Dynamic Slack Reclamation Strategy for Multiprocessor Systems

Imen MHEDHBI*†, Rabie BEN ATITALLAH†and Abderrazak JEMAI‡§
*Ecole Polytechnique de Tunisie B.P. 743 - 2078 La Marsa, Tunisie
† LAMIH, Université de Valenciennes et du Hainaut Cambrésis
‡ Institut National des Sciences Appliquées et de Technologie, 1080 Tunis cedex, Tunisie
§Laboratoire LIP2-Facult des Sciences de Tunis, 2092 Manar 2 Tunis, Tunisie

***Abstract*—Emerging trends in applications with the requirement of considerable computational performance and decreasing time-to-market have urged the need of multiprocessor systems. With the increase in number of processor there is an increased demand to efficiently control the energy and power budget of such embedded systems as well. Dynamic voltage frequency scaling (DVFS) strategies permits to control this budget by actively changing the power consumption profile of the system. These techniques exploit the execution times variation of Real-time applications for dynamically adjusting the voltage and frequency of processors in order to reduce power and energy consumption. This paper presents one DVFS strategy called Dynamic slack reclamation (DSR) for real time applications. It is based on detecting early completion of tasks. DSR determines the amount of dynamic slack (if any) by comparing the worst-case execution requirement of a tasks job with its actual execution requirements. Experimental results will be extracted from the simulation of MPSoC architectures using the SoCLib platform; they show that DSR approach gives better power consumption performance.**

***Index Terms*—DVFS, DSR, MPSoC, energy and power optimization.**

I. Introduction

Modern embedded multimedia applications are becoming more and more sophisticated and resource demanding. Examples of the concerned applications are numerous such as video encoding/decoding, computer games, interactive TV, etc. The computation requirements of such systems are very important in order to meet real-time constraints and high quality of services. At the same time, the recent advances in silicon technologies offer a tremendous number of transistors integrated on a single chip. For this reasons, embedded hardware designers are directed more and more towards parallel and symmetric Multiprocessor System-on-Chip (MPSoC) architectures as a promising solution to deal with the potential parallelism inherent from multimedia applications. Recently, the ITRS [11] and HiPEAC 1 roadmaps promote power defines performance and power is the wall. In fact, power consumption is becoming a critical pre-design metric in complex embedded systems such as MPSoC. Facing this issue, designers should calculate and optimize the power consumption as early as possible in the design flow to reduce the time-to-market and the development cost. Today, system level power estimation is considered a vital premise to cope with the critical design constraints. However, the development of tools for power estimation and optimization at the system level is in the face of extremely challenging requirements such as the seamless power-aware design methodology, the accurate and fast system power modeling and estimation approach, and the efficient power optimization technique.

II. Related works

To ensure a high level of energy and power optimization for modern embedded real-time multimedia application, several studies have been proposed for the exploration of scheduling policy and dynamic Voltage/Frequency management. On one hand, authors present in [1] [2], the dynamic power management (DPM) techniques for multiprocessor real-time systems. These techniques improve power conservation capabilities by changing selectively the multiple idle states taking into account the cost of transitions power [3] [4]. They are divided into predictive schemes trying to predict future scheduling input to the system and stochastic schemes designing power management through the controlled Marcov process [3]. Bhatti [1] proposes a DPM technique called the Assertive dynamic Power management that allows the extraction of all idle time from some processors and clusters them on some others to elongate the duration of idle time in order to turn-off or to reduce the performance of system components unexploited and then we have a power more efficiently. On the other hand, the dynamic voltage and frequency scaling (DVFS) is another widely used energy reduction strategy. It lowers dynamically voltage and frequency to reduce both dynamic and static power consumption [5] [6]. Real time DVFS techniques are classified into intra task based on redistribution of slack time between tasks [6] [7] and intra tasks based on redistribution of slack time inside the same task [1]. Many related works discussed these approach(s), in this paper we explore a technique in the category of inter task DVFS called Dynamic Slack Reclamation (DSR) on embedded multimedia application. In multiprocessor systems, the energy minimization by DVFS strategy has many limits such as, its integration dynamically in applications. In this work, we integrate the DSR in a simulation platform in order to dynamically change the power consumption's profile for different processors.

III. Power/Energy Optimization

Many techniques used at various levels were then developed in order to make a gain in time and consumption in the final

978-1-4673-0784-0/12/$31.00 ©2012 IEEE

product. The Dynamic Voltage Frequency Scaling (DVFS) has been particularly distinguished by its efficiency to reduce CPU consumption. It can perform various tasks of an application to different couples voltage / frequency depending on the workload of the processor. Several strategies have been proposed to exploit certain aspects of DVFS and offer a particular method to build pseudo intermediate frequencies for use in conjunction with the techniques of Dynamic Voltage Scaling (DVS) [8] [9].

These pseudo frequencies are built at startup, and then at runtime, the algorithm retrieves the utilization of CPU and selects a pseudo frequency level at or above the rate of use. Finally, the algorithm corresponds the pseudo frequency with a frequency supported by the processor. This method is not permitted if the effective time of the tasks is less than their worst case execution time. In this paper we integrate one of these strategies, the dynamic slack reclamation (DSR) algorithm, in an open platform for simulation prototyping of multi-processors system on chip (MP-SoC) called SoCLib in order to reduce dynamically the energy of the embedded multimedia application.

IV. Dynamic Slack Reclamation Strategy

A. General view of the platform SOCLIB

The platform SoCLib [10] addresses a particular class of embedded systems: integrated multiprocessor systems on chip (MP-SoC), which are used for simulation in all industrial sectors: telecommunications, video and multimedia, automotive, transport, etc... In many industries, the chips that control the equipment are the main differentiating factors between products, and the time of design and development of a new product (time to market) plays a decisive role in the competition between manufacturers concerned. SoCLib provides two types of simulation models: Level models CABA (Cycle Accurate Bit-Accurate) that allow an accurate assessment of performance. Level models fciteTLM-T (Transaction Level Model with Timing) allow a significant reduction in simulation time. These simulation models use the SystemC language, derived from the C + +, suitable for both modeling levels retained, which is increasingly widely used in European industry. Moreover many simulation models are supported by SoCLib: (1) Processor Models like Power PC, ARM and MIPS, (2) Standard on-chip memories and (3) Several kinds of networks-on-chip. To validate our appoach, we have used MPIS processor and TLM-T Level models.

B. Architecture and application model

We configure the SoCLib with n identical MIPS processors ($\pi = \pi 1, \pi 2, \cdots, \pi n$). All processors support Dynamic Voltage Frequency Scaling. We consider a finite set of m real time, independent tasks ($t = t1, t2, \cdots, tm$). Each task is characterized by at least a quadruplet (r, c, d, T) where r is the release time, c is the worst-case execution requirement, d is the deadline and T is the periodicity. Moreover, a task may have runtime parameters such as best-case (B), worst-case (C) or actual-case (AET) execution times. For example, a task Ti has AETi as actual execution time, Ci (WCET) as estimated worst case time, Bi as best-case time and Li as laxity requirement (is a runtime parameter of a tasks job that is a measure of its urgency to execute relative to its deadline). These parameters are presented in fig 1.

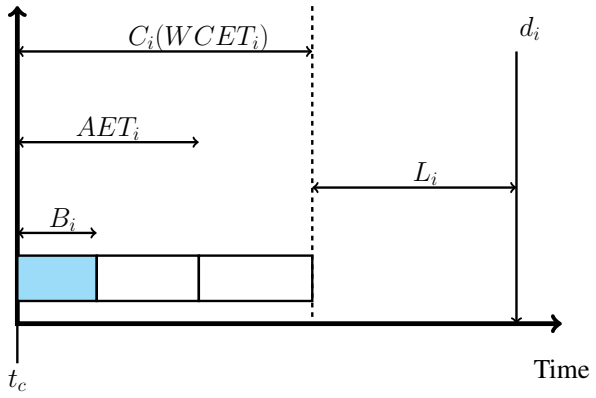


Fig. 1: DSR task's parameters

C. Dynamic Slack Reclamation (DSR) Algorithm

In real-time multimedia applications, timeliness guarantees (being at the right time) are provided through statically performed worst-case analysis. This is a conservative analysis because at runtime, real-time tasks can exhibit large variations in their actual execution time and thereby produce dynamic slack due to their early completion. The DSR algorithm is based on the slowdown strategy of reducing the processor power consumption. Slowdown is known to reduce the dynamic power consumption at the cost of increased execution time for a given computation task. It's based on detecting early completion of tasks. By comparing the actual execution time (AET) of a tasks job with its worst-case execution time (WCET), DSR determines the value of the dynamic slack. Knowing that it is not possible to determine the exact amount of actual execution time of the running task until it terminates, the algorithm computes the value of dynamic slack boundaries only. In addition, this slack, as the difference between WCET and AET allows reducing the speed of lower priority tasks. The DSR algorithm does not share dynamic slack with other processors in the system. Rather, the slack is fully consumed on the same processor by a subsequent job, which is assigned to that processor. In this paper our contribution is to integrate the dynamic slack reclamation algorithm of BATHI in transactional simulation platform in order to have dynamically real-time results.

As is described in the algorithm below, we consider that S^{Scan}, the scheduling tasks and the assigning of tasks to each processor according to the priority are defined by Mutekh: the operating system SOClib (lines 1-3). At line 4, we set the dynamic slack (ε), the slack factor (ϕ) and the initial frequency (f) for all processors. For each task completed, according to the value of ε, available or not available (line 8 and 16) calculated in line 7, the algorithm computes additional available time T_{av} at current processor frequency f (line 9).

Algorithm 1 Dynamic Slack Reclamation

1: **obtain** S^{Scan}
2: **sort** ready task queue w.r.t preemtive priority order
3: **assign** n highest priority task n processor
4: **set** $\epsilon \leftarrow 0$; $\phi \leftarrow 1.0$; $f \leftarrow f_{max}$
5: **for** Scheduling event **do**
6: **if** Scheduling event=termination **then**
7: $\epsilon \leftarrow C_{i-1,j} - AETI_{i-1,j}$
8: **if** $\epsilon > 0$ **then**
9: compute available t_{av} for t_{av} at f $\{(t_{av} = C_{i,j} + \epsilon)\}$
10: compute required t_{av} for $t_{i,j}$ at f $\{(C_{i,j})\}$
11: $\phi \leftarrow t_{av}/C^f_{i,f}$
12: update f w.r.t. ϕ
13: **if** $f < f_{min}$ **then**
14: $f \leftarrow f_{min}$
15: execute $T_{i,j}$ at f
16: **else if** $\epsilon = 0$ **then**
17: $v \leftarrow f_{max}$
18: **end if**
19: **end if**
20: **end if**
21: **end for**

The worst-case execution time $C^f_{i,j}$ of $T_{i,j}$ is estimated at current processor frequency f (line 10). Once T_{av} and $C^f_{i,j}$ are calculated, ϕ is computed and f is updated in the appropriate processor (line 11). It is possible that the precedent job $T_{i-1,j}$ has produced enough dynamic slack to reduce f below the allowable minimum processor frequency f_{min} . To avoid such situation, f is remplaced by f_{min} in DSR (lines 14). If there is no positive slack generated by $T_{i-1,j}$, processor k continues to execute with maximum frequency f_{max} (lines 16 -17).

Example: Let us consider two preemptive real-time tasks, namely T1 and T2, which are scheduled under the EDF scheduling algorithm. We suppose that T_1 has higher priority over T2 due to smaller deadline. As shown in Fig 2, the job $T_{1,j}$ of T1 finishes its execution earlier than its worst-case execution time ($C_{1,1}$) for its first job $T_{1,1}$ as shown in inset (a). In the canonical schedule S^{Scan}, if job $T_{1,1}$ would have finished with its worst-case execution time, the termination of $T_{1,1}$ was estimated at time instant $T^{can}_{1,1}$. However, $T_{1,1}$ terminates at time instant $T^{pra}_{1,1}$ after consuming $AET_{1,1}$ time units and produces ε units of dynamic slack. Similarly, job $T_{2,1}$ of T_2 was expected to start its execution at time instant $T^{can}_{1,1}$ and finish by $T^{can}_{1,2}$ as illustrated in inset (b). However, due to the early completion of $T_{1,1}$, now $T_{2,1}$ has T_{av} time units available to finish its execution as expected in its S^{Scan}. Since $AET_{2,1}$ is not known a priori, therefore, $C_{2,1}$ is considered as the execution requirement for $T_{2,1}$. Based on the knowledge of available slack ε and task boundary in S^{Scan}, execution of $T_{2,1}$ is slowed down by a factor of as shown in inset (c).

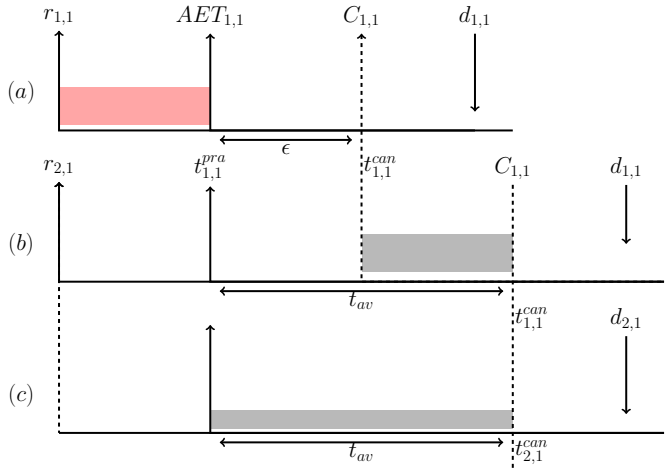


Fig. 2: Slack reclamation using the DSR algorithm

V. THE $H.264$ DECODER CASE STUDY

In many critical applications it is necessary to ensure a complete transmission of the instructions. For example, in the event that the battery does not provide the sufficient energy, we need to change the quality of transmission to run with reduced power consumption. The work proposed in this section is to apply hardware and software methodology for an operating mode of decoding based on $H.264$.

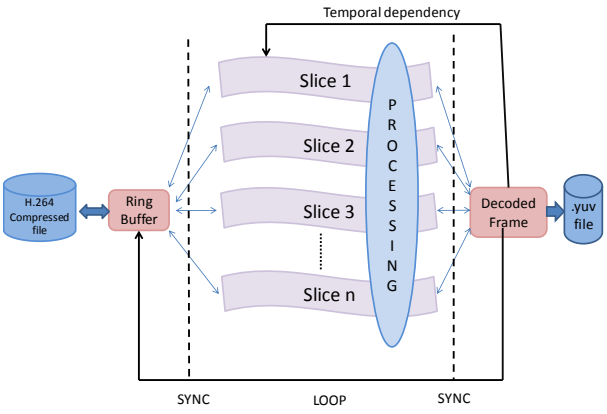


Fig. 3: Block diagram of H.264 decoding scheme slices version

In this work, we take H.264 as the main application. It is a high quality video compression algorithm relying on several efficient strategies extracting spatial (within a frame) and temporal dependencies (between frames). It is suited for all types of applications with different ranges of bit rates. H264 is characterized by a flexible coding, high compression and high quality resolution. Moreover, its a promising standard for embedded devices. The operating principle of this application is defined by these steps: First, a compressed bit stream coming from the NAL layer is received at the input of the decoder where the NAL is specified to format that data

and provide header information in a manner appropriate for conveyance by the transport layers or storage media. Then, the entropy decoded bloc begins with decoding the slice header where each slice consists of one or more 1616 macroblocks, and then it decodes the other parameters. The decoded data are entropy and sorted to produce a set of quantized coefficients. These coefficients are then inverse quantized and inverse transformed. Thereafter, the data obtained are added to the predicted data from the previous frames depending upon the header information. Finally the original block is obtained after the de-blocking filter to compensate the block artifacts effect. The H.264 video decoder application can be broken down into various tasks sets corresponding to different types of parallelization. In our experiments, we use the slices version, one of the task models of H.264 proposed by Thales Group, France [11] [12] in the context of French national project Pherma [13]. The main characteristic of this version is that the algorithm is parallelized on the slices of the frame as illustrated in fig 3 From this diagram, we consider that we have four types of tasks. First, we start with the NEW FRAME tasks (T1) that can access only sequentially to the input data buffer. Therefore, the NAL-DISPATCH task (T2) which provides access to a shared resource is protected by a semaphore. Then, SLICE PROCESS tasks (T3, T4... Tn) are launched simultaneously. Due to temporal dependencies between frames, it is not possible to compute the next frame if the previous one has not been completely decoded. Thus, at the end of each slice computation, tasks need to be resynchronized using task named SYNC before running the DECODED FRAME (Tn+1) task.

VI. Energy/power consumption analysis

The main purpose of this section is the deployment and evaluation of the dynamic slack reclamation strategy on the transactional virtual level of the power-aware design methodology. The related work describes a dynamically power consumption analysis of the decoder H.264 according to this configuration (3 slices, 25 frames). The platform is configured with 3 processors at 200MHz. Results in figure 4 shows tasks scheduling (T3, T4 and T5), according to the task model presented in fig 3. Based on this real time result, we determine the parameters of each task (release time, actual-case and worst-case execution time, relative deadline and periodicity).

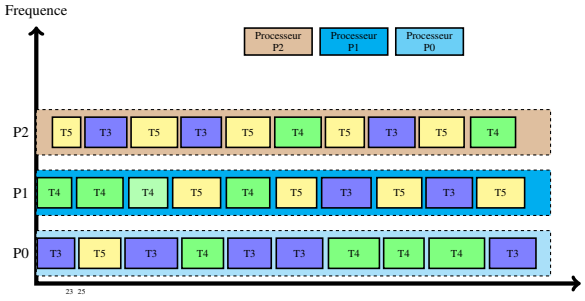


Fig. 4: Distributed real-time tasks on three processors

TABLE I: Power Models

Mapping	Voltage	Power laws
BRAM	1.5V	P(mW) = 0.40 $F_{processor}$ + 3.24 F_{bus} + 74
	2.5V	P(mW) = 5.37 F_{bus} + 1588
SDRAM	1.5V	P(mW) = 0.38 $F_{processor}$ + 3.45 F_{bus} + 79
	2.5V	P(mW) = 4.1γ+6.3F$_{bus}$ + 1599

During the execution of the decoder we take into account these parameters as an input of the DSR algorithm. First step, we obtain pseudo frequencies corresponding to each processor dynamically throughout the applications simulation. In order to optimize the power consumption of the overall system, we change dynamically the frequency of each processor according to these pseudo frequencies shown in fig 5.

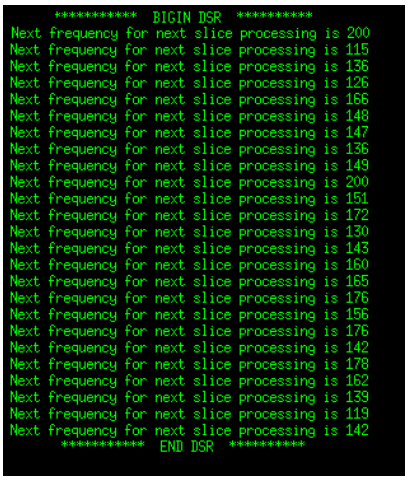
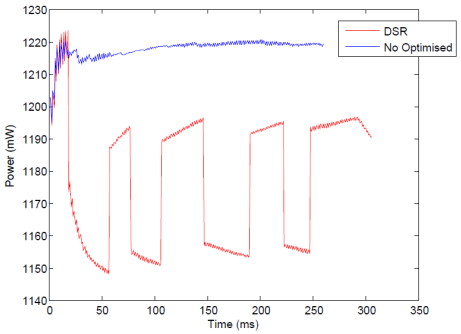


Fig. 5: DSR Algorithm results

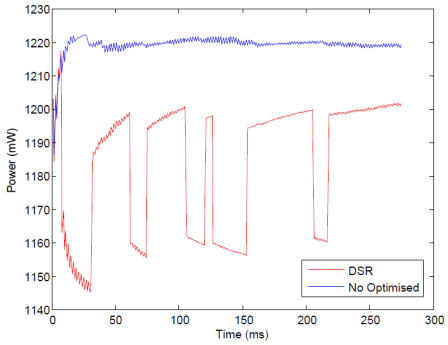
These states correspond to more efficient energy. To optimize the power consumption of the overall system, we dynamically change the frequency of each processor in terms of these pseudo frequencies. Given that MIPS processors support a certain frequency range then we set both values equal to 100 MHz and 200 MHz to meet deadlines for the execution of various tasks. At the beginning of each call to the task processing Slice, we change the CPU frequency based on the outputs of DSR algorithm. The change in frequency is equivalent to a change in time that requires a transition from the application layer (H.264) to the hardware layer (ISS). We then use an assembler instruction taking as parameter the new value off frequency to be detected by the ISS component that modifies its operating frequency in real time. To analyze the power consumption of processors during the applications execution, we use the power model elaborated by [14] given by table 1 where γ is the Cache miss rate for a processor.

Figure 6 show the simulation results on the changes in power consumption under non-optimized schedule and optimized schedule using DSR for H.264 video decoder application on the three processors.

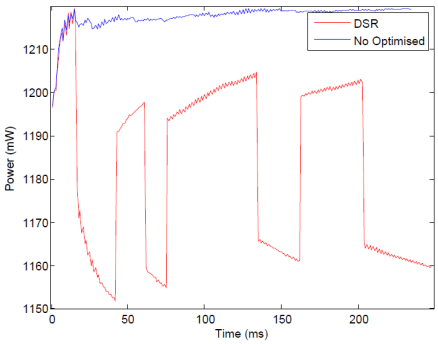
We note at the beginning of the execution, the power consumption does not change for the three processors by using



(a) Power optimization on processor 1



(b) Power optimization on processor 2



(c) Power optimization on processor 3

Fig. 6: Power optimization on the three processors

the optimization algorithm. This amounts to the execution of sequential tasks T1 and T2 shown in the task graph in figure 3. Starting the treatment of slices, we observe remarkable titling in terms of consumption power witch amount to a dynamic change of the operating frequency of the processors throughout the simulation. We managed to change the behavior of processors in a dynamic way to reduce the power dissipation of the on-time tasks. For example, we note that the first processor has a reduced power dissipation equal to 70 mW in the first switching frequency.

For a better validation of this result, we perform a more general exploration. We present the results of power consumption during the execution of the decoder given by figure 7 in different configurations of the multiprocessor platform SoCLib. We have different configurations of the number of active cores to verify the impacts on power consumption with

Nb Proc	DSR	Power(mW)	Time(ms)
1	NO	1591	244
	YES	1535	250
2	NO	2125	276
	YES	2019	277
3	NO	2647	247
	YES	2497	260

Fig. 7: Simulation Results

and without the optimization algorithm. This analysis takes into account the CPU load by varying the number of threads with different configurations decoder (1, 2, 3 slices). The table of simulation results shows the power dissipation and execution time resulting from the execution of the workload of H.264 in different configurations of active processors.

VII. Conclusion

In this paper, we have optimized and implemented a dynamic voltage and frequency scaling technique for real-time systems, called the Dynamic Slack Reclamation (DSR) technique, which falls in the category of inter-task DVFS techniques, in a simulation multiprocessor platform MPSoC. To evaluate DSR, H.264 video decoder application is taken as main use case application. Simulation results illustrate that the DSR algorithm can obtain power savings from 1220 mW to 1150 mW for agiven time according to one processor . Note that reported results are based on the assumption that it is possible to vary dynamically the operating frequency on every processor independently.

References

[1] BHATTI. K, Energy-aware scheduling for multiprocessor real-time systems. Ph.D. thesis, University Nice Sophia-Antipolis, 2011.

[2] S. Irani, R. Gupta, and S. Shukla., Competitive analysis of dynamic power management strategies for systems with multiple power savings states, Proceedings of the conference on Design, automation and test in Europe, pp. 117, 2002.

[3] L. Benini, A. Bogliolo, G. A. Paleologo, and G. De Micheli, Policy Optimization for Dynamic Power Management, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 18, no. 6, pp. 813–833, 1999

[4] Sandeep K. Shukla and Rajesh K. Gupta.i, A model checking approach to evaluating system level dynamic power management policies for embedded systems," Proceedings of the Sixth IEEE International High-Level Design Validation and Test Workshop, 2001.

[5] S. K. Shukla and R. K. Gupta, A model checking approach to evaluating system level dynamic power management policies for embedded systems, in Proceedings of the 6th IEEE International High-Level Design Validation and Test Workshop, pp. 5357, Monterey, Calif, USA, September 2001.

[6] D.SHIN and J.KIM, Fine-grained DVFS using on-chip regulators, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 8, 2011.

[7] D.SHIN and J.KIM, Intra-task voltage scheduling on DVS-enabled hard real-time systems, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 24, 10, 1530 1549, 2005.

[8] PILLAI, P. AND SHIN, K. G, Real-time dynamic voltage scaling for low-power embedded operating systems, In Proceedings of the eighteenth ACM symposium on Operating systems principles, 2011.

[9] K.Choi, W.Lee and M.Pedram DEPARTMENT OF EE-SYSTEMS, UNIVERSITY OF SOUTHERN CALIFORNIA, L. A.-C, Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation.

[10] The Soclib Website. https://www.soclib.fr/.
[11] ITRS. 2010. Design, 2010 edition. http://public.itrs.net/.
[12] Thales. Thales group (France), 2010. http://www.thalesgroup.com/ pp. 70, 72, 104, 126.
[13] QEMU. 2010. Qemu. http://wiki.qemu.org/.
[14] S.K. RETHINAGIRI, R. B. ATITALLAH, S. NIAR, E.SENN, and J-L.DEKEYSER. "Hybrid System Level Power Consumption Estimation for FPGA-Based MPSoC", 29th IEEE International Conference on Computer Design (ICCD), October 2011, Amherst, MA, USA.