System level design space exploration for multiprocessor system on chip

Issam Maalej, Guy Gogniat, Jean Luc Philippe
European University of Brittany - UBS
CNRS, FRE 3167, Lab-STICC
Lorient - FRANCE

Mohamed Abid
GMS Laboratory
ENIS, University of Sfax
Sfax, Tunisia

Abstract

Future embedded systems will integrate hundreds of processors. Current design space exploration methods cannot cope with such a complexity. It is mandatory to extend these methods in order to meet future design constraints. We believe one solution is to add a new design exploration step above current methods. This extension corresponds to an abstraction rising to provide designer with a restricted design space. We propose in this work to enrich the classical exploration approaches by a pre-exploration step which reduces the architecture design space. This new step i) simplifies ii) performs and iii) makes possible, for a complex application the architecture exploration for future tera-scale multiprocessor-based systems. This method drastically reduces the architecture space at a higher level of the design flow which mitigates the codesign complexity and enables the designer to explore a large set of architectures.

1 Introduction

Since several years, we are facing a significant increase in the complexity of systems on chip design. Indeed, technology advances allow more and more functionalities to be integrated into a SoC. In the future the technology scaling will continue to follow Moore's Law, providing integration capacity of billions of transistors and providing an abundance of interconnections to realize complex architectures [1]. Thus SoCs will be made of hundreds of software and hardware components communicating through many resources.

SoC designers are already facing with conflicting design requirements regarding performance, flexibility, power consumption, area, reliability and cost. A great part of those conflicts are solved when the architecture is defined. However, finding an efficient architecture, which satisfies constraints as real time, low power and also time to market, low cost becomes exponentially difficult in the jungle of hardware and software resources. Generally, system designers use their own experience to define their SoC architecture or rely on a platform based design approach.

In order to enable designer to face these current and future design challenges we propose in this paper a design approach based on extending the traditional exploration methods by performing a pre-exploration step before any other design tasks. This step consists in analyzing the application at a high level to reduce the size of the solutions space. This rising of abstraction is mandatory, when dealing with hundreds of processors, to efficiently address system level design decisions before refining the architecture through traditional codesign approaches.

In the following sections, we present the pre-exploration approach and the different metrics used to guide the exploration. We also introduce the genetic algorithm defined to automatically perform the pre-exploration step. Finally, we present some results to explain and validate our proposition.

2 Related work

The exploration of architecture consists in, theoretically, exploring all the space of architectures and extracting the architecture which best optimizes the costs of the system (e.g. time, area, power consumption). However, considering the complexity of the application, it is difficult to take into account all the architecture elements (hardware components, software components, communication).

In [2], Lahiri et al. explore, initially, the space of the architectures independently of the communication architecture and with a communication time equal to 0. Then, manually they choose the communication architecture and parameterize it with a tool they have developed. In [3] and [4], the authors initially consider the architecture of communications and then explore the architecture space. These methods offer a rather good exploration. However, facing expected applications and architectures containing hundreds of hardware, software and communication components, this type of exploration becomes increasingly complex and expensive. In [5], the authors explore the architecture by mapping the communication resources from a library. In [6], the authors propose a methodology based on platform. This method consists in mapping the application into a pre-defined platform. Such an approach significantly reduces the costs and the complexity of the application since the architecture space is reduced to the one offered by the platform. However, the system performance is limited by the platform performance and the architecture is constrained to the platform architecture. Finally in [7], the authors reduce the exploration process by dividing it into two steps.

978-0-7695-3170-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ISVLSI.2008.34

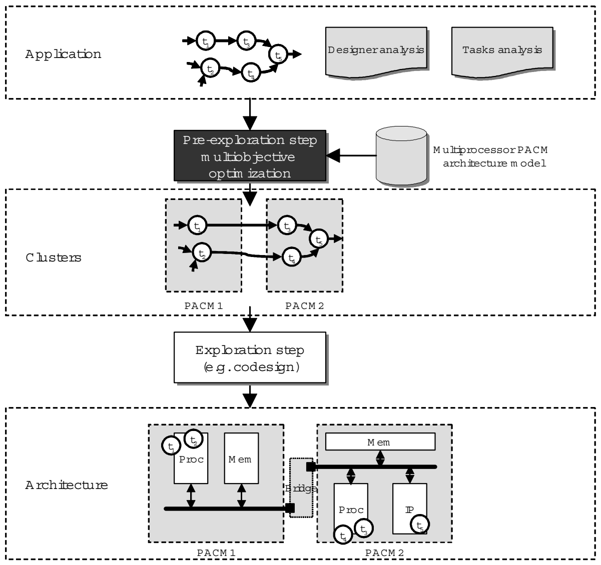
Application
Designer analysis
Tasks analysis
Pre-exploration step multiobjective optimization
Multiprocessor PACM architecture model
Clusters
PACM 1
PACM 2
Exploration step (e.g. codesign)
Architecture
Proc
Mem
Bridge
Mem
Proc
IP
PACM 1
PACM 2

Figure 1. Architecture exploration design flow.

They demonstrate the effectiveness of their method for a vast space of architecture. However, the cost of the application remains rather high since the two phases require estimates or computations related to the task performance on each software or hardware resources.

In conclusion, certainly an exploration in only one step of all the solution space enables the designer to converge toward an efficient architecture. However, this method is expensive and encounters an increasing vast architecture space. Thus, this kind of approach will reach its limits shortly and other solutions need to be considered. Taking into account the architecture at a higher level of exploration and splitting the exploration process will reduce the complexity and the costs of the exploration. We propose such an approach by extending current methods of exploration.

3 Extended new design flow

Our approach is motivated by the following concerns. We target a methodology which presents a tradeoff between: i) **Accuracy** - The capacity of the method to find an optimal architecture, ii) **Robustness** - The capacity of the method to quickly explore a complex set of architectures and iii) **Cost** - The capacity of the method to mitigate the exploration cost in time and money.

For that purpose we extend the traditional methods of explorations (as [3] and [4]), by taking benefit of the advantages of platform based design, progressive exploration and tasks analysis. As shown in Fig. 1 we perform a pre-exploration step which consists in collecting information from the application at a high level and computing information from the tasks analysis and the designer experience. This analysis takes into account a model of architecture in order to drive the exploration. This model is generic enough to cover an important part of the architecture space.

3.1 System designers analysis

System designer's experience is essential during the design process since he provides some guidance to the design flow. For example, designer usually analyzes the application before going through the design process in order to evaluate the overall system costs. He also checks how many components are available in his library before defining the system architecture. However, manually reaching a tradeoff between system costs and system performances may become impractical. Designer has to take some key decisions early in the design process that will significantly impact the final solution.

We believe it is essential to take into account designer's experience but also to help him during the strategic decision phase. For that, information from designer is collected to enrich the system specification. For example designer can specify if it is better or not to execute a task into a processor; he can also define if a task should be implemented into a hardware IP available from the library. This information is taken into account during the pre-exploration step in the $Affinity$ metric as will be explained in Section 4.1; however his decisions are not automatically respected if finally they penalize the overall performances and costs.

3.2 Tasks analysis

The most effective and simple way to control the impact of the tasks implementation consists in implementing each task into all the execution resources and to compute or to estimate all execution costs. It clearly appears that for simple architectures it is conceivable but intractable for multiprocessor architectures when dealing with hundreds of resources.

Another approach is to perform a high level analysis of the tasks. Several works as those presented in [8] and [9] propose such a technique. This analysis provides some rough evaluations of the system performances which help designer to take decision about his architecture early in the design process. This analysis, although it helps to define some system architecture parameters, is not sufficient to fully explore the architecture. Indeed, the communication, memory sharing and different other interactions between the application tasks have also a significant influence on the exploration process and have to be considered. We have defined several metrics (Section 4.2) that capture these characteristics which are used during the pre-exploration step.

3.3 PACM architecture model

To explore the architecture space, we use an architecture model. This architecture model is generic enough to cover a large number of architectures especially architectures which optimize flexibility, communication, memory organization and processor mapping. To target high complexity systems we propose a multi PACM architecture model. A PACM (Processor, Accelerator, Coprocessor and Memory) is a processing node which consists of one processor at most, some coprocessors if necessary, one or several memories, and some hardware components (i.e. hardware accelerators). A PACM can be specialized to perform specific processing depending on the processor characteristics and associated hardware IPs. Examples of specialized PACM are signal, cryptography, 3D, video processing elements. This multi PACM architecture consists of a number of PACMs which communicate through communication resources (e.g. bus, NoC). This architecture model is an abstract view and allows us to: i) have information about location of the tasks before the architecture generation, ii) have information about memory size at an early step, iii) simplify the communication design, iv) simplify the addition or the removing of processors or other components, and v) have information about data transfer location.

4 Pre-exploration step

The goal of the pre-exploration is to cluster the tasks into PACM models based on the analysis of the application (Fig. 1). The analysis is based on the interactions and the exchanges between tasks and on the properties of the tasks. The application is specified with a tasks graph describing the properties of the tasks and the dependencies between them. The analysis is based on metrics to quantify the properties of the application. A genetic algorithm is used to cluster the tasks into PACM in order to optimize the costs of the system.

4.1 Specification tasks graph of the application

The tasks graph describes the application at the tasks level. It describes the characteristics of each task within the application and the interactions between the tasks such as dependencies or data transfers [10]. The graph consists of nodes and edges.

A node represents a task and an edge corresponds to a data dependency. The node is weighted with different values, which characterize the task. Each node is characterized by the following attributes:

Throughput constraint (T_c bits/s). An application has generally some computing constraints especially input and output constraints.

Affinity vector. It is a vector, which gathers the affinity values of a task t_i to Processing Element (PE) possibilities. $Affinity(t_i)(PE_j)$ of task t_i to a PE PE_j is a real value between 0 and 1 (respectively unsuitable and suitable to the PE). These values can be determined based on prior experience of the designer and/or based on some work as proposed by Sciuto et al. [8].

Local memory size (LM_size). It determines the size of the memory resources required by the task. This value can be evaluated using existing tool [9].

Execution rate ($Exec\_rate$). This value indicates the number of executions of the task during one execution of the system.

An edge represents the interaction between tasks such as dependency and data transfer. It is labeled with information about data size transfer and the type of transfer. The $Word\_size$ is a value representing the granularity of the data transfer. The $Word\_number$ is a value representing the number of words transferred from a task to another.

4.2 Metrics for pre-exploration

Several metrics have been defined to perform the analysis of the system [10]. Each metric captures some specific properties of the system, communication, memory, parallelism, performance. The goal of these metrics is to provide information to converge toward the definition of a system that optimizes the performance, the area and the power consumption.

4.2.1 Communication metrics

These metrics evaluate the degree of optimization of communications and their impact on the system performances. Due to system complexity, several communication resources are required. The tasks graph is split into clusters (PACM model). Inside each cluster communication resources are shared. Communication resources are also required between clusters. To optimize the communication cost in the architecture, we target two objectives:

Equilibrate the exchanged data sizes within the different clusters To optimize the communication cost, exchanged data quantity must be the lowest in each cluster. Furthermore it must be equally distributed among the different clusters. For that, we define the E_De metric, which informs about the balance between the exchanged data quantity in each cluster. When the data are balanced between all the clusters, E_De is equal to 1.

Minimize data transfers between clusters To evaluate this point, we propose two metrics. The first one computes the degree of exchanged data between clusters. This metric is named $DataExchangeInterCluster$ metric $DEIC$. If all data is exchanged between clusters, the clustering is very poor, and the metric $DEIC$ is 0. The second one computes the degree of connections between clusters. It is named $ConnectionInterCluster$ metric CIC. When close to 1 it means that most of the connections are within each cluster which minimizes the total load onto the global communication architecture.

4.2.2 Memory metric

This metric enables designer to avoid memory bottleneck and to improve global performance (i.e. power, area, time). For that, the metric Mem informs about the best distribution of memory. This metric evaluates the memory resources size needed within a cluster: shared and local resources. For local resources, the local memory size LM_size is computed using an existing tool [9]. Shared memory resources is determined using the tasks graph. This metric evaluates from the tasks graph the percentage of memory used by each cluster and verifies the balance of the memory distribution (the value is defined between 0 and 1). This value is 1 when the memory is the same in each cluster.

4.2.3 Affinity metric

When a task t_i has the highest affinity to a PE_j, its performance is optimal when implemented on this PE_j. We define a metric AFF which is equal to 1 when all tasks are executed in the resource that optimizes their performances [10]. This metric results from the $Affinity\ vector$ associated to each task t_i of the tasks graph (Section 4.1).

4.2.4 Throughput constraint metric

Some tasks have a throughput constraint. This throughput is the minimum throughput that communication resources have to respect to meet constraints. The best clustering has to efficiently distribute the throughput constraints to optimize system costs and performance. We define this metric as throughput constraint metric T_c.

All the metrics presented above (detailed in [10]) are used to explore the architecture space. This step corresponds to a multiobjective problem where the goal is to find a solution which maximizes the metrics. We use a genetic algorithm to explore the solution space and to search for the best solution.

4.3 Genetic algorithm

Using a genetic algorithm is interesting to perform the pre-exploration step since several parameters, that may be in conflict, need to be optimized. The genetic algorithm initially explores an initial set of architectures. The pre-exploration metrics are then computed to evaluate the performances of these architectures. Solutions which maximize the metrics are selected to constitute the parents of the following generation. This process is iterated several times until the majority of the solutions are the same one (the best solution). Once the genetic algorithm ends its computation the pre-exploration is achieved. The application is partitioned into several clusters (i.e. PACM) and the solution is fully characterized with the exploration metrics.

5 Results

In this section, we evaluate our approach on two examples: an UMTS transmitter [11] and an AC3 decoder [12]. First, we discuss the different metrics by computing them on several examples related to UMTS. Then we evaluate the efficiency of the genetic algorithm for both applications.

Table 1. Characteristics of 2 solutions for a 2-cluster system.

	Solution 1		Solution 2	
	Cluster1	Cluster2	Cluster1	Cluster2
Tasks	1-2-3-4 8-12-13-14	5-6-7-9 10-11	1-2-3-4 5-6-7	8-9-10-11 12-13-14
data (bits)	80,368	80,550	7,168	192,150
$ComDegree(p)$	0.40	0.40	0.03	0.96
$shared\_mem(p)$	39,600	1,200	1,200	39,600
$Mem\_Degree(p)$	0.97	0.02	0.03	0.97
min(T_c)	116,800	15,000	116,800	15,500
max(T_c)	38,400,000	15,000	116,800	38,400,000
$Affinity\ metric$	0.91	1.00	1	0.04

Table 2. Metrics for the 2 solutions for a 2-cluster system.

Individual (solution)	E_De	$DEIC$	CIC	Mem	AFF	T_c
1	0.99	0.80	0.85	0.03	0.91	0.003
2	0.04	0.99	0.93	0.03	0.91	0.003

Using UMTS we compare our solution with a hand-made one and with AC3 we show the benefit of doing the pre-exploration step before a codesign exploration.

5.1 Metrics validation

5.1.1 UMTS application

The transmitter of the uplink UMTS terminal is composed of 14 tasks and 13 edges. The UMTS application is under a real time constraint since the application has to be executed every 10 ms. It leads to apply some throughput constraints to different tasks: task1 - 114 kbits/s, task8 - 37 Mbits/s, task10 - 15 kbits/s and task14 - 8 Mbits/s. Concerning the specification of the application no other information is required and the analysis can be performed. To discuss the proposed metrics, we use a two-cluster solution for dividing the UMTS application tasks into clusters (i.e. PACM). The values presented in Table 1 correspond to intermediate results required to compute the metrics presented in Table 2. These intermediate values are presented in order to enable the discussion that follows about the metrics.

In Table 2 we present the metrics computed for the two solutions. In the following sections, we use these two solutions to discuss the validity of the proposed metrics. Also, we demonstrate the impact of these metrics on the architecture design.

5.1.2 Communication metrics validation

Fig. 2 shows the distribution of the data exchanged for both solutions. The E_De metric evaluates the distribution of the transfers into the different clusters. When equals to 1, a same quantity of data is exchanged within each cluster and when equals to 0 all the data is exchanged in only one cluster. In solution 1, the two clusters of the UMTS application have the same quantity of exchanged data. The metric

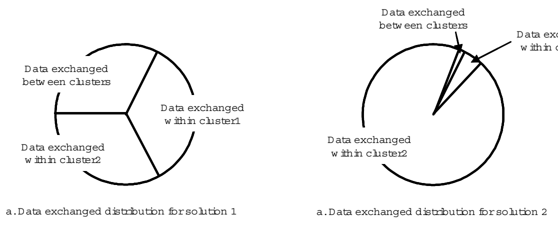

Figure 2. Distribution of the exchanged data into the architecture.

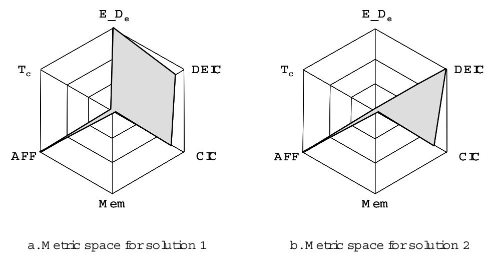

Figure 3. Metric space coverage for different solutions.

E_De is close to 1. In solution 2, 96% of data exchanged is in cluster2. The E_De metric highlights this point as it is close to 0. The amount of data exchanged between the two clusters in solution 1 is not very high. However it is greater than the data exchanged between the clusters in solution 2. The $DEIC$ metric for both solutions is close to 1 (Table 2). Using a classical architecture exploration tool, if the designer wants to find an architecture composed of two processors from the UMTS application, he has to define a communication architecture enabling the transfer of 200,000 bits. With our exploration method, one communication support is allocated for each cluster. In solution 1, each communication resource has to manage a maximum of 80,000 bits. As a consequence, the communication architecture will be faster and simpler to design.

5.1.3 Memory metric validation

In both solutions, the distribution of the memory is concentrated into one cluster (Table 1). Thus the Mem metric is close to 0. For such distributions, the architecture will use large memories which contain most of the data. This will strongly impact the performance of the system as access memory time will increase as well as power consumption.

5.1.4 Throughput constraint metric validation

Throughput constraint metric (T_c) for both solutions is close to 0 as throughput constraints are very different within each cluster. The communication architecture needs to be designed to manage the highest constraint even if most of the time this constraint is not faced. In Solution 1 the highest throughput is within cluster1 (38,400,000 bits/s). The lowest throughput constraint for the same cluster is $116,800$ bits/s. Such a clustering, from the throughput point of view, is far from optimal as the communication architecture as to manage too different constraints.

5.1.5 Metric space and its impact on the architecture

Fig. 3 shows the different metrics computed for the two solutions. The metrics of both solutions do not cover all the metric space. In solution 1 (Fig. 3.a), T_c and Mem metrics are close to 0. The analysis of these metrics is interesting to have a first idea of the performance and the cost of the architecture. In the case of solution 1 the architecture will need a large memory size which will impact the global performance. The communication architecture will have to face high fluctuating throughput which may lead to reliability problems. However the communications are well balanced within the architecture which is an important aspect. In solution 2 (Fig. 3.b), E_De, T_c and Mem are close to 0. This solution may lead to an unbalanced and inefficient architecture as many points are non-optimized. The metric space coverage provides the designer with first ideas of the quality of his solution. The goal of the genetic algorithm is to automatically perform the exploration in order to converge toward an efficient and optimized decomposition of the system.

5.2 Genetic algorithm validation

5.2.1 UMTS application

The UMTS transmitter has been designed and optimized by hand [13]. The targeted platform was based on Pentek boards composed of DSP and FPGA components. We have applied our approach to the UMTS application in order to evaluate the quality of the exploration process. During the exploration we have not considered the E_De metric as most of the communications within a cluster correspond to software/software communications (tasks executed on the DSP). The result of the pre-exploration step is illustrated in Fig. 4. The coverage of the metric space is quite good as any metric is penalized. The implementation proposed in [13] is the following one: the set of tasks $\{$1,2,3,4,5,6,7,9$\}$ are implemented onto a DSP, the set $\{$8,10,11,12,13$\}$ onto another DSP and the task 14 onto a FPGA. This partitioning corresponds to the one resulting from the pre-exploration step which indicates that our approach leads to good guidance for the designer. The pre-exploration step enables the designer to provide some constraints through the $Affinity$ $vector$ and to remove some metrics if not relevant for the exploration. Thus this tight interaction between the designer and the pre-exploration step enables to converge toward efficient solutions.

5.2.2 AC3 application

We have considered The AC3 application to analyze the benefit of using the pre-exploration step before performing

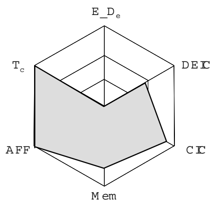

Figure 4. Metric space coverage for the UMTS solution.

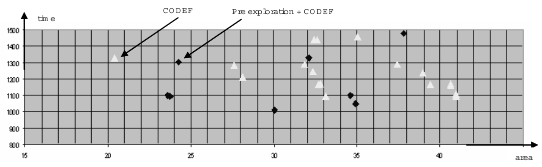

Figure 5. AC3 exploration space results.

a more traditional codesign flow. In order to perform the whole architecture exploration process, we have used the codesign CODEF tool [3]. The pre-exploration step aims at clustering the AC3 application into two clusters. Then the CODEF tool is used to perform the hardware/software partitioning. To handle the exploration with CODEF, a list of the tasks and their costs for all resources available in the library is needed. After the pre-exploration step, each task is associated to a cluster (i.e. PACM). Thus, each task can only be executed onto the processor or an hardware IP. So all other possibilities of execution (i.e. on other resources) are removed from the library used by CODEF. This reduces the architecture space to be explored by CODEF and speeds up the exploration.

The architectures found using the pre-exploration step and then CODEF are shown as rhombuses in Fig. 5. The solutions obtained when using only CODEF are shown as triangle in Fig. 5. Each solution is characterized by its performance in terms of area and time. CODEF goal is to optimize the area and to satisfy the time constraint. CODEF reaches its goal as the smaller solution is found by the tool. However, when using both pre-exploration and CODEF, more solutions which optimize the time or the time vs. area tradeoff are found. Several Pareto points are obtained using the combination of both approaches. Combining both steps allows designer to explore more rapidly the design space and to emphasize some interesting solutions that were not considered by CODEF.

6 Conclusion

The complexity of future applications and systems is raising a new design wall. Classical codesign flows will not be able to face this challenge in a near future as they will not be efficient enough to design MPSoC systems composed of hundreds of processors. It is mandatory to extend current approaches, as it has always been performed in the past, by rising the level of abstraction. We believe that an earlier, fast and automatic pre-exploration step is essential to identify the application's characteristics in order to provide some guidance to codesign flows. To perform such an approach, we have extended exploration methods by preceding them with a new step which analyzes the application. Our new pre-exploration step simplifies current exploration methodologies.

References

[1] Borkar Shekhar, Jouppi Norman, P. Stenstrom Per *"Microprocessors in the Era of Terascale Integration"* Design, Automation and Test in Europe Conference and Exhibition, 2007. DATE '07, pages 1 - 6, April 2007.

[2] K.Lahiri, A.Raghunathan, S.Dey, *"System-Level Performance Analysis for Designing On-Chip Communication Architectures"*, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no.6, pp.768-783, June 2001.

[3] M. Auguin, L. Capella, F. Cuesta, et E. Gresset. *"CODEF: a System Level Design Space Exploration Tool"* ICASSP, pages 1031-1034, Salt Lake City, USA, Mai 2001.

[4] Peter Voigt Knudsen and Jan Madsen. *"Integrating communication protocol selection with hardware/software codesign."* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pages 1077-1095, August 1999.

[5] Baghdadi, A.; Zergainoh, N.-E.; Cesario, W.O.; Jerraya, A.A.; *" Combining a performance estimation methodology with a hardware/software codesign flow supporting multiprocessor systems"* Software Engineering, IEEE Transactions on Volume 28, Issue 9, Pages 822 - 831, Sept. 2002.

[6] Wieferink, A.; Kogel, T.; Leupers, R.; Ascheid, G.; Meyr, H.; Braun, G.; Nohl, A.; *"A system level processor/communication co-exploration methodology for multi-processor system-on-chip platforms"* Design Automation and Test in Europe Conference and Exhibition, Pages 1256 - 1261, Proceedings Volume 2, 2004.

[7] B.P.Dave, G.Lakshminarayana and N.K. Jha, *"COSYN : Hardware software Co-synthesis of heterogeneous distributed embedded systems"*, In IEEE Transaction on software Engineering, volume 7, mars 1999.

[8] D.Sciuto, F.Salice, L.Pomante and W.Fornaciari, *"Metrics for Design Space Exploration of Heterogeneous Multiprocessor Embedded Systems"*, CODES'02, Estes Park, USA, May 2002.

[9] Yannick Le Moullec, Nader Ben Amor, Jean-Philippe Diguet, Mohamed Abid and Jean-Luc Philippe *" Multi-Granularity Metrics for the Era of Strongly Personalized SOCs"* Design, Automation and Test in Europe Conference (DATE 03), Munich, Allemagne, 3-7 Mars 2003.

[10] Issam Maalej, Guy Gogniat, Mohamed Abid, Jean Luc Philippe, *"Metrics for a high level analysis of a multiprocessor system on chip"* Embedded Real-Time Systems Implementation Workshop(ERTSI 2004), Lisbon, Portugal December 2004.

[11] Universal Mobile Telecommunication System, 2002, [Online]. Available: http://www.umtsworld.com/

[12] Advanced Television System Committee. Digital Audio Compression Standard (AC3), 1995.

[13] Christophe Moy, Apostolos Kountouris, Luc Rambaud,Pascal Lecorre, *"Full Digital IF UMTS Transceiver for Future Software Radio Systems"* ERSA'2001, Las Vegas, USA, 25-28 June 2001 (first international conference on Engineering of Reconfigurable Systems and Algorithms).