PREDICTING TIMING BEHAVIOR IN ARCHITECTURAL DESIGN EXPLORATION OF REAL-TIME EMBEDDED SYSTEMS

Rajeshkumar S. Sambandam[*] *Xiaobo (Sharon) Hu*[†]

Abstract

The degree of *flexibility* of a real-time system architecture indicates the capability of the system to tolerate perturbations in timing related specifications. Flexibility is also an important factor in the trade-off studies between cost and performance. In this paper, we identify the need for a *flexibility* metric and show that the existing real-time analysis results cannot be directly used as such a metric. We formulate new metrics and illustrate their effectiveness in comparing the flexibility of different system architectures.

1 Introduction

Embedded systems encompass a variety of hardware and software components which perform specific functions in host systems. Many embedded systems must respond to external events under certain timing constraints. Failure to respond to certain events on time may either seriously degrade system performance or even result in a catastrophe. Such systems are referred to as real-time embedded systems (RTES) and can be found in many applications, such as engine and transmission control of automobiles, navigation and landing control of aircraft, and communication networks.

In the design of a RTES, decisions made at the architectural design phase greatly affect the final implementation and performance of the system [1,4,6]. A system designer has to overcome a number of challenges at this phase. For instance, the details of a system specification are not fully specified. Even if it is fully specified, it is often the case that the specification will be altered from the time of the first formulation of the system requirements to the time of the final implementation of the system. Furthermore, information on software tasks (e.g., execution time and memory requirement) and hardware components (e.g., power consumption and chip area) is quite often based on some kind of estimates. Many of these uncertainty factors have an impact on the timing behavior of the RTES.

We introduce *flexibility* to indicate how well a particular system architecture can tolerate such perturbations with respect to (w.r.t.) satisfying real-time requirements. Given degrees of flexibility, one may compare and rank different implementations. A system with a higher degree of flexibility is more desirable. Flexibility measure can also facilitate trade-off studies between cost and timing behavior.

Study of flexibility is closely related to timing analysis or feasibility analysis of RTES. At the system architectural level, each software task (τ_i) is associated with the following parameters: computation time (c_i), deadline (d_i), period (p_i), activation or initial request time (a_i) [4,12]. The goal of timing analysis is to check if every request of each task meets its deadline. The timing behavior of a RTES depends on the task allocation scheme, scheduling scheme and scheduling algorithm. In our following discussion, we assume static task allocation, preemptive scheme and rate monotonic (RM) algorithm due to their respective advantages [2,3,8,9,11] and wide usage. Given the above, one approach to predict feasibility is to simulate the system execution for a given schedule. However, the simulation approach can be computationally expensive [7]. It has been shown that predicting the timing performance of a general task system is an *NP*-complete problem [10].

Another approach to feasibility analysis is based on the widely used rate-monotonic analysis (RMA) techniques [9,11,12]. A recent result by Yen and Wolf [12] extends the fixed priority scheduling theory to include data dependencies in the worst-case analysis. Their algorithms are based on heuristic techniques. All these techniques are based on pessimistic assumptions and intend to provide a *yes* or *no* answer to the question of if a system meets its timing requirements. No study has been done to evaluate if they can be employed to predict the flexibility of a system.

In this paper, we explore various potential metrics for flexibility. We will show that some intuitive measures may not be effective and will present more effective measures.

2 Experimental Set-up for Comparing Potential Metrics

In order to perform a fair study on potential metrics for measuring flexibility w.r.t. timing requirements, we used both real-world systems and systems composed of randomly generated tasks (RGT) instead of hand-crafted toy systems.

The real-world systems we used is based on the example presented in [5] which is a RTES implementing a subset of an engine control module. Since our concern is only on the timing details, we directly use the workload specifications given in [5] for our study. There are a total of nine tasks, all independent of each other, to be implemented either in hardware or software. In order to construct a reasonable set of data points we assume that there are various architectural designs (up to 2^9) that contain different combinations of the nine tasks. For this purpose, we also consider a processor which can offer a varying MIPS rate for the execution of the software tasks.

An example embedded computer system such as the one described in [5] may not capture all timing characteristics of real time systems. For instance, in that system, the period and deadline of some of the tasks are integral multiples of each other [5]. Clearly, the timing behavior could be much different if the numbers are relatively prime. Hence, the results we obtain for this system could be a biased one.

In order to ensure a fair comparison, it is imperative to perform timing studies on RGT systems. To compose an RGT system, we need to generate the parameters such as period (p_i), deadline (d_i) , activation (a_i) and computation time (c_i) for each task (τ_i). The ranges for period and computation time were chosen so as to obtain a reasonably good number of fairly uniformly distributed values for the various metrics to be introduced in the

[*] Currently with Level One Communications, Inc., Sacramento, CA 95827, USA. Work done at Department of Electrical and Computer Engineering, Western Michigan University, Kalamazoo, MI 49008, USA.

[†] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. Research supported in part by Hewlett Packard Laboratories, Bristol, England and by NSF grant number MIP-9612298.

next section. In our study the ranges were [10,8500] and [2,950] respectively. Another limitation on the upper limit of periods is the fact that having a higher value would lead to unnecessarily long validation time. Clearly, the deadline for any meaningful system should be greater than their corresponding sum of computation time and activation. Also, since we do not consider systems with deadline greater than the period, the range of the deadline is determined for each task individually. That is, for task τ_i, d_i is a random number within $[c_i, p_i]$. By fixing the above ranges, we can form a valid range for activation as $[0, d_i\text{-}c_i]$.

The analytical methods we describe in the next section estimates the feasibility of a real-time system. In order to compare the effectiveness of each method, we developed an SES/Workbench[1] model to perform event-driven simulation of the engine control system and RGT systems. The simulation time for a task set containing *n tasks* is $a_{max} + 2P$, where $P = \mathrm{LCM}(p_1,p_2,...p_n)$ and $a_{max} = \mathrm{MAX}(a_1,a_2,...a_n)$. As already pointed out, the analysis we are interested in are primarily used during the initial system design evaluation stage and hence details on overhead due to context switching, task scheduling and preemption may not be available. We thus neglect these overhead in our model as well as in our analysis. Note that ignoring these overheads should not greatly affect the relative effectiveness of potential flexibility metrics.

3 Study of Potential Flexibility Metrics

If we knew the precise amount of processing power (such as in terms of average clocks per instruction or MIPS) required to feasibly schedule all tasks and the actual processing power of the given processor, we would be able to deduce its flexibility with respect to the timing requirements. We define ρ as the ratio of the required processing power to the processing power of processor *P*. Then, ρ can be considered as a flexibility metric. However, as we have pointed out previously, finding the exact required processing power is a computationally inhibiting job. Hence, we need to investigate other possibilities of estimating the flexibility of a system.

3.1 Upper Bound Based Approach

One of the frequently used methods to determine feasibility is via an upper bound on the processor utilization. We would like to investigate if this is an appropriate metric for measuring flexibility. Liu and Layland [11] proved that for $d_i = p_i$ and under the worst case phasing (i.e., $a_i = 0$ for all i), feasibility is guaranteed if (1) is true. In some practical systems such

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq n\left(2^{\frac{1}{n}} - 1\right) \qquad (1)$$

as the one mentioned in Section 2, where $d_i \neq p_i$, or $a_i \neq 0$ for some τ_i's, (1) is no longer a valid upper bound. A simple modification gives the following worst case feasibility prediction formula.

$$U = \sum_{i=1}^{n} \frac{c_i}{d_i - a_i} \leq n\left(2^{\frac{1}{n}} - 1\right) \qquad (2)$$

The inequality given in (2) suggests that an upper bound on ρ can be obtained as

$$\rho^u = \left[n\left(2^{\frac{1}{n}} - 1\right)\right]^{-1} \sum_{i=1}^{n} \frac{c_i}{d_i - a_i} \qquad (3)$$

Now, ρ^u may be considered as a flexibility metric provided that a smaller value of ρ^u indicates a higher possibility of a system being feasible. Clearly, a system is feasible if $\rho^u \leq 1$. Careful study is needed when $\rho^u > 1$. We examine the timing behavior for systems with $\rho^u > 1$ as follows. Given the example task systems and more than a thousand RGT systems, we compute ρ^u for each system and then simulate to determine its feasibility. Figure 1 depicts the results of such an approach. The percentage feasible systems over a given range indicate the ratio of the actual number of systems feasible to the total number of systems whose ρ^u values fell in that range.

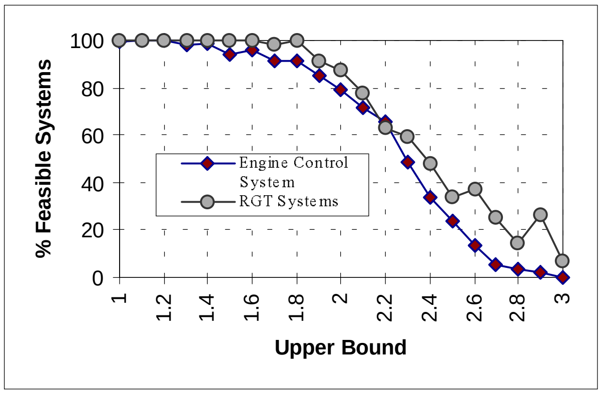

Figure 1. *Graph showing the likely-hood of systems being feasible when ρ^u is used as a predictor.*

From Figure 1, it is not difficult to note that even for an upper bound value of around 2, close to 80% of the systems are actually feasible. This shows the over-estimation of the schedulability of the tasks when we use the pessimistic prediction approach for determining the feasibility. Another interesting yet somewhat counter-intuitive fact is that there are ample number of cases where systems with higher values of ρ^u are more likely to be feasible than systems with lower values of ρ^u. This indicates that ρ^u as defined in (3) is not an appropriate measure for flexibility. That is, by simply computing ρ^u's of two systems, one cannot decide if one system is more flexible (or more likely to be feasible) than the other.

3.2 A Loose Lower Bound Based Approach

The inferences on the results based on the upper bound based techniques suggest that we may investigate its counterpart, the lower bound based approach. It is well known that a task set is definitely infeasible if the average processor utilization is greater than 1. This can be used to impose a lower bound on ρ. That is,

$$\rho^{l_1} = \sum_{i=1}^{n} \frac{c_i}{p_i} \qquad (4)$$

Consequently, if $\rho^l_1 > 1$, the system is infeasible. We would like to see whether the value of ρ^l_1 in the range [0,1] can indicate the flexibility of a system. The same analysis as for ρ^u was carried

[1]Product of Scientific and Engineering Software, Inc., Texas, Austin.

out and the results are summarized in Figure 2 (loose bound). An immediate observation is that this graph shows better monotonicity than that of ρ^u given in (3) though a peak is noticed for the RGT systems. Hence one may consider ρ^l_1 defined in (4) as a potential candidate for predicting flexibility.

A significant drawback of using ρ^l_1 as a metric is that the percentage feasible systems drops quickly as the value of ρ^l_1 increases. This means that when we have a system with relatively low ρ^l_1 (say 0.5), it is not clear whether it is worthwhile to investigate such system configuration further. In addition, using ρ^l_1 as the only timing analysis parameter may fail to detect certain obviously infeasible systems. For example consider the three task system shown in Table 1. The value of ρ^l_1 for this system is 46.7% which suggests that this system could be feasible. However, it is easy to find from the timing parameters that the third task would never meet its deadline since the minimum time it has to wait for higher priority tasks is 7 and hence the system is infeasible.

a_i	p_i	c_i	d_i
0	10	4	6
0	30	3	10
0	120	8	14

Table 1. A sample system.

3.3 A Tight Lower Bound Based Approach

We have seen that in (4) ρ^l_1 is close to being an appropriate candidate. In [4], a tighter lower bound on ρ was given, but no sufficient data were provided to illustrate the effectiveness of this bound. We will give a slightly modified lower bound calculation and study its behavior with respect to timing prediction. Consider a task set of n tasks. For the first execution request of τ_i, any task requests that have deadlines preceding d_i must be completed before d_i. Thus the total computation requirement between [0, d_i] includes at least satisfying all these task requests. We state the following lemmas that can be used to calculate the number of task requests in intervals $[a_j,d_i]$ for $d_j \le d_i$ and $[a_i,d_i]$.

Lemma 1 *For i tasks that are arranged in the ascending order of their deadlines, define*

$$k_j \equiv \begin{cases} \left\lceil (d_i - a_j)/p_j \right\rceil & \text{if} \left\lfloor (d_i - a_j)/p_j \right\rfloor . p_j + d_j \le d_i \\ \left\lfloor (d_i - a_j)/p_j \right\rfloor & \text{otherwise} \end{cases}$$

then the i tasks cannot be feasibly scheduled if

$$\sum_{j=1}^{i} \frac{k_j . c_j}{d_i - a_{\min}} > 1 \quad \textit{where} \quad a_{\min} = \min_{1 \le j \le i} a_j$$

Lemma 2 *For i tasks that are arranged in the ascending order of their deadlines, let k_j be the same as that defined in Lemma 1. Define*

$$h_j = \begin{cases} k_j - \left\lceil \frac{a_i - a_j}{p_j} \right\rceil & \text{if} \quad a_j < a_i \\ k_j & \text{otherwise} \end{cases} .$$

Then, the i tasks cannot be feasibly scheduled if $\sum_{j=1}^{i} \frac{h_j . c_j}{d_i - a_i} > 1$

The above lemmas can be readily proved by noting that k_j and h_j are the minimum number of times τ_j needs to execute between $[a_j,d_i]$ and $[a_i,d_i]$ respectively. Based on these lemmas, we can now define a tighter lower bound as given below.

$$\rho^l_2 = \max_{i=1}^{n} \left\{ \sum_{j=1}^{i} \frac{k_j . c_j}{d_i - a_j}, \sum_{j=1}^{i} \frac{h_j . c_j}{d_i - a_i} \right\} \qquad (5)$$

To investigate the effect of ρ^l_2, as given in (5), similar analysis as those in sub-section 3.1 was performed. Figure 2 (tight bound) depicts the results. Clearly, the graph is monotonic

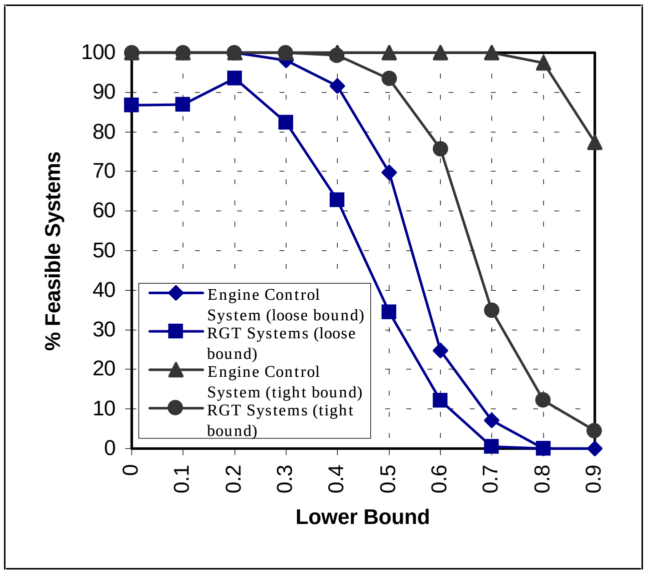

Figure 2. *Graph showing the likely-hood of systems being feasible when ρ^l_1 and ρ^l_2 are used as a predictor.*

and hence (5) can be used as a flexibility metric. It is interesting to note that the percentage feasible systems is quite high even when ρ^l_2 is relatively large (e.g. 0.5). Thus, ρ^l_2 is a quite reliable predictor for feasibility.

Though using ρ^l_2 can be an effective means to study potential design candidates in the architectural design exploration of real time systems, this analytical technique may become computationally intensive when the number of tasks grow larger. It would be desirable to have a metric that is less computationally involved.

3.4 A Feasibility-Factor Based Approach

In [4], the authors introduced a feasibility measure called feasibility factor based on the upper and lower bounds of throughput requirements. We would like to generalize the definition of feasibility factor and study its behavior. We define the feasibility factor as

$$\lambda \equiv \frac{1 - \rho^l}{\rho^u - \rho^l} \quad \textit{if}\ \rho^u \ne \rho^l \qquad (6)$$

Notice if $\rho^u = \rho^l$, we have a precise prediction of ρ. A set of tasks allocated on a processor are feasible if $\lambda \ge 1$ and they are not feasible if $\lambda < 0$. For $0 \leq \lambda < 1$, the feasibility of the system cannot be predicted solely based on λ and needs to be carefully analyzed.

To examine the behavior of λ, we performed a similar analysis that was done for previous metrics. Since λ can be computed based on any given formula for ρ^l and ρ^u, we obtained different λ values by using ρ^l_1 and ρ^l_2. They are summarized in Figure 3. Notice that this metric is monotonic for both ρ^l_1 and ρ^l_2 based calculations. Using our previous reasoning, λ is a reliable predictor for feasibility. Furthermore, it can be considered as an estimate of critical excess requirement ratio. Instead of using $1-\rho^l$ directly, the value of $1-\rho^l$ is scaled by $(\rho^u - \rho^l)$. Such a scaling gives better estimation since it includes the effect of ρ^u. Therefore, we can use λ directly to measure the flexibility of a system architecture. A larger value of λ indicates that the system is relatively more feasible. Note that when ρ^l_2 is used, the likely-hood of the system being feasible is higher, compared with the case based on ρ^l_2. Of course, the computational needs for the one based on ρ^l_2 is larger. Nevertheless, it is up to the discretion of the system designer to choose either of these methods, since both of them exhibit suitable characteristics.

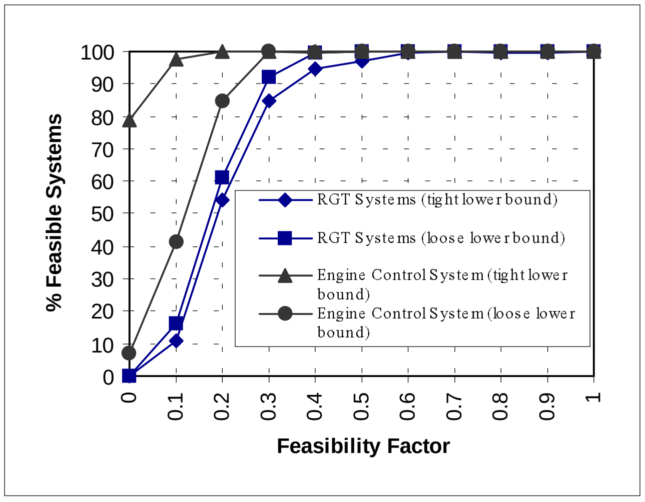

Figure 3. *Graph showing the likely-hood of systems being feasible when λ is used as a predictor.*

4 Summary and Discussion

In this paper, we motivated the need for a flexibility metric for the efficient analysis of potential design candidates in the architectural design exploration of real time embedded systems. We identified the shortcomings of certain metrics for measuring the flexibility of a real time system. We proposed two modified approaches to overcome these shortcomings. Our analysis can be used both as a constraint as well as an attribute in hardware/software partitioning of real time systems [4]. In the constraint case, we eliminate the infeasible solutions. In the attribute case, we can efficiently evaluate systems modeled and study trade-offs between flexibility and other attributes (e.g., cost and power consumption).

We have seen that ρ^l_2 and λ are quite reliable predictors for feasibility. Let us define critical excess requirement ratio as $\rho^c=1-\rho^l_2$. The value of ρ^c provides an estimate of the additional load the processor could handle after meeting the current task specifications. In the case of λ, the value of $1-\rho^l$ is scaled by $\rho^u-\rho^l$. Such a scaling gives better estimation since it includes the effect of ρ^u. The critical excess requirement ratio is useful in comparing systems which are guaranteed to be feasible since it reflects the amount of processing power yet available after meeting the current requirements. We are currently studying the capability of these potential metrics in reflecting the changes in the event of a processor overload.

The limitation of this work is that we have considered only uni-processor systems and assumed that the software tasks running on the processor do not have any dependency. We intend to expand our work to include these cases. In addition, it is worthwhile to carry out more statistical analysis to obtain confidence-interval related figures.

References

[1] K. Buchenrieder and C. Veith, "CODES: a practical concurrent design environment", *Handouts from International Workshop on Hardware-Software Codesign*, October 1992.

[2] S. K. Dhall, and C. L. Liu, "On a real-time scheduling problem", *Operations Research*, Vol. 26, No. 1, January, 1978, pp. 127-140.

[3] M. R. Garey, and D. S. Johnson, "Complexity results for multiprocessor scheduling under resource constraints", *Society for Industrial and Applied Mathematics Journal of Computing*, 4, 1975.

[4] X. Hu and J.G. D'Ambrosio, "Configuration-level hardware/software partitioning for real-time embedded systems", to appear in *Journal of Design Automation for Embedded Systems*.

[5] X. Hu, J.G. D'Ambrosio, B.T. Murray, and D. Tang, "Codesign of Architectures for Automotive Powertrain Modules", *IEEE Micro*, August 1994, pp. 17-25.

[6] S. Kumar, J.H. Aylor, B.W. Johnson and W.A. Wulf, "Object-oriented techniques in hardware design", *Computer*, vol. 27, no. 6, 1994, pp. 64-70.

[7] E. L. Lawler, and C. U. Martel, "Scheduling periodically occurring tasks on multiple processors", *Information Processing Letters*, Vol. 12, No. 1, 1981, pp. 9-12.

[8] J. P. Lehoczky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems", *Proceedings of Real-Time Systems Symposium*, December 1992, pp. 110-123.

[9] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior", *Proceedings of the 1989 IEEE Real-time System Symposium*, December 1989, pp. 166-171.

[10] J. Y-T Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks", *Performance Evaluation*, vol.2, 1982, pp.237-250.

[11] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", *Journal of the Association for Computing Machinery*, vol. 20, no. 1, 1973, pp. 46-61.

[12] T.-Y. Yen and W. Wolf, "Performance estimation for real-time distributed embedded systems", *Proceedings of the International Conference on Computer Design (ICCD'95)*, October 1995, pp. 64-69.