Static Task Allocation Algorithms in Mesh Networks: An Experimentation System and Analysis of Properties

Piotr Franz, Leszek Koszalka
Dept. of Systems and Computer Networks
Wroclaw University of Technology
Wroclaw, Poland
e-mail: leszek.koszalka@pwr.wroc.pl

Iwona Pozniak-Koszalka, Andrzej Kasprzak
Dept. of Systems and Computer Networks
Wroclaw University of Technology
Wroclaw, Poland
e-mail: iwona.pozniak-koszalka@pwr.wroc.pl

***Abstract*—The paper concerns the static task allocation problem in mesh structured system. Three allocation algorithms have been evaluated, including well-known First Fit and Stack Based Algorithm, and newly created by authors the Current Job Based First Fit algorithm. The evaluation of their properties and a comparison of their efficiencies have been done on the basis of simulation experiments. The reported investigations have been made with a designed experimentation system coded in C# language with use of .NET Framework for Windows platform. The discussion of results confirms that the created algorithm seems to be promising.**

Keywords - mesh structure; task allocation algorithm; experimentation system

I. INTRODUCTION

Nowadays, modern computer systems are often created by connecting many processing units into one big structure, in order to solve complex problem more efficient. The performance of such structures depends not only on computing power of single processing units, but also on efficiency of algorithms, which are responsible for allocating tasks in structure and those which are responsible to pick certain tasks from queue of ready for execution tasks. Problems of scheduling (task selection) and allocation are important in terms of reducing cost of computing (saving both time and resources) [1].

In the field of solving allocation problem with efficient algorithm still new ideas are proposed on basis on such approaches as Best Fit or Adaptive Scan or First Fit (see, e.g., [2], [3]) as well as algorithms based on evolutionary concepts (see, e.g., [4]).

The aim of this paper is to examine the three implemented allocation algorithms. The two well-known algorithms, FF (First Fit) algorithm and SBA (Stack Based Algorithm) [3], [5], are considered. We designed the third one, called CJB FF (Current Job Based First Fit), which was initially presented in [6].

The static allocation problem [7] considered in this paper, assumes the two-dimensional mesh topology with closed queue of ready tasks (during allocation process no new tasks are added to the queue/system). We assume that tasks from the queue may be picked for allocation using FIFO or SJF scheme [1].

For the purposes of this paper, the experimentation system was designed and implemented. The system allows multi-aspect comparison of the considered algorithms.

The rest of the paper is organized as follows: Section II contains the used nomenclature. In Section III, the three allocation algorithms are briefly described. Section IV contains description of the experimentation system. In Section V, results of investigations are presented and the obtained results of two complex experiments are discussed. Finally, in Section VI, the concluding remarks are stated.

II. PROBLEM STATEMENT

In order to formulate the task allocation problem considered in this paper, the basic definitions and ideas need to be described.

Mesh is a set of nodes (processors) connected in orderly fashion. The typical, full mesh *M (w, h)* is a rectangular two-dimensional matrix of sizes *w* and *h*, where *w* stands for width and *h* stands for height.

Nodes in a mesh are marked as *(i, j)*, where *i* stands for a column and *j* for a row in mesh structure.

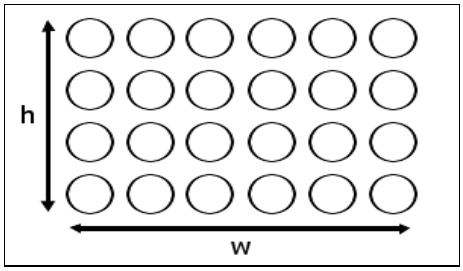

Figure 1. An example of the MESH structure M(6, 4).

Submesh S_M *(i, j, w, h)* is a rectangular set of (*w* x *h) nodes* that belong to a mesh *M (w, h)*. The node *(i, j)* is the foothold of submesh S_M in mesh *M*.

Free submesh is a submesh in which every node is free, i.e. it is not occupied with previously allocated task.

Busy submesh is a submesh in which at least one node is already assigned to execute a task.

Task J (w, h, t) is a rectangular form with known sizes *w* and *h* and execution time *t*. The tasks wait in a queue to be allocated within a mesh. The queue can be a simple FIFO structure or can be sorted (ascending or descending due to

 ISBN: 978-1-61208-202-8

execution time of needed nodes number). To allocate each task, the free sub-mesh with a specified size is needed.

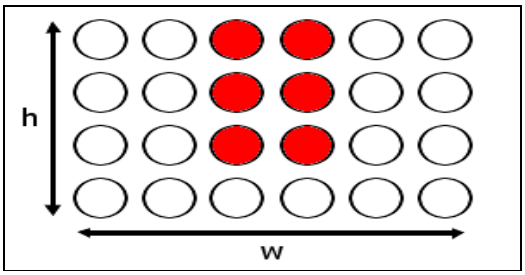

Figure 2. An example: MESH M(6, 4) and a submesh $S_M(3, 1, 2, 3)$.

Expected relative task's width p_w is a ratio of expected task width to mesh size.

Expected relative task's height p_h, similar to p_w, is a ratio of expected task width to mesh size.

Expected relative task's size p is a ratio of expected task size to mesh size (when expected values of task width and task height are equal, then $p = p_w = p_h$).

Allocation problem consists in picking and allocating on a mesh all queued tasks in a way that gives the best results in respect to the introduced quality indicators of allocation efficiency.

Quality indicators. In this paper, the following indicators of efficiency (the indices of performance) are introduced and considered:

The average allocation time t_A (1) needed for algorithm to allocate the task, measured in real time units.

$$t_A = \frac{1}{n}\sum t_{alloc}(i) \quad (1)$$

where: t_{alloc} (i) – time needed to allocate i-th task, n – total number of task in the system.

The total time T_A. The time needed for computing all tasks, measured in 'abstract' time units (so called mesh ticks). One tick passes when allocation algorithm is not able to allocate new task due to lack of free submeshes.

The average latency L_A (2). This is the average time which task needs to wait in a queue until being allocated.

$$L_A = \frac{1}{n}\sum L_i \quad (2)$$

where: L_i – latency of i-th task, n – total number of tasks in the system.

The fragmentation f_A. This is the ratio (3) of the total number of free nodes to the total number of nodes in mesh during algorithm's work (excluding the biggest free submesh).

$$f_A = \frac{w \cdot h - P - \sum_i^n w_i \cdot h_i}{w \cdot h - P} \quad (3)$$

where: w and h – sizes of mesh, P – number of nodes in the biggest free submesh, w_i and h_i – sizes of i-th task.

III. ALGORITHMS

A. First Fit Algorithm (FF)

The First Fit algorithm, is described in details in [2]. The algorithm was implemented as follows:

Step 1. Start searching a given mesh from the node (0, 0) for every single task.
Step 2. Search nodes row by row until free one is found.
Step 3. Check whether a free submesh (containing found free node as a foothold) matching a given task size may be found. If not, go to Step 2.
Step 4. Allocate the task. The matching free submesh becomes busy.
Step 5. End algorithm.

B. Stack Based Algorithm (SBA)

The detailed description of this algorithm can be found in [3]. The main idea of this algorithm consists in finding a base submesh for task, reducing the search space and avoiding unnecessary searches. The algorithm works as follows:

Step 1. For a given task create prohibited area (task if allocated in this area would stick out of mesh).
Step 2. Create coverage areas (respectively if task is going to be allocated in those areas, it will overlap on a busy submesh).
Step 3. Create base areas by spatial subtraction of prohibited and coverage area.
Step 4. Check if exists base area, in which task can be allocated. If yes allocate the task and end algorithm.
Step 5. Rotate the task by 90 degrees and go to Step 1.

C. Current Job Based First Fit Algorithm (CJBFF)

The created algorithm may be treated an improvement of First Fit algorithm. The main idea is to speed up the process of searching free nodes in the mesh structure by omitting already busy nodes belonging to discovered task. The algorithm works as follows:

Step 1. For a given task create a prohibited area (task if allocated in this area would stick out of mesh). Consider only nodes non-belonging to this area.
Step 2.. Start from the node (0, 0).
Step 3. Check whether the node is busy**.** If yes, go to Step 7.
Step 4. Check whether a free submesh (containing found free node as a foothold) matching a given task size may be found. If not, go to Step 7.
Step 5. Allocate the task. The matching free submesh becomes busy.
Step 6. End algorithm**.**
Step 7. Move to the node, next to the last busy node of the encountered task (in the same row). If the task's right edge adjacent to the mesh edge, then move to the next row. Go to Step 3.

 ISBN: 978-1-61208-202-8

IV. EXPERIMENTATION SYSTEM

In order to make simulation of the performance of the considered algorithms, an experimentation system was designed and implemented. The core of the system is simulator with block-scheme shown in Fig. 3.

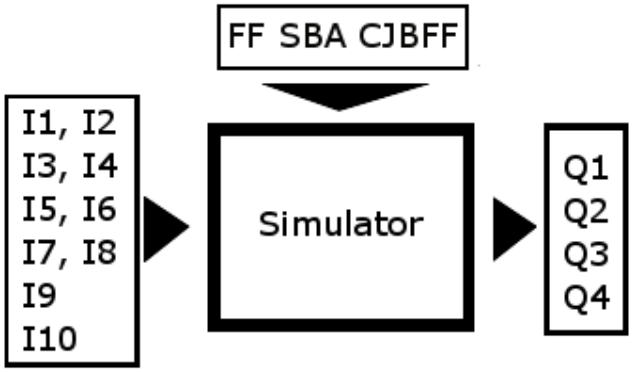

Figure 3. Model of the simulator.

Input parameters of the simulator are:

- I1, I2 – width and height of mesh structure,
- I3, I4 – minimum and maximum width of tasks,
- I5, I6 – minimum and maximum height of a task,
- I7, I8 – minimum and maximum time of a task,
- I9 – number of tasks,
- I10 – sorting type.

Output parameters of the simulator are:

- Q1 – average allocation time,
- Q2 – total computing time,
- Q3 – average latency,
- Q4 – fragmentation.

The system has been implemented using .NET Framework with C# language (it is working well on MS Windows platform with .NET packages). The system possesses the implemented GUI (shown in Fig. 4).

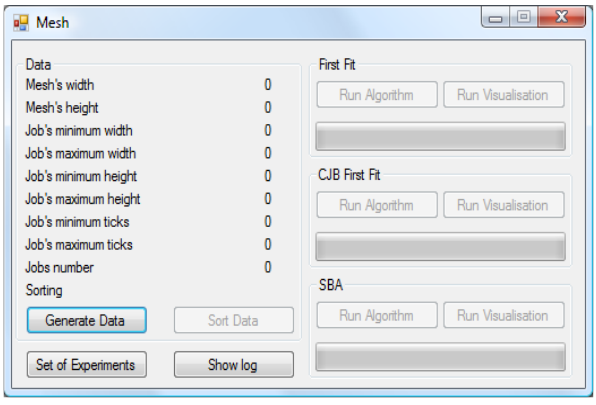

Figure 4. Main window of simulator.

For convenience, the system has implemented function of automatic repetition of the experiment (certain amount of times) for each algorithm with the same input parameters (shown in Fig. 5).

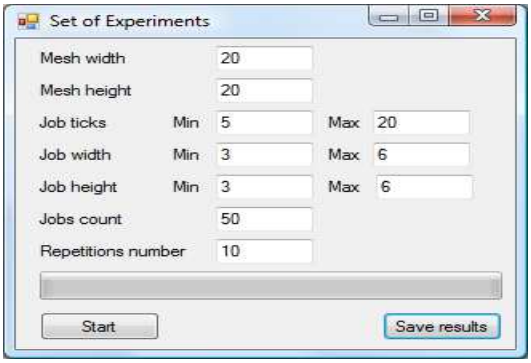

Figure 5. Experiment design window.

V. INVESTIGATION

The aim of the investigations was to compare efficiency of FF, SBA and CBJFF in the same environment. Three efficiency measures were taken into consideration:

t_A – the average allocation time (1),
L_A – the average latency (2),
f_A – the fragmentation (3).

Furthermore, in each experiment the impact of queue sorting on the received latency was examined.

A. *Experiment 1. Increasing number of tasks*

In the first experiment, the set of tasks (queue) was changed in series of experiments - increasing significantly with slightly growing meshes. Experiment design (combination of input values) is shown in Table 1.

TABLE I. INPUTS IN EXPERIMENT 1

Number of Tasks	*Relative Task's Size [%]*	*Mesh width*	*Mesh height*
60	22.5	20	20
140	15.0	30	30
240	11.3	40	40
380	9.0	50	50
540	7.5	60	60
730	6.4	70	70
840	5.6	80	80

Other inputs were taken as follows:

- min – max width of task: 3-6,
- min – max height of task: 3-6,
- min – max execution time of task: 5-20,
- sorting: unsorted, ascending, and descending.

The obtained results are shown in Figs. 6-8.

 ISBN: 978-1-61208-202-8

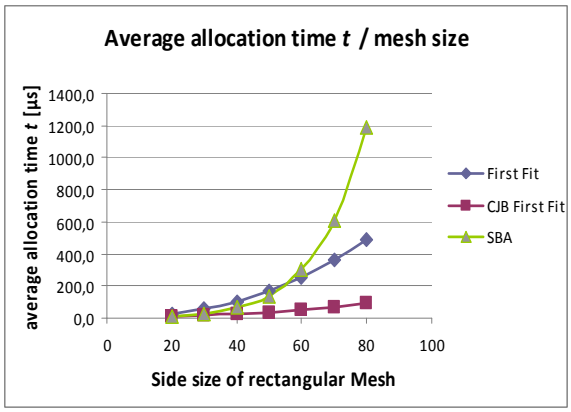

Figure 6. The average allocation time - Experiment 1.

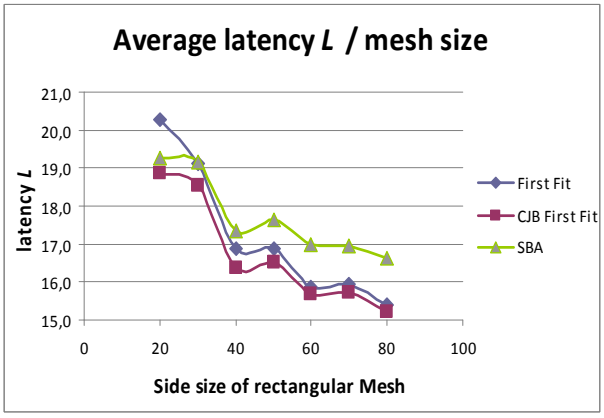

Figure 7. The average latency - Experiment 1.

The created CJBFF was characterized by the best allocation time, significantly lower than the other compared algorithms (Fig. 6). What is more it guaranteed the lowest latency (inversely proportional to mesh size); however for big mesh structures the difference between the CJBFF and FF starts to fade (Fig. 7). The obtained low latencies were possibly the result of low fragmentation maintained by CJBFF algorithm, especially in comparison to SBA (Fig. 8).

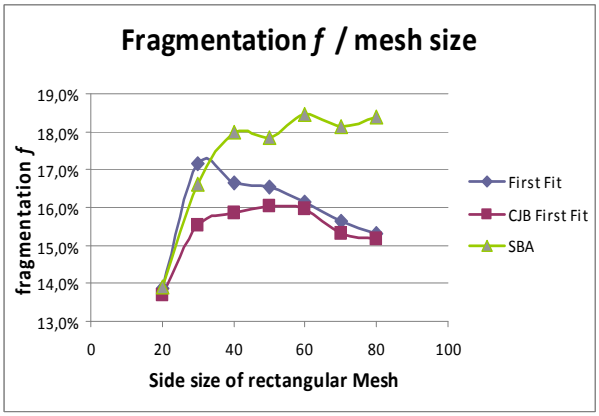

Figure 8. The fragmentation - Experiment 1.

The impact of the chosen queue's sorting type on average latency (in CJBFF) is shown in Fig. 9.

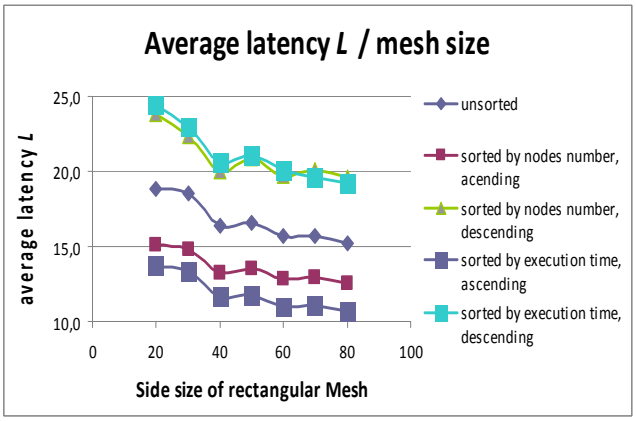

Figure 9. Average latency depending on queue sorting - Experiment 1.

For sorting the tasks in queue by execution time in ascending way, over 25% decrease of latency was obtained comparing to the case when no sorting was used. For descending sorting a remarkable increase of latency was noticed.

B. Experiment 2. Increasing mesh size.

In the second complex experiment the mesh size and the task generation parameters were chosen in such a way that the expected relative task's size *p* was always constant and equal 15% for increasing mesh size. Experiment design (combination of input values) is shown in Tab. II.

TABLE II. INPUT S IN EXPERIMENT 2

Task width	*Task height*	*Mesh width*	*Mesh height*
2	5	20	20
3	7	30	30
3	9	40	40
4	12	50	50
5	14	60	60
6	16	70	70
6	18	80	80

Other inputs were as follows:

- number of tasks: 134,
- min – max execution time of task: 5-20,
- sorting: unsorted.

The obtained results are shown in Figs. 10-12.

It may be observed that, in this experiment, the CJBFF was not the fastest among the considered allocation algorithms. In this case the SBA algorithm was slightly faster for larger mesh structures (see Fig. 10). However, once again the created algorithm proved to guarantee the smallest

 ISBN: 978-1-61208-202-8

latency from all tested algorithms, as it can be observed in Fig. 11.

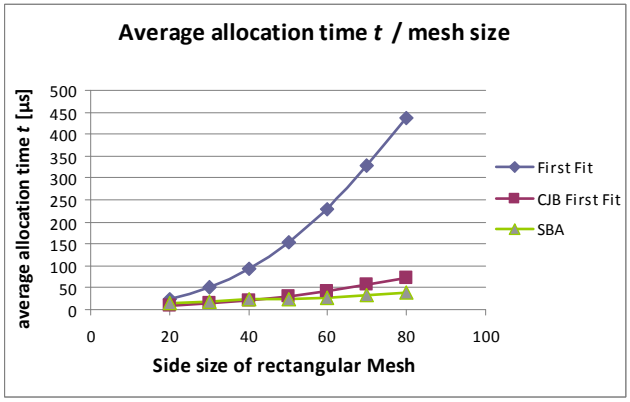

Figure 10. The average allocation time - Experiment 2.

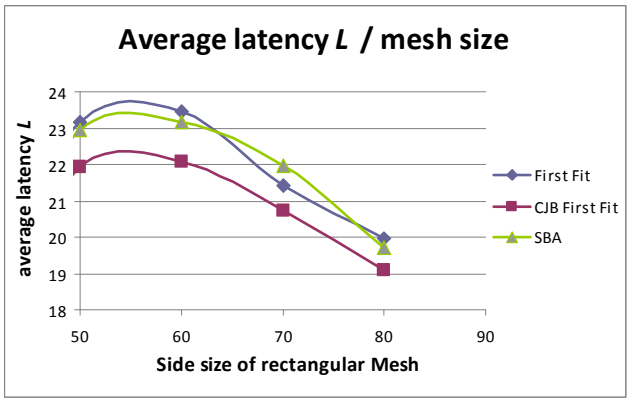

Figure 11. The average latency - Experiment 2.

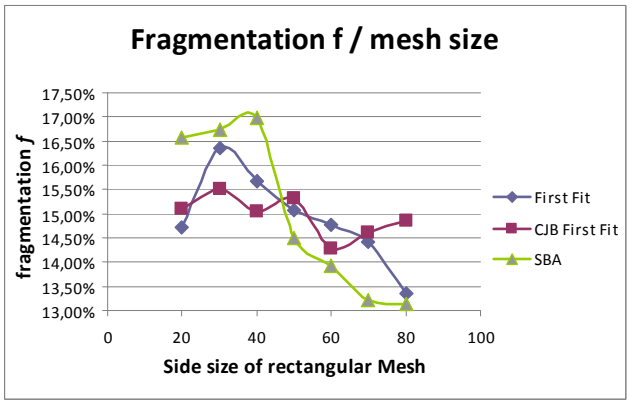

Figure 12. The fragmentation - Experiment 2.

Considering the fragmentation (Fig. 12) it can be seen that the CJBFF algorithm performed as the weakest algorithm; however, only for large meshes. Moreover, it may be observed that the variance of results obtained by all algorithms is rather small and it is not larger than 4%.

The impact of the chosen queue's sorting type (in CJBFF) on average latency is shown in Fig 13.

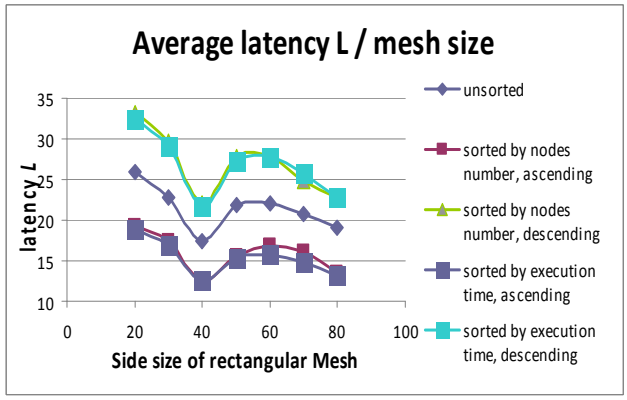

Figure 13. Average latency depending on queue sorting - Experiment 2.

Again, the best results were obtained when used sorting by execution time in ascending order and the worst when sorting in descending order.

VI. CONCLUSION AND FUTURE WORK

The analysis of the results of complex experiments confirms that the designed and implemented CJBFF allocation algorithm is easy to implement and fast in many cases. This algorithm can be recommended to use by designers of multi-processor systems with mesh structures [8], for which the most important factor is the latency of newly added tasks.

Moreover, a big advantage of CJBFF is that with increasing size of a mesh, the time needed for task allocation increases only slightly when comparing to FF and SBA. However, for larger mesh structures the CBJFF has the tendency to fragment the mesh in bigger scale than two other considered algorithms.

To additionally decrease of the latency of tasks (which means improving the allocation process) it may be desirable to apply sorting of task's queue. It is worth to be noticed that ascending sorting by execution times resulted even in a 20% decrease of latency, when comparing to results for unsorted queues.

The further development of the presented in this paper experimentation system will focus on implementing other allocation algorithms, e.g., algorithms based on evolutionary ideas [5].

Moreover, we plan preparing new modules of the system to ensure designing multistage experiments [7] in automatic way and store the results of experiments in problem-oriented data base.

ACKNOWLEDGMENT

We would like to thanks Mr. M. Halaczkiewicz, student of Electronics Faculty, Wroclaw University of Technology for remarkable help in preparing the programmed modules in the experimentation system.

 ISBN: 978-1-61208-202-8

REFERENCES

[1] A. S. Tanenbaum, Modern Operating Systems, 2nd edition, Prentice Hall, 2001.

[2] Y. Zhu, "Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers", J. Parallel & Distr. Computing, vol. 16, 1992, pp. 328-337.

[3] B.S. Yoo and C. Das, "A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicomputers", IEEE Transactions on Computers, vol 51, No. 1, 2002.

[4] W. Kmiecik, M. Wójcikowski, L.,Koszałka, and A. Kasprzak, " Task Allocation in Mesh Connected Processors with Local Seach Meta-heuristic Algorithms", Lecture Notes in Artificial Intelligence , vol. 5559, Springer, 2010, pp. 215-224.

[5] L. Koszalka, M. Kubiak, and I. Pozniak-Koszalka, "Allocation Algorithm for Mesh-Structured Networks", Proc. of 5th ICN, IEEE Comp. Society Press, 2006, pp. 24-29.

[6] M. Halaczkiewicz, "Implementation of Static Task Allocation Algorithms in Mesh Networks", M.Sc. project, Faculty of Electronics, Wroclaw University of Technology, 2009 /in Polish/.

[7] L. Koszalka, D. Lisowski, and I. Pozniak-Koszalka, "Comparison of Allocation Algorithms for Mesh- Networks with Multistage Experiments", Lecture Notes in Computer Science, vol. 3984, Springer, 2006, pp. 58-67.

[8] D. Zydek, H. Selvaraj, L. Koszalka, and I. Pozniak-Koszalka, "Evaluation scheme for NoC-based CMP with integrated processor management system", International Journal of Electronics and Telecommunications, vol. 56, no. 2, 2010, pp. 157-167.

 ISBN: 978-1-61208-202-8