

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of Accent Srl's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

Fast System-Level Design Space Exploration for Low Power Configurable Multimedia Systems-on-Chip

Flavio Polloni, Luca Mazzoni, Serge Di Matteo
Accent Design Technology Services - <http://www.accent.it/>

Abstract

In this paper we present a methodology to perform a very fast system-level design space exploration of parameterized multimedia architectures. The methodology is applied to a functional model of a wireless multimedia platform developed by Accent. Our approach guarantees the correctness of the dataflow and to find pareto-optimal solutions considering area, performances and power consumption. The implementation of an Earliest Deadline First like arbitration policy achieves a very efficient scheduling of bus requests whereas the architecture can be re-configured run-time in order to optimize power consumption dynamically. The results of applying this design methodology to an industrial project are presented.

I. Introduction

In system-on-a-chip (SOC) design, extensive reuse of pre-designed intellectual property (IP) cores has become necessary to design more and more complex systems with a shorter time-to-market. In recent years, many processors, memories, communication links, interfaces and peripherals have been developed to achieve this result. These IP blocks are highly parameterized to satisfy the requirements of as many as possible applications. Designers can choose among these components, each one configurable in several ways, in order to build a system optimized with respect to various and conflicting objectives: it has to achieve the performances required by the target application but with the minimum cost (silicon area) and power consumption [11]. In this scenario, it is crucial to face the system-level design with a quantitative approach, evaluating the greatest number of system configurations in a reasonable amount of time. In fact, the IP cores are often well characterized in area, performances and, since the last few years, also in power consumption but these figures are greatly affected by IP interaction at system level. A well-structured methodology for architectural design does not yet exist, even if many approaches have been proposed in recent research works. Some are based on simulations at system-

level, using accurate power and performance models of each component of the system [10]. The drawback of this approach is that simulations are quite slow, even with high-level models, whereas the design space is very large. For this reason, explorations based on parameters dependencies [8] or genetic algorithms [2] have been proposed.

This paper describes our solution for a fast design space exploration for parameterized SOCs. Our goal is to ensure the functional correctness of the dataflow and to optimize the architecture at system level finding the pareto-optimal solutions (i.e. supporting the target applications at the minimum cost – die area and energy consumption). Our methodology is based on some mathematical algorithms and thus avoids system-level simulations (the exploration bottleneck). The focus of our approach is the optimization of the system bus usage that affects greatly performances and power consumption of the system. In fact, the more advanced the silicon technology, the greater the impact (up to 40%) of the system bus on the global system power consumption [7]. The energy consumption can be reduced by setting the speed of the system according to the data rate of the application: such power optimization technique for multimedia applications is also used in [1] and [9]. To perform and speed-up the verification of system configurations, we analyze the worst case application workload and real-time constraints (e.g. for multimedia processing) applying the results of RTOS scheduling theory to the system architecture. In fact, the scheduling problem, extensively studied for tasks running on CPUs, has to be considered whenever a resource is shared among many users. Kettler et al. [5], [6] have evaluated different arbitration policies for the traffic on system bus but they considered only fixed priority, round robin and hybrid scheduling schemes. On the contrary, we have applied a non-preemptive Earliest Deadline First (EDF) policy that has been demonstrated to be the best non-preemptive scheduling policy [4].

This methodology, explained in details in section V, adopts a functional model of the entire system (CPU,

memories, bus, peripherals, DMAs). The models, presented in section III, are applicable to different bus protocols and permit a comparison among systems with different microprocessors and peripherals. The target application is multimedia processing whereas the referenced architecture is described in section II. The EDF scheduling policy is introduced in section IV. A case study taken from a real design, to which we successfully applied the methodology, is presented in section VI.

II. Target System Architecture

The system architecture considered in this work has been developed from the Accent design experience. The base architecture, shown in figure 1, is customizable with the insertion of all the IP cores required by the target application. All the parameters (bus width, cache size and policy, peripherals transfer rates,...) of the platform can be configured and most of them are run-time programmable in order to dynamically optimize the system. Finding their optimal values (with respect to area, performances and power consumption) is the goal of the methodology presented. The architecture is designed for real-time multimedia processing (where performances have to fit the rate of the data stream) and wireless portable devices (in which battery lifetime is critical). It has an embedded processor with cache memories and a Direct Memory Access (DMA) controller for each communication channel. This permits great efficiency in the use of the system bus. In fact, such dedicated DMAs reduce the number of data transfer on the system bus, and free the processor from spending clock cycles to cope with I/O data movement.

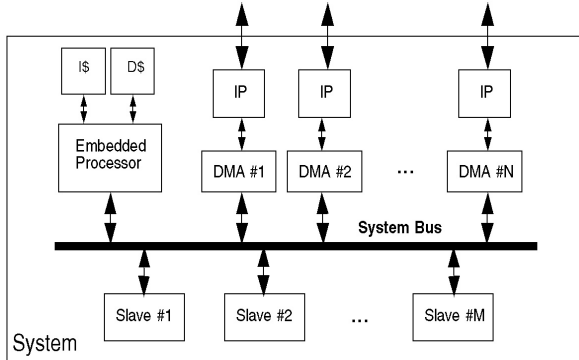


Fig. 1. Target architecture of the system.

III. Model of the System

First of all, we have developed a clock cycle accurate model of the entire system (bus, DMAs, microprocessor).

A. Bus Transfer Model

We have considered bus protocols supporting single and burst transfer modes. The time (T) required for a transfer depends on the amount of data to be transferred (D), the access time of master and slave devices and their clock periods, the bus clock period (T_{bus}) and bus protocol. The bus transfer model is summarized in (1) where N is the number of bus clock cycles required for each datum

transferred in burst mode whereas the higher transfer latency of the first datum is included in the additional term N_0 .

$$T = (N_0 + D \cdot N) \cdot T_{bus} \quad (1)$$

B. DMA Model

A DMA is a bus master that receives data at a constant transfer rate from a peripheral and, using its internal FIFO, transfers them in packets to the system memory. It performs the opposite process in transmission. We have modeled this function for bus scheduling purposes.

The k -th DMA is configured by the set of parameters $C_{DMA}^k = (FIFO^k, D_t^k, F_{IP}^k)$, respectively the depth of its FIFO buffers, the size of its bus transfers, the transfer rate of the peripheral it is connected to. For the bus requests scheduling the relevant set of parameters is $S_{DMA}^k = (T_{DMA}^k, d_{DMA}^k, P_{DMA}^k)$, respectively the bus transfer time, the relative deadline of the bus transfers, the time interval between two bus requests.

Figure 2 represents the model of a DMA in the receiving case. When, at time $t_{r(i)}$, the DMA reaches a previously defined threshold (D_t^k) of FIFO words, it requires the use of the bus to the arbiter. If the request is not immediately satisfied, the DMA has to wait and in the meanwhile it goes on with receiving data from the peripheral. When the DMA is granted access to the bus, it begins the transfer (at time $t_{s(i)}$).

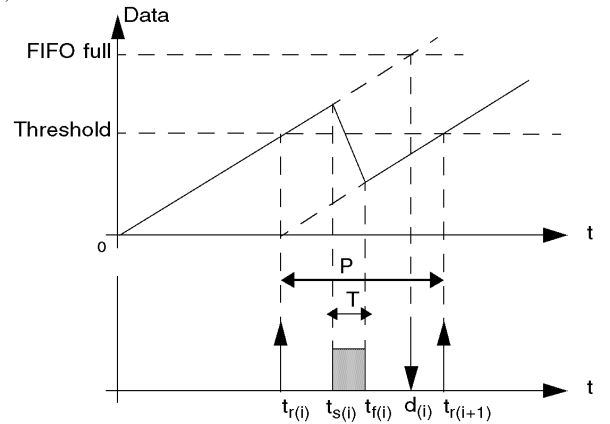


Fig. 2. DMA model.

The transfer time (T_{DMA}^k) depends on the amount of data to be transferred and on which slave is accessed according to the bus transfer model (1). We have assumed that the amount of transferred data is always equal to the threshold ($D = D_t^k$).

As the FIFO has a finite depth, an overflow occurs when all its locations are full and a new datum arrives. The datum is lost because it cannot be stored into the buffer and the system functionality may be compromised. Therefore the transfer must be performed before a certain *deadline* d_{DMA}^k , as in real-time systems. A FIFO underflow is less critical but anyway it should be avoided because it slows down data transfer. The deadline, in bus clock cycles, is

given by equation (2).

$$d_{DMA}^k = d_{(i)} - t_{r(i)} = \left\lfloor \frac{(FIFO^k - D_t^k) + 1}{F_{IP}^k} \cdot \frac{1}{T_{bus}} \right\rfloor \quad (2)$$

As the transferred data are always equal to the threshold, the interval between two bus requests is constant and given by:

$$P_{DMA}^k = t_{r(i+1)} - t_{r(i)} = \left\lfloor \frac{D_t^k}{F_{IP}^k} \cdot \frac{1}{T_{bus}} \right\rfloor \quad (3)$$

C. CPU Model

During instruction fetch and data access, the processor usually accesses the cache memories. It requires the use of the system bus only in case of cache miss, in order to access the main memory, or the I/O registers. The occurrence of these accesses is non-deterministic but application and data dependant. For this reason, the model of the processor can only estimate on average the frequency of bus requests using some parameters related to a profiling of the application. In particular, the frequency of main memory accesses depends on cache miss rates (related to cache size, block size and cache associativity), percentage of load/store instructions, processor CPI, processor and bus clock frequencies.

The transfer time (T_{mp}) is given again by (1) where the amount of transferred data (D) is given by the ratio between the cache block size and the bus width. The processor bus requests cannot be modeled as periodic (as for the DMAs) because embedded processors usually stall their execution when they are waiting for data in main memory and because the response time of a bus access is not constant but depends on the bus contention. Therefore, the model sets only the average time from the completion of the last transfer and the new bus request. The deadline (4) for the processor bus requests depends on the maximum permitted latency (l_{max}) for the bus response time. It is set to avoid unwanted effects on the real time execution (e.g. audio/video stream, interrupt service latency, etc.).

$$d^{mp} - t_{r(i)} = \left\lfloor \frac{l_{max} \cdot T_{mp}}{T_{bus}} \right\rfloor \quad (4)$$

IV. Bus arbitration policy

The choice of arbitration policy and scheduling algorithm used by the bus arbiter affects greatly the optimal system configuration. In many systems a non-preemptive algorithm is preferable (a request with higher priority cannot interrupt the current transfer) because it permits a better efficiency in the use of the bus (less handover cycles and longer burst transfers) with consequent benefits on power consumption.

A. Earliest Deadline First algorithm

The EDF algorithm [3], used in our architecture, schedules the bus requests giving the higher priority to the

bus master whose transfer has the earliest absolute deadline. It is dynamic because the priorities of the masters are not fixed *a-priori*, but depend on the actual deadlines. EDF is very effective for the scheduling (i.e. it permits to schedule sets of requests with tight requirements, not schedulable with other policies), as shown in [4], but it requires to compute at each time which master must be granted. To solve this drawback, we implement an EDF-like bus arbiter. It monitors continuously the actual number of free FIFO locations of all DMAs and uses a look-up table to work out priorities and to grant the bus requests.

B. Feasibility Conditions

The following conditions are an extension to the necessary and sufficient conditions described in [4]. Condition (5) requires the bus not to be overloaded: the cumulative bus utilization of all the bus masters cannot exceed the unit. Condition (6) is introduced by the non-preemptive feature of the scheduling. This condition has to be evaluated for each master, starting from the one with the shortest deadline, and considers the worst case of bus requests interaction (i.e. the master has just been granted the bus and, at the successive bus clock cycle, all the masters with a shorter deadline require the bus). If all these masters do not exceed their deadlines the scheduling is feasible. As only the DMAs requests are modeled as periodic events, condition (6) is applied to DMAs only, while the bus cycles used by the processor (mT_{mp}) are added when a processor transfer occurs. Each time the processor is granted the bus, the respect of its deadline has to be checked (7).

$$\frac{T_{mp}}{P_{mp} + T_{mp}} + \sum_{k=1}^n \frac{T_{DMA}^k}{P_{DMA}^k} \leq 1 \quad (5)$$

$$\left\{ \begin{array}{l} \forall k, 1 < k \leq n; \forall L, P_{DMA}^1 < L \leq P_{DMA}^k \\ L \geq T_{DMA}^k + \sum_{j=1}^{k-1} \left\lfloor \frac{L-1 + P_{DMA}^j - d^j}{P_{DMA}^j} \right\rfloor \cdot T_{DMA}^j + m \cdot T_{mp} \end{array} \right. \quad (6)$$

m : number of processor bus accesses in $[0, L]$

$$\text{For each processor access: } d^{mp} \geq t_{s(i)} + T_{mp} \quad (7)$$

V. Algorithm for Design Space Exploration

This algorithm explores the design space of DMA thresholds and bus frequencies. For each point of the design space, it adapts the FIFO depths in order to guarantee the correct system functionality, even in the worst case of bus contention. For each solution able to schedule the bus requests, it evaluates the cost functions and selects the set of pareto-points, discarding all the dominated solutions.

A. Description of the Design Algorithm

The flow diagram of the algorithm (fig. 3) is split into 3 phases:

1. The design space is defined from an analysis of the selected IPs and the application requirements.

2. For each point of the design space, the feasibility conditions are checked, starting from the minimum depths of the FIFOs (equal to the value of the thresholds). Their size may be increased until the conditions hold or the scheduling remains unfeasible even with the greatest possible sizes (in this case the configuration is discarded).

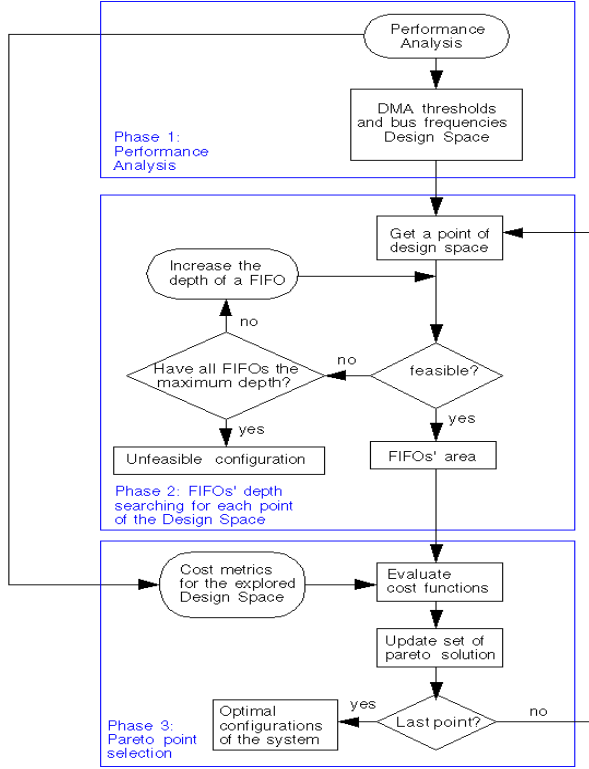


Fig. 3. Flow Diagram of the design algorithm.

3. After computing the cost functions of each solution, the set of pareto-optimal configurations is returned.

B. Cost Functions

We have defined four cost functions (based on chip area, bus efficiency, average processor execution time and impact on system power consumption), associated to each point of the design space. The area is evaluated as sum of the entire chip area plus the overhead of the FIFO depths resulting from the phase 2. The cost function related to the bus efficiency is the average time, over all the bus masters, to transfer a single datum. The performances of the processor are estimated as the average time required to execute a single instruction, given by the processor CPI (obtained by benchmarking) plus the average penalty in time due to main-memory or I/O accesses (time spent waiting for the grant plus transfer time). The impact on power consumption of the parameter values is evaluated in a technology independent way. It considers the clock frequency of the system bus (to which the power is directly proportional) and the power consumed in main memory page switching, assuming that each peripheral accesses a memory page different from the one accessed by the previously granted bus user. The power consumed by the components of the system has obviously to be added to the

described estimations but this depends on technology and the IP cores inserted in the system.

C. Run-time Reconfiguration

The algorithm can be applied not only at design time but also at run time in order to change dynamically the system configuration according to the actual workload, achieving further energy savings. In fact, the system must be designed to sustain the maximum workload, but at a certain time some peripherals may be inactive or working at a low data rate. Whereas the depths of the FIFOs are fixed at design time, the bus clock frequency and the DMAs thresholds are programmable. The algorithm described previously can be applied with the actual number of active DMAs and their transfer rates. In order to avoid the time overhead of executing the algorithm and reduce energy consumption, the optimal values of the target parameters for the most common and significant combinations of active DMAs can be pre-computed and stored in a look-up table.

VI. Case Study: a Multimedia System

The design methodology described in this paper has been applied to the real project of a 3G phone design. The system includes many IP cores: an ARM9 processor (with instruction and data cache and tightly coupled memories), SDRAM, Flash and SRAM memories and 12 peripherals (UART, I²C, IrDA, USB, MCI, I²S) each one with its own transfer rate (from 125 kbytes/s to 6 Mbytes/s) and a DMA controller. The system bus is a 32 bits AMBA AHB Bus [12] (supporting both single and burst transfers). This system has been modeled as previously described. The results refer to an implementation in 0.18 μm CMOS technology. The DMA engine used in this system was developed by Accent. It supports different burst sizes (up to 16 words) and FIFO depths (1,2,4,8,16,32 words) and is synchronous with the system bus. The area of the DMA increases linearly with the size of the FIFO. The area without FIFO is 12,954 equivalent NAND gates whereas for each added word it increases of 1,118.5 gates up to 47,745 gates for 32 word FIFO. The area of the whole system without FIFOs is 58 mm^2 (equivalent to 5M gates).

A. Configuration for the Design Space Exploration

All the combinations of possible DMA thresholds define a huge design space (5^{12} combinations), so we considered only a sub-set of them (3 thresholds for all the DMAs but only 2 for DMAs with the slowest and fastest data rate) with values such that about the same number of bus requests per second is obtained for all the DMAs. The parameters N and N_0 (reported in table I) are determined by memory access latencies and clock frequencies (up to 65 MHz the SDRAM and up to 52 MHz the Flash), bus protocol, CPU clock frequency (up to 140 MHz). We considered three main groups of configurations, each one exploiting the maximum clock frequency of SDRAM, Flash or CPU. The processor accesses both Flash and SDRAM

memories while the DMAs accesses only the SDRAM. Processor CPI (0.909), load/store percentage (35%), instruction and data cache miss rates (2% and 5% respectively) have been estimated for multimedia applications.

TABLE I
PARAMETERS OF THE BUS MASTERS

	GROUP I			GROUP II			GROUP III		
	MHz	N_0	N	MHz	N_0	N	MHz	N_0	N
Processor	130	11	2	104	10	1	140	9	1
DMA	65	9	1	52	9	1	46.7	7	1

B. Experimental Results

Using this methodology, we explored 708,588 configurations in only 1 hour of CPU time with a SPARC Ultra2 workstation. The methodology permits to discard all the configurations dominated by others or not permitting a feasible bus requests scheduling. Figures 4 and 5 show the area-bus efficiency and area-number of switches pareto-points. For each group of configurations (see table I), the point with minimum number of switches per second presents also the best bus efficiency. The solutions show a limited swing in area (because the area of the FIFOs is quite small with respect to the overall chip area) but an interesting 50% difference between minimum and maximum number of switches. It is important to note that without this methodology, we have no criterion to decide the FIFO sizes (with 32 words FIFOs the system area would be 64.55 mm²) and ensure the feasibility of bus requests scheduling. The estimated processor performances are constant within each group of configurations, but differ greatly from one to another. It is interesting to note that they are better for the second group than for the first one, despite a lower processor clock frequency (104 vs 130 MHz). This happens because the Flash access time does not change, but, reducing the bus clock frequency, the penalty in clock cycles for an access is lower (N and N_0 in our model). The comparison among all the configurations is summarized in table II.

TABLE II
COMPARISON AMONG SYSTEM CONFIGURATIONS

GROUP OF CONFIGURATIONS	AREA (mm ²)	BUS EFFICIENCY (NS / DATUM)	PROCESSOR PERFORMANCE (NS / INSTR.)	NR. OF SWITCHES (x100,000)
1	59.11-61.02	21.87-33.77	30.07	20.50-31.03
2	59.11-61.02	27.34-44.02	27.97	20.89-31.01
3	59.11-61.15	28.44-42.88	20.78	22.61-33.91

VII. Conclusions and future work

We have presented a methodology to perform a fast design space exploration of a parameterized multimedia System-on-Chip, to ensure the correctness of the dataflow at system-level and to find pareto-optimal solutions considering area, performances and power consumption. The results of applying the methodology to a real design have been presented. We are now working on improving the design space exploration algorithm, extending the

flexibility of the model and architecture parameters and the accuracy of the cost functions.

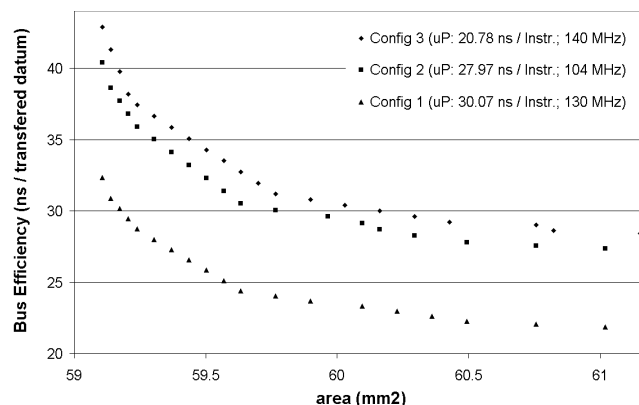


Fig. 4 Area-Bus Efficiency pareto points.

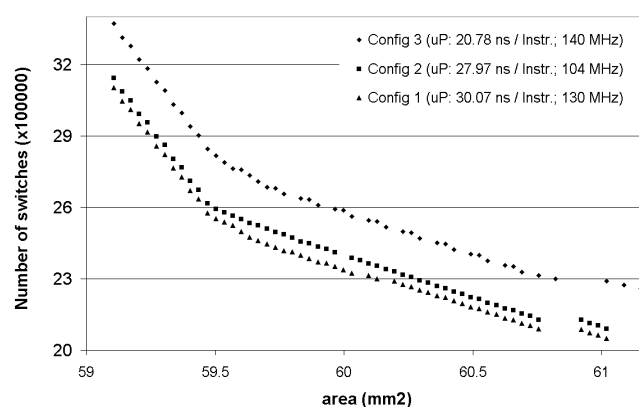


Fig. 5 Area-Number of Switches pareto points.

References

- [1] A. Acquaviva, L. Benini and B. Riccò, *An Adaptive Algorithm for Low-Power Streaming Multimedia in Processing* IEEE Design and Test in Europe Conference (DATE) Munich, 2001 pp. 100-110
- [2] G. Ascia, V. Catania and M. Palesi, *A Novel Approach to Design Space Exploration of Parameterized SOCs*, IFIP International Conference on Very Large Scale Integration, The Global System on Chip Design & CAD Conference, Montpellier, 2001, pp 449-454
- [3] G. C. Buttazzo *Hard Real-time computing Systems*, Kluwer Academic Publishers, 2000
- [4] K. Jeffay, D. F. Stanat and C. U. Martel, *On non-preemptive scheduling of periodic and sporadic tasks with varying execution priority*, IEEE Real Time System Symposium, 1991 pp 129-139.
- [5] K. A. Kettler and J. K. Strosnider, *Scheduling Analysis of the Micro Channel Architecture for Multimedia Applications*, IEEE Conference on Multimedia Computing and Systems, Boston, 1994
- [6] K. A. Kettler, J. P. Lehoczky and J. K. Strosnider, *Modeling Bus Scheduling Policies for Real-time Systems*, IEEE Real-Time Systems Symposium, 1995
- [7] T. D. Givargis, J. Henkel and F. Vahid, *Interface and Cache Power Exploration for Core-Based Embedded System Design*, IEEE Conference on Computer-Aided Design (ICCAD), 1999, pp. 270-273.
- [8] T. D. Givargis, J. Henkel and F. Vahid, *System-level Exploration for Pareto-optimal Configurations in Parameterized Systems-on-a-chip*. International Conference on Computer Aided Design (ICCAD), 2001.
- [9] Y. Shin, K. Choi and T. Sakurai *Power optimization of Real-time Embedded Systems on Variable Speed Processors* Proceedings of IEEE/ACM Conference on Computer Aided Design, Nov. 2000
- [10] T. Simunic, L. Benini and G. De Micheli, *Cycle-Accurate Simulation of Energy Consumption in Embedded Systems*, DAC 1999
- [11] A. Sangiovanni Vincentelli, *Defining platform-based design*, published on EEdesign, February 5, 2002
- [12] ARM AMBA Specification, Rev. 2.0