An Energy-Efficient Architecture for Nanometric Technologies with Strong Robustness to Process Variability : Design of a GALS node based on a MIPS R2000 processor

Sylvain Durand, H. Zakaria, Laurent Fesquet, Nicolas Marchand

An Energy-Efficient Architecture for Nanometric Technologies with Strong Robustness to Process Variability:
Design of a GALS node based on a MIPS R2000 processor

Sylvain Durand, Hatem Zakaria, Laurent Fesquet and Nicolas Marchand

***Abstract*—In this paper we present an energy-efficient solution for nanometric systems, where some variability problems which did not influence the circuit at a higher scale introduce some uncertainties at a sub-micrometric size. Therefore, some advanced control strategies are required to manage the energy-performance tradeoff in such an environment. The setup is based on some Dynamic Voltage and Frequency Scaling (DVFS) techniques applied to a Globally Asynchronous Locally Synchronous (GALS) architecture. Whereas a Vdd-hopping converter is able to provide discrete voltage levels, a Programmable Self-Timed Ring (PSTR) oscillator allows a variability robust source for generating adjustable clock frequencies. A fast predictive control law is also developed in this paper to calculate the frequency and voltage levels to apply to these actuators, minimizing the high voltage running time while ensuring good computational performance. Finally, the proposal is dedicated to a GALS node based on a MIPS processor over STMicroelectronics $45\,nm$ technology and some fine-grained simulation results are performed. Besides a noticeable reduction of the energy consumption, the performance of the closed-loop system is ensured and a strong robustness to technological variability is also demonstrated.**

***Index Terms*—Energy-performance tradeoff, nanometric GALS architecture, process variability robustness, fast predictive control, programmable self-timed ring oscillator.**

***EDICS Category*—CTRL100A5, POW170B0, DCS160-210.**

INTRODUCTION

The upcoming generations of embedded integrated systems in multimedia and telecommunication applications require several drastic technological evolutions since they have reached limits in terms of power consumption, computational efficiency and fabrication yield. As a result, the main problems that we are facing nowadays with the nanometric technologies can be categorized in three main points:

a) Variability refers to the unpredictability, inconsistency, unevenness, and changeability associated with a given feature or specification. At a sub-micrometric scale, variability has become one of the leading causes for chip failures and delayed schedules. Indeed, in nanometric design flows, variability is associated with design modes, power states, environmental conditions, manufacturing steps, and the behavior of devices and interconnects. Variability affects the entire physical design environment, from power management, through timing and signal integrity closure to manufacturability. A major problem facing the computer and semiconductor industries is the increasing CMOS process variability. In low-level circuit parameters, such as transistor gate length and gate oxide thickness, variability complicates system design by introducing uncertainty about how a fabricated system will perform [1]. Although a circuit or chip is designed to run at a nominal clock frequency, the fabricated implementation may vary far from this expected performance (see Fig. 1).

b) Leakage power is a growing concern in the overall design process. As voltages scale downward with geometries, threshold voltages must also decrease to gain performance advantages of the new technology. This reduction in threshold voltages has led to an exponential increase in leakage current in transistors. Thinner gate oxides have led to an increase in gate leakage current as well. At $65\,nm$ and below, leakage power accounts for a significant portion of the total power in high-performance designs [2], therefore its management is essential in the ASIC design process. Unlike dynamic power, which can be managed by reducing switching activity, leakage power effect exists as long as the power is on. That is why current process technologies are pushing designers to consider new design methods to reduce leakage power.

c) Yield success is much harder to achieve because of the increasing number and complexity of variables affecting manufacturability. Early on, the path to yield on integrated circuit design was fairly simple: comply with all the design rules and yield would more or less follow. Designers did not need to worry too much about what happened in the fabrication after tape-out [3]. Nevertheless, the game has changed in the nanometric era. The designer's strategy must now shift from simple design rule compliance to the definition and design of the optimal layout for the highest yield.

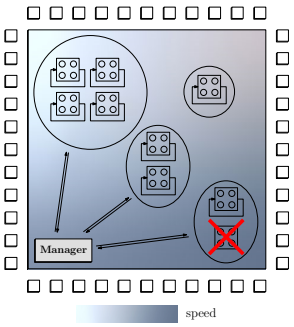


Fig. 1. Representation of the technological variability's phenomenon which appears in sub-micrometric integrated circuits: whereas a part of the chip could run with the expected computational speed, another part may run more slowly and another might not run at all. A controller has hence to manage these different domains to yield the best performance.

S. Durand is with the NeCS project-team, an INRIA Rhône-Alpes and GIPSA-lab joined team, Grenoble, France (sylvain.durand@inria.fr)

H. Zakaria is with the Electrical Engineering Department, Benha Faculty of Engineering, Benha University, Benha, Egypt (hatem.radwan@bhit.bu.edu.eg)

L. Fesquest is with the CIS Group, TIMA Laboratory, Grenoble University, Grenoble, France (laurent.fesquet@imag.fr)

N. Marchand is with the SysCO team, Control Department of GIPSA-lab, Grenoble University, Grenoble, France (nicolas.marchand@gipsa-lab.fr)

Regarding the previously mentioned motivations, one can conclude the suggested solutions in two main points. First, the chips in advanced technologies require a dynamic power management in order to highly reduce the energy consumption. Also, some architectural issues are needed for helping the yield enhancement of such circuits with strong technological uncertainties.

A. Power management technique

In today's technology nodes, *leakage power* is a significant contributor to the total power, as the gate length and threshold voltage are scaled down. Several techniques can be applied at the circuit level to reduce leakage power, including multi-threshold libraries, power gating and variable body biasing [4], [5]. However, in current CMOS integrated circuits, the average power consumption and the energy dissipation are dominated by the *dynamic power*, that arises from the charging and discharging of the load capacitance:

$$\begin{aligned} P_{avg}(t) &\propto C f_{clk}(t) V_{dd}(t)^2 \\ E(t) &\propto C V_{dd}(t)^2 \end{aligned} \quad (1)$$

where C is the load capacitance, V_{dd} is the supply voltage and f_{clk} is the clock frequency. These equations suggest that minimizing the load capacitance, reducing the supply voltage or slowing the clock can reduce power and energy. While the load capacitance can only be reduced during chip design (for example by minimizing on chip routing capacitances and reducing external component access), voltage scalable processor and power controllable peripheral devices make possible to manage power by a dedicated digital hardware or by an Operating System (OS). For instance, the OS can control the processor frequency and its voltage and/or put the devices in low-power sleep states using a dynamic power management [6]. The power minimization can be achieved by resolving an off-line stochastic optimization problem but this is not always possible. Therefore the optimization has to be performed on-line by the control system dedicated to the power management. These techniques run slower the processors at a reduced voltage according to the instantaneous computational demand.

As previously shown in (1), reducing the frequency f_{clk} and the supply voltage V_{dd} decreases the energy and the power. The energy reduction is a quadratic function of the voltage and a linear function of the clock frequency. As a result, Dynamic Voltage Scaling (DVS) can be used to efficiently manage the energy consumption of a device [7]. Supply voltage can be reduced whenever slack is available in the critical path, but one has to take care that scaling the voltage of a microprocessor changes its speed as well. Therefore, adapting the supply voltage is very interesting when possible but implies the use of Dynamic Frequency Scaling (DFS) to keep correct the system behavior. The addition of DFS to DVS is called Dynamic Voltage and Frequency Scaling (DVFS) and results in simultaneously managing the frequency and the voltage. In many cases, the only performance requirement is that the tasks meet a deadline, like depicted in Fig. 2(a) where a given task has to be computed before a given time. Such cases create opportunities to run the processor at a lower computing level and achieve the same perceived performance while consuming less energy. Fig. 2(b) shows that decreasing the processor clock frequency reduces power consumption but simply spreads the computation out over time, thereby consuming the same total energy as before. Fig. 2(c) shows that reducing the voltage level as well as the clock frequency achieves the desired goal of reduced energy consumption at an appropriate performance level [8].

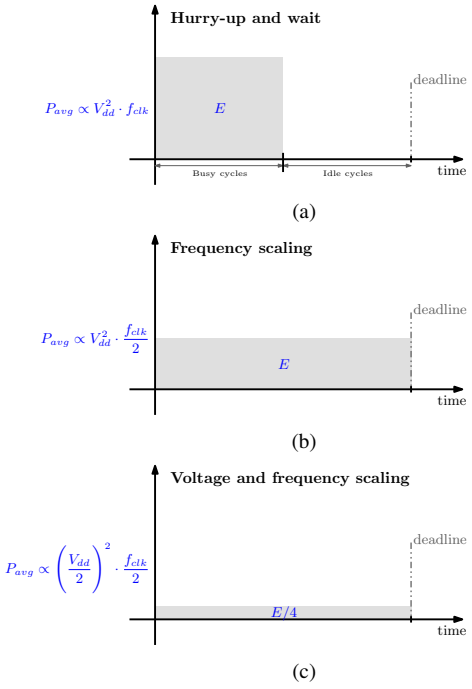


Fig. 2. Dynamic power management: energy consumption vs. power consumption. Whereas decreasing the frequency reduces the instantaneous power (but yields the same energy consumption), reducing both the voltage and frequency leads to a drastic energy saving with similar performance.

Several behaviors are known to minimize the energy consumption while guaranteeing good computational performance (see [9] for further details). Classically, each task has to be considered independently. Thus, when several tasks – with their own and different computational load – have to be executed, the voltage level has to be calculated for each one rather than considering a global scaling for all the tasks. Moreover, the execution time has to fit with the deadline regardless the chip runs with a continuously or a discretely varying voltage range. When only the maximal voltage level is possible – as this is the case for a classical processor (without DVS) – the hurry-up and wait running is performed but leads to an important energy consumption. On the other hand, a continuously varying voltage scaling allows the use of a unique level to fit with the deadline. The consumption is hence optimal and cannot be decreased anymore. Nevertheless, a discretely varying voltage scalable processor can also reduce the energy consumption by using the available voltage levels to fit the task with its deadline. In this case, the lowest energy consumption is achieved using the two immediate neighbors of the optimal level. Another essential rule is that selecting some suitable voltage levels leads to a drastic energy reduction

even if the number of levels is very small. In this case, a frequency range is available for each voltage level [10], which can be useful to fit the task with its deadline. Alternatively, the *clock-gating* technique can be implemented. To save power, the clock-gating technique adds more logic to a circuit to prune the clock tree, thus disabling some portions of the circuitry in order their switching power consumption goes to zero [4]. In fact, this behavior leads to "pause" the clock when required to save energy. Typically, the clock is paused when a task is performed before its deadline.

A task scheduling is also a solution for power management. In [11] for instance, a simple method consists in monitoring the total activity of the chip – without knowing task by task information – and in applying a high voltage when the processor is busy or a low voltage when the computation load is low. In [12], [13], the different tasks are sorted into three possible throughput (the number of instructions to treat in a given amount of time): some *compute intensive and short-latency processes* for tasks which require the maximal performance and will be executed with a high voltage level and some *background and high-latency processes* for non-critical tasks which will be computed at low voltage. The tradeoff between a maximal throughput and a minimal energy consumption is thereafter the key point. More sophisticated scheduling policies propose to integrate a DVFS scheduler and a feedback controller, as proposed in [14] where a (earliest deadline first) scheduler is mixed with a (proportional integral derivative) controller. As a result, closed-loop control laws are required to manage the energy-performance tradeoff in electronic devices and some new strategies are developed in this sense in this paper. The main idea consists in ***i)*** reducing the penalizing supply voltage so far as possible in order to minimize the energy consumption and ***ii)*** adapting the clock frequency to the computational load to fit the task with its deadline.

B. Globally Asynchronous Locally Synchronous (GALS) paradigm

Embedded integrated systems have two means of implementation. Firstly, the conventional clocked circuits with their global synchronization – in which one faces the huge challenge of generating and distributing a low-skew global clock signal and reducing the clock tree power consumption of the whole chip – makes them difficult for implementation. Secondly, Systems-on-Chip (SoCs) built with predesigned IP-blocks running at different frequencies need to integrate all the IP-blocks into a single chip. Therefore, global synchronization tends to be impractical [15]. By removing the globally distributed clock, GALS circuits provide a promising solution for SoC design. Such an architecture is depicted in Fig. 3. GALS techniques allow each locally synchronous module to be set independently, making voltage and frequency scaling more convenient than with the standard synchronous approach. Moreover, a GALS architecture can mitigate the impact of process and temperature variations, because a globally asynchronous system does not require that the global frequency was dictated by the longest path delay of the whole chip, i.e. the critical path. In this case, each clock frequency is only determined by the slowest path in its voltage/frequency domain.

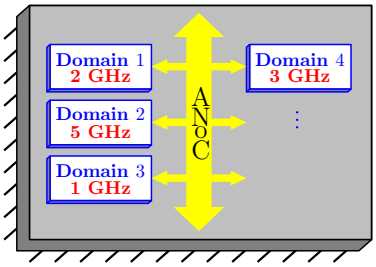


Fig. 3. Globally Asynchronous Locally Synchronous (GALS) architecture. In fact, GALS chips are split into multiple frequency domains, where each domain is synchronous with respect to its clock. The different domains are mutually asynchronous – thanks to an Asynchronous Network-on-Chip (ANoC) – in that they may run at different clock frequencies.

Structure of the paper

The paper is organized as follows. In section I, we explain why a closed-loop architecture becomes essential in nanometric technologies. We present a three overlapped loop scheme and introduce a practical example giving some architectural solutions to overcome the barrier of sub-scale systems. Then, a robust control setup is depicted in section II for the microprocessor MIPS R2000, detailing the actuators, sensors and control strategy of such a closed loop. Finally, some fine-grained simulation results are presented in section III using *Modelsim*.

I. Controlling uncertainty and handling the process variability

In a reconfigurable GALS system, the process variability and the fabrication yield can be improved by smartly removing some tasks over fault or some low performance frequency domains and assign them into other high performance ones. Whereas each performance can be measured by a sensor, a global system manager is able to distribute the tasks over the different computational nodes. The task assignment takes into account the node performance and the task processing load. The main target of this manager is then to ensure an overall chip performance. As a result, it is no more required to separately guaranty a performance for each domain with this kind of approach, which hence relaxes the fabrication constraints and permits a yield enhancement. An important conclusion is that control loops become essential in advanced technologies. This is more detailed in subsection I-A. A case study is then proposed in subsection I-B.

A. Essential feedback control loops

In a multiprocessor GALS system, one can choose to slow down some parts of the circuit while allowing others to operate at the maximal frequency. This enables more energy saving opportunities than conventional systems built around one processor and allows adapting the clock speed to the local process quality. Moreover, it has also been shown that

multiple-clock designs with voltage scaling are even better not only in terms of power and performance, but also in terms of variability [16]. As a result, building a system based on the implementation of hardware resources whose performance are unpredictable at the fabrication time requires to have some global management strategies, adapting the voltage/frequency in order to respect real-time constraints of the application and the allocated energy budget. Therefore, it is proposed to use automatic feedback loops based on ***i)*** the measurement of the real local performance and the actuation of the voltage/frequency variables (hardware level) as well as ***ii)*** the suitable hardware resource allocation for the execution of a task in the assigned time/energy budget (operating system level). Then, the idea is the use of DVFS techniques with task scheduling to dynamically manage not only the energy budget but also the activity of the processing nodes. Some advanced control strategies will allow an optimal regulation of the frequency/voltage converter according to the computational load and the load distribution in the various GALS processors. Furthermore, in order to compensate for the process variation due to the technology dispersion, and optimize the operation of the circuit, the dynamic voltage/frequency regulation should be self-adjustable with variable loads and dispersion models and robust against process variability.

One of the main points of interest of the proposal is to handle the uncertainty of a processing node over a GALS system and also to reduce its energy consumption by means of automatic control methods. Activity sensors are supposed to be embedded in each processing unit. These sensors provide a real performance measurement of different processing nodes after the fabrication process, which will be used afterwards by the operating system to distribute tasks over different nodes. This means the need for the rescheduling of tasks in each processing node to meet the new assigned deadlines, and this will be achieved by controlling its voltage/frequency, which in accordance will control its consumption of energy. The control system is based on three overlapped control loops, applied in different architecture levels, which are depicted in Fig. 4:

1) Control of the processing power (supply voltage and clock frequency).
2) Control of the tradeoff between energy consumption and computational performance.
3) Control of the quality of the running application.

Voltage, frequency and energy control loops are used in order to adapt the energy consumption and the process variability effect. The other loop is needed to deal with the Quality of Service (QoS) at the application level, with the limitation of processing power and/or channel of communication and with some constraints in energy consumption. This latter loop also manages the different domains of the chip (denoted *cluster* in Fig. 4) with their own performance.

The different control loops are nested. Actually, the operating system (or the high-level loop) provides a set of information – required computational speed, in terms of number of instructions and deadlines for each task to treat – which can be statically inserted into the code or dynamically computed at run time by the OS. These information about

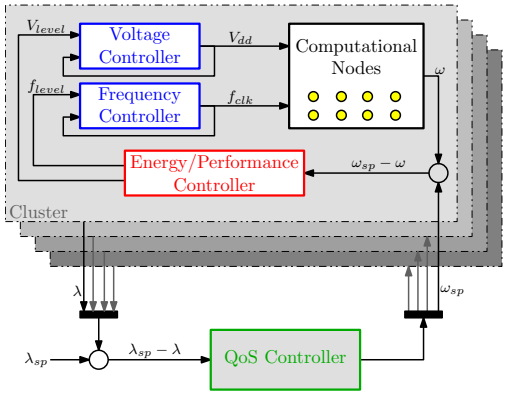


Fig. 4. Automatic control are essential in nanometric chips to manage the computational activity and energy consumption. Three overlapped loops are hence introduced: the voltage and frequency control (low-level), the energy-performance tradeoff control (middle-level) and the applicative quality of service control (high-level).

the real-time requirements of the applications enable to create a computational load profile with respect to time. There are also sensors embedded in each processing unit in order to provide real-time measurements of the processor speed. Consequently, combining such a profile with such a monitoring makes possible to apply a fine-grain power/energy management allowing application deadlines to be met. The DVFS hardware part (or low-level loop) contains voltage/frequency converters, such as a DC-DC converter and a programmable clock generator. Then, a controller (the one from the middle-level loop) dynamically controls these power actuators to scale the supply voltage as well as the clock frequency, in order to satisfy the application computational needs with an appropriate management strategy. Of course, this controller should have a strong robustness against process variability for a correct system behavior. These solutions – the three overlapped feedback control loops with some specific sensors to monitor the activity of the chip – were established within the ARAVIS project context which is now introduced.

B. Case study overview: the ARAVIS project

Using both a Network-on-Chip (NoC) distributed communication scheme and a GALS approach offers an easy integration of different functional units thanks to a local clock generation [17]. Moreover, it allows better energy savings since each functional unit can easily have its own independent clock frequency and voltage. Hence, such an architecture appears as natural enablers for distributed power management systems as well as for local DVFS. A detailed block diagram for modeling DVFS and local process quality management in a GALS system is shown in Fig. 5. This practical example is part of the ARAVIS project which aims at providing architectural solutions for manufacturing circuits in nanometric technologies with strong technological uncertainties.

Since in advanced technologies, the associated processor leakage has an important contribution to the system energy consumption, a sleep mode management block is used to put useless processors into a sleep mode in order to limit

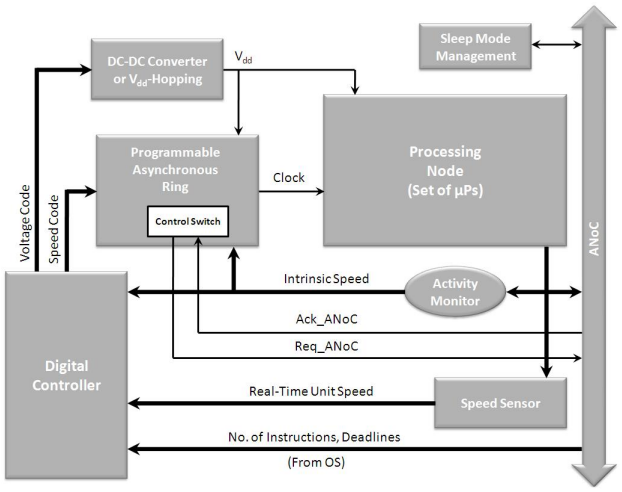


Fig. 5. Energy-performance management architecture proposed in the ARAVIS project to control a multi-processor system with high process variability.

their static power consumption. On the other hand, some speed sensors are used to calculate the real-time computational speed of the processing units (in Million of Instructions Per Second for instance). Activity monitors disseminated into each voltage-frequency island are also used to locally evaluate the process quality in terms of its relative speed with respect to the other processing nodes (denoted intrinsic speed). The intrinsic speed value defines the upper clock frequency bound of the operating voltage-frequency island. There are also two actuators: the DC-DC converter which provides the supply voltage and the programmable asynchronous ring which generates the desired clock frequency to each local processing node. Finally, the digital controller manages the voltage/frequency couple. It processes the error between the unit speed and the speed setpoint information extracted from the information sent by the operating system (in terms of number of instructions and deadlines) within a closed-loop system, and by applying a well-suited compensator sends the desired voltage and frequency code values to the actuators. Consequently, the system is able to locally adapt their output voltage and clock frequency values clock domain by clock domain, in order to limit their dynamic power consumption. Furthermore, the Asynchronous NoC (ANoC) is the reliable communication path between the different domains. In this model, we propose that data communications between two different processing nodes can fix the speed to the slowest communicating node. This is done by a local synchronization mechanism which is based on handshakes. Thus, the slowest communicating node of the ANoC fixes the throughput in order to have a secure communication without metastability problem and an adaptation to process variability too [18].

II. Robust control design to process variability: Application to the MIPS R2000

MIPS R2000 is a 32-bit Reduced Instruction Set Computer (RISC) initially developed by the Stanford University. It stands for a Microprocessor without Interlocked Pipeline Stages. This means that it has single execution cycle instructions so that the compiler can schedule them to avoid conflicts. The MIPS architecture includes thirty-two general-purpose 32-bit registers and fifty-eight instructions, each 32 bits long. The instructions are processed in a five-stage pipeline: fetch, decode, execute, memory, and write back. MIPS R2000 also includes a coprocessor to handle exceptions and hold configuration bits. Only few configuration bits are used in the R2000 architecture, and are mostly used to enable/disable exceptions and to configure the caches. The MIPS architecture supports exception handling and interrupts. Due to the MIPS R2000 simplicity in terms of architecture, programming model and instruction set, as well as availability as an open core, it has been used as our main case of study in the DVFS control system design for a GALS-NoC, that compensate for the CMOS uncertainties in nanometric technologies (STMicroelectronics $45\,nm$ technology in the present case). The MIPS R2000 is used as a processing node in the energy-performance management architecture previously introduced in Fig. 5. Finally, the integration of the MIPS R2000 is depicted in Fig. 6. The actuators (programmable oscillator and Vdd-hopping), the speed sensor and the digital controller required for a robust control to process variability are then detailed in the following subsections. On the other hand, the ROM memory is loaded with a factorial program to test its execution by the processor. Details of the MIPS R2000 synthesis and analysis results will be shown in section III.

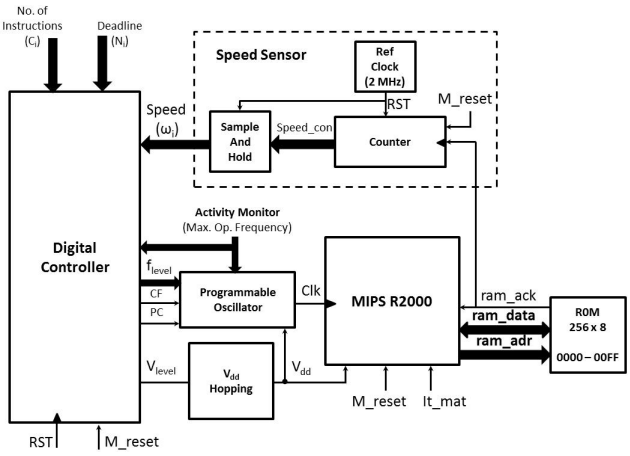


Fig. 6. Architecture of the controlled system in the MIPS environment. The digital controller calculates the voltage and frequency levels to send to the Vdd-hopping and the programmable oscillator in order to manage the energy-performance tradeoff of the MIPS R2000. A speed sensor is also used to provide the real-time computational speed.

A. Vdd-hopping to scale voltage

The DC-DC converter is a circuit that converts a voltage source (of direct current) from one voltage level to another. Two kinds of DC-DC converter can be used. The first class is a continuous DC-DC converter which provides an accurate supply voltage, but with a weak efficiency. The second kind of converter is a digitally controlled step-converter (Vdd-hopping converter) which has a better efficiency but discrete output values. Such a mechanism was described in [19], where two voltages V_{low} or V_{high} could supply the chip. In that case,

the system simply goes to low or high voltage when the input signal becomes $V_{level} = V_{level\_low}$ or $V_{level} = V_{level\_high}$ respectively, with a given transition time and dynamics that depend upon an internal control law (one could refer to the reference above for more details). The Vdd-hopping technique is also modeled with two possible voltage levels in the present paper. Eventually, considering that this inner-loop is extremely fast with respect to the loop considered in this paper, one can neglect the dynamics of the Vdd-hopping.

B. Programmable self-timed ring to scale frequency and manage variations

The application of the proposed DVFS to a system requires the use of a process variability robust source for generating adjustable clocks. For example, these clocks can be derived from analog Voltage Controlled Oscillators (VCO), which are a part of a Phase Locked Loop (PLL). However, VCOs have a limited operating range and require a stabilization time when changing the frequency [20]. Another solution is to use a standard clock divider, but this will make the time resolution coarser due to counting integer periods of the input frequency [21]. In addition, they give regular time step which implies irregular frequency step (usually frequency step follows "$1/x$" curve). Today many studies are oriented to Self-Timed Ring (STR) oscillators which present well-suited characteristics for managing process variability and offering an appropriate structure to limit the phase noise. Therefore, they are considered as a promising solution for generating clocks even in presence of process variability. In [22], [23], they are efficiently used to generate high-resolution timing signals. Their robustness against process variability in comparison to inverter rings is proven in [24]. Moreover, self-timed rings can easily be configured to change their frequency by just controlling their initialization at reset time. At the opposite, inverter rings are not programmable.

Whereas the background, definitions and principles of the Programmable Self-Timed Ring (PSTR) are detailed next, a simple model based on the behavior of the fully programmable/stoppable oscillator of [25] is firstly given. Such a frequency controller has a linear variation of its output frequency with respect to the digital controller feed value. This digital controller value defines the operating frequency. The frequency controller also changes its output frequency with respect to the input voltage, as shown in Fig. 7. The resulting model is $f_{clk} = \gamma f V_{dd}$, where γ is a constant while the desired frequency f depends on the input signal, i.e. $f = \psi(f_{level})$, using such a look-up table mechanism for instance. This asynchronous ring was modeled in *Matlab/Simulink* and tested under the two specified voltage levels of the DC-DC converter for $45\,nm$ CMOS technology. In fact, only some limited frequency values are possible, that is $f_{level} = f_{level\_n}$ with $n \in \{1, 2, \ldots, N\}$ and $f_n > f_{n+1}$, and switching from one frequency to another can be considered as instantaneous.

1) Self-timed rings: The C-element is the basic element in asynchronous circuit design, introduced by D. E. Muller [26]. The C-elements set their output to the input values if their inputs are equal and hold their output otherwise. In Fig. 8(a), a possible CMOS implementation is showed where the initialization circuit is omitted. Each stage of a STR is composed of a C-element and an inverter connected to one of its inputs. The input which is connected to the previous stage is marked F (Forward) and the input which is connected to the following stage is marked R (Reverse), C denotes the output of the stage, as shown in Fig. 8(b).

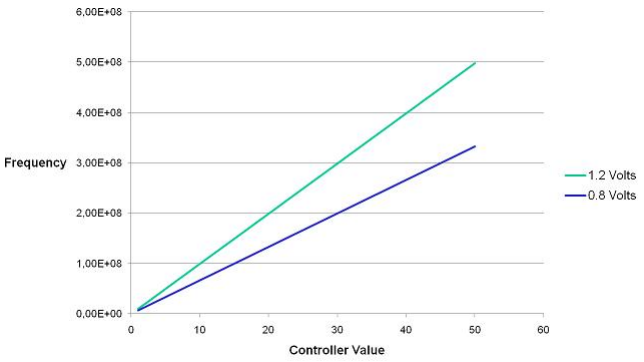
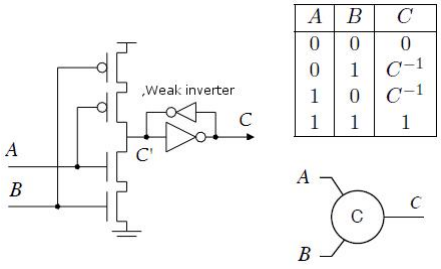
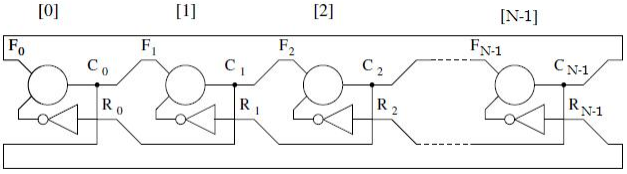


Fig. 7. Behavioral variation of the frequency controller with respect to the input voltage values.



A	B	C
0	0	0
0	1	C^{-1}
1	0	C^{-1}
1	1	1

(a) Muller C-element.



(b) Self-timed ring.

Fig. 8. Representation of a possible CMOS implementation of a self-timed ring, whose each stage is composed of a C-element and an inverter.

The behavior of the STR is mainly based on the tokens "T" and bubbles "B" propagation rule. $Stage_i$ contains a token if its output C_i is not equal to the output C_{i+1} of $Stage_{i+1}$. On the other hand, $Stage_i$ contains a bubble if its output C_i is equal to the output C_{i+1} of $Stage_{i+1}$. The number of tokens and bubbles will be respectively denoted N_T and N_B, with $N_T + N_B = N$, where N is the number of the ring stages. For keeping the ring oscillating, N_T must be an even number. The reader can think about this as the duality of designing the inverter ring by odd number of stages. Each stage of the STR contains either a token or a bubble. If a token is present in a $Stage_i$, it will propagate to $Stage_{i+1}$ if and only if $Stage_{i+1}$ contains a bubble. The bubble of $Stage_{i+1}$ will move backward to $Stage_i$. This implies a transition

on $Stage_{i+1}$ output. For instance, hereafter is depicted the token/bubble movements in a five stage STR which contains four tokens and one bubble.

Example: TTBTT (01001) ⇒ TBTTT (01101) ⇒ BTTTT (00101) ⇒ TTTTB (10101) ⇒ TTTBT (10100) ⇒ TTBTT (01001)

2) Programmable self-timed rings: The oscillation frequency in STRs depends on the initialization (number of tokens and bubbles and hence the corresponding number of stages). The oscillation frequency in a self-timed ring can be approximated according to the number of tokens and bubbles by the formula from [27], which is

$$F_{osc\_STR} = \frac{1}{2D(R+1)}$$

$$(D,R) = \begin{cases} (D_{rr}, {}^{N_T}/_{N_B}) & \text{if} \quad {}^{D_{ff}}/_{D_{rr}} \geq {}^{N_T}/_{N_B} \\ (D_{ff}, {}^{N_B}/_{N_T}) & \text{if} \quad {}^{D_{ff}}/_{D_{rr}} < {}^{N_T}/_{N_B} \end{cases}$$

where D_{ff} is the static forward propagation delay from input F to output C and D_{rr} is the static reverse propagation delay from input R to output C.

Programmability can be simply introduced to STRs by controlling the tokens/bubbles ratio and the number of STR stages. Programmable Self-Timed Ring (PSTR) was presented for the first time in [25]. It uses STR stages based on Muller gates which have a set/reset control to dynamically insert tokens and bubbles into each STR stage. Moreover, to be able to change the number of stages, a multiplexer is placed after each stage. The idea presented in [25] was also extended to have a fully Programmable/Stoppable Oscillator (PSO) based on the PSTR. Look-up tables loaded with the initialization token control word (i.e. to control ${}^{N_T}/_{N_B}$), and the stage control word (i.e. to control N) was used to program the PSTR with a chosen set of frequencies.

3) PSTR programmability applied to MIPS R2000: Presently, the variability is captured in the design by using simulation corners, which correspond to the values of certain process parameters that deviate by a certain ratio from their typical value. In the STMicroelectronics $45\,nm$ CMOS technology, three PVT (Process, Voltage, and Temperature) corners are available, denoted *Best*, *Nominal* and *Worst*. All standard logic cells were characterized at each of these three corners. So, we use Synopsis Design Vision tool to implement a GALS-NoC island based on a MIPS-R2000 using STMicroelectronics $45\,nm$ CMOS libraries in order to test its behavior at each of the previously specified PVT corners. Since, our main goal is to define the optimal operating clock frequency needed by the processing load that compensates for the propagation delay variations due to the process variability impact. Therefore, the critical path delay of the synthesized MIPS R2000 with respect to the supply voltage is analyzed at the three different PVT corners. According to the STMicroelectronics $45\,nm$ libraries, we choose $V_{low} = 0.95\,volts$ and $V_{high} = 1.1\,volts$. Consequently, the optimal clock frequency needed by the MIPS R2000 at the specified two voltage levels with the three different process variability corners are defined as shown in Table I.

In order to adapt the generated clock frequency with respect to the current located process variability impact and to the processed workload, activity sensors are required. As already explained in section I, they play a critical role in DVFS systems and must be carefully selected. The period of the reference clock is crucial since it determines the accuracy of the calculated average speed. Moreover, it determines the system speed response. In fact, the speed sensor integrates also a register to memorize the computational speed on a predefined period and determines the average speed on each rising-edge of the reference clock (i.e. RST signal in Fig. 6). If this period is short, the system will be fast but the calculated average speed will not be accurate. On the opposite, a long period leads to a slow system but to a more accurate speed. Therefore, according to the set of clock frequencies available for the MIPS R2000, the reference clock frequency was chosen to be 2 MHz, in order to count a considerable amount of instructions with a proper system response. To conclude, the computational speed is now applied in terms of number of instructions executed per $500\,ns$ to the digital controller.

TABLE I
MIPS R2000 OPTIMAL CLOCK FREQUENCIES REQUIRED TO COMPENSATE FOR THE PROCESS VARIABILITY IMPACT ON THE $45\,nm$ CMOS TECHNOLOGY.

Voltage level (V)	**Clock frequency** (MHz) for different process variability conditions		
	Worst	*Nominal*	*Best*
0.95	60	75	85
1.1	95	115	145

Whereas three defined process variability corners are defined in the $45\,nm$ CMOS libraries provided by STMicroelectronics, the contents of the PSO code memory presented in [25] are split into three main pages. Based upon the activity monitor output, the corresponding page will be selected. Each page contains a set of programming codes that generates the suitable clock frequencies for the MIPS R2000 which compensate for the delay variation due to the process variability effects. In each programming code set, we have one code corresponding to the clock frequency F_{high} at the high voltage V_{high}, and two other codes corresponding to the clock frequencies F_{low_1} and F_{low_2} at the low voltage V_{low} (a discussion on the number of frequency levels to use follows in subsection II-C). Consequently, our programmable oscillator can be simply represented by the block diagram shown in Fig. 9. Note that FC is the new frequency code which is sent by the digital controller with the activity monitor evaluation of the process variability to the PSO. CF is the change frequency pulse that indicates the presence of a new frequency code and PC is the pause clock pulse to stop the oscillator, refer to [25]. PCW is the PSTR programming code word that specifies the ring initialization and its corresponding number of stages.

C. Control of the energy-performance tradeoff with strong process variability robustness

The digital controller introduced in Fig. 5 is upstream from the Vdd-hopping and the asynchronous ring since it calculates the voltage and frequency level values that have to be sent to these actuators in order to minimize the energy consumption while guaranteeing good computational performance. The

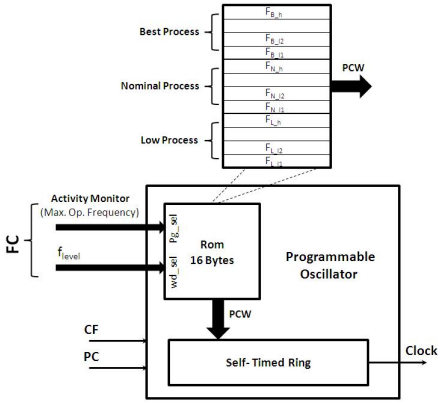


Fig. 9. Memory mapping of the programmable oscillator. It selects the PSTR programming code to apply in function of a given frequency level calculated by the digital controller.

resulting system architecture from a control point of view is given in Fig. 10, where the *device* denotes the electronic system to control (a processor or a computational node for example, or the MIPS R2000 in the present case). A dynamic scaling of the supply power – and therefore the energy consumption – is possible introducing a closed-loop controller which monitors the activity of the device (its computational speed ω in number of instructions per second) in order to adapt the control variables with respect to a given computational load ref to treat. Finally, the model of the whole controlled system (the device and the two actuators) can be approximated by an affine function [28], that is

$$\omega = \sigma f V_{dd} \tag{2}$$

where σ is an unknown parameter which, in fact, can be identified but highly varies with temperature and location on the chip (variability). Nevertheless, the dynamics introduced by the control law will make possible to control the system without any information on this parameter.

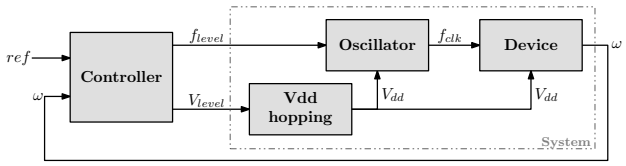


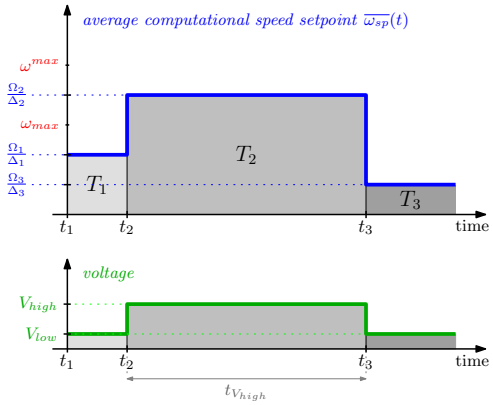
Fig. 10. Representation of the system architecture to control the energy-performance tradeoff. The controller calculates the control variables (voltage level V_{level} and frequency level f_{level}) from the error between the measured computational speed ω and a given setpoint ref, and sends them to both actuators (the Vdd-hopping and the ring oscillator respectively).

Actually, the control of the energy-performance tradeoff in a voltage scalable device consists in minimizing the energy consumption (reducing as much as possible the supply voltage) while ensuring some good computational performance (fitting the tasks with their deadline). This is why we propose to dynamically calculate an energy-efficient computational speed setpoint that the system will then have to track. This setpoint is based on some information provided by a higher level (the operating system for example) for each task T_i to treat, that are the computational load – i.e. the number of instructions Ω_i to execute – and the deadline Δ_i. Moreover, let Λ_i denote the laxity, that is the remaining available time to complete a given task. Note that these parameters can change during the running time of a task (if the OS decides to update them for instance), this is why they are time-dependant.

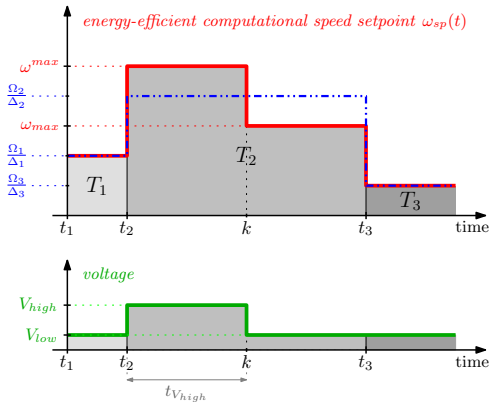
The presence of deadline and time horizon to compute tasks naturally leads to predictive control. Predictive control consists in finding an open-loop control profile over some time horizons and in applying it until the next time instant. The control problem is then reconsidered using the new state variables and a new control profile is generated. This finally yields a closed-loop control and the stability relies in the way the open-loop control is chosen. The horizon can be constant, infinite or less classically contractive as in the present paper. The key point is the choice of the open-loop strategy and its computational cost. Indeed, if predictive control is known to be a robust approach, it is also often associated to high computational cost which is not acceptable in the present case. Whereas the classical strategy consists in minimizing some cost functions, the strategy adopted here is called *fast predictive control* and consists in taking advantage of the structure of the dynamical system to fasten the finding of the open-loop control [29]. The simplicity of system (2) considered here is very suitable for such strategies. The main idea of the predictive strategy is firstly intuitively explained and its formal expression is given in the sequel.

1) Speed setpoint building: Let ω^{max} denote the maximal computational speed when the system is running at high voltage, that is $\omega^{max} = \sigma F_{V_{high}max} V_{high}$ from (2), where $F_{V_{high}max}$ is the maximal frequency in the available range at V_{high}. Respectively, let ω_{max} denote the maximal possible speed at low voltage, that is $\omega_{max} = \sigma F_{V_{low}max} V_{low}$, where $F_{V_{low}max}$ is the maximal frequency at V_{low}. It follows that the high voltage level is necessary as soon as the average speed setpoint of a task is higher than ω_{max} in order to not miss a given deadline. An intuitive method consists in building the average speed setpoint of each task – that is the ratio Ω_i/Δ_i – in such a way that the number of instructions to do is performed at the end of the task. This is depicted in Fig. 11(a). However, this method is not energy-efficient since a whole task can be computed with the penalizing high supply voltage, such as highlighted by $t_{V_{high}}$ for task T_2. Moreover, this technique implies to have an infinite number of voltage levels. Nevertheless, a suggested solution consists in splitting the tasks into two parts. This is represented in Fig. 11(b). Firstly, the chip begins to run at high voltage – if required – with the maximal available frequency in order to achieve the maximal possible speed ω^{max} to go faster than the average speed, such as for T_2 from time t_2 to k. Then, the task could be finished at low voltage – which, consequently, highly reduces the energy consumption – with a speed lower than ω_{max}. A key point in this strategy is that the switching time to go from V_{high} to V_{low} has to be suitably calculated in order to ensure some good computational performance. However, k is not *a priori* known and, therefore, a predictive control law has to be used to dynamically calculate this switching time.

In fact, the lower is the supply voltage the better will be reduced the energy consumption since the supply voltage is the



(a) Intuitive average speed setpoint.



(b) Energy-efficient speed setpoint.

Fig. 11. Different computational speed setpoint buildings can be used to save energy consumption while ensuring some good performance.

penalizing parameter in DVFS. For this reason, we propose to have only one possible frequency F_{high} (i.e. $F_{V_{high}max}$) when running at V_{high} and so minimize the penalizing high voltage running time. On the other hand, several frequency levels F_{low_n} are possible at the low voltage level V_{low} because, as the energy consumption could not be reduced anymore – since no lower voltage level exists – the degree of freedom on the frequency will allow to fit the task with its deadline (as much as this is possible). In the following, we propose to use two frequency levels at V_{low} and we decide that the maximal one is $F_{low_1} = F_{V_{low}max}$. Whereas the maximal levels F_{high} and F_{low_1} are determined from the optimal frequency values in Table I (regarding the variability of the chip), one could note that the second frequency level at low voltage, i.e. F_{low_2}, is equal to $F_{low_1}/2$, which enables to have 3 dB reduction in the power consumption. Note that F_{low_2} is included to add more power saving opportunities to our DVFS once possible.

2) Fast predictive control: Actually, the predictive issue can be formulated as an optimization problem. For each task T_i to treat, what is the computational speed setpoint which minimizes the high voltage running time $t_{V_{high}}$ while guaranteeing that the executed instruction number is equal to the number of instructions to do, that is

$$\min \ t_{V_{high}} \qquad \text{s.t.} \qquad \int_{\Delta_i(t)} \omega(t)\ dt = \Omega_i(t)$$

where $\int \omega dt$ corresponds to the executed number of instructions for the current task. This optimal criteria allows to solve the predictive problem but is too complex to be implemented in an integrated controller with low resources, as in the present case. Nevertheless, the closed-loop solution yields an easier and faster algorithm since, in fact, one simply needs to know ***i)*** *the computational load to treat* and ***ii)*** *how much time is available to do it*. The remaining time before the end of the task is hence necessary, this is why the laxity Λ_i will be used next instead of the deadline Δ_i. The speed required to fit the task with its deadline regarding what it has already been executed – afterwards denoted the *predicted speed* δ – is dynamically calculated at each sampling instant as follows

$$\delta(t_{k+1}) = \frac{\Omega_i(t_k) - \sum_{\tau_i}^{t_k-\tau_i} \omega(t_k)}{\Lambda_i(t_k)} \tag{3}$$

where τ_i is the beginning of the task T_i, t_k and t_{k+1} are the current and next sampling time respectively. The implementation of the previous equation then becomes

$$\begin{aligned} \Omega(t_k) &= \Omega(t_{k-1}) + T_s\omega(t_k) \\ \delta(t_{k+1}) &= \frac{\Omega_i(t_k) - \Omega(t_k)}{\Lambda_i(t_k)} \end{aligned} \tag{4}$$

where Ω is the integration of the computational speed ω, T_s is the sampling period and t_{k-1} is the last sampling time. Furthermore, a conditional instruction is added to be coherent with (3). Indeed, as the real-time speed is integrated on the running time of each task, the variable Ω has to be reset when a task is executed, which means in the last sampling time before its deadline. More precisely, it is not set to zero to prevent the case when the task is not completely executed at its deadline but it is adjusted with the difference between what it has already been done and what it was required to do, such that

$$\Omega(t_k) = \Omega(t_k) - \Omega_i(t_k) \quad \text{if} \quad \Lambda_i(t_k) \leq T_s$$

The energy-efficient speed setpoint ω_{sp} is then directly deduced from the value of the predicted speed, and so are the voltage and frequency levels. Indeed, the system has to run at V_{high} and F_{high} when the required setpoint is ω^{max}, else the low voltage will be enough to finish the task. In fact the computational speed setpoint is not really required since the control variables are easily deduced, but we notice it anyway (for a well understanding) in the control decisions, that are

$$\left| \begin{array}{l} \omega_{sp}(t_{k+1}) = \omega^{max} \\ V_{level}(t_{k+1}) = V_{high} \\ f_{level}(t_{k+1}) = f_{high} \end{array} \right. \quad \text{if} \quad \delta(t_{k+1}) > \omega_{max}$$

$$\left| \begin{array}{l} \omega_{sp}(t_{k+1}) = \omega_{max} \\ V_{level}(t_{k+1}) = V_{low} \\ f_{level}(t_{k+1}) = f_{low_n} \end{array} \right. \quad \text{otherwise}$$

where a frequency control strategy is needed to determine the frequency level f_{low_n}. However, the principle is not detailed here since it remains the same than for the voltage decision. Also, one could note that the division in (4) could be removed for a practical implementation, simply replacing the condition $\delta(t_{k+1}) > \omega_{max}$ in the previous control decision by $\Omega_i(t_k) -$

$\Omega(t_k) > \omega_{max} \cdot \Lambda_i(t_k)$. At the end, the algorithm of the digital controller is hence quite easy to implement in a circuit with limited resources.

The performance are guaranteed because the execution of a task always starts with the penalizing high voltage level – by construction of the predictive control law – and the low one will not be applied while the remaining computational load is important (higher than the maximal possible speed at V_{low}). As a result, it is not possible to make better. Furthermore, even if the voltage/frequency levels discretely vary, the speed setpoint to track is always higher or equal than required by construction. On the other hand, the Lyapunov stability is based on an elementary physical constatation: if the total energy of the system tends to continuously decline, then this system is stable since it is going to an equilibrium state. Let $V = x^T P x$ be a candidate Lyapunov function, with P positive definite and $x(t_k) = \Omega_i(t_k) - \Omega(t_k)$. This latter expression comes from (4), where x refers to the remaining load in the contractive time horizon of the task. Therefore, the Lyapunov function intuitively decreases – because the speed of the processor can only be positive – and so is ensured the stability of a task.

3) Estimation of the maximal speeds: The maximal speeds ω^{max} and ω_{max} might be obtained from the equation of the system model (2). They are proportional to the voltage and the maximal frequency for each voltage level. However, even if the proportional gain is inherent to the device it could vary with temperature or location (variability), and yet, the control law has to be robust to such an uncertainty. Furthermore, the value of the different parameters are not known. For these reasons, we propose to estimate the maximal speeds. Let $\tilde{\omega}_m$ denote the estimated speeds where m denotes the different power modes. A solution consists in measuring the system speed for each couple voltage/frequency levels. Moreover, we propose to use a weighted average of the measured speed in order to filter the (possible) fluctuations of the measurement, which yields

$$\text{if} \quad \begin{cases} V_{level}(t_{k-1}) = V_m \\ f_{level}(t_{k-1}) = f_m \end{cases}$$
$$\tilde{\omega}_m(t_k) = (1-\rho)\tilde{\omega}_m(t_{k-1}) + \rho\omega(t_k)$$

where $0 \leq \rho \leq 1$ is the weighted value. Furthermore, the proposed estimation of the computational speeds – which leads to a control law without any information on the system parameters – yields a robust strategy which will self-adapt whenever the performance of the controlled chip. This is very important for process variability (see [28] for further details on how to bound ρ to satisfy such a robustness).

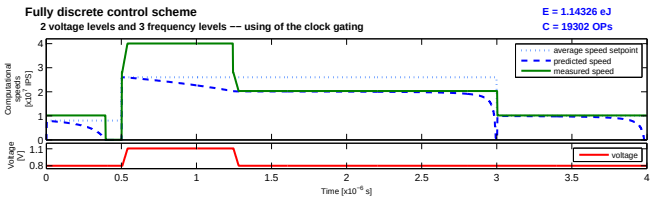
4) Clock-gating principle: On top of the proposed strategy, a last control decision is also possible "deactivating" the clock of the device. This is called the clock-gating principle, as explained in introduction. In this case, the processor runs with the low voltage and a null frequency. This behavior is useful when a task is performed before its deadline to pause the clock until the beginning of the following one. However, in order to minimize the using of the clock-gating principle, we decide to make available this principle only if the beginning of the following task is not too close, that is when $\Lambda_i(t_k) > \Lambda_{min}$, where Λ_{min} is a tunable parameter.

5) Coarse-grain simulation results: Some initial simulation results are realized in *Matlab/Simulink* in order to evaluate the efficiency of the proposed digital controller. A scenario with three tasks to execute is proposed: the first task starts with 4 instructions to do in $0.5\,\mu s$, then a 65 instruction task has to be executed in $2.5\,\mu s$ and the last one has to compute 10 instructions in $1\,\mu s$. These data are considered as provided by the OS. Fig. 12(a) shows the simulation results of the system with two voltage levels and three frequency levels. The top plot shows the average speed setpoint of each task (for guideline), the predicted speed (for guideline) and the measured computational speed, while the bottom plot shows the supply voltage. The energy consumption is calculated in order to have an idea of the achieved reduction. The control strategy is compared with a system without DVFS mechanism – where the supply voltage is fixed to the most penalizing level, i.e. $V_{dd} = V_{high}$, and so is the clock frequency $f_{clk} = F_{high}$ – and without DVS mechanism where only the frequency can be scaled. Fig. 12(a) shows that with the two possible voltage levels the system runs during about $80\,\%$ of the simulation time at low voltage. As a result, a reduction of about $30\,\%$ and $65\,\%$ of the energy consumption is achieved (in comparison with a system without DVS and DVFS mechanism respectively) with the three-task test bench proposed. One could refer to [28] for further details.

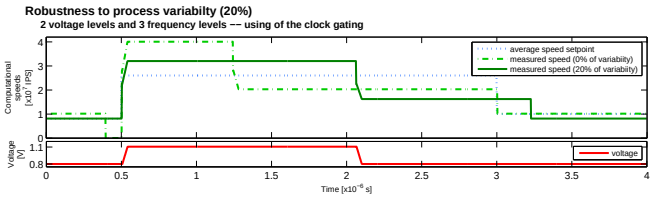
As the proposed control strategy does not use any information on the system parameters, the controller adapts itself with these uncertainties. Fig. 12(b) shows how the system is still working for $20\,\%$ of process variability effects, that is when the real performance of the circuit are $20\,\%$ less than expected. One could see that the estimation of the maximal computational speed allows the system to still work, even if the processing node does not work as expected. Of course, in order to compensate a lower computational speed induced by the process variability, the system will run a longer time at the penalizing supply voltage. Note that the robustness is limited by the maximal possible activity of the processing node anyway. Indeed, if the chip is not enough fast to compute the task while running at the maximal speed (the chip runs with the highest voltage and highest frequency), the controller would not be able to do anything to solve this failure. The only way is to migrate the task allocated to this processing node to a higher performance one, and this has to be done by the operating system. More fine-grain simulation results are shown in section III for the whole MIPS-R2000 architecture with the $45\,nm$ CMOS parameter variations.

III. Simulation results

The design presented in section II is implemented using STMicroelectronics $45\,nm$ CMOS standard libraries for the physical implementation [30]. The digital controller main principle of operation is based on implementing the energy-efficient control algorithm, described in subsection II-C. In order to evaluate the efficiency of the proposed control strategy under different process variability conditions, a post layout simulation with *Modelsim* has been performed. A scenario with three tasks is proposed: the first task starts with 100 instructions to do in $2\,\mu s$, then a 340 instruction task has



(a) Simulation results with three frequency levels, one for the high voltage level and two for the low one, and the clock-gating principle.



(b) Simulation results to test the robustness of the controller with 20% of process variability.

Fig. 12. Simulation results of the energy-performance tradeoff control in *Matlab/Simulink*.

to be executed in $4\,\mu s$, and the last one has to compute 160 instructions in $3\,\mu s$. These data are supposed to be provided by the OS. The simulation results of the system under different process variabilities are shown in Fig. 13. The different available frequencies for each case were summarized in Table I.

a) Nominal process variability results are shown in Fig. 13(a). The used set of clock frequencies are $F_{high} = 115\,\text{MHz}$, $F_{low_1} = 75\,\text{MHz}$ and $F_{low_2} = 37\,\text{MHz}$. Task 1 was completed successfully with both V_{level} and F_{level} equal to the low level, all over the allocated time for the task. For task 2, the digital controller speeds up the MIPS R2000 in order to be able to complete the task at the proposed deadline, by selecting the maximal supply voltage V_{high} and clock frequency F_{high}. Once the digital controller has detected that task 2 can be completed with relaxed conditions, the system switches back to V_{low}. Therefore after running the MIPS R2000 for $1.52\,\mu s$ at 1.1V, it is now supplied with 0.95V. For task 3, the digital controller keeps supplying the MIPS R2000 with V_{low} and F_{low_2} all over task 3 execution time till its deadline.

b) In Fig. 13(b), the simulation results of the system under ***worst process variability*** are depicted. In this case, the used set of clock frequencies are now $F_{high} = 95\,\text{MHz}$, $F_{low_1} = 60\,\text{MHz}$ and $F_{low_2} = 30\,\text{MHz}$. Therefore, the programmable oscillator now generates the proper set of clock frequencies that has to be used with the reduced performance of the MIPS R2000 (i.e. increased critical path delay), under worst process variability effect. As a result, task 2 runs for $3.02\,\mu s$ at V_{high}, which is twice longer than processing under nominal process viability. However, it then switches to V_{low} where it is able to complete the task successfully.

c) The same three tasks workload is simulated again for the MIPS R2000 under ***best process variability***. The simulation results are shown in Fig. 13(c). Now, the used set of clock frequencies generated by the programmable oscillator are $F_{high} = 145\,\text{MHz}$, $F_{low_1} = 85\,\text{MHz}$ and $F_{low_2} = 43\,\text{MHz}$. These clock frequencies correspond to the proper set that can be used under best process variability. When using these frequency configurations, the MIPS R2000 is able to successfully complete all the three tasks at V_{low}, which adds much more power/energy saving opportunities than under the nominal case. Therefore, our proposed DVFS control architecture is able to not only compensate for the delay variations with different process variability impacts, but also exploit the enhanced response of the system under best variability conditions to gain more in terms of energy savings.

To evaluate the proposed energy-efficient DVFS control for GALS-NoC architecture, the implemented chip is characterized for its average dynamic power, energy consumption, area overhead and robustness to process variability. Table II shows a comparison between the proposed energy-efficient DVFS control using dynamic set of clock frequencies with an intuitive average based DVFS control using fixed set of clock frequencies. The values are given with respect to a system without DVFS. From these results, it is clear that the DVFS control with the energy-efficient control under nominal process variability is 1.5 more power and energy saving efficient than the average based control. Energy-efficient DVFS control is able to save $21.18\,\%$ of the energy consumption and $51.42\,\%$ of the average dynamic power consumed by a system without DVFS. Our proposed energy-efficient DVFS control has the ability to adapt the set of clock frequencies generated by the PSO with respect to the process variability impact. Therefore the energy-efficient DVFS control was able to exploit this enhanced performance of the system (i.e. reduced critical path delay) to save more energy consumption (i.e. $25.41\,\%$ under best process variability impact). However, the average based DVFS control saves the same amount of energy regardless of the reduced process variability impact. Under worst process variability conditions, the used set of clock frequencies for a system without DVFS (i.e. 115 MHz) and even that for a system with average based DVFS control (i.e. 115 MHz at V_{high} and 75 MHz at V_{low}) violates the MIPS R2000 critical path delay. As a result, the MIPS R2000 will have erroneous output results. Therefore this GALS-NoC processing node has to be neglected and its allocated tasks have to be distributed over other high performance processing nodes. However, with the proposed DVFS control architecture, the MIPS R2000 was still able to complete the allocated tasks successfully by using the proper set of maximal clock frequencies (i.e. 95 MHz at V_{high} and 60 MHz at V_{low}). This drastically relaxes the fabrication constraints and helps the yield enhancement.

The whole DVFS control system including the effect of different parts depicted in Fig. 5 is also evaluated, as shown in Table III. The values are given for a GALS-NoC voltage-frequency island with a single processing element (i.e. MIPS R2000) and compared with another one with eight processing elements. Under nominal process variability, the average dynamic power and energy saving values of the whole DVFS control system are smaller than but not too far from those presented in Table II. On the other hand, we have a single DVFS control system for all the voltage-frequency islands in a GALS-NoC system. Therefore, in a voltage-frequency island with multiprocessing elements, the efficacy of the DVFS control system will be more effective in saving power/energy

TABLE II
A COMPARISON BETWEEN ENERGY-EFFICIENT AND NORMAL AVERAGE BASED DVFS CONTROL WITH RESPECT TO A SYSTEM WITHOUT DVFS AT DIFFERENT PROCESS VARIABILITY CORNERS.

Process variability impact	**Energy-efficient control**	
	Average dynamic power savings	Energy savings
Best	51.39 %	25.41 %
Nominal	51.42 %	21.18 %
Worst	Achieve the requested performance with a reduced set of clock frequencies (Yield enhancement)	
Process variability impact	**Average based control**	
	Average dynamic power savings	Energy savings
Best	36.78 %	14.12 %
Nominal		
Worst	Used set of nominal clock frequencies violates the MIPS R2000 critical path (Erroneous data outputs)	

consumption, see Table III. Moreover, the area overhead of the extra DVFS hardware will be approximately divided by the number of processing elements per a GALS island. For example, the area overhead in a processing island with eight processors is $4.15\,\%$.

TABLE III
PERFORMANCE ANALYSIS OF THE WHOLE ENERGY-EFFICIENT GALS-NOC DVFS CONTROL SYSTEM UNDER NOMINAL PROCESS VARIABILITY IMPACT.

No. of processing elements per GALS-NoC island	**Whole energy-efficient control**		
	Average dynamic power savings	Energy savings	Area overhead
1	45.62 %	14.86 %	33.21 %
8	50.7 %	19.92 %	4.15 %

CONCLUSIONS

In this paper a survey of different problems facing designers over the nanometric era was first presented. Some solutions were suggested in order to reduce the impact of process variability, improve the yield enhancement and decrease the leakage power consumption. A GALS system was taken as an issue with the application of DVFS technique. Also, a closed-loop scheme clearly appears as necessary in such systems and an architecture was hence proposed in this way. The idea to use integrated sensors embedded in each clock domain is presented as one of the promising solutions to reduce the process variability impact. Regarding the actuators, a Vdd-hopping converter and a Programmable Self-Timed Ring (PSTR) oscillator provide discrete supply voltage and clock frequency levels respectively. A control algorithm has also been proposed, based on a fast predictive control law. The proposed feedback controller especially adapts more smartly voltages and frequencies (energy-performance tradeoff) with strong process variability. Finally, a practical validation of the proposed ideas was realized on the MIPS R2000 microprocessor over STMicroelectronics $45\,nm$ technology to get more information about the power consumption and area overhead of each unit in the power management architecture. Global results for a multicore system was also presented. Through this example, we hence demonstrated that a dedicated feedback system associated to the GALS paradigm and some DVFS techniques, with correct sensors and actuators, is able to achieve better robustness against process variability.

ACKNOWLEDGMENTS

This research has been supported by the ARAVIS project, a Minalogic project gathering STMicroelectronics with academic partners, namely TIMA and CEA-LETI for microelectronics and INRIA for operating system and control. The aim of the project is to overcome the barrier of sub-scale technologies ($45\,nm$ and above).

REFERENCES

[1] B.F. Romanescu, M.E. Bauer, D.J. Sorin, and S Ozev. Reducing the impact of process variability with prefetching and criticality-based resource allocation. In *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*, 2007.

[2] B. Pangrle and K. Shekhar. Leakage power at 90nm and below. In *EE Times Asia*, 2005.

[3] A. Nicoli. Achieving yield in the nanometer age. In *Mentor Graphics Corp.*, 2007.

[4] W. Kuzmicz, E. Piwowarska, A. Pfitzner, and D. Kasprowicz. Static power consumption in nano-cmos circuits: Physics and modelling. In *Proceeding of the 14th International Conference Mixed Design of Integrated Circuits and Systems*, 2007.

[5] K. von Arnim, E. Borinski, P. Seegebrecht, H. Fiedler, R. Brederlow, R. Thewes, J. Berthold, and C. Pacha. Efficiency of body biasing in 90nm CMOS for low-power digital circuits. *IEEE Journal of Solid-state Circuits*, 40(7):15491556, 2005.

[6] Y.H. Lu and G. De Micheli. Comparing system-level power management policies. *IEEE Design and Test of Computers*, 18:10–19, 2001.

[7] K. Flautner, D. Flynn, D. Roberts, and D.I. Patel. An energy efficient soc with dynamic voltage scaling. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2004.

[8] A. Varma, B. Ganesh, M. Sen, S.R. Choudhury, L. Srinivasan, and J. Bruce. A control-theoretic approach to dynamic voltage scheduling. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2003.

[9] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proceedings of the International Sympsonium on Low Power Electronics and Design*, 1998.

[10] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001.

[11] K. Flautner, S. K. Reinhardt, and T. N. Mudge. Automatic performance setting for dynamic voltage scaling. In *Mobile Computing and Networking*, 2001.

[12] T.D. Burd and R.W. Brodersen. Processor design for portable systems. *The Journal of VLSI Signal Processing*, 13(2):203–221, 1996.

[13] TD Burd, TA Pering, AJ Stratakos, and RW Brodersen. A dynamic voltage scaled microprocessor system. *IEEE Journal of Solid-State Circuits*, 35(11):1571–1580, 2000.

[14] Y. Zhu and F. Mueller. Feedback dynamic voltage scaling dvs-edf scheduling: Correctness and pid-feedback. In *Workshop on Compilers and Operating Systems for Low Power*, 2003.

[15] L. Fesquet and H. Zakaria. Controlling energy and process variability in system-on-chips: Needs for control theory. In *Proceedings of the 3rd IEEE Multi-conference on Systems and Control - 18th IEEE International Conference on Control Applications*, 2009.

[16] D. Marculescu and E. Talpes. Energy awareness and uncertainty in microarchitecture-level design. *IEEE Micro*, 25:64–76, 2005.

[17] M. Krstic, E. Grass, F.K. Gurkaynak, and P. Vivet. Globally asynchronous, locally synchronous circuits: Overview and outlook. *IEEE Design and Test of Computers*, 24:430–441, 2007.

[18] T. Villiger, H. Käslin, F.K. Gürkaynak, S. Oetiker, and W. Fichtner. Self-timed ring for globally-asynchronous locally-synchronous systems. In *9th International Symposium on Asynchronous Circuits and Systems*, 2003.

[19] Carolina Albea Sánchez, Carlos Canudas de Wit, and Francisco Gordillo. Control and stability analysis for the vdd-hopping mechanism. In *Proceedings of the IEEE Conference on Control and Applications*, 2009.

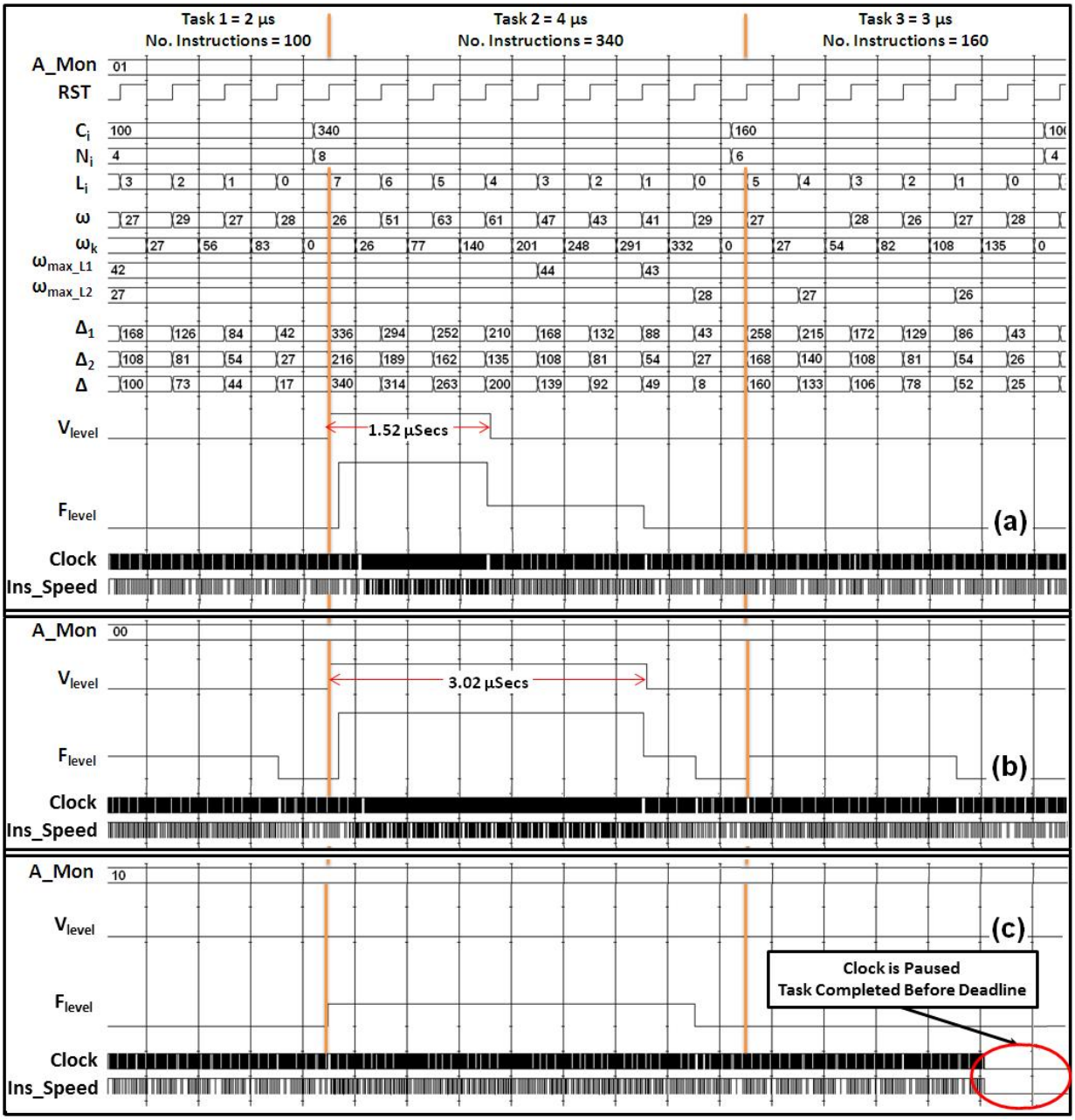


Fig. 13. Timing diagram of the digital controller behavior with 3 different MIPS R2000 workloads under different process variability effects: *Nominal*, *Worst* and *Best*.

[20] F.R. Boyer, H.G. Epassa, and Y. Savaria. Embedded power-aware cycle by cycle variable speed processor. *IEE Proceedings of Computers and Digital Techniques*, 153(4):283 290, 2006.

[21] M. Stork. Digital building block for frequency synthesizer and fractional phase locked loops. In *Joint First Workshop on Mobile Future and Symposium on Trends in Communications*, 2003.

[22] S. Fairbanks and S. Moore. Analog micropipeline rings for high precision timing. In *Proceeding of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 2004.

[23] V. Zebilis and C. P. Sotiriou. Controlling event spacing in self-timed rings. In *11th IEEE International Symposium on Asynchronous Circuits and Systems*, 2005.

[24] J. Hamon, L. Fesquet, B. Miscopein, and M. Renaudin. High-level time-accurate model for the design of self-timed ring oscillators. In *14th IEEE International Symposium on Asynchronous Circuits and Systems*, 2008.

[25] E. Yahya, O. Elissati, H. Zakaria, L. Fesquet, and M. Renaudin. Programmable/stoppable oscillator based on self-timed rings. In *15th IEEE International Symposium on Asynchronous Circuits and Systems*, 2009.

[26] D. Muller and W. Bartky. A theory of asynchronous circuits. In *Proceedings of International Symposium on the Theory of Switching*, 1959.

[27] O. Elissati, E. Yahya, L. Fesquet, and S. Rieubon. Oscillation period and power consumption in configurable self-timed ring oscillators. In *Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference*, pages 1 – 4, 2009.

[28] S. Durand and N. Marchand. Fully discrete control scheme of the energy-performance tradeoff in embedded electronic devices. In *Proceedings of the 18th World Congress of IFAC*, 2011.

[29] M. Alamir. *Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes: A Parametrized Approach for Fast Systems*, volume 339. Springer-Verlag, 2006.

[30] H. Zakaria. *Asynchronous Architecture for Power Efficiency and Yield Enhancement in the Decananometric Technologies: Application to a Multi-Core System-on-Chip*. PhD thesis, University of Grenoble (France), 2010.