Journal of Advanced Research in Technology and Management Sciences
Volume: 01 Issue: 03 ISSN: 2582-3078 November 2019
Available online at: http://www.jartms.org

Survey of Real Time Task Scheduling Algorithms for Multicore Processors

[1]Akhila.B, [2]Lordwin Cecil Prabaker.M and [3]Brahmaiah Naik.J

[1] PG Scholar, Dept of ECE, Vignan's Lara Institute of Technology and Science, Guntur
Akhilaboggavarapu456@gmail.com

[2] Associate Professor, Dept of ECE, Vignan's Lara Institute of Technology and Science, Guntur, cecillord@gmail.com

[3] Associate Professor, Dept of ECE, Vignan's Lara Institute of Technology and Science, Guntur, brahmaiahnaik@gmail.com

ABSTRACT
We consider a survey on scheduling of real time on multicore processors. In this survey, we observe that the cores are distributed for the incoming task according to their workloads. Depends on the task workload the cores may be splitted into big and small core. The big cores are having more Floating Point and multiply accumulate(FPMAC) units also the small cores having less number of FPMAC units. We can use the big core for complicated tasks and small one for easy tasks. Because of this dividing of cores we can use cores efficiently and the output will be reliable one. The survey outlines fundamental results about multicore processor real-time scheduling that hold independent of the scheduling algorithms employed. The scheduling algorithm such as LTF scheduling, EDF scheduling, Priority scheduling are used according to their algorithm.

Keywords: Multicore Processor, Realtime schedulling, Schedulling Algorithms.

1. INTRODUCTION

A Multicore processors is a computer processor integrated circuit with two or more separate processing units, called cores, each of which reads and executes program instructions, as if the computer had several processors. A scheduler is a software product that allows an enterprise to schedule and track computer batch tasks. By using the schedulers we can execute the high priority task and just hold on to low priority task.

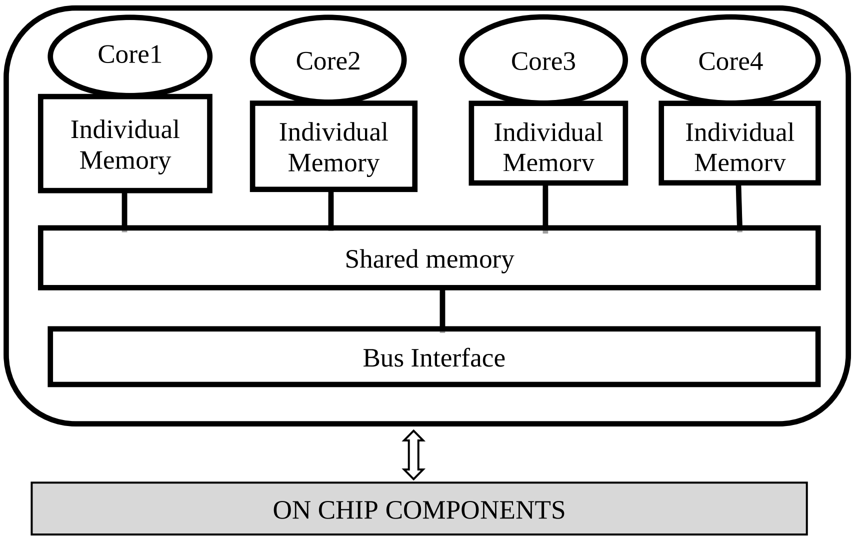

Fig 1: Structure of multicore processor

Today, real-time embedded systems are found in many applications such as automotive electronics, avionics, telecommunications, space systems, medical imaging, and consumer electronics. By using real time the above areas has gained improvement in their fields. Companies building embedded real-time systems are driven by a profit motive. To succeed, they aim to meet the needs and desires of their customers by providing systems that are more capable, more flexible, and more effective than those of their competitors ,and by bringing these Systems to market earlier. This desire for technological progress has resulted in a rapid increase in both software complexity and the processing demands placed on the underlying hardware.

2. SYSTEM MODEL AND TASK MODEL

2.1 System Model for dynamic scheduling heterogeneous task:

The heterogeneous Multicore Architecture consists of following two modules:

- Module I: Consisting of simple small cores with low power. The cores are homogeneous in nature. This module is dedicated for executing low power tasks and the module is operational upto a certain low power level. $0 < P <= p_i$, where, P is the power level and $p_i$is the maximum power level to operate the cores. The module I consist of a number of small low power similar types of cores as described in the work. In addition to this we have incorporated a run time dispatcher in the architecture. The roles of the online dispatcher are as follows: To dispatch the matching jobs into small cores. To migrate the rejected jobs from small cores into big cores for execution. These simple small cores are designed for high parallel performance with low power.
- Module II: Consisting of complex big cores with high power and is responsible for executing high power tasks and the module is operational from a certain high power level. p_i<P<=$p_m$where p_m is the maximum power level to operate the cores.

Total number of cores is C where,

$$C = c_s + c_b \tag{1}$$

(c_s= total number of small cores; c_b = total number of big cores).

Each small and big core has their own first level cache L1 and a second level L2 cache which is shared by all core. The architecture of module II has been incorporated additionally where a few big complex cores are placed for high serial performance. These cores are designed for highly specialized application-specific task with high power. Each big core has their own local queue and local cache (L1) to store the task for allocation into the core. Ready Queues equal to the number of big cores have been incorporated to store all the instance of a particular big complex task to a particular Ready Queue as we consider the partition schema where all the instances of a task are executed on the same core. The instances of the task are next forwarded from the Ready Queue to local queue and subsequently into the big core corresponding to the Ready Queue using task partitioning scheduling.

2.2 Task Model

In the earlier study the task model is considered as,

$$u_{ij} = x_{ij}/P \tag{2}$$

Where,X_{ijB}<=X_{ijS} for all Ti and the utilization of the big cores satisfies 0 <=X_{ijB}/P <= 1 when X_{ijB}<P. If X_{ijB}> P, then we set U_{ij} to ∞ to indicate that task Ti cannot be mapped into the core C_j.

This results a utilization matrix

$$U = [U_{ijnxm}] \tag{3}$$

where n is the number of tasks and m is the number of cores.

2.4 System model for DAG task(parallel recurrent task)

We consider the preemptive GFP scheduling of n recurrent DAG tasks in set

$$\Gamma= \{G_1, ...G_n\} \qquad (4)$$

A multicore processor having m cores. The (normalized) speed of each core is 1. Each DAG task $G_k \in \Gamma$ generates potentially infinite number of DAG instances, called the jobs of the task, such that the release of two consecutive jobs of G_k is separated by a minimum distance. Each DAG task G_k is characterized using four parameters

$$G_k = (T_k, D_k, V_k, E_k) \qquad (5)$$

Where
• T_k is the minimum inter-arrival time of consecutive jobs (also, called the period) of task G_k;
• D_k is the relative deadline such that $D_k \leq T_k$;
• $V_k = \{v_k, 1, ...v_k, n_k\}$ is a set of n_k subtasks2; and
• $E_k \subseteq (V_k \times V_k)$ is a set of directed arcs (or edges).

The parameter D_k specifies the real-time constraint. If the source (i.e., the first) subtask of some job of task $G_k$becomes ready for execution at time t. The execution of all the subtasks of that job must complete by time (t $+D_k$).

3. SCHEDULERS IN MULTICORE ARCHITECTURE

In real time systems, process scheduling refers to the order of execution of real time tasks so that they do not miss their deadlines. During the early 1970's, several real time scheduling algorithms were extensively studied. This paper analyses a few widely used existing real time scheduling algorithms. A few important terms related to real time scheduling are:

- Utilization (Processor utilization) - It refers to the fraction of time spent by the CPU in executing a set of tasks.
- Schedulability Test - It is a test to determine whether the tasks can meet their deadlines.
- Optimal Scheduler - If a set of tasks cannot be scheduled by an optimal scheduler, then it cannot be feasibly scheduled by any other scheduler. Tasks are said to be feasibly scheduled if all of them meet their deadlines.

4. SCHEDULING ALGORITHM

Scheduling is the method by which work is assigned to resources that complete the work. The work may be virtual computation elements such as threads, processes or data flows, which in turn scheduled onto hardware resources such as processors, network links or expansion cards. Schedulers are often implemented so they keep all computer resources busy. The scheduling algorithms are priority scheduling, shortest job first, round robin scheduling, earliest deadline first. Some of them are explained below.

4.1 The LTF scheduling algorithm

To schedule a list L = $(T_1, T_2, ..., T_n)$ of parallel tasks on a single multicore processor with M cores, algorithm LTF first sorts the n tasks in a non increasing order of task sizes. Without loss of generality, let us assume that ,

$$S_1 \geq S_2 \geq \cdots \geq S_n.$$

Algorithm LTF then schedules the n tasks in the order of $T_1, T_2, ..., T_n$.

Initially, tasks in the beginning of L, say$T_1, T_2, ..., T_i$ are scheduled for execution,
where i is as large as possible, i.e., the total size of $T_1, T_2, ..., T_i$ does not exceed M, $S_1 + S_2 + \cdots + S_i \leq$M, but the total size of $T_1, T_2, ..., T_i, T_{i+1}$ exceeds M, $S_1 + S_2 + \cdots + S_i + S_{i+1} >$ M.

Whenever a task is completed, algorithm LTF checks whether there are enough available cores for taskT_{i+1}. If so, task Ti is scheduled for execution; otherwise, algorithm LTF waits for the next completion of a task. The above procedure is repeated until all tasks are scheduled.

4.2 Task-level Priority Assignment

The fixed priorities of the n tasks $\{G_1,G_2, \ldots G_n\}$ are assigned based on Deadline Monotonic (DM) priority ordering

4.2.1 Subtask-level Priority

During the subtask-level priority assignment, a subtask at a lower level of the DAG is given higher priority in comparison to the one at a higher level. If two subtasks have the same level, then the subtask having relatively higher (subtask) index is given higher priority. Given a subtask V_k, j $\in V_k$ of particular task Gk, the set of subtasks in V_k having higher fixed priorities than that ofV_k, j.

4.2.2 The two-level GFP Scheduler

The subtasks of the DAGs are executed based on a two-level scheduler. The task-level scheduler determines the highest-priority ready task and the subtask level scheduler selects the highest-priority subtask of for execution. A newly released relatively higher-priority subtask is allowed to preempt the execution of a lower-priority subtask when all the cores are busy. Under the subtask-level fixed-priority assignment policy, subtask j has lower fixed priority than all the subtasks in set.

4.3 EDF scheduling

Earliest-deadline first (EDF) and least-laxity first (LLF) are optimal dynamic priority algorithms for uni-processor systems. Dynamic priority algorithms provide better system schedulability but suffer from high run-time costs.
The multicore real-time scheduling techniques are generally divided into three categories: partitioned, global and semi-partitioned.

4.3.1 Partitioned technique

partition the given task set on the basis of some bin-packing heuristics such as First-Fit (FF), Next-Fit (NF), Best-Fit (BF), Worst-Fit (WF) etc., and assign each partition to a separate processing core. Each processing core executes its assigned portion of tasks using some uni-processor scheduling approach. During execution; task migrations are not allowed.

4.4 Global scheduling

In this technique, all of the tasks are placed in a single prioritized queue and the scheduler assigns them to cores according to their priority. Therefore, during execution tasks can migrate from one core to another. Generally, global techniques perform better than partitioned approached but they incur more run-time overheads. Semi-partitioned scheduling is presented as a compromise between pure partitioned and global scheduling in order to reduce the runtime overheads associated with the global scheduling and to improve the performance of partitioned scheduling.

4.5 Semi-partitioned Technique

In this technique extend the partitioned scheduling by allowing a small number of tasks to migrate which results in improved performance.

5. TAXONOMY OF MULTICORE PROCESSORS

The problem in Dynamic Scheduling of Real-Time Tasks in Heterogeneous Multicore Systems is
Our objective is to introduce a real-time scheduling algorithm in heterogeneous multicore system where tasks are allocated to cores based on the type of tasks as well as cores. There are two types of tasks and cores in terms of power namely low power task and high power task.

5.1 General Task Problems

In general, a task is called low or high power task if it has the following properties:

- Lower or higher data rate, hence lower or higher energy spent per unit of time to process the data.
- Less or more complex in nature, so numbers of computation steps are less or more which
- Consumes low or high energy.
- Circuit complexity is less or high or operates at lower or higher clock frequency. Multiprocessor scheduling can be viewed as attempting to solve two problems.
- The allocation problem and priority problem
- Scheduling algorithms for multiprocessor systems can be classified according to when changes to priority and allocation.

5.1 According to Allocation Problems

- No migration - Each task is allocated to a processor and no migration is permitted.
- Task-level migration - The jobs of a task may execute on different processors; however, each job can only execute on a single processor.
- Job-level migration.

5.2 According to Priority

- Fixed task priority- Each task having a single fixed priority applied to all of its jobs.
- Fixed job priority –The job may have different priorities, but each job has a single static priority. An example of this is earliest deadline first t (EDF) scheduling.
- Dynamic priority- A single job may have different priorities at different times, for example least laxity first (LLF) scheduling.

5.3 Schedulability, Feasibility, and Optimality

If a task is said to be feasible in a given system and there exists some scheduling algorithm that can schedule all the tasks and execute without missing any deadlines.
If a scheduling algorithm is said to be optimal with respect to a system and a task model if it can schedule all of the tasks that comply with the task model and are feasible on the system.
If a task is referred to as schedulable according to a given scheduling algorithm if its worst-case response time under that scheduling algorithm is less than or equal to its deadline. Similarly, a tasks is referred to as schedulable according to a given scheduling algorithm if all of its tasks are schedulable. The following table.1 summaries the earlier work and the various simulators used for evaluation.

Table 1. Summary of recent multicore processor scheduling techniques

References	**Domain& Simulator**	**Algorithm & Implementation**	**Inference**
1	Multicore Processor& cycle-accurate simulator	Optimization	Proposed a four-step methodology for efficient design space exploration and tunable parameters optimization for multicore/manycore architectures.
2	Multicore Processor	Prediction	Some of the most popular prediction and classification for prediction techniques employed across multiple levels of the computing stack.
3	Multicore Processor& state-of-the-art and sniper simulator	Instruction Level Parallelism & Single Instruction Multiple Data	We present a new analytic model for predicting application performance on modern multi-core processors with vector extensions. It captures contention on shared caches and memory bandwidth, and differentiates ILP per instruction type for increased accuracy.
4	Multicore Processor	Configuration	Application specific soft processor descriptions in Verilog HDL are generated automatically using the configurator which analyses the application program and extracts the only components/paths necessary for the application to be executed as well as provides options in terms of the number of processor cores and the structure of cache memory.
5	Multicore Processor	Scheduling	CAMPS accurately tracks the progress that the various threads in the workload make when running on the different core types throughout the execution, and enforces fairness by evening out the progress across threads.
6	Multicore	Multithreaded	Custard: ASIC Workload-Aware Reliable Design for

	Processor& logic simulator	Switching activity	Multicore IOT Processors.
7	Multicore Processor& c programming environment simulation	Dynamic Scheduling	It incorporates heterogeneity in dynamic scheduling of tasks for multi-core real-time systems
8	Multicore Processor& cycle-accurate instruction set simulator	Implantation-CMOS technology	Compared with other embedded processors, HAVA improves the performance of low-bit-rate vocoders by over 40% with nearly no efficiency loss for multichannel processing
9	Multicore Processor	Allocation	we have addressed the problem as multivariable optimization problems and solved them algorithmically and numerically. Our approach is based on a queueing model of a virtual machine
10	Multicore Processor& 8086 simulator	Scheduling	We propose a reliability-aware scheduler that samples the reliability characteristics of running applications on either core type, and dynamically schedules applications on big versus small cores to improve overall system reliability.
11	Multicore Processor	Scheduling	A simple method to assign fixed priorities to the subtasks of a DAG task is proposed based on the structure (i.e., topology) of each DAG task.
12	Multicore Processor&grid-type thermal simulation	Scheduling	This work breaks away from the common assumption of initially static operation modes in 3D-MCP systems and to develop dynamic operation-mode prediction with machine learning models to reduce hotspots and optimize system performance.
13	Multicore Processor& electromagnetic simulation	Sequential	This paper shows that sliding window Gauss–Seidel exhibits scalable behaviour on CMP, particularly when the problem size is larger than the available cache.
14	Multicore Processor	Advanced Encryption Standard based on Counter with Chaining Mode	We have proposed an Authentication based Matrix transformation cum Parallel-encryption implemented on Multicore Processor to achieve a comprehensive performance, high throughput and secure AES-CCM, yet low overheads of the power dissipation and the latency.
15	Multicore Processor	Accelerating Wait-Free	These two techniques can help a wait-free algorithm leverage a multicore processor’s caches and write buffers better. Besides, they can be applied to a wait-free algorithm while maintaining the control flow of the original algorithm without dramatic changes.
16	Multicore Processor	Joint DVFS- QoS-aware Task mapping	To optimally solve the MILP problem, we propose a JDQT algorithm. It reduces the computational complexity by iterating two smaller, but highly correlated, sub problems: an MP problem for task-to-processor allocation and frequency-to task assignment, and a SP problem for task scheduling and task adjustment.
17	Multicore Processor	Low Density Parity Check decoding	We have presented an efficient LDPC software decoder in terms of throughput for multi-core device. This LDPC decoder was implemented on an x86 processor.
18	Multicore Processor	Optimization	We propose PH-Sifter, a fast operating point sifting technique that starts at the lowest operating point, filters all the points until the next operating point that achieves the highest Delta Performance / Delta Power value, and continues until all the points are considered.
19	Multicore Processor& SPICE simulation	novel decomposition ,Optimal Joint	We formulated the problem of task mapping on DVFS-enabled homogeneous multicore as an MILP, with the goal of maximizing QOS without violating the real-time and the

		Task Mapping	energy supply constraints. This problem is optimally solved by the OJTM algorithm. A novel algorithm, HJTM, is proposed to further reduce the computational complexity.
20	Interconnection & event-driven simulator	routing	We designed a new type of wide-sense non-blocking interconnection network for multicast connections, the MCL. Our motivation is to deal with the conflict between the performance and the cost. We managed to reduce the cost of the interconnection networks without harming their non-blocking multicast capability.
21	Interconnection	Implementation on silicon chip	Proposed a Capacitive-Coupling based RF interconnection for the heterogeneous integration of silicon-chip and printed paper electronics. Inkjet printing technology is utilized as a cost-effective manufacture approach to fabricate the interconnections on paper substrate.
22	Multicore Processor	Adaptive Deblocking Filter Order	This paper proposes an architecture of ILF to effectively provide parallelism among multiple decoding cores for high performance applications. The important feature of the architecture is enabling multicore decoding using novel memory organization and efficient neighbouring data control for tile or WPP without additional ILF for partition boundaries
23	Multicore Processor& Monte Carlo simulation	co-schedulers	In this paper, a graph is constructed for the co-scheduling problem. Then finding the optimal co-scheduling solution is modelled as finding the shortest valid path in the graph.
24	Multicore Processor& all type of simulations	framework	This paper introduces a novel multicore framework that leverages isolationatthegranularityofclustersofcorestoprovidedeterministic execution of multiple concurrent applications, along with efficient resiliency and security guarantees.
25	Multicore Processor& DineroIV trace-driven simulator	Prediction	A new cache replacement policy “PVISAM” is presented in this article to allow unified, shared cache miss upper bound prediction by using the existing private cache prediction techniques. Additionally, the policy makes efficient utilization of cache space.
26	Multicore Processor& architecture simulators	Parallel	Our solution combines a high-performance core suitable for sequential execution, and several lightweight low power cores devoted to parallel execution.

6. CONCLUSION

The survey among the real time scheduling multicore processors are performed in this article. By using the multicore processors we can execute multiple tasks and we can save the time also. They have used the dynamic scheduling algorithms to give priority for their tasks to execute. According to the workload the cores are splitted into small and big cores. Because of splitting the tasks can efficiently usage of the cores and moreover the result will be accurate and efficient.

References

[1] F. Oboril and M. B. Tahoori, “ExtraTime: Modeling and analysis of wear out due to transistor aging at microarchitecture-level,” in Proc. 42nd Annu. IEEE/IFIP Int. Conf. Dependable Sys.

[2] D. Koufaty, D. Reddy, and S. Hahn “Bias Scheduling in Heterogeneous Multi-core Architectures” in Proc.

[3] N. Chitlur, G. Srinivasa, S. Hahn, P. K. Gupta, D. Reddy, D. Koufaty, P. Brett, A. Prabhakaran, L. Zhao, N. Ijih, S. Subhaschandra, S. Grover, X. Jiang, and R. Iyer "QuickIA: Exploring Heterogeneous Architectures on Real Prototypes", in Proc. HPCA/IEEE, Washington, DC, USA, 2012, pp. 1-8.
[4] G. Tong and C. Liu "Supporting Soft Real-Time Sporadic Task Systems on Uniform Heterogeneous Multiprocessors with No Utilization Loss", IEEE Transactions on Parallel and Distributed Systems, vol. 27, no.9, pp. 2740-2752, Sep, 2016.
[5] K. Baital and A. Chakrabarti, "An Efficient Dynamic Scheduling of Tasks for Multi-core Real Time Systems", in Advances in Computing Applications, Chakrabarti A., Sharma N., Balas V. (eds), Singapore: Springer, 2016, pp. 31-47.
[6] W. Munawar, H. Khdr, S. Pagani, M. Shafique, J. Chen, and J. Henkel "Peak Power Management for Scheduling Real-time Tasks on Heterogeneous Many-Core Systems", in Proc. ICPADS/IEEE, Hsinchu, Taiwan, 2014, pp.200-209.
[7] S. Baruah, "Task partitioning upon heterogeneous multiprocessor platforms", in Proc. RTAS/IEEE, Toronto, Canada, 2004, pp. 536543.
[8] M. A. Kinsy and S. Devadas "Algorithms for Scheduling Taskbased Applications onto Heterogeneous Many-core Architectures", in Proc. HPEC/IEEE, MA, USA, 2014, pp. 1-6.
[9] C. Tan, T. S Muthukaruppan, T. Mitra, and L. Ju " Approximation-Aware Scheduling on Heterogeneous Multi-core Architectures", in Proc. ASP-DAC/IEEE, Chiba, Japan, 2015, pp.618-623.
[10] F. Yan, L. Cherkasova, Z. Zhang, and E. Smirni "DyScale: a MapReduce Job Scheduler for Heterogeneous Multicore Processors", IEEE Transactions on Cloud Computing, vol. 5, no. , pp. 317-330, 2017.
[11] Y. Li, J. Niu, and M. Qiu "Energy Efficient Scheduling with Probability and Task Migration Considerations for Soft Real-time Systems", in Proc. ComComAp/IEEE, Beijing, China, 2014, pp. 287-293.
[12] D. Liu, J. Spasic, P. Wang, and T. Stefanov "Energy-Efficient Scheduling of Real-Time Tasks on Heterogeneous Multicores Using Task Splitting", in Proc. Int. Conf. RTCSA/IEEE, Daegu, South Korea, 2016, pp. 149-158.
[13] S. Saha, Y. Lu, and J. S. Deogun "Thermal-Constrained EnergyAware Partitioning for Heterogeneous Multi-Core Multiprocessor Real-Time Systems", Int. Conf. RTCSA/IEEE, Seoul, South Korea, 2012, pp.41-50.
[14] F. A. Bower, D. J. Sorin, and L. P. Cox "The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling", IEEE Micro, vol. 28, no. 3, pp. 17-25, Jun, 2008.
[15] Y. Chen, H. C. Liao, T. Tsai "Online Real-Time Task Scheduling in Heterogeneous Multicore System-on-a-Chip", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 1, pp. 118-130, Jan, 2013.
[16] S. Xu, I. Koren and C. M. Krishna "Thermal-Aware Task Allocation and Scheduling for Heterogeneous Multi-core CyberPhysical Systems", Int'l Conf. Embedded Systems, Cyber-physical Systems & Applications ESCS, Las Vegas, USA, 2016.
[17] D. Chen, A. Mok, and S. Baruah "On Modeling Real-Time Task Systems", in Lectures on Embedded Systems. EEF School 1996. Lecture Notes in Computer Science, Rozenberg G., Vaandrager F.W, Berlin, Heidelberg, Springer 1998.
[18] A. S. Radhamani and E. Baburaj, "Performance Efficient Heterogeneous Multicore Scheduling Strategy based on Genetic Algorithm", ARPN Journal of Engineering and Applied Sciences.
[19] A.Naithani, S.Eyerman, and L.Eeckhout, "Reliability-aware scheduling onheterogeneousmulticoreprocessors,"inProceedingsofthe23rdIEEE Symposium on High
[20] Performance Computer Architecture (HPCA).
[21] G.Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Third Edition, Springer.

[22] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in Proc. of AFIPS, 1967.

[23] R. D. Blumofe and C. E. Leiserson, "Scheduling multithreaded computations by work stealing," Journal of the ACM, vol. 46, no. 5, pp. 720–748, 1999.

[24] K. Lakshmanan, S. Kato, and R. Raj Kumar, "Scheduling parallel realtime tasks on multi-core processors," in Proc. of RTSS, 2010.

[25] G. Nelissen, V. Berten, J. Goossens, and D. Milojevic, "Techniques optimizing the number of processors to schedule multi-threaded tasks," in Proc. of ECRTS. H. S. Chwa, J. Lee, K.-M. Phan, A. Easwaran, and I. Shin, "Global edf schedulability analysis for synchronous parallel tasks on multicore platforms," in Proc. of ECRTS.

[26] C. Maia, M. Bertogna, L. Nogueira, and L. M. Pinho, "Response-time analysis of synchronous parallel tasks in multiprocessor systems".