A Cooperative, Accurate Solving Framework for Optimal Allocation, Scheduling and Frequency Selection on Energy-Efficient MPSoCs

Martino Ruggiero[†], Pari Gioia[†], Guerri Alessio[†],
Luca Benini[†], Milano Michela[†], Davide Bertozzi[⋆], Alexandru Andrei[‡]
[†] DEIS, University of Bologna, Italy. [⋆] ENDIF, University of Ferrara, Italy.
[‡] ESLAB, Linkoeping University, Sweden.
{ mruggiero-gpari-aguerri-lbenini-mmilano@deis.unibo.it, dbertozzi@ing.unife.it, alean@ida.liu.se }

Abstract

Most problems addressed by the software optimization flow for multi-processor systems-on-chip (MPSoCs) are NP-complete, and have been traditionally tackled by means of heuristics and high-level approximations. Complete approaches have been effectively deployed only under unrealistic simplifying assumptions. We propose a novel methodology to formulate and solve to optimality the allocation, scheduling and discrete voltage selection problem for variable voltage/frequency MPSoCs, minimizing the system energy dissipation and the overhead for frequency switching. We integrate the optimization and validation steps to increase the accuracy of cost models and the confidence in quality of results. Two demonstrators are used to show the viability of the proposed methodology.

1. Introduction

Multi-Processor Systems-on-Chip (MPSoCs) represent today the main trend for future architectural designs, since they are able to provide scalable computation horsepower while still retaining the flexibility to support different job mixes. Unfortunately, as the complexity of MPSoCs evolves toward ultra-large scale integration, design technology for MPSoCs is hampered by limited scalability and composability. Task mapping problem or compilation subproblems are combinatorial optimisation problems [18] and have been shown to be NP-complete.

Traditionally, the three main approaches followed by the system design community when facing a combinatorial optimization problem are: (i) Modelling and solving the problem with Integer Linear Programming (ILP). Unfortunately, scheduling problems are not well tackled by ILP approaches. In addition, pure ILP formulations are suitable only for very small problem instances. (ii) Deployment of heuristic methods [20] to provide good (even if not optimal) solutions. However, heuristic algorithms still impose significant computational requirements without guarantees on the optimality of final solutions. (iii) Moving from highly simplifed, abstract modelling assumptions and problem instances to make them more tractable. As the complexity of MPSoCs raises, second order effects are going to impair the quality of derived solutions, lowering the confidence level on constraint satisfaction and on the value of the objective function.

Many optimisation problems can be decomposed into well known, structured and widely studied sub-problems. It is widely acknowledged that exploiting the structure of these problems improves the performances of the corresponding algorithms. In general, merging different algorithmic aspects leads to an efficient solving process and may determine significant performance speed ups in finding an optimal solution (see [22]). As a result, many practical problem configurations, traditionally tackled by means of approximate methods and over-symplifying assumptions, become now tractable by complete approaches providing the optimal solution in reasonable time. On the other hand, the critical issue of how to make the different algorithms interact in a cooperative solving framework arises.

The objective of this paper is to extend this trend to one of the most important optimization problems in system level design for variable voltage/frequency MPSoCs: power consumption.

The major challenge that we face in this paper is to devise a cooperative solving framework where the allocation, the scheduling and the discrete voltage/frequency selection problem models can be suitably accommodated and solving algorithms integrated. By leveraging the principle of logic-based Benders Decomposition [23], we come up with an iterative two-step mapping framework that is proved to converge to the optimal solution. We tackle problem complexity without paying the price of a lack of accuracy in modelling assumptions. In fact, an MPSoC virtual platform was deployed to refine the theoretical framework and prove the accuracy of the solutions provided by the optimizer.

Our methodology derives static allocation, scheduling and frequency settings, therefore targets applications with design-time predictable behaviour. Signal processing and multimedia applications employing pipelining as workload allocation policy are the most common example of such applications, therefore our optimizer was tuned for energy-efficient mapping of pipelined task graphs on MPSoCs. Finally, we used two demonstrators to prove the applicability of the developed methodology to real-life scenarios.

This paper is structured as follows: we first describe previous work in the field. The target architecture and the virtual platform environment are presented in section 3 while the model of the Dynamic Voltage Scaling Problem is defined in section 4. Discussions on computational efficiency, validation and experimental results follow.

2. Related Work

In the following, we restrict ourselves to off-line voltage/frequency selection techniques, since the presented approach falls into this category.

A number of techniques have been developed for single processor systems. Yao et al. proposed in [12] the first DVS approach which can dynamically change the supply voltage over a continuous range. Ishihara and Yasuura [6] modeled the discrete voltage selection problem using an integer linear programming (ILP) formulation. Xie et al. [11] present an algorithm for calculating the bounds on the power savings achievable through voltage selection. Jejurikar and Gupta [8] propose an algorithm that combines voltage scaling and shutdown in order to minimize dynamic and leakage energy in single.

Andrei et al. [2] proposed an approach that solves optimally the voltage scaling problem for multi-processor systems with imposed

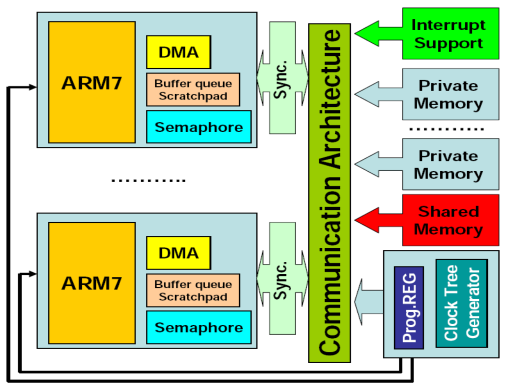

Figure 1. Distributed MPSoC architecture.

time constraints. The continuous voltage scaling is solved using convex nonlinear programming with polynomial time complexity, while the discrete problem is proved strongly NP hard and is formulated as mixed integer linear programming (MILP).

The previously mentioned approaches, assume that the mapping and scheduling are given. However, the achievable energy savings of dynamic voltage scaling are greatly influenced by the mapping and the scheduling of the tasks on the target processors.

Task mapping and scheduling are known NP complete problems [3] that have been previously addessed, without and with the objective of minimizing the energy. Both heuristic [9], [5] and exact solutions [1] have been proposed.

Assuming the mapping of the tasks on the processors is given as input, the authors from [4] present a scheduling technique that maximizes the available slack, which is then used to reduce the energy via voltage scaling. Schmitz et al. [9] present a heuristic approach for mapping, scheduling and voltage scaling on multiprocessor architectures.

A leakage-aware approach for combined dynamic voltage selection and adaptive body-biasing has been proposed in [2, 24]. Although we concentrate in this paper on the dynamic power and supply voltage selection, our methodology can handle with minor changes the combined supply and body bias scaling problem with only marginal implications on computational complexity.

The closest approach to our work is the one of Leung et al., [7]. They propose a mixed integer programming formulation for mapping, scheduling and voltage scaling of a given task graph to a target multiprocessor platform. They assume continuous voltages, so the overall result is suboptimal.

3. Target Architecture

The target architecture for our mapping strategy is a general template for a parallel MPSoC architecture. The platform consists of computation tiles, an AMBA AHB-compliant communication architecture (a shared bus) and of a shared memory for inter-tile communication(see Fig. 1). The computation tiles are supposed to be homogeneous and consist of ARM7 cores (including instruction and data caches) and of tightly coupled software-controlled scratch-pad memories.

Messages can be exchanged by tasks through communication queues, which can be allocated at design time either in scratch-pad memory or in remote shared memory, depending on whether tasks are mapped onto the same processor or not. Synchronization between a producer/consumer pair is implemented by means of distributed hardware semaphores. The software support is provided by a real-time operating system called RTEMS and by a set of high-level APIs to support message passing on the underlying hardware architecture [13].

Our virtual platform environment provides power statistics for ARM cores, caches, on-chip memories and AMBA AHB bus, leveraging technology-homogeneous power models for a 0.13 μ m technology provided by STMicroelectronics. When all tasks mapped on a processor core are suspended, then the core enters power save mode, where the power consumption is assumed to be negligible. The virtual platform supports different working frequencies for each processor core [16] (see the synchronization modules, and the programmable register driving the clock tree generator in Fig. 1). The maximum AMBA AHB frequency of 200MHz was kept as the maximum processing core frequency, to which frequency dividers were applied. The scaling factor for the power supply was derived from [17].

4. Dynamic Voltage Scaling Problem - DVSP

We consider a task graph G whose nodes represent a set of T tasks, that are annotated with their deadline dl_t and with the worst case number of clock cycles WCN_t. Arcs represent dependencies among tasks. Each arc is annotated with the amount of data two dependent tasks should exchange, and therefore the number of clock cycles for exchanging (reading and writing) these data WCN_R and WCN_W. Tasks are running on a set of processors P. Each processor can run with M energy/speed modes and has a maximum load constraint dl_p. Each task spends energy both in computing and in communicating. In addition, when the processor switches between two modes it spends time and energy. We have energy overhead E_f for switching from frequency f to any other, and time overhead T_f for switching from frequency f to any other.

The Dynamic Voltage Scaling Problem (DVSP) is the problem of allocating tasks to processors, define the running speed of each task and schedule each of them minimizing the total energy consumed. In order to solve the DVSP to optimality without incurring accuracy limitations, we applied the concept behind the logic-based Benders decomposition technique [23] to this new application problem.

We therefore decompose the problem in two parts: the first, called Master Problem, is the allocation of processors and frequencies to tasks and the second, called Subproblem, is the scheduling of tasks given the static allocation and frequency assignments provided by the master. The master problem is tackled by an Integer Programming solver while the subproblem through a Constraint Programming solver. The two solvers interact via no-good and cutting planes generation. The solution of the master is passed to the subproblem in an iterative procedure that is proved to converge to the optimal solution [23].

4.1 The Master Problem model

We model the allocation problem with binary variables X_{ptm} which take value 1 if task t is mapped on the processor p and runs in (energy-speed) mode m, 0 otherwise. Since we also take into account communication, we assume that two tasks consume energy and time for communication only if they are allocated on two different processors. Both the read and write activities are performed at the same speed of the task and use the bus (which instead works at the maximum speed). For modelling this aspect, we introduce in the model two variables $R_{pt_1t_2m}$ and $W_{pt_1t_2m}$ taking value 1 if the task t_1 running on processor p reads (resp. writes) data (at mode m) from (resp. for) a task t_2 not running on p.

Any task can be mapped on only one processor and can run at only one speed, that is:

$$\sum_{p=1}^{P}\sum_{m=1}^{M} X_{ptm} = 1 \ \ \forall t$$

Also, if each task can read data from (resp. write data for) only one task as for instance in a pipelined workload, we have these constraints:

$$\sum_{t_2=1,t_2\neq t_1}^{T}\sum_{p=1}^{P}\sum_{m=1}^{M} R_{pt_1t_2m} \leq 1 \ \ \forall t_1$$

$$\sum_{t_2=1,t_2\neq t_1}^{T}\sum_{p=1}^{P}\sum_{m=1}^{M} W_{pt_1t_2m} \leq 1 \ \ \forall t_1$$

The objective function is to *minimize the energy consumption of the task execution, and of the task communication*:

$$E_{comp} = \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{t=1}^{T} X_{ptm} WCN_t t_{clock_m} P_{tm}$$

$$E_{Read} = \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{t,t1=1}^{T} R_{ptt_1m} WCN_{Rtt_1} t_{clock_m} P_{tm}$$

$E_{Write} = \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{t,t1=1}^{T} W_{ptt_1m} WCN_{Wtt_1} t_{clock_m} P_{tm}$

$OF = E_{comp} + E_{Read} + E_{Write}$

where P_{tm} is the power consumed by task t when running in execution mode m.

The objective function defined up to now depends only on master problem variables. However, switching from one speed to another introduces transition costs, but their value can be computed only at scheduling time. Therefore, we update the objective function with frequency transition (or setup) costs:

$OF_{Master} = OF + \sum_{p=1}^{P} Setup_p$

but we force them to be 0 in the first iteration, while from the second iteration cuts are produced by the subproblem, constraining variables $Setup_p$.

Finally, we impose that on each processor the sum of the time spent for the the computation, plus the time spent for communication (read and write) should be less than or equal to the deadline of the processor, in oder to prevent trivially infeasible solutions:

$T^p_{comp_p} = \sum_{t=1}^{T} \sum_{m=1}^{M} X_{ptm} \frac{WCN_t}{f_m}$

$T^p_{read} = \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{t=1}^{T} R_{ptt_1m} \frac{WCN_{Rtt_1}}{f_m}$

$T^p_{write} = \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{t=1}^{T} W_{ptt_1m} \frac{WCN_{Wtt_1}}{f_m}$

$T^p_{comp} + T^p_{read} + T^p_{write} \leq dl_p \ \ \forall p$

In the same way task deadlines can be captured, which are the same formulas but the sums are computed for each task.

4.2 The Sub-Problem model

For the scheduling part each task t has an associated variable representing its starting time $Start_i$. The duration is fixed since the frequency is decided, i.e., $duration_i = WCN_i/f_i$. In addition, if two communicating tasks t_i and t_j are allocated on two different processors, we should introduce two additional activities (one for writing data on the shared memory and one for reading data from the shared memory). We model the starting time of these activities $StartWrite_{ij}$ and $StartRead_{ij}$. These activities are carried on at the same frequency of the corresponding task. If t_i writes and t_j reads data , the writing activity is performed at the same frequency of t_i and its duration $dWrite_{ij}$ depends on the frequency and on the amount of data t_i writes, i.e., WCN_{Wij}/f_i. Analogously, the reading activity is performed at the same frequency of t_j and its duration $dRead_{ij}$ depends on the frequency and on the amount of data t_j reads, i.e., $WCN_{R_{ij}}/f_j$. Clearly the read and write activities are linked to the corresponding task:

$Start_i + duration_i \leq StartWrite_{ij} \ \ \forall j$

$StartRead_{ij} + dRead_{ij} = Start_j \ \ \forall i$

In the subproblem, we model precedence constraints in the following way: if tasks t_i should precede task t_j and they run on the same processor at the same frequency the precedence constraint is simply:

$Start_i + Duration_i \leq Start_j$

If instead the two tasks run on the same processor at different speed, we should add the time T_i for switching between the two frequencies.

$Start_i + Duration_i + T_i \leq Start_j$

If the two tasks run on different processors and should communicate we should add the time for communicating.

$Start_i + Duration_i + dWrite_{ij} + dRead_{ij} \leq Start_j$

Resources are modelled as follows. We have a unary resource constraint for each processor, modelled through a cumulative constraint having as parameters a list of all tasks sharing the same resource p, their durations, their resource consumption (which is a list of 1) and the capacity of the processor which is 1

$cumulative(TaskList_p, DurationList_p, [1], 1) \ \ \forall p$

We model the bus through an additive model we have already validated in [15]. We have an activity on the bus each time a task writes or reads data to and from the shared memory.

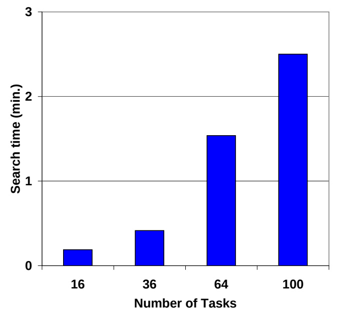

Figure 2. Search time for an increasing number of tasks.

The objective function we want to minimize in the scheduling problem is the setup energy, i.e., the energy spent for frequency switchings. For this purpose we have energy overheads E_i for switching from frequency i to any other frequency.

Once the subproblem has been solved, we generate Benders Cuts. The cuts are of two types: (i) if there is no feasible schedule given an allocation, we have to compute a no-good on variables X_{ptm} avoiding the same allocation to be found again.
(ii) if a feasible and optimal schedule exists, we cannot simply stop the iteration since the master objective function depends also on subproblem variables. Therefore, we have to produce cuts saying that the one just computed is the optimal solution unless a better one exist with a different allocation. These cuts produce a lower bound on the setup costs of single processors.

The procedure converges when the master problem produces a solution with the same objective function value of the previous one.

5. Computational efficiency

We tested the computation efficiency of our hybrid approach on a 2GHz Pentium 4 machine with 512 Mb RAM and leveraged state-of-the-art professional solving tools, namely ILOG CPLEX 8.1, ILOG Solver 5.3 and ILOG Scheduler 5.3. We increased the number of tasks and of processors, and noticed that the algorithm scales quite smoothly in both cases. In Fig.2 we plot the search time for an increasing number of tasks. The behavior is similar for increasing number of processors (going from 4 to 10).

We also noticed that the phase transition of the problem happens when the deadline is not too tight to have few solutions (among which is easy to find the optimal one) and not too loose so as the problem is trivially solvable assigning all tasks to the same processor and the lowest speed. In addition, varying the deadline constraints, the best and the worst search time remain within an order of magnitude, so our methodology efficiently faces instances with different density of feasible solutions.

6. Experimental Results

We have deployed a cycle accurate MPSoC simulator [14] to characterize task model parameters for the optimizer and validate the optimizer results. Two types of *validation experiments* were performed, namely (i) comparison of simulated energy and throughput with optimizer-derived values, and (ii) prove of viability of the proposed approach for real-life demonstrators (GSM, JPEG).

6.1 Validation of optimizer solutions

We have deployed the virtual platform to implement the allocations, schedules and frequency assignments generated by the optimizer for 200 problem instances. The results of the validation phase are reported in Fig.3, which shows the distribution of energy deviations. The average difference between measured and predicted energy values is 2.9%, with 1.72 standard deviation. Fig.4 shows the cumulative probability of throughput differences: in this case

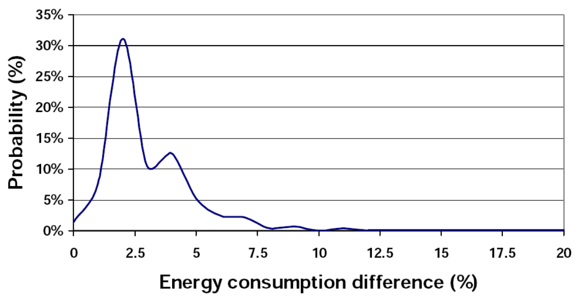

Figure 3. Distribution of Energy Consumption differences.

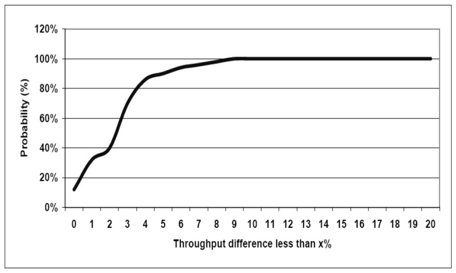

Figure 4. Cumulative probability of throughput deviations.

the average difference between measured and predicted values is 4.7%, with 0.08 standard deviation. This confirms the high level of accuracy achieved by the developed optimization framework in modelling real-life MPSoC systems with the assumed architectural template.

6.2 Demonstrators

The methodology has been applied to a GSM codec parallelized in 6 pipelined tasks. The validation process on the virtual platform showed an accuracy on processor energy by 2% When restricting the real-time requirement, the behaviour of the optimizer can be deduced from Fig.5. When the deadline is loose, all tasks are allocated to one single processor at the minimum frequency. As the deadline gets tighter, the optimizer prefers to employ a second processor and to progressively balance the load, instead of increasing task frequencies. Only under very tight deadlines, the optimizer leverages increased task frequencies to speed-up the system. To the limit, the system works with 1 task on each processor, although not all tasks run at the maximum frequency. In fact, the GSM pipeline turns out to be unbalanced, therefore it would be energy inefficient to run the shorter tasks at maximum speed, and would not even provide performance benefits. The problem becomes infeasible if more stringent deadlines than 710 ns are required.

Our methodology was then applied to a JPEG decoder partitioned in 4 pipelined tasks, and the accuracy on energy estimation was again very high (3.1%). In contrast to GSM, user requirements on a JPEG decoding usually consist of the minimization of the execution time and not of a deadline to be met. However, a performance-energy conflict arises, therefore we derived the Pareto-optimal frontier in Fig.6. The constraint on the execution time on the x-axys has been translated into a constraint on the block decoding time. The curve is not linear since there is a discrete number of voltage-frequency pairs, which makes the problem for the optimizer much more complex.

7. Conclusions

In this paper, we built a cooperative framework to solve the allocation, scheduling and voltage/frequency selection problem to optimality for energy-efficient MPSoCs. The integration of the optimizer with a virtual platform allowed us to prove the accuracy of our methodology.

Deadline (ns)	Number of Processors	# Task Allocated on Core	Task Frequency Divider	Energy Consumption (nJ)
6000	1	1, 1, 1, 1, 1, 1	3, 3, 3, 3, 3, 3	5840
5500	2	2, 1, 1, 1, 1, 1	3, 3, 3, 3, 3, 3	5910
5000	2	1, 1, 1, 1, 1, 2	3, 3, 3, 3, 3, 3	5938
4500	2	1, 1, 1, 1, 2, 2	3, 3, 3, 3, 3, 3	5938
4000	2	1, 1, 1, 2, 2, 2	3, 3, 3, 3, 3, 3	5938
3500	2	1, 1, 1, 2, 2, 2	3, 3, 3, 3, 3, 3	5938
3000	3	1, 2, 2, 3, 3, 3	3, 3, 3, 3, 3, 3	6008
2500	3	1, 1, 2, 2, 3, 3	3, 3, 3, 3, 3, 3	6039
2000	4	1, 2, 3, 3, 4, 4	3, 3, 3, 3, 3, 3	6109
1500	6	1, 2, 3, 4, 5, 6	3, 3, 3, 3, 3, 3	6304
1000	6	1, 2, 3, 4, 5, 6	3, 2, 2, 2, 3, 2	6807
900	6	1, 2, 3, 4, 5, 6	3, 1, 2, 2, 2, 2	9834
750	6	1, 2, 3, 4, 5, 6	2, 1, 2, 2, 2, 2	9934
730	6	1, 2, 3, 4, 5, 6	2, 1, 1, 2, 2, 2	12102
710	6	1, 2, 3, 4, 5, 6	2, 1, 1, 1, 2, 2	14193

Figure 5. Behaviour of the optimizer with varying real-time requirements. Allocation is given as an array indicating the processor ID on which each task is mapped. Similarly, the frequency of each task is expressed in terms of the integer divider of the baseline frequency. Only 3 dividers are used for this example.

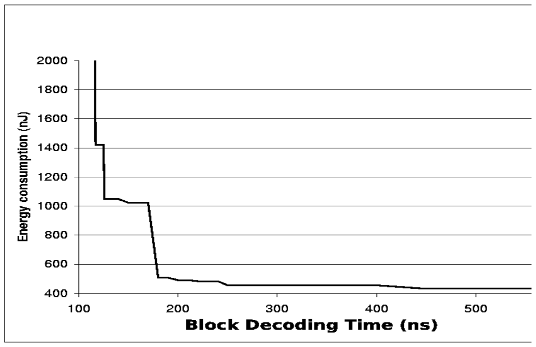

Figure 6. Pareto-optimal frontier in the performance-energy design space

8. REFERENCES

[1] L. Benini, D. Bertozzi, A. Guerri and M. Milano, Allocation and Scheduling for MPSoCs via Decomposition and No-Good Generation. In *"Proc. of International Joint Conference on Artificial Intelligence (IJCAI)"*, August 2005, pages 1517–1518.

[2] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al-Hashimi. Overhead-Conscious Voltage Selection for Dynamic and Leakage Power Reduction of Time-Constraint Systems. In *"Proc. Design, Automation and Test in Europe (DATE)"*, pages 518–523, Feb 2004.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[4] F. Gruian and K. Kuchcinski. LEneS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors. In *"Proc. ASP-DAC'01"*, pages 449–455, Jan 2001.

[5] J. Hu and R. Marculescu. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. In *"Proc. Design, Automation and Test in Europe (DATE)"*, pages 234–239, Feb 2004.

[6] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *"Proc. Int. Symp. Low Power Electronics and Design (ISLPED)"*, pages 197–202, 1998.

[7] L-F. Leung, C-Y. Tsui, and W-H. Ki. Minimizing Energy Consumption of Multiple-Processors-Core Systems with Simultaneous Task Allocation, Scheduling and Voltage Assignment. In *"Proc. of Asia South Pacific Design Automation (ASP-DAC)"*, pages 647–652, Jan 2004.

[8] R. Gupta R. Jejurikar. Dynamic Slack Reclamation with Procrastination Scheduling in Real-Time Embedded Systems. In *Design Automation Conference (DAC)*, Jun 2005.

[9] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles. Iterative Schedule Optimization for Voltage Scalable Distributed Embedded Systems. *ACM Transactions on Embedded Computing Systems*, 3:182–217, 2004.

[10] Nicolas Ventroux, Frédéric Blanc, and Dominique Lavenier. A low complex scheduling algorithm for multi-processor system-on-chip. In *Parallel and Distributed Computing and Networks*, pages 540–545, 2005.

[11] F. Xie, M. Martonosi, and S. Malik. Bounds on power savings using runtime dynamic voltage scaling: an exact algorithmand a linear-time heuristic approximation. In *"Proc. Int. Symp. Low Power Electronics and Design (ISLPED)"*, pages 287–292, Aug 2005.

[12] F. Yao, A. Demers, and S. Shenker. A Scheduling Model for Reduced CPU Energy. In *IEEE Symposium on Foundations of Comp. Science*, pages 374–382, 1995.

[13] F.Poletti and A.Poggiali, Flexible Hardware/Software Support for Message Passing on a Distributed Shared Memory Architecture, In *Proc. of DATE Conf.*, pages 736–741, 2005.

[14] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini and R. Zafalon, Analyzing On-Chip Communication in a MPSoC Environment, In *Proc. of the Design, Automation and Test in Europe Conference and Exhibition 2004*, Paris, France, Feb 16-20, 2004, pp. 752-757

[15] L.Benini, D.Bertozzi, A.Guerri and M.Milano, Allocation and Scheduling for MPSoCs via Decomposition and No-Good Generation., In *Proc. of CP 2005*, pages 107-121.

[16] M.Ruggiero, A.Acquaviva, D.Bertozzi, L.Benini, Application-Specific Power-Aware Workload Allocation for Voltage Scalable MPSoC Platforms, In *Proc. of ICCD 2005*, pages 87-93.

[17] K. J. Nowka et al. A 32-bit PowerPC System-on-a-Chip with support for dynamic voltage scaling and dynamic frequency scaling., In *IEEE JSSC*, Vol. 37, no. 11, pp. 1441–1447, Nov. 2002.

[18] Y.K. Kwok, I. Ahmad. Static Scheduling Algorithms for allocating directed task graphs to multiprocessors. In *ACM Computing Surveys*, vol. 31, no. 4, pp. 406-471, 1999.

[19] Weixiong Zhang. Modeling and Solving a Resource Allocation Problem with Soft Constraint Techniques. In *Technical Report: WUCS-2002-13*, 2002

[20] J. Axelsson. Architecture Synthesis and Partitioning of Real-Time Synthesis: a Comparison of 3 Heuristic Search Strategies. In *Procs. of the 5th Intern. Workshop on Hardware/Software Codesign (CODES/CASHE97)* pp. 161-166, 1997.

[21] P. Eles and Z. Peng and K. Kuchcinski and A. Doboli and P. Pop Scheduling of Conditional Process Graphs for the Synthesis of Embedded Systems. In *Procs. of the conference on Design, automation and test in Europe*, pp. 132-139, 1998.

[22] M. Milano et al. Constraint and Integer Programming Toward a unified methodology, Operations Research/Computer Sciences Interfaces, In *Kluwer Academic Publisher*, 2003.

[23] J. N. Hooker and G. Ottosson Logic-based Benders decomposition, In *Mathematical Programming*, 2003, 33–60

[24] S. Martin and K. Flautner and T. Mudge and D. Blaauw Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads In *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, pp. 721-725, 2002.