An Introduction to Multi-Core System on Chip - Trends and Challenges

Lionel Torres, Pascal Benoit, Gilles Sassatelli, Michel Robert, Fabien Clermidy, Diego Puschini

Chapter 1
An Introduction to Multi-Core System on Chip – Trends and Challenges

Lionel Torres, Pascal Benoit, Gilles Sassatelli, Michel Robert, Fabien Clermidy and Diego Puschini

1.1 From SoC to MPSoC

The empirical law of Moore does not only describe the increasing density of transistors permitted by technological advances. It also imposes new requirements and challenges. Systems complexity increases at the same speed. Nowadays systems could never be designed using the same approaches applied 20 years ago. New architectures are and must be continuously conceived. It is clear now that Moore's law for the last two decades has enabled three main revolutions. The first revolution in the mid-eighties was the way to embed more and more electronic devices in the same silicon die; it was the era of System On Chip. One main challenge was the way to interconnect all these devices efficiently. For this purpose, the Bus interconnect structure was used for long time. Anyway, in the mid-nineties the industrial and academic communities faced a new challenge when the number of processing cores became two numerous for sharing a single communication medieum. A new interconnection scheme based on the Network Telecom Fabrics, the Network On Chip was born; over the past decade intense research efforts have led to significant improvements. The last breakthrough was due to the need to interconnect a set of processors on the same chip, in early 2000. When previously developed systems embedded a single processor, the master of the chip, multiple masters must now share the overall control. The first Multiprocessors System-on-Chip (MPSoCs) emerged [1]. They combine several embedded processors, memories and specialized circuitry (accelerators, I/Os) interconnected through a dedicated infrastructure to provide a complete integrated system. Contrary to SoCs, MPSoCs include two or more master processors managing the application process, achieving higher performances. Since then, an important number of research and commercial designs have been developed [2]. They have started to get into the marketplace and are expected to be widely available in even greater variety in the next few years [3]. It is now

L. Torres (✉)
University of Montpellier 2, UMR CNRS, France
e-mail: Lionel.Torres@lirmm.fr

M. Hu¨bner and J. Becker (eds.), *Multiprocessor System-on-Chip: Hardware Design and Tool Integration*, DOI 10.1007/978-1-4419-6460-1_1

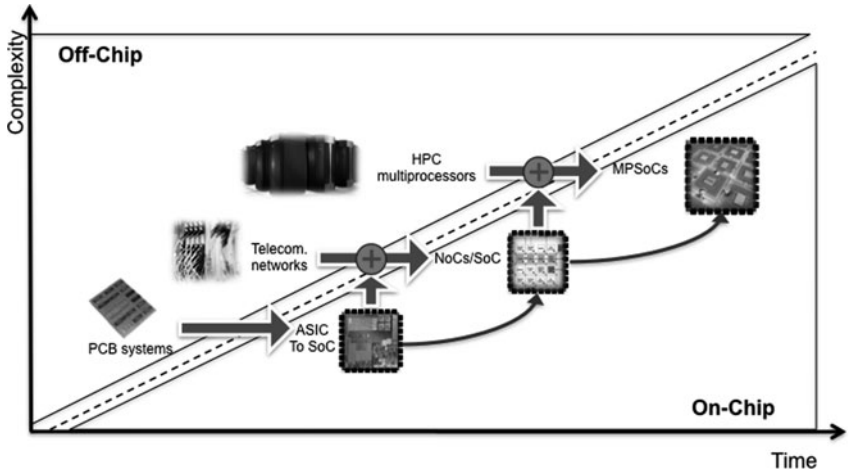

Fig. 1.1 From SoC to MPSoC

clear that this third revolution will change drastically the way to consider System On Chip Architecture. Figure 1.1 summarizes these 3 revolutions that occured in less than 20 years.

1.2 General Structure of MPSoC

This section describes a generic MPSoC, only introducing the key elements in order to formulate valid assumptions on the architecture. In general MPSoC is composed of several Processing Elements (PE) linked by an interconnection structure as it is presented in Fig. 1.2.

1.2.1 Processing Elements

The PEs of an MPSoC are related to the application context and requirements. We distinguish two families of architectures. From one side, heterogeneous MPSoCs are composed of different PEs (processors, memories, accelerators and peripherals). These platforms were certainly pioneered: the C-5 Network Processor [4], Nexperia [5] and OMAP [6], as shown in [2]. The second family represents homogeneous MPSoCs, pioneered by the Lucent Daytona architecture [2, 7], where the same tile is instantiated several times. This chapter targets both families and Fig. 1.2 represents either a homogeneous or heterogeneous design. For instance numerous works consider that processors as well as flexible hardware such as reconfigurable fabrics compose heterogeneous PEs.

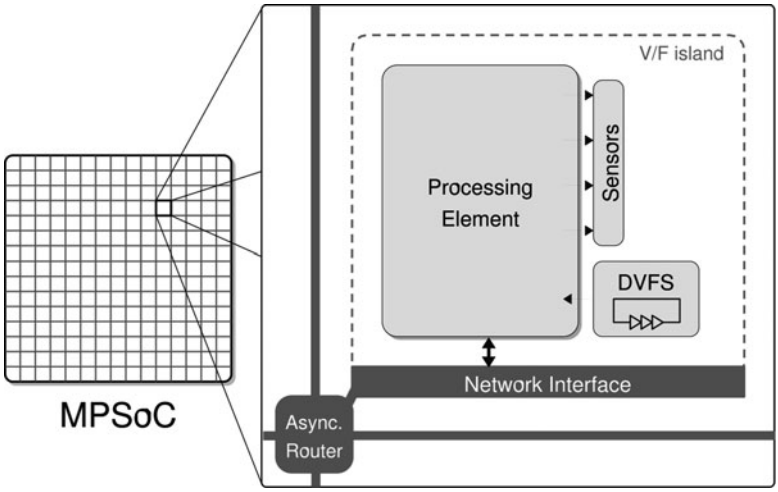

Fig. 1.2 General MPSoC architecture

1.2.2 Interconnection

The PEs previously described are mostly interconnected by a Network-on-Chip (NoC) [8–11]. A NoC is composed of Network Interfaces (NI), routing nodes and links. The NI implements the interface between the interconnection environment and the PE domain. It decouples computation from communication functions. Routing Nodes, also called routers, are in charge of routing are arbitrating the data between the source and destination PEs through the links. Several network topologies have been studied [12, 13]. Figure 1.2 represents a 2D mesh interconnect. The sizing of the offered communication throughput must be enough for the targeted application set.

The NoCs facilitate the design of Globally Asynchronous Locally Synchronous (GALS) property by implementing asynchronous-synchronous interfaces in the NIs [14, 15]. In Fig. 1.2, an example of an asynchronous router is presented to highlight this property.

1.2.3 Power Management

One of the major challenges nowadays is the way to achieve energy efficiency for embedded systems. The GALS feature allows partitioning the MPSoC into several voltage/frequency islands (VFI) [16]. In this example, each VFI contains a PE clocked at a given frequency and voltage. This approach allows fine-grain power management [17]. As in [18, 19], the considered MPSoC incorporates distributed Dynamic Voltage and Frequency Scaling (DVFS): each PE includes a DVFS device. The power optimization consists in adapting the voltage and frequency of

each PE in order to balance power consumption and performance. In more advanced MPSoCs, a set of sensors integrated within each PE provides information about consumption, temperature, performance or any other metric needed to manage the DVFS. Anyway, due to the cost of adding dedicated circuitry, coarser grain power management including multiple PEs in one VFI are used in many MPSoCs, providing a different level of control for the power management.

1.3 Power Efficiency and Adaptability

As presented in the introduction, MPSoCs are following Moore's law [20]. This empirical law has demonstrated to be true during several decades. Figure 1.3 shows some examples of processor with their transistor counts. But for MPSoCs, what are the challenges coming with Moore's law? More transistor density also means more performance (but also increased power consumption) thanks to a multiplication of the number of cores. But it also means more power consumption. During recent years power optimization has become one of the hottest design topics not only for battery-powered devices but also for large variety of application domains such as household electronic to high performance computing. The ITRS [21] predicts an increase by a factor of 2 for the next five years in the power consumption of stationary consumer devices (see Fig. 1.4). Moreover, it is predicted that leakage and dynamic power consumption will be equivalent for such devices for both logic and memory parts. These trends, combined with the increasing performance demand, turn the problem into a real challenge for MPSoC architects [5, 4]. How can we manage the power/performance trade-off on multi-million transistor designs? It is

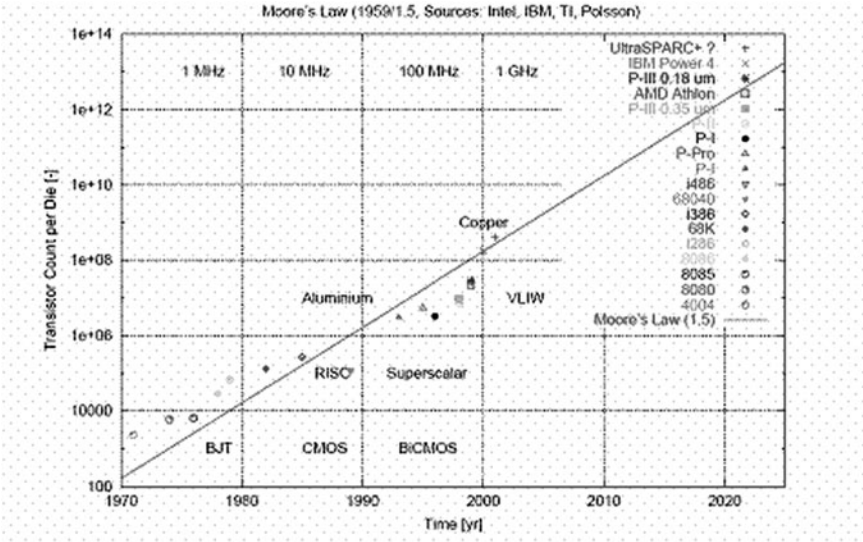

Fig. 1.3 CPU transistors count (http://www.ausairpower.net/moore-iw.pdf)

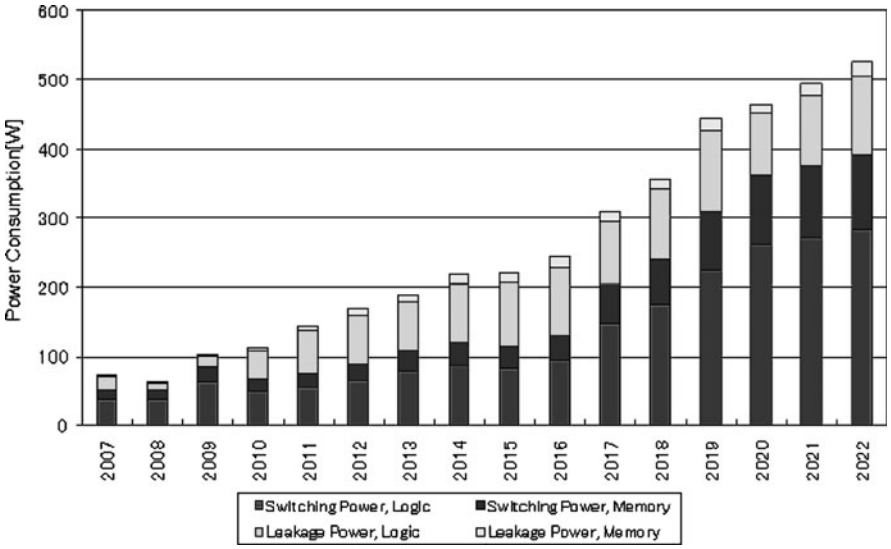

Fig. 1.4 SoC consumer stationary power consumption trends [21]

admitted that advanced energy management is mandatory to achieve efficiency, not only for mobile devices but also for all kind of electronic equipments.

While MPSoC should be designed to be power efficient, the operating environment can no more be considered as static. Let's take a simple example to understand the concept considering fourth generation of telecommunication applications. Computation-intensive complex channel estimation algorithms are needed to sustain a high throughput with bad quality transmission channels. Anyway, when the mobile terminal goes near to a base station a simpler scheme can be used to save energy. How can we manage these modifications in the environment?

A second example of the environmental conditions considers technological variability. Moore's law predicting more and more transistors with improved performance also carries variability problems. Variability is a phenomenon, which always existed in the manufacturing process of CMOS transistors and has been historically taken into account with design margins using statistics of discrepancy between chips of the wafer. However, as transistor size shrinks this phenomenon increases, coping with variability has become a real challenge: the dispersion of parameters within the same chip has now an unquestionable impact on system operation. MPSoCs are affected by this phenomenon. For example, not all PEs of the same system are able to run at the same clock frequency. As a consequence, two specimens of the same MPSoC often achieve unequal performance levels. Hence, how can designers guarantee the performance management under such variations in the manufacturing process?

In order to improve power efficiency in dynamic environments under variability, the answer can be self-adaptability. In other words, the solution can be a system able to adjust itself according to changes in its environment or in parts of the system itself in order to fulfill the requirements.

1.4 Complexity and Scalability

As stated in the introduction, the advances predicted by Moore's law have also accelerated the complexity by multiplying the number of processing elements. To illustrate this increasing complexity, Fig. 1.5 shows the trends predicted by the ITRS [21]: the number of processing cores in SoC consumer portable equipments will increase by a factor of about 3.5 times in the next five years. Moreover, the memory size and logic size will follow the same trends. In this context, how will we manage the more than six hundred processors predicted in 10 years?

There is an underlying problem to the complexity wall: the scalability. Scalability is a property of a system, which indicates its ability to be scaled uo to larger realizations. For MPSoCs, it refers to the capability of a system to increase the total computational power when resources are added. A system, whose performance improves after adding hardware, proportionally to the capacity added, is said to be a scalable system. An algorithm, design, networking protocol, program, or other system is said to scale if it is suitably efficient and practical when applied to large situations.

The good solution for today is probably not the good solution for tomorrow: platform based design and core reuse have driven industrial system designers for obvious productivity and performance reasons. Thes design techniques are increasigly questioned and may not scale any further. One major drawback is that these solutions are poorly scalable in terms of software and hardware. We strongly believe that an alternative is possible from a basis of a scalable hardware and software framework. For this, the distribution of the management functions of an MPSoC is crucial.

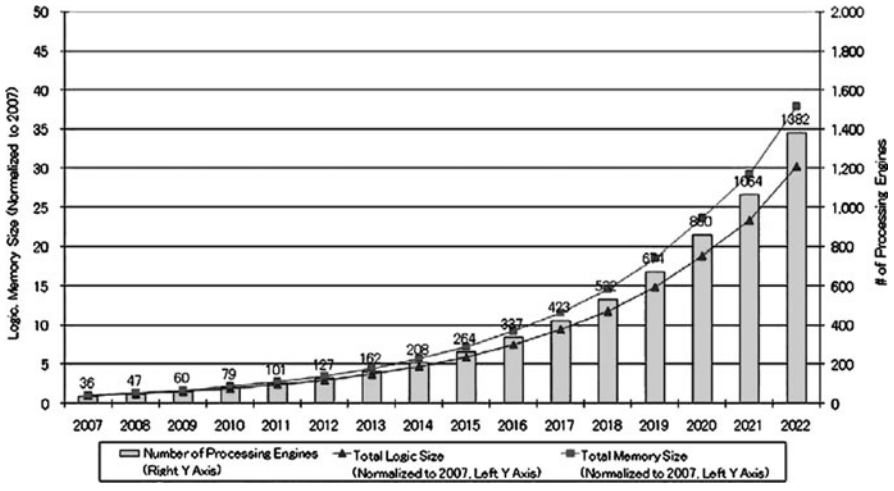

Fig. 1.5 SoC consumer portable design complexity trends [21]

1.5 Heterogeneous and Homogeneous Approaches

In the context of scalability requirement associated with self-adaptability need, MPSoCs are becoming an increasingly popular solution that combines flexibility of software along with potentially significant speedups. As stated in the introduction section, we will make a difference between:

- *Heterogeneous MPSoC*, also referred to Chip Multi-Processing or Multi (Many) Core Systems: these systems are composed of PEs of different types, such as one or several general purpose processors, Digital Signal Processors (DSPs), hardware accelerators, peripherals and an interconnection infrastructure like a NoC.
- *Homogeneous MPSoC*, in this approach, the basic PE embeds all the elements required for a SoC: one or several processors (general purpose or dedicated), memory and peripherals. This tile is then instantiated several times, and all these instances are interconnected through a dedicated communication infrastructure.

Basically, the first approach offers the best performance on power consumption trade-off and the second one is obviously more flexible and scalable but less power efficient. Due to their good power efficicency, heterogeneous MPSoC approaches are used for portable systems, and more generally embedded systems, while homogeneous approaches are commonly used for video game consoles, desktop computers, servers and supercomputing.

1.5.1 Heterogeneous MPSoC

A heterogeneous MPSoC is a set of interconnected cores with different functionalities. The Fig. 1.6 provides an overview of a generic heterogeneous MPSoC, composed of a set of general-purpose processor (CPU), several accelerators (video, audio, etc.), memory elements, peripherals and an interconnection infrastructure.

Beyond its hardware architecture, an MPSoC system is generally running a set of software applications divided into tasks and an operating system devoted to manage

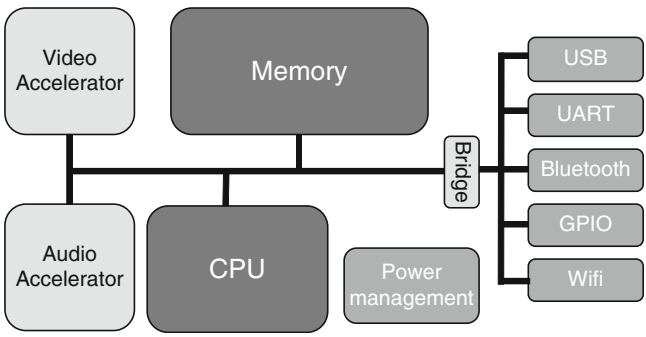

Fig. 1.6 Simplified overview of a heterogeneous MPSoC

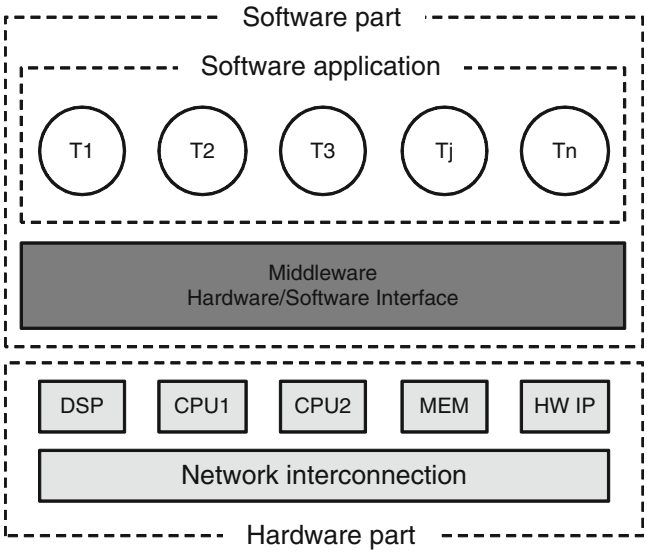

Fig. 1.7 MPSoC abstract view

both hardware and software through a middleware layer (*e.g.* drivers). Figure 1.7 illustrates an abstract view of an MPSoC, and the interaction between software and hardware.

In order to illustrate the general principles presented in the previous section, we can cite The Philips Nexperia, or ST Nomadik or the well-known TI OMAP Platform, or the MORPHEUS MPSoC [22] from the MORPHEUS European project. The functional and structural heterogeneity of these platforms permits obtaining good performance and energy efficiency, allowing them to be integrated in portable devices such as mobile phones.

The term "platform" also confers some flexibility to this approach. Indeed, it is possible with the same platform to customize the system for some specific applications thanks to a basic processor-memory-bus infrastructure, and a library of optional accelerators and peripherals. This approach allows reducing NRE costs and the Time-to-Market, but also presents some drawbacks. The flexibility is limited to the design phase or to some minor extent after fabrication since dedicated accelerators functionalities cannot be reconfigured. The scalability is also a problem of such platforms, since required communication bandwidth depends on the number and types of accelerators and thus can require some adaptation for each design.

1.5.2 Homogeneous MPSoC

As discussed in the previous section, heterogeneous MPSoC systems provide today the best performances/power efficiency tradeoffs and are natural choice for embedded systems, but they also suffer from limited flexibility and scalability.

An alternative lies in building a homogeneous system based on the same programmable building block instantiated several times. This architectural model is often referred in the literature to as parallel architecture model. Parallel architectures were particularly studied in Computer Science and Computer Engineering during the past 40 years. There is nowadays a growing interest for such approaches in embedded systems. The basic principle of an architecture that exhibits parallel processing capabilities relies on increasing the number of physical resources in order to divide the execution time of each resource. Theoretically, an architecture made of N processing resources may provide a speedup of at most N; however this speedup is difficult (or impossible) to obtain in practice. Another benefit of using multiple processing elements versus a single one is that this allows decreasing the frequency correspondingly; and therefore the power supply voltage: as the consumed power is bound to voltage to the power of 2, this decreases the dynamic power consumption significantly. The dynamic power consumption is: $P_{dyn} = \alpha.C_{LOAD}.VDD^2.F_{CLK}$ with P_{dyn} the dynamic power consumption, α is the activity factor, i.e., the fraction of the circuit that is switching, C_{load} the circuit equivalent capacitance, VDD the supply voltage, and f_{clock} the clock frequency. Assuming that it is possible to reduce the clock frequency by a factor r (with $0 < r < 1$), it is then possible to reduce the supply factor by the same factor thanks to DVFS techniques (Dynamic Voltage and Frequency Scaling). Finally, the dynamic power consumption is: $P_{dyn} = \alpha.C_{LOAD}.(r.VDD)^2.(r.F_{CLK})$; with $r = 0{,}8$, the dynamic power is almost divided by 2.

A homogeneous MPSoC based on programmable parallel processors could provide performance thanks to the "speed-up" and a reduced power-consumption by decreasing the operating frequency and the power supply and could be considered as a real alternative to heterogeneous MPSoC. Moreover, their inherent structure is more flexible and more scalable than heterogeneous systems. Practically, exploiting efficiently the parallelism is not straightforward; flexibility and scalability could also be limited due to several factors such as the organization of the memory, the interconnection infrastructure, etc.

Parallel architectures have been studied intensively during the past 40 years; there is consequently a huge amount of books and references related to this topic and we will therefore only focus on general concepts.

The first famous classification was proposed by Flynn [23]. He classifies architectures according to the relationship between processing units and control units. He defines four execution models: SISD (Single Instruction Single Data), SIMD (Single Instruction Multiple Data), MISD (Multiple Instruction Single Data) and MIMD (Multiple Instruction Multiple Data). The SISD model is the classical Von Neumann model [24], where a single processing resource executing a single instruction per unit time processes a single data flow. In SIMD architecture, a single control unit shares the data flows and distributes data to each processing resource. The MISD architectures execute several instructions simultaneously on a single data flow. Finally, several control units manage several processing units in the MIMD architectures.

In Fig. 1.8, Flynn's classification has been extended to take into account the organization of the memory that can be shared (8.a) or distributed (8.b). In shared memory architecture, processes (executed by different processors) can easily exchange information through shared variables; however it requires handling

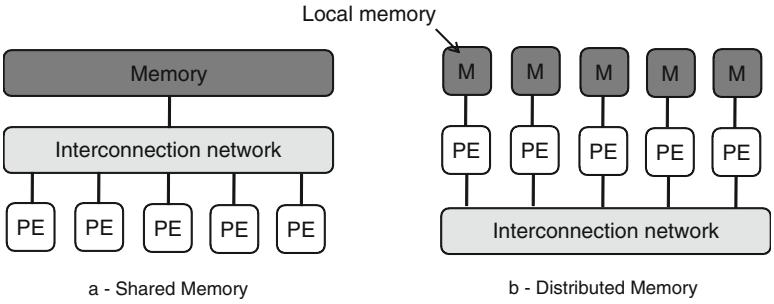

Fig. 1.8 Memory organization

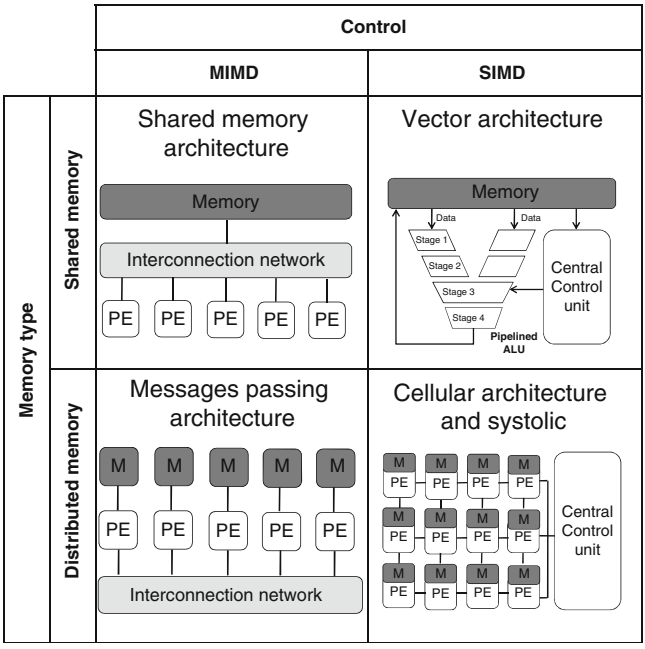

Fig. 1.9 Architecture taxonomy

carefully synchronization and memory protection. In distributed memory architecture, a communication infrastructure is required in order to connect processing elements and their memories and allow exchanging information.

Based on the memory and the control organization, the Fig. 1.9 depicts an architecture classification of parallel homogeneous processing architectures. It distinguishes the centralized control (SIMD) and decentralized control (MIMD), shared and distributed memories.

It is important to observe here that the organization of the control and the memory will provide different trade-offs in terms of scalability and management of the system. For instance, an architecture based on a fully distributed control and

memory organization, will be more scalable but less flexible to manage, than architecture based on a centralized control and a shared memory.

1.6 Multi variable Optimization

Due to the increasing complexity of MPSoC architectures, optimization is a real challenge since it may target multiple opposite objectives: application performance, power consumption/energy, temperature, load balacing, etc. In the literature, there are several methods developed to address this problem. Classical approaches are static and try to optimize the system at design time. More recent techniques are employed at run-time and try to adapt the system dynamically. Most advanced methods aim at taking advantage of the distributed decision capabilities of the processing elements in order to improve the scalability of the system.

1.6.1 Static Optimization

In the context of MPSoC, a static optimization approach is a way to improve the system at design time. Several authors have proposed static optimization techniques to improve the power efficiency. For example, in [25] authors use genetic algorithms to solve the optimization problem at design time. They explore metrics including communication traffic, memory occupation and throughput aspects.

In [26], authors analyze three static optimization methods: greedy algorithm, tabu search and simulated annealing. The problem of how to find a task schedule with the minimum power consumption while satisfying some timing constraints is studied as a part of the design space exploration process. Firstly, the system description is decomposed into Synchronous Data Flow (SDF) graphs [27] in a single frequency domain including timing constraints. Then, an extension of traditional SDFs to multi-frequency domain graphs is proposed.

In [28], a static policy based on linear models is proposed to optimize the power consumption while guaranteeing real-time constraints. The problem of selecting the best operating frequency for each block of a distributed design is studied. The author models a set of frame-based pipelined applications by using SDF graphs. Then, applications are mapped on a distributed platform integrating fine-grain DVFS.

1.6.2 Dynamic Optimization

Static explorations are always necessary to make design-time decisions. Nevertheless, considering the increasing uncertainty of implementation technologies and

applicative scenarios of such systems, dynamic optimizations are becoming mandatory to provide flexible approaches and reliable designs [29]. Centralized and distributed approaches are reported in the following subsections.

1.6.2.1 Centralized Approaches

Contrarily to static optimization, dynamic approaches offer adaptability. Figure 1.10 shows a schematic view representing the common dynamic approaches existing for MPSoC: a centralized optimization subsystem is in charge of the whole system management. It analyzes global information and optimizes each processing element in the system.

In [30], the frequency and voltage selection for GALS systems based on VFIs is addressed. A centralized method based on non-linear Lagrange optimization is used to select the frequencies and voltages. They present static and dynamic algorithms. Moreover, authors affirm that ideally in latency-constrained systems, the assignment of optimal voltages would need a global strategy decision.

Similarly, in [31, 32], authors propose a centralized energy management using models inspired by Kirchhoff's current law. They argue that while local energy dissipation of each PE can be minimized using DVS techniques based on workload predictions, it can be shown that these local minimums usually do not represent the

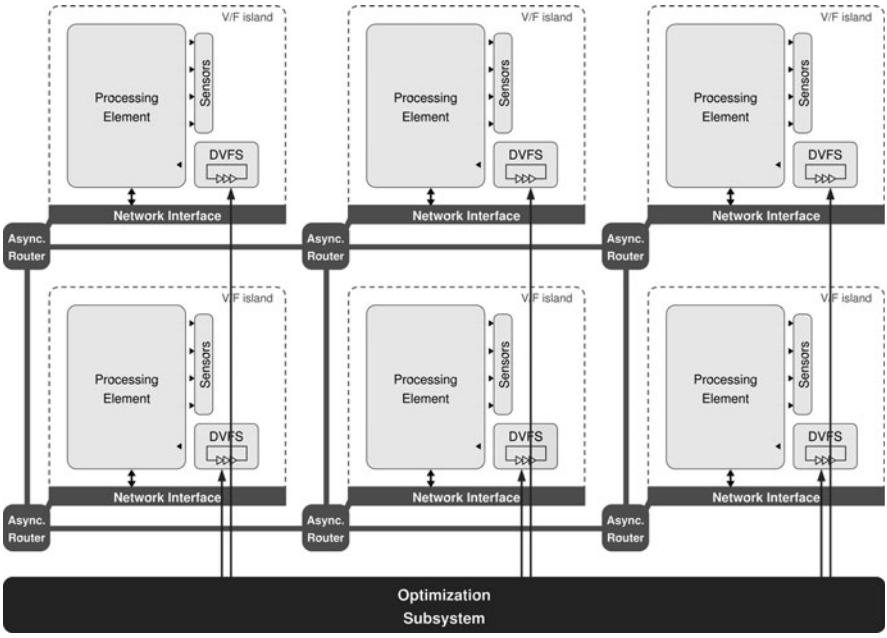

Fig. 1.10 Centralized dynamic optimization on MPSoC

global optimum. Moreover, the global optimum can be reached by considering the relative timing dependencies of all tasks running in the system. Their approach is based on an online global energy management unit that controls the PEs through a power source and a clock generator. The block diagram of such approach is represented in Fig. 1.11. Authors exploit the analogy between the energy minimization problem under timing constraints in a general task graph and the power minimization problem under Kirchhoff's current law constraints in an equivalent resistive network.

In [33], authors use convex optimization for temperature-aware frequency assignment on MPSoC. Firstly, they present a complex temperature model. Then, the problem is formulated for both, steady-state and dynamic-state: assign a single frequency to each processor, maintaining the temperature and power consumption below user-defined thresholds. In steady state, frequency and voltage are assigned once and remained constant, without taking advantage of DVFS. In dynamic state, the frequencies and voltages are varied over time to better optimize the system performance.

Authors formulate both scenarios as convex optimization problems. Then, they propose the steady state and dynamic-state optimization procedures. For the dynamic case, a 2-phase algorithm is used. Nevertheless, authors only present the mathematic formulation. Using a Matlab convex-optimization solver solves the demonstrative scenario shown in this work. The same authors propose in [34] to pre-calculate some valid solution at design time by using the convex-optimization method, and to implement a control to choose at run-time the best solution for each case. Figure 1.12 shows the flow for the design-time table construction.

In [35], authors survey some studies for energy-efficient scheduling in real-time systems on platforms integrating DVFS. After a long review of techniques applied to single-processor systems, the article divides multiprocessor platform into homogeneous and heterogeneous ones. For the first one, it briefly describes some techniques applied to frame-based real-time task scheduling, periodic real-time

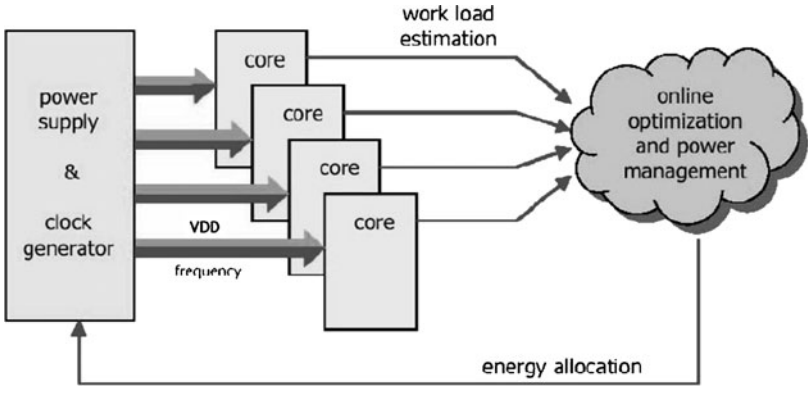

Fig. 1.11 Centralized online global energy management [32]

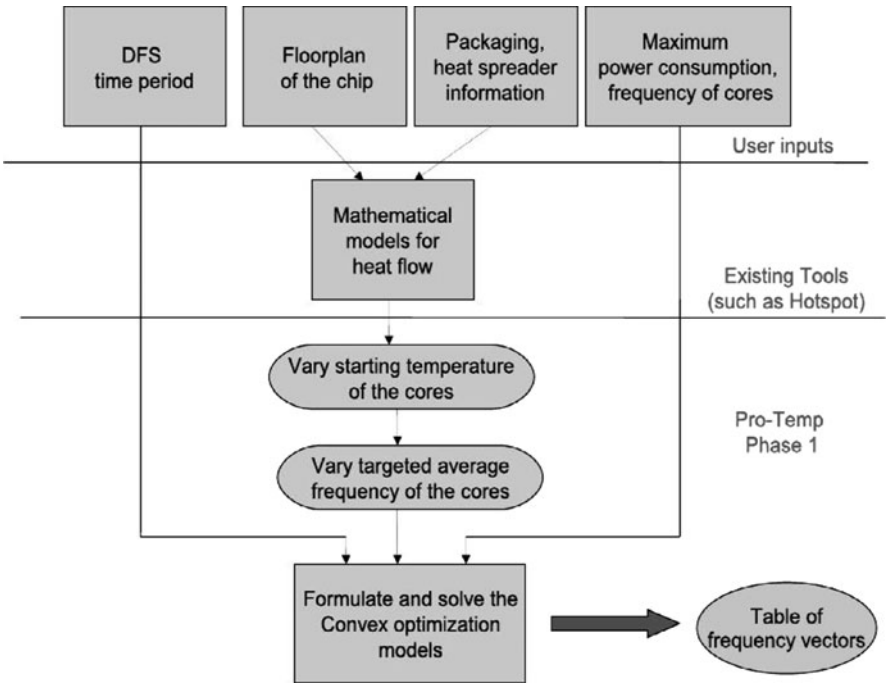

Fig. 1.12 Design-time table construction by convex optimization [34]

scheduling, leakage-aware energy-efficient scheduling and slack reclamation scheduling. For heterogeneous systems, it presents some techniques for periodic real-time tasks and allocation cost minimization under energy constraints.

In [36], heuristics for optimal task mapping are discussed in a NoC-based heterogeneous MPSoC. There are several approaches where tasks are moved in order to balance computation workload and homogeneously dissipate the power, for example in [37]. In that sense, in [38], hot-spot avoidance and thermal gradient control become an important optimization problem regarding the reliability of MPSoC.

In [38], authors investigate dynamic OS-based schedulers for thermal management of MPSoCs. In [39], thermal gradients are also minimized. They focus on MPSoCs where workload is not known a priori and generally not easy to predict. They propose an OS-based task migration and scheduling policy that optimizes the thermal profile of the chip by balancing the system load. Authors claim to obtain significant reductions in temporal and spatial temperature variations.

In [40, 41], authors propose a design time Pareto exploration and characterization combined with run-time management. They firstly make a multi-dimensional design-time exploration. The space includes costs (e.g. energy consumption), constraints (e.g. performance) and used platform resources (e.g. memory usage, processors, clocks, communication bandwidth). A low-complexity run-time manager implemented in the OS takes critical decisions during a second phase.

In [42], the interest is how to select at run-time an energy-efficient mapping on heterogeneous multi-processor platforms. Considering that many different possible implementations per application can be available and the selection must meet the application deadlines under the available platform resources, authors model as a NP-hard problem: the Multi-dimension Multi-choice Knapsack Problem. In order to find a near-optimal solution, they propose a heuristic-based OS implementation.

1.6.2.2 Distributed Approaches

With forecasted hundreds of processing elements (PE), scalability is also a major concern for the optimization process. For this reason, an alternative approach is to handle optimization dynamically in a distributed way. Static optimization does not provide adatability at run-time. On the contrary, existing dynamic approaches provide reactivity during execution time but they are centralized solutions. They do not provide scalability since they are not based on distributed models

An alternative solution to centralized approaches is to consider distributed algorithms. One interesting approach is to conceive the architecture illustrated in Fig. 1.13: each processing element of an MPSoC embeds an optimization subsystem based on a distributed algorithm. This subsystem manages the local actuators (DVFS in the figure) taking into account the operating conditions. In other words, the goal is to conceive a distributed and dynamic optimization algorithm.

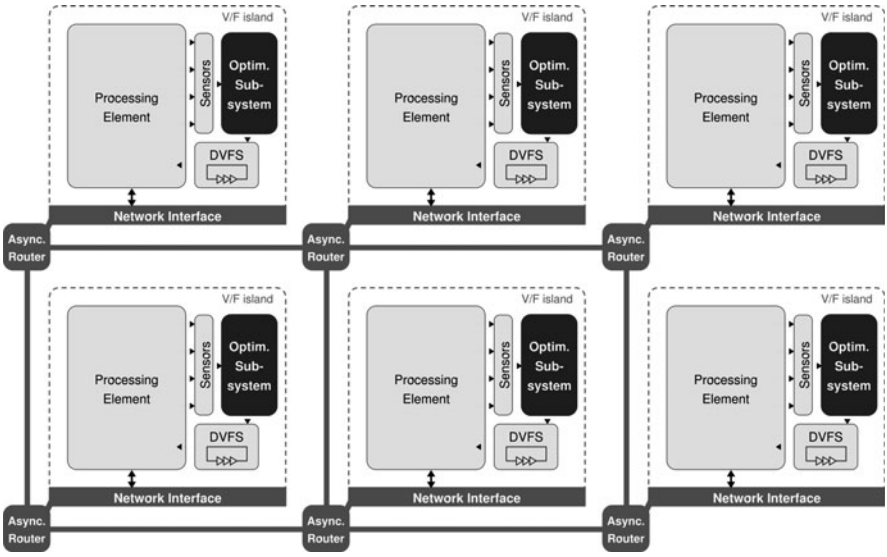

Fig. 1.13 Distributed dynamic optimization on MPSoC

To avoid hotspots and control the temperature of the tiles, dynamic voltage-frequency scaling (DVFS) can be applied at PE level. At system level, it implies to dynamically manage the different voltage-frequency couples of each PE in order to obtain a global optimization. In [43], an original approach based on game theory is presented, which adjusts at run-time the frequency of each PE. It aims at reducing the tile temperature while maintaining the synchronization between the tasks of the application graph. A fully distributed scheme is assumed in order to build a scalable mechanism. Results show that the proposed run-time algorithm find solutions in few calculation cycles achieving temperature reductions of about 23%. In [44], results show that the proposed run-time algorithm requires an average of 20 calculation cycles to find the solution for a 100-processor platform and reaches equivalent performances when comparing with an offline method.

In [45], this adaptive technique is applied to reduce power consumption. It optimizes the frequencies of local processors while fulfilling applicative real-time constraints. The obtained power consumption gains on a telecommunication test case are between 10% and 25%, while the reaction time to temporal variations due to application reconfiguration is less than 25µs.

1.7 Static vs Dynamic Centralized and Distributed Approaches

Table 1.1 summarizes the optimization methods described in the previous section. Several approaches have been considered, representing the directions of MPSoC optimizations. This table allows a qualitative comparison. The methods are compared regarding the off-line and dynamic phases, their complexities, and their implementations (centralized or distributed). Approaches presented in [34, 33] and [30] have complex off-line optimization phases. In [40, 41, 42], most of the work is done at design time. These approaches can be hardly used at run-time due to their high complexity.

The solution proposed in [30, 32] operates with a dynamic optimization subsystem with a low complexity. In the same direction, approaches [38, 39] and [40, 41, 42] provide run-time management. Nevertheless, all these approaches fail when distributed aspects are considered. When OS-based implementation is used ([38, 39] and [41, 42, 43]) a distributed implementation can be imagined. However, we do not consider doing that since they are not based on distributed models. The approach based on Game Theory [44, 45] is intrinsically based on a distributed model, which improves the scalability of the system. Furthermore, this low complexity method can be easily implemented at run-time, which implies a good adaptability required by dynamic systems.

Finally, Table 1.1 also compares the metrics used in each case. One can note that not every approach includes constrained scenarios but all of them propose multi-objective optimization. Some of these models can be reused in novel formulations.

Table 1.1 MPSoC optimization synthesis

	EPFL	Stanford & EPFL	Carnegie Melon	SUN & U. California	IMEC	LIRMM & CEA LETI
Reference	[32, 33]	[34, 35]	[31]	[39, 40]	[41, 42, 43]	[43, 44, 45]
Model	Kirchoff's current law analogy	Convex Optimization	Non-linear Lagrange Optimization	Integer Linear Programming	Off-line exploration	Game Theory
Offline phase	Yes	Yes	Yes	No	Yes	No
Complexity	Medium	High	High	–	Very High	–
Dynamic Phase	Yes	–	–	Yes	Yes	Yes
Complexity	Low	–	–	Medium	Medium	Low
Distributed Imp.	No	No	No	Possible	Possible	Yes
Distrib. model	No	No	No	No	No	Yes
Objective metrics	Energy	Performance	Energy, Throughput	Hot-spot, Temp. gradient	Multiple	Mulitple
Constraints	Latency	Temperature	Latency	–	–	Latency, Energy
Actuator	Global power manager	DVFS	Vdd & freq.	OS-based task migration scheduling	OS-based	Local DVFS

1.8 Conclusion

Semiconductors currently undergo profound changes due to several factors such as the approaching limits of silicon CMOS technology as well as the inadequacy of the machine models that have been used until now. These challenges require devising new design approaches and programming of future integrated circuits. Hence, parallelism appears as the only solution for coping with the ever-increasing demand in term of performance. The solutions that are suggested in the literature often rely on the capability of the system to take online decisions for coping with these issues, such as scaling supply voltage and frequency for increasing energy efficiency, or testing the circuit for identifying faulty components and discarding them from the functionality.

MPSoCs are certainly the natural target for bringing these techniques into practice: provided they comply with some design rules they may prove scalable from a performance point of view. Further, since they are in essence distributed architectures they are well suited to locally monitoring and controlling system parameters.

In this chapter, we have studied multiprocessor systems and proposed an overview of a template that we believe is representative of tomorrows' MPSoCs. The important characteristics that have been considered are mostly flexibility, scalability and adaptability, considering decentralized control, Homogeneous or Heterogeneous array of processing elements, Distributed memory, Scalable NoC-style communication network.

Finally, we think that adaptability is an approach that will in the near future be widely adopted in the area. Not only because of the here mentioned limitations such as technology shrinking, power consumption and reliability but also because computing undoubtedly go pervasive. Pervasive, or ambient computing is a research area on its own and in essence implies using architecture that are capable of self-adapting to many time-changing execution scenarios. Be it mobile sensors deployed for monitoring various natural phenomena or computing devices embedded in clothes (wearable computing), such systems have to cope with many limitations such as limited power budget, interoperability, communication issues, and finally, scalability.

GLOSSARY

SoC	System On Chip
MPSoC	Multiprocessor System On Chip
NoC	Network On Chip
HPC	High Performance Computing
ASIC	Application Specific Integrated Circuit
PE	Processing Element
SW	Software

HW	Hardware
OMAP	Open Multimedia Applications Platform
DVFS	Dynamic Voltage and Frequency Scaling
GALS	Globally Asynchronous Locally Synchronous
NI	Network Interfaces
VFI	Voltage/Frequency Islands
ITRS	International Technology Roadmap for Semiconductors
CPU	Central Processing Unit
SISD	Single Instruction Single Data
SIMD	Single Instruction Multiple Data
MISD	Multiple Instruction Single Data
MIMD	Multiple Instruction Multiple Data
OS	Operating System

References

1. Ahmed Jerraya and Wayne Wolf. Multiprocessor Systems-on-Chips. Elsevier Inc, 2004.
2. Wayne Wolf, Ahmed Jerraya, and Grant Martin. Multiprocessor system-on-chip (MPSoC) technology. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 27(10):1701–1713, Oct. 2008.
3. Wayne Wolf. The future of multiprocessor systems-on-chips. In DAC '04: Proceedings of the 41st annual Design Automation Conference, pages 681–685, New York, NY, USA, 2004. ACM.
4. Freescale Semiconductor, Inc. C-5 Network Processor Architecture Guide, 2001. Ref. manual C5NPD0-AG http://www.freescale.com.
5. S. Dutta, R. Jensen, and A. Rieckmann. Viper: A multiprocessor SOC for advanced set-top box and digital TV systems. Design & Test of Computers, IEEE, 18(5):21– 31, Sep-Oct 2001.
6. Texas Instruments Inc. OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide, 2006. Tech. article SPRU748C. http://www.ti.com.
7. B. Ackland, A. Anesko, D. Brinthaupt, S.J. Daubert, A. Kalavade, J. Knobloch, E. Micca, M. Moturi, C.J. Nicol, J.H. O'Neill, J. Othmer, E. Sackinger, K.J. Singh, J. Sweet, C.J. Terman, and J. Williams. A single-chip, 1.6-billion, 16-b MAC/s multiprocessor DSP. Solid-State Circuits, IEEE Journal of, 35(3):412–424, Mar 2000.
8. P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In DATE '00: Proceedings of the 2000 Design, Automation and Test in Europe Conference and Exhibition, pages 250–256, 2000.
9. William J. Dally and Brian Towles. Route packets, not wires: on-chip inteconnection networks. In DAC '01: Proceedings of the 38th Design Automation Conference, pages 684–689, New York, NY, USA, 2001. ACM.
10. L. Benini and G. De Micheli. Networks on chips: a new SoC paradigm. IEEE Computer, 35 (1):70–78, Jan 2002. [cited at p. 3]
11. Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of Network-on-chip. ACM Comput. Surv., 38(1):1, 2006.
12. Partha Pratim Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Perfor- mance evaluation and design trade-offs for network-on-chip interconnect architec- tures. Computers, IEEE Transactions on, 54(8):1025–1040, Aug. 2005.

13. D. Bertozzi and L. Benini. Xpipes: a network-on-chip architecture for gigas- cale systems-on-chip. Circuits and Systems Magazine, IEEE, 4(2):18–31, 2004.
14. E. Beigńe, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin. Asynchronous NOC Architecture Providing Low Latency Service and Its Multi-Level Design Framework. In ASYNC '05: Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems, pages 54–63, Washington, DC, USA, 2005. IEEE Computer Society.
15. J. Pontes, M. Moreira, R. Soares, and N. Calazans. Hermes-glp: A gals network on chip router with power control techniques. In Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual, pages 347–352, April 2008.
16. Umit Y. Ogras, Radu Marculescu, Puru Choudhary, and Diana Marculescu. Voltage- frequency island partitioning for GALS-based Networks-on-Chip. In DAC '07: Proceedings of the 44th Annual Design Automation Conference, pages 110–115, New York, NY, USA, 2007. ACM.
17. James Donald and Margaret Martonosi. Techniques for multicore thermal man- agement: Classification and new exploration. In ISCA '06: Proceeding of the 33rd International Symposium on Computer Architecture, pages 78–88, 2006.
18. Edith Beigńe, Fabien Clermidy, Sylvain Miermont, and Pascal Vivet. Dynamic voltage and frequency scaling architecture for units integration within a gals noc. In NOCS, pages 129–138, 2008.
19. Edith Beigńe, Fabien Clermidy, Sylvain Miermont, Alexandre Valentian, Pascal 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: Proceeding of the 34th European Solid-State Circuits Conference, Edinburg, UK, Sept. 2008.
20. G. E. Moore. Cramming More Components onto Integrated Circuits. Electronics, 38 (8):114–117, April 1965.
21. The International Technology Roadmap for Semiconductors. International Technology Roadmap for Semiconductors 2008 Update Overview. http://www.itrs.net.
22. Davide Rossi, Fabio Campi, Antonello Deledda, Simone Spolzino and Stefano Pucillo, A Heterogeneous Digital Signal Processor Implementation for Dynamically Reconfigurable Computing, IEEE Custom Integrated Circuits Conference (CICC) September 13 - 16 2009,
23. M. Flynn. Some Computer Organizations and Their Effectiveness, IEEE Trans. Computer, vol. 21, pp. 948, 1972
24. A. W. Burks, H. Goldstine, and J. von Neumann. Preliminary Discussion of the Logical Design of an Electronic Computing Instrument, Inst. Advanced Study Rept., vol. 1, June, 1946
25. Issam Maalej, Guy Gogniat, Jean Luc Philippe, and Mohamed Abid. System Level Design Space Exploration for Multiprocessor System on Chip. In ISVLSI '08: Proceedings of the 2008 IEEE Computer Society Annual Symposium on VLSI, pages 93–98, Washington, DC, USA, 2008. IEEE Computer Society.
26. Bastian Knerr, Martin Holzer, and Markus Rupp. Task Scheduling for Power Opti- misation of Multi Frequency synchronous Data Flow Graphs. In SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design, pages 50–55, New York, NY, USA, 2005. ACM.
27. Edward Ashford Lee and David G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. IEEE Trans. Comput., 36(1):24–35, 1987.
28. Philippe Grosse, Yves Durand, Paul Feautrier: Methods for power optimization in SOC-based data flow systems. ACM Trans. Design Autom. Electr. Syst. 14(3): (2009)
29. A. K. Coskun, T. Simunic Rosing, K. Mihic, G. De Micheli, and Y. Leblebici. Analysis and Optimization of MPSoC Reliability. Journal of Low Power Electronics, 2(1):56–69, 2006.
30. Koushik Niyogi and Diana Marculescu. Speed and voltage selection for GALS sys- tems based on voltage/frequency islands. In ASP-DAC '05: Proceedings of the 2005 Conference on Asia South Pacific Design Automation, pages 292–297, New York, NY, USA, 2005. ACM.
31. Zeynep Toprak Deniz, Yusuf Leblebici, and Eric Vittoz. Configurable On-Line Global Energy Optimization in Multi-Core Embedded Systems Using Principles of Analog Computation. In IFIP 2006: International Conference on Very Large Scale Integration, pages 379–384, Oct. 2006.

32. Zeynep Toprak Deniz, Yusuf Leblebici, and Eric Vittoz. On-Line Global Energy Optimization in Multi-Core Systems Using Principles of Analog Computation. In ESSCIRC 2006: Proceedings of the 32nd European Solid-State Circuits Conference, pages 219–222, Sept. 2006.
33. Srinivasan Murali, Almir Mutapcic, David Atienza, Rajesh Gupta, Stephen Boyd, and Giovanni De Micheli. Temperature-aware processor frequency assignment for MPSoCs using convex optimization. In CODES+ISSS '07: Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis, pages 111–116, New York, NY, USA, 2007. ACM.
34. Srinivasan Murali, Almir Mutapcic, David Atienza, Rajesh Gupta, Stephen Boyd, Luca Benini, and Giovanni De Micheli. Temperature control of high-performance multi-core platforms using convex optimization. In DATE'08: Design, Automation and Test in Europe, pages 110–115, Munich, Germany, 2008. IEEE Computer Society.
35. Jian-Jia Chenand Chin-Fu Kuo. Energy-Efficient Schedulingfor Real-Time Systems on Dynamic Voltage Scaling (DVS) Platforms. In RTCSA '07: Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pages 28–38, Washington, DC, USA, 2007. IEEE Computer Society.
36. Ewerson Carvalho, Ney Calazans, and Fernando Moraes. Heuristics for dynamic task mapping in noc-based heterogeneous MPSoCs. In RSP '07: Proceedings of the 18th IEEE/IFIP International Workshop on Rapid System Prototyping, pages 34–40, Washington, DC, USA, 2007. IEEE Computer Society.
37. G. M. Link and N. Vijaykrishnan. Hotspot prevention through runtime reconfiguration in Network-on-Chip. In DATE '05: Proceedings of the 2005 Conference on Design, Automation and Test in Europe, pages 648–649, Washington, DC, USA, 2005. IEEE Computer Society.
38. Ayse Kivilcim Coskun, Tajana Simunic Rosing, and Keith Whisnant. Temperature aware task scheduling in MPSoCs. In DATE '07: Proceedings of the conference on Design, automation and test in Europe, pages 1659–1664, San Jose, CA, USA, 2007. EDA Consortium.
39. Ayse Kivilcim Coskun, Tajana Simunic Rosing, Keith A. Whisnant, and Kenny C. Gross. Temperature-aware mpsoc scheduling for reducing hot spots and gradients. In ASP-DAC '08: Proceedings of the 2008 conference on Asia and South Pacific design automation, pages 49–54, Los Alamitos, CA, USA, 2008. IEEE Computer Society Press.
40. Ch. Ykman-Couvreur, E. Brockmeyer, V. Nollet, Th. Marescaux, Fr. Catthoor, and H. Corporaal. Design-Time Application Exploration for MP-SoC Customized Run- Time Management. In SOC'05: Proceedings of the International Symposium on System-on-Chip, pages 66–73, Tampere, Finland, November 2005.
41. Ch. Ykman-Couvreur, V. Nollet, Fr. Catthoor, and H. Corporaal. Fast Multi-Dimension Multi-Choice Knapsack Heuristic for MP-SoC Run-Time Management. In SOC'06: Proceedings of the International Symposium on System-on-Chip, pages 195–198, Tampere, Finland, November 2006.
42. Ch. Ykman-Couvreur, V. Nollet, Th. Marescaux, E. Brockmeyer, Fr. Catthoor, and H. Corporaal. Pareto-based application specification for MP-SoC Customized Run-Time Management. In SAMOS'06: Proceedings of the International Conference on Embedded Computer Systems: Architectures, MOdeling, and Simulation, pages 78–84, Samos, Greece, July 2006.
43. D. Puschini, F. Clermidy, P. Benoit, G. Sassatelli, and L. Torres. Temperature-Aware Distributed Run-Time Optimization on MP-SoC Using Game Theory, Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual, 2008, pp. 375-380.
44. D. Puschini, F. Clermidy, P. Benoit, and G. Sassatelli. A Game-Theoretic Approach for Run-Time Distributed Optimization on MP-SoC, International Journal of Reconfigurable Computing, vol. 2008, 2008, p. 11.
45. D. Puschini, F. Clermidy, P. Benoit, G. Sassatelli, and L. Torres. Adaptive energy-aware latency-constrained DVFS policy for MPSoC, 2009 IEEE International SOC Conference (SOCC), IEEE, 2009, pp. 89–92.