# Mesochronous NoC Technology for Power-Efficient GALS MPSoCs

Daniele Ludovici[†], Alessandro Strano[†], Georgi N. Gaydadjiev[§], Davide Bertozzi[†]
[†] ENDIF, University of Ferrara, 44100 Ferrara, Italy.
[§] Computer Engineering Lab., Delft University of Technology, The Netherlands.

## Abstract

*MPSoCs are today frequently designed as the composition of multiple voltage/frequency islands, thus calling for a GALS clocking style. In this context, the on-chip interconnection network can be either inferred as a single and independent clock domain or it can be distributed among core's domains. This paper targets the former scenario, since it results in the homogeneous speed of the NoC switching elements. From a physical design viewpoint, the main issues lie however in the chip-wide extension of the network domain and in the growing uncertainties affecting nanoscale silicon technologies. This paper proves that partitioning the network into mesochronous domains and merging synchronizers with NoC building blocks, two main advantages can be achieved. First, it is possible to evolve synchronous networks to mesochronous ones with marginal performance and area overhead. Second, the mesochronous NoC exposes more degrees of freedom for power optimization.*

## 1. Introduction

Networks-on-chip (NoCs) are proving capable of easing the communication bottleneck arising in multi-core computing platforms [11, 12, 17, 18], thus overcoming the fundamental performance, power and physical design limitations of shared and multi-layer busses.

There is today little doubt on the fact that a high-performance and cost-effective NoC can only be designed in 45nm and beyond under a relaxed synchronization assumption [18, 19]. One effective method to address this issue is through the use of globally asynchronous and locally synchronous (GALS) architectures, where the chip is partitioned into multiple independent voltage and frequency domains. Each domain is clocked synchronously while inter-domain communication is achieved through specific interconnect techniques and circuits [16].

The methodology of inter-domain communication is a crucial design point for GALS architectures. One approach currently experimented in GALS NoC prototypes consists of using purely asynchronous clockless handshaking for transferring data words across clock domains [20, 21]. A few chip demonstrators prove the viability of this solution [13–15], but they have not achieved widespread adoption of asynchronous NoCs in the industrial arena yet.

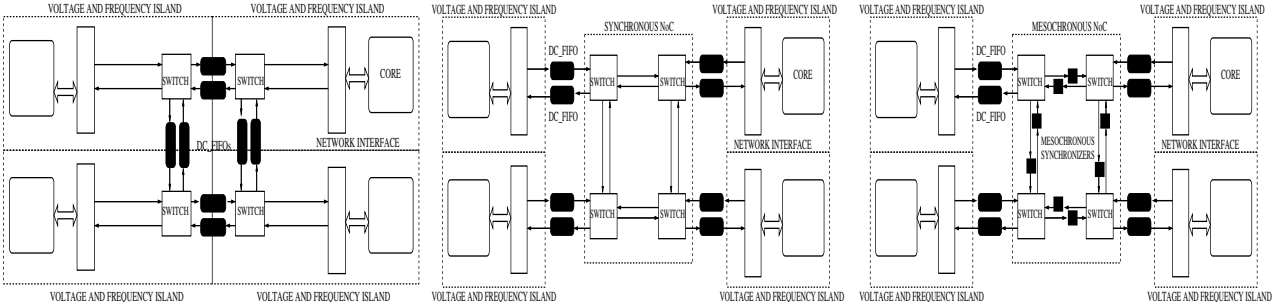
In order to more incrementally evolve current industrial practice, our previous work in [3–6] has developed synchronizer-based GALS NoC technology. In particular, design techniques merging synchronizers with network building blocks (named the *tightly coupled design style*) have proved area, power and performance efficient with respect to loosely coupled solutions, where synchronizers are placed as external blocks to NoC switches. All our previous work concerns architecture design space exploration and quality metrics assessment of synchronizer-based communications at the switch level.

This paper builds on these milestones and moves a step forward by taking the network-level perspective. While the migration from fully synchronous parallel systems to GALS systems with voltage/frequency decoupling between IP cores is taken as a matter of fact in this paper, there are significant GALS NoC architecture variants the designer can still choose from. The first one consists of placing NoC switches in the clock domains of the IP cores they are connected with. In contrast, an alternative solution consists of inferring the on-chip network as an independent clock domain, disjoint from those of the IP cores. In this scenario, dual-clock FIFOs need to be instantiated only at the network boundary, since the network is synchronized by a single and independent clock signal. The homogeneous performance of NoC switches, the fewer amount of dual-clock FIFOs required and the possibility to have an always on system interconnect fabric make this solution more attractive to this paper. However, the feasibility and efficiency of this solution is now mainly on burden of the physical designer. In fact, he has again to deal with a large synchronous clock domain (the NoC itself) distributed throughout the entire chip. A workaround for this problem consists of inferring the network as a set of mesochronous domains, instead of a global synchronous domain, yet retaining a globally synchronous perspective of the network itself. The granularity of a mesochronous domain can be as fine as a NoC switch, which is the case considered in this paper. The communication between neighboring switches is then mesochronous as the top-level clock tree might not be equilibrated. This brings the additional advantage that mesochronous synchronizers are typically more lightweight than dual-clock FIFOs for use in switch-to-switch links.

This paper leverages mature mesochronous communication technology to perform a comprehensive crossbenchmarking of a mesochronous NoC with a fully synchronous NoC for use in a GALS system. Both networks share the same baseline MPSoC-oriented NoC architecture for the sake of fair comparison. The tightly coupled design principle is followed for mesochronous links, so that their unique optimization opportunities in the NoC domain are fully exploited. The paper relies on actual implementations on a 65nm industrial technology library and provides the assessment of a wide range of design quality metrics, some of them of special interest for nanoscale silicon technologies: performance, area, power, and clock tree synthesis efficiency. This way, this paper can provide useful guidelines for those industrial designers currently committed to the development of next generation NoC-based MPSoCs. Concisely, the main contribution of this work can be summarized as the crossbenchmarking of two GALS systems, the former implemented with a fully synchronous NoC; the latter leveraging a mesochronous NoC. Both systems have been compared from an area and power viewpoint. Since dual-clock FIFOs for connection to network interfaces are common to both solutions, they have not been considered in this work.

The remainder of this paper is organized as follows. Section 2 will present the GALS platforms under analysis whereas Section 3 will review the architecture of the synchronizer block utilized as baseline to build the mesochronous network. Section 4 will describe the method utilized to synthesize the fully synchronous and


*INA-OCMC '11,* January 23, 2011, Heraklion, Greece




(a) The NoC is split into multiple VFIs. (b) The NoC is a global clock domain. (c) Mesochronous NoC for GALS systems.

**Figure 1. Paradigms for GALS synchronization**

the mesochrnous GALS system. Section 5 will present a comparison in terms of area, wiring and power overhead. Finally, Section 6 concludes this work with a final discussion and directions for future work.

## 2. Target GALS Architecture

A GALS-based design style fits nicely with the concept of voltage and frequency islands (VFIs), which has been introduced to achieve fine-grain system-level power management and is currently driving the transition of most MPSoCs to GALS systems. In these systems, if network components belong to the core's VFIs as in Fig.1(a), then performance of communication flows would be determined by the slowest domain crossed on the way to destination. Also, in case a VFI is shut down, global connectivity is jeopardized.

An alternative solution is illustrated in Fig.1(b),Âăwhere the NoC lies in its independent VFI. This way, performance of the whole switching fabric would be homogeneous, with only boundary effects to take care of. Also, the network would be loosely coupled with the cores' VFIs, and each core/cluster of cores could be shutdown without any impact on global network connectivity.

The main issue with an independent NoC VFI consists of the feasibility of its clock tree. The reverse scaling of interconnect delays and the growing role of process variations are some of the root causes for this. Even though inferring a global clock tree for the entire network will still be feasible for some time, it will probably come at a significant power cost. Moreover, it is unclear when the difficulty of tightly and globally enforcing the skew constraint will truly become a roadblock.

However, a workaround for this problem does exist, as illustrated in Fig.1(c). The network could be inferred as a collection of mesochronous domains, instead of a global synchronous domain, yet retaining a globally synchronous perspective of the network itself. There are several methods to do this. One simple way is to go through a hierarchical clock tree synthesis process. In practice, a local clock tree is synthesized for each mesochronous domain, where the skew constraint is enforced to be as tight as in traditional synchronous designs. Then, a top-level clock tree is synthesized, connecting the leaf trees with the centralized clock source, with a very loose clock skew constraint. This way, many repeaters and buffers, which are used to keep signals in phase, can be removed, reducing power and thermal dissipation of the top-level clock tree. The granularity of a mesochronous domain can be as fine as a NoC switch block.

The communication between neighboring switches is then mesochronous as the clock tree is not equilibrated, while the communications between switch and IP cores are fully asynchronous because they belong to different clock domains. Bi-synchronous FIFOs are therefore used to connect the network switches to the network interfaces of the cores, as showed in Fig.1(c).

This synchronization paradigm comes with additional advantages. First, it makes a conscious use of area/power-hungry dual-clock FIFOs, which end up being instantiated only at network boundaries. Instead, more compact mesochronous synchronizers are used inside the network, thus minimizing the cost for GALS technology.

Finally, unlike fully asynchronous interconnect fabrics, the synchronizer-based source-synchronous GALS architecture illustrated in Fig.1(c) is within reach of current mainstream design toolflows with just incremental effort. Typically, some scripting effort within commercial synthesis frameworks enables these latter to meet the physical requirements of source-synchronous designs [2, 10].

In the rest of this paper, the architectures in Fig.1(b) and Fig.1(c) will be compared from many viewpoints by means of physical synthesis runs, to quantify when exactly to migrate away from the architecture of Fig.1(b) and the actual overhead of the architecture of Fig.1(c).

The xpipesLite NoC architecture [1] is used as baseline experimental setting to implement both GALS platforms. The flow control protocol used by xpipesLite is stall/go: a forward signal, synchronous with data, flags data availability (valid), while a backward signal flags a destination buffer full (stall) or empty (go) condition.

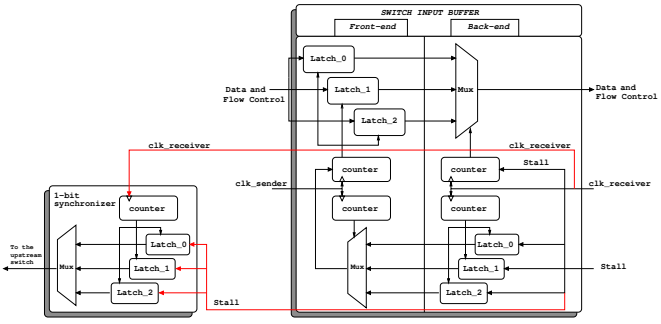



**Figure 2. Hybrid mesochronous synchronizer.**

## 3. Hybrid coupling of synchronizer with the NoC

Usually, mesochronous synchronizers are just placed in front of the downstream switch (the *loosely coupled design style*). This has implications on the size of the switch input buffer as well, which should cover the round trip latency to sustain maximum throughput. Given the large buffering and latency overhead of this approach, we proposed in [5] to bring the synchronizer deeper into the downstream switch, as illustrated in Fig.2.

The reference synchronizer architecture receives as its inputs a bundle of NoC wires representing a regular NoC link, carrying data and/or flow control commands, and a copy of the clock signal of the sender used as a strobe signal for them. The circuit is composed by a front-end and a back-end. The front-end is driven by the incoming clock signal, and strobes the incoming data and flow control wires onto a set of parallel latches in a rotating fashion, based on a counter. The back-end of the circuit leverages the local clock, and samples data from one of the latches in the front-end thanks to multiplexing logic which is also based on a counter. The rationale is to temporarily store incoming information in one of the front-end latches, using the incoming clock wire to avoid any timing problem related to the clock phase offset. Once the information stored in the latch is stable, it can be read, processed and sampled by the target clock domain.

In the architecture of Fig.2, the synchronizer output now directly feeds the switch arbitration logic and its internal crossbar, thus ma-

terializing the tight coupling concept of the mesochronous synchronizer with the switch architecture. The ultimate consequence is that the mesochronous synchronizer becomes the actual switch input stage, with its latching banks serving both for performance-oriented buffering and synchronization. A side benefit is that the latency of the synchronization stage in front of the switch is removed, since now the synchronizer and the switch input buffer coincide. The buffering overhead in the switch input buffer because of flow control is also removed accordingly.

The main change required for the correct operation of the new architecture is to bring the stall/go flow control signal to the front-end and back-end counters of the synchronizer, in order to freeze their operation in case of a stall. While this signal is already in synch with the back-end counter, it should be synchronized with the transmitter clock before feeding the front-end counter. The backward-propagating stall/go is then directly synchronized with the transmitter clock available in the front-end by means of a similar but smaller (1-bit) synchronizer.

For this architecture solution, only 3 latching banks are needed in the synchronizer front-end, since link latency has been minimized. In practice, only 1 slot more than the input buffer in the fully synchronous switch. A loosely coupled approach would require a 4 slot input buffer and a 3 latch banks synchronizer.

As regards the control path, a 1-bit synchronizer is replicated in front of the upstream switch. This synchronizer is not merged with the downstream buffer, since this would give rise to overly tight timing constraints [3]. In contrast, integrating only the data-path synchronizer is denoted as the *hybrid coupling* and gives more guarantees for timing closure, and is the approach taken hereafter.

# 4. Synthesis of GALS Platforms

Both the synchronous and the mesochronous platforms have been designed to be seamlessly integrated into an industrial design flow using commercial tools for physical synthesis. Only standard cells are used and no full custom components.

The reference topology of our experiment is a 4x4 mesh network where each switch is connected to either a core or a memory (of size 1.5mm). As far as the physical synthesis is concerned, the same bottom-up methodology has been utilized for both platforms. Specifically, each network switch has been placed and routed in isolation with a target frequency of 500MHz. The clock tree of each switch has been synthesized with a tight skew constraint of 5% of the target clock period. Once the local clock tree is characterized with its input delay, skew and input capacitance, a *macromodel* is built in order to be used in the next design step. Furthermore, in order to implement a hierarchical clock tree synthesis, a buffer has been inserted to the input clock pin of each switch block. Once the switches have been placed and routed, they are imported as macro blocks in the main network design along with their libraries detailing both timing and physical characteristics. The next step consists of performing a top-level clock tree synthesis by leveraging the switch macromodels previously extracted. In fact, this model can be used to characterized the bottom clock tree given that these local clock trees will not be modified by the place&route tool. Therefore, in order to preserve the clock tree local to the switches, a $PreservePin$ tag must be used in the CTS specification file.

Please notice that the hierarchical CTS has been used both for the synchronous and the mesochronous platforms, since this is a standard methodology for parallel hardware platforms. *The only difference is the skew constraint in the top level clock tree, which can be loosened for the mesochronous design while should be tightly enforced for the synchronous one.*

Final step of our hierarchical methodology consists of routing the switch-to-switch links and performing parasitics extraction for accurate static-timing analysis and power estimation.

Timing closure for both the synchronous and mesochronous NoC has been achieved at 500 MHz by performing exactly the same physical synthesis steps.

# 5. Experimental results

## 5.1 Area and Wiring Overhead

Figure 4 reports post-place&route area and wiring statistics for the architectures under analysis. From an area viewpoint, both systems exhibit the same footprint. More in detail, our baseline architecture (i.e., the fully synchronous mesh) features a 2-slots input and 6-slots output buffers. On the other hand, its mesochronous counterpart has 3-slots input buffer and exactly the same amount of output buffering. Nonetheless, the area overhead is identical.

This is due to the fact that synchronization mechanisms, tightly coupled in the input buffer, are implemented through latch banks, which require typically a smaller area footprint compared to the flip-flops adopted in the input buffer of the baseline architecture. The ultimate result is an equal area occupation in both platforms although this comes with a somewhat more challenging testing framework.

From the wiring point of view, a 23% net saving is achieved by the fully synchronous platform. The reason lies in the fact that the mesochronous platform features an additional clock wire per output port utilized as strobe signal for data synchronization and a further external single bit synchronizer for backward flow control synchronization instantiated in each of the 48 switch–to–switch channels of the network. Last but not least, the slightly more complex network topology contributes to a more complex structure of the clock tree.

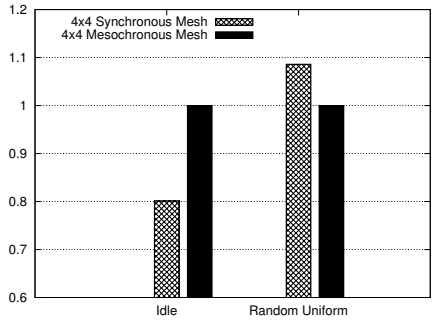



**Figure 3. Power consumption with no activity and with uniform random traffic (normalized with respect to the mesochronous network).**

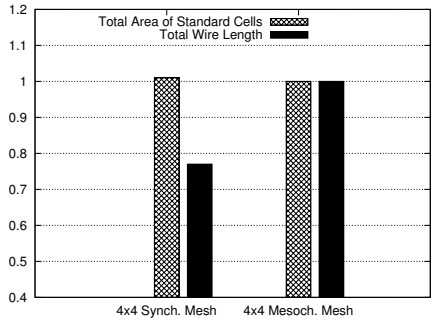



**Figure 4. Area and wiring intricacy (normalized with respect to the mesochronous network).**

## 5.2 Power analysis

By leveraging post-layout netlists and back-annotated switching activity, we were able to achieve very accurate power figures.

In fact, cycle-accurate simulations have been carried out with uniform random traffic as well as with all the network switches in idle conditions. A value-change-dump file (VCD) has been annotated from the simulations and consequently utilized to carry out a very accurate power estimation with Synopsys PrimeTimePX. Figure 3 reports power consumption of both fully synchronous and mesochronous networks. Idle power plays in favor of the fully synchronous network. This is mainly due to the additional switch–to–switch clock wire used as strobe signal for data synchronization. This result calls for further evolution of mesochronous NoC technology, to implement a form of clock gating on these lines.

On the other hand, when stimulating the networks with a uniform random traffic pattern, the mesochronous Network-on-Chip exhibits a smaller power consumption with respect to the fully synchronous one. The reason lies in the inherent architectural difference between the input buffer of the mesochronous switch and of the synchronous one. In this latter, both flip-flop banks are triggered at each clock cycle. Conversely, latch banks of the mesochronous input buffer are triggered by an enable signal driven by a counter. Since the counter logic enables only a single latch bank at a time, the ultimate register power consumption of the mesochronous input buffer is smaller than its synchronous counterpart.

With a mesochronous NoC, an interesting opportunity pointed by [7–9] is to exploit hierarchical clock tree synthesis to reduce power of the top level clock tree. The tuning knob to materialize power savings is the relaxation of the skew constraint, so that less buffers are instantiated in the top-level tree. We experimented this on the 4x4 mesochronous mesh by incrementally relaxing the skew constraint. Given the relatively small system size, we constrained the top level tree to be placed and routed outside IP core area, which captures the challenging requirements of many real-life MPSoC designs.

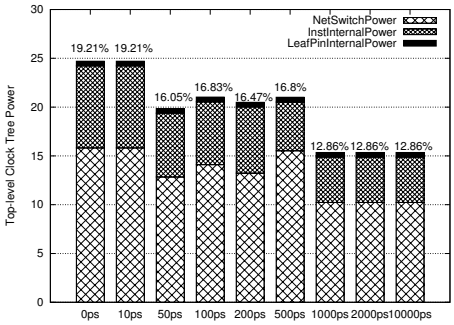



**Figure 5. Power of the top level clock tree as a function of required skew.**

Power of the top-level clock tree is reported in Fig.5 as a function of the required clock skew, ranging from 0 to 10000 ps. Transition time constraints are set to be very tight for the CTS tool (Cadence SoCEncounter). The percentage on top of the bars indicates the impact of the top level tree on the total clock tree power. We can see that power of the top level tree can be decreased by up to 40%, from roughly 25mW to 15mW. Also, the impact of the top level tree on total clock tree power can be as large as 20%.

Fig.5 may be misleading. In fact, the required skew does not keep up with the actually enforced skew by the CTS tool. In fact, when 2000 ps or more were required, the achieved clock skew saturates at about 600ps. Power savings could be even more than those reported if only the CTS tool was able to infer larger skews while saving clock tree area. Unfortunately, the CTS has not been natively conceived for this, but rather for the opposite: minimizing the skew.

| Required Skew | Actual Skew | Top tree Power | % of Total clock tree |
|---|---|---|---|
| 10ps | 320ps | 64.817mW | 13.93% |
| 1000ps | 973ps | 61.307mW | 13.17% |

**Table 1. Top Clock Tree Power for a 64 cores system**

There is another effect that becomes apparent when the same experiment is performed on an 8x8 2D mesochronous mesh with 64 cores. Results are reported in Table 1. The CTS tool was not able to meet the required upper bound on the skew of the top level clock tree. When 10 ps were required, the actual skew resulted 320 ps. The clear message here is that as the system size becomes large and (not showed here) feature sizes shrink, it will become impossible to meet the desired skew constraint in the top level clock tree. This calls for a skew absorbing mechanism in the NoC architecture.

## 6. Conclusions and Discussion

Evolution of MPSoCs to GALS systems is an ongoing process, driven by the immediate need to decouple voltage and frequency of IP cores from each other for power management. However, when a GALS NoC is implemented as an independent clock domain with dual-clock FIFOs at the boundaries, then physical designers have again to deal with a global chip wide clock tree (i.e., the one of the network itself). By capitalizing on mature mesochronous technology, this paper compares a mesochronous NoC and a fully synchronous NoC (both for use in a GALS system) in a systematic way. The lesson learned from this experimental work can be summarized as follows:

1. A fully synchronous NoC can be evolved to a mesochronous NoC at no area and latency overhead because of the hybrid coupling design style.
2. During network activity, the mesochronous NoC proves more power-efficient because of the inherent clock gating implemented at its tightly coupled synchronizers in the switch input buffer. In contrast, a 20% higher standby power is incurred because of the transmitted and continuously switching clock signals in source synchronous links. A clock gating technique applied to the switch input buffer of the synchronous NoC and to the source synchronous links of the mesochronous NoC may align power results of the two solutions. In any case, the mesochronous NoC does not feature any significant overhead from a power viewpoint.
3. Hierarchical clock tree synthesis in a mesochronous NoC can potentially reduce total power of the top level clock tree at the cost of progressively loosening skew constraints in the same tree. However, power savings achieved in this way are not significant yet, for a number of concurrent reasons. First of all, there is a gap between the required maximum skew and the obtained one, since the CTS tool has been conceived for minimizing skew, and not for increasing it. Therefore, to take full advantage of this effect, CTS tools should be customized accordingly.
4. On the other hand, when tight skew constraints are required under challenging physical and timing constraints, the CTS tool is not able to meet the target. In practice, this means that with current CAD tools it will become rapidly impossible to enforce tight skew constraints in the top level clock tree. Under these operating conditions, it is important to have an underlying architecture with inherent skew robustness. In our experiments, mesochronous NoCs prove capable of meeting this requirement.

## Acknowledgements

This work has been partially supported by the GALAXY European Project (FP7-ICT-214364), by the Hipeac Network of Excellence (Interconnect Cluster)

## 7. References

[1] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, G. De Micheli, “xpipes Lite: a Synthesis Oriented Design Library for Networks on Chips”, Proc. of DATE, pp.1188–1193, 2005.
[2] Kakoee, M.R. and Loi, I. and Benini, L., “A New Physical Routing Approach for Robust Bundled Signaling on NoC Links”, Proc. of GLSVLSI, pp.3-8, 2010.
[3] D. Ludovici and A. Strano and G. N. Gaydadjiev and L. Benini and D. Bertozzi, “Design Space Exploration of a Mesochronous Link for Cost-Effective and Flexible GALS NOCs”, Proc. of DATE, pp.679-684, 2010.
[4] D. Ludovici, A. Strano, D. Bertozzi, “Architecture design principles for the integration of synchronization interfaces into Network-on-Chip switches”, Proc. of NoCArc, pp.31-36, 2009.
[5] D. Ludovici, A. Strano, D. Bertozzi, L. Benini, G.N. Gaydadjiev, “Comparing tightly and loosely coupled mesochronous synchronizers in a NoC switch architecture”, Proc. of NOCS, pp.244-249, 2009.
[6] A. Strano and D. Ludovici and D. Bertozzi, “A Library of Dual-Clock FIFOs for Cost-Effective and Flexible MPSoCs Design”, Proc. of SAMOS, 2010.
[7] T.N.K.Jain, “Asynchronous Bypass Channels: Improving Performance for Multi-Synchronous NoCs”, Proc. of NOCS, pp.51-58, 2010.
[8] F.Vitullo et al. “Low-Complexity Link Microarchitecture for Mesochronous Communication in Networks-on-Chip”, IEEE Trans. on Computers, Vol.57, issue 9, pp.1196-1201, 2008.
[9] D. Mangano, R. Locatelli, A. Scandurra, C. Pistritto, M. Coppola, L. Fanucci, F. Vitullo, D. Zandri, “Skew Insensitive Physical Links for Network on Chip, 1st International Conference on Nano-Networks (NANO-NET 2006)”, Proc of NANO-NET, 2006.
[10] I.M.Panades, F.Clermidy, P.Vivet, A.Greiner, “Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture”, Proc. of NOCS, pp.139-148, 2008.
[11] D.Wentzlaff et al., “On-Chip Interconnection Architecture of the Tile Processor”, IEEE Micro, vol.27, no.5, pp.15-31, 2007.
[12] D. A. IIitzky, J. D. Hoffman, A. Chun and B. P. Esparza, Architecture of the Scalable Communications Core’s Network on Chip”, IEEE Micro, vol.27, Issue 5, pp.62 - 74,2007.
[13] F.Clermidy, R.Lemaire, X.Popon, D.Ktenas, Y.Thonnart, “An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application”, Proc of DSD, pp.62-74,2009.
[14] F.Clermidy, C.Bernard, R.Lemaire, J.Martin, I.Miro-Panades, Y.Thonnart, P.Vivet, N.Wehn, “A 477mW NoC-based Digital Baseband for MIMO 4G SDR”, ISSCC’2010, pp.278-279, 2010.
[15] Y.Thonnart, P.Vivet, F.Clermidy, “A Fully-Asynchronous Low-Power Framework for GALS NoC Integration”, Proc. of DATE, pp.33-38, 2010.
[16] M. Krstic et al., “Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook”, IEEE Design and Test of Computers, vol. 24, no. 5, pp. 430-441, 2007.
[17] S.Vangal et al.; “An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS”, IEEE Journal of Solid-State Circuits, Vol.43, Issue 1, pp.29-41,2008.
[18] E. Flamand, “Strategic Directions Towards Multicore Application Specific Computing”, Proc. of DATE, pp.1266, 2009.
[19] S.Borkar, “Design Perspectives on 22nm CMOS and Beyond”, Proc. of DAC 2009.
[20] R.Dobkin, V.Vishnyakov, E.Friedman, R.Ginosar, “An Asynchronous Router for Multiple Service Levels Networks on Chip”, Proc. of ASYNC, pp.44-53, 2005.
[21] T.Bjerregaard, J.Sparso, “A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip”, Proc. of DATE, pp.1226-1231, 2005.