# Design of a HW/SW Communication Infrastructure for a Heterogeneous Reconfigurable Processor

A. Deledda, C. Mucci, A. Vitkovski
ARCES, University of Bologna, Italy

M. Kuehnle, F. Ries, M. Huebner, J. Becker
ITIV, University of Karlsruhe, Germany

P. Bonnot, A. Grasset, P. Millet
THALES Research and Technology, France

M. Coppola, L. Pieralisi, R. Locatelli, G. Maruccia
STMicroelectronics, France

F. Campi, T. DeMarco
STMicroelectronics, Italy

## Abstract

*Reconfigurable architectures and NoC (Network-on-Chip) have introduced new research directions for technology and flexibility issues, which have been largely investigated in the last decades. Exploiting run-time adaptivity opens a new area of research by considering dynamic reconfiguration. In this paper, we present the architecture and associated development tools of an heterogeneous reconfigurable SoC focusing on the chosen communication infrastructure. The SOC integrates units of various sizes of reconfiguration granularity. The included NoC approach demonstrates the mentioned benefits and scalability for actual and future SoC design.*

*On a reference CMOS090 implementation the described interconnect system works at the system reference frequency of 200 MHZ sustaining the required run-time bandwidth on a set of reference applications, at a price $< 10\%$ in area in power consumption with respect to the overall system.*

## 1. Introduction

Data intensive processing in embedded systems is receiving relevant attention, due to rapid advancements in multimedia computing and high-speed telecommunications. Applications demand high performance under real-time requirements, and computation power appetite soars faster than Moore's law. Processor efficiency is impaired by the memory bandwidth problem of traditional Von Neumann architectures. On the other hand, the conventional way to boost performance through Application Specific Integrated Circuits (ASIC) suffers from sky-rocketing manufacturing costs and long design development cycles. This results in an increasing need of post-fabrication programmability at both software and hardware level. Field Programmable Gate Arrays (FPGA) bring maximum flexibility with their fine grain architecture, but imply severe overheads in timing, area and consumption. Word or sub-word oriented Run-time Reconfigurable Architectures (RAs) [1] offer highly parallel, scalable solutions combining hardware performance with software flexibility. Their coarser granularity reduces area, delay, power consumption and reconfiguration time, but introduces tradeoffs in the design of the processing elements, that need to be tailored for a given application domain. A possible way to mitigate this aspect for building a flexible yet efficient signal processor is to substitute each ASIC accelerator with a specific domain-oriented RAs, inducing a graceful shift of SoCs from application specific circuits to domain oriented platforms, where different flavours of reconfigurable hardware, each more suited to a given application environment, are merged with ASIC and general purpose processors to provide ideal tradeoff between performance and post-fabrication programmability. The immediate advantage is that the higher computational density of RAs allows to build networks composed of a significantly smaller number of nodes. The immediate drawback is the need to synchronize units that are intrinsically different and provide independent application mapping styles and entry languages. In this context, critical issues are related to the definition of

- a toolset that must be capable to hide RA heterogeneity and hardware details providing a consistent and homogeneous programming model to the end user
- a data interconnect infrastructure, that must sustain the bandwidth requirements of the computation units

The work presented in this paper is done within the MORPHEUS project (IST FP6, project no. 027342), which is sponsored by the European Commission under the 6th Framework program.

978-3-9810801-3-1/DATE08 © 2008 EDAA

while retaining a sufficient level of programmability to be adapted to all the different data flows defined over the architecture in its lifetime

These aspects are strictly correlated and their combination, together with the strategy deployed for RA computation synchronization represents the signal processor interface toward the end-user. In particular, the architecture view shall be abstracted as much as possible for the user, providing a programming model looking like purely functional code. Program parts requiring acceleration on RAs should be identifiable in the easiest possible way. A toolset can then handle and program the code corresponding to data movements and reconfigurations related to these accelerating parts. In this context not only computation but also communication aspects must indeed be considered. This will enable performance optimization by masking communication time by computation time through a "pipelined" behaviour. The scheduling of these accelerating parts among each other, including loading configuration and execution, may be managed at compilation time based on RTOS-oriented services.

This work was performed in the context of the MORPHEUS project. The project aims at realizing an heterogeneous reconfigurable SoC platform, where state-of-the-art RAs of different size and nature are grouped together in a processor-controlled system. In particular, this paper aims at describing the most significant challenges and design choices that have been faced in the deployment of a well known NoC infrastructure (the ST Spidergon NoC approach [10]) to the MORPHEUS context and the consequent impact on the architecture and toolset definition. We believe that the most relevant innovation aspects of this work are: (1) A significant milestone in the field of Heterogeneous multi-core SoCs. (2) The first design-case challenging the deployment of the NoC concept to a network of high-bandwidth computation intensive RAs.

## 2. Related Works

In the case of most of state-of-the art signal processors (OMAP by TI, Nomadik by ST, PXA by Intel) the interconnect strategy is two-fold: a multi-layered bus is utilized to ensure flexible chip level communication, control and synchronization. Computation intensive sections are normally implemented on hardwired cores or application-specific programmable accelerators. In this case the interconnect is designed as part of the SoC and optimized for the specific application domain. In the case of MORPHEUS, the user is required to partition computational demands of the application over available units after fabrication. Order, direction and relative bandwidth of the traffic between each HREs and to/from IO facilities remains run-time configurable and may be required to change significantly between successive applications and between different stages of the same computations. Similar specifications constrain the design of multi-processor Systems-on-chip (MPSoCs): MPSoCs are often heterogeneous but differently from MORPHEUS processing nodes tend to provide comparable granularity and similar IO bandwidths and access patterns to the communication infrastructure. From the implementation point of view MPSoCs may be tile-oriented [11, 12] or randomly placed. In the first case the cores are placed in a regular Mesh, and interconnect infrastructure is strongly hierarchical and distributed between the different tiles with the possible addition of global wires. An example is the RAW processor: RAW incorporates two types of networks (dynamic and static) that handle different classes of traffic. The dynamic network is a dimension-ordered wormhole network, while the static network implements time-division multiplexing. Networks-on-Chip [2, 3, 10] are a largely successful communication pattern for MPSoCs. The NoC concept in itself can certainly be applied to networks of heterogeneous processing nodes, but to our knowledge has never been explicitly applied to nodes that feature computational capability and bandwidth requirements of state-of-the-art RAs. In [4] a deployment of the NoC concept for a RA template is described, but in this case the NoC is used to interconnect different elements of a given RA array rather than intrinsically different coarser RAs. In [15] also a NoC based communication infrastructure is introduced for a multicore runtime reconfigurable System-on-Chip, utilizing packet-switched interconnect with virtual channels. The design provides guaranteed as well as best effort services introducing a significant amount of control overhead in hardware. A significant feature is that applied RAs are homogeneous, whereas heterogeneity is one of the major challenges of MORPHEUS. Pleiades [6, 13] can be considered the first Heterogeneous Multi-Core SoC appeared in literature, although it was targeted at low power consumption rather than processing efficiency. It is composed by an ARM8 core, memory modules, an embedded FPGA and a set of "general purpose" hardware accelerators fed by address generation engines. Interconnect is circuit-switched, based on a two-level mesh composed of hierarchical switch-boxes located at key connection points.

## 3. Proposed Architecture

Figure 1 describes the Morpheus architecture. The main entities are an ARM9 processor performing control and synchronization, the interconnect infrastructure, IO facilities, an automated mechanism to handle run-time reconfiguration and the three main RA units defined in this context HREs (Heterogeneous Reconfigurable Engines). HREs have been chosen in order to provide complementary features that can cover the spectrum of existing RA ap-

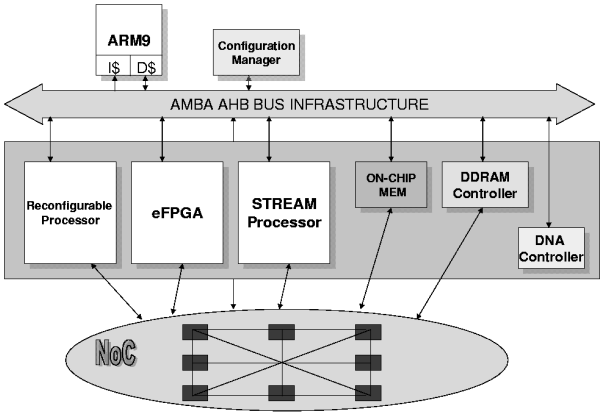


**Figure 1. Morpheus Architecture**

proaches: a stream processor based on coarse grained 16-bit units [7] suited to data-intensive word-level computation; a reconfigurable processor (also termed DREAM) [8], that is a standard RISC processor featuring instruction set extension on a 4-bit grained run time programmable datapath; a state-of-the-art embedded FPGA [9]. The MORPHEUS programming model is based on the Molen paradigm [14]: HREs provide instruction set extension, and tasks running on HREs should be seen as operators of the processor. Extensions are handled by the user only when for optimization reason he/she will program manually extension operations on HREs. Otherwise, they will be handled by specific libraries or by the MORPHEUS toolset. Increasing the granularity of operators from ALU-like instructions to HRE tasks, we are forced to increase accordingly the granularity of the operands. Operands become structured data chunks, referenced through their addressing pattern, be it simple or complex (vectorized and/or circular addressing based on multi-dimensional step/stride/mask parameters).

As described above, the aim of this work is provide a consistent interface and programming model to the end user. This is accomplished by a global toolset for platform-level design strictly related to a flexible interconnect infrastructure that can meet the requirements of the HREs. The most relevant differences between a standard MPSoC environment and a network of heterogeneous reconfigurable engines can be described as follows:

1. RAs, differently from standard processors, feature clock speed that depend on their granularity and also on the chosen application and the strategy deployed in its mapping. Hence, a multi-core platform based on RAs should feature run-time programmable, independent clock domains associated to each computational core and "elastic" communication channels to minimize bottlenecks.

2. RAs provide relevant computation capability so a network of HREs will feature a much smaller number of nodes but relevant bandwidth requirements for each transfer with respect to a network of processors. Also, data flows to/from RAs are typically regular, rather then organized in bursts. As a result congestion issues should be less frequent, but they should be avoided at all costs as they could dramatically reduce computation capabilities starving computation nodes.

3. RAs usually feature long reconfiguration times. Hence, they tend to iterate the same computation kernel repetitively over large chunks of data. The configuration of the interconnect infrastructure can thus be defined as quasi-static: transfers are normally coarser than those observed in a MPSoC environment and the routing can be often maintained identical for a significant set of consecutive transfers generated by a given initiator node. As a consequence, a circuit switched communication pattern appears suitable. On the other hand, such approach may limit the flexibility of the communication and is more prone to congestion issues.

4. RAs are often based on streaming data access patterns [7] or on automated regular addressing mechanisms [8, 15]. They can hardly behave as traffic initiators, because they rarely provide the addressing and bus management flexibility of a standard processor. Even in cases where this may be possible, the end-user would be forced to design and synchronize the system data flow programming different heterogeneous machines with inherently different languages and programming styles (e.g C, HDL).

According to the points outlined above, the most suitable approach to a NoC-based interconnect for a RA-based signal processor appears to be a network composed by few nodes, but with relevant bandwidth capabilities for each node-to-node connection. To avoid congestion, the topology should be designed as to provide direct and almost dedicated connections on the most critical paths. This notwithstanding, a circuit switched approach appears too application-specific, and does not provide sufficient guarantees for post-fabrication definition of new data-flows. A packet-switched net is preferable, where some paths are given absolute priority at design time but alternative minor connections are still possible and to a given extent can be eased altering at run-time priority schemes. Moreover, it is convenient to structure HREs as independent clock domains. Dual port memory buffers (defined in this context DEB, Data Exchange Buffers), organized as FIFOs or Scratchpads, can be used as local storage repository, to hide computation and synchronization latencies and to ensure safe clock domain crossing. Finally, to allow the end user a single homogeneous interface when describing

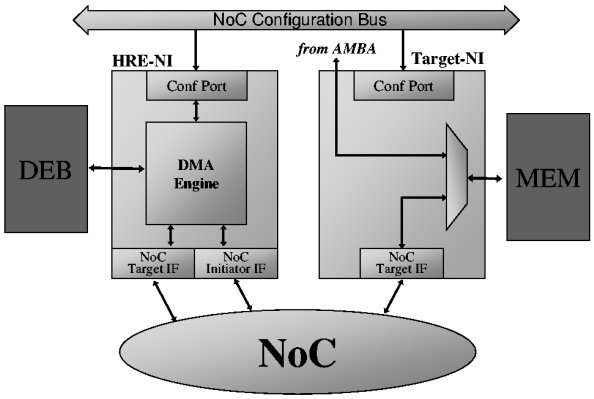


**Figure 2. Description of the network interfaces for HREs and NIs**

data transfers HREs may be provided with local DMA-like data transfer engines, programmed and controlled at system level through the toolset. Communication synchronization and control may be handled by software routines running on the main processor, but this may impact performance and impose an awkward programming interface. As suggested in [4] this task can be performed by RTOS services, and part of the OS can be implemented in hardware: hardwired centralized control can be provided to support multi block transfers as well as of a synchronisation concept and therefore disburden the processor/user from low-level tasks. From the application point of view, the programmer may then choose the preferred approach depending on constrains such as e.g. transfer data size or the activity level of the control processor.

Chip level interconnect was organized on two levels: (a) a communication kernel implementing transfers between nodes and (b) a HW/SW infrastructure that provides communication/synchronization towards the processor core (and thus the end user) and injects/extracts data to/from the communication kernel. The STNoC/Spidergon [10] concept was adopted as physical layer for the communication.

Spidergon NoC is based on a scalable, regular, point-to-point topology. The Spidergon network connects an even number of nodes in the ring in such a way that every node has three bidirectional connections: two to the neighbour nodes and one to the opposite node. (Fig.3). The main advantages of such topology are: regular structure, sufficiently short network diameter (e.g. minimum amount of hops between any two nodes in the network); low interconnection complexity comparing to the other network topologies and a simple routing mechanism. The Spidergon NoC implements a packet-based communication, adopting wormhole switching. Indeed wormhole scheme allows to reduce the amount of network buffering (queues with the flit granularity instead of packet) and to pipeline packet propagation. A deterministic, shortest-path routing algorithm has been defined for the Spidergon topology. The relevant implementation is very simple, without posing the need for expensive routing tables and ensuring fast processing of the packet header. The idea is to move along the ring, in the proper direction, to reach nodes which are close to the source node, otherwise to use the cross link to be in the oppisite part of the network. Two virtual channels in the ring links (clockwise and anticlockwise) guarantee deadlock avoidance. The concept of the deterministic routing and virtual channel scheduling avoids costly disordered end-to-end transfers: the routing path of any packet does not depend on the route of any other packet.

The NoC structure is specifically designed to hide implementation details, so that a consistent programming model can be developed without considering implementation parameters and may remain valid changing the number or the nature of HREs. The number of computation nodes and in particular the number of routers can be changed depending on architectural choices and floor-plan/timing analysis. A NoC is by definition a distributed communication platform with a set of initiator nodes (e.g. processor cores) issuing transfers and a set of target storage nodes (e.g. memory units) responding to transfer requests. HREs represent peculiar nodes: they should be both NoC initiators (require transfers from some storage units such as onchip or offchip RAM), and targets (process external requests such as a transfer request from another HRE or ARM). This is implemented over the NoC through a distributed DMA pattern: in order to act as traffic initiator each HRE node Network Interface (HRE-NI) is enhanced with a local software programmable data transfer engine (Fig.2). HRE-NIs are thus capable to load data chunks from HREs and store them through the NoC to the target repository and vice-versa. From the core/user point of view this approach describes the NoC as an enlarged and highy parallel DMA architecture. ARM can require any transfer between HREs, as well as from any HRE to any storage unit (Onchip memory, Memory controllers). Transfers are initiated programming specific configuration registers on the HRE network interface, through a specific configuration bus reaching all HRE-NIs. NoC router priority schemes may also be programmed through the same configuration bus.

As mentioned above, quasi static data transfers are expected to be the greatest volume of communication load on the network. This positively affects system control, performed by ARM by means of RTOS, since transfers can be issued as standard DMA trasnfers. On the other hand, since MORPHEUS is a single master system, it may suffer syncronisation overload due to an intensive interrupt activity (depending on the granularity of the tasks mapped on HREs). A possible solution is to map part of the RTOS services on hardware. A hardwired DNA (Data Network access) controller has been designed, featuring setup mech-

| Entity | Size | Area ($mm^2$) | Power (Dyn,Leak) ($\mu$W/MHZ-mW) |
|---|---|---|---|
| HRE-NIs | 4x90 Kgate | 1.58 | 148 - 1.8 |
| DNA Controller | 40 Kgate | 0.25 | 24 - 0.3 |
| NoC Routers | 8x44 Kgate | 2.8 | 429 - 5 |
| NoC Initiators | 5x18 Kgate | | |
| NoC Targets | 3x9 Kgate | | |
| DREAM Interc. | 50 Kgate | 2.4 | 282 - 6 |
| DREAM DEBs | 64 KB | | |
| M2K Interc. | 15 Kgate | 1.1 | 140 - 2 |
| M2K DEBs | 32 KB | | |
| Pact Interc. | 10 Kgate | 0.6 | 198 - 1 |
| Pact DEBs | 16 KB | | |
| Total Interc. | | 8.4 | 217 - 16.1 |

**Table 1. Hardware features of the main components of the interconnect infrastructure**

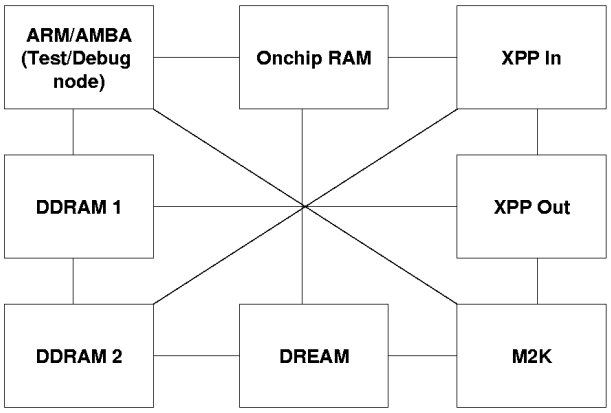


**Figure 3. Spidergon topology chosen for the CMOS090 MORPHEUS demonstrator**

anisms for data transfers and a hardware handshake protocol that support computation/communication synchronization and preserve data flow consistency. Its programming model is similar to a common DMA interface so that it remains transparent for the end user. Eight concurrent programmable channels are supported, each capable to control multiblock transfers. The DNA hides synchronization details from the end-user programming and reiterating NoC transfers configuring independently the HRE-NIs. The DNA handles interrupt based multiblock transfers with block sizes of up to 128kbyte by monitoring and evaluating the status of computation over the HRE nodes and of DMA transfer channels over the HRE-NIs. Configuration items for channel setup can be stored in the DNA memory beforehand and related events will be catched and resolved by the DNA independently from further ARM commands.

## 4 Application Mapping on HRE

HRE-specific code can be seen as the code necessary to compute HRE Instructions in a transparent way for the user. It includes bitstreams as well as code handling data transfers. Operators are synthesized from a high-level programming model or provided as libraries. They are described within the toolset as a data-flow graph where each subtask is a C function. The synthesis flow relies on the HRE providers tools to generate an HRE specific bitstream. ARM code handling communications is automatically generated and integrated in the application code. HRE operands are transferred between on-chip system memory and local HRE memories (DEBs). Being structured data, such as image frames, their size can be larger than DEB size. In this case, a specific tool [16] maps Macro-operators on the HREs. Application kernels have to be computed over many iterations working on macro-operand fragments. Hence, the tool designs a sort of lower granularity pipeline inside the HRE. In the data-flow graph of the kernel, each subtask executes iteratively a basic function on a data subset. Loop iterations are scheduled in a way that the data subset produced by the first N iterations of a stage are sufficient for the next stage to work. The tool assists the user to do it by providing a framework for loop transformations and loops fusion [16]. As the kernel model includes information on what data are "consumed" at each iteration, the tool can compute data transfers parameters. ARM code handling communication is generated by the compiler using these parameters. The toolset is also capable to generates HRE bitstreams.

Basic functions of each subtask are synthesized on HREs and are fed with specific data streams. According to loop iterations scheduling, specific addressing patterns are programmed inside the HRE to access to the DEBs. A Finite State Machine (FSM) is implemented inside each HRE (in SW or in HW) to control the progress of the pipeline and to synchronize it with the system. It controls all levels of computation vs data transfer synchronization. The input for the compilation flow is the C application code to compile and execute on the ARM processor. When writting an application, the programmer uses specific `#pragma` instructions in its C code to set in the application which function is accelerated. Doing so, the compiler identifies which function is to be computed on the HREs and modifies the function calls by generating code to (1) load the defined HRE with the corresponding bitstream, (2) program the DNA for chunk data movement between memories and DEBs to feed the accelerated function and (3) load back results after computation (i.e. DNA configuration). The code generated by the compiler optimizes scheduling loading the HRE bitstream as early as possible in order to make available the accelerated function when the application needs it. The scheduling is defined at compile time, and can be affected by events such as interrupts or bus congestion. Such occurrances are handled by RTOS ensuring the correctness of a part of the schedule at run-time, another part is directly

| Application | M2K/ Onchip RAM | DREAM/ Onchip RAM | M2K/ DDRAM | XPP/ Onchip RAM | XPP/ DREAM | XPP/ M2K | DREAM/ M2K |
|---|---|---|---|---|---|---|---|
| OTU-k frame processing | 10 Mb/s | 10 Mb/s | | | | | |
| IEEE 802.11j | | 312 Mb/s | 365.4 Mb/s | | 288 Mb/s | 390.4 Mb/s | 24 Mb/s |
| Motion Detection | | 354 Mb/s | | 177 Mb/s | | 177 Mb/s | 22 Mb/s |

**Table 2. Application Bandwidth Requirements**

manages by the hardware with handshaking mechanisms.

## 5 Quantitative Results

Currently, an instance of MORPHEUS platform is considered to be implemented for STMicroelectronics CMOS090 technology: most architectural features were silicon proven on preliminary test-chips while a complete prototype of the entire processor is currently under implementation. For the interconnect infrastructure, an 8-nodes NoC topology (Fig.3) has been implemented. It includes five initiator ports to: ARM, XPP In, XPP Out, M2K and DREAM; and seven target ports to: XPP In, XPP Out, M2K, DREAM, DDRAM 1, DDRAM 2 and On-chip RAM, which allow to maintain a tight connections with the communication critical components. As the XPP stream processor was reputed to be the most data hungry HRE in the design, two independent connections were reserved: XPP In and XPP Out. The nodes related to the same hardware component are placed close to each other due to the floor planning reasons. The eFPGA is provided with an additional direct IO access, so that it can also be used for dynamical mapping of IO protocols. For this reason, each of the other two HREs has a direct connection to eFPGA. Each link is set at 64-bit width, at the price of significant routing complexity, in order to sustain the relevant computational demands of the HREs. It was chosen to run all interconnects at the system reference speed of 200MHZ, discarding the STNoC intrinsic capability of running physical links at a different speed with respect to the Network Interfaces. This decision allows to minimize area overheads and simplify the implementation phase. More precisely the overheads imposed by integration of the three HREs and the NoC from the computational data side are presented in table 1.

In order to validate the proposed approach several applications were investigated, as shown in table 2. Their dataflows were mapped on the described architecture, considering to implement critical kernels in the most appropriate RA. Each column of table 2 represents the total bandwidth required for each physical link. According to the simulation results the implemented communication infrastructure is able to satisfy all the real-time constraints imposed by the mapped applications.

## 6. Conclusions

In the proposed architecture the combination of an innovative NoC infrastructure togheter with a distribuited memory subsystem introduces an efficient communication and storage mechanisms that allow to hide from the end user heterogeneus specifics of different reconfigurable engines.

Considering that in the MORPHEUS test-chip the estimated area is about 90 $mm^2$ and the average power consumption is in the range of 3000 mW, our goal is achieved with a price of $< 10\%$ overheads.

## References


[1] R.Hartenstein, *A decade of Reconfigurable Computing: a visionary Retrospective*, Proceedings DATE 2001

[2] A. Jantsch, H. Tenhunen *Networks on Chip* Springer, 2003, ISBN: 978-1-4020-7392-2

[3] G. De Micheli, L. Benini *Networks on Chips*, Morgan Kaufmann, 2006

[4] T.A.Bartic et.al. *Topology adaptive NoC design and implementation* IEE Computers and Digital Techniques, July 2005

[5] AMBA *Specification, rev. 2.0*, ARM Ltd. http://www.arm.com/products/solutions/AMBA_Spec.html

[6] M.Wan et al. *Design Methodology for a low-energy reconfigurable single-chip DSP* JVLSI Signal processing, Jan2001

[7] M. Vorbach, J. Becker, *Reconfigurable processor architectures for mobile phones* IPDPS, 2003

[8] F.Campi et al *A dynamically adaptive DSP for heterogeneous reconfigurable platforms* Proceedings of DATE 2007

[9] M2000 *Embedded FPGA* http://www.m2000.fr

[10] M.Coppola et al, *Spidergon: a novel on-chip communication network*, IEEE SOC 2004

[11] Bedford M. et al, *Evaluation of the Raw microprocessor*, Proceedings of ISCA04

[12] R.Baines et al. *A total cost approach to evaluate different RA for baseband processing in wireless receivers* IEEE Communication magazine, Jan 2003

[13] H. Zhang, et al. *A 1V Heterogeneus Reconfigurable Processor IC for Baseband Wireless Applications*, Proceeding of ISSCC 2000, pp 68-69

[14] S. Vassiliadis et al., *The MOLEN Polymorphic Processor*, IEEE Transactions on Computers, November 2004

[15] Gerard Smit et al., *Overview of the 4S Project*, International Symposium on System-on-Chip, Nov. 2005

[16] E. Lenormand, G. Edelin, *An industrial perspective: A pragmatic high end signal processing design environment at Thales*, SAMOS 2003