Dynamic and Distributed Frequency Assignment for Energy and Latency Constrained MP-SoC

Diego Puschini*†, Fabien Clermidy*, Pascal Benoit†, Gilles Sassatelli† and Lionel Torres†
*CEA, LETI, MINATEC, F38054 Grenoble, France
†LIRMM, CNRS and University of Montpellier 2, Montpellier, France

***Abstract*—In this paper we present an adaptive technique to locally adjust the frequency of processing elements on MP-SoC. The proposed method, based on Game Theory, optimizes the system while fulfilling dynamic constraints. A telecom test-case has been used to demonstrate the effectiveness of our technique. For the evaluated scenario, the proposed technique has obtained up to 20% of latency gain and 38% of energy gain.**

I. Introduction

Energy saving is one of the biggest challenges designers have to face for Multiprocessor System-on-Chip (MP-SoC) integrating several processing elements (PE). With future systems supporting a large variety of real-time applications, it is natural to think that power management has to be self adaptive. The objective is to develop a system able to adapt itself to new applicative requirements as well as changes in the environment. As an example, the available energy for a telecom application is strongly reduced under weak battery conditions on mobile terminals while real-time constraints depend on the telecom configuration. In this paper, we propose an adaptive and distributed technique to select the frequency of local processors or subsystems while fulfilling energy consumption and applicative latency constraints.

A. Motivation

Scalability and dynamic reconfiguration for adaptability of distributed systems are widely discussed in the bibliography of MP-SoC design challenges [1]. Fine-grain Dynamic Voltage and Frequency Scaling (DVFS) has been explored in [2], and [3] demonstrates their efficiency. The problem is how to optimize the system under given constraints, i.e. defining a method for choosing the frequencies of each processor. For instance, consider telecom mobile terminals. The battery-powered feature limits energy consumption. On the other side, real time is required, leading to latency-bounded applications. Between these two constraints, the system can be optimized for different objectives. We propose to analyze two cases. In the the first one, the system should optimize itself to minimize the energy consumption while ensuring the latency with a given energy budget. Examples of this are mobile systems under very-weak battery condition where the power consumption is the priority. The second scenario corresponds to the application latency minimization for real-time applications with given energy budget. The system should minimize the applicative latency to free PE time to process other tasks: the priority is the performance improvement. We present a technique for distributed dynamic frequency assignment on MP-SoC with fine-grain DVFS to solve these two scenarios. We present a new algorithm and we analyze it using a real context: a telecom latency-constrained application under limited energy conditions. When the constraints are externally modified our algorithm adapts the system to the new requirements.

B. Related Works

Setting parameters on local processors of MP-SoC is studied as an optimization problem. A large number of papers deal with static optimization and some of them also provide a dynamic method to select the best configuration found at design time. In [4], authors build configuration tables at design time by using convex optimization and dynamically choose the appropriate solution among the pre-calculated set. Some authors provide run-time optimization methods for parametric management. In [5], a centralized method is used to set frequencies and voltages of GALS systems. Nevertheless, they do not scale with the number of processors, becoming inappropriate for distributed designs [1]. Finally, in [6] game theory has been used to dynamically optimize the temperature and the synchronization between tasks on a MP-SoC on a distributed way. Nevertheless, it does not explore constrained scenarios. To the best of our knowledge, no solution has been presented to dynamically adapt the frequencies in a distributed way on MP-SoC under latency and energy constraints.

II. System-Level Model

The dynamic optimization proposed is inspired by game theory, where a *non-cooperative game* is a scenario with several players interacting by actions and consequences [7]. Basically, players individually choose an action within a defined set, resulting in consequences. Each player tries to maximize its outcome according to its preferences, leading to global optimization. If this sequence is repeated, under certain conditions, the game finds a solution formalized as Nash Equilibrium. These principles provide strong concepts to model the behavior of reactive systems where decisions are taken in a distributed and dynamic way, justifying the choice of the game theory for our approach.

We model the PEs as players and the latency and power consumption as a local objective function u_i that depends on the global state of other PEs. Then, a distributed algorithm selects the best solution. The objective functions are built by using different terms: EC_i models the energy contribution

978-3-9810801-5-5/DATE09 © 2009 EDAA

of each PE to the whole consumption, LC_i describes the applicative latency contribution to the total latency and E_{pen} and L_{pen} are penalty functions modeling energy and latency constraints. We consider a MP-SoC composed of n PEs interconnected by an asynchronous Network-on-Chip (NoC). Each PE integrates a DVFS that regulates the local voltage and frequency couple among a finite number of solutions. We denote T_i the clock period corresponding to PE-i and $T_{i-} = (T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_n)$ the periods of all other PEs in the MP-SoC. We assume that an external mechanism has mapped an application on the MP-SoC, each PE handling a unique task. The task assigned to PE-i takes N_i cycles of T_i to be processed. We denote as L_{max} the system latency constraint and we consider that each task is launched every T_0 seconds. In the next sections, energy and latency contributions and energy and latency constraints are modeled. Then, two objective functions are built according to the formulated scenarios, i.e. energy or latency minimization under conditions.

A. Energy Model

The energy of the whole PE-i (including processors, memories or other subsystems) during T_0 is given by the static and dynamic consumption. We consider the first term as constant, being $E_{stat} = T_0\ P_{stat}$. The dynamic consumption is a function of PE-i activity and is modeled as follows. The task assigned to PE-i is processed in $N_i T_i$ seconds. During this period, PE-i has a high activity and consumes $E_{Hdyn,i}$. During the rest of the time until the task is launched again, the PE is in low-activity state, due to classical low-power techniques such as gated clocks. Thus, during $(T_0 - N_i T_i)$ seconds, it consumes $E_{Ldyn,i}$. The total energy consumption for PE-i during T_0 becomes $E_i = T_0\ P_{stat} + E_{Hdyn,i} + E_{Ldyn,i}$. The dynamic power is $P_{dyn} = \alpha {V_N}^2 / T_N$, where α includes the circuit activity, V_N is the nominal voltage supply and T_N is the nominal clock period. Thus, the dynamic energy needed to feed the block during a nominal period is $E_N = P_{dyn} T_N = \alpha {V_N}^2$. Considering that α remains constant, for a different voltage supply V we obtain $\alpha = E_N / {V_N}^2 = E/V^2$. Thus, after applying local DVFS the energy becomes $E = E_N\,(V/V_N)^2$. Since the circuit frequency is proportional to the voltage supply, the dynamic energy consumption is $E = E_N\,(T_N/T)^2$. Using these results in E_i, consumption of PE-i becomes:

$$\begin{aligned} E_i(T_i) &= T_0 P_{stat,i} + N_i E_{N,i} {T_{N,i}}^2 / {T_i}^2 + \\ &+ \gamma_i\,(T_0/T_i - N_i)\,E_{N,i} {T_{N,i}}^2 / {T_i}^2 \end{aligned} \quad (1)$$

where $E_{N,i}$ is the reference energy consumption of PE-i during a nominal clock period $T_{N,i}$ and $(1-\gamma_i)$ is the energy reduction in low-activity state. The consumption of each PE allows to deduce the contribution of PE-i to the total energy as:

$$EC_i(T_i, T_{i-}) = \rho_i E_i(T_i) / \sum_k E_k(T_k) \quad (2)$$

where $\rho_i = N_i / \sum_k N_k$ is the weight of PE-i in terms of number of clock cycles required over the total application requirements. Expression (2) has the form of figure 1(a).

B. Latency Model

We define L_i the applicative latency added by PE-i. The latency contribution of PE-i is described by:

$$LC_i(T_i, T_{i-}) = \rho_i L_i(T_i) / \sum_k L_k(T_k) \quad (3)$$

where ρ_i is the weight of PE-i in terms of number of clock cycles required over the total application requirements and $\sum_k L_k(T_k)$ is the total latency of the application. Equation (3) has the form of figure 1(b).

C. Energy Constraint

When the system is running a given application, each PE consumes E_i (equation (1)) to process the assigned task. In order to keep the whole system energy under a given constraint E_{max}, we penalize the solutions that exceed the allowed consumption by introducing the next term in the objective function of each PE:

$$E_{pen,i} = \begin{cases} a(T_{me} - T_i) & \text{if } \sum_k E_k(T_k) > E_{max} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $\sum_k E_k$ is the total energy given T_i and T_{me} is the smallest clock period for PE-i that satisfies the constraint. a is a constant defining how much the solutions are penalized. This penalty function discards solutions for T_i that do not satisfy the energy constraint by increasing the result of the objective function in an amount equal to $a(T_{me} - T_i)$. Equation (4) has the form of figure 1(c).

D. Latency Constraint

Tasks assigned to PEs introduce computational latencies. The whole latency should not exceed L_{max} for real-time purpose. We consider the critical path of the applications, meaning that the total latency is the sum of all local latencies. In order to keep the whole latency under L_{max}, we penalize the solutions that exceed this value by introducing the next term in the objective function:

$$L_{pen,i} = \begin{cases} 0 & \text{if } \sum_k L_k(T_k) \leq L_{max} \\ a(T_i - T_{ml}) & \text{otherwise} \end{cases} \quad (5)$$

where $\sum_k L_k$ is the total latency given T_i, a is the same constant used in equation (4) and T_{ml} is the biggest clock period for PE-i that satisfies the latency constraint. By increasing the objective function in an amount equal to $a(T_i - T_{ml})$, $L_{pen,i}$ discards solutions that do not satisfy the latency constraint. $L_{pen,i}$ has the form of figure 1(d).

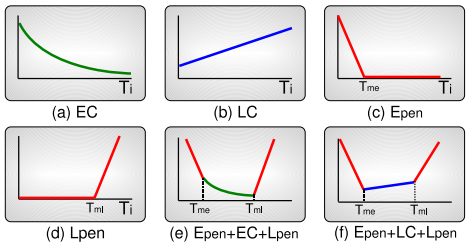


Fig. 1. Objective functions and their components

E. Objective Function for Energy Minimization

The energy minimizing objective function under constraints is built by using equations (2), (4) and (5) as follows:

$$u_{e,i}(T_i, T_{i-}) = -(E_{pen,i} + EC_i + L_{pen,i}) \quad (6)$$

The minus sign denotes that the optimization problem is defined as a maximization. Typically, $-u_{e,i}$ has the form of figure 1(e). When maximizing this objective function, the solutions are bounded by the energy and latency constraints (T_{ml} and T_{me}) and minimizes the energy consumption. If constraints overlap, i.e. $T_{ml} < T_{me}$, the solution maintains the energy constraint violating the maximal latency: since $E_{pen}+L_{pen} = 0$ when $T_{ml} < T_i < T_{me}$ the decision is driven by EC, minimizing the energy consumption and respecting the energy budget.

F. Objective Function for Latency Minimization

The latency minimizing objective function under constraints is built by using equations (3), (4) and (5) as follows:

$$u_{l,i}(T_i, T_{i-}) = -(E_{pen,i} + LC_i + L_{pen,i}) \quad (7)$$

Figure 1(f) shows the form of $-u_{l,i}$. Now, the solutions are bounded by the energy and latency constraints and minimize the latency. If constraints overlap ($T_{ml} < T_{me}$), the solution maintains the latency constraint violating the maximal energy: since $E_{pen} + L_{pen} = 0$ when $T_{ml} < T_i < T_{me}$, the decision is driven by LC, minimizing the latency.

Depending on the MP-SoC characteristics, applications and system objectives, $u_{e,i}$ or $u_{l,i}$ are chosen as the objective function u_i of each PE. $u_{e,i}$ can be used for systems with hard energy requirements and real-time constraints which can be loosen. $u_{l,i}$ can be used for systems with real-time constraints. In these cases, an energy overflow should trigger a higher-level reconfiguration as shown in the test-case below.

III. Dynamic Optimization

Each u_i of the n functions, presented in the previous section, is maximized in a distributed way by Algorithm 1. It is dynamically executed in parallel in each PE to choose the local T_i that maximizes u_i constrained by decisions made by other PEs, T_{i-}. A decision loop is then performed. The DVFS can set frequencies within a discrete number of clock periods with a granularity ΔT between T_{max} and T_{min}. We assume that the last local choice (or initial condition) T_i is known.

Algorithm 1 : Local Decision Maker

Require $u_i, \Delta T$, **Ensure** T_i
loop
  get T_{i-}
  convert $u_i(T_i, T_{i-}) \rightarrow u_i^*(T_i)$
  maximize $u_i^*(T_i)$
end loop

The decision loop is composed of three operations. The first one obtains from other PEs their last decisions T_{i-}. The second instruction converts $u_i(T_i, T_{i-})$ into a scalar function $u_i^*(T_i)$ by fixing T_{i-} as parameters and T_i as a variable. Then, the last instruction maximizes the objective function. This is performed by Algorithm 2 that maximizes u_i^* by affecting the former T_i with a ΔT step, obtaining nT_i. Algorithm 2 is composed of few simple operations (additions, multiplications and comparisons), maintaining low complexity. Since u_i is continuous and concave on the space T_{min} to T_{max} with steps ΔT and this space is a nonempty compact convex subset of an Euclidean space; the existence of a Nash Equilibrium solution is guaranteed [7].

Algorithm 2 : u_i^* Maximization

Require $T_i, \Delta T, u_i^*$, **Ensure** nT_i
if $T_i = T_{max}$ **then**
  if $u_i^*(T_i) < u_i^*(T_i - \Delta T)$ **then** $nT_i \Leftarrow T_i - \Delta T$ **else** $nT_i \Leftarrow T_i$
else if $T_i = T_{min}$ **then**
  if $u_i^*(T_i) < u_i^*(T_i + \Delta T)$ **then** $nT_i \Leftarrow T_i + \Delta T$ **else** $nT_i \Leftarrow T_i$
else
  if $u_i^*(T_i) < u_i^*(T_i - \Delta T)$ **then** $nT_i \Leftarrow T_i - \Delta T$
  else if $u_i^*(T_i) < u_i^*(T_i + \Delta T)$ **then** $nT_i \Leftarrow T_i + \Delta T$
  else $nT_i \Leftarrow T_i$
end if

IV. Results

A. Test-Case

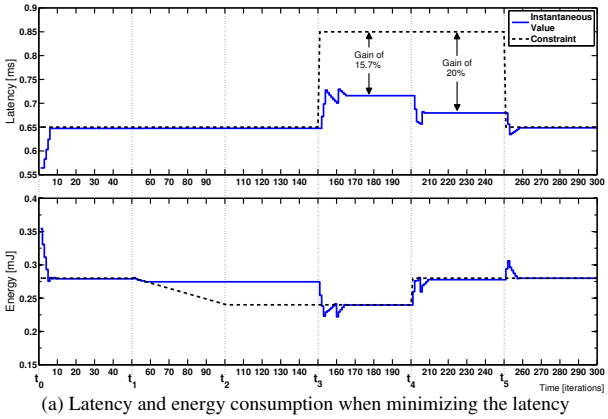
The test-case under consideration is a telecom System-on-Chip integrated in 130nm CMOS and interconnected by an asynchronous NoC [8]. In this paper we consider the TX subsystem composed of 5 heterogeneous PEs. We have applied distributed DVFS [3] able to set the frequency among 32 values between 50 and 300MHz. The application under consideration is an advanced telecommunication protocol, the MC-CDMA Matrice [9]: TX QPSK, 1-user, rate 1/2 and spreading 8, offering a bandwidth of approximately 890Kbits/s. It is composed of five tasks (figure 2): encoder, interleaving, mapping, FHT and OFDM modulator. It is mapped on the MP-SoC by assigning one task per PE. Table I shows the number of calculation cycles needed to process the tasks (N_i) and the equivalent latency per PE measured in number of clock periods (L_i/T_i). The maximum allowed latency is 650μsec for a whole frame. The consumption of PEs is characterized in therms of energy ($E_{N,i}$) per clock cycle at 200Mhz in a post back-end netlist (table I). It has been observed reductions of 80% in idle states when using gated clock techniques ($\gamma_i = 0.2$ in equation 1).

Fig. 2. MC-CDMA TX application

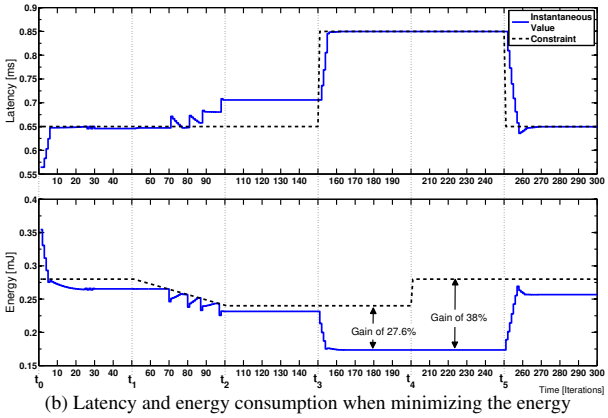
TABLE I

	Encod	Inter	Map	FHT	OFDM
N_i [cycles]	576	560	560	24496	115248
L_i/T_i [cycles]	132	128	1	24496	115248
$E_{N,i}$[nJ/cycle]*	77.62	183.18	183.18	251.50	1516.50

** Values measured on a 130nm demonstrator [8].*



(a) Latency and energy consumption when minimizing the latency



(b) Latency and energy consumption when minimizing the energy

Fig. 3. Temporal simulation results

Behavioral simulations have been made with Matlab using the two objective functions of sections II-E and II-F. The constraints were externally modified during the simulation time. The maximal energy E_{max} is 2.8mJ between instants t_0 and t_1. Simulating external factors, e.g. low battery in mobile systems, the available energy is decreased between t_1 and t_2, to reach 2.4mJ at t_2. Then, it is remained constant until t_4, when it is augmented to 2.8mJ. On the other side, the latency constraint L_{max} is 650μsec. Nevertheless, since it has been observed that a solution is unattainable at very low consumption (less than 2.65mJ), it has been relaxed to 850μsec between t_3 and t_5. Finally, after t_6 the constraints are $E_{max} = 2.8$mJ and $L_{max} = 650\mu$sec as in t_0. The next sections explain the results when optimizing the energy consumption or the application latency under these constraints.

B. Latency Minimization Results

Figure 3 shows temporal results for each configuration. Figure 3(a) shows the latency and energy consumption for the system minimizing the latency. The x-axis represents the time in number of iteration cycles. After a transient time after t_0, the system finds a solution fulfilling both constraints. After t_1 and once it finds that it can no more guarantee both constraints, it adjusts the frequencies to fulfill at least the maximal latency. Once the latency constraint is relaxed in t_3, the system adapts the frequencies to again ensure both constraints. After t_4, the energy budget has been augmented, leaving the system to reconfigure it self to reduce the latency. Finally, when both constraints were reinitialized, a new solution is found offering similar performances than before t_1.

C. Energy Minimization Results

Figure 3(b) shows the latency and energy for the optimization of the energy minimization. Note that now, after t_1 when there is no solution fulfilling both constraints, the system only violates the latency, always guarantying the maximal energy consumption. Moreover, between t_3 and t_5, the system takes advantage of the increased allowed latency to reduce the energy consumption. Finally, after t_5 the system finds a new solution offering similar performances than before t_1 and satisfying both constraints.

D. Optimization Gain

For both scenarios, the optimization gain is measured when the constraints are loosen to allow the improvements, i.e. between t_3 and t_5. Between t_3 and t_4 it obtains 15.7% of time gain over the constraint when minimizing the latency, or 27.6% of energy reduction over the given budget when minimizing the energy. Between t_4 and t_5, it frees 20% of PE time or it reduces 38% the total energy budget.

V. Conclusion

In this paper we have presented an adaptive technique to locally adjust the frequency of processing elements on MP-SoC. The proposed method, based on Game Theory, optimizes the system while fulfilling dynamic constraints. A telecom test-case has been used to demonstrate the effectiveness of our approach. For the evaluated case, the proposed technique has obtained up to 20% of latency gain and 38% of energy gain.

References

[1] G. Martin, "Overview of the MPSoC design challenge," in *DAC '06*. San Francisco, CA, USA: ACM, 2006, pp. 274–279.

[2] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *ISCA '06*, 2006, pp. 78–88.

[3] E. Beigné, F. Clermidy, S. Miermont, A. Valentian, P. Vivet, S. Barasinski, F. Blisson, N. Kohli, and S. Kumar, "A fully integrated power supply unit for fine grain DVFS and leakage control validated on low-voltage SRAMs," in *ESSCIRC'08*, Edinburg, UK, Sept. 2008.

[4] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. D. Micheli, "Temperature control of high-performance multi-core platforms using convex optimization," in *DATE'08*. Munich, Germany: IEEE Computer Society, 2008, pp. 110–115.

[5] K. Niyogi and D. Marculescu, "Speed and voltage selection for GALS systems based on voltage/frequency islands," in *ASP-DAC '05*. Shanghai, China: ACM, 2005, pp. 292–297.

[6] D. Puschini, F. Clermidy, P. Benoit, G. Sassatelli, and L. Torres, "Temperature-aware distributed run-time optimization on MP-SoC using game theory," in *ISVLSI '08*. Montpellier, France: IEEE Computer Society, 2008, pp. 375–380.

[7] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. MIT Press, 1994.

[8] D. Lattard, E. Beigne, C. Bernard, C. Bour, F. Clermidy, Y. Durand, J. Durupt, D. Varreau, P. Vivet, P. Penard, A. Bouttier, and F. Berens, "A telecom baseband circuit based on an asynchronous network-on-chip," *IEEE ISSCC 2007. Digest of Technical Papers.*, pp. 258–601, Feb. 2007.

[9] Information Society Technology - MATRICE web site. http://www.ist-matrice.org.