Efficient Voltage Scheduling and Power-Aware Co-Design for Real-Time Embedded Systems

ABSTRACT

This paper presents an integrated methodology and a tool for system-level low power/energy co-synthesis of real-time embedded systems. Static voltage scheduling (SVS) is being applied to utilize the inherent slacks in the system. The static voltage schedule is generated based on a global view of all tasks' mapping and their energy profiles. The tool explores the three dimensional design space (performance-power-cost) to find implementations that offer the best trade-off among these design objectives. Unnecessary power dissipation is prevented by refining the allocation/binding in an extra synthesis step. The experimental results show that our approach remarkably improves the efficiency of SVS to reduce power consumption, especially, for designs with stringent design constraints.

General Terms

Design, algorithms.

Keywords

Co-synthesis, scheduling, design-space exploration, power consumption, evolutionary algorithms, dynamic voltage scaling.

1. INTRODUCTION

Power consumption is one of the major challenges that face nearly all types of present and future battery-operated and embedded systems. Reducing the power/energy dissipation makes these systems more competitive and extends the battery life time. At the same time, packaging and cooling expenses are directly related to the power/energy dissipation in the system. Hence, without an integrated methodology that sharply reduces power consumption, mobile electronics will suffer from short operation periods or heavy battery weights.

Surveying the work already done in this area reveals that this problem has been investigated at different abstraction levels. Low level methodologies supported by CAD tools, such as SPICE can barely achieve energy reductions of more than 50%. This is related to the fact that decisions made at these levels have only limited and local effect on the consumed power. Moreover, modifying the design at these levels causes longer design cycle since different design decisions have been already taken at higher levels of abstraction. Therefore, high level tools can reduce the design cycle significantly and can lead to design alternatives that are more efficient from the power/energy point of view.

Typically, tackling power issues at the highest possible abstraction level has the most global effect. So, using integrated and automated co-design methodologies starting at the system level is recommended to achieve drastic power reduction and best system optimization. It is worth noticing that system level methodologies are applied as a preliminary step for low power system optimization that can be combined with low level approaches.

But, automated co-design set up at high abstraction levels needs at least two supporting requirements: Firstly, a specification language that supports automated implementation and verification of functional and temporal behaviour of real-time embedded systems [1]. Secondly, the required information for performing such automated implementation has to be abstracted from low levels and supplied at the intended high level where it can be used by automatic synthesis and optimization tools [2].

The methodology proposed in this paper tackles at the first place the dynamic power consumed in embedded systems. Although, the proposed algorithms are general and can be extended to handle issues related to static power. In general, the dynamic power consumed in a digital device is related to the switched capacitance C, the applied frequency f and the square of the operating voltage V_{dd}; $P \propto C \cdot V_{dd}^2 \cdot f$. Therefore, reducing the supply voltage yields a quadratic energy reduction. Based on this, dynamic voltage scaling (DVS) was suggested for trading performance for power without sacrificing the peak performance of the device. At high levels of abstraction, the voltage level(s) required for executing each task can be statically planned for applications that have predictable computational loads and predetermined limits on computation performance. Considering power profiles of the allocated components when scaling the voltage is a source of extra power/energy reduction. This is related to the fact that the higher the energy consumption of a task the more energy saving it causes once scaling its supply voltage.

The remainder of this paper is organized as follows: Section 2 presents a summary of selected related work in the area of power/energy minimization and design space exploration. Section 3 presents our automated co-design methodology for low power/energy. The experimental results are presented in section 4. We conclude in section 5 and suggest some issues to be handled in future work.

2. RELATED WORK

In recent years, tools have been devised for exploring the design space at high abstraction levels. Thiele et al. have suggested a system level design space exploration methodology for architectures of packet processing devices [3]. An evolutionary-based approach for system level synthesis and design space exploration was suggested in [4]. Slomka et al. have presented a tool for hardware/software co-design of complex embedded systems with real-time constraints, Corsair [5]. The co-synthesis process was based on a three-level tabu search algorithm. The above mentioned approaches did not handle the power problem at all or did not tackle it concretely.

A power estimation framework for hardware/software System-on-Chip (SoC) designs was introduced in [6]. The approach was based on concurrent execution of different simulators for different parts of the system (hardware and software parts). Although, this approach could be fairly accurate it is very slow, especially for large systems when a huge number of design alternatives is available.

Hybrid search strategies (global/local) for power optimization in embedded DVS-enabled multiprocessors were introduced in [7]. The approach used a local optimization based on hill climbing and Monte Carlo search inside a genetic-based global optimization. Although, this approach yields the required voltage levels that minimize the energy per computation period, it is time consuming, especially when applied at high abstraction levels. In addition, the influence of the power profiles of the tasks was not included.

Gruian has introduced two system level low-energy design approaches based on DVS-enabled processors [8]. The first was based on performance-energy tradeoffs whereas the second was based on energy sensitive scheduling and mapping techniques. In this approach, simulated annealing was used for generating task-processor mappings.

An energy conscious scheduling method was introduced in [9]. This methodology assumed a given allocation and tasks-processors assignment (DVS-enabled processors). The energy was minimized by selecting the best combination of supply voltage levels for each task executing on its processor.

A low power co-synthesis tool (LOPOCOS) was suggested in [10]. The objective was to help the designer in identifying an energy-efficient application partitioning for embedded systems implemented as heterogeneous distributed architectures. This approach assumed DVS-enabled architectures. Although it performs better than previously suggested approaches, for applications with stringent delay constraints, it has moderate reduction effect on the consumed power/energy.

Quite recently, power optimized and performance optimized components' types were suggested in [11]. In an automated co-design methodology, the allocation/binding were refined by an extra step to make benefit of these types. The effect of using power optimized components on the overall power consumption was studied. The maximum achieved power reduction was about 20% for the included benchmarks in the study.

Many of the previously introduced approaches dealt with the power problem at high abstraction levels and utilized the power-performance tradeoffs by using DVS-enabled architectures. However, the following issues are not yet solved satisfactorily: 1) The special needs of optimizing the co-synthesis process when applying DVS. 2) For applications with stringent performance constraints, DVS may fail to cause significant power reductions. 3) The combined effect of using different components' types and voltage scaling was not addressed at all.

Our proposed methodology for low power/energy co-design deals with issues mentioned above. Starting at the level of FDTs (formal description techniques), the tool is able to explore the available design space while handling design tradeoffs. It yields power optimized design(s) under pre-defined stringent performance limits with low cost. The voltage schedule is static and based on a global view of energy profiles of tasks and their mappings. The integrated library is enhanced with a set of special features to enable fast design space exploration and to improve the efficiency of SVS. Combining these issues together in one system-level tool leads to drastic power/energy reduction, especially for real-time systems with stringent design constraints.

3. DESIGN FLOW AND DESIGN SPACE EXPLORATION

To be able to handle the complexity of designing large embedded systems with the presence of time constraints, the design process is decomposed in our co-design methodology into four phases: System specification, co-synthesis, implementation synthesis, and evaluation and validation. These steps are described bellow before explaining our voltage scheduling methodology.

3.1. Design Phases

The overall automated co-design methodology consists of the following steps:

3.1.1. System Specification

This phase transforms the informal specifications into formal specifications. We use the SDL/MSC which is one of the prominent and the successfully applied techniques in telecommunication industry [12]. SDL (specification and description language) is used to describe the functional specification. MSC (message sequence chart) is extended to describe timing requirements and other non-functional aspects. All (Performance) MSC requirements are automatically transformed to SDL to yield an integrated co-design specification in SDL*.

3.1.2. Co-Synthesis

An internal system model, a problem graph (PG), and an architecture graph (AG) are automatically generated from the specification. The PG is a directed acyclic graph $Fp(\Psi,\Omega)$, where Ψ represents the set of vertices in the graph ($\psi_i \in \Psi$) and Ω is the set of directed edges representing the precedence constraints ($\omega_i \in \Omega$). The AG is $FA(\Theta,\Re)$, where Θ represents the available architectures ($\theta_i \in \Theta$) and ($\rho_i \in \Re$) represents the available connections between hardware components. For each hardware component ($\theta_i \in \Theta$), a finite set of resource types (S) is defined. For each resource type ($s_i \in S$) there is a set of associated ratios (R_s) that specify power, delay, and cost scaling when using this type for a selected component.

The automated co-synthesis methodology optimizes the allocation, binding and scheduling (time and voltage). So, the co-synthesis can be seen as a multi-objective optimization problem that searches the design space to find an implementation that satisfies the design constraints. The search-space engine we present in this article is based on evolutionary algorithm (section 3.3). Evolutionary algorithms are able to process a set of different implementation candidates at the same time. This inherent parallelism made evolutionary algorithms suitable for problems which have complex and large search space. Figure 1 shows the

basic optimization steps. The power estimation and evaluation is based on a library of pre-characterized components.
The components' library offers hardware and software components of different types. Also, components of different granularity are modelled. These features improve the performance of exploring the design space as well as the estimation accuracy. Estimating the power consumed by a design alternative is performed by combining the number of accesses to each allocated component with the power model of that component. The power model of each component is loaded from the library.

Input: $F_p(\Psi,\Omega)$, $F_A(\Theta,\Re)$, *technology library*
Output: *allocation/binding, schedule (time and voltage)*

Step 0: Generate initial population.
Step 1: Decode implementation.
Step 2: Repair infeasible implementations.
Step 3: Evaluate and refine each implementation:
- Compute a time schedule (*if any*).
- Refine the allocation/binding.
- Compute a *voltage schedule (Figure 2)*.
- Compute objective values.
- Force penalty to reflect design constraints violation.

Step 4: Check termination (design constraints).
Step 5: Assign fitness and perform selection (SPEA2).
- Environmental selection (archive update)
- Mating selection (produce the mating pool)

Step 6: Variation: recombination operators
(Crossover & mutation)
Go to Step 1.

Figure 1. Global optimization algorithm

Each individual in the population represents a candidate implementation. The allocation/binding refinement step refines the allocation and binding to handle power-performance-cost tradeoffs in a better way. This step deals with an ordered list of types for each component and leads to allocating performance optimized instances to execute critical-path tasks and power optimized ones for non-critical path tasks. A new schedule is generated after the refinement step.
Since we assume DVS-enabled architectures, the scheduling issue in this case is transformed into a two dimensional problem: time and voltage. A list-based scheduler performs the time scheduling, whereas the voltage schedule is computed in such a way that the available slack is utilized efficiently without violating performance constraints. The computed voltage schedule is stored in a table-like form, which keeps the overhead of voltage scheduling at minimum during run-time.

3.1.3. Implementation Synthesis and Evaluation and Validation

Commercial tools and our own SDL compiler are used for translating the SDL* specifications into software implementation in C and hardware implementations in VHDL. Compilation for VHDL and C is carried out by commercial tools, which are readily available from many vendors.

3.2. Applying SVS

For applications that have predictable computational loads with a pre-determined upper constraint on performance, it is possible to estimate the benefits of SVS [13], but applying SVS introduces two new overheads: Transition time and transition energy that represent the required time and energy for changing the voltage from *level1* to *level2*, respectively [14]. The overhead of applying SVS is considered in our methodology and assumptions related to this overhead (energy and cost) are taken from [7]. At the same time, reducing the supply voltage increases the circuit delay. The following equation shows the relation between circuit delay and supply voltage:

$$delay = k_d \frac{V_{supply}}{(V_{supply} - V_t)^2} \qquad (1)$$

where: V_{supply} is the supply voltage and V_t is the threshold voltage. So, the voltage may only be reduced if the corresponding degradation in performance can be tolerated.
Using the introduced notation, the energy consumed by a task executed at voltage level V_{level} is calculated as follows:

$$E'(\psi_i) = \left(\frac{V_{level}^2}{V_{supply}^2} \right) E(\psi_i) \Big|_{V=V_{supply}} \qquad (2)$$

The time needed to execute a task is increased when this task is executed at a voltage level lower than the maximum. This in return may affect (increase or decrease) the available slack or idle time interval for other tasks which are not necessarily mapped to the same hardware. This is related to the mapping of these tasks and the precedence constraints between tasks. So, in order to take into consideration this inter-task relation, we perform the voltage level planning based on a global view of all tasks and their energy profiles. The voltage scheduling algorithm is depicted in Figure 2.

Input: $F_p(\Psi,\Omega)$, $F_A(\Theta,\Re)$, *mapping, time schedule, step.*
Output: *Voltage schedule* $V_{ss}(t)$

Step 0:
- Calculate ΔEN_i of all tasks $\psi_i \in \Psi$
- Assign $P_{priority}$ to all tasks $\psi_i \in \Psi$
- Create empty list *LS* of size *y*

Step 1:
Arrange the tasks in *LS* in a descending order of $P_{priority}$.
Step 3:
Get a (ψ_j) with the highest non-zero $P_{priority}$ from *LS*.
- If (V_{dd} is no longer > $2V_t$) → remove ψ_j from *LS*.
- Else, extend the task (ψ_j) in steps of (*n*step*).
- Update the tasks profile and propagate delay effects.

Step 4:
Return if *LS* is empty OR all tasks have $P_{priority} = 0$
Step 5:
- Calculate ΔEN of all tasks in LS.
- Assign $P_{priority}$ to all tasks.
- Go to step 1.

Figure 2. Voltage scheduling algorithm

In the figure above, (y) refers to the number of tasks. ΔEN_i refers to the energy saving for task (i) when extending its execution time (by one time step (Step, n = 1)) by scaling its operating voltage:

$$\Delta EN_i = E(\psi_i) - E'(\psi_i)\big|_{n=1} \qquad (3)$$

The achieved energy reduction is closely related to ΔEN_i [15]. So, tasks with larger energy profile are given more preference to extend their execution. The power priority ($P_{priority}$) for task ψ_i is proportional to the calculated ΔEN_i multiplied by sl_i which is defined as:

$$sl_i = \begin{cases} 1, & slack_i \neq 0 \\ 0, & otherwise \end{cases} \qquad (4)$$

The task which has the maximum effect on the energy consumption is selected firstly to extend its execution by scaling the voltage. After extending the execution of a task by means of reducing the supply voltage, the power value is updated for the selected task and the effect of the time extension for this task is propagated through other related tasks. The algorithm above terminates when one of two conditions is satisfied: 1) When the list LS is empty. This case occurs if the voltage level is reduced to a value around $2V_t$ for all tasks. Actually, the minimum possible value for V_{dd} is set relative to the worst case threshold voltage. A practical limit for it is about $2V_t$ [20]. 2) When the $P_{priority}$ = 0 for all tasks which means there is no available slack to be exploited by any of the tasks.

In the aforementioned algorithm, a task has the opportunity to stay in the *LS* list although it has temporarily no more available slack. So, artificial slacks created by other extended tasks can be utilized.

3.3. Evolutionary Algorithm Design

We created an evolutionary algorithm based synthesizer by integrating the widely used evolutionary multi-objective optimizer SPEA2 [16] to our automated co-design framework. The optimization goal is to find design alternatives with Pareto-optimal objective vectors.

In Figure 1, the initial population is generated randomly. Repair heuristics and penalty functions are used to deal with the infeasibility problem in the generated implementations. The repair mechanism is based on a priority list of hardware components that can be allocated to execute a task.

Violating design constraints (delay, cost, and power) is handled using appropriate penalty functions. Each penalty function takes into consideration the number of violations of this design constraint's type and the distance from the feasible region. For example, the penalty function for violating power constraints in a given path is given by:

$$p(g_i, J) = \mu \cdot m \cdot \sum_{i=1}^{m} \alpha_i \left(\frac{\left(P_i(\alpha, \beta) - P_{MAX_i} \right)}{P_{MAX_i}} \right) \qquad (5)$$

where P_{MAX_i} is the forced power constraint on path i in the task graph, $P_i(\alpha, \beta)$ is the actual consumed power for a given allocation α and binding β, α_i indicates how crucial violating this constraint is, m is related to the number of violations of this type of constraint, and μ is a controlling parameter. The fitness function $f(J)$, when a certain design constraint is violated, is the sum of the penalty function and the objective function $h(J)$. The obtained fitness function represents the new objective function $h'(J)$. So, the problem is transformed into another problem with $f(J) = h'(J)$. The optimization process is guided by the objective function $h'(J)$ and the fitness of an implementation is calculated based on the Pareto-optimization scheme (SPEA2).

The variation process includes applying reproduction rules (crossover and mutation) to change the genetic material of individuals. Different types of crossover (such as uniform and single point) and mutation (such as one-point and independent) are implemented. The types of these operators are specified by the user. Different controlling parameters have been experimentally specified. The selection process is performed by SPEA2. The termination condition is the satisfaction of design constraints or exceeding a pre-specified number of generations.

4. BENCHMARKS AND EXPERIMENTAL RESULTS

To investigate the effectiveness of our integrated methodology, we applied it to a set of benchmarks, part of them are taken from real-life examples. The benchmarks themselves can be categorized into two groups: The first group includes: the one dimensional FFT, the Gaussian Elimination (GE), both taken from [17], and the Optical Flow Detection (OFD) which is part of an autonomous model of a helicopter [18]. The OFD algorithm runs originally on two DSPs with an average current of 760 mA at 3.3V. It uses a repetition rate of 12.5 frames of 78×120 pixels per second. The original data for power consumption and performance are partially taken from [10]. The second group includes a set of benchmarks originally generated using the "Task Graphs For Free" TGFF [19].

The results are reported using the achieved power/energy reduction in percent. These results are categorized in three groups: The benefit of using power optimized types and architectures by using allocation/binding refinement [11], the benefit of applying SVS, and the achieved benefit by combining both SVS and the allocation refinement step. To clearly demonstrate the effect of allocation refinement using different hardware types on the performance of SVS, extra experiments are performed by forcing more stringent performance constraints while applying our design methodology.

4.1. Using Power Optimized Types

Column two in Table 1 shows the obtained power reduction when applying the methodology presented in [11]. The results indicate that the maximum power reduction that can be achieved is barely exceeding 20%. Nevertheless, this extra refinement step is valuable when combined with voltage scaling. The influence of this combination is presented bellow in more details.

4.2. The Effect of Applying SVS

The benefit of applying SVS is shown in column 3 of Table 1. The minimum benefit when applying SVS is obtained by the GE. This can be related to the stringent performance constraints forced on this application and the data dependency in the task graph. For the first group of benchmarks, the FFT achieves a power reduction of about 42% when applying SVS whereas TGFF6 seems to make the best benefit of SVS (83.5%). It is clearly seen that SVS is superior to only refining the allocation.

4.3. Using Allocation Refinement Combined with SVS

When delay constraints provide enough slack intervals to be utilized by the SVS, the effect of refining the allocation/binding is extremely limited. TGFF1 is an example on this case. Column 4 of Table 1 shows the effect of the allocation refinement step on SVS. It is clearly seen that the performance of SVS is improved, but in varying degrees.

Table 1. Power reduction in %

Benchmark	Allocation Refinement [11]	SVS	SVS & Allocation Refinement
TGFF1	19,7	68,1	69,1
TGFF2	14,8	36,4	45,1
TGFF3	12,7	64,6	77,1
TGFF4	10,8	82,6	88,1
TGFF5	13,1	60,1	64,9
TGFF6	18,1	83,5	93,6
TGFF7	13,8	30,2	70,4
TGFF8	20,3	76,6	78,9
TGFF9	16,7	37,3	83,6
TGFF10	2,7	19,6	49,5
FFT	17	42	66
GE	15	18	26
OFD	6	22	27

The effect of this refinement step increases and becomes clearer as the performance constraints get more and more stringent. Table 2 shows the achieved power reduction when more stringent performance constraints are forced by scaling the original constraints. In order to enable fair comparison, the power reduction obtained under these tight performance constraints when applying SVS are evaluated and presented in column 2 of the same table. The last column of this table shows the joint influence of SVS and the extra refinement step on the attained power reduction.

The results presented in Table 2 obviously show that the potential benefit of applying SVS is sharply limited under tight performance constraints. The maximum power reduction that could be achieved in this experiment is about 25% only.

It is clearly seen that under this condition, the effect of applying SVS can be noticeably improved when combined with the suggested allocation refinement step. For example, the effect of using SVS is almost tripled when combined with the extra allocation refinement step for TGFF4*, whereas the power reduced is improved by a factor of 15 for TGFF10*. This justifies the need for this extra optimization step.

Table 2. Power reduction in %

Benchmark	SVS	SVS& Allocation Refinement
TGFF1*	25,4	36,8
TGFF2*	20,4	43,6
TGFF3*	11,9	53,6
TGFF4*	22,3	61,4
TGFF5*	16,2	53,6
TGFF6*	20,0	84,9
TGFF7*	5,7	34,5
TGFF8*	17,3	76,5
TGFF9*	7,2	64,3
TGFF10*	3,1	45,3

As we pointed out previously, system level power optimization is not suggested to replace but to aid low level system optimization methodologies. Throughout the design process, the system will undergo other optimization steps at low abstraction levels.

5. CONCLUSIONS AND FURTHER WORK

This paper adds a new dimension to the multi-objective optimization problem of system co-synthesis, by optimizing the voltage schedule as an integrated part in the co-synthesis process. The proposed automated co-design tool for low power/energy real-time embedded systems is of valuable help for system level designers. The tool is able to explore the performance-power-cost design space. It guides the design process to low power design alternatives that satisfy other design constraints.

SVS has been successfully integrated in the co-synthesis process and its performance is remarkably enhanced by using the suggested extra refinement step. All benchmarks included in this study showed the effectiveness of the presented approach.

Currently, we are developing methodologies to reduce the overhead related to switching the voltage level when applying SVS. Further on, it is part of the future work to find out how to encode the voltage level in the gene itself.

6. REFERENCES

[1] Münzenberger, R., Dörfel, M., Hofmann, R., and Slomka, F. A General Time Model for the Specification and Design of Embedded Real-Time Systems. *Microelectronics Journal*, vol. 34, 2003, pp. 989-1000.

[2] Mohsen, A., and Hofmann, R. Characterizing Power Consumption and Delay of Functional/Library Components for Hardware/Software Co-design of Embedded Systems. In *the 15th IEEE Int. Workshop on Rapid System Prototyping (RSP'04)*, Geneva, 2004, pp. 45-52.

[3] Thiele, L., Chakraborty, S., Gries, M., and Künzli, S. Design Space Exploration of Network Processor Architectures. In *Network Processor Design: Issues and Practices*, Vol. 1, October, 2002.

[4] Teich, J., Blickle, T., and Thiele, L. An Evolutionary Approach to System-Level Synthesis. In *the 5th International Workshop on Hardware/Software Co-Design (Codes/CASHE '97)*, March, 1997.

[5] Slomka, F., Dörfel, M., Münzenberger, R., and Hofmann, R. Hardware/Software Codesign and Rapid Prototyping of Embedded Systems. *IEEE Design & Test of Computers*, 2000, pp-28-38.

[6] Ljolo, M., Raghunathan, A., Dey, S., Lavagno, L., and Sangiovanni-Vincentelli, A. Efficient Power Estimation Techniques for HW/SW systems. In *Proc. of the IEEE VOLTA'99 International Workshop on Low Power Design*, Italy, 1999, pp. 191-199.

[7] Bambha, N., Bhattacharyya, S., Teich, J., and Zitzler, E. Hybrid Global/Local Search for Dynamic Voltage Scaling in Embedded Multiprocessor. In *Proc. of the 1st int. symposium on Hardware/Software Co-design (CODES'01)*, 2001, pp. 243-248.

[8] Gruian, F. System-Level Design Methods for Low Energy Architectures Containing Variable Voltage Processors. In *Proc. of Power-Aware Computing Systems Workshop*, November 12, Cambridge (MA), US, 2000.

[9] Gruian, F., and Kuchcinski, K. LEneS: Task-Scheduling for Low Energy Systems Using Variable Supply Voltage Processors. In *proc. of Asia and South Pacific Design Automation Conference (ASP-DAC'01)*, 2001, pp. 449-455.

[10] Schmitz, M., Al-Hashimi, B., and Eles, P. Synthesizing Energy-efficient Embedded Systems with LOPOCOS. *Design Automation for Embedded Systems*, 6, 2002, pp 401-424.

[11] Mohsen, A., and Hofmann, R. Power Modelling, Estimation, and Optimization for Automated Co-Design of Real-Time Embedded Systems. In *Proc. of the 14th International Workshop on Power and Timing Modelling, optimization and Simulation, (PATMOS'04)*, Greece, 2004, pp. 643-651.

[12] Mitschele-Thiele and Slomka, F. *Co-design with SDL/MSC*. IT Press, 1999.

[13] Pering, T., Burd, T., and Broderson, R. The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms. *ISELPED 98*, (Monterey, CA USA, August 10-12, 1998), ACM, 2000, pp. 76-81.

[14] Burd, T., and Broderson, R. Design Issues for Dynamic Voltage Scaling. In *Proc. of the 2000 international symposium on Low power electronics and design*, Italy, 2000, pp. 9 – 14.

[15] Schmitz, M. and Al-hashimi, B. Considering Power Variation of DVS Processing Elements for Energy Minimization in Distributed Systems. In *Proc. of the international symposium on System Synthesis (ISSS'01)*, 2001, pp. 250-255.

[16] Zitzler, E., Laumanns, M., and Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. *Evolutionary Methods for Design, Optimization, and Control*, CIMNE, Barcelona, Spain, 2002, pp. 95-100.

[17] Topcuouglu, H., Hariri, S., and Wu, M. Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing. *IEEE trans. on Parallel and Distributed Systems*, Vol. 13, No. 3, 2002, pp 260-274.

[18] The Wallenberg Laboratory for Research on Information Technology and Autonomous Systems. Available at http://www.ida.liu.se/ext/witas.

[19] Dick, R., Rhodes, D., and Wolf, W. TGFF: Tasks Graphs for Free. In *proc. of Intl. Workshop on Hardware/Software Codesign*, March, 1998.

[20] Landman, P. *Low-Power Architectural Design Methodologies*, Ph.D. Thesis, U.C. Berkeley, Aug. 1994.