Providing Better Multi-Processor Systems-on-Chip Resources Utilization by Means of Using a Control-Loop Feedback Mechanism

Gabriel Marchesan Almeida, Remi Busseuil, Sameer Varyani, Nicolas Hébert, Gilles Sassatelli, Pascal Benoit, Lionel Torres, Michel Robert

2010 International Conference on Reconfigurable Computing

Providing Better Multi-Processor Systems-on-Chip Resources Utilization by Means of Using a Control-Loop Feedback Mechanism

Gabriel Marchesan Almeida, Sameer Varyani, Rémi Busseuil, Nicolas Hebert,
Gilles Sassatelli, Pascal Benoit, Lionel Torres, Michel Robert
Laboratory of Informatics, Robotics and Microelectronics of Montpellier (LIRMM)
Department of Microelectronics
161 Rue Ada, Cedex 5, 34095
Montpellier, France
{*marchesan, varyani, busseuil, hebert, sassatelli, benoit, torres, robert*}*@lirmm.fr*

***Abstract*—In this paper we propose a strategy for better exploiting Multi-Processor Systems-on-Chip resources utilization by means of using a control-loop feedback mechanism. We apply the proposed techniques in a purely distributed memory MPSoC architecture that is composed of a frequency scaling module responsible for tuning the frequency of processors at run-time. Results show very promising in terms of adaptation capabilities for system with dynamic workload. Performance results demonstrate the effectiveness of the proposed approach when workload requirements for applications may vary, affecting the overall performance of the system. For validating the proposed approach we have implemented a multi-thread MJPEG decoder application and created an architecture model with/without perturbations in the system.**

***Keywords*-MPSoC, homogeneous, adaptive, NoC, distributed memory, RTOS**

I. Introduction

Over the years scientists in all domains have put huge efforts for better exploiting and optimizing available resources. In massively parallel systems, especially in Multi-Processor Systems-on-Chip (MPSoC) resource management is mostly modeled targeting power optimization, performance and fault-tolerance strategies in such systems.

Since this work targets massively parallel on-chip multiprocessor systems, adaptability is a major concern in the approach. In the direction of scalable systems we have developed an homogeneous MPSoC architecture with distributed memory making use of a message passing programming model. Given the large variety of possible use cases that these platforms must support and the resulting workload variability, offline approaches are no longer sufficient as application mapping paradigms, because they do not allow coping with time changing workloads. Furthermore, the large number of parameters that play a role on the platform performance makes it difficult to estimate the best system response at design time.

In [1] we have proposed an adaptive strategy that is responsible for making decisions at run-time. Decisions are taken by processors in a distributed fashion and relate mostly to application performance.

This paper goes into the direction of adaptive strategies for homogeneous MPSoC architectures. We propose a strategy for better exploiting architecture resources utilization by means of using a control-loop feedback mechanism and mostly addresses both the benefits brought by this technique as well as the associated optimization. The new architecture relies on a Real-Time Operating System (RTOS) with support for semaphores, mutexes and task priority based scheduling algorithm. Moreover, a tiny implementation of a communication stack comprising UDP TCP/IP protocols is used aimed at providing important concepts such as ports and enabling advanced features such as reordering, rerouting, error detection, etc. [2].

The rest of the paper is organized as follows. The next section presents related works. Section III describes the proposed architecture. Section IV discusses the proposed controller while experimental evaluations are presented in Section V. Finally some conclusions are drawn in Section VI.

II. Related Works

A. Adaptation in MPSoCs Architectures

Recently, researchers have put focus on adaptation techniques in order to handle with dynamic and unpredictable behaviors that can appear in nowadays embedded systems. This section presents some works that have been conducted in this direction using techniques as such as task migration mechanisms and dynamic voltage and frequency scaling.

1) Task Migration Mechanisms: A number of work in the literature based on distributed memory systems has been using shared memory as a mean for enabling task migration [3][4][5]. In [4] each core runs a single operating system instance in its logical private memory. Processor cores execute tasks from their private memory and explicitly communicate with each other by means of shared memory. The target platform uses a shared bus as interconnect. In [6] authors propose to implement a scalable shared memory many-cores architecture with global cache coherence. The architecture is built around 4096 cores which makes use of

a logically shared memory but physically distributed, with cache coherence enforced by hardware, using a directory-based protocol.

In the case of distributed memory MPSoCs, in [1] we have proposed an adaptive strategy that is responsible for making decisions at run-time. Decisions are taken by processors in a distributed fashion and relate mostly to application performance. In [2] authors address both the benefits brought by task migration mechanisms as well as the associated performance penalty in a purely distributed homogeneous MPSoC architecture.

Taking into account the future homogeneous MPSoC systems, scalable architectures with purely distributed memory system are suitable. To the best of our knowledge, our architecture is the only one with a purely distributed memory system in which does not rely on using shared memory for enabling task migration [2]. Instead, it uses the NoC as a communication link where the tasks are transmitted during the migration process.

2) Dynamic Voltage and Frequency Scaling (DVFS): DVFS has been a widely applied technique for reducing power consumption in many domains, especially those concerning micro-architectures such as embedded systems. In [7] authors show a technique for minimizing the total power consumption of a Chip Multiprocessor (CMP) while maintaining a target average throughput. The proposed solution relies on a hierarchical framework, which employs core consolidation, coarse-grain dynamic voltage and frequency scaling (DVFS). The problem of optimally assigning dependent tasks in multiprocessor system has not been addressed in their paper.

In this paper we propose a novel solution targeting both intra/extra-communication between tasks. Intra-communications are handled by means of using local FIFOs, while extra-communication makes use of a communication stack. The model is based on Message Passing Interface (MPI) and tasks communicate with each other by exchanging messages over a Network-on-Chip (NoC) [8].

Adaptability has been explored under a number of aspects in embedded systems, ranging from adaptive modulation used in the future 3GPP-LTE standard (SDR for software defined radio) [9] to adaptive cores instantiation in dynamically reconfigurable FPGAs [10]. Two of the foremost popular techniques are here discussed.

This paper presents two major contributions:

1) Propose a novel and purely distributed memory architecture with adaptation capabilities by means of using a Proportional Integral Derivative (PID) controller;
2) Analyze and discuss the benefits of using such strategy for better exploiting architecture resources;

We demonstrate how soft-real time application requirements in homogeneous MPSoC architectures can be handled by better exploiting architecture resources using a PID controller and how we can obtain a model for the system that is essential to determine the PID controller parameters. The proposed strategy is validated by scenarios where processors frequency can be tuned (reducing/increasing) without violating soft-real time constraints.

III. SHoP Architecture

The key motivations of our approach are scalability and adaptability; the system presented in the rest of this paper is built around a distributed memory/message passing system that provides efficient support for task migration. For these reasons the architecture is named SHoP (**S***elf-adaptive* **Ho***mogeneous* **P***latform*). This system aims at achieving continuous, transparent, and decentralized run-time task placement on an array of processors for optimizing application mapping according to various potentially time-changing criteria. Figure 1 presents a structural view of the SHoP architecture. The architecture is made of a homogeneous array of Processing Elements (PEs) communicating through a packet-switching network. For this reason, the PE is called Network Processing Unit (NPU).

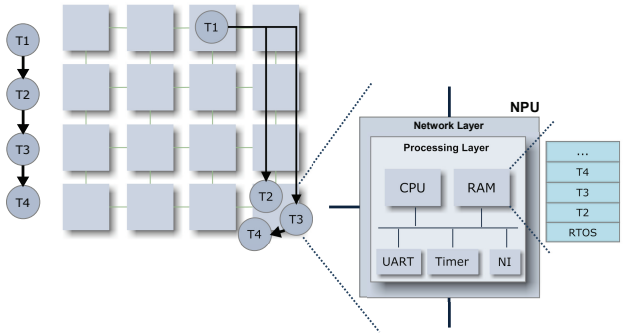

Figure 1. Structural View of the SHoP Architecture

The NPU is built of two main layers, the network layer and the processing layer. The network layer is essentially a compact routing engine based on the Hamiltonian Routing Algorithm [11]. The Network-on-Chip (NoC) used in this work was proposed in [8].

The processing layer is based on a simple and compact RISC microprocessor, its static memory, and a few peripherals (one timer, one interrupt controller, one UART). The processor used has a compact instruction set comparable to a MIPS-1 [12]. It has 3 pipelines stages, no cache, no Memory Management Unit (MMU), and no memory protection support in order to keep it as small as possible. A multitasking tiny Real-Time Operating System (RTOS) implements the support for time-multiplexed execution of multiple tasks. The original version of the RTOS as well the RTL description of the processor used as part of this work are available at [13].

IV. PID Controller

A Proportional Integral Derivative (PID) controller is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller calculates an *error* value as the difference between a measured process variable and a desired *setpoint*. The controller attempts to minimize the error by adjusting the process control inputs. In the absence of knowledge of the underlying process, a PID controller is the optimal controller [14]. However, for best performance, the PID parameters used in the calculation must be tuned according to the nature of the system while the design is generic, the parameters depend on the specific system.

In this paper we propose the usage of a PID controller for adjusting the appropriated frequency of the processors at the same time as deadline miss ratio is reduced. Figure 2 summarizes a traditional PID controller.

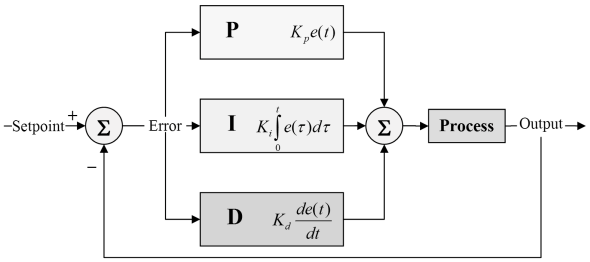

Figure 2. PID Controller

The PID controller algorithm involves three-term control: the **P***roportional*, the **I***ntegral* and **D***erivative* values. The proportional value determines the reaction to the current error, the integral value resolves the reaction based on the sum of recent errors, and the derivative value sets the reaction based on the rate at which the error has been changing [14]. Heuristically, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change.

The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output, the final form of the PID algorithm is:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}e(t) \quad (1)$$

where:

Proportional gain, K_p: larger values typically mean faster response since the larger the error, the larger the proportional term compensation. An excessively large proportional gain will lead to process instability and oscillation.

Integral gain, K_i: larger values imply steady state errors are eliminated more quickly. The trade-off is larger overshoot: any negative error integrated during transient response must be integrated away by positive error before reaching steady state.

Derivative gain, K_d: larger values decrease overshoot, but slow down transient response and may lead to instability due to signal noise amplification in the differentiation of the error.

V. Case Study and Results

A. Introduction

In order to demonstrate the efficiency of our approach we have performed our experiments based on an implemented MJPEG decoder algorithm mapped onto the SHoP platform. The application is basically composed of four main tasks: the sender (T_1), responsible for feeding the application with the input stream and three additional tasks which belong to the MJPEG application (IVLC (T_2), IQ (T_3) and IDCT (T_4)). The initial application mapping is the following: T_1 is mapped onto NPU$_1$ while T_2, T_3 and T_4 are hosted in NPU$_2$. Figure 3 shows the MJPEG application task graph and its initial mapping.

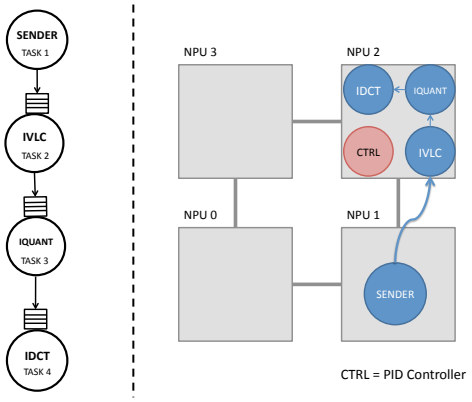

Figure 3. MJPEG Task Graph and Initial Mapping

B. System Characterization

In order to ensure optimized PID control loop, the process characterization has been performed. It has to be either linear or pseudo-linear and the dynamic response of the process has to be evaluated.

The proposed PID controller is modeled as a service provided by the RTOS and it is available in each NPU of the architecture in a distributed fashion. We will consider the *process* as a black box where input is the current processor frequency and output the application throughput. Aiming to validate its linearity, a scenario composed of steps in frequency, changing from 55 to 355MHz has been used. The observing window is consisted of the time interval between two frequencies changes. By applying this method we can consider the process at a steady state before performing a state changing.

Figure 4 shows a linear interpolation of the process according to the MJPEG *performance vs processor frequency*. Marks in blue represent obtained throughput during the time considered as steady state. The standard deviation is approximately 150KB/s, which is less than 12% of the average throughput. The linearity consideration of the proposed process can be assumed.

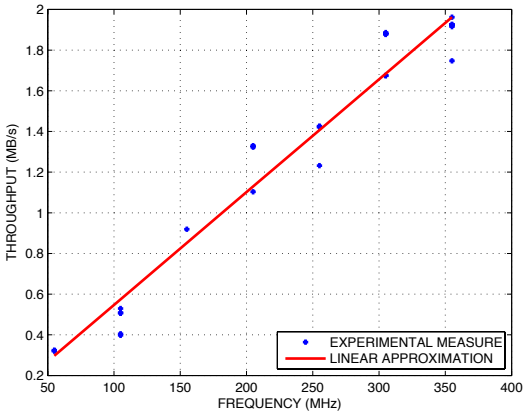

Figure 4. MJPEG Application Performance

The dynamic response of the process has been evaluated by changing the frequency from 55MHz to 355MHz. Figure 5 shows the process reactiveness when changing processor frequency. The rise time of the process is calculated in approximately $400ms$ (from $0.49s$ to $0.53s$). This value can be explained by the average necessary to reject noises due to burst of incoming frames.

Indeed, as three tasks of the MJPEG are mapped onto the same NPU, the computation power utilized by T_2, T_3 and T_4 is shared. This creates a considerable delay with no processed frames in the output.

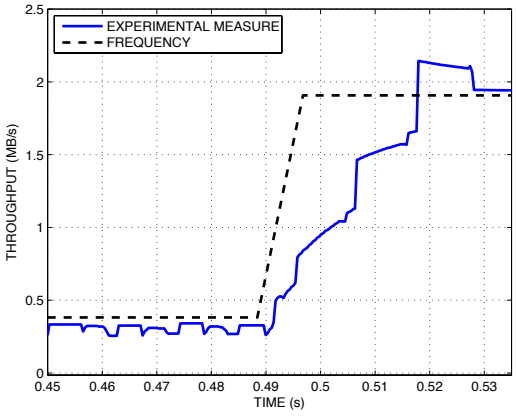

Figure 5. System Reactiveness

C. Task Migration

One of the purpose of the feedback controller is to be able to adapt the processor frequency when a perturbation occurs in the system, and in particular a task migration. To prove the efficiency of the controller, a scenario using a task migration has been created. In this scenario, two tasks communicating with each other have been mapped onto NPU_3. Later, one of those two tasks is migrated to the NPU_2, generating an overhead in the application performance of the NPU_2. Figure 6 shows the throughput variation due to this migration. The NPU frequency has been maintained at 55MHz. The task is migrated at $1.86s$, represented by the dashed line.

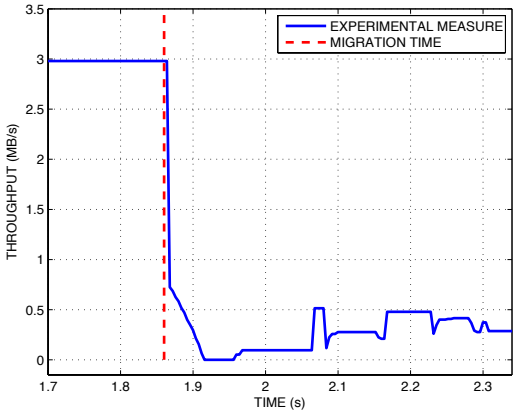

Figure 6. Migration Cost Without PID Controller Regulation

The throughput goes from 3.15MB/s down to 0.45MB/s, reducing the performance of the system. This behavior can be considered as instantaneous and it is model as perturbation. For such scenarios we are capable of measuring the cost in terms of reaction time for having a dynamic response generated by the PID controller. Hence, tuning the PID controller to reject the perturbation will be the same as tuning it for the dynamic response of the process.

D. PID Controller Tuning

Using the Takahashi method [15], we can optimize the controller only using the dynamic response as shown in Figure 5. The principle of the Takahashi method is to use the step response of a process to compute the PID coefficient, K_p, K_i and K_d. It takes into account the latency of the process and the rise time to compute the coefficient which minimize the overall error:

$$\sum_{k \geq 0} |\epsilon_k|$$

where ϵ_k is the error at sampled time k.

We demonstrate in the next section how the controller regulates the throughput of the MJPEG application in two

different scenarios: in the first one the process is constant and no perturbation is modeled. In the second scenario a task migration is performed in order to generate a perturbation in the system and the behavior of the controller is analyzed.

E. Regulation Without Perturbation

The PID controller has been implemented as a task in the architecture. Thanks to an *absolute* timer not affected by the frequency scaling, the controller function is executed at fixed time slots. The information about packets arrival is sent to the controller by the last task of the application.

In the first scenario we have modeled a MJPEG decoder which requires a minimum throughput of 1MB/s. This specification is used as a *setpoint* for the PID controller. A set of 400 frames is computed by the MJPEG application and the controller is responsible for adjusting the frequency of the processor at run-time. The system has to ensure a minimum throughput of the application according to the *setpoint*. Figure 7 shows the frequency regulation and the obtained throughput.

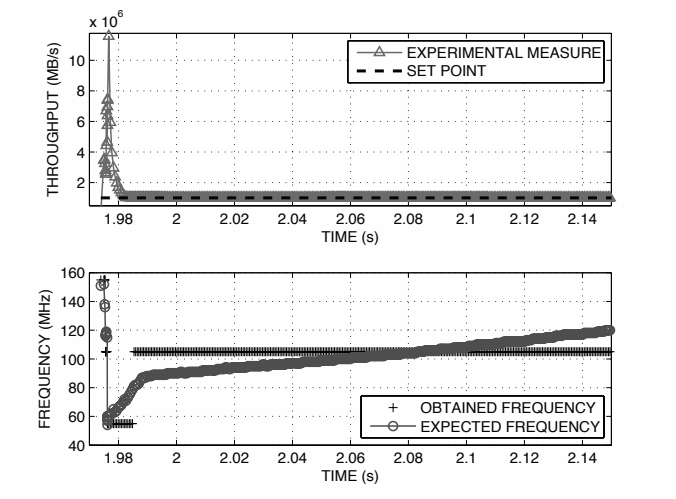

Figure 7. Frequency Regulation

We can observe a big overshoot at the beginning of the regulation. This phenomenon can be explained by two factors. First, there is a burst of incoming packets induced by the internal organization of the system that reports to the controller instantaneous information of the system. Second, traditional PID controllers produce this kind of error which is minimized at the time. These two phenomena lead to a *snowball effect*: as no packet arrives, the controller increases the processor frequency, which results to a faster computation of frames by the IVLC and IQUANT tasks, generating a burst of packets to the input of the last task (IDCT) of the application.

This overshoot should occurs only at the beginning of the computation, or in scenarios with huge variations of *setpoints* values, which can be considered as scarce phenomena.

F. Regulation With Perturbation

In order to validate the adaptability of our proposed strategy, a different scenario has been modeled with perturbations that are triggered at run-time. The scenario is similar to the one presented in the previous section. The difference is that a task migration to NPU_2 is performed in the middle of the regulation. Figure 8 shows the behavior of the proposed controller in a scenario with system perturbation. In this scenario, task migration is triggered at $1.9s$.

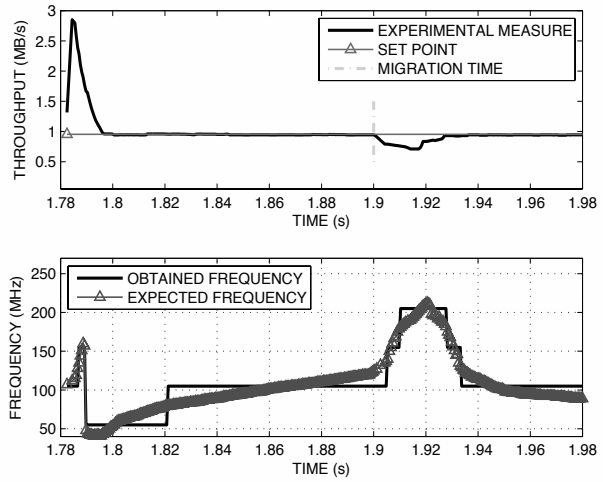

Figure 8. Frequency Regulation with System Perturbation

We can observe the efficiency of the regulation: the cost in terms of performance due to the migration process is rapidly amortized by the process: after approximately $250ms$ the application *setpoint* is again guaranteed by applying changes in the frequency. The maximum overshoot is around 330KB/s, which is coherent with the value obtained for the scenario without regulation. The different between obtained frequency and the expected frequency is due the fact that values are rounded to the closest available frequency in the architecture (55, 105, 155, 205, 255, 305 and 355MHz).

VI. Conclusions

In this paper we have proposed a strategy for better exploiting architecture resources utilization in a purely distributed memory homogeneous MPSoC architecture. Results show very promising regarding to the adaptability of the system in scenarios with dynamic workload changing.

We have demonstrated the efficiency of the proposed PID controller by presenting two different scenarios with/without perturbations. For validating our approach we have implemented a multi-thread version of the MJPEG decoder which utilizes a model based on message passing interface.

Results shown that by using the proposed strategy obtained throughput is very close to the expected throughput in scenarios where perturbations are not considered. In

scenarios where perturbations are taken into account, we have shown that the system is capable of reacting to the throughput dropping by means of using a frequency scaling module proposed in the architecture.

As perspectives we plan to investigate the behavior of the system for different types of regulators i.e non-linear regulators. We also aim to scale the system by adding more NPUs and analyze the efficiency of the proposed approach for applications with different processing requirements.

REFERENCES

[1] G. Marchesan Almeida, G. Sassatelli, P. Benoit, N. Saint-Jean, S. Varyani, L. Torres, and M. Robert. An adaptive message passing mpsoc framework. *International Journal of Reconfigurable Computing*, Volume October, 2009.

[2] G. M. Almeida, S. Varyani, R. Busseuil, G. Sassatelli, L. Torres, E. A. Carara, and F. G. Moraes. Evaluating the impact of task migration in multi-processor systems-on-chip (full paper). In *23rd Symposium on Integrated Circuits and Systems Design (SBCCI'2010)*, Sao Paulo, Brazil, September 2010.

[3] S. Bertozzi, A. Acquaviva, D. Bertozzi, and A. Poggiali. Supporting task migration in multi-processor systems-on-chip: A feasibility study. In *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, volume 1, pages 1–6, 2006.

[4] A. Acquaviva, A. Alimonda, S. Carta, and M. Pittau. Assessing task migration impact on embedded soft real-time streaming multimedia applications. *EURASIP J. Embedded Syst.*, 2008:1–15, 2008.

[5] M. Pittau, A. Alimonda, S. Carta, and A. Acquaviva. Impact of task migration on streaming multimedia for embedded multiprocessors: A quantitative evaluation. In Samarjit Chakraborty and Petru Eles, editors, *ESTImedia*, pages 59–64. IEEE, 2007.

[6] Alain Greiner. Tsar: a scalable, shared memory, many-cores architecture with global cache coherence. In *9th International Forum on Embedded MPSoC and Multicore (MPSoC'09)*, Savannah, Georgia, USA, 2009. IEEE Press.

[7] Mohammad Ghasemazar, Ehsan Pakbaznia, and Massoud Pedram. Minimizing the power consumption of a chip multiprocessor under an average throughput constraint. In *International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2010.

[8] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost. Hermes: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38(1):69–93, 2004.

[9] Fabien Clermidy, Romain Lemaire, Xavier Popon, Dimitri Ktenas, and Yvain Thonnart. An open and reconfigurable platform for 4g telecommunication: Concepts and application. In *DSD '09: Proceedings of the 2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, pages 449–456, Washington, DC, USA, 2009. IEEE Computer Society.

[10] Diego Puschini, Fabien Clermidy, Pascal Benoit, Gilles Sassatelli, and Lionel Torres. Dynamic and Distributed Frequency Assignment for Energy and Latency Constrained MP-SoC. In *DATE'09: Design Automation and Test in Europe*, pages 1564–1567, Nice, France, 04 2009.

[11] X. Lin, P. K. McKinley, and L. M. Ni. Deadlock-free multicast wormhole routing in 2-d mesh multicomputers. *IEEE Trans. Parallel Distrib. Syst.*, 5(8):793–804, 1994.

[12] MIPS Corp. Mips technologies (http://www.mips.com).

[13] S. Rhoads. Plasma - most mips i(tm) (http://www.opencores.org/project,plasma).

[14] Stuart Bennett. *A History of Control Engineering, 1800-1930*. Institution of Electrical Engineers, Stevenage, UK, UK, 1979.

[15] S. Takahashi and al. Method of designing control system based on a partial knowledge of control object. *Transactions of the Society of Instrument ans Control Engineers*, 5(4), 1979.