Methods for Power Optimization in SOC-based Data Flow Systems

PHILIPPE GROSSE, CEA-LETI
YVES DURAND, CEA-LETI
and
PAUL FEAUTRIER, Université de Lyon

Whereas the computing power of DSP or general-purpose processors was sufficient for 3G baseband telecommunication algorithms, stringent timing constraints of 4G wireless telecommunication systems require computing-intensive data-driven architectures. Managing the complexity of these systems within the energy constraints of a mobile terminal is becoming a major challenge for designers. System-level low-power policies have been widely explored for generic software-based systems, but data-flow architectures used for high data-rate telecommunication systems feature heterogeneous components which require specific configurations for power management. In this study, we propose an innovative power optimization scheme tailored to self-synchronized data-flow systems. Our technique, based on the synchronous data-flow modeling approach, takes advantage of the latest low-power techniques available for digital architectures. We illustrate our optimization method on a complete 4G telecommunication baseband modem and show the energy savings expected by this technique considering present and future silicon technologies.

Categories and Subject Descriptors: B.8.2 [**PERFORMANCE AND RELIABILITY**]: Performance Analysis and Design Aids; C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Signal processing systems; C.4 [**Performance of systems**]: Modeling techniques

General Terms: Algorithms, Design

Additional Key Words and Phrases: Power optimisation, Data-driven SOC, 4G base-band modem, Synchronous Data-flow Graph,

1. INTRODUCTION

In signal processing or wireless telecommunication, Systems-on-Chip (SOCs) with tens of millions of gates are quite common. These chips have stringent constraints in terms of energy consumption, as do their applications in terms of performance. In addition, the current trend for wireless communication is reconfigurability: SOCs have to support multiple and very different operating modes. Because each oper-

Author's address: Philippe Grosse is now with Fraunhofer Institute for Integrated Circuits, Nordostpark 93, D-90411 Nürnberg, GERMANY Email: philippe.grosse@iis.fraunhofer.de
Yves Durand, CEA-LETI MINATEC, 17 rue des Martyrs Grenoble F-38054, FRANCE Email: yves.durand@cea.fr
Paul Feautrier, ENS de Lyon/LIP (INRIA, CNRS, UCBL) 46 Allée d'Italie Lyon F-69364, FRANCE Email: paul.feautrier@ens-lyon.fr

ating mode has its own constraints and execution profile, it becomes very difficult to design a generic power management policy which fits all modes.

In this study, a systematic method for computing the configuration of a compute intensive SOC is developed for a given application scenario and its associated constraints, in order to minimize its energy dissipation. The method takes advantage of the latest possibilities offered by silicon technology, such as voltage and frequency scaling. The applicative scenario is translated into a set of linear constraints for each individual component of the system, which capture not only the throughput constraint of the system, but also the initial latency. The method computes the voltage and frequency parameters which optimize the energy dissipation of the SOC globally, i.e. taking both transient and established mode into account.

This technique can be used in combination with the widely used on-off technique, which is limited to powering the individual components of the system off and on. It offers a significant gain with negligible added complexity on the embedded system itself. The technique is illustrated on a specific baseband processor SOC for high data-rate wireless communication. However, the technique may be generalized to any self-synchronized data-flow system as long as they are instrumented both for on-off power management and voltage and frequency scaling.

2. PROBLEM STATEMENT

2.1 General problem statement

From an abstract point of view, a distributed data-flow system is considered as a set of N functional units U_i, which operate asynchronously and communicate via synchronization FIFOs. The applications considered have periodic behavior. The time is divided into *frames* of constant duration T_f, including a guard period of sufficient length to ensure that all processing is finished before the next frame begins. During a frame and for a given mode, the amount of processing to be done by each unit is known and predefined.

It is assumed that the system is self-synchronized, and that every unit switches off when its processing is complete. The number of iterations of each function is known, and for each unit, the intrinsic delay, i.e. the number of clock cycles necessary to generate the output data is predictible. It is also considered that the depth of synchronization FIFOs is known and guarantees the absence of overflow even for the most demanding mode of the system.

It is also considered that the individual operating frequencies f_i and supply voltages V_i of each of the functional units can be adjusted continuously between boundaries that depend only on the technology[1].

Figure 1 summarizes the parameters relative to each unit. The system supports K different functional modes either in transmission or in receipt. It may switch from one mode to another between time frames.

For each mode m, the applicative constraints, *i.e.* throughput constraints for the established rate phase and latency constraints for the transient phase, may be

[1]Some technologies may provide only a discrete scale of voltages and frequencies, in which case our solution must be rounded up to the nearest scale point at the cost of loss of efficiency

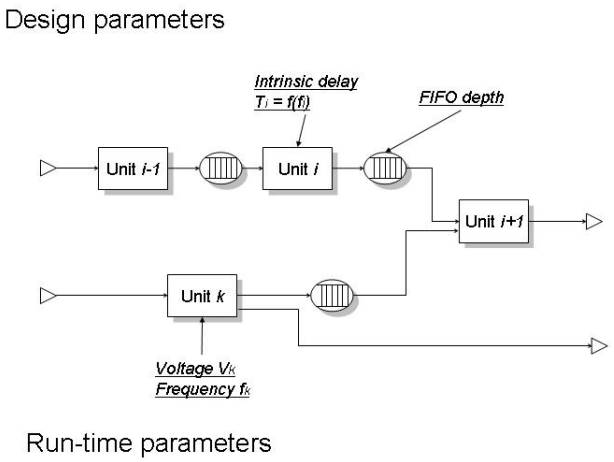

Fig. 1. parameters influencing power dissipation

broken down into a set of inequalities $\mathcal{F}_m(\frac{1}{f_0}, ..., \frac{1}{f_N}) < \mathcal{C}_m$, where $\mathcal{C}_m$ is a constant vector. We show later in this study that these inequalities become linear when expressed in terms of the periods $T_i = 1/f_i$.

For a given mode m, the energy consumption during a time frame E_m may be modeled by the sum of the energies $E_{i,m}$ consumed by the individual units, which depends on the mode, the unit operating frequency f_i and its supply voltage V_i, plus an overhead energy O_m which depends only on the mode. Using the operating clock periods $Tc_i = 1/f_i$, this may be written as:

$$E_m = O_m + \sum_{i=1}^{N} E_{i,m}(Tc_i, V_i) \tag{1}$$

Since the periods Tc_i and the voltages V_i are coupled variables, we may express $E_{i,m}(Tc_i, V_i)$ as a non linear function of T_i only.

The problem consists in finding a set of periods $\{T_{0,m}, ..., T_{N,m}\}$ that minimizes E_m during a time frame in mode m, and satisfies the constraints $\mathcal{F}_m(Tc_0, \ldots, Tc_N) < C_m$.

2.2 Design considerations

The design of an energy-effective system is a trade-off between architectural choices and run-time parameters. For a data-flow system, the key parameter is the computing efficiency of the cores, but it in turn impacts the maximum depth of the synchronization FIFOs, and the size of the embedded memories.

The simplest optimization, called *dynamic power management*, in brief DPM, consists in switching the cores off when they are inactive. In a distributed system, this can be managed at individual unit level without adding extra control. However, this has a cost in energy and latency, therefore other techniques may be be preferred

or combined with DPM, as will be shown later in this study.

For real-time, CPU-based systems, fine-grain task scheduling may also help to save energy, but this does not apply to self-scheduled systems, on which we focus here. Finally, voltage scaling may be applied as a complement to further reduce the energy consumption. Recent techniques such as voltage and frequency hopping [Miermont et al. 2007] offer the possibility to vary both parameters jointly in a quasi-continuous range at the level of each functional unit, with, of course, a performance penalty linked with the frequency decrease.

However, maximum performance is not always mandatory: at design time, the system is dimensioned for the most demanding mode, according to the application constraints. In real usage, the quality of the air channel may enforce different modes: this leads to hundreds of different configurations in which the system does not operate at full rate. In these other modes, excessive dimensioning of the system can be compensated by reducing the operating frequencies of the blocks, while still satisfying the application constraints. The goal of this study is to elaborate a systematic, optimal method for configuring the operating frequencies for all modes.

During a time frame T_f, the system behavior may be be divided into two phases, a transient phase and an established phase, which give two distinct sets of constraints. Section 2.3 illustrates this point on a specific case of the FAUST system, and shows how it translates into a system of linear inequalities. However, in our system, it is very difficult to change individual frequencies *during* the frame time, whilst they are easily set at the beginning of the frame. The main reason is that there is no way for a central configuration agent to identify the end of the transient phase at the level of each unit. Therefore, we focus here on optimizing both phases of the frame globally for a given mode m. Individually optimizing each phase would have been more straightforward but inapplicable to our design, since it would have required a centralized fine-grain control, which is difficult to implement in a high data-rate distributed system.

2.3 The FAUST modem example

FAUST is a complex SOC system, which implements the baseband modem function of a wireless communication terminal, i.e. encoding of the numerical information just before its final modulation by the carrier. It targets different 4G wireless telecommunication protocols such as 4MORE [Kaiser et al. 2004], or a specific version of 3GPP/LTE [Phan Huy et al. 2008]. The first version of the system was released in January 2006 in HCMOS 130 nm technology [Lattard et al. 2008]. A second version, using a 65 nm technology, should be available in the third quarter of 2008.

The baseband modem is an aggregation of 46 functional units, some of them specialized, others more configurable, organized around a high-throughput meshed network. Each of the units is a Voltage/Frequency island which works synchronously under its own clock, which may be adjusted dynamically. The design is a FIFO-based implementation of the globally asynchronous locally synchronous (GALS) approach, which means that the exchange of data between units takes place fully asynchronously through bounded-size FIFOs [Suhaib et al. 2007]. Figure 2 gives a global view of the baseband part of the 4MORE mobile terminal.

This architecture offers several very interesting aspects, in terms of simplification

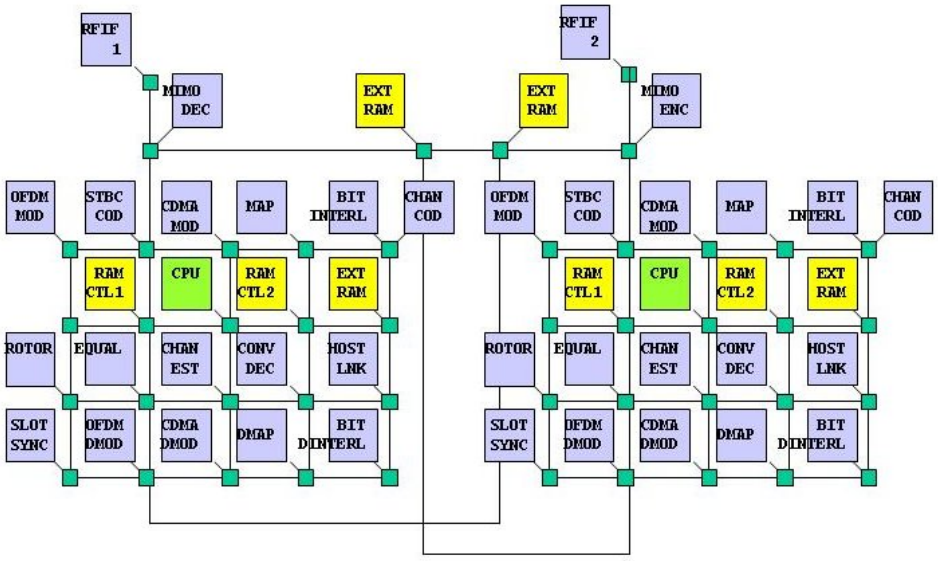

Fig. 2. Global topology of the 4MORE baseband modem

of the control structure, network bandwidth, and reconfigurability. These aspects, which were developed in [Durand et al. 2005], are out of scope here. Nevertheless, the following two characteristics of the architecture are fundamental for power management: the fact that processing is self-synchronized, and the Globally Asynchronous Locally Synchronous (GALS) implementation of the chip. Our power management technique is dependent on these two features, and conversely, may apply to any system which implements them.

The system must satisfy two types of constraints which derive from the application specifications. The *rate constraints* are associated to the required data throughput. In simple terms, this means that the system has to sustain a continuous traffic, either in receipt or in transmission, in any mode. For example, the 4MORE air interface protocol [Kaiser et al. 2004] is organized in a Time Division Duplex fashion: the time is divided into frames of duration $T_F = 666\mu$s, during which the modem may be receiving, transmitting data, or be idle. If T_D is the time to decode the full frame, i.e. the time between decoding of the first data and decoding of the last data, the throughput constraint is $T_D \leq T_F$.

In our case, the *latency constraints* derive from the specification of the timing between decoding of a frame and transmission of another frame. This is very common in wireless transmissions, where the delay between a transmission and its acknowledgment has to be kept minimal. For simplification sake, we arbitrarily break this global constraint down into two constraints, independent from the mode m: one for the receipt process, which specifies the time T_{LRx} between the receipt of the first data sample from the radio and decoding of the first byte of the frame, and one on transmission, which relates to the time T_{LTx} between submission of the first data sample for the frame and transmission of the first radio sample. This

enables independent optimization of receipt and transmission modes.

To sum up, our study focuses practically on determining the optimal frequency and voltage pairs $(V_{i,m}, f_{i,m})$ for each of the functional units of the FAUST system, considering different modes m for different rates of the 4MORE protocol either in receipt or in transmission. Two kind of constraints are considered: throughput constraints, which depend on the mode, and latency constraints, which do not.

3. OPTIMIZATION OF ENERGY CONSUMPTION FOR DISTRIBUTED TRANSMISSION SYSTEMS

3.1 Preliminary estimates

In a preliminary study [Grosse et al. 2006], we ran a power simulation campaign of the FAUST design with an accurate gate power analyzer[2]. We based our simulations on a real transmission/receipt scenario (4More project specifications) considering two silicon implementations:

—a 130nm implementation corresponding to the first release of the chip (synthesized with a standard design flow),

—a 65nm implementation synthesized with a power-aware design flow (use of Multi-VT and automatic gated-clock synthesis).

We used the results of these simulations in the remainder of this section to guide our optimization.

3.2 Optimization of energy consumption for the FAUST system

The energy consumption of a SOC is the sum of the contributions of its network on chip (NOC) E_{NOC}, its shared memory buffers E_{MEM}, its CPU E_{CPU} and the sum of the contributions of the dedicated data processing units $E_{i,m}$.

In the case of a transmission system such as FAUST, the contributions E_{NOC}, E_{MEM}, E_{CPU} do not depend on the mode m. Our simulations have shown that E_{NOC} depends only on the number of asynchronous nodes and on the amount of data transferred, and represents 6% of the total consumption. The CPU accounts for 8% of the energy. However, our specific implementation of the CPU is not instrumented for DVS. Moreover, a DPM policy would significantly increase the very critical interrupt response times of the system. Therefore, we do not consider optimizing the CPU contribution by switching it off and on. We thus concentrate only on optimization of the energy of the functional units, since they represent 80% of the whole circuit consumption.

The energy dissipated by these hardware units during a frame time[3] T_f is divided into a *static* and *dynamic* contribution:

$$E_{i,m} = P_{sta,i} T_f + E_{dyn,i,m} \tag{2}$$

The non-critical paths of the design are implemented with low-leakage transistors using Multi-Vt synthesis. However, our application is data-intensive, even for the less demanding modes. Therefore, static dissipation does not represent a significant

[2]We used a commercially available tool: SYNOPSYS Prime-Power

[3]T_f includes the guard period between successive frames

part of the global dissipation: our simulations have shown that the static power is reduced to less than 2% of the total dissipated power, and this result was later confirmed by actual measurements in 65 nm [Beigné et al. 2008]. Therefore we can consider static power as a constant and negligible, and focus on dynamic power optimization.

Simulations have also shown that the functional units have different but characterized consumptions profiles depending on whether they are *active*, i.e. processing data, or *inactive*, i.e. waiting for data. Therefore, we further separate the dynamic contributions into two parts:

$$E_{i,m} \sim E_{dyn,i,m} = P_{active,i,m}\frac{n_{i,m}}{f_{i,m}} + P_{inactive,i}\frac{(\mathcal{N}_{i,m} - n_{i,m})}{f_{i,m}} \tag{3}$$

where P_{active} (resp. $P_{inactive}$) is the power dissipated during active (resp . inactive) phase, $\mathcal{N}_{i,m}$ is the number of execution cycles for a frame, $n_{i,m}$ the number of active cycles during a frame, $f_{i,m}$ the clock frequency of unit i in mode m and $V_{i,m}$ is the supply voltage.

The afore-mentioned study has shown that DVS is the most effective way to save energy on the most critical blocks of our system, when using the reference 130 nm and 65 nm technologies. Still, it may be envisaged to use a Dynamic Power Management policy for the smaller blocks, i.e. to switch them on or off when inactive. However, our simulations showed that gated clock synthesis reduces the dynamic power dissipated during inactive phases by a factor ten, compared to active phases. Hence, even for these non-critical blocks, the energy saved by switching on and off becomes marginal compared to inactive mode only, and does not justify the overhead of implementing the DPM circuitry.

Therefore, our optimization concentrates on dynamic energy in active phase, which may be expressed simply as:[4]

$$E_{dyn,i,m} = n_{i,m}\frac{1}{2}C_i V_{i,m}^2 \tag{4}$$

where C_i is the equivalent switched capacitance in active mode for unit i.

3.3 Related works

The objective of dynamic voltage scaling (DVS) is to reduce the quadratic contribution of V into E during system activity. Recent works have focused on *compiler-based DVS*, which exploit the specific instructions which modern processors provide for controlling their own hardware [Xie et al. 2003]. In this context, voltage adjustment is actually implemented by software calls, called *scaling points*, actually triggered by scheduler interrupts, either regular (*interval-based*), triggered by some event (*e.g. task-based*) or even fixed at compile time. When active, these scaling points adjust the frequency and voltage of the system, either for specific tasks (*intra-task* scheduling) or for the complete system (*inter-task* scheduling). Some algorithms exploit only dynamic parameters such as the number of cache misses, workload, etc. For example, Wu *et al* describe in [Wu et al. 2005] a method based on adaptative control for regulating a processor, which is modeled as a set of queues.

[4]For the sake of clarity, the reference to m is omitted in the remainder of this study.

The queue occupancy is used as input parameter. However, most techniques use a preliminary *off-line* calculation, such as task worst-case computation execution times (WCET) like in [Pillai and Shin 2001]. Some authors use even more sophisticated analysis: for instance, Saputra *et al* [Saputra et al. 2002] provide a pre-calculated table of the energy consumption with pre-selected voltages to their integer linear programming solver. The resulting pre-calculated tables are used at scaling points to verify that a schedule satisfies the real-time constraint. These techniques are mainly appropriate for processors, although the underlying methods for computing the static parameters are often similar to our formulation. For example, the real-time latency constraints often translate into a system of linear equations.

Recent progress in the DVS circuitry, such as the use of *VDD hopping* in ([Miermont et al. 2007]), have masked the switching overhead (*i.e.* the time to switch from one voltage to another) which was taking some tens of microseconds in recent cores, and increased the number of available voltage levels. Thus, DVS becomes attractive for highly-constrained, data-intensive systems. However, the problem is quite different for these specialized chips, with little or no CPU control, and where latencies are counted in nanoseconds rather than in microseconds.

The self-timed execution model [Ghamarian et al. 2006] is better suited to compute-intensive data-flow system than CPU based control which suffers from the long latencies associated with interruptions. Self-timed circuits are by nature well suited to DVS, because they are already partitioned into uncorrelated islands, which may run at different frequencies. The idea of combining both techniques has been reported earlier in the literature [Nielsen et al. 1994]. The central issue is a trade-off between energy savings and timing constraints, because reducing the supply voltage increases the gate traversal delay [Khouri and Jha 2002] and affects the execution times.

Still, many algorithms have been proposed and developed for this kind of system: here, the classification into *off-line* and *on-line* (or *run-time*) approaches is fundamental. In the context of self-timed circuits, central frequency control is very costly to implement during the normal flow of operations, and cannot be synchronized accurately because of the difference of timescales between a processor and functional units. Thus, a realistic *run-time* voltage adjustment can only rely on criteria that are local to the individual components . Most proposals implement local feedback controls based on the state of the input (or output) FIFOs at unit level to adapt the voltage [Alimonda et al. 2006].

Off-line optimization may be used when the application behavior is predictable, as is the case in most transmission systems. Here, its only drawback is the size of configuration data that have to be embedded in the system.

Niyogi [Niyogi and Marculescu 2005] proposes a power management method, tailored to data-flows systems, which takes the latency and throughput constraints of the tasks into account . From a task graph, they compute a set of voltage/frequency values for the different processing elements, which minimizes the consumption and fulfills the performance constraints. However, their optimization method does not guarantee optimality of the computed settings. Furthermore, their latency expression is simply a sum of worst case execution times which does not take the rate

mismatches during transient mode into account.

4. POWER MANAGEMENT MODEL

4.1 Basic model and notations

The circuit to be studied is modeled as a system of interconnected units. Each unit has one or more input ports, and one or more output ports. Ports are connected by one-to-one channels, which are modeled as perfect FIFO buffers[5]. The behavior of each unit is cyclic. The unit waits until enough tokens are present on each of its input ports, collects these tokens, processes them, and writes the result tokens on its output ports.

The structure of the system may be represented as a directed graph $\Gamma = (U, C)$, with a vertex set U (the units) and an edge set C (the channels). We will assume this graph to be acyclic. Graphs with feedback edges seldom occur in data processing applications. Their consideration would needlessly complexify the simplified framework we intend to use.

The system graph can be represented as an incidence matrix Γ, where element Γ_{ij} represents the number of tokens produced (or consumed, in which case Γ_{ij} is negative) by unit i on FIFO j.

We focus on the case of a system where each FIFO j has exactly one producer, i, and one consumer k[6]. At each cycle of the producer, Γ_{ij} tokens are injected into channel j; similarly, at each cycle of the consumer, $-\Gamma_{kj}$ tokens are extracted. Communication systems have periodic behavior: time is divided into frames which are separated by guard periods. FIFOs are empty at the begining of a frame, and must be empty again at the end of the next guard period. There is usually one input unit, which creates or injects tokens, and one output unit, which extracts processed tokens from the system and transmits them to the outside world. Our aim now is to collect the time-related and other constraints that such a system must satisfy for proper operation.

4.2 Consistency constraints

Let N_k be the number of iterations that are executed by unit k during each frame and the associated guard period. From the assumption that the FIFOs are empty at frame start time, it follows that the number of tokens in FIFO j at the beginning of the next frame is:

$$N_i \Gamma_{ij} + N_k \Gamma_{kj} = 0 \tag{5}$$

and that this quantity must be null. It follows from this that the N_i set is a solution of a system of homogeneous linear equations. This system has a non-zero solution if and only if the determinant of its coefficient matrix is null; this is a consistency condition on the design of the system. It depends only on the Γ_i set and on the topology of the system. In the remainder of this section, we assume that the considered graph satisfies this consistency condition.

[5]In the actual hardware, connection is via an asynchronous Network on Chip. However, preliminary studies have shown that the delay and power consumption of the NoC are negligible [Grosse et al. 2006]

[6]This is reminescent of the classical rule which insure determinism of Kahn Process Networks

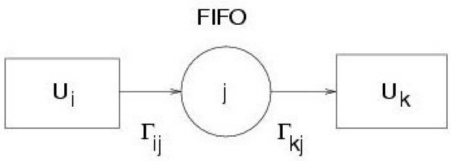

Fig. 3. A channel and its producer and consumer

4.3 Balance constraints

Let T_i be the period of unit U_i, i.e. the time interval from acquisition of its input token(s) to emission of its output token(s). If we adjust all T_i to their maximum value, $T_i = T_f/N_i$, then Eq. (5) translates into the balance equation:

$$\Gamma_{ij}/T_i + \Gamma_{kj}/T_k = 0 \tag{6}$$

which is at the heart of SDF scheduling [Lee and Messerschmidt 1987].

However, for several reasons, the system may not be able to fully satisfy these equations: one reason may be that the required period is more than the maximum period allowed by the technology:

$$T_{i,min} \leq T_i \leq T_{i,max} \tag{7}$$

We may also have to decrease T_i to meet the latency constraints (in transient mode), as explained later in Sect. 4.4. It therefore becomes necessary to relax the equality to a *balance inequality*, which expresses the necessary condition for avoiding starvation of the consumer unit.

$$\Gamma_{ij}/T_i + \Gamma_{kj}/T_k \geq 0 \tag{8}$$

When these constraints are satisfied, each unit will firstly wait for enough input tokens for its first iteration. Tokens will then accumulate in its input FIFOs faster than it can consume them, and its behavior will be periodic. Eventually, upstream units will cease producing tokens; the input FIFOs will empty and the unit will stop. When the T_k and T_f are known, computing the size of the FIFOs is straightforward.

Additionally, each mode is associated with a given throughput constraint, which is expressed as follows: at least one unit U_ω (usually the output node) must run fast enough to supply the outside world (e.g. a radio interface) with tokens at sampling period $\mathcal{T}$:

$$T_\omega \leq \mathcal{T} \tag{9}$$

4.4 Latency

Each unit U_k can be associated with an affine schedule θ_k:

$$\theta_k(n) = nT_k + L_k \tag{10}$$

which gives the epoch of the n-th activation of unit k (activations are numbered starting from 0). L_k, the time of first activation of unit k, is the *activation latency*.

Computing the activation latency – or at least having a good approximation – is important because in many cases the total latency of the design is constrained by the application.

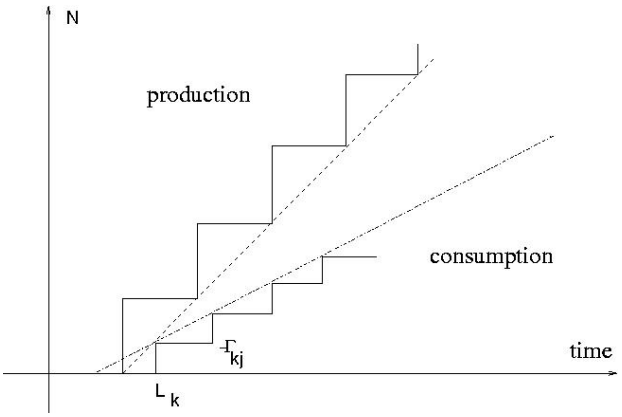

Fig. 4. Latency computation.

Let us again consider the situation of Fig. 3. At time t the producer has executed $\lfloor \frac{t-L_i}{T_i} \rfloor$ iterations and created $\lfloor \frac{t-L_i}{T_i} \rfloor \Gamma_{ij}$ tokens in channel j. At the same time, the consumer has extracted $-\lceil \frac{t-L_k}{T_k} \rceil \Gamma_{kj}$ tokens. In these formulas, we have assumed that Γ_{ij} tokens are created "in a lump" at the end of an iteration, while $-\Gamma_{kj}$ tokens are consumed, also in a lump, at the begining of an iteration. This explains why one of the formulas involves a ceiling and the other a floor. Production and consumption curves can therefore be plotted (see Fig. 4). The condition for periodic behavior is that the production curve always dominates the consumption curve.

The lower envelope of the production curve, is given by:

$$N = \frac{t - T_i - L_i}{T_i} \Gamma_{ij} \tag{11}$$

while the upper envelope of the consumption curve is:

$$N = -(\frac{t - L_k}{T_k} + 1)\Gamma_{kj} \tag{12}$$

According to constraint (8), the slope of the production envelope always exceeds the slope of the consumption envelope. Hence, it is sufficient to ensure that the first non-zero point of the second curve, $t = L_k, N = -\Gamma_{kj}$ is on the first curve. This gives:

$$\frac{L_k - T_i - L_i}{T_i} \Gamma_{ij} + \Gamma_{kj} = 0 \tag{13}$$

or:

$$L_k \geq L_i + T_i(1 - \Gamma_{ij}/\Gamma_{kj}) \tag{14}$$

In addition to these inequalities, some units (the input units) have zero latency, and at least one unit, U_α, has bounded activation latency[7]:

$$L_\alpha \leq L \tag{15}$$

We must also ensure that all tokens are processed within the frame time:

$$N_k T_k + L_k \leq T_f \tag{16}$$

There are at least two ways of handling the latency constraints. One possibility is to consider the latencies as supplementary unknowns and to let the solver handle them as such. Another possibility is to consider all paths in the communication graph from an input unit to unit U_α and to sum the corresponding instances of (14). This procedure has the advantage of eliminating half of the unknowns, at the price of introducing as many constraints as there are paths. Each new constraint is of the form:

$$\sum_{FIFO_j \in \text{path}} (1 - \Gamma_{ij}/\Gamma_{kj}) T_k \leq L \tag{17}$$

The first solution is probably better from the implementation point of view. However, we will stick to the second one when reasoning about an optimization algorithm, especially as we will see that the objective function does not depend on the latencies.

4.5 The objective function

As we have seen in Sect. 3, the energy for one cycle of unit U_k is proportional to the square of its supply voltage. Besides, we consider that period and voltage are inversely proportional to each other[8]. Hence, we may write

$$E_k = E_k^{ref} (T_k^{ref}/T_k)^2 \tag{18}$$

where T_k^{ref} is a reference period (e.g., the minimum period according to (7)) and E_k^{ref} is the corresponding energy. To obtain the total energy consumption, we have to multiply by the number of iterations per frame, and sum over all the units:

$$E = \sum_k N_k E_k^{ref} (T_k^{ref}/T_k)^2 \tag{19}$$

This is the quantity to minimize, whilst satisfying the constraints (7,8,9,14, 16) and (15). The problem has linear constraints, but a non-linear objective function.

[7]Our solution to the power minimization problem would still stand if there were several latency constrained units.

[8]This commonly used assumption is far from obvious for deep submicron technologies: we have conducted SPICE simulation including parasitic effects for our 65 nm technology. We have obtained a satisfactory linearity for the our 'usable' voltage range 0.8V - 1.2V, and when considering $0.2 < V_t < 0.5$. Besides, we will see that our solution does not depend on the exact shape of the period / voltage relation, as long as we can compute the energy gradient, either symbolically or numerically. In the same way, if static power ever becomes significant, it can be included in the objective function without materially changing the solution method.

Hence, we cannot resort to a linear solver. A two-step approach will be used instead. Firstly, the objective function will be replaced by a linear approximation which is then applied to a linear solver. Then, starting from this first approximation, an iterative algorithm will be used to reach the optimum.

4.6 Finding an initial solution

Let us consider the following objective function:

$$E' = \max_{k=1}^{N} N_k E_k^{ref} (T_k^{ref}/T_k)^2 \tag{20}$$

Since $E' \leq E \leq NE'$, it is likely that the minimum of E' and the minimum of E are reasonably close to each other. Let us introduce a new variable Z such that $E' = 1/Z^2$. Minimizing E' is equivalent to maximizing Z under the additional system of constraints:

$$Z \leq \frac{T_k}{T_k^{ref}\sqrt{N_k E_k^{ref}}} \tag{21}$$

Now, the objective function Z and all the constraints are all linear, hence the problem may be solved by the simplex algorithm. The solution may not be the minimum energy. In fact, according to (20), once the largest E_k has been minimized, the simplex algorithm may not attempt to minimize the other $E_{k'}$, an action which would reduce the total consumption (19). However, we have found experimentally that this approximate solution is always close to the actual optimum.

4.7 Improving the solution

The set of feasible solutions of our optimization problem is defined by constraints (7,8,9,14, 16) and (15). In order to simplify the presentation, we assume that the latencies have been eliminated as explained at the end of Sect. 4.4.

There are many methods for solving constrained optimization problems. In our case, since the constraints are linear and the objective function is convex, the method of Frank and Wolfe [Frank and Wolfe 1956] is especially convenient. The method is iterative; the approximate solution which has been found in the preceding section can be taken as the starting point, $T^{(0)}$. Let $T^{(k)}$ be one of the iterates. In the vicinity of $T^{(k)}$, since the energy gradient (19) is never null, we can use the following Taylor approximation:

$$E(T) \approx E(T^{(k)}) + \nabla E(T^{(k)})(T - T^{(k)}) \tag{22}$$

This is linear, and hence can be optimized under the constraints of the problem by any linear programming code. Let $T'^{(k)}$ be the optimum. A one-dimensional minimization of the energy is then executed on the segment $[T^{(k)}, T'^{(k)}]$. Note that all points of this segment are feasible. The next iterate, $T^{(k+1)}$, is the point of minimum energy. There are many algorithms for one-dimensional minimization; we used a trichotomy algorithm. Since the set of feasible points is bounded, we are in a position to apply Theorem 9 of Chapter V of [Minoux 1986], which guarantees convergence of the process to a local optimum. Furthermore, since both the feasible

set and the objective function are convex, the optimum found in this way is global. In practice, we have found that the algorithm converges in very few iterations.

5. APPLICATION TO THE FAUST SYSTEM

5.1 Extraction of the linear inequalities

We illustrate the concepts presented in Sect. 4 on the FAUST architecture. We consider transmission (Tx) and receipt (Rx) as two independent processes. Thus, each mode m of both processes leads to a different flowgraph representation.

In the remainder, we only consider one graph among the $2 \times m$ which characterize the air interface. We first extract a spanning tree from the incidence matrix Γ_{ij} representing the SDFG. Each edge in the spanning tree generates a balance constraint (8). Furthermore, application of (5) to each tree edge allows calculation of the N_i set. The consistency of the dataflow graph is then verified by checking that (5) is still valid for non-tree edges. As a consequence, the inequalities (8) for non-tree edges are redundant. In fact, violation of the consistency constraints would reveal a serious design flaw.

5.1.1 *Structural constraints.* As explained earlier in section 4.3, the key of our method is to accelerate some units beyond the needs of the throughput constraint in order to accomodate either a latency constraint (15) or a technological constraint (7). To avoid starvation, this acceleration has to be propagated backwards – from consumer to producer – with the help of (8), rewritten in linear form:

$$\Gamma_{ij} T_k + \Gamma_{kj} T_i \geq 0 \tag{23}$$

The system also obeys *technological constraints*, which bound the values T_i for each unit. It is useful to break the period T_i down into three terms:

—The core data acquisition time, which is function of the operating frequency f_i, the amount of incoming data D_{i_i}, and the input throughput $\phi_i f_i$[9]. The core data acquisition time $T_{i,0}$ can be calculated by:

$$T_{i,0} = \frac{D_{i_i}}{\phi_i f_i} \tag{24}$$

—The computation time, i.e. the number of clock cycles N_{c_i} which depends only on the mode m:

$$T_{i,1} = \frac{N_{c_i}}{f_i} \tag{25}$$

—The core data restitution time which is function of the frequency f_i, the amount of outgoing data (D_{o_i}), and the output throughput of the unit (ψ_i)

$$T_{i,2} = \frac{D_{o_i}}{\psi_i f_i} \tag{26}$$

[9]We define the input and output throughput as the number of elementary 32-bit data packets (called "flits" in FAUST) which can be transferred from the input buffer to the core (ϕ_i) and from the core to the output buffer (ψ_i) in one clock cycle.

The parameters D_{i_i} and D_{o_i} may be easily derived from the incidence matrix Γ, in which the (i, j) entry represents the size of the token produced or consumed by unit j on link i. Thus, the values of D_{i_i} and D_{o_i} can be calculated by simple summing:

$$D_{i_i} = \sum_{j:\Gamma_{i,j}<0} |\Gamma_{i,j}| \tag{27}$$

$$D_{o_i} = \sum_{j:\Gamma_{i,j}>0} \Gamma_{i,j} \tag{28}$$

As a result, the elementary period T_i is equal to:

$$T_i = T_{i,0} + T_{i,1} + T_{i,2} = \frac{1}{f_i}(\frac{D_{i_i}}{\phi i} + \frac{D_{o_i}}{\psi i} + N_{c_i}) \tag{29}$$

As explained earlier, we assume that the frequency of unit j is proportional to the supply voltage V_i. T_i is therefore related to V_i via an experimental parameter β_i such that: $T_i = 1/\beta_i V_i$. This parameter is obtained from the simulations. Finally, since the input voltage is between V_{min} and V_{max}, we obtain:

$$T_{i,min} = \frac{1}{V_{max}\beta_i}(\frac{D_{i_i}}{\phi i}+\frac{D_{o_i}}{\psi i}+N_{c_i}) \leq T_i \leq T_{i,max} = \frac{1}{V_{min}\beta_i}(\frac{D_{i_i}}{\phi i}+\frac{D_{o_i}}{\psi i}+N_{c_i}) \tag{30}$$

Since all values depend on the design (D_{i_i},D_{o_i}, ϕi,ψi) or the technology (V_{max},V_{min}) or both (β_i), we simply obtain a set of upper (resp. lower) bounds $T_{i,max}$ (resp. $T_{i,min}$) for each T_i.

5.1.2 *Applicative constraints.* The *throughput constraint* is easily derived from the sampling rate in a transmission system. It defines the period T_ω in equation (9) for the sink or source units (the last one for the Tx and the first one for the Rx). The set of *latency constraints* are obtained by applying equation (17).

5.1.3 *The objective function.* The expression of the objective function is obtained from equation (19).

5.2 Numerical results

We applied the method described above to the FAUST architecture running the specific 4More wireless transmission protocol ([Kaiser et al. 2004]). The transmission and receipt chains are represented in Figure 5 and Figure 6

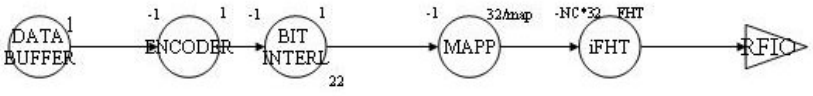

Fig. 5. FAUST Tx chain communication graph

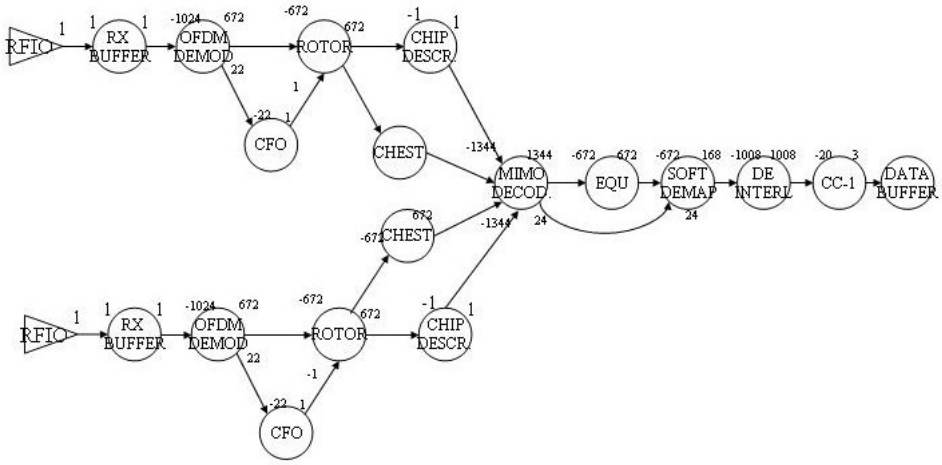

Fig. 6. FAUST Rx chain communication graph

We ran our algorithm on two typical MC-CDMA transmission/receipt scenarios (full-load and half-load mode) [Kaiser et al. 2004] with technology parameters corresponding to a 130 nm and 65 nm bulk CMOS implementation of our design. Some significant parameters of our scenario are listed hereafter:

—The minimal admissible throughput is 1,048 Mb during a 666μs time frame for transmission (Tx). It increases to 2,450 Mb for receipt (RX). This gives the values for $\mathcal{T}$ in Eq. (9) for full-load mode.

—For both processes, we considered the latencies for processing the first sample: this gives the activation latencies for both processes: $L_{Rx} \leq 650\mu s$ and $L_{Tx} \leq 24\mu s$.

—The maximum admissible supply voltage: 1.2V for a frequency of 200 Mhz (for 65 and 130 nm design).

—The minimum admissible supply voltage depending on the technology used: 0.6 V for 130 nm ; 0.4 V for 65 nm based design.

Using the simulation setup described in [Grosse et al. 2006][10], we compared our results with two other energy policies:

—*Nominal* dissipation is obtained by using RTL design techniques alone, e.g. gated clocks.

—In what we call *balanced* DVS, the frequency scaling is linearly derived only from the balance equations. We obtained these results by implementing an algorithm very similar to the "rate derivation" method described in [Niyogi and Marculescu 2005].

[10]actual measurements will be available with the 65nm release of the complete system

Figures 7 and 8 compare the results of these three methods considering the active and idle power dissipation. The ratio include the energy contributions which we have considered as constant, i.e. static power, NOC, CPU.

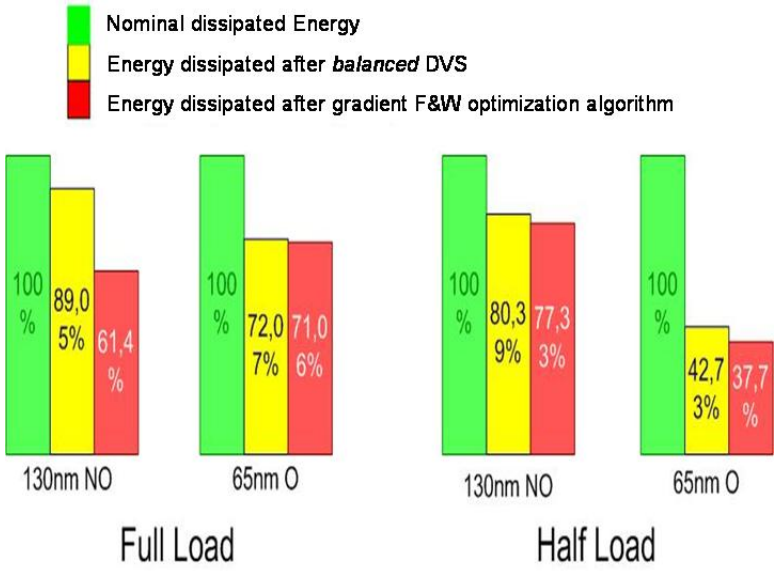

Fig. 7. Comparison of power optimization algorithms for TX chain

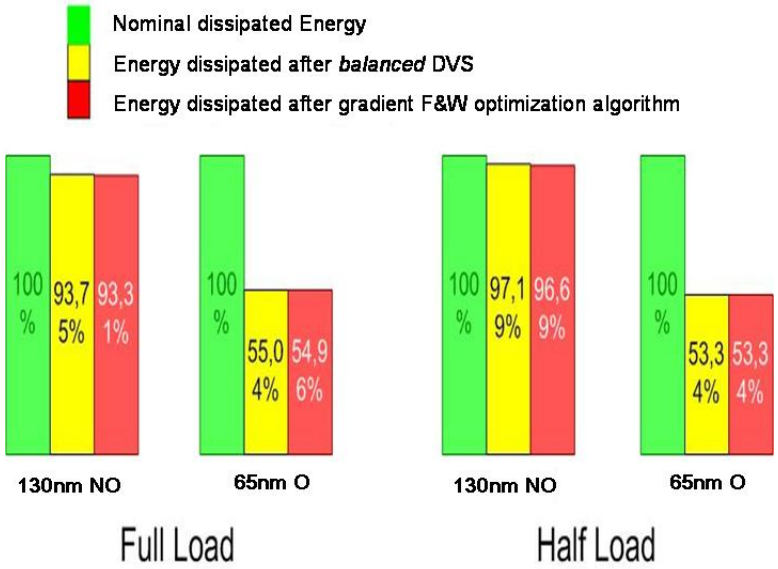

Fig. 8. Comparison of power optimization algorithms for RX chain

Simulations compare the average savings obtained by our power optimization algorithm regarding the nominal energy dissipated by the baseband modem (given for a slot transmission with all units at V_{max}) for all admissible latency values. Analysis of these results leads to the following conclusions:

—Compared to a *balanced DVS*, the Frank and Wolfe optimization algorithm allows unbalanced unit throughput. Saturating the constraints of the less consuming units enables the time constraints of the other units to be relaxed and leads to optimal power consumption.

—After completion of the simplex, the Frank and Wolfe algorithm iterates three times on average and fullfills the Kuhn Tucker conditions for every simulated scenario. This guarantees the optimality of the result.

—Our optimization is more effective on a 65nm implementation. Compared to a 130nm technology, 65nm bulk CMOS transistor runs faster with the same supply voltage. Moreover, the supply voltage can be reduced to 0.4V with this technology (compared to 0.6V for 130nm). Therefore, time constraints are de facto relaxed and the supply voltage can be further reduced and leads to better optimization.

—For the Rx chain, the results of the two DVS optimization algorithms are limited by the performance constraint of a single unit: the Chip de-scrambler. This unit is so tightly constrained that our algorithm cannot reduce its frequency (while its consumption is insignificant); hence, the two algorithms achieve the same optimum.

—The results of optimization can be taken as baseline for design of the system. For instance, it may be useful to improve the performance of some blocks (*e.g.* "Chip de-scrambling" and OFDM for RX, OFDM for TX) and the consumption of others (*e.g.* OFDM in RX and TX chain, Mimo decoding in RX chain). Thus, our power optimization method gives precious hints for improving the system.

6. CONCLUSION

In this study, we have demonstrated a power optimization method which is applicable to any complex data-flow SOC, although it is specially tuned for self-scheduled, distributed systems running predictable scenarios. The method individually optimizes the frequency of each functional unit according to the global constraints of the scenario in terms of throughput and latency. We have shown the benefit of taking into account the initial transient phase of the processing in the optimization. The method is particularly suitable for systems that are dimensioned for worst-case conditions, as is often the case for high data-rate telecommunications systems: it helps to reduce energy for less demanding modes without altering the functional behaviour.

Our technique saves around 40 % of energy compared to application of an on-off policy alone. However, it is only applicable when fine-grain voltage and frequency tuning is available on the system: in other words, it may be applied to GALS designs where the voltage of functional units may be controlled individually. We strongly believe that this prerequisite will be satisfied by most complex SOCs in the near future.

The study presented here also assumes a linearity between the supply voltage and the traversal delay which may need to be reconsidered for newer technologies, or if a larger voltage range is considered. However, this only impacts the objective function, and therefore the gradient value. The other constraints (applicative, technological) may be safely transposed for smaller technologies.

One strong point is that the method is easily applicable, and will be applied on LETI's complex SOCs. Its overhead on the embedded control software is low in terms of memory footprint, and does not consume CPU time since the frequencies are computed off-line for each different configuration. When used during the design phase, it enables fine tuning of the size of the synchronization FIFOs between the different processing elements of the system.

REFERENCES

Alimonda, A., Carta, S., Acquaviva, A., and Pisano, A. 18-20 Oct. 2006. Non-linear feedback control for energy efficient on-chip streaming computation. *Industrial Embedded Systems, 2006. IES '06. International Symposium on*, 1–8.

Beigné, E., Miermont, S., Valentian, A., Vivet, P., Barasinski, S., Blisson, F., Kohli, N., and Kumar, S. Sept 2008. A fully integrated power supply unit for fine grain dvfs and leakage control validated on low-voltage srams. In *ESSCIRC 2008*.

Durand, Y., Bernard, C., and Lattard, D. 2005. FAUST: On-chip distributed architecture for a 4G baseband modem SoC. In *Design & Reuse IP-SoC, Grenoble, France*. IEEE Computer Society, Grenoble, France.

Frank, M. and Wolfe, P. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly 3*, 95–110.

Ghamarian, A. H., Geilen, M. C. W., Stuijk, S., Basten, T., Theelen, B. D., Mousavi, M. R., Moonen, A. J. M., and Bekooij, M. J. G. 2006. Throughput analysis of synchronous data flow graphs. In *ACSD '06: Proceedings of the Sixth International Conference on Application of Concurrency to System Design*. IEEE Computer Society, Washington, DC, USA, 25–36.

Grosse, P., Durand, Y., and Feautrier, P. 2006. Power modelling of a noc based design for high speed telecommunication systems. In *Proceeding of the PATMOS 2006 conference*. Vol. Springer Verlag LNCS.

Kaiser, S., Durand, Y., Hérault, L., Hélard, J., Mottier, D., Gameiro, A., Rodriguez, J., Barquinero, C., Berens, F., Bauer, F., Rabineau, R., and Casajus, F. J. 2004. 4G MC-CDMA Multi Antenna system on chip for Radio Enhancements (4MORE). In *Proceedings of 13th IST mobile and wireless communications summit in Lyon, France*.

Khouri, K. S. and Jha, N. K. 2002. Leakage power analysis and reduction during behavioral synthesis. In *Transaction on VLSI Systems 2002*.

Lattard, D., Beigne, E., Clermidy, F., Lemaire, R., Durand, Y., Vivet, P., and Berens, F. 2008. A reconfigurable baseband platform based on an asynchronous network-on-chip. In *IEEE Journal of Solid-State Circuits, vol.43, no.1*. IEEE Computer Society, San Francisco, USA, pp.223–235.

Lee, E. and Messerschmidt, D. 1987. Synchronous data flow. In *Proceedings of the IEEE*. Vol. 75. IEEE.

Miermont, S., Vivet, P., and Renaudin, M. 2007. A power supply selector for energy- and area-efficient local dynamic voltage scaling. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*. Vol. 4644/2007. Springer.

Minoux, M. 1986. *Mathematical Programming, Theory and Algorithms*. J. Wiley.

Nielsen, L., Niessen, C., Sparsø, J., and van Berkel, K. 1994. Low-power operation using self-timed circuits and adaptative scaling of the supply voltage. In *IEEE transactions on very large scale integration (vlsi) systems vol. 2, no4*. IEEE Computer Society, San Francisco, USA, pp.391–397.

Niyogi, K. and Marculescu, D. 2005. Speed and voltage selection for gals systems based on voltage/frequency islands. In *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*. ACM, New York, NY, USA, 292–297.

Phan Huy, D., Legouable, R., Kténas, D., Brunel, L., and Assaad, M. 2008. Downlink b3g mimo ofdma link and system level performance. In *IEEE 67th Vehicular Technology Conference*. IEEE Computer Society.

Pillai, P. and Shin, K. 2001. Real time dynamic voltage scaling for low power embedded operating systems. In *ACM Todaes*. Vol. ACM. 89–101.

Saputra, H., Kandemir, M., Vijaykrishan, N., Irwin, M., Hu, J., Hsu, C.-H., and Kremer, U. 2002. Energy-conscious compilation based on voltage scaling. In *LCTES02*. 2–11.

Suhaib, S., Mathaikutty, D., and Shukla, S. 2007. Dataflow architectures for gals. In *Formal Methods for Globally Asynchronous Locally Synchronous Design*.

Wu, Q., Juang, P., Martonosi, M., Peh, L., and Clark, D. 2005. Formal control techniques for power-performance management. *IEEE Micro 25,* 5, 52–62.

Xie, F., Martonosi, M., and Malik, S. 2003. Compile-time dynamic voltage scaling settings: Opportunities and limits. In *Proceedings of the ACM SIGPLAN Conference on Programming Languages Design and Implementation (PLDI 03)*.