System Level Performance Analysis and Optimization for The Adaptive Clocking based Multi-Core Processor

Byung Su Kim[1], Joon-Sung Yang[2]

[1]Design Technology Team, Foundry, Samsung Electronics, Korea

[2]Department of Semiconductor and Display Engineering, Sungkyunkwan University, Korea

bs2014.kim@skku.edu, js.yang@skku.edu

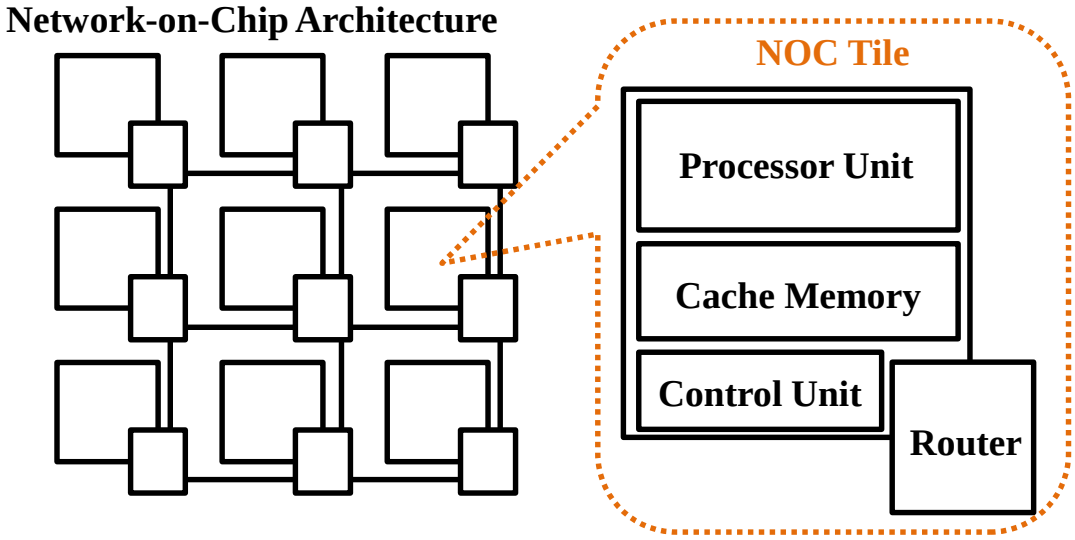
Robust to variation

- Process / Environments variation explosion due to
  - Advanced technology scaling
    - 10nm, 7nm FinFET
    - Double patterning
  - Use wide voltage range
    - DVFS (Dynamic Voltage Frequency Scaling)
    - NTC (Near Threshold Computing)
  - Life time degradation
    - Transistor (NBTI, TDDB, HCI) / Battery
    - It can not be presented at release date, so it make potential risk
  - Worst case design is too pessimistic
    - Binning
    - Adaptive / Reactive Clocking

Many / Multi Core Processors

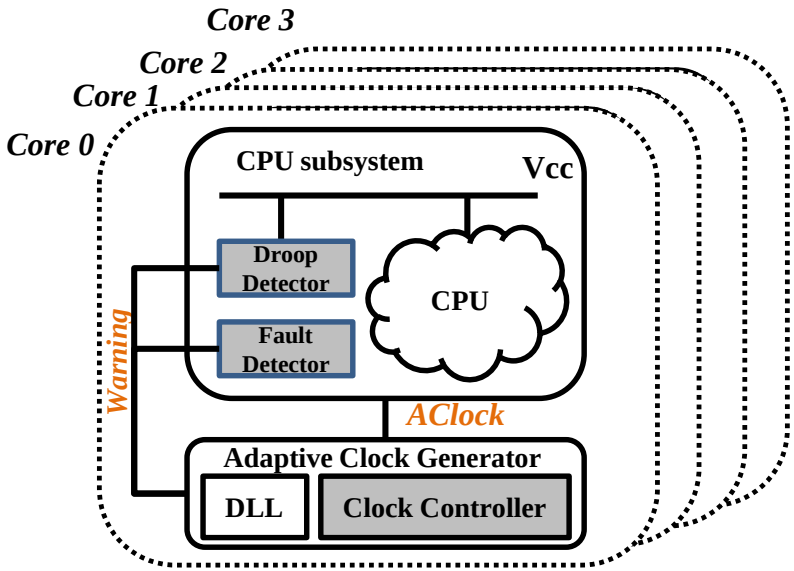
- The Many/Multi core based design is becoming a distributed computing system.
  - Each core has own cache memory and communicates with other core and main memory by global bus system.
  - For reducing cache coherency and communication efforts, each core is operated like independent system.

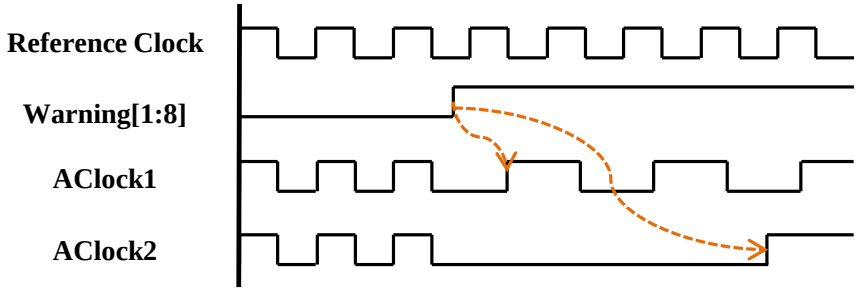
- Furthermore, NoC (Network On Chip) based many/multi core design is presented.



Adaptive Clocking Based Design Methodology

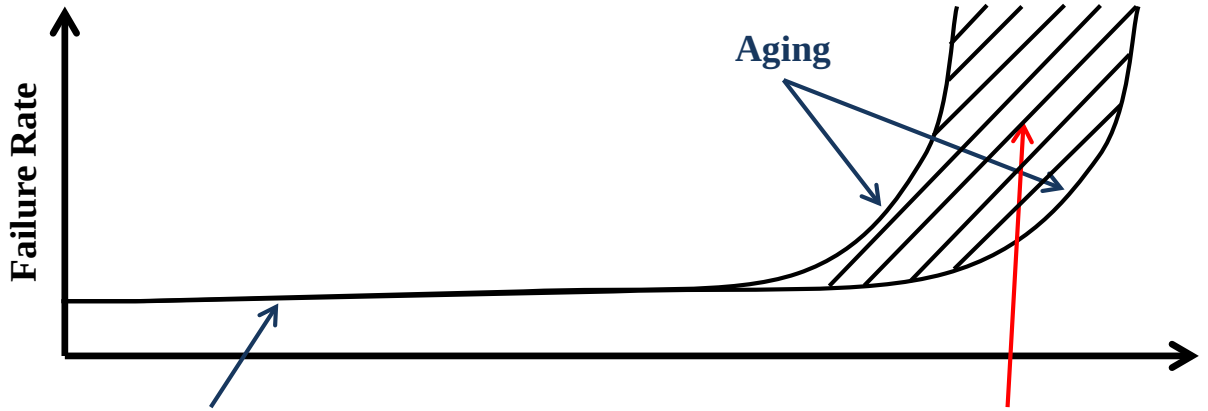
- Adaptive techniques tune system parameters based on variations in silicon-grade and ambient conditions.
  - Voltage Droop Detector (Intel, IBM)
  - Fault Detector (Canary FF, LASER)
  - Ring Oscillator / Critical Path replica circuit
  - Global clock and voltage control





Difficulty of system level analysis

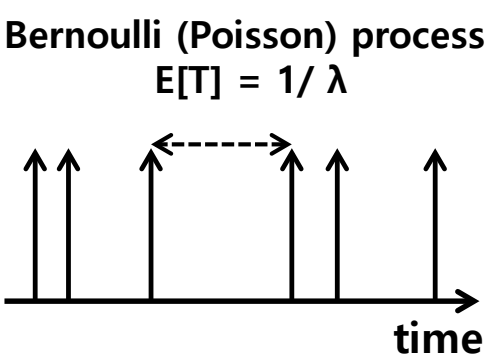
- How often does adaptive clocking operate?
  - When chip is fresh, Failure rate is very low.
  - If environments is changed like transistor / battery aging, Adaptive clocking operation is often worked.

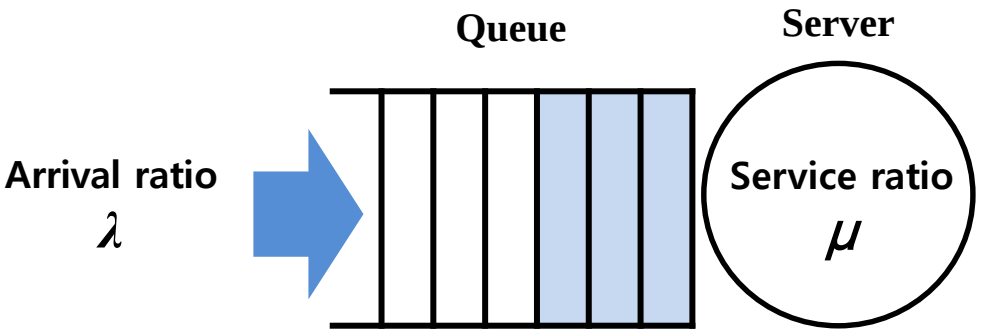


- Many/Multi-core design with the adaptive clocking
  - Each processor core may have a different clock speed.
  - Homogeneous system is changed to non-homogeneous system.
  - Need average (typical) system performance estimation / optimization method

Introduction of Queuing theory

- M/M/1 Queue
  - The M/M/1 Queue is made of a Poisson arrival, one exponential (Poisson) server, FIFO queue of unlimited capacity and unlimited customer population.
  - Easily calculate average (typical) system performance for design exploration





Utilization $\rho = \frac{\lambda}{\mu}$

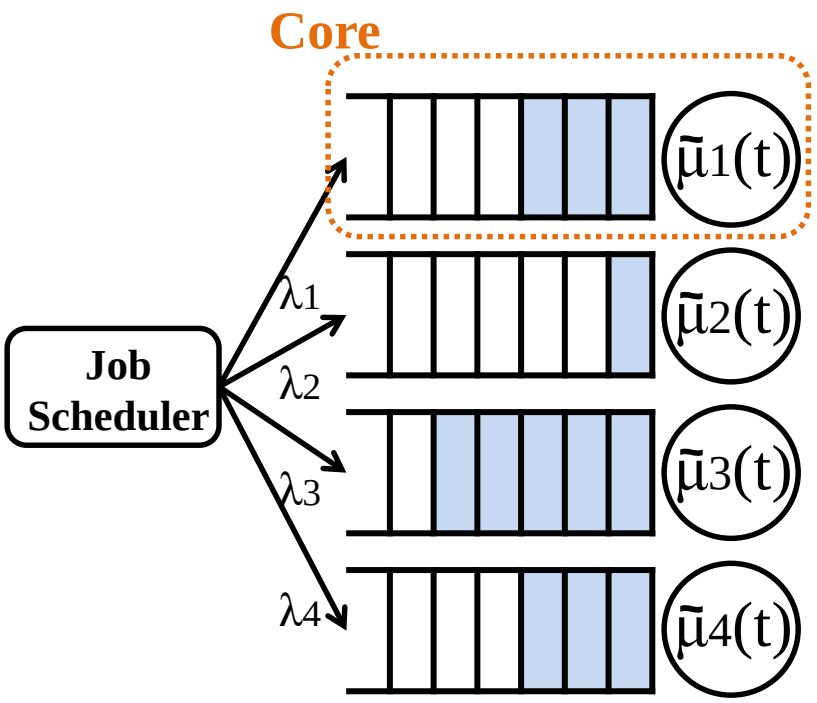
Number of Jobs $N = \frac{\rho}{1-\rho}$

Average response time $T = \frac{N}{\lambda} = \frac{1}{\mu - \lambda}$

By Little's Laws

Analytical Model for the Adaptive Clocking

- NoC (Network On Chip) based many/multi core processors is presented by distributed computing model.
- Symbols and notations used in analytic modeling



Notation	Description
$\alpha_i(t)$	A performance adaptive factor
λ_i, λ	Job arrival rate for queue and system(jobs/s)
μ_i	Initial service rate for server (jobs/s)
$\tilde{\mu}_i(t)$	$\frac{\mu_i}{\alpha_i(t)}$, Adapted service rate of server (jobs/s)
$\tilde{\rho}_i(t)$	$\frac{\lambda_i}{\tilde{\mu}_i(t)}$, Adapted utilization factor of core
$N_i(t), N(t)$	Average number of jobs in i^{th} core and system
$T_i(t), T(t)$	Response time of i^{th} core and system
$P_i(t), P(t)$	Power consumption of i^{th} core and system
P_{LE}	Leakage power consumption of core when core is *enabled*
P_{LD}	Leakage power consumption of core when core is *disabled*

Analytical Model for the Adaptive Clocking

- Core's performance modeling

Number of Jobs

$$N_i(t) = \frac{\tilde{\rho}_i(t)}{1-\tilde{\rho}_i(t)} = \frac{\lambda_i}{\tilde{\mu}_i(t)-\lambda_i} = \frac{\lambda_i \cdot \alpha_i(t)}{\mu_i - \lambda_i \cdot \alpha_i(t)}$$

Average response time

$$T_i(t) = \frac{N_i(t)}{\lambda_i} = \frac{1}{\tilde{\mu}_i(t)-\lambda_i} = \frac{\alpha_i(t)}{\mu_i - \lambda_i \cdot \alpha_i(t)}$$

Average power consumption

$$\begin{aligned} P_i(t) &= \rho_i(t) \cdot C_{core} \cdot freq(t) \cdot vdd^2 \\ &\quad + \rho_i(t) \cdot P_{LE} + (1-\rho_i(t)) P_{LD} \\ &= \lambda_i \cdot C_{core} \cdot \xi \cdot vdd^2 + P_{LE} + \frac{\lambda_i}{\tilde{\mu}_i(t)} \left(P_{LE} - P_{LD}\right) \end{aligned}$$

$$\left(\because \xi = \frac{freq(t)}{\tilde{\mu}_i(t)} = \frac{freq \cdot \alpha_i(t)}{Const * freq \cdot \alpha_i(t)} = \frac{1}{Const} \right)$$

Analytical Model for the Adaptive Clocking

- Systems' performance modeling

Number of Jobs

$$N(t) = \sum_{i=1}^{m} N_i(t)$$

Average response time

$$T(t) = \frac{\lambda_1}{\lambda} T_1(t) + \frac{\lambda_2}{\lambda} T_2(t) + \ldots + \frac{\lambda_m}{\lambda} T_m(t) = \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} T_i(t)$$

$$\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_m$$

Average power consumption

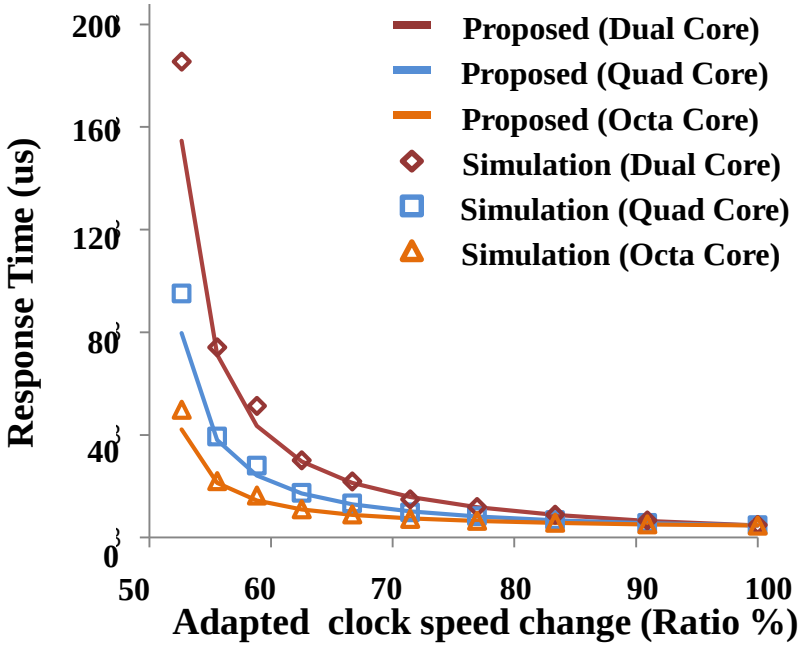
$$P(t) = \sum_{i=1}^{m} P_i(t)$$

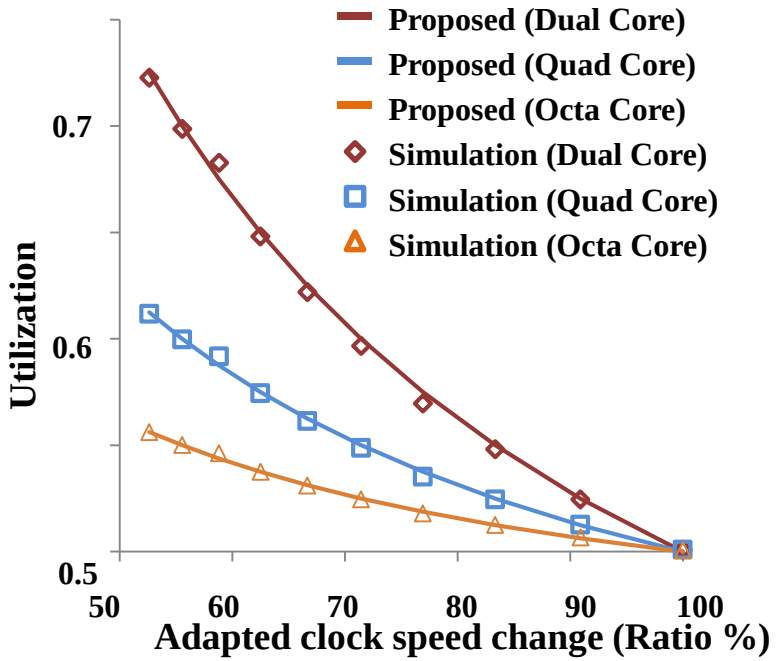
$$= \sum_{i=1}^{m} \lambda_i \cdot C_{core} \cdot \xi \cdot vdd^2 + P_{LD} + \frac{\lambda_i}{\tilde{\mu}_i(t)} \left(P_{LE} - P_{LD} \right)$$

$$= \lambda \cdot C_{core} \cdot \xi \cdot vdd^2 + m \cdot P_{LD} + \sum_{i=1}^{m} \frac{\lambda_i}{\tilde{\mu}_i(t)} \left(P_{LE} - P_{LD} \right)$$

Experimental Results - Analytical Model

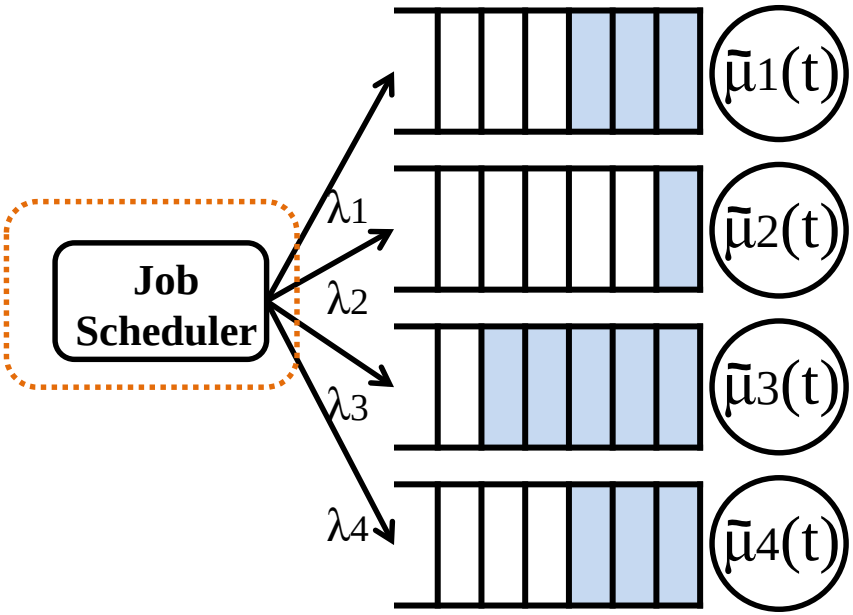
- Model evaluation using Dual/Quad/Octa Core processor
  - Compared with a JMT Computer / Network modeling and simulation tools (Monte-Carlo Simulation Based Simulator)
  - Each core can process 5.8DMIPS/Mhz (2.8Ghz and 64GB/s for 64-bit instruction)
  - Only one core's clock speed varies from 2.8 GHz (100%) to 1.4 GHz (50%)





Optimal Job Scheduling Method

- Original job distributor use R.R. (Round Robin) method because all core have same service time → need new policy (heterogeneous system)



Minimize $T(\lambda_1, \lambda_2, ..., \lambda_m)$

Subject to

$P(\lambda_1, \lambda_2, ..., \lambda_m) \leq P_{\text{target}}$

$\forall \lambda_i \geq 0$

$\forall \lambda_i < \tilde{\mu}_i(t)$

$\lambda = \lambda_1 + \lambda_2 + ... + \lambda_m$

Optimal Job Scheduling Method

- New closed form method
  - Change average number of jobs minimization problem
  - Not only minimizes the average response time but also the average power.

$$T(t) = \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} T_i(t) = \frac{1}{\lambda} \sum_{i=1}^{m} N_i(t)$$

$$N_i(t) = N_{Q,i}(t) + N_{S,i}(t)$$
$$= \lambda_i W_i + \lambda_i S_i$$
$$= \lambda_i W_i + \frac{\lambda_i}{\tilde{\mu}_i(t)}$$

$$P(t) \propto \sum_{i=1}^{m} \tilde{\rho}_i(t) = \sum_{i=1}^{m} \frac{\lambda_i}{\tilde{\mu}_i(t)}$$

N_Q : avg. number of job in queue
N_S : avg. number of job in server
W : avg. waiting time in the queue
S : avg. service time in in the server

$$T(t) = \sum_{i=1}^{m} \lambda_i W_i + \sum_{i=1}^{m} \frac{\lambda_i}{\tilde{\mu}_i(t)} \propto N_Q(t) + P(t)$$

Optimal Job Scheduling Method

- New closed form method
  - use the inequality of arithmetic and geometric means (in-direct solution)

$$T(t) = \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} T_i(t) = \frac{1}{\lambda} \sum_{i=1}^{m} N_i(t) \propto \tilde{T}(t) = \sum_{i=1}^{m} N_i(t)$$

a minimum condition of equation by holding equality if and only if $N_1 = N_2 = \ldots = N_m$.

$$\sum_{i=1}^{m} N_i = N_1(t) + N_2(t) + \ldots + N_m(t) \geq m \cdot \sqrt[m]{N_1(t) N_2(t) \ldots N_m(t)}$$

Optimal Job Scheduling Method

- New closed form method
  - use the inequality of arithmetic and geometric means (in-direct solution)

$$N_1 = N_2 = .. = N_m = N_{\min}$$

$$\frac{\lambda_1}{\tilde{\mu}_1(t) - \lambda_1} = N_{\min}$$

$$\frac{\lambda_2}{\tilde{\mu}_2(t) - \lambda_2} = N_{\min}$$

$$\ldots$$

$$\frac{\lambda_m}{\tilde{\mu}_m(t) - \lambda_m} = N_{\min}$$

$$\lambda_1 = N_{\min}\left(\tilde{\mu}_1(t) - \lambda_1\right)$$

$$\lambda_2 = N_{\min}\left(\tilde{\mu}_2(t) - \lambda_2\right)$$

$$\ldots$$

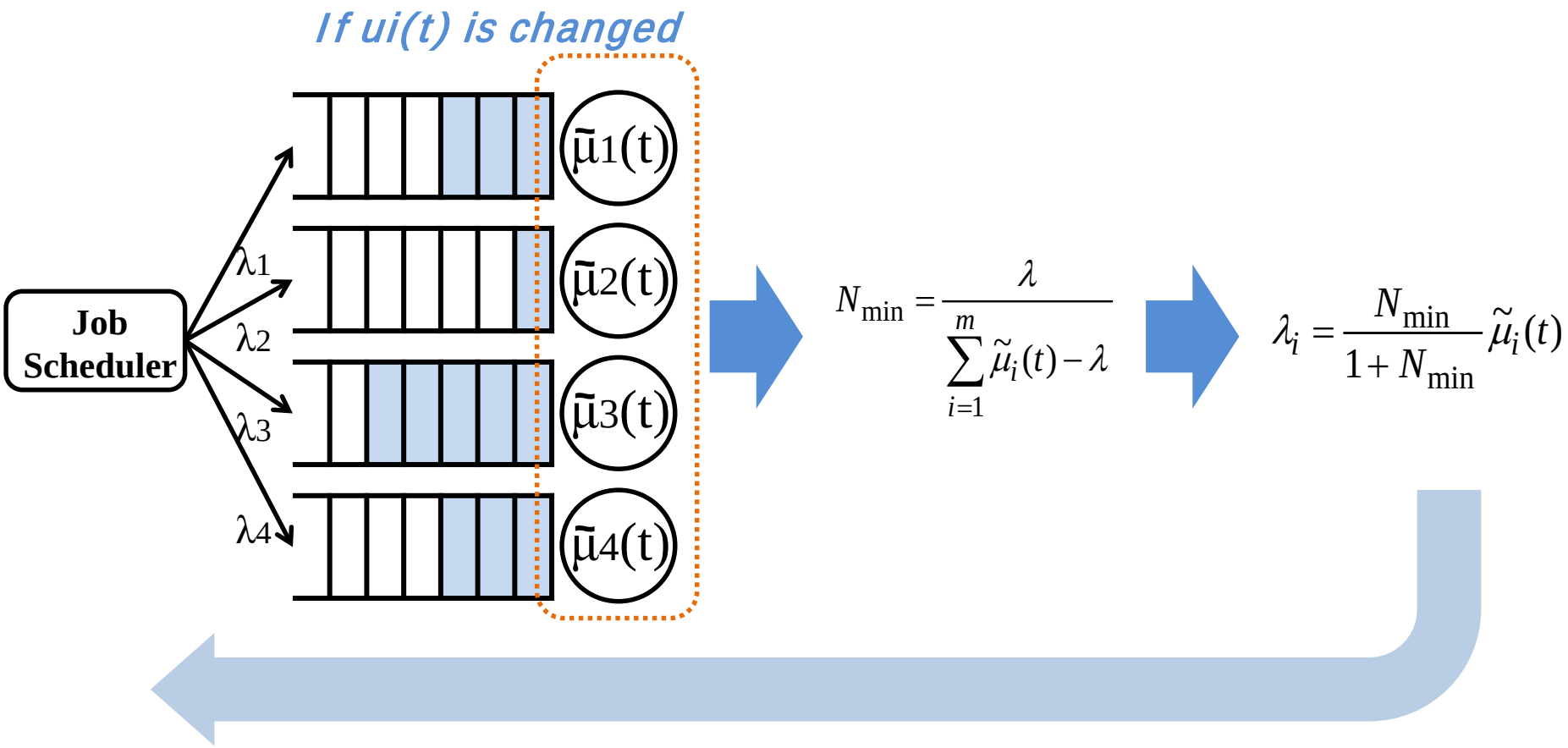
$$\lambda_m = N_{\min}\left(\tilde{\mu}_m(t) - \lambda_m\right)$$

$$\sum_{i=1}^{m} \lambda_i = N_{\min}\left(\sum_{i=1}^{m} \tilde{\mu}_i(t) - \sum_{i=1}^{m} \lambda_i\right)$$

$$\lambda$$

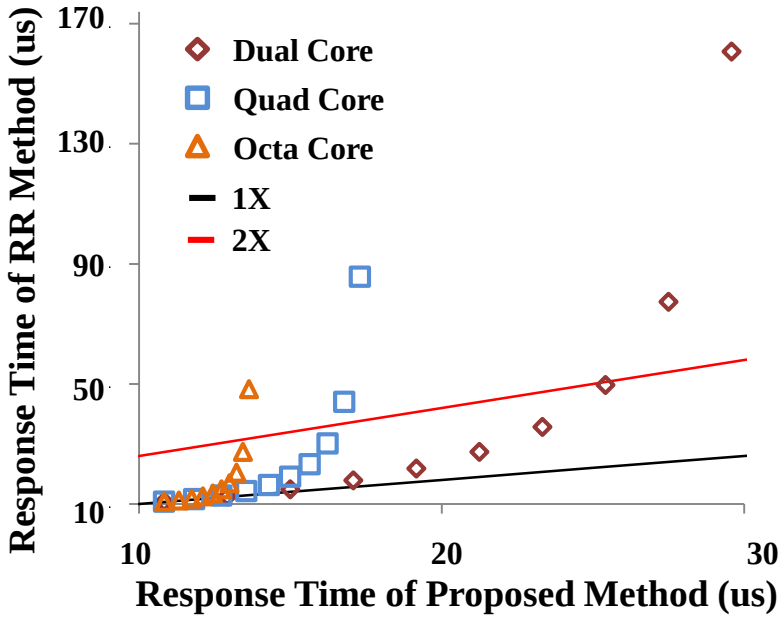
Optimal Job Scheduling Method

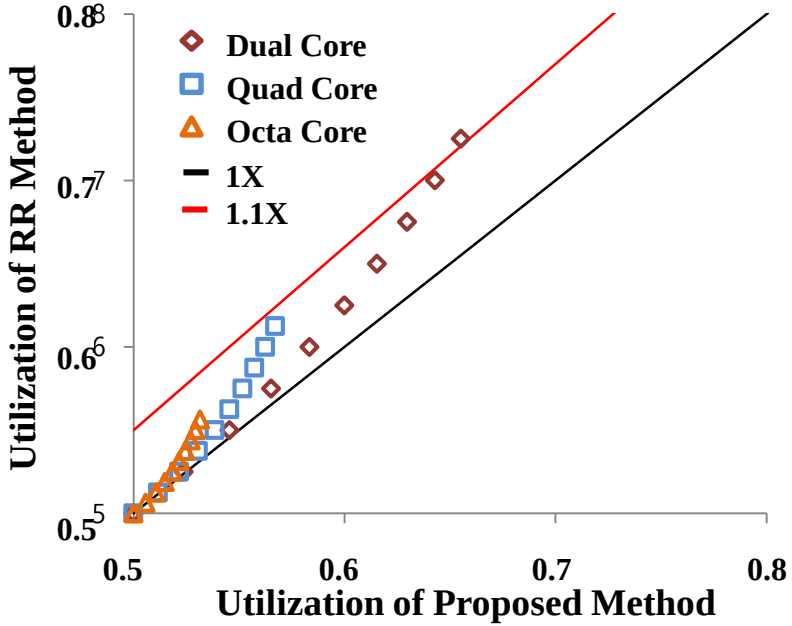
- New closed form method
  - use the inequality of arithmetic and geometric means (in-direct solution)



Experimental Results – Job Scheduling

- Optimal job scheduling evaluation using Dual/Quad/Octa Core processor
  - Compared between R.R and our proposed job scheduler
  - Each core can process 5.8DMIPS/Mhz (2.8Ghz and 64GB/s for 64-bit instruction)
  - Only one core's clock speed varies from 2.8 GHz (100%) to 1.4 GHz (50%)





Experimental Results – Job Scheduling

- Analyze timing failures in Dual Core processor by adaptive clocking operations
  - Compared between R.R and our proposed job scheduler
  - Use different initial utilization (0.4~0.8)
  - Use clock speed change (1~1.9X, Adaptive factor)

round-robin job scheduling

*1	Initial Utilization				
	0.4	0.5	0.6	0.7	0.8
1.0	1.30E-05	1.56E-05	1.95E-05	2.60E-05	3.91E-05
1.1	1.42E-05	1.74E-05	2.24E-05	3.17E-05	5.53E-05
1.2	1.55E-05	1.95E-05	2.65E-05	4.23E-05	1.37E-04
1.3	1.71E-05	2.23E-05	3.28E-05	6.94E-05	-1.07E-04
1.4	1.89E-05	2.60E-05	4.39E-05	2.86E-04	-2.60E-05
1.5	2.12E-05	3.13E-05	6.84E-05	-1.04E-04	-9.77E-06
1.6	2.39E-05	3.91E-05	1.66E-04	-3.91E-05	-2.79E-06
1.7	2.73E-05	5.21E-05	-3.22E-04	-2.19E-05	1.09E-06
1.8	3.16E-05	7.81E-05	-7.81E-05	-1.40E-05	3.55E-06
1.9	3.74E-05	1.56E-04	-4.32E-05	-9.47E-06	5.26E-06
Diff*2	2.88	10.0	N/A	N/A	N/A

buffer overflow (system failure)

Proposed job scheduling

*1	Initial Utilization				
	0.4	0.5	0.6	0.7	0.8
1.0	1.30E-05	1.56E-05	1.95E-05	2.60E-05	3.91E-05
1.1	1.41E-05	1.72E-05	2.20E-05	3.07E-05	5.06E-05
1.2	1.51E-05	1.88E-05	2.47E-05	3.61E-05	6.70E-05
1.3	1.61E-05	2.03E-05	2.74E-05	4.23E-05	9.23E-05
1.4	1.71E-05	2.19E-05	3.04E-05	4.97E-05	1.37E-04
1.5	1.80E-05	2.34E-05	3.35E-05	5.86E-05	2.34E-04
1.6	1.89E-05	2.50E-05	3.68E-05	6.94E-05	6.25E-04
1.7	1.98E-05	2.66E-05	4.02E-05	8.30E-05	-1.33E-03
1.8	2.07E-05	2.81E-05	4.39E-05	1.00E-04	-3.52E-04
1.9	2.15E-05	2.97E-05	4.79E-05	1.24E-04	-2.12E-04
Diff*2	1.65	1.90	2.45	4.57	N/A

*1: change of clock speed, an adaptive factor (times)
*2: Average response time of 1.9 / Average response time of 1.0 at each initial utilization (times)

Conclusions

- System level analysis is essential
  - The adaptive clocking method is good solution for In-situ variations.
  - Transistor / battery aging accurate adaptive clocking operation because failure rate is increased.
  - System level analysis can help architecture or design planning in the early stage of system development

- Average (typical) system performance estimation / optimization method
  - Propose a queueing theory based analytical model
  - An optimal job scheduling method using the inequality of arithmetic and geometric means