INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Scheduling For The Real Time Antilock Braking System

V.Mallika
Pg Scholar, Saveetha Engineering College

Kalai Vani
C.T.Me,(Ph.D)
Associate Professor, Saveetha Engineering College

Abstract:

In this project we have proposed an algorithm named Earliest Deadline First-Virtual Deadline. Many safety-critical embedded systems are subject to certification requirements; some systems may be required to meet multiple sets of certification requirements, from different certification authorities. Certification requirements in such "mixed-criticality" systems give rise to interesting scheduling problems, which cannot be satisfactorily addressed using techniques from conventional scheduling theory. In this paper, we study a formal model for representing such mixed-criticality workloads. We demonstrate first the intractability of determining whether a system specified in this model can be scheduled to meet all its certification requirements, even for systems subject to merely two sets of certification requirements. Then we quantify, via the metric of processor with improved speedup factor. This method produces a better speed up factor value compare to the existing one. It introduces the procedure for implementing logarithmic computational time. The proposed system derives utilization bound and simulates the behaviour of utilization bound. And also own criticality based priority is used for the scheduling of the job which contain same critical level.

1.Introduction

Nowadays there is a increasing trend in the embedded system towards implementing the multiple functionalities in a common platform..In such system, mixed criticality problem will occur. Mixed criticality system is nothing but multiple events are implemented in the common platform. While implementing such a mixed criticality system sum of the parameters must known. They are arrival time, worst case execution time, deadline, period,low criticality, high criticality. In such systems, mixed criticalities can mean two different things. The first is the obvious one upon platforms that over support for multiple functionalities, it is some of these functionalities are probably more important to the overall welfare of the platform than others. For instance, it is more important to the correct behaviour of an automotive control system that the antilock brake system (ABS) works correctly than that the on-board radio does so. However, there is another aspect to mixed criticalities that arises in application domains (such as civilian and defence Avionics) that are subject to mandatory certification requirements by statutory organizations. Usually critical system can be classified into two types. They are mission critical and flight critical.

Mission critical system is validated by design team. Worst Case Execution Time (WCET) determined by extensive experimentation the mission-critical functionalities, concerning reconnaissance and surveillance objectives, like capturing images from the ground, transmitting these images to the base station.

Flight critical system is certified by Certification Authorities. Worst Case Execution Time (WCET) determined by cycle-counting under pessimistic assumptions. the flight-critical functionalities: to be performed by the aircraft to ensure its safe operation. This is a manual calculation. Different from the mission critical. Usually the real time system classified into two types Hard real time system and Soft real time system. Soft real time system not need to meet their deadline. Hard real time system must meet their deadline. If it fails means there will be the big problem in the surroundings. So the scheduling is most important for the hard real time system.

A multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time-multiplexing. Part of the reason for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU. Another main reason is the need for processes to perform I/O operations in the normal course of computation. Since I/O operations ordinarily require orders of magnitude more time to complete than do CPU instructions, multiprogramming systems allocate the CPU to another process whenever a process invokes an I/O operation

For the deadline based scheduling usually Earliest Deadline First is used. The latest deadline job will executed first.if any jobs missed their deadline means error will occur. For that we are going for an algorithm called Earliest Deadline First With Virtual Deadline (EDF-VD).In this a new modified parameter is calculated called virtual deadline. This modified parameter is added with the arrival time.

2. Related Work

Usually sporadic system which consist infinite number of jobs, is mainly characterised by the following parameter sporadic task $\tau_{i,}$ deadline $D_{i,}$ arrival time A_i and a period Ti; such a task generates an unbounded sequence of jobs with successive jobs arriving at least Ti time units apart, and each job needing up to Ci units of execution by a deadline that occurs Di time units after the job's arrival.

The pre-emptive uniprocessor scheduling of collections of mixed-criticality independent jobs was studied in [5], [4], [7], [6]. An efficient scheduling algorithm and associated polynomial-time schedulability test was proposed that makes the following guarantee: any dual-criticality system that can be scheduled by an optimal clairvoyant algorithm on a given processor can be scheduled by this algorithm on a processor that is

$(1+\sqrt{5})/2=1.62$ times as fast. In [15], [12], this result was extended to mixed-criticality sporadic task systems: a scheduling algorithm and associated pseudo polynomial time schedulability test was proposed that makes the same guarantee.

These scheduling algorithms, however, have too large a run-time complexity to be implementable in practice: the run-time complexity per scheduling decision of the algorithm in [15] is pseudo-polynomial in the representation of the task system, while the one in [12] is quadratic in the number of tasks

An important special case of sporadic task systems are task systems in which each task i satisfies the property that Di = Ti such systems are called implicit-deadline.

2.1.Definition

A mixed-criticality (MC) implicit-deadline sporadic task system τ consists of a finite specified collection of MC implicit-deadline sporadic tasks, each of which may generate an unbounded number of MC jobs. Dual-criticality systems: systems with two distinct

criticality levels, which we denote as LO and HI. Parameters: Ji = (ai; di; xi; ci(LO); ci(HI)), where
ai=release time di=deadline. We require that di> ai. Xi=Criticality of the job. ci(LO)= specifies the worst case execution time (wcet) estimate of Ji that is used by the system designer (i.e., the wcet estimate at the lo criticality level). ci(HI) specifies the worst case execution time (WCET) estimate of Ji that is used by the system designer (i.e., the WCET estimate at the LO criticality level).

3.Methodology Used

A more refined analysis of Earliest Deadline First-Virtual Deadline (EDF-VD) showing that EDF-VD can actually make a better performance guarantee any task system that can be scheduled by an optimal clairvoyant algorithm on a given processor can be scheduled by EDF-VD on a processor that is 4:3 times as fast. This new analysis is based upon some sophisticated new techniques complexity claimed .We also perform further analysis on the behaviour of EDF-VD,deriving a utilization based schedulability test and exploring its behaviour under certain extreme conditions. and deep insights that we have recently developed, and represents a substantial improvement over the bound proved.

3.1.Own Criticality Based Priority(OCBP)

Construct fixed priority table offline. At each scheduling decision point, dispatch the job with the highest priority**.** a sporadic task system will potentially release an infinite number of jobs, at any time we only need to consider the jobs that can be released in the current busy interval. This is because before the system goes into the next busy interval, there must be a time point at which the processor becomes idle and the system is reset to the same state as in the beginning of the previous busy interval.
The first step of OCBPS's off-line computation is to compute a priority order for all the jobs in I. Since all the jobs from the same task are identical, we can always assign priorities to jobs from the same task in the way that later jobs never have higher priorities.

3.2.OCBP Table Creation

Assume any one of the job is highest priority.Find out the average working time. Similarly consider another job is a highest priority and find out the average working time. In this the minimum average working time value has been taken first.

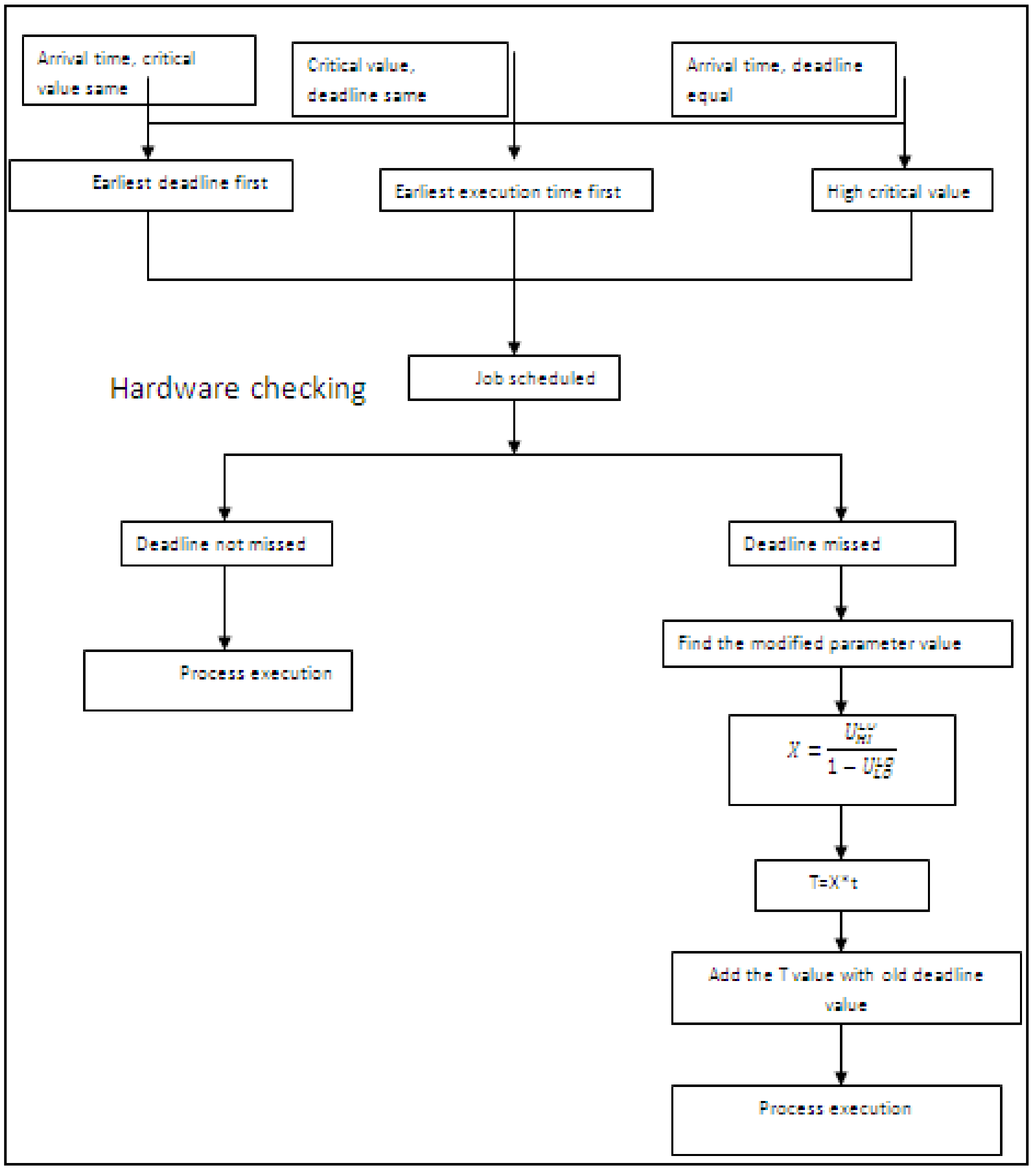
Arrival time, critical value same
Critical value, deadline same
Arrival time, deadline equal
Earliest deadline first
Earliest execution time first
High critical value
Hardware checking
Job scheduled
Deadline not missed
Deadline missed
Process execution
Find the modified parameter value
$X = \frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}}$
T=X*t
Add the T value with old deadline value
Process execution

Figure 1: Flow Chart

Get the no of jobs, arrival time, critical value (low & high), dead line, Execution Time

Assume any one of the job is highest priority.Find out the average working time. Similarly consider another job is a highest priority and find out the average working time. In this the minimum average working time value has been taken

Job	Release Time	Critical Level	Execution Time	Deadtime
1	0	2	2	4
2	0	2	2	5
3	0	2	4	10

Tabler 1: List of Jobs

Assume J1 is low priority=p2+p3=4

J1 not meets it deadline

Assume J2 is low priority=P1+P3=4

J5 not meet deadline

Assume j3 is low priority=p1+p2=6

All jobs meet deadline

3.3.Earliest Deadline First-Virtual Dead Line(EDF-VD)

Each task in an EDF scheduler is assigned a deadline(e.g. a moment in the future at which the task _must_ be completed). Every time a task is inserted in the system or completed, the scheduler looks for the task which has the closest deadline and selects it for execution. In order to ensure that the scheduler is still able to meet each deadline, a 'monitor must evaluate if each new task doesn't overload the system and deny execution if it will do so.

In order to implement EDF-VDbased system, one will have to know both the _deadline_ of the task (which could optionally be computed as "no more than X ms in the future") and the expected time needed to perform the task (required by the monitor). QoS network routers usually implement variants of EDF-VD scheduling.

Again, there is a utilisation based test for EDF-VD systems. The limit is simpler however - it is always 100%, no matter how many processes are in the set. This makes dynamic analysis of schedulability easier. Not only that, but the EDF-VD utilisation test is both sufficient and necessary, so can be relied on to provide an accurate indication of schedulability.

3.4.Algorithm For EDF-VD

Pre-processing

1. Compute x as: X=$U^{\frac{LO}{HI}} / \frac{LO}{LO}$

U=Upper bound

2. If (x$U\frac{LO}{LO}$ +$U\frac{HI}{HI}$≤1) then declare success and compute

Else declare failure and exit

3.5.Utillisation Based Analysis

It is used to check whether the system is schedulable or not.The formula used for this is

$$U = \sum_{I=1}^{N} \frac{C}{T} \leq N(2^{\frac{1}{N}} - 1)$$

C=Critical level

T=Dead line

N=No of job

Real-time system is schedulable

$$\sum U \leq n\ (2^{1/n}-1)$$

Theorem: a system of n independent pre-emptable periodic tasks with Di = pi can be feasibly scheduled on one processor using RM if

$$\sum U \leq n\ (2^{1/n}-1)$$

Let us consider the following table

Process	Period	Computation time	Priority	utilization
1	4	1	1	0.25
2	5	1	2	0.20
3	10	1	3	0.10

Table 2: List of Jobs

4.Experimental Result

The given job is scheduled first. After that they have to check whether the given system is schedulable or not. Finishing the scheduling process if deadline is missed means find out the modified parameter value and add with the arrival time.

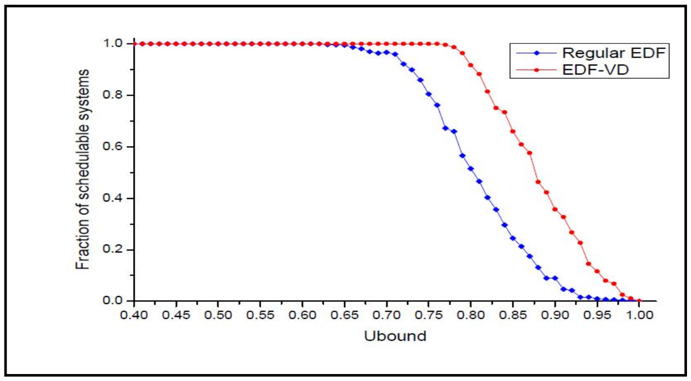

Figure 2

5.Conclusion

Devising more cost-efficient techniques for obtaining certification for safety-critical embedded systems has been identified as a prime research challenge. We believe that in mixed-criticality systems, these certification considerations give rise to fundamental new resource allocation and scheduling challenges that are not adequately addressed by conventional real-time scheduling theory. In this paper, we consider the scheduling, upon pre-emptive uni-processors, of mixed-criticality systems that can be modelled using a mixed criticality generalization of the implicit-deadline sporadic tasks model. An algorithm called EDF-VD was proposed in for scheduling such systems. We have proved that EDF-VD is speedup-optimal by showing that (i) it has a processor speedup factor equal to 4/3 and (ii) no non-clairvoyant algorithm can have a smaller speedup factor. The result in Theorem 4 improves on an earlier result presented in which had shown a speedup factor of ($\sqrt{5}$+1)=1:62. This improved speedup factor was obtained by first deriving a superior sufficient schedulability condition

6.Referance

1. J. Barhorst, T. Belote, P. Binns, J. Hoffman, J. Paunicka, P. Sarathy, J. S. P. Stanfill, D. Stuart, and R. Urzi. White paper: A research agenda for mixed-criticality systems, April 2009. Available at http://www.cse.wustl.edu/˜ cdgill/ CPSWEEK09 MCAR.
2. S. Baruah and A. Burns. Sustainable scheduling analysis. In Proceedings of the IEEE Real-time Systems Symposium, pages 159–168, Rio de Janeiro, December 2006. IEEE Computer Society Press.
3. S. Baruah, A. Burns, and R. Davis. Response-time analysis for mixed criticality systems. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Vienna, Austria, 2011. IEEE Computer Society Press.
4. S. Baruah, H. Li, and L. Stougie. Mixed-criticality scheduling: improved resource-augmentation results. In Proceedings of the ICSA International Conference on Computers and their Applications (CATA). IEEE, April 2010.
5. S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable mixed-criticality systems. In Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS). IEEE, April 2010.
6. S. K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti- Spaccamela, N. Megow, and L. Stougie. Scheduling real-time mixed-criticality jobs. IEEE Transactions on Computers. To appear.
7. S. K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti- Spaccamela, N. Megow, and L. Stougie. Scheduling realtime mixed-criticality jobs. In P. Hlinen´y and A. Kucera, editors, Proceedings of the 35th International Symposium on the Mathematical Foundations of Computer Science, volume 6281 of Lecture Notes in Computer Science, pages 90–101. Springer, 2010.
8. S. K. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti- Spaccamela, S. van der Ster, and L. Stougie. Mixed-criticality scheduling of sporadic task systems. In Proceedings of the 19th Annual European Symposium on Algorithms, pages 555–566, Saarbrucken, Germany, September 2011. Springer-Verlag.
9. A. Burns and S. Baruah. Timing faults and mixed criticality systems. In Jones and Lloyd, editors, Dependable and Historic Computing, volume LNCS 6875, pages 147–166. Springer, 2011.

10. A. Burns and B. Littlewood. Reasoning about the reliability of multi-version, diverse real-time systems. In Proceedings of 31st IEEE Real-Time Systems Symposium, pages 73–81.IEEE Computer Society Press, December 2010.
11. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. MIT Press, third edition, 2009.
12. N. Guan, P. Ekberg, M. Stigge, and W. Yi. Effective and efficient scheduling for certifiable mixed criticality sporadic task systems. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Vienna, Austria, 2011. IEEE Computer Society Press.
13. N. Guan and W. Yi. Improving the scheduling of certifiable mixed criticality sporadic task systems. Technical report, Uppsala University, 2012.
14. J. Y.-T. Leung and J. Whitehead. On the complexity of fixedpriority scheduling of periodic, real-time tasks. Performance Evaluation, 2:237–250, 1982.
15. H. Li and S. Baruah. An algorithm for scheduling certifiable mixed-criticality sporadic task systems. In Proceedings of the Real-Time Systems Symposium, pages 183–192, San Diego, CA, 2010. IEEE Computer Society Press.
16. C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 20(1):46–61, 1973.
17. A. Mok. Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983. Available as Technical Report No. MIT/LCS/TR-297.
18. A. Mok. Task management techniques for enforcing ED scheduling on a periodic task set. In Proc. 5th IEEE Workshop on Real-Time Software and Operating Systems, pages 42–46, Washington D.C., May 1988