Using Randomized Rounding to Satisfy Timing Constraints of Real-Time Preemptive Tasks

Anupam Datta
Computer Science Dept.,
Stanford University, U.S.A
danupam@stanford.edu

Sidharth Choudhury
Dept. of Computer Sciences,
University of Texas at Austin, U.S.A
sidharth@cs.utexas.edu

Anupam Basu
Dept. of Computer Science & Engg.,
IIT Kharagpur, India
anupam@cse.iitkgp.ernet.in

Abstract

In preemptive real-time systems, a tighter estimate of the Worst Case Response Time(WCRT) of the tasks can be obtained if the layout of the tasks in memory is included in the estimation procedure. This is because the Cache Related Preemption Delay(CRPD) depends on the inter-task interference in the cache. We develop a response time analysis framework which takes the layout of the tasks into account. We present an ILP formulation which generates a layout of the tasks such that all timing constraints are satisfied. To overcome the issue of non-scalability associated with an ILP solution, we also present a linear programming relaxation of the ILP formulation, which offers an approximate solution. The performance of the proposed formulation is demonstrated.

1. Introduction

In a real-time system, it is essential to guarantee that all tasks meet their timing deadlines. This can be done by estimating the Worst Case Response Times(WCRTs) of the tasks, and verifying that they are less than the corresponding deadlines. In order to get a tighter estimate of the WCRTs, the effect of the instruction cache has to be included in the analysis. This causes some immediate complications. Let us assume that an executing task is preempted. When it resumes execution, it has to reload into the cache all those blocks which were displaced by higher priority tasks in the intervening period. This additional time, called the Cache Related Preemption Delay(CRPD) should also be incorporated into the WCRTs of the tasks. Response time analysis techniques, which take into account the CRPD are presented in [2] and [3].

We observe that the CRPD depends on the layout of the tasks in the memory, i.e., the actual address assignment to the code. Including the task layout in the estimation of the CRPD thus results in a tighter bound on the WCRTs. Task sets which are not schedulable under an arbitrarily chosen layout may become schedulable when the layout is decided more carefully. We develop a response time analysis framework which takes the layout of the tasks into account. The objective is to find a layout of the tasks in memory (if it exists) such that the timing constraints of all the tasks are met. An Integer Linear Programming (ILP) formulation of the problem is detailed in [13]. While achieving the desired solution to the constraint satisfaction problem, that approach had an inherent drawback since ILP is NP-hard and all standard solutions take exponential time in the worst case. A dynamic programming based solution to the same problem is presented in [14]. Although that approach performs quite well in practice, its worst case running time is exponential. In this paper, we present an alternative approach for the solution of the problem. We solve a Linear Programming (LP) relaxation of the original problem and then apply the technique of *randomized rounding* ([4]) to restore integrality. Since LP is solvable in polynomial time ([5]), this approach scales up quite well.

2. Related Work

The problem of estimating the WCRT of tasks (including the effect of the cache) in real-time systems has been addressed in related research. There are two aspects of this problem.

The first deals with estimating the Worst Case Execution Time (WCET) of a task in a single task environment. This problem is, in general, undecidable and is equivalent to the halting problem ([7]). Some approaches for the solution of this problem are presented in [8], and [9].

The second aspect of this problem is that of estimating the inter-task interference in the instruction cache and is the focus of this paper. This requires incorporating the cache related preemption cost into the schedulability analysis. For fixed priority scheduling, the two main schedulability analysis techniques are the utilization bound approach ([10]) and the response time approach ([11], [12]).

In [1], Basumallick and Nilsen extend the rate monotonic analysis to take into account the effect of the inter-task interference. Another technique, based on the response time approach, is presented by Busquets-Mataix et al. in [2]. They use the following equation to compute the WCRT of a task:

$$R_i = C_i + \Sigma_{j \epsilon hp(i)} \lceil \frac{R_i}{T_j} \rceil (C_j + \gamma_j)$$

Here, R_i = WCRT of τ_i
T_i = Period of τ_i
hp(i) = Set of tasks which have higher priority than τ_i

γ_j is the cache related preemption cost that task τ_j might impose on lower priority tasks. It is given by the product of the cache refill time and the number of cache blocks used by τ_j. In [3], the concept of *usefulness* of cache blocks is introduced. A *useful* cache block at an execution point is defined as a cache block that contains a memory block that may be rereferenced before it is replaced by another cache block. The number of useful cache blocks at an execution point is taken as an upper bound on the cache-related preemption cost that may be incurred if the task is preempted at that point. The augmented response time equation used in [3] is given by:

$$R_i = C_i + \Sigma_{j \epsilon hp(i)} \lceil \frac{R_i}{T_j} \rceil C_j + PC_i(R_i)$$

where $PC_i(R_i)$ is the total CRPD of task τ_i during R_i, i.e., the total cache reloading times of $\tau_1, \tau_2, \cdots, \tau_i$ during R_i.

There is a pertinent issue which has not been addressed in any of these approaches. It is to be noted that if a cache block used by a task is not used by any other task which has a higher priority, then that cache block will not contribute to the CRPD of that task. Thus, increasing the number of such blocks for a task will effect a reduction in the WCRT of that task. Also, if we are able to identify the number of such blocks, a tighter estimate of the WCRT can be obtained. We observe that this can be achieved by placing the tasks in memory in such a way that their interference in the cache is reduced.

3. Preliminaries

3.1. Randomized Rounding

Randomized rounding([4]) is a probabilistic technique to construct a provably good solution for a $0-1$ integer program from a solution of its rational relaxation. We now give a general outline of the technique. Let Π_I be a $0-1$ linear program with $x_i \epsilon \{0,1\}$. Let Π_R be its rational relaxation with $x_i \epsilon [0,1]$. The basic algorithm consists of the following two phases: a) solve Π_R; let the variables take on values $\hat{x_i} \epsilon [0,1]$; b) set the variables x_i randomly to one or zero according to the following rule: $Pr[x_i = 1] = \hat{x_i}$. The idea is to ensure that $E[x_i = 1] = \hat{x_i}$.

3.2. Execution Environment

We make the following assumptions about the execution environment: a) the priorities of the tasks are static and decreases successively from τ_0 to τ_{N-1}; b) a priority based preemptive scheduling policy is followed; c) all the tasks are periodic; d) the deadline of each task is equal to its period; e) the worst case execution time (WCET) excluding the cache related preemption delay is known for each of the tasks; f) the instruction cache is direct mapped; g) the placement of the tasks in memory is static; h) all the memory blocks used by a task are contiguous. This paper focuses only on the instruction cache, and effects of the data cache are ignored.

3.3. Response Time Equation

The response time equation for the i^{th} task is given by:

$$R_i = C_i + \sum_{j<i} \lceil \frac{R_i}{T_j} \rceil (C_j + \gamma_{ij}) \quad (1)$$

where R_i denotes the WCRT for τ_i, C_i the WCET for τ_i without preemption, T_j the time period for τ_j, and γ_{ij} the CRPD that task τ_j imposes on the lower priority tasks $(\tau_{j+1}, \tau_{j+2}, \cdots, \tau_i)$.

We define γ_{ij} as follows. Let U_k be the set of cache blocks that are used by task τ_k and $|U_k|$ denote the number of elements in the set U_k. Then γ_{ij} is given by:

$$\gamma_{ij} = CRT \times |U_j \cap (U_{j+1} \cup U_{j+2} \cup \cdots \cup U_i)| \quad (2)$$

where CRT denotes the cache refill time. Thus, to obtain the CRPD imposed by τ_j during the response time of τ_i, we compute the number of cache blocks that are used by τ_j and *atleast* one of $\tau_{j+1}, \tau_{j+2}, \cdots, \tau_i$. This number gives an upper bound on the number of cache blocks that τ_j causes to be reloaded in this time. That number multiplied

by the cache refill time yields the desired upper bound on the CRPD.

We observe that γ_{ij} depends on the layout of the tasks $\tau_j, \tau_{j+1}, \cdots, \tau_i$ in memory. An attempt is made to generate a layout such that the timing constraints of all tasks are met.

4. An ILP Formulation to Find Task Layout in Cache

In this section, an ILP formulation is presented which finds a layout of the tasks in the cache. Since the reverse mapping from cache to memory is trivial, this is quite sufficient for our needs. Also, since the cache is much smaller than the memory, it helps reduce the number of variables in the formulation substantially.

Let L denote the number of blocks in the instruction cache, M the number of blocks in the primary memory, N the number of tasks, and $s(i)$ the size of τ_i. Then, the objective function of the ILP formulation to be minimized is defined by the following formula:

$$R_i^{k+1} = C_i + \sum_{j<i} \lceil \frac{R_i^k}{T_j} \rceil (C_j + \gamma_{ij}) \tag{3}$$

Below we describe a set of constraints all of which have to be satisfied.

First, we define new variables $y(i,l)$'s. $y(i,l)$ is 1 if τ_i is placed starting from the l^{th} cache block. Otherwise, $y(i,l)$ is 0. From this definition, we derive the following constraints:

$$y(i,l) \in \{0,1\} \tag{4}$$

$$\sum_{l=0}^{L-1} y(i,l) = 1 \tag{5}$$

Next, we define a matrix $b(i,l,k)$ whose elements are constants where $0 \leq l \leq L-1$ and $0 \leq k \leq L-1$. $b(i,l,k)$ is 1 if the k^{th} cache block contains the code of τ_i, where τ_i is placed in cache starting from the l^{th} block.

Define $c(i,k) = 1$ if cache block k is used by task τ_i. We have the following constraints.

$$c(i,k) \epsilon \{0,1\} \tag{6}$$

$$c(i,k) = \sum_{l=0}^{L-1} b(i,l,k) y(i,l) \tag{7}$$

We further define variables $e(i,j,k)$'s. $e(i,j,k)$ is 1 if the k^{th} cache block is used by atleast one task whose priority is less than that of τ_j and is not less than that of τ_i, i.e., by atleast one of the tasks $\tau_{j+1}, \tau_{j+2}, \cdots, \tau_i$. Thus, $e(i,j,k)$ is 1 if and only if $\sum_{h=j+1}^{i} c(h,k)$ is greater than 0. Otherwise, $e(i,j,k)$ is 0. Now we derive the following constraints:

$$e(i,j,k) \in \{0,1\} \tag{8}$$

$$\sum_{h=j+1}^{i} c(h,k) - e(i,j,k) \geq 0 \tag{9}$$

$$\sum_{h=j+1}^{i} c(h,k) - e(i,j,k).(N-1) \leq 0 \tag{10}$$

We define variables $f(i,j,k)$'s. $f(i,j,k)$ is 1 if the k^{th} cache block is used by τ_j and atleast by one of the tasks $\tau_{j+1}, \tau_{j+2}, \cdots, \tau_i$. Otherwise, $f(i,j,k)$ is 0. In other words, $f(i,j,k)$ is 1 if both $c(j,k)$ and $e(i,j,k)$ are 1. Consequently, the following constraints are arrived at:

$$f(i,j,k) \in \{0,1\} \tag{11}$$

$$c(j,k) + e(i,j,k) - 2f(i,j,k) \geq 0 \tag{12}$$

$$c(j,k) + e(i,j,k) - f(i,j,k) \leq 1 \tag{13}$$

We define $g(i,j)$ as the total number of cache blocks used by τ_j *and* atleast one of $\tau_{j+1}, \tau_{j+2}, \cdots, \tau_i$. Thus, to use the notation introduced previously, $g(i,j) = |U_j \cap (U_{j+1} \cup U_{j+2} \cup \cdots \cup U_i)|$. Clearly, $g(i,j)$ is defined as follows:

$$g(i,j) = \sum_{k=0}^{L-1} f(i,j,k) \tag{14}$$

Then, we reach the definition of the CRPD γ_{ij} as shown below:

$$\gamma_{ij} = CRT \times g(i,j) \tag{15}$$

The other constraints are obtained from the task deadlines and the response time equations as follows:

$$R_i^{k+1} = C_i + \sum_{j<i} \lceil \frac{R_i^k}{T_j} \rceil (C_j + \gamma_{ij}) \tag{16}$$

$$R_i^{k+1} \leq T_i \tag{17}$$

4.1. The Complete Procedure

The iterative procedure used for computing the response times is detailed below.

1. $\forall i, R_i^0 = C_i$
2. $k = 0$
3. Solve the ILP problem presented in this section to obtain a layout and R_i^{k+1} for all tasks.
4. Terminte if $R_i^{k+1} = R_i^k$ for all tasks. Otherwise, set $k = k+1$ and goto step 3.

Thus, this procedure terminates when all the R_i's converge to stable values. These values of the R_i's are compared with the deadlines of the corresponding tasks to verify the schedulability of the task set.

5. An Approximation Approach

We note that the formulation presented in Section 4 requires the ILP problem to be solved iteratively a number of times. In this section, we propose an approximation technique in which the ILP problem is solved only once. It is to be noted that if the task set is schedulable, then R_i is less than T_i. Thus, substituting T_i for R_i in formula (1), we obtain a more pessimistic estimate of the response time of τ_i. If this estimate satisfies the timing constraints, then the actual task will certainly meet its deadline. Since T_i is a constant, this substitution helps transform the recursive equation into a linear one which is amenable to an ILP formulation.

5.1. An Alternative ILP Formulation of the Problem

This formulation is also along the same lines as the one presented in the previous section. The new objective function is defined by:

$$R_i = C_i + \sum_{j<i} \lceil \frac{T_i}{T_j} \rceil (C_j + \gamma_{ij}) \tag{18}$$

The new ILP formulation inherits the constraints (3)–(14) from the previous one. Instead of constraints (15) and (16), the following two constraints are derived:

$$R_i = C_i + \sum_{j<i} \lceil \frac{T_i}{T_j} \rceil (C_j + \gamma_{ij}) \tag{19}$$

$$R_i \leq T_i \tag{20}$$

6. Linear Programming Relaxation

In this section, we present a linear programming relaxation of the ILP problem discussed in Section 5. We use the technique of randomized rounding to restore integrality. We prove that all constraints are satisfied in the expected sense. First, we note that constraints (19), (20) can be rewritten as

$$\Sigma_{j<i} w_{ij}\gamma_{ij} \leq B_i \tag{21}$$

(Here B_i is a constant.) Consider the LP problem derived from the ILP formulation where all the integrality constraints are removed and the above mentioned constraint is changed to:

$$\Sigma_{j<i} w_{ij}\gamma_{ij} \leq B_i/(N-1) \tag{22}$$

Also constraints (12) & (13) are replaced by:

$$f(i,j,k) = 1/2(c(j,k) + e(i,j,k)) \tag{23}$$

The idea is to solve this LP problem and then use randomized rounding to round the $y(i,l)'s$. Once the $y(i,l)'s$ are fixed, all other values can be computed and it can be verified whether all the timing constraints are satisfied in the original ILP problem. Thus, we desire that after applying randomized rounding constraint (1) is satisfied. Indeed, the expected value of the response time for each task obtained through this process is less than the corresponding deadline as we prove next.

Claim: $E[\Sigma_{j<i} w_{ij}\gamma_{ij}] \leq B_i$

Proof:

We will use x^* to denote the solution obtained from the LP solution for variable x .

$E[y(i,l)] = y(i,l)^*$

$E[c(i,k)] = c(i,k)^*$

$E[e(i,j,k)] \leq \Sigma_{h=j+1}^{i} c(h,k)^* \leq (i-j)e(i,j,k)^*$

$E[f(i,j,k)] \leq 1/2(E[c(j,k)] + E[e(i,j,k)]) \leq (N-1)f(i,j,k)^*$

$E[\gamma(i,j)] \leq (N-1)\gamma(i,j)^*$

$E[\Sigma_{j<i} w_{ij}\gamma_{ij}] \leq (N-1)\Sigma_{j<i} w_{ij}\gamma_{ij}^* \leq B_i$

Although a high probability bound cannot be obtained through the standard union bound method, based on a recent result [6], we conjecture that positive correlation between the inequalities can be exploited to prove a tighter bound.

7. Alternative Linear Programming Relaxation

A problem with the LP relaxation of the previous section is that scaling down the $B_i's$ by a factor of $N-1$ might make the problem infeasible. Instead, in this section we solve an LP problem where each of the $B_i's$ are scaled by a factor of λ and we change the objective function to minimize λ. Since the $B_i's$ are constants, the resulting constraints remain linear.

Thus, the new LP problem is:

minimize: λ

$\Sigma_{j<i} w_{ij}\gamma_{ij} \leq \lambda B_i$

The other constraints remain unchanged from Section 6.

The idea is to solve this LP problem and then use randomized rounding to round the $y(i,l)'s$. The following claim can be easily proved.

Claim: $E[\Sigma_{j<i} w_{ij}\gamma_{ij}] \leq (N-1)\lambda^* B_i$

Note that if $\lambda^* \leq 1/(N-1)$, then $E[\Sigma_{j<i} w_{ij}\gamma_{ij}] \leq B_i$. So, the value of λ^* is an indicator of the "chance" of getting a feasible solution to the ILP problem.

8. Experimental Results

In this section, we validate the effectiveness of the approaches presented in the previous sections by comparing the predicted WCRT with the measured/observed WCRT. We also compare our estimated WCRT values with those obtained by applying the approach proposed by Busquets-Mataix *et al* in [2].

We present simulation results for two different architectures: the TMS320C5x series of microprocessors and DEC's Alpha AXP processor. The simulators take the task layout, the WCET and periods of the tasks as input , and generate the WCRT of the tasks. These values will be referred to as the *observed WCRTs* of the tasks in the subsequent discussion. The *estimated WCRTs* were obtained by plugging into formula (1) the γ_{ij} values obtained by solving the linear programs. The linear programs were solved using the public domain software lp_solve [1]).

For TMS, we have two sets of results. In one set, the layout was obtained by solving the ILP problem described in Section 5. For the other set, the layout was obtained by solving the LP problem detailed in Section 7. The benchmark task sets used are detailed in Table 1. Three tasks were used: Matrix Multiplication (MM), a 128 taps Adaptive FIR, and a 16 point FFT. The WCET of a task includes the time taken to execute it and the blocking time of the task, i.e., the overhead imposed on it by lower priority tasks. Table 2 summarizes the experimental results. In these experiments we tried to minimize the WCRT of τ_1, while satisfying the constraints of al l other tasks. The primary memory and cache sizes where set to 150 and 40instruction memory blocks. We deliberately used a small cache to ensure that the CRPD made a significant contribution to the WCRT of a task. All timing resul ts are in units of processor cycles. We see that a tight estimate of the WCRTs of the tasks is obtained. The maximum difference between the estimated and observed WCRTs is less than 1%. Also, the observed WCRTs for the layout obtained by solving the ILP problem is only marginally better than the ones obtained by solving the LP relaxation. The approximation seems to work quite well in practice.

Table 1. Benchmark Task Sets for TMS

Task Set	Task	Period	WCET without preemption	# instruc. memory blocks
Γ_1	τ_0 : FIR	200000	115037	10
	τ_1 : FFT	600000	133422	34
Γ_2	τ_0 : MM	50000	8769	6
	τ_1 : FIR	200000	115037	10
	τ_2 : FFT	600000	133422	34

The benchmark task sets for the DEC Alpha experiments are detailed in Table 3. Three tasks were used: INSERTSORT, COMPRESS and LAPLACE. The primary memory and cache sizes were set at 512 and 128 instruction memory blocks for the first task set and at 1024 and 256 for the second. For these task sets, the ILP problem presented in Section 5 was intractable. The results obtained by solving the linear programming relaxation of Section 7 are presented in Table 4. We see that a tight estimate of the WCRTs of the tasks is obtained. The maximum difference between the estimated and observed WCRTs is less than 1.5%.

[1] ftp://ftp.es.ele.tue.nl/pub/lp_solve

Table 3. Benchmark Task Sets for DEC Alpha

Task Set	Task	Period	WCET without preemption	# instruc. memory blocks
Γ_1	τ_0 : COM	100000	74368	76
	τ_1 : LAP	500000	106928	114
Γ_2	τ_0 : INS	50000	19296	105
	τ_1 : COM	200000	74368	76
	τ_2 : LAP	600000	106928	114

Table 4. Results for DEC Alpha

Task Set	Task	WCRT (observed)	WCRT (estimated)	WCRT BM
Γ_1	τ_0 : COM	74360	74368	74368
	τ_1 : LAP	478720	479148	479148
Γ_2	τ_0 : INS	19288	19296	19296
	τ_1 : COM	130688	132310	132571
	τ_2 : LAP	535944	542756	543758

The column labelled **WCRT-BM** gives the estimated value of the WCRT of the corresponding task using the approach proposed by Busquets-Mataix *et al* in [2]. We note that in all cases, our estimation is atleast as tight as theirs. The difference between the two estimation procedures will be more tangible when the task sets include more and larger tasks.

9. Conclusions and Future Work

The cache related preemption delay in a multitasking environment depends on the layout of the tasks in memory. We have proposed a technique based on randomized rounding an LP problem to find a layout which minimizes the WCRT of one task, while satisfying the deadlines of all the other tasks. We have proved that all the constraints will be satisfied in the expected sense under this scheme. Our experimental results for two different architectures show that observed values of the WCRTs of the tasks are within 1.5% of the estimated values indicating that the estimation technique is tight. Our approach also seems to give tighter bounds than the approach presented by Busquets-Mataix *et al* in [2].

Table 2. Results for TMS

Task Set	Task	WCRT-ILP (observed)	WCRT-LP (observed)	WCRT (estimated)	WCRT-BM (estimated)
Γ_1	τ_0 : FIR	115034	115034	115037	115037
	τ_1 : FFT	363407	363423	363516	363516
Γ_2	τ_0 : MM	8766	8766	8769	8769
	τ_1 : FIR	141236	141236	141359	141362
	τ_2 : FFT	583201	583233	583839	583863

However, we have not been able to prove that the linear programming constraints will be satisfied with high probability after integrality is restored by applying the randomized rounding technique. Although there seems to be some evidence towards this claim ([6]), a complete proof is desirable. Another possible extension of this work would be to combine the idea of task layout with the notion of the usefulness of cache blocks ([3]) to obtain even tighter bounds on the WCRTs of the tasks. Also, these ideas could possibly be applied towards building a similar framework for the analysis of data cache behaviour.

References

[1] S. Basumallick, K. Nilsen, "Cache issues in real-time systems", *Proc. ACM SIGPLAN Workshop on Language, Compiler, and Tool Support for Real-Time Systems*, June 1994.

[2] J.V. Busquets-Mataix, J.J. Serrano-Martin, R. Ors, P.Gil, A.Wellings, "Adding instruction cache effect to schedulability analysis of preemptive real-time systems", *RTAS*, June 1996.

[3] Chang-Gun Lee, Joosun Hahn, Yang-Min Seo, Sang Lyul Min, Rhan Ha, Seongsoo Hong, Chang Yun Park, Minsuk Lee, Chong Sang Kim, "Analysis of cache-related preemption delay in fixed-priority preemptive scheduling," *IEEE Transactions on Computers*, vol. 47, no. 6, June 1998.

[4] P.Raghavan, C.D.Thompson, " Randomized rounding: a technique for provably good algorithms and algorithmic proofs," *Combinatorica* , vol. 7, no. 4, 1987, pp. 365-374.

[5] N.Karmakar, " A new polynomial time algorithm for linear programming," *Combinatorica* , vol. 4, 1984, pp. 373-396.

[6] A.Srinivasan, "Improved approximation guarantees for packing and covering integer programs," *SIAM Journal of Computing*, vol. 29, no. 2, 1999, pp. 648-670.

[7] Jyh-Charn Liu, Hung-Ju Lee, "Deterministic upper-bounds of the worst-case execution times of cached programs," *RTSS*, December 1994, pp. 182-191.

[8] Jai Rawat, "Static analysis of cache performance for real-time programming," Master's Thesis, Iowa State University of Science and Technology, November 1993.

[9] Yau-Tsun Steven Li, Sharad Malik, Andrew Wolfe, "Performance estimation of embedded software with instruction cache modeling," *ICCAD*, November 1995.

[10] C.L. Liu, J.W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, vol. 20, January 1973, pp. 46-61.

[11] M. Joseph, P. Pandya, "Finding response times in a real-time system," *The BCS Computer Journal*, vol. 29, October 1986, pp. 390-395.

[12] K. Tindell, A. Burns and A. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," *The Journal of Real-Time Systems*, vol. 6, March 1994, pp. 133-151.

[13] A. Datta, S.Choudhury, A.Basu, H.Tomiyama, N.Dutt, "Task Layout Generation to Minimize Cache Miss Penalty for Preemptive Real-Time Tasks: An ILP Approach," *Proc. Workshop on Synthesis and Systems Integration of Mixed Technologies*, April 2000.

[14] A. Datta, S.Choudhury, A.Basu, H.Tomiyama, N.Dutt, "Satisfying Real-Time Constraints of Preemptive Real-Time Tasks Through Task Layout Techniques", *Proc. 14th International Conference on VLSI Design* , January 2001.