Random Task Scheduler Algorithms as a Comparison and Access to the Best to Use in Real Time

Haeder Talib Mahde Alahmar
ALIraqia University, College of Engineering, Computer Engineering Department, Baghdad, Iraq
haeder.ahmar@aliraqia.edu.iq

Abstract--Today's real-time systems are the core of most ICT applications. The rapid development of such systems has attracted researchers' attention to optimize performance and to minimize as much as possible the problems and disadvantages they suffer in order to improve their performance in proportion to the volume of tasks entrusted to them.

There are many major challenges facing real-time systems, which are mainly the problem of task scheduling on processor nuclei in the quantity of multi-core processors. Several methods have been proposed, including the general method, where any task can be executed on any kernel, the split method depends on the allocation of a specific kernel for each specific set of tasks. There is also the semi-fragmented method, which is a hybrid of the two previous methods, where a set of tasks is assigned to be executed on a particular kernel, while other functions are allowed to execute on any nucleus of the nucleus Treatment.

In this paper we compare the performance of random task scheduling algorithms on a multi-core platform in order to determine the best algorithm in terms of a set of parameters adopted by researchers in this field, which in turn gives us precise details about the quality of such algorithms when applied to a set of distributed random tasks The unified logarithmic probability.

The simso simulator, which has proven the reliability of high performance by many researchers in this field as well as provides the possibility of generating tasks according to specific probability distributions, and simulates accurate details in-depth characteristics of random tasks.

Keywords--Scheduling, Random Tasks, Multi-Core Processor, Probability Distribution.

1 INTRODUCTION

A real-time system performs a set of tasks. The task is defined as the basic implementation unit in the program, which results in the execution of a given result and constitutes a basic service from the services provided by any real-time system or application [1].

Sporadic Tasks are an important part of the software control systems and are present in most real-time systems such as fire alarms at a particular facility. But the main problem is usually the speed of response to such tasks when they are received because they are randomly generated without breaks but sometimes it is possible to predict roughly when these tasks will occur [2].

2 The importance of the research and its objectives:

The main objective of this research is to test the scalability of random tasks when applying a set of algorithms used in scheduling on a real-time operating system which consists of a set of periodic tasks in order to access algorithms capable of respond to random tasks when they are received and implemented in a way that ensures that the time constraint associated with this [3] type of task is not exceeded.

The SIMSO is used as an effective simulator by many researchers in this field because it simulates accurate details that delve deeper into random tasks such as predictability of repetition times [4] or the possibility of subjecting these times to a given probability distribution.

3 Research methods and materials:

The following is a brief description of the nature of the real-time system in terms of the characteristics of the tasks in it, as well as the characteristics related to the hardware of the system processor, memory and so on[5].

3.1 Processor Architecture

The processor used in this system is the multi-core processor, which is the architecture of modern models of processors, which was addressed after the emergence of the problem of inability to increase the frequency of a single-core processor to a high value enables us to obtain high performance and effective[6], as each increase Above the highest frequency reached result in additional problems that are deepened by the emergence of a number of parasitic capacities that usually arise in electronic circuits at high frequencies[7], and effective

synchronization between the elements and gates in the processor circuit is difficult due to the time delay caused by the work of some In the logical processor and accounts. [8] [4]

Figure (1) shows the architecture of the multi-core processor, where we observe how the cores are placed in addition to the cache memory at different levels[9][10].

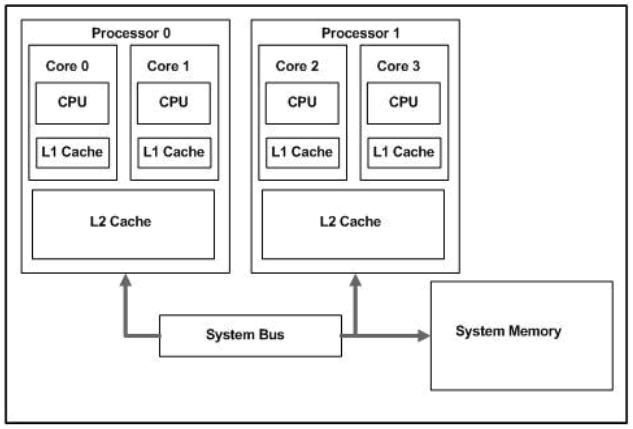

Figure (1) shows the architecture of the multi-core processor

3.2 Sporadic Tasks Model:

The sporadic tasks used in this research follow the following model, where each task is assigned a set of distinct parameters: [11] [7]

1. Time of receipt of the assignment: It is described as the time when a notice or notification of a new event is reported in the real-time system[12], and two events are not required at the same time for this type of task:

$$\forall A, B \in E \implies t_A \neq t_B \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

2. The smaller interval (ε) between each successive frequency of one or two different events is always positive[13], ie:

$$\forall A_1, A_2, B \in E \implies |t_{A_1} - t_{A_2}| \geq \varepsilon \; and \; |t_{A_1} - t_B| \geq \varepsilon \ldots(2)$$

3. Execution Time It is the processor occupancy time by task to execute [14].
4. Time constraint: It is described as the time during which the task is to be carried out before it is reached [15].

3.3 Simulation program used:

The SIMSO simulator was used in this research because it has the following characteristics [16]:

1. Open source: Any file from the libraries contained in this emulator can be modified, including files that describe the work of scheduling algorithms. [17] [6]
2. Graphical User Interference [18]: It Provides a set of menus and options that allow you to adjust the simulation settings quickly and efficiently.
3. Supports the introduction of values for detailed parameters within the simulation options, and mention of these parameters[19]:
   - Clock Hour Instruction (CPI per Instruction): This is the number of rotations required to perform each of the instructions included in the task.
   - Instruction Number [20]: The number of instructions that the task contains, since the task can sometimes contain more than one instruction [21].
   - Memory Access Rate [22]: The ratio between the numbers of generalizations that require access to memory relative to the number of quantitative instructions, called MIX. [23] [20]
   - Stack Distance Profile (SDP): It includes the distribution of the various cache contents called Cache Lines in the cache memory and is measured by the number of unique Cache Lines that separate two consecutive accesses to the same Cache Line.
4. It is written in the Python language, which is one of the advanced programming languages. It also supports Object Oriented Programming (Object Oriented Programming)
5. It also supports an interactive graphical interface that allows the user to use the emulator without having to install the software on the operating system. And figure (2) shows the interactive graphical interface on the Internet [8].

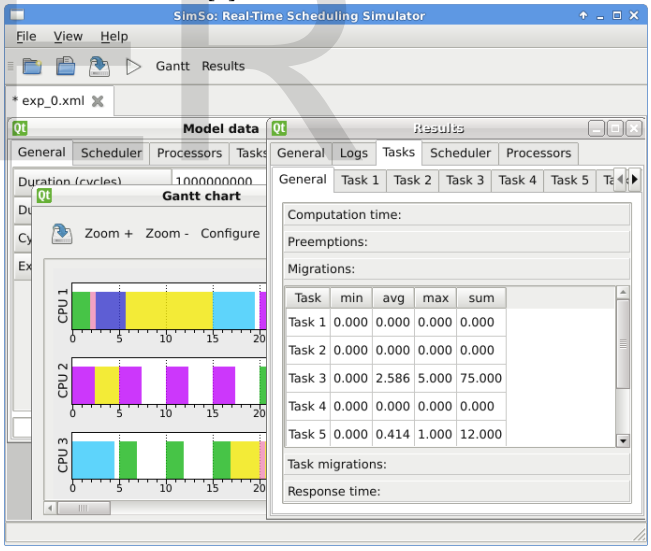

Figure 2 shows the graphical interface of the SIMSO simulator

4 Results and discussion:

Three different scenarios were studied, including three algorithms:

4.1 PD2 (Pseudo Deadline) algorithm
4.2 EARF (Earliest Deadline First) algorithm
4.3 LREF (Largest Local Remaining Execution time First) algorithm[11]

In the first scenario, simulations are carried out on two nuclei, in the second scenario on four nuclei, and in the third

scenario on eight nuclei. Taking into account the following parameters:

1. Each kernel contains L1 cache, 4 kb cache, 1 cycle and 9 cycle cost.
2. L2 Cash shared between all nuclei of 64 kb and 10 cycle time and cost of loss of 90 cycle.
3. (1 cycle = 1 nano second), considering that (100 cycles) in RAM are the cost of access
4. Cost of context switching (100 cycles).

Table (2) also shows Sporadic Tasks with its parameter values [10].

Table (1) shows random tasks

Task	Execution Time WCET(ms)	Activation Dates(ms)	Deadline (ms)	Instructions Count	Instruction ratio Memory MIX
T1	2	5, 25, 40, 55	11	225111	1.3
T2	3	10, 30, 45, 65	12	260000	1.35
T3	3.6	20, 35, 60, 75	13	320000	1.4
T4	4	10, 40, 60, 80	15	350000	1.5

Results were compared based on the following parameters:

- Load CPU processor: The amount of time the processor is busy performing for the quantum time of simulation.
- Context Switching Overheads: This includes the elapsed time until a task that is currently being performed is assigned another task of higher priority than the current task, resulting in the saved task status and loading the processor recorders with new values for the other task [13].
- Scheduling Overheads: The time consumed by the processor includes a new event that calls the scheduler call to make a scheduling decision and the consequent processing of some operations so that it can make the scheduling decision [14].

4.4 The first scenario:

In this scenario, the application of the three scheduling algorithms mentioned above was simulated on a set of random tasks. Figures (3), (4) and (5) compared the three scheduling algorithms in terms of processor load [20], load of context switching operations and scheduling workloads on the two.

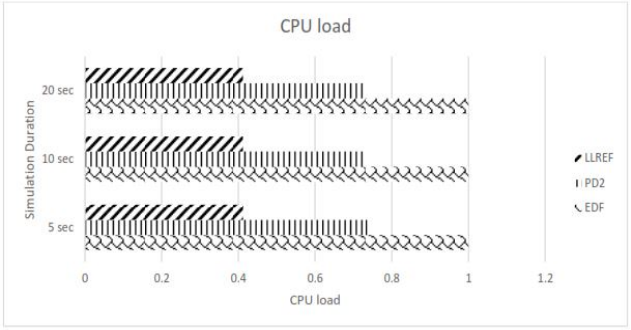

Figure (3) Comparison in terms of processor load.

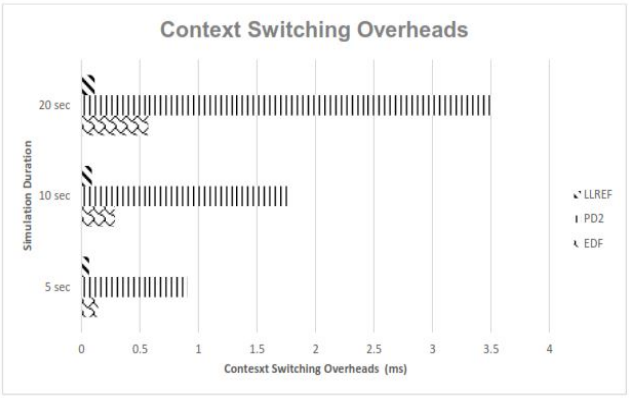

Figure (4) shows the comparison in terms of context switching loads

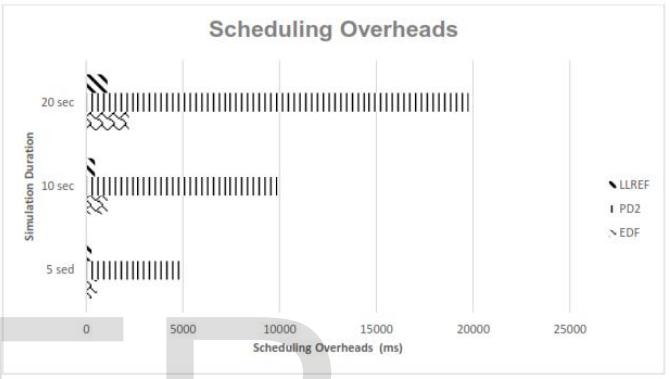

Figure (5) shows the comparison between scheduling loads

4.5 The second Scenario:

In the second scenario, the psychological steps taken in the first scenario were followed but with four nuclei. Figures (6), (7) and (8) show the comparison of the three scheduling algorithms in terms of processor load[23], load of context switching operations, when working on a fourth point.

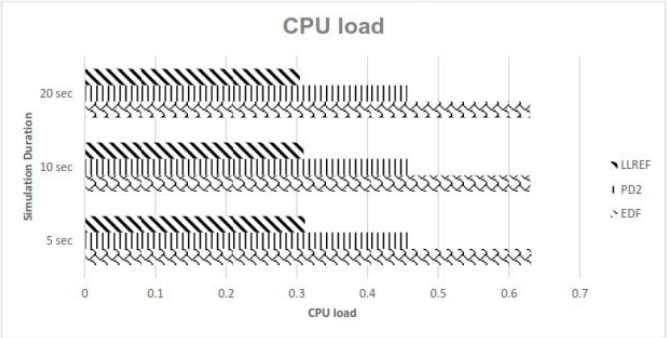

Figure (6) shows the comparison in terms of processor load

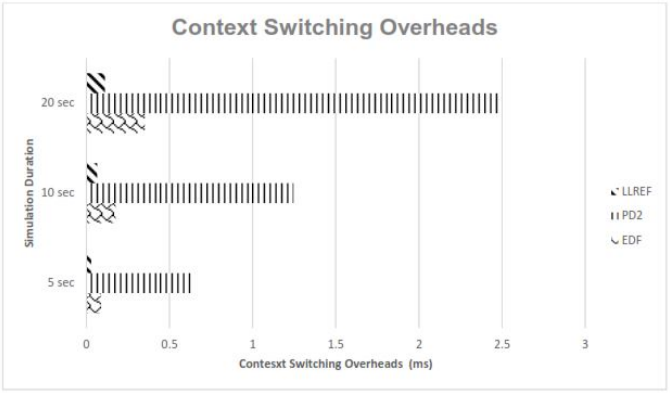

Figure (7) shows the comparison in terms of context switching loads

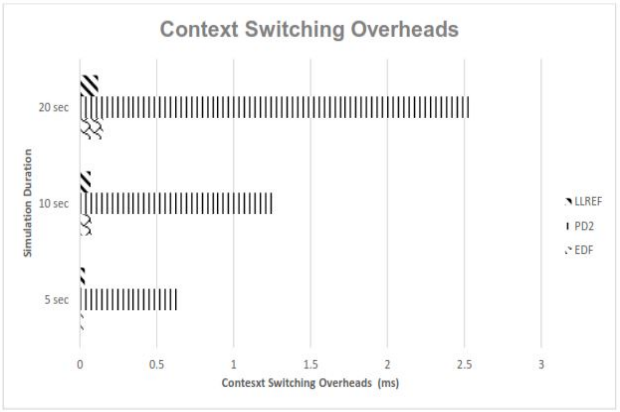

Figure (10) shows the comparison in terms of context switching loads

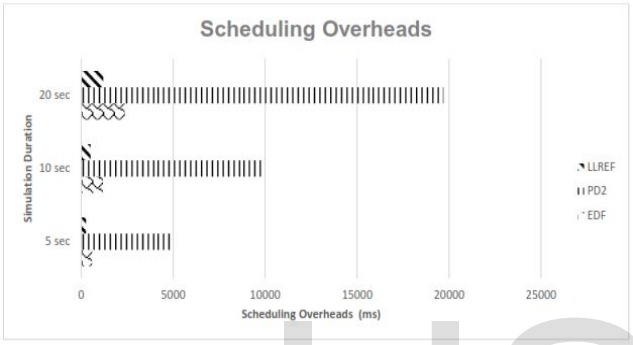

Figure (8) shows the comparison in terms of scheduling load

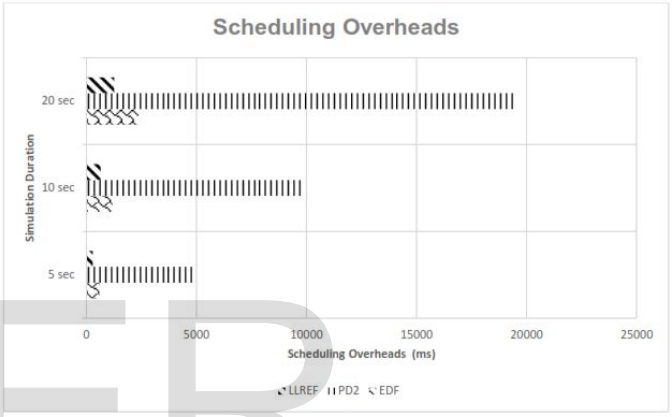

Figure (11) shows the comparison in terms of scheduling load

4.6 The third Scenario:

In this scenario, the psychological steps in the first scenario were followed but with eight nuclei. Figures (9), (10) and (11) show the comparison between the three scheduling algorithms in terms of processor load [22], the burden of context switching operations, when working on eight nuclei.

Comparison of the performance of random task scheduler algorithms in real time systems [21].

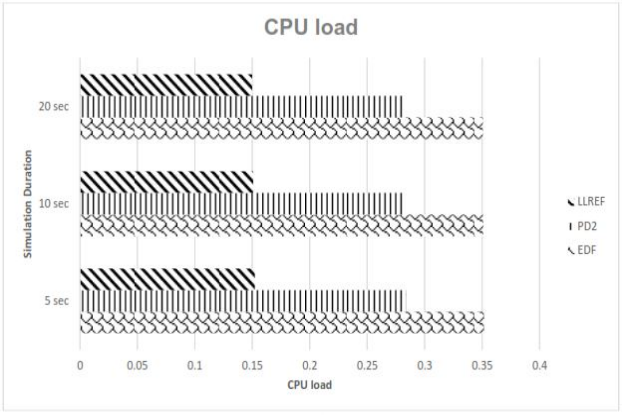

Figure (9) shows the comparison in terms of processor load

5 Conclusions and recommendations:

5.1 Conclusions:

We conclude by looking at the results in the three scenarios studied:

1 The LLREF algorithm gives us better performance than the rest of the algorithms in terms of processor load. The average load rate of each kernel decreases with the increase in the number of nuclei significantly compared to the decrease in the rest of the algorithms.

2 Process load from context switching is small in the LLREF algorithm compared to other algorithms, while the burden is greater in the PD2 algorithm because this algorithm is restricted to time periods unlike the LLREF algorithm in event-based.

3 The load from scheduling increases in the LLREF algorithm slightly with the increase in the simulation time, while this increase is more noticeable in the rest of the algorithms due to the large number of times the scheduler calls to make the scheduling decision in the other algorithms.

4 The LLREF algorithm has better characteristics than other algorithms when compared with the three parameters mentioned above due to the dynamic

performance of this algorithm with the various task types as well as the top of events that are triggered when scheduling a selected random set of tasks compared to other algorithms.

5.2 Recommendations:

According to the results of the research and according to the scenarios studied, the study can be expanded by increasing the number of studied nuclei or by increasing simulation times. The algorithm that achieved the best results in this study can be compared and studied with more algorithms not studied in order to teach more about performance this algorithm and work to meet the defects that may arise from the experimental study, if any.

ACKNOWLEDGMENT

For all those who contributed to the success in ascending
The level of algorithms and enable access to the real time
And thanks to the Iraqi University / Faculty of Engineering
Thanks to its software development environment.

REFERENCES

[1] AMJAD MAHMOOD SALMAN A. KHAN Energy-Aware Real-Time Task Scheduling in Multiprocessor Systems Using a Hybrid Genetic Algorithm [Article] // Electronics,. - May 19, 2017. - Volume 6. - 2.

[2] Anjaria K., & Mishra, A. Thread scheduling using ant colony optimization: An intelligent scheduling approach towards minimal information leakage [Article] // leakage. Karbala International Journal of Modern Science. - 2017. - pp. 241-258.

[3] ANKUR J. Multishare Task Scheduling Algorithm For Real Time Microcontroller Based Application [Article] // Mechatronics and Applications: An International Journal (MECHATROJ). - 2015. - 1 : Vol. 1.

[4] Baek H., Lee, J., Lee, J., Kim, P., & Kang, B. B. Real-Time Scheduling for Preventing Information Leakage with Preemption Overheads [Article] // Advances in Electrical and Computer Engineering. - 2017. - pp. 123-133.

[5] CHARU R., & MANJUGODARA, M Real Time System Scheduling Algorithms & Fault Tolerance [Article] // International Journal of Advanced Research in Computer and Communication Engineering. - 2015. - vol. 4. - Vol. Issue 7.

[6] Dan McNulty Lena Olson, Markus Peloquin A Comparison of Scheduling Algorithms for Multiprocessors [Article] // University of Maryland. - December 13, 2010. - p. 17.

[7] GIRISH S. THAKARE, PRASHANT D. R. and DESHMUKH R. Performance Analysis of Real Time Task Scheduling Algorithm [Article] // International Journal of Innovative Research in Computer and Communication Engineering IJIRCCE. - 2017. - pp. 11866-11869.

[8] GOKULSIDARTHTHIRUNAVUKKARASU* RAGIL KRISHNA Scheduling Algorithm for Real-Time Embedded Control Systems Using Arduino Board [Conference] // KnE Engineering | The International Conference on Design and Technology. - Australia : Deakin University, School of Engineering, Waurn ponds, Australia, 2017. - pp. 258-266.

[9] I. Xu ،M. ،Phan ،LTX ،Sokolsky ،O. ،Xi ،S. ،Lu ،C. ،Gill ،C. ،& Lee ، Cache-aware compositional analysis of real-time multicore virtualization platforms [Article] // Real-Time Systems. - November 2015. - pp. pp 675–723.

[10] JAIN Ankur and GODFREY W. Wilfred. Multishape Task Scheduling Algorithm For Real Time MicroController Based Application [Article] // Mechatronics and Applications: An International Journal (MECHATROJ). - 2015.

[11] Lakshmanan K. S. Scheduling and Synchronization for Multi-core Real-time Systems [Report]. - [s.l.] : Carnegie Mellon University Pittsburgh, PA, 2011. - Doctoral dissertation,.

[12] Levin G., Funk, S., Sadowski, C., Pye, I., & Brandt, S. DP-FAIR: A simple model for understanding optimal multiprocessor scheduling [Conference] // In 2010 22nd Euromicro Conference on Real-Time Systems .. - [s.l.] : IEEE, 2010. - pp. pp. 3-13.

[13] LINLINTANGA KAIQIANG MA, ZUOHUA LI A New Scheduling Algorithm Based on Ant Colony Algorithm and Cloud Load Balancing [Article] // Harbin Institute of Technology Shenzhen Graduate School Shenzhen, China. - 2016.

[14] PERONAGLIO Fernanda F., et al. Modeling Real-Time Schedulers For Use In Simulations Through A Graphical Interface [Conference] // ANSS '17 Proceedings of the 50th Annual Simulation Symposium. - Virginia : [s.n.], 2017.

[15] Pradhan S. R., Sharma, S., Konar, D., & Sharma, K. A comparative study on dynamic scheduling of real-time tasks in multiprocessor system using genetic algorithms [Article] // International Journal of Computer Applications. - 2015. - Volume 120. - No.20.

[16] Qamhieh M. Scheduling of parallel real-time DAG tasks on multiprocessor systems [Report] : Doctoral dissertation. - Paris : Université Paris-Est, 2015.

[17] RADHAKRISHNA NAIK el Periodic and Aperiodic Real -Time Task Scheduling Algorithms Simulator [Article] // International Journal of Pure and Applied Mathematics. - 2017. - pp. 2681-2687.

[18] Rochange Christine. Parallel real-time tasks, as viewed by WCET analysis and task scheduling approaches [Conference] // International conference proceedings. - [s.l.] : Institut de Recherche en Informatique de Toulouse - IRIT (Toulouse, France) - TRACES, 2016. - pp. 1-11.

[19] Schoeberl M., Pezzarossa, L., & Sparsø, J. A multicore processor for time-critical applications [Article] // IEEE Design & Test. - 2018. - 35. - 2. - pp. 38-47.

[20] SHINDE Vijayshree and BIDAY Seema C. Comparison of Real Time Task Scheduling Algorithms [Article] // International Journal of Computer Applications. - 2017. - pp. 37-41.

[21] Strong R., Mudigonda, J., Mogul, J. C., Binkert, N., & Tullsen, D. Fast Switching of Threads between Cores [Article] // ACM SIGOPS Operating Systems Review. - 2009. - pp. 35-45.

[22] TRAN Hai Nam Cache Memory Aware Priority Assignment and Scheduling Simulation of Real-Time Embedded Systems [Report] : PhD Thesis. - [s.l.] : Brest., 2017.

[23] Xi S., Xu, M., Lu, C., Phan, L. T., Gill, C., Sokolsky, O., & Lee, I. Real-time multi-core virtual machine scheduling in xen [Conference] // In 2014 International Conference on Embedded Software (EMSOFT) . - [s.l.] : IEEE., 2014. - pp. pp. 1-10.