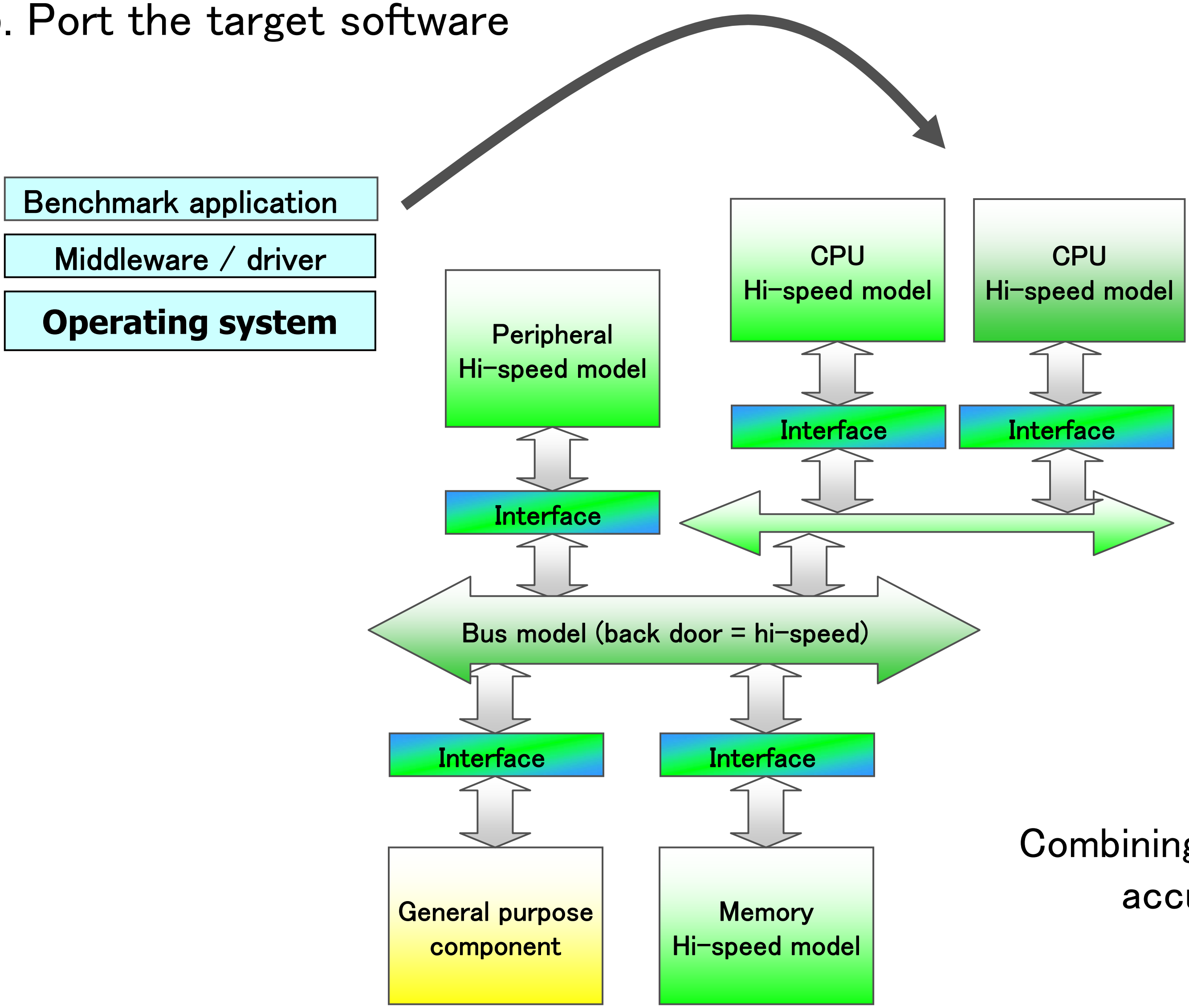
1. Draw image of the target architecture with initial parameters
2. Pick up re-use modules and missing modules
3. Put all of the components and complete the target architecture model anyway



3.Software centric system design methodology

4. Put the interface modules between different accuracy / speed policy components

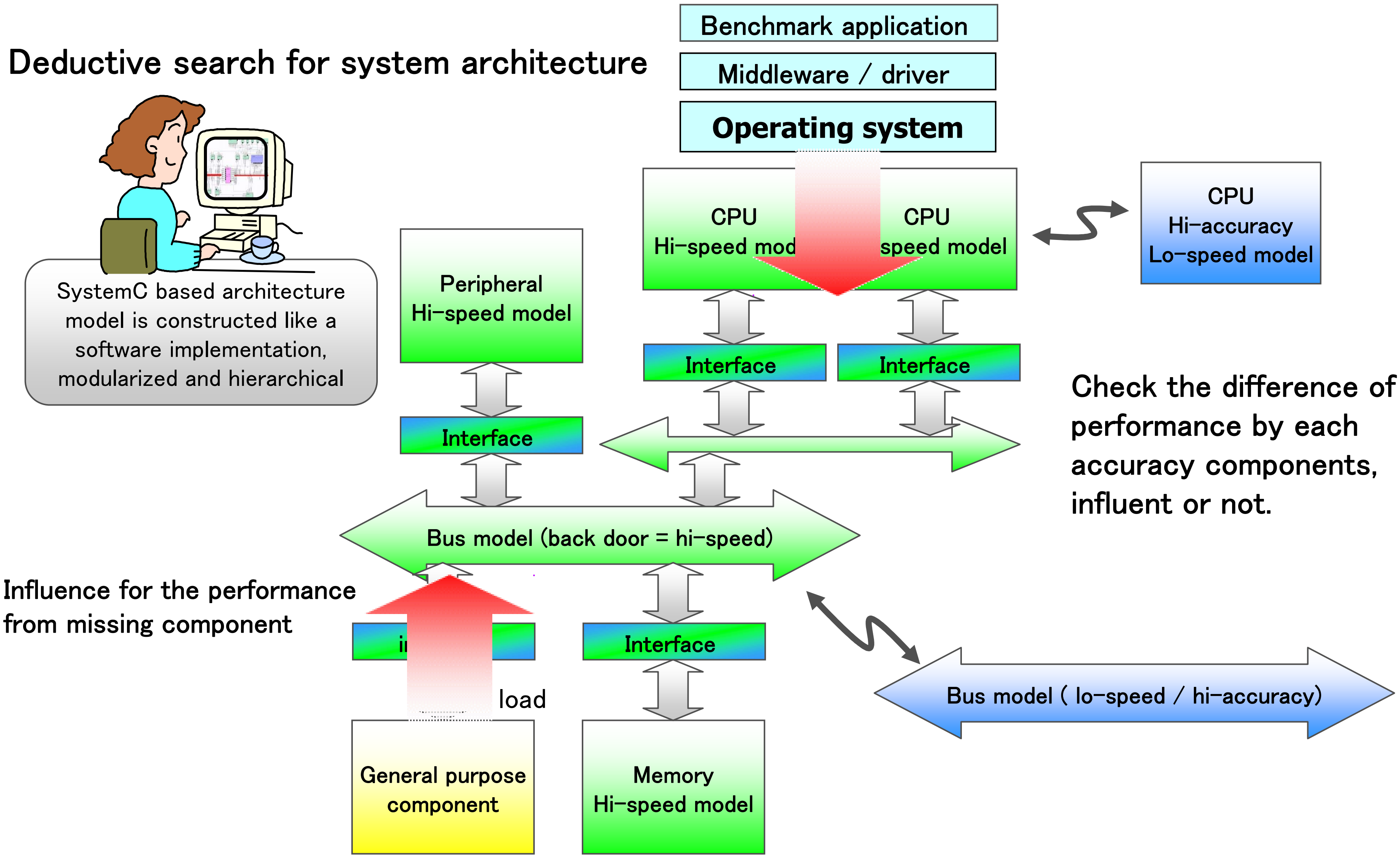
5. Port the target software



Combining speed-priority and accuracy-priority.

3.Software centric system design methodology

Repeat evaluation with changing parameters, topologies and accuracy of the parts



Conclusion

- Multi-core system getting complex for its architecture, and hard to estimate which performance before to implement it. And it is very difficult to assess the overhead. The key items for the performance must be the application level software.
- To solve this problem, we proposed the methodology by using ESL technology.
- We developed ESL environment that uses software on OS, assessed and fed the results of the analysis back into the design process.
- This demonstrated the feasibility of overall system assessments without having to wait for actual implementation.
- Model-based assessments can be reiterated within a short period, enabling estimation of system-level performance even during upstream design. This makes it possible to optimize design plans without the need to wait for actual chip development to be completed, thereby reducing the risk of having to go back and re-design.

Q and A

3.Software centric system design methodology

- Deductive approach to develop the ESL based architecture model.
- An architecture evaluation environment by SystemC based ESL technology which can execute application software.

⇨ **Model-based assessments can be reiterated within a short period, enabling estimation of system-level performance even during upstream design. This makes it possible to optimize design plans without the need to wait for actual chip development to be completed,**

Remained item …

Is "accuracy" enough by this approach?

3.Software centric system design methodology

- "Accuracy" check of this ESL structure

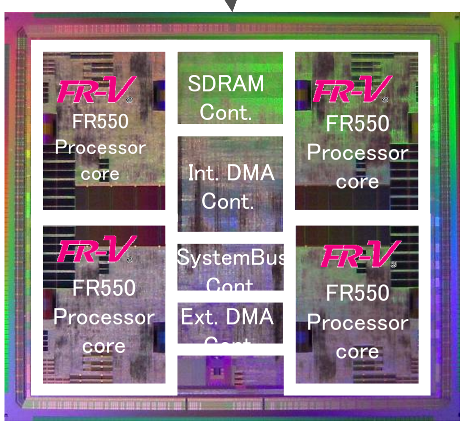
  1) Prepare manufactured multi-core processor.

  2) Prepare ESL simulation model of 1), based on combining speed-policy and accuracy-policy components.

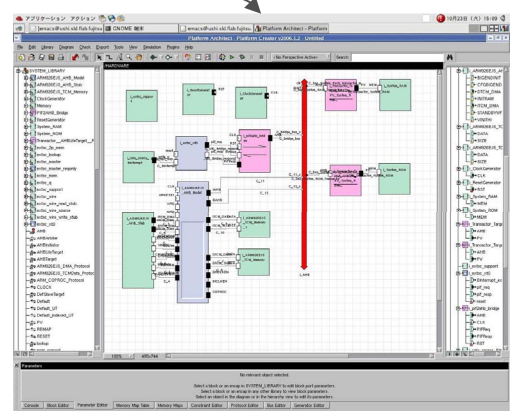
  3) Execute benchmark applications (EMMBC) on each environment.

  4) Compare the result of 1) and 2) and check difference of exec-time.

EMMBC benchmarks



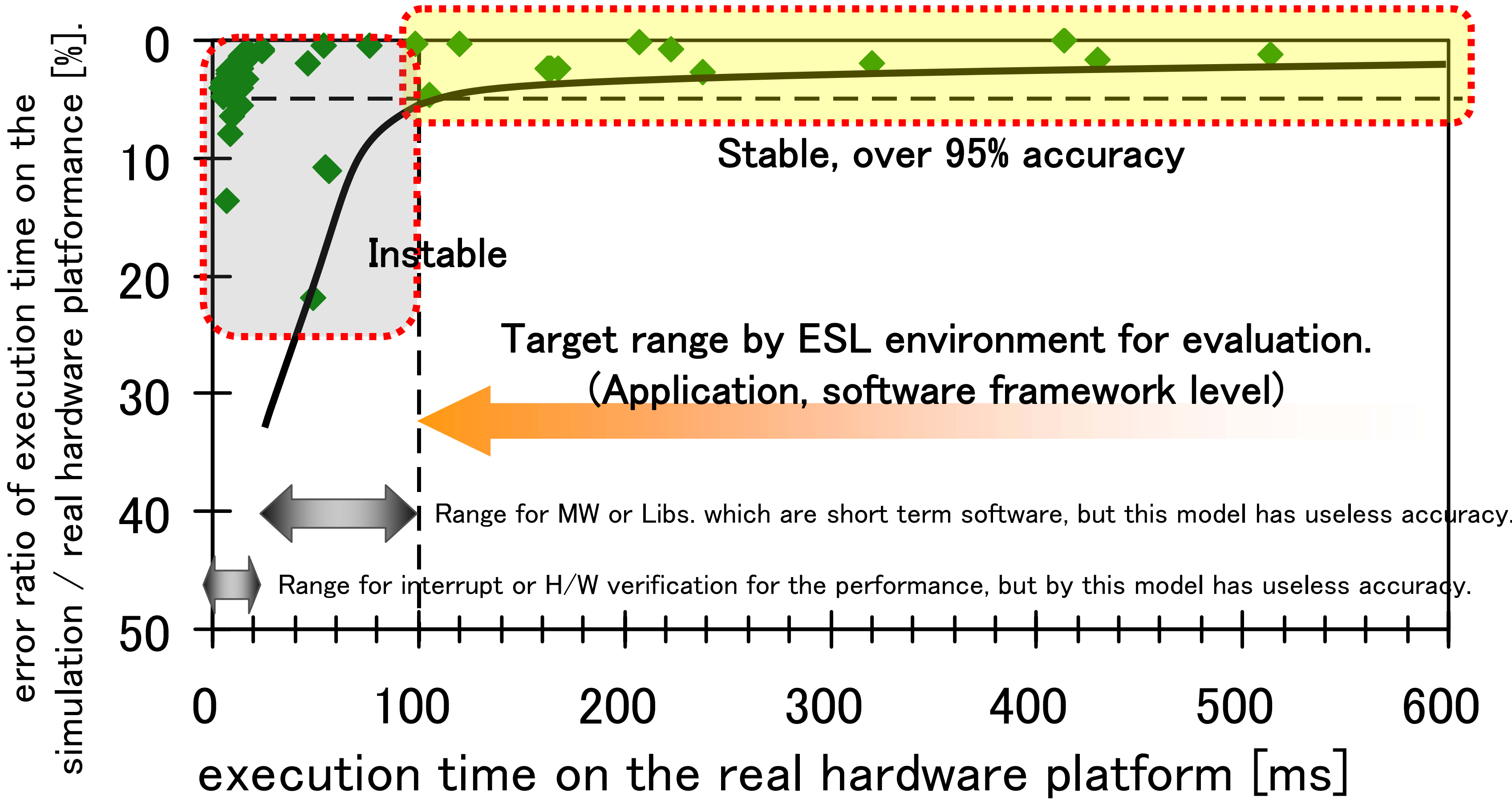
1) Actual multicore processor



2) ESL based combining model

3.Software centric system design methodology

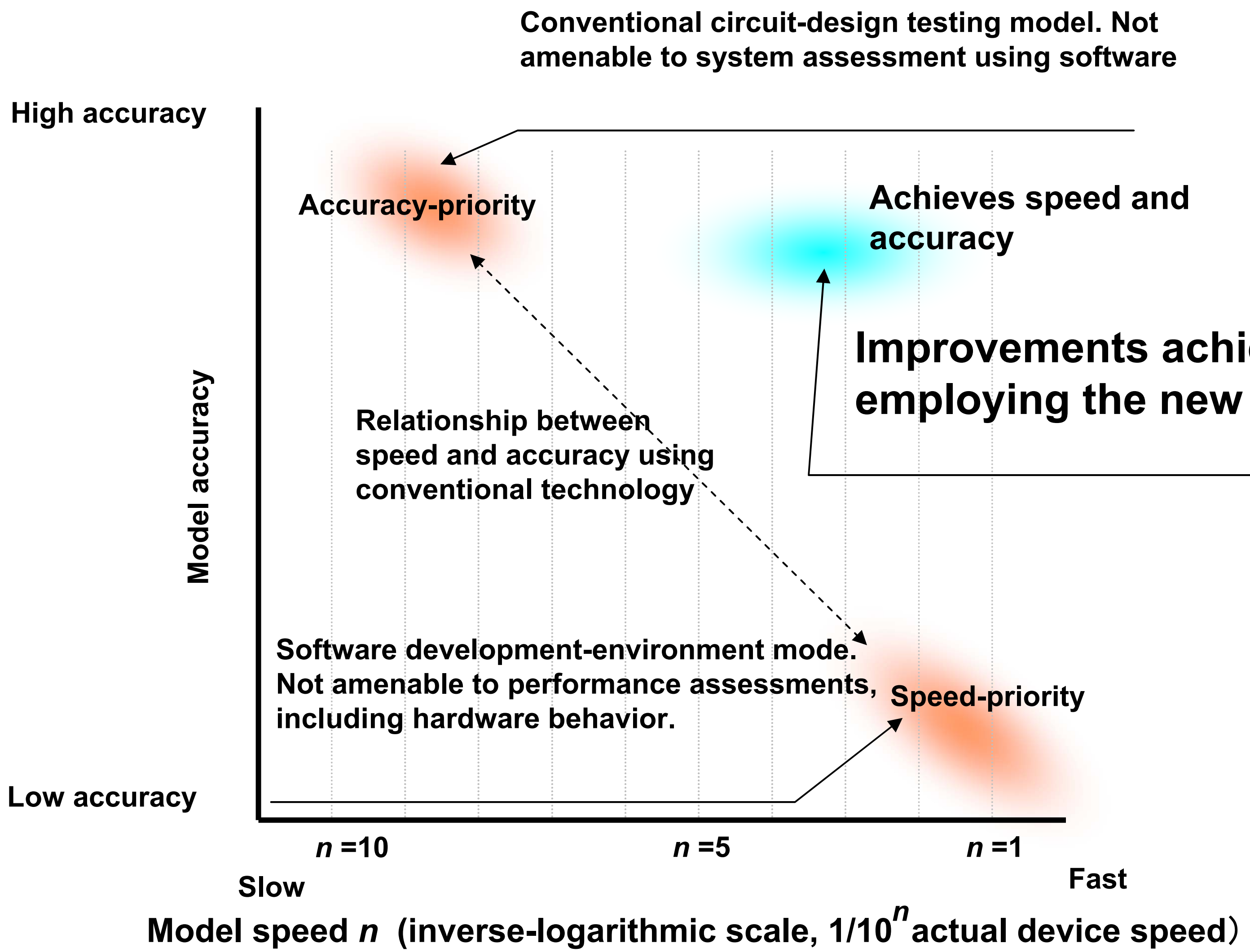
- Verification of accuracy by EEMBC benchmark between manufactured processor and ESL model



50 kinds of benchmarks of EEMBC which are running various execution time.

Comparison result of Fujitsu FR550 multicore processor manufactured SoC and speed/accuracy mixtured ESL model

3.Software centric system design methodology



http://www.fujitsu.com/global/news/pr/archives/month/2009/20090414-02.html