An Energy-Aware Scheduler for Dynamically Reconfigurable Multi-Core Systems

Robin Bonamy, Sébastien Bilavarn, Fabrice Muller

An Energy-Aware Scheduler for Dynamically Reconfigurable Multi-Core Systems

Robin Bonamy, Sébastien Bilavarn, Fabrice Muller
LEAT, University of Nice Sophia Antipolis, CNRS, France
robin.bonamy@unice.fr, sebastien.bilavarn@unice.fr, fabrice.muller@unice.fr

***Abstract*—This paper describes an energy-aware scheduling approach intended for use in heterogeneous multiprocessors supporting hardware acceleration with Dynamic and Partial Reconfiguration. Scheduler decisions rely on pragmatic power and energy models to map the load across cores and reconfigurable regions with regards to the actual power costs. Results on a multithreaded H.264/AVC profile decoder with three possible hardware functions on a Xilinx Zynq based platform report energy gains up to 44.1% over full software execution and 49.6% over static hardware / software execution, while ensuring real-time decoding requirement.**

I. Introduction

The use of heterogeneous multiprocessor System-on-Chips has increased because of their potential to address energy efficiency, power and heat density problems. Despite the promises, the increasing level of heterogeneity greatly affects the design and mapping complexity in upcoming systems and applications. Indeed it is no longer simply a question of mapping efficiently concurrent processes to the best cores, but also to consider dynamic aspects such as power management or hardware acceleration and their impact on energy efficiency.

Hardware acceleration is a relatively well-known player in the energy performance equation which is regaining attention today with the advent of Dynamic and Partial Reconfiguration (DPR). Partial reconfiguration is a technique related to FPGAs that can be used to extend their inherent flexibility: it allows to reprogram specific regions with new functionality while other regions continue to run. Drastic reduction of hardware resource utilization results in less static power thus significant better energy efficiency, adding to the inherent benefits of dedicated hardware. However, a variety of parameters such as FPGA partitioning, accelerator parallelism or software execution strongly affect the actual processing efficiency, and other techniques such as blanking or DPR based clock gating can also be used to further decrease power. As a result, the quantity and scope of decisions in a dynamically reconfigurable multi-core system highly complexifies the scheduler's job. It is however critical to provide good support at this level since bad decisions can affect energy efficiency to the point of total ineffectiveness. We address in the following the definition and evaluation of an energy-aware scheduling and mapping approach for multi-processor systems supporting hardware acceleration with DPR, and report representative application results and achievements that are denoting promising prospects in this field.

978-1-4673-7942-7/15/$31.00

The outline of the paper is the following. We first review existing works in the field of heterogeneous multi-core scheduling, pointing out relative novelty in the use of DPR for that matter. In section 3, we introduce pragmatic power and energy models underlying the proposed scheduling procedure. We then describe the decision schemes used in the energy-aware scheduling and mapping process. Detailed results on a multithreaded H.264/AVC decoder on a Xilinx Zynq based platform are analyzed and discussed. Finally, main conclusions from these results are presented and future directions for extension and exploitation are proposed.

II. Overview of heterogeneous multiprocessor strategies

With the rise of multi-core heterogeneous architectures, there has been many works addressing efficient scheduling application workloads with multiple cores, possibly of different types. Early investigations focused primarily on improving load balancing to achieve better performance (throughput, instructions-per-cycle, etc.). Recently the challenge of heterogeneous thread scheduling and global power management switched on delivering higher power-performance levels (performance per Watt). Many works explored the energy efficiency benefits of heterogeneity, that are more extensively discussed for instance in [1]. There is a broad consensus on the fact that exploiting heterogeneity is essential for a better use of energy, the necessary counterpart being that it greatly complexifies the scheduler.

Of the work addressing this problem, [2] proposed a scheduler for a system of processors based on execution prediction to map future processing needs to the most suited processor. Their method applies to single-ISA heterogeneity supporting differing voltages and frequencies. [3] extended a symbiotic scheduling heuristic [4], originally developed to enhance throughput and lower response time, for chip multiprocessors with simultaneous multithreading cores. They report up to 7.4% savings in energy, 10.3% savings in energy-delay product, and 35% savings in power. [5] is another contribution considering both scheduling and power management to address process variations in CMPs. They conducted a design exploration, proposed a number of schedulers to satisfy different objectives, and developed a linear programming solution for power management. The authors in [6] examined the scalability problem for manycores, comparing basic scheduling heuristics and proposed scheduling and power management algorithms for heterogeneous systems scaling up to 256 cores. A recent study [7] addressed a study including the ARM big.LITTLE architecture, associating an energy

efficient processor cluster (Cortex-A7) with a higher performance processor (Cortex-A15). They report that different class of resource allocation heuristics (race-to-idle vs. never-idle) have very different results on different platforms, indicating that the efficiency of a strategy greatly depends on platform characteristics and can go as far as getting inefficient in some cases.

Aside from the question of core specialization, heterogeneity also extends to the aspects arising at run-time due for example to power management [8]. The dynamic use of different power states (P-states, sleep states) available for each core matches the problem of low power scheduling which had a long history of research over the last 20 years. Surveys have been described for example in [9] and [10] from the abundant literature. Our own previous experimental studies in this field [11] came to similar conclusions as [7] and [12] concerning the importance of platform characteristics and application knowledge on the efficiency of a strategy. Likewise, results indicate that custom strategies, i.e. more specialized schedulers dedicated to an application or application domain, can reach significant levels of energy gains (in the 5% to 50% range for video processing applications on representative platforms) compared to existing OS and platform-based strategies.

From this large literature, existing works often consider a specific perspective on the heterogeneous scheduling problem: a type of architecture, one precise objective (manycore scaling, process variability), limited heterogeneity (similar cores, DVFS), etc. In addition, a type of heterogeneity remains uncovered regarding the recent advancement of graphic and reconfigurable processing units: the possibility of hardware (or accelerated) execution of tasks. [1] is thus extending heterogeneity to hardware acceleration enabled with the progresses of Dynamic and Partial Reconfiguration (DPR) and High-Level Synthesis (HLS). Hardware acceleration delivers orders of magnitude performance and energy efficiency compared to software execution, and the flexibility introduced with DPR can improve these benefits. The underlying heterogeneous mapping and scheduling problem have started lately to be re-investigated and this paper extends the conclusions of [13] on the definition of low power scheduling policies able to support efficient execution of dynamic hardware and software tasks.

III. Power modeling

The remainder of this paper addresses a scheduling method capable of taking relevant run-time decisions to reduce the energy use while satisfying performance constraints. As such it requires that the power impact of resource allocation decisions can be realistically predicted. This section describes the underlying formal characterization, which results in great part from previous work reported in [13], and is divided in two main components: a platform and an application model.

A. Platform model

The platform is described by enumerating the execution units (EU) available and providing information on the corresponding power characteristics. To cope with the target reconfigurable multi-core platform, two main execution units are considered : a) processor cores and b) reconfigurable regions (RR).

The power model used for CPU cores is based on the type of core and is defined by the related static power $P^{static}_{core_j}$, the idle power $P^{idle}_{core_j}$ and the run power $P^{run}_{core_j}$, where j is the id of the execution unit.

As a RR needs to be configured before a task can run, which takes time and power that has also to be assessed, the reconfiguration controller can be modeled by its power P^{reconf}. In turn, the reconfiguration time depends on the area of the RR to configure ($T^{reconf}_{RR_j}$).

The power model for the RRs is defined by the static power consumption of the resources (e.g. slices, BRAM, clock tree, etc) in the given RR ($P^{static}_{RR_j}$). The related idle and run power depend on the actual task to be configured on this region. So the corresponding idle and run power are not defined in the platform model, instead they are characterized in the following application model.

B. Application model

An application is characterized by a set of tasks ($\mathcal{G}$) with their data and execution dependencies modeled by a task flow graph. Each task can have one or several implementations available, at least every task has a software implementation. An implementation is a description of how the task is being executed on a software or hardware EU. The model of an implementation reflects the task id (i), the EU id (j), execution time ($T_{i,j}$), and idle / run power consumption for hardware execution (P^{idle}_{i,RR_j}, P^{run}_{i,RR_j}). In case of software execution, the task idle / run power numbers are derived from the core idle and run power described in the platform model.

IV. Scheduler specification

The scheduling procedure involves two steps: a waiting task list is first determined and sorted according to a specific strategy (e.g. Earliest Deadline First), then a task implementation is mapped on a given execution unit according to an energy efficiency criteria. In the following, this process is further referred to as Energy Aware Heterogeneous Scheduler (EAHS).

A. Task Scheduling

Most common task scheduling strategies are a) FIFO (First In First Out) the first task arriving in the waiting state will be the first scheduled, b) EDF (Earliest Deadline First) which consists of sorting tasks to first schedule the one that must finish first and finally c) priority sort, a priority is assigned to each task and the highest priority task is executed first.

However these strict definitions may be alleviated in case of heterogeneous scheduling. Considering a scenario where the top task in the waiting list is blocked, waiting for matching execution unit to be free, then all other tasks in the waiting list are blocked despite the fact that they may be run on other free EUs. This will increase the application delay and execution units under-utilization. Thus we propose a *soft* scheduling approach which consists of using one of the previously mentioned strategies, but letting the possibility to bypass a task blocked because a corresponding mapping is not possible for the moment (e.g. either the reconfiguration controller is busy or compatible execution unit(s) not free).

B. Task Mapping

When managing task allocation on the architecture resources, the choice of an implementation has a major impact on reducing energy consumption. A cost function is thus defined to help identifying the most energy efficient mapping from the different possibilities. This cost is computed each time a task has to be mapped, for all possible implementations, which is based on energy and execution time estimations of the implementation being processed.

Energy estimation is given in equation (1):

$$\forall j \ ; \ E_j^{cost} = \left\{ \begin{array}{ll} E_{i,j}^{run} & \text{when } \rho_{i,j} = 0 \\ E_{i,j}^{run} + E_j^{conf} & \text{when } \rho_{i,j} = 1. \end{array} \right\} \quad (1)$$

where

$$\begin{aligned} E_{i,j}^{run} &= P_{i,j}^{run} \times T_{i,j} \\ E_j^{conf} &= T_{RR_j}^{reconf} \times P^{reconf} \end{aligned}$$

and $\rho_{i,j}$ represents the need to perform a reconfiguration. For instance if the same implementation is already configured, $\rho_{i,j} = 0$ means that RR_j can be used directly to run task i, otherwise $\rho_{i,j} = 1$ and a reconfiguration is required before execution. For all non reconfigurable EUs: $\rho_{i,j} = 0$.

To execute a new task, the scheduler has to check first that the execution unit EU_j currently evaluated is free. If not, the task can be implemented later and this delay must be considered for decision. Task execution time is thus estimated in equation (2):

$$\forall j \ ; \ T_j^{cost} = T_{i,j} + T_{RR_j}^{reconf} \times \rho_{i,j} + T_j^{busy} \quad (2)$$

where T_j^{busy} represents the time during which the EU_j is busy, running another task i'. It is derived from the execution time $T_{i',j}$, the begin time of the task currently running, and the current scheduling time.

The final cost for EU_j is computed in equation (3), where α is a user parameter (real value between 0 and 1) to promote performance (α close to 0) or energy efficiency scheduling (α close to 1).

$$\begin{aligned} &\forall j \ ; \\ &Cost_j \quad = \alpha \ \frac{E_j^{cost}}{max(E^{cost})} + (1-\alpha) \ \frac{T_j^{cost}}{max(T^{cost})} \end{aligned} \quad (3)$$

The cost function is evaluated for all available implementations and execution units for the current top task in the waiting list. The lowest $Cost_j$ value is selected and the task is implemented on EU_j. However, if EU_j is busy the task is kept waiting and will be implemented later.

V. Application Study

The application considered in this case study is a H.264/AVC profile video decoder. The input specification code used is a version derived from the ITU-T reference code [16] to better cope with hardware design constraints. From the code profiling, five major functions are highlighted and define the task set $\mathcal{G}$ as

$$\begin{aligned} \mathcal{G} \quad = \quad & \{Exp\_Golomb, \ MB\_Header, \ Inv\_CAVLC, \\ & Inv\_QTr, \ Inv\_Pred, \ DB\_Filter\}. \end{aligned} \quad (4)$$

An ESL design methodology [15] is used to provide real implementations for the possible hardware functions. The deblocking filter (DB_Filter), inverse CAVLC (Inv_CAVLC), and inverse quantization and transform block (Inv_QTr) contribute together to 76% of the global execution time on a single CPU core. They represent the three functionalities of the decoder that are generated from high level synthesis for hardware execution and can be either software or hardware executed.

This video decoder supports the possibility of slice decomposition of frames as defined in the H.264/AVC standard. A slice represents an independent zone of a frame, therefore decoding one slice (of a frame) is independent from another (slice of the same frame). This way, the decoder can process different slices of a frame in parallel and this application is an interesting case study for multi-core scheduling. We have thus considered a decomposition of the image where four streams process four slices of a same frame.

A. Modeling

The platform used in this case study is a Xilinx ZC702 [17] which is composed of a dual core ARM Cortex-A9 and a Xilinx Artix-7 equivalent FPGA. This development board allows power measurement for both FPGA and CPU core power supply. Thus measurement series are carried out under the following default setup conditions: 1V core voltage, 22°C room temperature, 667MHz ARM core frequency and 100MHz FPGA. The power values exposed in table I are setup from direct measurements on the ZC702 evaluation board.

TABLE I. Model parameters for a Zynq ZC702 platform.

Model	Value	Model	Value
P_{core}^{static}	89.82 mW	P^{reconf}	72 mW
P_{core}^{idle}	55.83 mW	$T_{RR_1}^{reconf}$	1.36 ms
P_{core}^{run}	119.4 mW	$T_{RR_2}^{reconf}$	1.36 ms

The task graph corresponding to the H.264 use case is shown in Figure 1. In this application configuration, reconfigurable regions defined for DPR execution are sized to host all hardware tasks and maximize the use of parallelism, with the methodology described in [18]. The two resulting RRs (RR_1 and RR_2) are then implemented and characterized in terms of

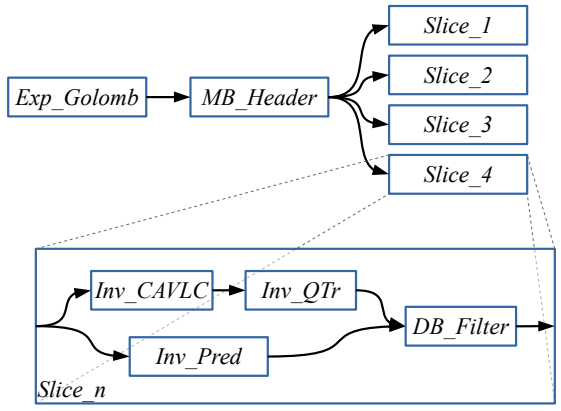

Fig. 1. H.264/AVC profile decoder task graph.

power and execution time on the ZC702 platform, leading to the task model parameters reported in table II.

TABLE II. MODEL PARAMETERS FOR H.264/AVC PROFILE DECODER ON A XILINX ZYNQ PLATFORM.

Task i	EU j	$P^{run}_{i,j}$ (mW)	$P^{idle}_{i,j}$ (mW)	$T_{i,j}$ (ms)
Exp_Golomb	$core_1$ $core_2$	119.4	-	6
MB_Header	$core_1$ $core_2$	119.4	-	5.9
Inv_CAVLC	$core_1$ $core_2$	119.4	-	6.6
	RR_1 RR_2	33.5	55.1	4.5
Inv_QTr	$core_1$ $core_2$	119.4	-	3.1
	RR_1 RR_2	5.8	34.2	1.5
Inv_Pred	$core_1$ $core_2$	119.4	-	3.2
DB_Filter	$core_1$ $core_2$	119.4	-	10.5
	RR_1 RR_2	6.4	44.3	0.9

B. Scheduling Results

The proposed scheduler is compared against four common scheduling strategies. User parameter α is set to 0.8 in a way to promote energy efficiency and to introduce a few performance concerns. Analysis on the H.264 decoder is carried out using a SystemC/TLM simulator [18] and results are reported in Table III and Figure 2. FPGA area is expressed in terms of Xilinx slice resources along with execution time and energy consumed to decode one frame. An energy-delay metric is also computed to provide additional measure and comparison of the energy performance trade-off [14]. Two solutions are highlighted for an EDF full software scheduling execution. The first one is tailored for the proposed platform (two cores) but can not achieve real-time video decoding (58.7ms per frame). As the application hyperperiod should be less than 40ms, the platform model is extended for a total of four ARM Cortex-A9 cores.

Another reference result is the scheduling profile obtained using two cores and static hardware accelerators (EDF Static). Indeed for this application configuration and task parallelism, a very large reconfigurable area would be required (39536 slices), which can be hard to implement on a real FPGA. Therefore another solution is introduced (EDF SmartStatic) which implements only one static accelerator for each possible hardware function (DB_Filter, Inv_CAVLC, Inv_QTr) to sequentially map different task invocation to the unique corresponding accelerator.

Software execution requires a large amount of energy (35mJ) while static implementations benefit from dedicated hardware efficiency with 21.1mJ (EDF Static) and 19.5mJ (EDF SmartStatic). However EDF SmartStatic is slower than EDF Static (35.4ms vs. 19.3ms) because less accelerator instances limit the exploitation of task parallelism. The outcome is visible in the energy-delay product: EDF Static obtains the best score (407) followed by EDF SmartStatic (690) and EDF Software (1350).

As for the DPR solution (two cores, two RRs), EAHS has a score which is close to EDF Static (478). In this case, energy consumption dropped from 21.1mJ to 16mJ compared to EDF Static, but execution time increased due to reconfigurations and fewer hardware resources. In addition, it is worth noting that EAHS provides a solution which consumes 3.4mJ less and is 5.6ms faster than EDF SmartStatic thanks to a better use of reconfigurable resources.

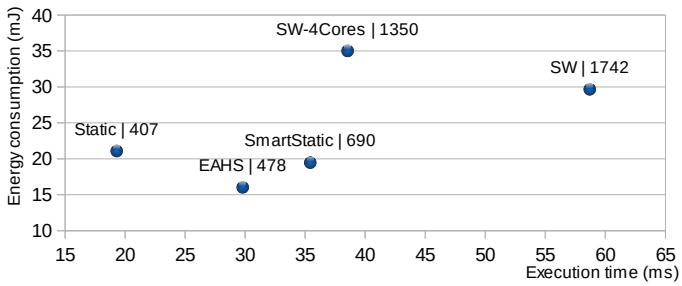

Fig. 2. Energy consumption versus time for different scheduling solutions along with energy-delay product.

Although these results are indicative of energy efficiency, they are not completely rigorous regarding the effective execution constraint which is to process frames periodically, every 40ms typically. In this case, the objective is not so much to minimize the performance energy product but to meet execution deadlines at the lowest energy cost. Therefore, previous energy results are adjusted in Table IV for an execution constraint of 40ms.

The result is that EDF Static consumes more energy than the four other solutions, close to 40mJ. It is due to the large static and idle power consumption arising from the amount of hardware accelerators when they are not in use. This solution which had the best energy-delay previously is in fact the worst if we consider the effect of idle power consumption. EAHS takes advantage of less reconfigurable resources in this case and provides the best energy efficiency with 20.06mJ, which is about 50% better than EDF Static and 6.8% more efficient than EDF SmartStatic.

The corresponding EAHS scheduling profile is presented in Figure 3, showing the usage of each execution unit during time for two 40ms periods. As defined in the platform model, two Cortex-A9 cores are used together with two reconfigurable regions. Reconfigurations are clearly visible (in pink) between different executions of tasks on the two RRs. All four invocations of each hardware accelerator are successively configured. The first four invocations of Inv_CAVLC are implemented using both RRs, so that they can run in parallel by pair. Then one RR is used for four sequential invocations of Inv_QTr and the other is configured to run four sequential invocations of DB_Filter. We see clearly in this application example the ability of the scheduler to maximize the use of hardware execution (more energy efficient), to take advantage of parallel execution, and to minimize the number of reconfigurations (having non-negligible energy overheads), which are essential conditions ensuring DPR energy efficiency in practical applications.

VI. CONCLUSION

This paper described a hardware / software scheduling strategy for energy efficient application execution on hetero-

TABLE III. COMPARISON OF DIFFERENT SCHEDULER RESULTS ON H.264 VIDEO APPLICATION.

Scheduler	N# Cores	FPGA area (eq. slices)	Execution time (ms)	Energy consumption (mJ)	energy-delay product
EDF Software	2	0	**58.7**	29.67	1742
EDF Software	**4**	0	38.54	35.02	1350
EDF Static	2	**39536**	19.31	21.09	407
EDF SmartStatic	2	6372	35.4	19.47	690
EAHS	2	6600	29.8	16.04	478

TABLE IV. COMPARISON OF DIFFERENT SCHEDULER ENERGY CONSUMPTION ON A 40MS PERIOD H.264/AVC PROFILE DECODER.

40ms period energy (mJ)	Energy Gain	EDF SW 4Cores	EDF Static	EDF SmartStatic	EAHS
35.87	EDF SW 4Cores	-	9.9%	-66.7%	-78.8%
39.81	EDF Static	-11.0%	-	-85.0%	-98.5%
21.52	EDF SmartStatic	40.0%	45.9%	-	-7.3%
20.06	EAHS	**44.1%**	**49.6%**	**6.8%**	-

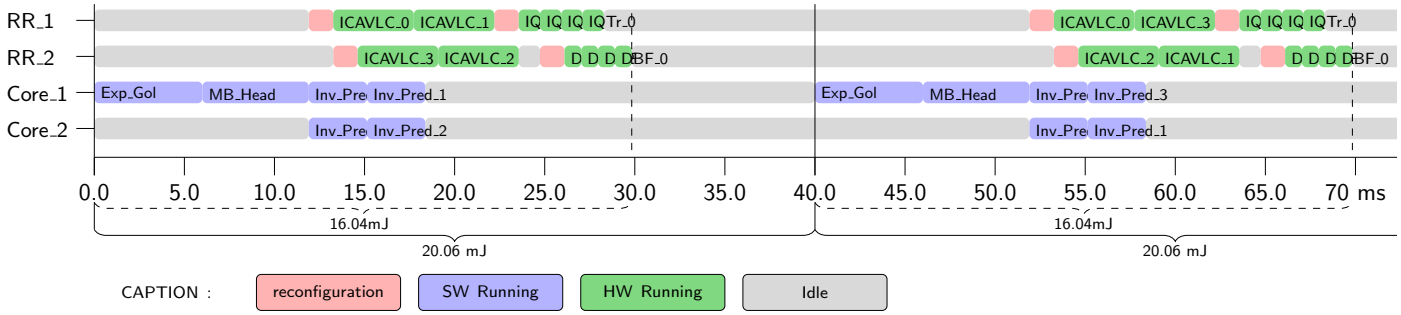

Fig. 3. EAHS scheduling and mapping results on the H.264/AVC profile decoder.

geneous multi-core systems with dynamically reconfigurable accelerators. Results show net improvements in energy savings both against full software execution (44.1%) and static hardware / software execution (49.6%). The global study shows therefore that a great potential lies in the exploitation of DPR assuming scheduling is based on reliable decision support able to clearly identify a low energy solution among the large overall mapping space. It is also likely that the acceleration potential of applications gives room for improving these results as more accelerators would fit in less space. Future works will thus address methods for exploring this potential. Interesting improvements can also be foreseen from extending the proposed EAHS strategy with DVFS in the first place, as a standard multiprocessor technique to reduce power, but also blanking to further exploit DPR in accelerator static power limitation.

ACKNOWLEDGMENTS

This work is carried out under the BENEFIC project (CA505), a project labelled within the framework of CATRENE, the EUREKA cluster for Application and Technology Research in Europe on NanoElectronics.

REFERENCES

[1] C. Leech, T.J. Kazmierski, Energy Efficient Multi-Core Processing. ELECTRONICS, 18, (1), pp. 3-10, jun 2014. (DOI:10.7251/ELS1418003L).

[2] S. Ghiasi, T. Keller, F. Rawson, Scheduling for Heterogeneous Processors in Server Systems, Proceedings of Computing Frontiers, 2005.

[3] M. DeVuyst, R. Kumar, D.M. Tullsen, Exploiting Unbalanced Thread Scheduling for Energy and Performance on a CMP of SMT Processors, Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS 06), 2006, pp. 117.

[4] A. Snavely, D. Tullsen, Symbiotic jobscheduling for asimultaneous multithreading architecture, Eighth International Conference on Architectural Support for Programming Languages and Operating Systems, Nov. 2000.

[5] R. Teodorescu, J. Torrellas, Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors, Proceedings of the 35th International Symposium on Computer Architecture (ISCA), June 2008, pp. 363-374.

[6] J.A. Winter, D.H. Albonesi, C.A. Shoemaker, Scalable Thread Scheduling and Global Power Management for Heterogeneous Many-Core Architectures, 19th international conference on Parallel architectures and compilation techniques (PACT'10), pp. 29-40, 2010.

[7] C. Imes, H. Hoffmann, Minimizing Energy Under Performance Constraints on Embedded Platforms, 4th Embedded Operating Systems Workshop, EWiLi'14, Lisboa, Portugal, nov 13-14, 2014.

[8] F.A. Bower, D.J. Sorin, L.P. Cox, The Impact of Dynamically Heterogeneous multi-core Processors on Thread Scheduling, IEEE Micro, vol 28, Issue 3, pp. 17-25, jun 2008, (DOI) 10.1109/MM.2008.46

[9] L. Benini, A. Bogliolo, G. De Micheli, A Survey of Design Techniques for System-Level Dynamic Power Management, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 8, 3, pp. 299-316, 2000

[10] P. Singh, V. Chinta, Survey Report on ynamic Power Management, Survey report of University of Illinois, Chicago (ECE Department), Chicago, USA, 2008.

[11] S. Bilavarn, J.J. Khan, C. Belleudy, M.K. Bhatti, Effectiveness of Power Strategies for Video Applications: A Practical Study, Journal of Real-Time Image Processing, Springer, jan 2014. (DOI) 10.1007/s11554-013-0394

[12] H. Javaid, M. Shafique, J. Henkel, S. Parameswaran, System-level application-aware dynamic power management in adaptive pipelined MPSoCs for multimedia, Proceedings of the International Conference on Computer-Aided Design, pp. 616-623, 2011.

[13] R. Bonamy, S. Bilavarn, D. Chillet, O. Sentieys, Power Consumption Models for the Use of Dynamic and Partial Reconfiguration, Microprocessors and Microsystems, Elsevier, feb 2014.

[14] M. Horowitz, T. Indermaur, R. Gonzalez, Low-power digital design, In

Low Power Electronics. Digest of Technical Papers., IEEE Symposium (pp. 8-11), 1994.

[15] T. Damak, I. Werda, S. Bilavarn, N. Masmoudi. 2013, Fast Prototyping H.264 Deblocking Filter Using ESL tools, Transactions on Systems, Signals & Devices, Issues on Communications and Signal Processing, pp. 345362, 2013.

[16] 2005. ISO/IEC 14496-10, Advanced Video Coding for Generic Audio-visual Services, ITU-T Recommendation H.264, Version 4. (2005).

[17] Xilinx Zynq-7000 All Programmable SoC ZC702 Evaluation Kit, http://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html (2015).

[18] F. Duhem, F. Muller, R. Bonamy, S. Bilavarn, FoRTReSS: a flow for design space exploration of partially reconfigurable systems, Design Automation for Embedded Systems, Springer Verlag, feb 2015.