Correct and Energy-Efficient Design of a Multimedia Application on SoCs

Adolf Abdallah, Abdoulaye Gamatié, Rabie Ben Atitallah, Jean-Luc Dekeyser

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Correct and Energy-Efficient Design of a Multimedia Application on SoCs

Adolf Abdallah — Abdoulaye Gamatié — Rabie Ben Atitallah — Jean-Luc Dekeyser

N° 7715

August 2011

Embedded and Real Time Systems

ISSN 0249-6399 ISRN INRIA/RR--7715--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche LILLE - NORD EUROPE

Correct and Energy-Efficient Design of a Multimedia Application on SoCs

Adolf Abdallah*, Abdoulaye Gamatié†, Rabie Ben Atitallah‡, Jean-Luc Dekeyser§

Theme : Embedded and Real Time Systems
Algorithmics, Programming, Software and Architecture
Équipes-Projets Dart

Rapport de recherche n° 7715 — August 2011 — 16 pages

Abstract: This report presents the design and analysis of multimedia applications such as the JPEG encoder on multiprocessor architectures. A model-based approach is adopted by using the UML Marte specifications. An abstract clock analysis is proposed to deal with the correctness of system behaviors and to find the most suitable execution platform configurations regarding performance and energy consumption. We claim that our approach offers a rapid and reliable design space analysis, which is crucial when implementing complex systems.

Key-words: Multimedia applications, abstract clock, system-on-chip, configuration analysis, correctness

* adolf.abdallah@lifl.fr
† abdoulaye.gamatie@lifl.fr
‡ rabie.benatitallah@univ-valenciennes.fr, LAMIH - University of Valenciennes, Le Mont Houy, 59313 Valenciennes - France
§ jean-luc.dekeyser@lifl.fr

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne
40, avenue Halley, 59650 Villeneuve d'Ascq
Téléphone : +33 3 59 57 78 00 — Télécopie : +33 3 59 57 78 50

Conception correcte et efficace du point de vue énergétique d'une applications multimédia sur système-sur-puce

Résumé : Ce rapport présente la conception et l'analyse d'applications multimédia telles qu'un encodeur JPEG sur une architecture multiprocesseur. Une approche basée sur des modèles est adoptée en considérant des spécifications en UML Marte. Une analyse reposant sur des horloges abstraites est proposée afin d'aborder la correction des comportements d'un système, et trouver les configurations les plus adéquates pour son exécution, du point de vue de la performance et de la consommation d'énergie. Notre approche permet une analyse rapide et fiable de l'espace de conception ; cela est crucial pour l'implantation des systèmes complexes.

Mots-clés : Applications multimédia, horloges abstraites, système-sur-puce, analyse de configuration, correction

Contents

1 Introduction

Multimedia embedded systems implemented on system-on-chip (SoCs), e.g., found in digital cameras and cellular phones, are omnipresent in our daily life. They become increasingly sophisticated and resource demanding to meet quality of service. Their developers must guarantee their correctness and satisfactory execution performances. Multiprocessor platforms offer a very good opportunity to efficiently implement such systems. In addition, energy consumption is another major issue when such systems are embedded in portable devices depending strongly on batteries.

All these aspects make the efficient design of multimedia SoCs very challenging. To adequately address the aforementioned design challenges, developers need methodologies for high-level software/hardware co-modeling and analysis of SoCs. The considered models must be expressive and precise enough for an early assessment of possible design decisions.

Contributions. In this paper, we propose an approach for embedded system design and analysis at a high abstraction level in order to reduce the design space exploration (DSE) efforts in a SoC codesign framework [9]. We describe a system implementing the JPEG encoder with the standard UML Marte profile [15], dedicated to the *Modeling and Analysis of Real-Time Embedded systems*. We analyze the behavior of the system by considering abstract clocks inspired by the synchronous approach [5]. Our clock-based analysis aims to guarantee the functional correctness of the system. It also exploits clock traces to assist a designer in a choice of system configurations offering a good compromise about correctness, performance and energy consumption. Our approach is typically very useful in frameworks adopting platform-based design [16], where high level specifications of system functionality and architecture are refined with well-characterized components and analyzed so as to rapidly meet design requirements.

Related works. Among existing studies devoted to model-based design and analysis of embedded systems, we mention ModES [14], Sesame [11] and synchronous modeling of AADL [10]. In ModES, the authors use UML to model a system. An internal tool (called H-Spex), adopting a probabilistic metaheuristic approach, provides designers with a number of processors required for system execution, maps tasks to processors, allocates processors to bus segments and sets the voltage of each processor. Its main goal is to minimize as much as possible the communication costs and the voltage of processors. In Sesame, authors address SoC design by using a specific UML profile. They generate code in two languages: SystemC for simulation and Lotos for formal analysis of temporal ordering between events. In [10], AADL models describing an embedded system architecture are translated to a synchronous language in order to verify the preservation of the system functional properties. The result is combined with an associated synchronous program encoding the system functionality in order to obtain a complete system model for validation. Compared to above mentioned works, our proposition aims at offering a SoC co-design and analysis methodology. We specify a system using Marte as in [12]. We propose a validation of system functional properties as in [10] and [11]. This enables to

guarantee the suitability of chosen execution platforms regarding the correctness of the system. We also address execution platform properties via processor frequencies so as to find relevant system configurations optimizing the energy consumption as in [14]. For that, we make use of well-known techniques in literature such as *dynamic voltage/frequency scaling* (DVFS) [6] or *power shutdown* (PS) [13].

Beyond model-based approaches, we can also mention existing commercial tools that provide RTL (register transfer level) simulation and emulation environments for low-level system prototyping [3]. Such environments assist software developers in driver debug and integration, and allow hardware engineers to keep their traditional view of embedded systems. Unfortunately, RTL level tools cannot adequately support the complexity of multi-processor SoCs required for multimedia applications because they are very slow for a meaningful execution of application software. In order to reduce simulation time, many research efforts have been conducted towards the use of Cycle-Accurate (CA) simulators. They simulate a micro-architecture at the clock-cycle level and provide widely used simulators. At a higher abstraction level, an Instruction Set Simulator (ISS) sequentially executes instructions without any notion of concurrency of a micro-architecture. An ISS description can be enhanced with timing annotations for approximating the execution time and obtaining a behavioral model. But, the simulation speed with CA or behavioral ISS simulators are limited to few hundred thousands of simulated cycles per second. Contrarily to RTL- and CA-based techniques, our approach does not require any simulation platform. In addition, only an abstract characterization of the hardware and software parts of a system is required. As a result, no actual IP is needed to run and analyze a system, and this eliminates the tedious coding/debug efforts from the problem.

Outline. In the following, Section 2 describes system design with Marte and abstract clocks. Section 3 presents our clock-based analysis. Section 4 reports experimental results on a JPEG encoder by comparing our clock-based approach with SystemC simulation. Finally, Section 5 gives conclusions.

2 System design with Marte

In this study, sequences of images (of size 256×256) are encoded at a constant frame rate of 15 frames per second. So, the JPEG algorithm processes a frame every 66 ms. In order to encode each frame before the next one, a deadline of 66 ms is chosen. We consider finite image sequences.

2.1 Functional part

The JPEG encoder algorithm, modeled on top of Fig. 1, consists of several steps. First, it splits images into blocks of size 64 pixels (8×8). Here, we consider images of type luminance/chrominance such as YUV and YCbCr, to obtain the best compression ratios. Then, a color transformation is applied to pixel blocks in order to extract information about the luminance and chrominance. Afterward, a *Discrete Cosine Transformation* (DCT) is applied to each resulting

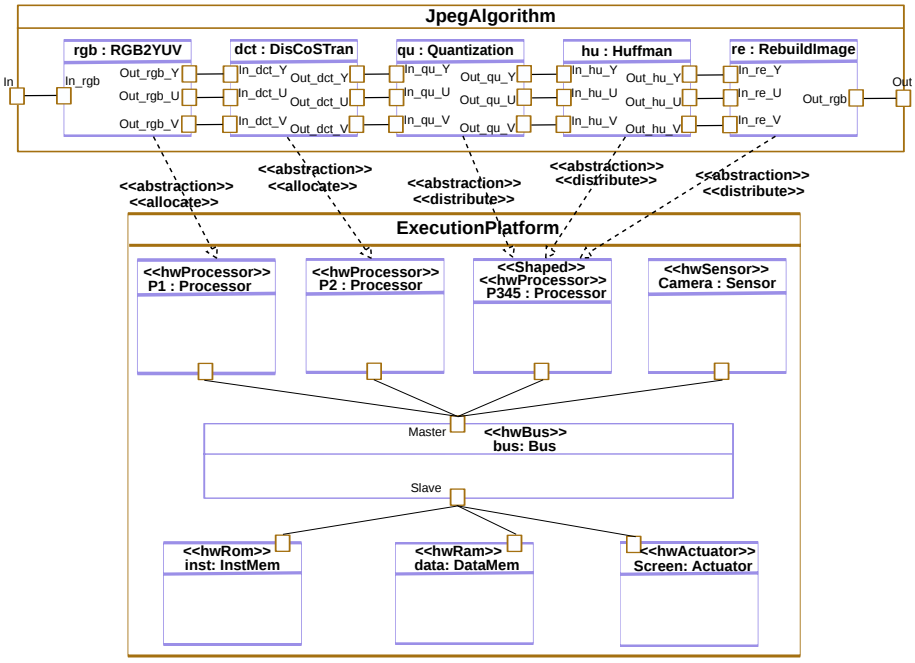

Figure 1: Model of the JPEG system in Marte.

block to extract information in terms of frequency and amplitude. A quantification is applied to the result. It produces matrices of size 8×8 to which a Huffman coding is applied. Finally, compressed images are built from transformed pixel matrices.

In Fig. 1, the components (or tasks) `rgb`, `dct`, `qu`, `hu` and `re` represent respectively color transformation, discrete cosine transformation, quantization, huffman and finally the image reconstruction. Furthermore, we define activation rate relations between these components by using an abstract clock notation [7]. Each component is associated with a *finite* clock *clk*, defined as a sequence of logical instant values: an occurrence of 1 means the component is active and is *simultaneously*[1] *consuming, executing and producing data* (i.e. synchrony hypothesis [5]), while 0 means no activation.

clk_{rgb}:	1	1	1	1	1	...	0	0	0	0
clk_{dct}:	0	1	1	1	1	...	1	0	0	0
clk_{hu}:	0	0	1	1	1	...	1	1	0	0
clk_{qu}:	0	0	0	1	1	...	1	1	1	0
clk_{re}:	0	0	0	0	1	...	1	1	1	1

Figure 2: Trace of functional clocks associated with JPEG tasks.

According to a pipelined execution of the JPEG algorithm, the activations of `dct` (resp. `qu`, `hu` and `re`) are preceded by activations of `rgb` (resp. `dct`, `qu`

[1] Actually, an active logical instant corresponds to several processor cycles, i.e., it is not "instantaneous". In our approach, this number is determined during the mapping of functionality on a platform.

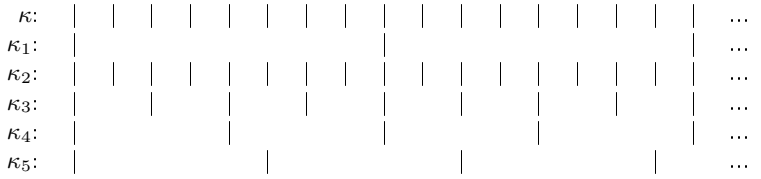

Figure 3: Trace of physical clocks associated with processors ($f_1 = 25MHz, f_2 = 200MHz, f_3 = 100MHz, f_4 = 50MHz$ and $f_5 = 40MHz$.

and `hu`). This is described in Fig. 2, where the *functional* clocks clk_{rgb}, clk_{dct}, clk_{hu}, clk_{qu} and clk_{re} are associated respectively with `rgb`, `dct`, `hu`, `qu` and `re`. The trace reflects the precedence relations about components activations. Note that more complex execution models can be easily captured by such traces [2, 1]. In the rest of the paper, we refer to such relations as *functional clock properties* of the JPEG encoder.

2.2 Execution platform

We consider a *parallel random access machine* (PRAM) architecture model [8] to execute the JPEG encoder. PRAM is composed of processors operating synchronously according to a global clock and communicating via a shared memory. Processors are given the *concurrent read concurrent write* semantics for memory access (this may induce access conflicts). In Marte, the *Hardware Resource Modeling* (HRM) package offers concepts to model hardware resources as illustrated in in the bottom of Fig. 1, e.g. the stereotypes `hwProcessor` and `hwRam` denote processor and memory. Note that Marte also provides a compact representation of parallelism (in both application and hardware parts) via the `Shaped` stereotype, which means that the associated entity consists of multiple instances. For instance, in Fig. 1, the component `P345` is formed of three processor instances.

So, we consider five processors P_1, P_2, P_3, P_4 and P_5 (the last three form `P345`), with the frequencies $f_1 = 25MHz, f_2 = 200MHz, f_3 = 100MHz, f_4 = 50MHz$ and $f_5 = 40MHz$ respectively. If $1/f$ denotes the period of a processor with frequency f, we have $0.04\mu s, 0.005\mu s, 0.01\mu s, 0.02\mu s$ and $0.025\mu s$ as period values for P_1, P_2, P_3, P_4 and P_5 respectively. We use these values to define the activation instants of each processor. Furthermore, for synchronization purpose, we need to consider a reference (or ideal) clock κ that provides a common time base. We define the period of κ as $1/LCM(f_1, ..., f_5) = 0.005\mu s$, where LCM is the Least Common Multiple. Fig. 3 depicts the periodic activation rates between processors P_1, P_2, P_3, P_4 and P_5 according to their periods and the reference clock κ. We refer to these activations as *physical* clocks κ_j of processors.

2.3 Mapping and scheduling of functionality on platform

2.3.1 Modeling of allocation

For the execution of tasks `rgb`, `dct`, `qu`, `hu` and `re` specified in the functional part, we model and analyze several possible mapping scenarios onto five processors P_1, P_2, P_3, P_4 and P_5. The *Alloc* package of Marte provides one with adequate concepts for specifying the considered tasks allocations on processors.

In Fig. 1, an example of mapping is shown where `rgb` and `dct` are respectively associated with `P1` and `P2` by using the `allocate` connector. The tasks `qu`, `hu` and `re` are distributed onto the remaining three processors `P3`, `P4` and `P5` by using the `distribute` connector whose attributes (not shown here) exactly state how each task is allocated.

Furthermore, the number of processor cycles corresponding to task activations needs to be specified statically during the mapping: we assume that *each activation of a task takes c cycles.*

To capture mapping and scheduling choices with abstract clocks, we define a projection of functional clocks on physical clocks. Then, we use this projection to describe the scheduling of mapped tasks on processors.

2.3.2 General principle: clock projection

Let us assume the number of instants in a functional clock clk_i, i.e. its length, is smaller to that of a physical clock κ_j. The mapping (or projection) of clk_i on κ_j is defined as follows:

1. for each $k \in [1, length(clk_i)]$, the status (i.e., 0 or 1) of the k^{th} instant of clk_i is mapped onto the j^{th} instant of κ_j, where $j = (nb\_occ(clk_i, k, 0) + (nb\_occ(clk_i, k, 1) \times c) + 1)$. The expression $nb\_occ(clk_i, k, 0)$ returns the number of instants with the status 0 in clk_i that have occurred before the k^{th} instant whereas $nb\_occ(clk_i, k, 1)$ concerns instants with the status 1. The constant c is the number of processor cycles performed by an activation.

   Let $L = length(clk_i)$, every k^{th} instant of κ_j s.t. $(k > nb\_occ(clk_i, L, 0) + nb\_occ(clk_i, L, 0) \times c)$ is associated with the status value 0. Let us call clk'_i the clock resulting from this step;

2. in clk'_i, the value -1 is inserted at all empty positions where the reference clock κ has an instant occurrence while clk'_i has not any instant occurrence. Also, $c-1$ values of -1 are inserted after each occurrence of a 1 to delimit a whole activation. After this step, clk'_i and κ have the same length.

From the above clock projection, the occurrence of 1 at an instant in a clock clk'_i indicates that the processor corresponding to κ_i is *active* and is executing one instruction cycle at that instant. The value 0 indicates that the processor is in the *Nop* state. The meaning of -1 is rather contextual: when a sequence of such a value is immediately preceded by 1, then it denotes *potentially active at those instants*; otherwise, it denotes *idle*.

2.3.3 Scheduling of tasks on processors

We distinguish several possible task mapping scenarios on an execution platform:

- *Mono-task scheduling on each processor.* Each processor executes one task (itself executed only on this processor). For each task associated with a functional clock clk_i, its scheduling on a processor associated with a physical clock κ_j is captured by the clock projection defined previously, according to a number of cycles corresponding to each task activation. An illustration is given in Fig. 4(a) where we map `rgb`, `dct`, `qu`, `hu` and `re` onto

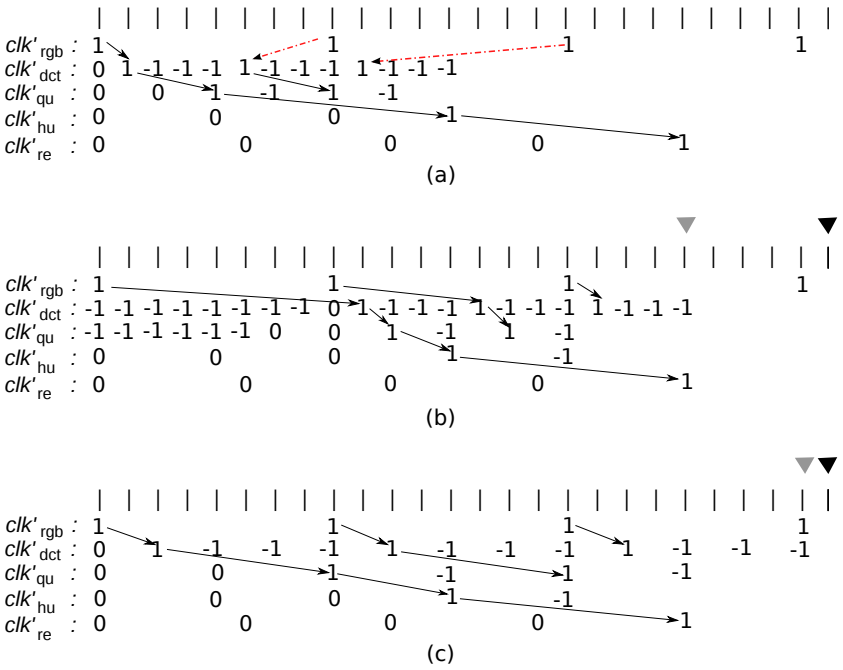

Figure 4: Trace sketch of clocks clk'_{rgb}, clk'_{dct}, clk'_{qu}, clk'_{hu} and clk'_{re} resulting from a mono-task scheduling on each processor (cycles per task activation: $c_{rgb} = 1$, $c_{dct} = 4$, $c_{qu} = 2$, $c_{hu} = 3$ and $c_{re} = 3$), with: (a) $f_1 = 25MHz, f_2 = 200MHz, f_3 = 100MHz, f_4 = 50MHz$ and $f_5 = 40MHz$ (precedence constraints violated); (b) $f_1 = 25MHz, f_2 = 200MHz, f_3 = 100MHz, f_4 = 50MHz$ and $f_5 = 40MHz$ with a delay of eight logical instants of the ideal clock for clk'_{dct} and six logical instants for clk'_{qu} (precedence constraints satisfied) and (c) $f_1 = 25MHz, f_2 = 100MHz, f_3 = 50MHz, f_4 = 50MHz$ and $f_5 = 40MHz$ (precedence constraints satisfied).

processors P_1, P_2, P_3, P_4 and P_5 respectively. For the sake of simplicity, the very simple values $c_{rgb} = 1$, $c_{dct} = 4$, $c_{qu} = 2$, $c_{hu} = 3$ and $c_{re} = 3$ are associated with `rgb`, `dct`, `qu`, `hu` and `re` respectively. The arrows represent the activation precedences specified in the functional part after the scheduling is applied.

- *Multi-task scheduling on processors.* Here, a processor can execute several tasks. To define such a scheduling, we apply a time slicing to the functional clocks clk_i of all tasks to be executed on a processor with a physical clock κ_j. We partition every clock clk_i into smaller *slices* (sub-sequence of clk_i) with equal sizes, e.g., a slice s can be a period in the binary notation of clk_i. The scheduling of s is defined as the clock projection of s on κ_j. Then, we define the scheduling of the whole tasks as follows: the processor repeatedly switches from the scheduling of a task to another after the scheduling of every slice. This is performed until all slices of all tasks are scheduled.

Beyond the above two schedulings, one may also need to *schedule either multiple tasks on multiple processors, or one task on several processors* (useful for a parallel execution of a task). These problems are solved by using the same kind of reasoning as above.

In [12], Marte is used for the co-modeling of SoCs and system-on-programmable components. Three abstraction levels are defined with Marte in order to describe SoC architectures: an *abstract platform architecture* consisting of an untimed system model, an *execution platform architecture* refining the untimed model with timing constraints and separating software from hardware components, and a *detailed platform architecture* including further details such as power consumption. While this separation of concerns is interesting, authors do not address at all analysis issues, necessary for system validation. The next section deals with a clock-based system analysis.

3 Clock-based system analysis

We assess the design choices resulting from the previous section, w.r.t. the correctness of functional clock properties, performance and energy consumption.

3.1 Analysis of functional clock properties

It is important to preserve the functional clock relations when synthesizing a scheduling of a system after a mapping. Typically, a data consuming component must not be executed without having received its required data. To check the precedence imposed on activations of `rgb`, `dct`, `qu`, `hu` and `re` tasks, we analyze the clock trace in Fig. 4 (a). We can deduce that the clock properties are not respected all. The constraint between clk'_{rgb} and clk'_{dct} is violated since the second (resp. the third) activation instant in clk'_{rgb} is preceded by the second (resp. the third) activation instant in clk'_{rgb}. In other words, processor P_2 is activated very frequently while processor P_1 has not produced yet the data required by P_2 for processing.

To solve the above issue, we distinguish different solutions. The first solution consists in delaying the activation of the fastest clock so as to postpone the execution of its associated processor. This solution works here because the analysis is done over finite image sequences, hence with finite clocks. With infinite clocks, we have to reason on a macro-period over considered clocks, i.e., on a repetitive finite portion of an infinite clock trace. In Fig. 4 (b), by inserting eight logical instants of the reference clock in clk'_{dct} and six instants in clk'_{qu}, the activation precedences become correct. The black and gray triangles respectively indicate deadline and specific response times.

Another solution consists in modifying the frequency in order to satisfy the functional clock properties. Let us reduce the frequency of P_2 from $200MHz$ to $100MHz$. We obtain the trace in Fig. 4 (c) where the functional clock properties are satisfied between clocks clk'_{rgb} and clk'_{dct}. An important benefit of the new processor configuration is the subsequent reduction of power consumption since P_2 operates at a lower frequency while ensuring a correct functional behavior and deadline constraint.

3.2 Evaluation of execution time

To compute the total execution time of the system for performance evaluation, we have to determine the amount of computational workload of each processor. If a given processor has a frequency f and executes C activation cycles during a complete execution of an application, then its corresponding execution time is given by: $\frac{C}{f}$.

In Fig. 4 (c), it takes 9 cycles in clk'_{dct} to execute the first two activations. Hence, the corresponding execution time by processor P_2 at $100MHz$ is $\frac{9}{100} = 0.09\mu s$.

3.3 Minimizing energy consumption

During an execution, we define the *slack time* of a task as the difference between its completion time and its associated deadline. In Fig. 4(b), the deadline value is pointed out by the black triangle on the 26^{th} instant of the reference clock. The response time of `rgb` task is pointed out by the gray triangle on the 21^{th} instant. In Fig. 4(c), this response time occurs at the 25^{th} instant after a reduction of the associated processor frequency. As a result, the slack time is shorter, which also reduces the energy consumption, while the functional properties are still verified. This reasoning is applied statically on the set of all correct traces obtained from our clock-based scheduling.

Another way to estimate the energy consumption E of a system in our framework is to consider the execution time T calculated previously with power consumption W information of every task in the JPEG, obtained from a profiling on given experimental platforms: $E = T \times W$.

4 Validation on JPEG encoder: clock-based approach vs SystemC simulation

To validate our clock-based design and analysis, we compare our results with those obtained with the cycle-accurate (CA) simulation in SystemC. CA simulation is largely used in industry and is provides good performance accuracy. However, it is time consuming. We use the SoCLib library [17], which provides an MPSoC simulation environment at cycle accurate level. We executed the JPEG application according to the mentioned configurations.

4.1 Configuration specification

We consider nine mapping configurations of the JPEG encoder, summarized in Table 1. In the first five configurations (i.e., configurations 1, 2, 3, 4 and 5), the JPEG tasks are distributed according to the number of processors corresponding to a pipelined execution model. For the configurations 6, 7, 8, and 9, processors carry out the same algorithm of JPEG encoder on different image blocs corresponding to single program multiple data (SPMD) execution model.

In addition, Table 2 gives pre-profiling data about the JPEG, used as input information for our clock-based approach. It shows measures of processor cycles and power consumption per task activation for one image, on a RISC processor

Configurations ID	Number of Proc.	Allocation type
1	1	task/processor
2	2	task/processor
3	3	task/processor
4	4	task/processor
5	5	task/processor
6	2	sub-image/processor
7	3	sub-image/processor
8	4	sub-image/processor
9	5	sub-image/processor

Table 1: Mapping configurations for the JPEG encoder.

Task	# Cycles	Power (mW)
rgb	7534	3741
dct	7214	3776
qu	6920	3858
hu	5504	3675
re	4594	3653

Table 2: Cycles and power consumption per task activation in JPEG.

of type PowerPC 405 CPU Core, at 300MHz (using the Power Analyzer N6705A of Agilent).

4.2 Results

Fig. 5 shows the experimental results of an evaluation of execution time throughout the clock based analysis and CA simulations in SystemC. The results obtained from the clock-based analysis keep the same tendency as those observed in SystemC. This provides a designer with a consistent basis to assess the different configurations.

We consider a rate of 15 frames per second for the encoding, a duration of $1/15 \approx 0.66ms$ is obtained for encoding each image. Let us consider this duration as an image processing deadline. Provided a mapping configuration where the precedence relations on task activations are satisfied and the processing deadline is met, we can determine the minimal frequencies based on the clock-based analysis so as to reduce the slack time much as possible. Those frequencies contribute to reducing the energy consumption. They are indicated in Table 3 for configurations 1, 6, 7, 8 and 9.

Beyond the above qualitative reasoning about the energy minimization by shortening the slack time, we can also estimate quantitatively the energy consumption. For that purpose, given the execution time evaluated for each task with the clock-based approach, we compute the energy dissipation in configurations 1, 6, 7, 8 and 9 as shown in Fig. 6. This is achieved by accumulating the energy consumption of all tasks via the product of their corresponding predetermined power consumption (see Table 2) and execution times. We observe again that our approach leads to the same tendency as the considered CA simulation.

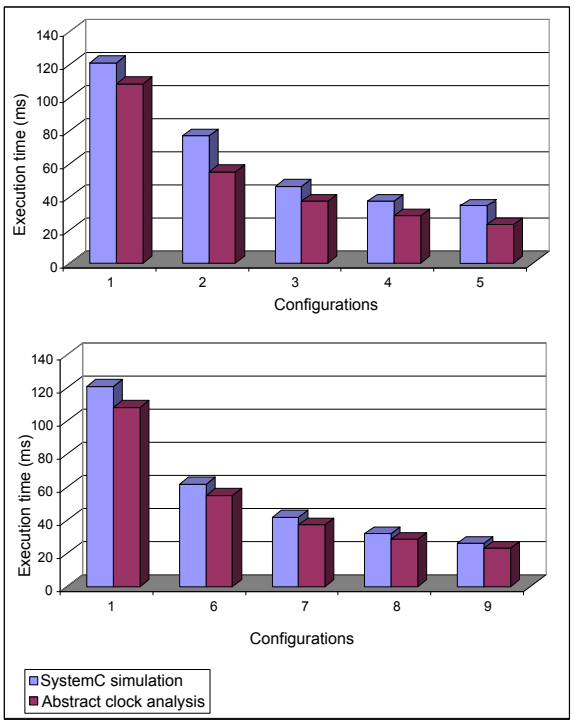

Figure 5: Comparison of execution time according to the two approaches.

Processors	**Processor cycles**	**Minimal frequency (MHz)**
Proc1	32529428	493
Proc1	16490607	250
Proc2	16604443	252
Proc1	14668654	223
Proc2	12622744	192
Proc3	6369273	97
Proc1	8497715	129
Proc2	8556568	130
Proc3	8556568	130
Proc4	8615442	131
Proc1	8638148	131
Proc2	6523480	99
Proc3	6523480	99
Proc4	6523480	99
Proc5	6583327	100

Table 3: Frequency values for configurations 1, 6, 7, 8 and 9.

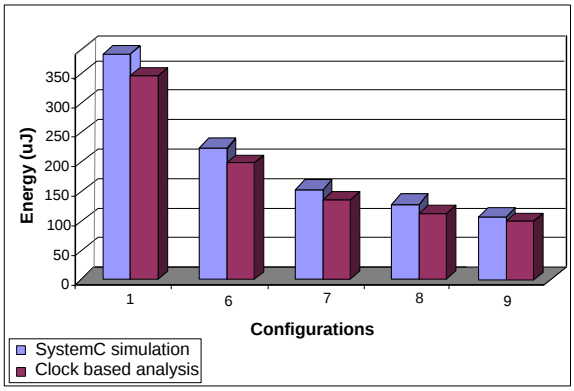

Figure 6: Comparison of energy consumption to the two approaches.

Concerning the accuracy of our clock-based approach compared to CA simulation (which is more precise), the maximum estimation error is of 28% for execution time (Fig. 5) and of 11.6% for energy consumption (Fig. 6). Indeed, this difference is explained by the fact that the clock-based analysis does not fully take into account synchronizations overhead in multiprocessor system as well as additional activities that are intrinsic to parallel processing such as shared data communication overheads which are accurately evaluated with SystemC simulator.

On the other hand, since the SystemC implementation and CA simulation requires a significant time (about one week to achieve all experiments) compared to the clock-based technique (half a day), the latter approach is therefore preferable for a preliminary rapid exploration of large design spaces. Finally, our clock-based analysis is applicable modularly to any periodic clock trace independently from the number of clocks in the trace. Since multimedia applications have generally regular (repetitive) behaviors, they can be characterized with periodic clock traces. In that sense, our proposition is scalable.

Finally, we can also notice that the presented work contributes to bridging the gap existing between on the one hand, the functional specification of a system where computations and communications are considered as synchronous, and on the other hand, the corresponding implementation on an asynchronous platform. Compared to existing works on the same topic [4], we target an explicit asynchronous hardware platform model from which useful information such as processor characteristics allow us to address both functional correctness, performance and energy issues.

5 Concluding remarks

This paper illustrated the design and analysis of a JPEG encoder on a multiprocessor hardware architecture. Starting from a high-level modeling with the standard UML Marte profile, we proposed an analysis approach by considering abstract clocks inspired by the synchronous approach [5], for a fast reasoning about functional correctness and best design choices regarding temporal performance and energy consumption. In particular, we addressed the

energy consumption by reasoning on clock traces expressing correct execution of system functionality on given architecture configurations. We were able to identify minimal processor frequencies in such configurations in order to save energy. We also estimated the energy consumption for some software/hardware mapping configurations. While the clock-based reasoning is less accurate than a cycle accurate simulation in SystemC, it provides very similar observations in a faster way.

To achieve our clock-based analysis, we combined the TimeSquare[2] clock simulator, and a few abstract clock comparison functions encoded in Ocaml [1]. However, a more complete and seamless implementation of the approach remains to be done.

As future work, we plan to improve the presented clock-based modeling by refining it with an estimation of synchronization overhead in multiprocessor system and shared data communication overhead at high abstraction level. The aim is to make the analysis more accurate in order to reduce the gap between our results and those obtained with lower level simulation approaches.

References

[1] Adolf Samir Abdallah. *Conception de SoC à Base d'Horloges Abstraites : Vers l'Exploration d'Architectures en MARTE.* These, Université des Sciences et Technologie de Lille - Lille I, March 2011.

[2] Xin An, Eric Rutten, and Abdoulaye Gamatié. Safe design of dynamically reconfigurable embedded systems. In *2nd Workshop on Model Based Engineering for Embedded Systems Design (M-BED'2011).* ECSI, 2011.

[3] A. Piziali B. Bailey, G. Martin. *ESL Design and Verification: A Prescription for Electronic System Level Methodology.* Elsevier Morgan Kaufmann, 2007.

[4] Albert Benveniste, Benoit Caillaud, and Paul Le Guernic. From synchrony to asynchrony. In *International Conference on Concurrency Theory (CONCUR'99)*, pages 162–177, London, UK, 1999. Springer-Verlag.

[5] Albert Benveniste, Paul Caspi, Stephen Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone. The synchronous languages twelve years later. *Proc. of IEEE*, 91(1):64–83, Jan. 2003.

[6] Zhen Cao, Brian Foo, Lei He, and Mihaela van der Schaar. Optimality and improvement of dynamic voltage scaling algorithms for multimedia applications. In *Design Automation Conference (DAC'08)*, pages 179–184, Anaheim, CA, USA, 2008.

[7] Albert Cohen, Marc Duranton, Christine Eisenbeis, Claire Pagetti, Florence Plateau, and Marc Pouzet. N-sychronous Kahn networks. In *ACM Symp. on Principles of Programming Languages (PoPL'06)*, Charleston, South Carolina, USA, Jan. 2006.

[2] `http://www-sop.inria.fr/aoste/?r=9&s=30&`

[8] Martti Forsell. On the performance and cost of some PRAM models on CMP hardware. In *International Parallel and Distributed Processing Symp. (IPDPS'08)*, pages 1–8, Miami, Florida, USA, 2008.

[9] A. Gamatié, S. Le Beux, É. Piel, A. Etien, R. Ben-Atitallah, P. Marquet, and J.-L. Dekeyser. A model driven design framework for high performance embedded systems. Research Report 6614, INRIA, 2008. http://hal.inria.fr/inria-00311115/en.

[10] E. Jahier, N. Halbwachs, P. Raymond, X. Nicollin, and D. Lesens. Virtual execution of AADL models via a translation into synchronous programs. In *Proc. of the 7th ACM & IEEE Conf. on Embedded software(EMSOFT'07)*, pages 134–143, New York, USA, 2007.

[11] Daniel Knorreck, Ludovic Apvrille, and Renaud Pacalet. Fast simulation techniques for design space exploration. In *47th International Conf. TOOLS EUROPE*, pages 308–327, Zurich, Switzerland, 2009.

[12] A. Koudri, D. Aulagnier, D. Vojtisek, P. Soulard, C. Moy, J. Champeau, J. Vidal, and J.C. Le Lann. Using MARTE in a Co-Design Methodology. In *Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile workshop co-located with DATE'08*, Munich, Germany, march 2008.

[13] S. P. Mohanty, N. Ranganathan, E. Kougianos, and P. Patra. *Low-Power High-Level Synthesis for Nanoscale CMOS Circuits*, chapter Power Reduction Fundamentals, pages 131–162. Springer, 2008.

[14] Marcio F. S. Oliveira, Eduardo W. Brião, Francisco A. Nascimento, and Flávio R. Wagner. Model driven engineering for MPSOC design space exploration. In *Proc. of the 20th Annual Conf. on Integrated circuits and systems design (SBCCI'07)*, pages 81–86, Copacabana, Rio de Janeiro, 2007.

[15] OMG. MARTE Web Site, 2009. `www.omgmarte.org`.

[16] Alberto Sangiovanni-Vincentelli, Luca Carloni, Fernando De Bernardinis, and Marco Sgroi. Benefits and challenges for platform-based design. In *41st annual Design Automation Conference (DAC'04)*, 2004.

[17] The SoCLib project: An open modelling and simulation platform for system on chip design. `http://soclib.lip6.fr/`.

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399