AN EFFECTUAL NON-PREEMPTIVE SCHEDULING METHODOLOGY THROUGH APERIODIC AND SPORADIC TASKS IN REAL-TIME EMBEDDED PROCESSING SYSTEMS

Shaik Saidulu Research Scholar Dept of ECE, SunRise University-Alwar
Dr. Yash Pal Singh Prof, Dept of ECE, SunRise University

Abstract

The Scheduling algorithm is one of the most important portions of embedded operating systems especially for real-time embedded operating systems. The performance of scheduling algorithm will influence the performance of the whole system. Real time embedded operating system needs better response time for real-time process; it is more rigid on response time for hard real-time embedded operating system. Scheduling is the process which assigns the resources to the Different task depending on its deadline. Scheduling algorithms will be individual for each of the task priority (or) deadline. Different approaches for scheduling are like Rate monotonic, Deadline monotonic, and earliest deadline first, least slack time, Round-robin, weighted Round-robin. The problem of inconsistencies occurs in these algorithms such as optimality, transient overload condition, resource sharing problem and CPU utilization ratio and task collision. Earliest Deadline First (EDF) is an optimal scheduling algorithm for uniprocessor real-time systems. When non-preemptive EDF can share a common stack, leading to vastly reduced memory requirements and needed to solve resource sharing and task collision. According to the EDF approach, it is one of the method to improve the processor utilization ratio and by evaluating number of computations on task period. The performance of the EDF is calculated in terms of Success Ratio and effective CPU Utilization.

Time constraint is the main factor in real-time operating system. Multitasking embedded systems with precise timing may use a real-time operating system (RTOS) to schedule tasks at runtime using priority-based cooperative or preemptive scheduling techniques. Task scheduling will be used to allocate some time frames and used to schedule the task based on the priorities of the task and the execution will arrives. Our goal here is to verify for which scheduling strategy is best suited for real-time systems. Although these systems are multiprocessor based heterogeneous, the task scheduling will

Be very difficult and a challenge.

Keywords: *- Real-Time System, Task Scheduling, Earliest Deadline First, Rate Monotonic, Deadline Monotonic, Round Robin.*

1. Introduction: - Today, embedded electronics have stronghold in every space of the society. To full fill the demand of people lots of research and development are going on in this domain. In Particular, frequently changing needs of people gives rise to Reconfigurable real time system. Those systems are typically embedded Devices implemented on Microcontroller based platforms which support Reconfiguration in the resources even during run-time. This resource reconfiguration is managed by the Real time operating system (RTOS). RTOS is operating systems designed for real time embedded systems. RTOS is a collection of different real time algorithms for managing resources, schedule the tasks and provide lots of other functionality for real time applications. In this paper we propose a novel Multi shape task scheduling algorithm for scheduling real-time task and compare with the conventional algorithms [1]. Real time systems are those systems which supports real-time constraints. Real time systems can be categorized into three categories i.e. hard real time systems, soft real time systems, and firm real time systems. Hard real time systems are those systems in which time delay in response can results potential failure to the system or the loss of human life. In general there is a cost function associated with the system. Many of these systems are considered to be safety critical. Soft real time system is those system where deadline overruns are tolerable, but not desired. There are no hazardous consequences of missing one or more deadlines. Firm real time systems are mix of hard real time and soft real time system. Here infrequent deadline missing is tolerable but it degrades the performance of the systems. To avoid the deadline missing, we need to schedule the task effectively [2]. To solve a scheduling problem, we try to find a schedule for the tasks to execute in such a way so that tasks can be completed before the deadline. Scheduling can be used in any domain where is the limited number of resources to be scheduled efficiently. This efficiency means optimizing desired criteria. Such criteria could be to minimize the schedule length, to maximize the resources utilization ratio to maximize the number of accepted tasks. In this paper, we analyzed algorithms for the periodic tasks. A periodic task is an infinite

sequence of jobs with periodic ready times, where the deadline of a job could be equal to, greater than or less than ready time of the succeeding jobs [3]. Scheduling algorithms can be classified in, Uni-processor vs Multiprocessor Scheduling, Soft real time vs hard real time Scheduling, Static vs Dynamic Scheduling, Fixed vs Dynamic Priority Scheduling, Pre-emptive vs Non pre-emptive Scheduling. So for real time systems different Task scheduling algorithms have been studied to avoid the deadline missing. Some of the popular real time task Scheduling algorithms are Earliest Deadline first (EDF), Fixed priority Task scheduling, rate monotonic (RM), Deadline monotonic (DM) etc [4] [5]. Rate monotonic Scheduling algorithm is mostly researched, reviewed, and analyzed algorithm. It is a priority driven algorithm in which priorities are assigned according to the cycle period of the job i.e. the task which has less cycle duration, get the higher priority. So for periodic Tasks, priority of the all instances of the task is known before the arrival and it will be same for all instances. It supports pre-emption and work well on uni-processor system [6]. In DM is a scheduling algorithm, priorities are assigned according to relative deadline Di i.e. higher priority is assigned to the task which has less deadline. It will be fixed for all the future instances. In general, DM is optimal for systems that consist of pre-emptive synchronous independent tasks whose relative deadline is less to their period (Di $\leq$ Period).DM could be used with periodic, aperiodic and sporadic tasks systems [7]. EDF algorithm is a dynamic priority algorithm, the priorities are assigned based on the value of relative deadline of the tasks in real time. The task with nearest deadline is given highest priority and it is selected for execution. EDF could be applied to various tasks models (preemptive, non preemptive, periodic, non-periodic, etc.).EDF has also been shown to be optimal in the case of non-periodic tasks. EDF scheduling outperforms RM and produces less pre-emption compared to RM and it is very fast.

Criteria for a good scheduling algorithm:

- Fairness: all processes get fair share of the CPU
- Efficiency: keep CPU busy 100% of time
- Response time: minimize response time
- Turnaround: minimize the time batch users must wait for output

throughput: maximize number of jobs per hour

1.1 Non-preemptive scheduling

On the other hand, under non-preemptive scheduling, the scheduler does not allow the currently executing task to be preempted. Any higher priority tasks released during the execution of a task has to wait for it to complete its execution. This means that the scheduler need not implement or execute mechanisms to suspend the currently running task, save its context or reload the context when it resumes execution, implying negligible runtime overheads when compared to a preemptive scheduler [54] showed that no optimal online scheduling algorithm that uses inserted idle times exists for non-preemptively scheduling sporadic real-time tasks. Jeffay et al. [4] showed that, under a non-idling scheme, non-preemptive EDF is optimal on a uniprocessor for sporadic task systems. Putting these two results together, when it comes to non-preemptively scheduling sporadic real-time tasks on uniprocessors, non-preemptive EDF is optimal, i.e., if there exists an algorithm that can non-preemptively schedule a sporadic real-time task set, non-preemptive EDF can schedule it. However, note that non-preemptive EDF is not an optimal uniprocessor scheduling algorithm, unlike preemptive EDF. To our knowledge there exists no optimal non-preemptive scheduling algorithm on multiprocessors. In 1980, Kim and Naghibdadeh [73], and August, 2014 in 1991, Jeffay et al. [74], gave exact schedulability tests for implicit-deadline task sets under non-pre-emptive EDF scheduling. These tests were extended by George et al. [44] in 1996, to the general case of sporadic task sets with arbitrary deadlines. While pre-emptive EDF is an optimal single processor scheduling algorithm, in the non-pre-emptive case no work conserving algorithm is optimal. (A work-conserving scheduling algorithm is one that does not idle the processor when there are tasks ready to be executed). This is because in general, with non-pre-emptive scheduling, it is necessary to insert idle time to achieve a feasible schedule. The interested reader is referred to [44] for examples of this behavior. In 1995, Howell [46] showed that for non concrete strictly periodic task sets, where the times at which each task may be first released are unknown, and the problem of determining a feasible non-pre-emptive schedule is NP hard. Further, they showed that for sporadic task sets, no optimal online inserted idle time algorithm can exist. In other words, clairvoyance is needed to

determine a feasible non-preemptive schedule. While no work-conserving algorithm is optimal in the strong sense that it can schedule any task set for which a feasible non pre-emptive schedule exists; in 1995, George et al. [45] showed that non-pre-emptive EDF is optimal in the weak sense that it can schedule any task set for which a feasible work-conserving, non-pre-emptive schedule exists. For fixed priority non-pre-emptive scheduling of arbitrary deadline task sets, George et al. [44] derived an exact schedulability test based on the approach of Tindell et al. [67] for the pre-emptive case. George et al. showed that unlike in the pre-emptive case, deadline monotonic priority ordering is not optimal for constrained-deadline task sets scheduled non preemptively. Further, they showed that Audsley‟ s optimal priority assignment algorithm [7] is applicable, and can be used to determine an optimal priority ordering in this case. Subsequent research by Bril et al. [17] has refined exact analysis of non-pre-emptive fixed priority scheduling, correcting issues of both pessimism and optimism, and extending the schedule ability tests to co-operative scheduling where each task is made up of a number of non-pre-emptive regions. There are various approaches that allow limited preemption and thus represent a compromise between fully preemptive and fully non-pre-emptive scheduling.

When a process enters the state of running, the state of that process is not deleted from the scheduler until it finishes its service time. On-preemptive scheduling is more efficient than preemptive scheduling since preemption incurs context switching overhead which can be significant in fine grained multithreading systems. The concept of a task that is invoked periodically is central to a real-time system. Tasks are executed on a processor and must complete execution in a timely manner. Based on the task characteristics, priorities are assigned to tasks, which drive the scheduling decisions. Task scheduling can be classified into two broad categories: preemptive scheduling and non-preemptive scheduling. Under preemptive scheduling, the current task execution can be preempted by a higher priority task, whereas under non-preemptive scheduling, a higher priority task can be scheduled only after the completion of the current task. Though preemptive scheduling can guarantee higher system utilization, there are scenarios where properties of hardware devices and software configuration make preemption either impossible or

prohibitively expensive. Non-preemptive scheduling also has the advantages of accurate response time analysis, ease of implementation, no synchronization overhead and reduced stack memory requirements. Non-preemptive scheduling is used in light weight multi-tasking kernels and has been shown to be beneficial in multimedia applications [35].

This work focuses on energy efficient scheduling of non-preemptive real-time tasks. The two major techniques of minimizing the processor energy consumption are: shutdown and slowdown. Slowdown through dynamic voltage and frequency scaling (referred to as DVS) is known to be effective in energy minimization [36-38]. A reduction in the supply voltage decreases the power consumption of the processor because of the quadratic relationship between power and voltage. However, the transistor gate delay (and hence frequency) depends on the voltage and a decrease in voltage has to be accompanied by a decrease in processor frequency. There is a linear dependence between frequency and voltage [39], resulting in a linear increase in the execution time of a task. Thus voltage scaling provides the ability to perform an energy delay tradeoff in the system. Real-time systems have strict timing requirements and slowdown has to be performed judiciously in achieving our goal of minimizing energy. Previous works on energy aware scheduling have mainly focused on preemptive scheduling. Among the earliest works, Yao et al. [40] presented an optimal off-line algorithm to schedule a given set of jobs with arrival times and deadlines. For a similar task model, optimal algorithms have been proposed for fixed priority scheduling [41, 42] and scheduling over a fixed number of voltage levels [38], [43]. Energy efficient scheduling of periodic real-time task sets has also been addressed. Real-time feasibility analysis has been used in previous works to compute static slowdown factors for tasks [44], [45]. Aydin et al. [37] have addressed the problem of energy minimization considering the task power characteristics. When tasks complete earlier than the worst case, there is opportunity for additional (dynamic) slowdown which increases the energy savings [46-48]. The problem of maximizing the system value for a specified energy budget, as opposed to minimizing the total energy, is addressed in [49, 50]. Note that these works assume a preemptive task system. Non-preemptive scheduling has been addressed primarily in the context of

multiprocessor scheduling [51]. Zhang et al. [52] have given a framework for non-preemptive task scheduling and voltage assignment for dependent tasks on a multi-processor system. They have formulated the voltage scheduling problem as an integer programming problem. The problem of minimizing the energy consumption by performing a slowdown tradeoff in the computation and communication subsystems is addressed in [46].

Non-preemptive scheduling may also be enforced by the hardware. For example, messages in control area network (CAN) buses are not preemptable [1]. When considering non preemptive scheduling, it may be necessary that the processor idles, even if there are jobs in the ready queue, to ensure the schedulability, e.g., when the computation time for some task is larger than another tasks deadline. If a task set is sporadic, no online algorithm can decide whether the processor should idle or not [14]. It was shown that Non-Preemptive EDF (EDF-NP) is optimal among work-conserving non-preemptive schedulers [12], i.e., the processor is not allowed to go idle if at least one job is ready to be executed.

1.7 Task scheduling:

Task Scheduling is an allocation decision. Scheduling decisions allocate resources over relatively short time periods. There must be some parameters are taken for task scheduling. One of the main goals of real-time systems design is to provide temporal guarantees for the real-time tasks. This is typically achieved using schedulability and/or a feasibility test.

• A schedulability test determines whether or not, for any given task set and a specified scheduler, deadlines will be missed in the schedule generated by that scheduler August, 2014

• A feasibility test determines the existence of a valid real-time schedule for any given task set, independent of the scheduling algorithm. If a task set is deemed to be feasible using a suitable feasibility test, the scheduling algorithm that can schedule the task set still needs to be found. There exists several utilization based [40], response time based [41][42] [30] and demand bound based [43] [44] schedulability tests for major uniprocessor and multiprocessor scheduling algorithms. In the uniprocessor case, the well known feasibility tests [43] [45] are derived building on the uniprocessor optimality of several scheduling algorithms e.g., EDF on uniprocessor.

The real-time tasks have to be executed on a computing platform with one or more processors while satisfying the hard or soft timing requirements. The numbers of processors are typically fewer in number than the number of tasks and hence appropriate real-time scheduling algorithms are required to guarantee the timeliness of the real-time tasks sharing the processors. Real-time scheduling algorithms can be classified as either an online or an offline algorithm depending on when the scheduling decision is made. In offline scheduling, the scheduling decisions are made offline and the schedule is stored, e.g., in a table. In this case, the advantage is that, much of the complexities concerning the schedule generation can be handled offline using very complicated tools and techniques, while ensuring a simple runtime mechanism for dispatching the tasks. However, the main disadvantage is that the schedule needs to be recomputed every time there is a change in the task set. In online scheduling, on the other hand, scheduling decisions happen during runtime using suitable criteria, e.g., priorities. The main advantage of using an online scheduling algorithm is that the scheduler has the possibility of adapting to changing factors e.g., tasks executing for less than their worst case. On the other hand, since the scheduling decisions are made at runtime, it can potentially increase overheads in the schedule. Typically, online scheduling algorithms are priority driven, and hence can be classified as fixed task priority, fixed job priority and dynamic priority scheduling depending on the priority assignment policy. In fixed task priority scheduling, the task priorities are assigned offline and all the jobs of any given task execute with the same priority, e.g., Rate Monotonic (RM) scheduling. In fixed job priority scheduling, different jobs of any given task can have different priorities, however, the priority of any given job does not change during its execution, e.g., Earliest Deadline First (EDF) scheduling. In dynamic priority scheduling, on the other hand, the job priorities are recomputed online and the same job of any given task can have different priorities at different time instants, e.g., least laxity first scheduling. In the context of a processing platform with at least two processors (i.e., a multiprocessor system), the scheduling algorithms can also be classified as either partitioned or global scheduling. In partitioned scheduling, the tasks are partitioned onto individual processors using a suitable bin-packing strategy offline, and then the tasks are scheduled using suitable uniprocessor scheduling

algorithms, e.g., partitioned EDF. In global scheduling, the tasks are dispatched to the processors from a global queue according to the specified priority assignment strategy, e.g., Global Preemptive EDF. Real-time scheduling algorithms can also be classified depending on whether the currently executing task can be suspended and replaced with a higher priority task. If a preemptive scheduler is employed, then the currently executing task can be suspended and replaced with a higher priority task. On the other hand, under non-preemptive scheduling, if a higher priority task is released during the execution of a lower priority task, the scheduler waits for the lower priority taskAugust, 2014 to complete before allowing the higher priority task to execute.

2. LITERATURE SURVEY

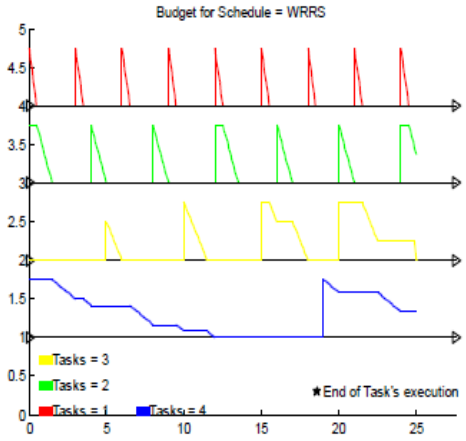
A scheduling algorithm can be seen as a rule set that tells the scheduler how to manage the real-time system, that is, how to queue tasks and give processor-time. The choice of algorithm will in large part depend on whether the system base is uniprocessor, multiprocessor or distributed. A uniprocessor system can only execute one process at a time and must switch between processes, for which reason context switching will add some time to the overall execution time when preemption is used. A multiprocessor system will range from multi-core, essentially several uniprocessors in one processor, to several separate uniprocessors controlling the same system. A distributed system will range from a geographically dispersed system to several processors on the same board. In a distributed system the nodes are autonomous while in a multiprocessor system they collaborate somewhat more, but this line is not as clear cut as it may sound as similar communication delays will occur. In real-time systems processes are referred to as tasks and these have certain temporal qualities and restrictions.[4] All tasks will have a deadline, an execution time and a release time. In addition there are other temporal attributes that may be assigned to a task. The three mentioned are the basic ones. The release time, or ready time is when the task is made ready for execution. The deadline is when a given task must be done executing and the execution time is how long time it takes to run the given task. In addition most tasks are recurring and have a period in which it executes. Such a task is referred to as periodic. The period is the time from when a task may start until when the next instance of the same task may start and the length of the

period of a task is static.

A uniprocessors system is that system having only one central processing unit for the execution of tasks. In uniprocessing all the processing tasks are sharing the single central processing unit. In this the system executes one process at a time and takes the next job when the scheduled process is completed. Burah et.al., [1][2]deals with scheduling algorithms on uniprocessors for preemptive, non-preemptive and complex tasks. Alan Burns et.al., [6]describes problems on uniprocessors and they can be removed by dynamically assigning priority to the tasks. In Strosnider[10]it describes that periodic real time systems having regular arrival times and hard deadlines (a hard work, with the understanding that the deadline will be extended.) while the periodic have deadline, which is carved in stone, a soft deadline provides authors with a target date stop submitting their irregular interval and soft deadlines. This phenomena works when the fewer tasks are in queue. The number of tasks increases; it creates halting and slowing a system. To overcome such problems the concept of multiprocessing introduced.

3. APPROACHE: Real-time computer systems are being ubiquitously deployed in many mission and safety critical applications, and are increasingly becoming the backbone of most modern cyber-physical systems, e.g., autonomous vehicles. They are typically based on contemporary uniprocessor and multiprocessor platforms which support performance enhancing hardware, e.g., caches and instruction pipelines to pre-fetch data and instructions that significantly improve average system performance. Preemptively scheduling hard real-time tasks on such platforms typically imply non-negligible preemption and migration related overheads, potentially causing deadline misses. Consequently, the deployment of such modern processors in real-time systems requires a careful analysis of the resulting hardware-software ecosystem. HighAugust, 2014 preemption and migration related overheads are considered to be an emerging problem in many real-time applications in autonomous vehicles where data intensive operations, such as image processing for vehicular vision systems, form a critical part of the software. On the other hand, as pointed out by Short [5], non-preemptive scheduling is often favored for applications with severe resource constraints due to its low memory requirements and simple implementation. However, non-preemptive scheduling has received less

attention as compared to preemptive scheduling. The basis for our scheduling algorithm is the EDF scheduling policy [47]. The EDF is not always optimal for our synthesis problems. The following summary of observations, which can be easily proved or are already proved, guided us to develop an effective and efficient heuristic for the EDF-based task level scheduler.



SIMULATION RESULTS AND ANALYSIS

The above figure shows the budget schedule for the weighted round robin algorithm. It is similar to the round robin scheduling, but there is a slight difference between weight is assigned to the task. In the weighted round robin scheduling the time slice varied and the period line along with the task. The time space between the two tasks is longer. From the simulation results there is no budget calculation for the weighted round robin scheduling.

CONCLUSION:

Earliest Deadline First algorithms are presented the least complexity according to their performance many of the supposed „problems“ that have been attributed to this type of scheduling technique. It is clear that earliest deadline first is the efficient scheduling algorithm if the CPU utilization is not more than 100% but does scale well when the system is overloaded. In the experimental environment EDF scheduling algorithm can meet the needs of real-time applications.

REFERENCES:

[1]. Jag beer Singh, Satyendra Prasad Singh “An Algorithm to Reduce the Time Complexity of Earliest Deadline First Scheduling Algorithm in Real-Time System”, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 2, No.2, pp 31-35 February 2011

[2]. Jashweeni Nandanwar “An Adaptive Real Time Task Scheduler”, International Journal of Computer Science Issues (IJCSI), Vol. 9, Issue 6, No 1, pp 335-339 November 2012 ISSN (Online): 1694-0814

[3]. Miss. Rina V. Bhuyar1 Dr. D.

G. Harkut "Adaptive Neuro Fuzzy Scheduler for Real Time Task", International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 2, pp 393-396 February 2014 ISSN: 2277 128XI.

[4]. Dr. D. G. Harkut, Anuj M. Agrawal, "Comparison of Different Task Scheduling Algorithms in RTOS", International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, pp 1236-1239, Issue 7, July 2014.

[5]. Michael Short "The Case For Non-Preemptive, Deadline-driven Scheduling in Real-time Embedded Systems", Proceedings of the World Congress on Engineering 2010 WCE 2010, pp June 30 - July 2, 2010, London, U.K.

[6].Umm-I-aiman &Sherafzal kha "review of different approaches for optimal performance of multi-processors", August, 2013 VFAST Transactions on 2013 Volume 1, Number 2, pp 7-11,July-August 2013.

[7]. Sanjoy baruah "Priority-driven Scheduling of periodic task systems on multiprocessors", pp 188-201, 2003.

[8]. [8]. Hamza Gharsellaoui, "New Optimal Solutions for Real-Time

Reconfigurable Periodic Asynchronous OS Tasks with Minimizations of Response Times", pp 1 -18.

[9]. R. Kalpana, S. Keerthika "An Efficient Non-Preemptive Algorithm for Soft Real-Time Systems using Domain Cluster–GroupEDF", International Journal of Computer Applications (0975 – 8887) Volume 93 – No.20, pp 1-7 May 2014.

[10]. Ishan Khera, Ajay Kakkar, "Comparative Study of Scheduling Algorithms for Real Time Environment", International journal of computer applications, Vol 44, No. 2 pp15-22, April 2012.

[11] Leey, Miodrag Potkonjaky and Wayne Wolfz "Synthesis of Hard Real-

Time Application Specific Systems" Chunho yComputer Science Dept.,

University of California, Los Angeles,

CA zDept. of Electrical Engineering, Princeton

University, Princeton, NJ.

[12]. Miao Liu et.all, “On Improving Real Time Interrupt Latencies of Hybrid Operating Systems with Two- Level Hardware Interrupts”, IEEE n Transactions On Computers, Vol. 60, No 7, July 2011, pp. 978-991.

[13]. Mian Dong and Lin Zhong, “Power Modeling and Optimisation for OLED Displays”, IEEE Transactions on Computers,Vol. 11, No 9, September 2012, pp. 1587-1599.

[14]. Q. Li and C. Yao, “Real-Time Concepts for Embedded Systems”. CMP Books, 2003.

[15]. Tangyin, “Real-Time Operating System Application development Guide”, China Electric Power Press, July 2002.

[16]. C. L. Liu, J. W. Layland, "Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment," Journal of the ACM, 20-91, Jan, 1973.

[17]. Ralf Steinmetz, "Analyzing the Multimedia Operating System," IEEE Multimedia, Spring, 1995.

[18]. C.L. Liu, J.W.Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment", Journal of the ACM, 20(1) pages 46-61, 1973.

[19]. M.L. Dertouzos, “Control Robotics: The Procedural Control of Physical Processes”. In Proceedings of the IFIP congress, pages 807-813, 1974.

[20]. A.K. Mok, “Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment,” Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1983.

[21]. J.Y.-T. Leung, J. Whitehead, "On the complexity of fixed-priority scheduling of periodic real-time tasks". Performance Evaluation, 2(4), pages 237-250, 1982.

[22]. M. Joseph, P.K. Pandya, “Finding Response Times in a Real-time System”. The Computer Journal, 29(5), pages 390–395, 1986

[23]. J.P. Lehoczky, L. Sha, Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behaviour". In proceedings IEEE Real-Time Systems Symposium (RTSS), pages 166–171, 1989.

[24]. J.P. Lehoczky, "Fixed priority scheduling of periodic task sets withAugust, 2014 arbitrary deadlines". In proceedings IEEE RealTime Systems Symposium (RTSS), pages 201–209, 1990.N.C. Audsley, A. Burns, M. Richardson, A.J. Wellings, "Applying new Scheduling Theory to Static Priority Preemptive Scheduling". Software Engineering Journal, 8(5), pages 284-292, 1993.

[25]. N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", Technical Report YCS 164, Dept. Computer Science, University of York, UK, 1991.

[26]. N.C. Audsley "On priority assignment in fixed priority scheduling", Information Processing Letters, 79(1): 39- 44, May 2001.

J) K.W. Tindell, A. Burns, A.J. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks". Real-Time Systems. Volume 6, Number 2, pages 133-151, 1994.

K) A. Zuhily, A. Burns, "Optimal (D - - monotonic priority assignment",

Information Processing Letters, Vol 103, No 6, pages 247-250, 2007.

[27]. S.K. Baruah, A.K. Mok, L.E. Rosier, "Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor". In proceedings IEEE Real-Time Systems Symposium (RTSS), pages182-190, 1990.

[28]. S.K. Baruah, L.E. Rosier, R.R. Howell, "Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic Real-Time Tasks on one Processor". Real-Time Systems, 2(4), pages 301-324, 1990.

[29]. L. George, J. Hermant, "A norm approach for the Partitioned EDF Scheduling of Sporadic Task Systems." In proceedings Euromicro Conference on Real-Time Systems (ECRTS), 2009.

[33]. F. Zhang, A. Burns, "Schedulability Analysis for RealTime Systems with EDF Scheduling," IEEE

Airo International Research JournalTransactions on Computers,

[34]. F. Zhang, A. Burns, A. “Schedulability Analysis of EDF Scheduled Embedded Real-Time Systems with Resource Sharing”. ACM Transactions on Embedded Computing Systems 9, 4, Article 39, March 2011.

[35]. S. Dolev and A. Keizelman. Non-preemptive real-time scheduling of multimedia tasks. Journal of Real-Time Systems, 17(1):23–39, 1999.

[36]. Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In Proceedings of International Conference on Computer Aided Design, pages 365–368, Nov. 2000

[37]. F. Zhang and S. T. Chanson. Processor voltage scheduling for real-time tasks with non-preemptible sections. In Proceedings of IEEE Real-Time Systems Symposium, pages 235– 245, Dec. 2002

[38]. H. Aydin, R. Melhem, D. Moss´e, and P. M. Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In Proc. of EuroMicro Conference on Real-Time Systems, Jun. 2001.

[39]. N. Weste and K. Eshraghian. Principles of CMOS VLSI Design. Addison Wesley, 1993.

[40]. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In Proc. of IEEE Symposium on Foundations of Computer Science, pages 374–382, 1995.

[41]. H. Yun and J. Kim. On energy-optimal voltage scheduling for fixed-priority hard real-time systems. Trans. on Embedded Computing Sys., 2(3):393–430, 2003.

[42]. G. Quan and X. Hu. Minimum energy fixed-priority scheduling for variable voltage processors. In Proc. of

[43]. August, 2014 Design Automation and Test in Europe, pages 782–87, 2002

T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processor. In International Symposium on Low Power Eletronics and Design, pages 197–202, Aug. 1998

[43]. Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In Proceedings of International Conference on Computer Aided Design, pages 365–368, Nov. 2000.

[45]. F.Gruian. Hard real-time scheduling for low-energy using stochastic data and dvs processors. In Proceedings of International Symposium on Low Power Electronics and Design, pages 46–51, Aug. 2001.

[46]. P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In Proc. of Symposium on Operating Systems Principles, 2001.

[47]. H. Aydin, R. Melhem, D. Moss´e, and P. M. Alvarez. Dynamic and aggressive scheduling techniques for poweraware real-time systems. In Proceedings of IEEE Real-Time Systems Symposium, pages 95–105, Dec. 2001

[48]. W. Kim, J. Kim, and S. L. Min. A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis. In Proceedings of Design Automation and Test in Europe, pages 788–794, Mar. 2002.

[49]. C. Rusu, R. Melhem, and D. Mosse. Maximizing rewards for real-time applications with energy constraints. ACM Transactions on Embedded Computer Systems, 2(4):537– 559, Nov. 2003.

[50]. C. Rusu, R. Melhem, and D. Mosse. Multi-version scheduling in rechargeable energy-aware real-time systems. In Proceedings of EuroMicro Conference on Real-Time Systems, pages 95–104, 2003.

Author's Profile

Mr. Shaik Saidulu*, His completed UG&PG Engineering from JNTUH. He is a Ph. D Research Scholar from SUNRISE University, Alwar. He has been published 35 international journals and conference Papers in various hi-indexed Journals so far. His projects awarded prizes in various capitations. His research interests are Soft Computing, IoT, Embedded Networking,*

Processor Architectures and Intelligent Systems.

Contact Details:

Shaik Saidulu

Plot N0-101, Road No-3,

Suryodaya Colony, LB Nagar, HYD-500074.

Contact: 91-9603066613
Email:
sk.saidulu@gmail.com

August, 2014

Dr. *Yash Pal Singh is having vast 31 years of experience in Teaching, Research and Industry. He got the Ph.D from poona University, PUNE in the year 1987. He is the member & incharge of many committees formed by AICTE, MHRD, Govt of India etc.* Project In-charge Govt. Of Delhi to generate Internal Revenue for the Govt. he had worked as Programme Coordinator to impart training to the Officers and officials of various Deptt of Delhi of Govt. He is Published 86 National & 210 International papers in Various Hi-Indexed Journals. He registered as Ph.D guide; Subject expert, Guest Lecture and key note speaker of various well known universities. As on 16 Ph.d guided and 11 ongoing.

Consultancy Projects:

1) U.T.S. system, Northern Railway,Delhi.

2) Networking Support for IMPRESS System for IRCTC.

3) Training on soft skills for P.G.T.‟ S,

Vice Principal for Directorate of Education, Govt of Delhi.

4) Indian Army for the Microwave tubes and other allied antennas components.

Books Published:

(I) Electronics & Materials.

(II) SEMICON DUCTOR DEVICES (III) Introduction to Computers. (IV) BASIC ELECTRONICS

(V) Laser and its application

(VI) Wireless & Mobile Communications

Contact Details:

AN EFFECTUAL NON-PREEMPTIVE SCHEDULING METHODOLOGY THROUGH APERIODIC AND SPORADIC TASKS IN REAL-TIME EMBEDDED PROCESSING SYSTEMS

Dr.Yash Pal Singh

R.Z.E.-II/24,New Roshanpura,

Najafgarh, New-Delhi- 43 (INDIA)

Mob**:09810516527** & **011-25015911**

E-mail: ypsingh10@rediffmail.com