

On Line Power Optimization of Data Flow Multi-core Architecture Based on Vdd-Hopping for Local DVFS

Pascal Vivet¹, Edith Beigne¹, Hugo Lebreton¹, and Nacer-Eddine Zergainoh²

¹ CEA-Leti, Minatec, Grenoble, France

² TIMA, Grenoble, France

{pascal.vivet, edith.beigne, hugo.lebreton}@cea.fr,
nacer-eddine.zergainoh@imag.fr

Abstract. With growing integration, power consumption is becoming a major issue for multi-core chips. At system level, per-core DVFS is expected to save substantial energy provided an adapted control. In this paper we propose a local on-line optimization technique to reduce energy in data-flow architecture, thanks to a Local Power Manager (LPM) using Vdd-Hopping for efficient local DVFS. The proposed control is a hybrid global and local scheme which respects throughput and latency constraints. The approach has been fully validated on a real MIMO Telecom application using a SystemC platform instrumented with power estimates. Local DVFS brings 45% power reduction compared to idle mode. When local on-line optimization benefit from computation time variations, 30% extra energy savings can be achieved.

Keywords: Low Power, DVFS, VDD-Hopping.

1 Introduction

In today's System on Chip, power consumption is becoming a major issue. Dedicated mechanisms have been proposed in order to reduce both static and dynamic power consumption at different levels: from technology up to system level. At system level, Dynamic Power Management (DPM) techniques are classically used, such as advanced standby modes or efficient Dynamic Voltage and Frequency Scaling (DVFS). The main difficulty of DPM techniques is to design efficient dedicated control up to application level. Power management is often specific to the low-power design techniques and must take into account architecture and application.

In future multi-cores, the Globally Asynchronous Locally Synchronous paradigm is a natural enabler to help architecture partitioning and facilitate clock and power management [1][12]. In GALS scheme, each IP unit has its own frequency, and communicate asynchronously through a global interconnect. GALS scheme enables local power management: each IP unit is an independent Voltage and Frequency Island (VFI). This is also commonly called "per-core DPM": further energy savings are obtained, since the power optimum is not limited by the most constrained IP core but can be reached independently on each IP cores.

Considering that the energy is square Vdd dependant, DVFS technique is the most promising in terms of overall energy reduction. Due to the usage of external DC-DC converters, today's DVFS techniques are mostly CPU centric and not applied at IP level. Recently, a low cost and efficient DVFS technique, called Vdd-Hopping, has been proposed [2][3]. By using only two external voltages and a dynamic voltage selector switch, DVFS can be efficiently offered locally to each IP core.

In this paper, we target heterogeneous data-flow like architecture with Telecom applications as an example [14]. Regarding the application, execution time variations are decisive. In non real time systems, voltage and frequency selection consists usually in a tradeoff between performance and energy. In case of real time system with data flow application, timing constraints must be met and are twofold: a throughput constraint for each IP and an overall latency constraint on the whole data-flow [4][5][6]. Heuristic algorithms can be used, based on the worst case application scenarios [7][8]. In an heterogeneous architecture using dedicated IP engines, contrary to homogeneous multi-cores, task allocation is static and directly driven by the architecture. In that case, to reduce energy in a multi-application context and benefit from all available dynamic slack time, on-line optimization associated with a fast hardware DPM controller is required [10][11]. The VDD-Hopping technique has been introduced early by T. Zakurai group, which proposed some software control techniques [16] but not yet adapted to hardware heterogeneous architecture.

In this paper, we propose an on-line optimization technique, to reduce energy in data-flow heterogeneous architecture, by using a dedicated DPM controller, which uses the efficient Vdd-Hopping technique for local DVFS. This consists in a hybrid global and local technique, as in [11], which respects throughput and latency constraints, and using only two voltage/frequency points. The proposed technique has been applied to a real GALS NoC architecture targeting MIMO telecommunication applications [14]. Energy savings have been estimated on a SystemC simulation platform which has been instrumented with power estimates [15]. The outline of the paper is as follows: Section 2 introduces the targeted GALS NoC low-power architecture, and Section 3 describes the proposed Vdd-Hopping control for local DVFS. The local on-line optimization is described Section 4. Finally, the experimental results are given in Section 5.

2 Low Power GALS NoC Architecture

The low power overall architecture is organized within a complex GALS NoC fully implemented in asynchronous logic [14]. As shown in Figure 1, each synchronous IP unit of the SoC is integrated with advanced low-power mechanisms, such as in [12]. A programmable Local Clock Generator is implemented within each unit to generate a variable frequency F in a predefined applicative range. A local Power Supply Unit (PSU) manages the local unit voltage V , sharing a power switch between a Vdd-hopping technique and a classical MTCMOS technique. The PSU uses two external voltages with two power switches: V_{HIGH} and V_{LOW} which are automatically switched during DVFS phases. The Network Interface (NI) is in charge of communications with respect to the NoC protocol.

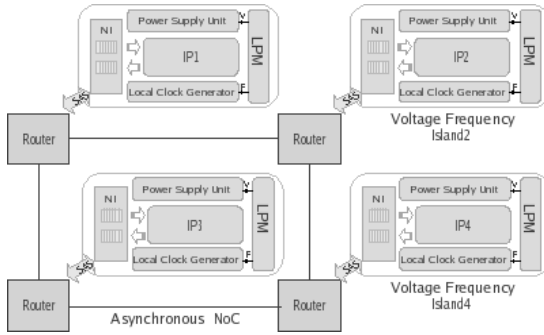


Fig. 1. Low Power GALS NoC overall Architecture

The Local Power Manager (LPM) implements the proposed DPM and on-line optimization techniques. The LPM is activated by the NI in a data-flow manner according to NoC traffic and HW tasks. The NoC architecture targets data flow applications, where task control and complex data flows are handled by the NI. For each executed task, the NI loads a configuration for the IP core and associated input/output data flows, and then computation starts.

2.1 IP Unit Integration for Power Optimization

Each synchronous IP unit is defined as an independent power domain (using its dedicated local voltage V) and an independent frequency domain (using its dedicated local clock frequency F). Each IP unit can be set in one of the 4 power supply modes:

- *HIGH* mode, local supply voltage V is V_{HIGH} and core clock is on. This is the “nominal” high performance working mode.
- *LOW* mode, core clock is on, but supply is switched to V_{LOW} . Frequency is lower than nominal, energy per cycle decreases. This is “low power” mode.
- *IDLE* mode, core clock is off and leakage power is further reduced thanks to V_{LOW} supply voltage. This is the “low-power dormant” mode.
- *OFF* mode, the unit is switched off when not used in the application, to further reduce the leakage power.

For each unit, all power modes can be programmed through the Network Interface and the Local Power Manager, except the OFF mode which is programmed through top level signals (main CPU).

2.2 Local DVFS Using Two Voltages Set Points

In order to perform efficient local Dynamic Voltage Scaling (DVS), the main objective is to avoid as much as possible low-level software control to ensure minimal latency cost. Within the Power Supply Unit, a hardware controller called Vdd-hopping automatically switches between V_{HIGH} and V_{LOW} (Figure 2).

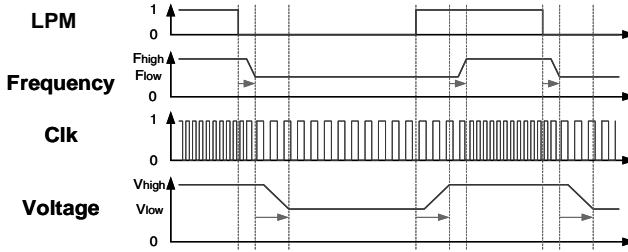


Fig. 2. LPM control, Vdd-Hopping sequence example

During smooth DVFS transitions, the synchronous IP can continue its own computations or communications. To obtain an average value between V_{HIGH} and V_{LOW} , the LPM controls the target performance by switching between these 2 values. The power efficiency of the proposed Vdd-Hopping [2] is more than 95%. In a given V_{HIGH} or V_{LOW} voltage, there are no losses except those in a standard power switch; there are only energy losses during the transitions (less than 100 ns). There is no latency cost, and no need for real time software, fast and robust transitions are ensured by hardware. The VDD-Hopping mechanism has been implemented and validated in a test-chip in 65nm [13], which prove high reliability. In order to minimize energy per operation, the IP unit should run at maximum achievable frequencies f_h and f_l . The LPM objective is then to spend more time at V_{LOW} to decrease energy, while respecting timing constraints. The proposed hybrid local and global DVFS principle and associated LPM schemes are introduced in next section.

3 Local DVFS Control

On data flow architectures with latency constraint on the whole chain, a global management is required to ensure the deadline. In order to guarantee latency, due to dynamic variations of the computation on each core, centralized control or software control cannot be done since it would not respond fast enough to handle all the dynamic variations. We choose a Worst Case Execution Cycle (WCEC) based static management to select a set point for each task. A heuristic based algorithm, as in [7], can be used. To benefit from dynamic slack time induced when the actual number of cycles to complete a task is less than WCEC, a local control is implemented. Such a hybrid (local and global) approach has also been adopted in [11].

Based on worst case, a global power manager (such as the host processor) dispatches the available latency among tasks. Hence, each core is given a timeslot to complete its task. For each IP core, its Local Power Manager (LPM) controls the Vdd-Hopping by spreading the computation over the given timeslot. The LPM is activated by the NI in a data-flow manner according to NoC traffic and HW tasks. Two control schemes are proposed and presented below, with NI task or IP Core task synchronization. One must notice that NOC bandwidth must be enough to tolerate uncorrelated IP frequencies variations, to smooth applicative traffic, hypothesis which is respected in the addressed application and corresponding NOC (see section 5).

3.1 NI Task Synchronization

The first proposed solution interacts with the NoC platform programming model to control the power modes, in a generic way. As soon as a new task is loaded in the NI, the Vdd-Hopping transitions can start. The LPM control of the IP is thus activated in a data-flow manner according to the NoC incoming traffic and task.

Given the WCEC N_{wcec} , the number of cycles to spend at high voltage N_h and at low voltage N_l can be derived from the given timeslot τ for the task. Let f_h and f_l be the maximum available frequency at respectively high set point and low set point, we have:

$$\tau = \frac{N_h}{f_h} + \frac{N_l}{f_l} = \frac{N_h}{f_h} + \frac{N_{wcec} - N_h}{f_l} \tag{1}$$

For the task computation, the number of cycles at high and low level is given by:

$$N_h = \frac{f_h}{f_h - f_l} (N_{wcec} - \tau \times f_l) \quad \text{and} \quad N_l = N_{wcec} - N_h \tag{2}$$

The timeslot is equivalent to a mean frequency: $f_{target} = \frac{N_{wcec}}{\tau}$.

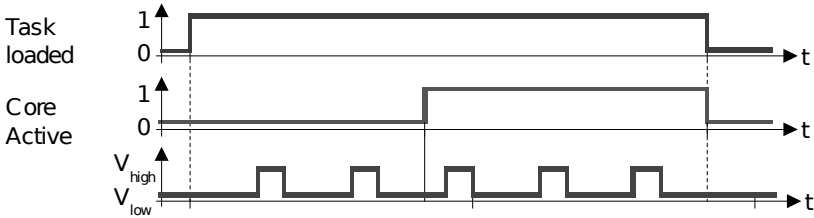


Fig. 3. NI task synchronization

The LPM switches periodically from high to low while the task is loaded in the NI, so that the target frequency is reached when the core is actually computing (Figure 3). If the hopping frequency is increased while keeping the N_h and N_l ratio, the mean frequency is not modified and NoC traffic is smoothened. Since extra energy is consumed during transitions, a tradeoff is required between transition number, NoC traffic regularity and energy.

Lastly, if the targeted frequency is lower than the fastest frequency at Vlow, the frequency is decreased (this is DFS at Vlow). Finally, as seen Figure 3, task loading in the NI may not match the actual computation phase, because the IP core may wait for additional data before starting. In that case, extra energy could be saved thanks to a tighter control.

3.2 Core Task Synchronization

Better control is obtained if LPM is synchronized with actual IP core computation. In this case, a dedicated signal must be generated by the IP core to indicate its own

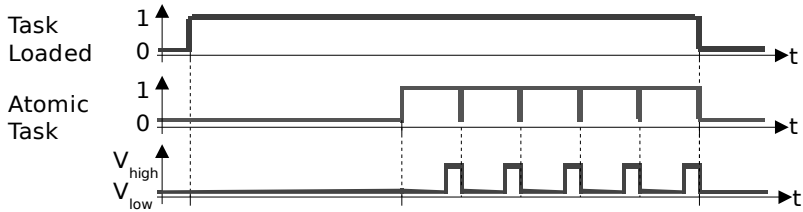


Fig. 4. Core task synchronization

activity/inactivity. The number of cycles N_h and N_l are still calculated as described in section 4.1.

Instead of controlling with NI task activity, the LPM performs the Vdd-Hopping transitions with IP core task activity. An atomic task is defined when the number of cycles and the number of input/output data are known. In order to balance the frequency of hops, the LPM is able to perform switching over several atomic tasks or within a single task.

In case the actual number of cycles of the atomic task is less than the worst case, it is possible to start the computation at low level [17]. In Figure 4, the NI task consists of five atomic tasks, with only one transition low to high done within each task. The unit gets back to low level as soon as the task is completed; most of the computation is spent at low level.

4 Local On-Line Optimization

The Actual number of Execution Cycles (AEC) needed by a task may be less than the WCEC. The computation time may depend on data, the communication time is variable and the architecture can have unpredictable events such as cache defaults, leading to dynamic slack time. The LPM can exploit this dynamic slack time by reducing the speed of the unit. Even though it is possible to predict the number of cycles for next task from the execution history, this approach may not meet the timing constraints. A prediction mistake will induce a timing violation. We rather assume the current task still runs at WCEC and benefit from the dynamic slack time from the previous task. The cycle budgets at high and low levels are updated according to the remaining cycles at high and low levels.

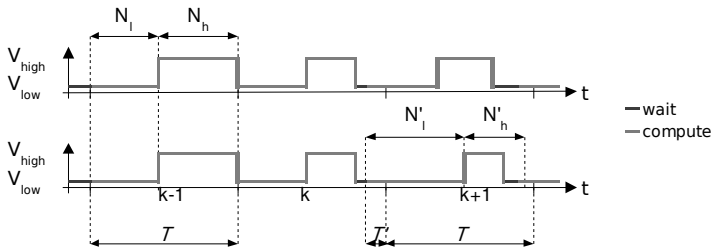


Fig. 5. Local on-line optimization principle

Figure 5 presents the on-line optimization principle. The first chronogram shows the LPM control without online optimization. The second uses the online optimization. The first task $k-1$ runs at WCEC while the following tasks do not use as much cycles. In this case, the third task is slowed down while respecting the deadline.

When a task k is over and cycles are remaining, respectively n_h at high level and n_l at low level, the unit switches to low level and keeps on counting the number of elapsed cycles. Hence, when the next task starts, the remaining cycles n_h and n_l reflect the dynamic slack time. The timeslot for the following task is extended to:

$$\tau + \tau' = \tau + \frac{n_h}{f_h} + \frac{n_l}{f_l} \quad (3)$$

The updated number of cycles N_h' is then given by:

$$\begin{aligned} N_h' &= \frac{f_h}{f_h - f_l} (N_{wcec} - (\tau + \tau') \times f_l) \\ &= N_h - \frac{f_h}{f_h - f_l} n_l - \frac{f_l}{f_h - f_l} n_h \end{aligned} \quad (4)$$

Thus, before the next task ($k+1$) begins, we compute its parameters with the extended time:

$$\begin{cases} N_h' = N_h - \frac{f_h}{f_h - f_l} \left(n_l + \frac{f_l}{f_h} n_h \right) \\ N_l' = N_{wcec} - N_h' \end{cases} \quad (5)$$

The above equations provide the main principles of the on-line optimization algorithm. In order to implement such control efficiently in the hardware LPM controller, some simplifications are required. The LPM requires mainly two counters to keep track of the number of elapsed cycles in high and low voltage. In order to have simple hardware, the computations of both ratios $f_h/(f_h - f_l)$ and f_l/f_h must be either done in software or simplified to be done in hardware. The new budget N_h' should not be overestimated; otherwise the deadline might be violated. If those ratios are underestimated, then the efficiency is reduced. Assuming $f_h = 2f_l$, we obtain the following simplified equations for the updated cycle budgets at V_{HIGH} and V_{LOW} with regard to dynamic slack time of previous task:

$$\begin{cases} N_h' = N_h - 2 * (n_l + 0,5n_h) \\ N_l' = N_{wcec} - N_h' \end{cases} \quad (6)$$

The LPM controller is then programmed with the two input parameters: the timeslot τ , and the target N_{WCEC} . It implements two counters, and it can be implemented as a simple state machine to control any of the AEC mode, the NI mode or the CORE mode.

The LPM controller has been fully modeled in SystemC. From the algorithmic complexity and the number of registers, the LPM is estimated to be less than 2K gates. The area cost of the PSU including the Vdd-Hopping is 3% of the core area for a 200K gates IP core.

5 Case Study on a 3GPP LTE Telecom Application

The targeted application and circuit [14] is based on the 3GPP LTE telecommunication protocol; we focus on the baseband demodulation of the downstream. Once the application is mapped onto the NoC architecture, the application is divided into several sequential phases, a whole frame is constituted of 14 OFDM symbols. There are three main phases and each phase is separated by memory buffering. The IP core tasks are periodic and sequenced in a data flow manner (Figure 6).

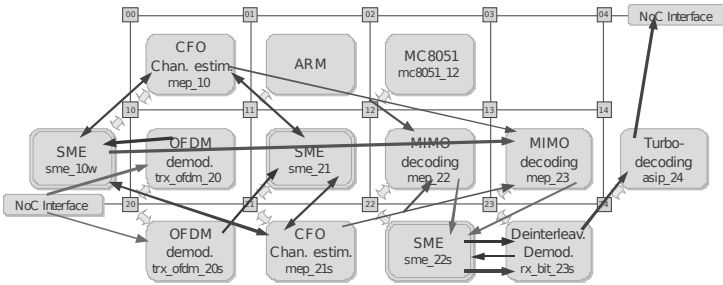


Fig. 6. Task mapping on the Low Power GALS NoC architecture

The GALS NoC architecture is build with dedicated hardware engines, such as TurboCode, RX/TX bit engines, OFDM modulation/demodulation, MEP engines (advanced configurable VLIW-like core) and finally some SME (Smart Memory Engine) used to handle memory buffers. Each IP core is encapsulated with a PSU, a LPM and a LCG providing 16 frequencies in the [400MHz-1GHz] range, with additional scaling factors.

5.1 Simulation Platform and Applicative Scenarios

The simulation platform used to qualify the energy savings is based on an existing timed SystemC/TLM platform. The power consumption has been included in the simulation platform, along with DVFS modeling [15]. The SystemC model takes into account leakage current, dynamic power, the inactivity phase's consumption, and the variation of energy per operation due to Vdd-Hopping. For each IP blocks, power consumption values have been extracted from post Place&Route gate-level simulation thanks to PrimePower® tool. As a conclusion, fast power estimation and exploration at high level can be performed on a real application. The tool provides power profile traces (in vcd) and power statistics (per core, per mode, ...).

For the targeted 3GPP-LTE application, the global constraints (timeslot and NWCEC) have been derived manually for each IP, to enforce throughput and latency constraints. For all proposed LPM scenarios (Table 1), except the first two ones, IDLE mode is used as soon as end of task is reached. All scenarios respect the application timing constraints, except the first one at *Low* level, which is given as a reference.

Table 1. Power Mode Scenarios

<i>Low</i>	LOW mode at maximal achievable f_{low}
<i>High</i>	HIGH mode at maximal achievable f_{high}
<i>On/Off</i>	HIGH mode at f_h max, and IDLE when tasks complete
<i>DFS</i>	HIGH mode using only Dynamic Frequency Scaling
<i>DVFS NI</i>	DVFS synchronized with NI
<i>DVFS Core</i>	DVFS synchronized with CORE
<i>DVFS AEC</i>	DVFS synchronized with CORE, plus on-line optimization using Actual Execution Cycle

5.2 Obtained Energy Savings

For each LPM scenario, power profiling has been done, the achieved energy savings are presented per IP (Figure 7).

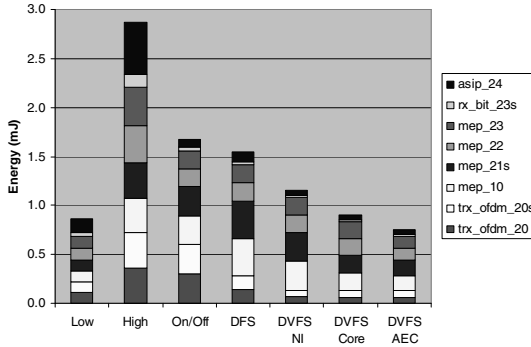


Fig. 7. Energy consumption per IP Core

The On/Off scenario exhibits substantial energy savings, thanks to the efficiency of IDLE mode (we recall that IDLE is done with IP clock gating at Vlow). When only using DFS, there is almost no gain since the computation is only spread over time, reducing peak power but not energy. When using DVFS, we observe that energy savings clearly depend on core profile. For under-constrained cores (trx-ofdm cores) with low target frequency, DVFS enables high energy savings compared to On/Off scenario. Synchronization with task loading is relevant as these units does not spend time waiting for data. These units have a steady number of computation cycles, and online optimization is useless. Synchronization of DVFS with core computation will bring benefits when the IP cores wait for a long time incoming data (mep_10, mep_21s). For more constrained cores (mep_22, mep_23) with high target frequency, when they require less cycles than the predicted WCEC to complete their task, local optimization is relevant. For tasks with target frequency close to f_h , up to 30% energy savings has been achieved compared to simple core synchronization. We exhibit 45% extra energy savings with DVFS AEC compared to On/Off scenario.

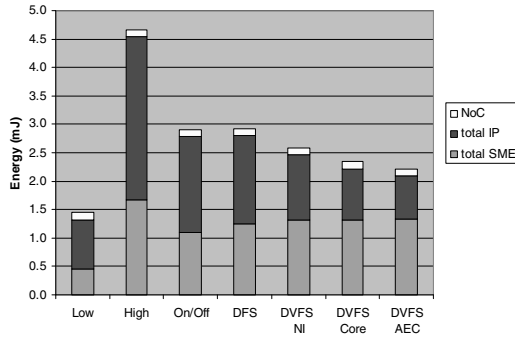


Fig. 8. Energy savings for NoC, SME and IP Cores

In Figure 8 is given power consumption for the whole SoC, considering HW IPs, SME IPs and the NoC. The NoC power consumption represents only 5% of the total power consumption, and is slightly equivalent for each scenario. The advanced IDLE mode from On/Off scenario brings 35% power reduction on the whole chip. As a global result, the power reductions obtained on the IP Cores (Figure 7) are mitigated due to inefficient power reduction on Smart Memory Engines. Because SMEs do not actually perform computation but must run fast enough to handle data traffic, a power control based on traffic arrival like in [10] should be efficient. Finally, the total chip budget is reduced from 340 mW at full speed (High mode) to 160 mW using the DVFS scheme with on-line optimization.

6 Conclusions

In this paper, we presented a new Local Power Manager unit to reduce energy in a data-flow heterogeneous architecture by using the Vdd-Hopping technique. The Vdd-Hopping is an efficient DVFS technique with only two set points and zero overhead, which can be easily integrated for per-core DVFS. In the proposed LPM, we use a hybrid local and global scheme to enforce timing constraints, a LPM synchronization scheme with core computation to benefits from all inactivity phases, and an on-line optimization technique to distribute dynamic slack time. Energy savings have been qualified on a real application, using a SystemC platform instrumented with power. Results show that advanced idle mode achieves significant energy savings (35%). As expected, DFS achieves few energy savings. DVFS enables to reduce energy by 45% compared to IDLE mode. Finally, when number of cycle per-task varies, 30% additional energy savings are achieved by local on-line optimization. Future work will address the design of an efficient DVFS control for SMEs, the RTL design of the LPM, as well as HW task automatic profiling.

References

1. Bhunia, S., Datta, A., Banerjee, N., Roy, K.: GAARP: A Power-Aware GALS Architecture for Real-Time Algorithm-Specific Tasks. *IEEE Transactions on Computer, Special Issue on low-Power Design (99)*, 752–766 (June 2005)

2. Sylvain, M., Vivet, P., Renaudin, M.: A Power Supply Selector for Energy- and Area-Efficient Local Dynamic Voltage Scaling. In: Azémard, N., Svensson, L. (eds.) *PATMOS 2007*. LNCS, vol. 4644, pp. 556–565. Springer, Heidelberg (2007)
3. Truong, D., et al.: A 167-processor 65 nm Computational Platform with Per-Processor Dynamic Supply Voltage and Dynamic Clock Frequency Scaling. In: *Proc. Symposium on VLSI Circuits* (June 2008)
4. Mishra, R., Rastogi, N., Zhu, D., Mosse, D., Melhem, R.: Energy aware scheduling for distributed real-time systems. In: *Proc. of Parallel and Distributed Processing Symposium* (April 2003)
5. Watanabe, R., Kondo, M., Imai, M., Nakamura, H., Nanya, T.: Task Scheduling under Performance Constraints for Reducing the Energy Consumption of the GALS Multi-Processor SoC Design. In: *DATE 2007* (2007)
6. Xian, C., Lu, Y., Li, Z.: Energy-Aware Scheduling for Real-Time Multiprocessor Systems with Uncertain Task Execution Time. In: *DAC 2007*, pp. 664–669 (2007)
7. Grosse, P., Durand, Y., Feautrier, P.: Methods for Power Optimization in SoC-based Data Flow Systems. *ACM Transactions On Design Automation of Electronic Systems (TODAES 2009)* 14(3), Article No. 38 (2009)
8. Niyogi, K., Marculescu, D.: Speed and voltage selection for GALS systems based on voltage/frequency islands. In: *Proceedings of, ASP-DAC 2005* (2005)
9. Puschini, D., Clermidy, F., Benoit, P., Sassatelli, G., Torres, L.: Temperature-Aware Distributed Run-Time Optimization on MP-SoC Using Game Theory. In: *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2008*, pp. 375–380 (2008)
10. Alimonda, A., Acquaviva, A., Carta, S., Pisano, A.: A Control Theoretic Approach to Run-Time Energy Optimization of Pipelined Processing in MPSoCs Design. In: *Proceedings of Design Automation and Test in Europe, DATE 2006* (2006)
11. Maxiaguine, A., Chakraborty, S., Thiele, L.: DVS for buffer-constrained architectures with predictable QoS-energy tradeoffs. In: *3rd International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2005*, pp. 111–116 (2005)
12. Beigné, E., Clermidy, F., Miermont, S., Vivet, P.: Dynamic Voltage and Frequency Scaling Architecture for Units Integration within a GALS NoC. In: *Proceedings of NOCS 2008* (2008)
13. Beigné, E., et al.: An Asynchronous Power Aware and Adaptive NoC based Circuit. *IEEE Journal Of Solid State Circuits* 44, 1167–1177 (2009)
14. Clermidy, F., et al.: A 477mW NoC-Based Digital Baseband for MIMO 4G SDR. In: *Proceedings of IEEE International Solid-State Circuits Conference, ISSCC 2010* (2010)
15. Lebreton, H., Vivet, P.: Power Modeling in SystemC at Transaction Level, Application to a DVFS Architecture. In: *Proc. of Int. Symposium on VLSI, ISVLSI 2008*, pp. 463–466 (2008)
16. Soongsoo, L., Sakurai, T.: Run-time Voltage Hopping for Low-Power Real-time Systems. In: *Proc. of 37th Design Automation Conference, DAC 2000*, pp. 806–809 (June 2000)
17. Yan, Z., Zhijian, L., Lach, J., Skadron, K., Stan, M.R.: Optimal procrastinating voltage scheduling for hard real-time systems. In: *DAC 2005*, pp. 905–909 (June 2005)