# HeMPS Station: an environment to evaluate distributed applications in NoC-based MPSoCs

**Cezar R. W. Reinbrecht, Gerson Scartezzini, Thiago R. da Rosa, Fernando G. Moraes**
{cezar.rwr, gersonscar, thiagoraupp00}@gmail.com, fernando.moraes@pucrs.br

**Undergraduate Work**
**PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul**

## Abstract

*Multi-Processor Systems-on-Chip (MPSoCs) are increasingly popular in embedded systems. Due to their complexity and huge design space to explore for such systems, CAD tools and frameworks to customize MPSoCs are mandatory. The main goal of this paper is to present a conceptual platform for MPSoC development, named HeMPS Station. HeMPS Station is derived from the MPSoC HeMPS. HeMPS, in its present state, includes the platform (NoC, processors, DMA, NI), embedded software (microkernel and applications) and a dedicated CAD tool to generate the required binaries and perform basic debugging. Experiments show the execution of a real application running in HeMPS.*

## 1. Introduction

The increasing demand for parallel processing in embedded systems and the submicron technologies progress make technically feasible the integration of a complete system on chip, creating the concept of SoCs (System-on-chip). In a near future, SoCs will integrate dozens of processing elements, achieving a great computational power in a single chip. This computational power is commonly used by complexes multimedia algorithms. Thus, systems based on a single processor are not suitable for this technological trend and the development of SoCs with more complexity will be multi processed, known as MPSoCs (Multi Processors System-on-chip).

Several MPSoC examples are available in academic and industrial areas. Examples of academic projects are HeMPS [CRIS07][CAR09], MPSoC-H [CAR05], [LIN05] and the HS-Scale MPSoC [SAI07]. Examples of industrial projects are Tile64 [TIL07] and [KIS06] from IBM.

The improvement of the MPSoC architectures makes possible the execution of high demands applications in a single chip. These applications exist in different areas, mainly in science, for example, genetic, organic chemical or even nuclear physics. Therefore, MPSoCs could solve them efficiently and with low cost. The cost issue refers that nowadays these applications are solved with efficiency just by super computers or clusters. In addition, the fact this technology has a great potential for parallel applications; it seems a better investment for industrial purposes.

MPSoCs are mostly used in embedded systems, usually without a user interface. However, designers should use dedicated environments to evaluate the performance of the distributed applications running in the target MPSoC. Such environments include: (*i*) dedicated CAD tools running in a given host; (*ii*) fast communication link between host and MPSoC to enable real time debugging and evaluation; (*iii*) monitoring schemes inserted into the MPSoC architecture enabling performance data (throughput, latency, energy) collection. The goal of this work is to present the proposal of an environment to help the development and evaluation of embedded applications targeting a NoC-based MPSoC. Such environment is named HeMPS Station.

## 2. HeMPS Station

HeMPS Station is a dedicated environment for MPSoC, basically a concept for a system structure to evaluate the performance of distributed applications in a given architecture running on an FPGA. This environment aims dedicated CAD tools running in a given host, fast communication link between host and MPSoC to enable evaluation during the execution and monitoring schemes inserted into the MPSoC architecture, enabling performance data collection. Therefore, HeMPS Station is composed by an MPSoC, an external memory and a Host/MPSoC Interface System, as shown in Figure 1.

The MPSoC is a master-slave architecture and it contains three elements: (i) interconnection element; (ii) processing element; (iii) monitoring element. The interconnection element is a Network-on-Chip [BEN02] with mesh topology. The processing element comprises a RISC processor and private memory. This element also needs a network interface and a microkernel. The monitoring element can be inserted in the interconnection element, in the processing element or in both elements. The interconnection monitors are inside the routers or attached at the wire and their functions are to collect information of the network like throughput, latency or

energy. On the other hand, the processing monitors are systems calls inserted in the tasks codes. With the systems calls, each processor informs a report about the task state.

The external memory is a repository attached to the master processor, storing all tasks of the application to be executed in the system. The master access this memory to allocate the tasks based on demand and availability of the system.

The Host/MPSoC Interface System manages three elements, the host, the master and the external memory. This system is responsible for a fast communication link between host and MPSoC, for loading code in the external memory, for the messages of debug and control with master and for the host's software that presents the performance data and translate the applications in code for the platform. The communication link is the bottleneck of the environment, so it needs to be practical and fast. Aiming these features the communication link uses Ethernet technology and Internet protocols. This make possible to reach high transmission rates since 10Mbps till 1000Mbps, and the internet protocols allows remote access.

The performance analysis is executed at the host, displaying to the user the performance parameters and the placement and state of each task during the execution. All collected data is stored in the host for further analysis. The main idea is to use the HeMPS Station through the internet, enabling the user to upload distributed applications into the MPSoC, and to receive after application(s) execution performance data.

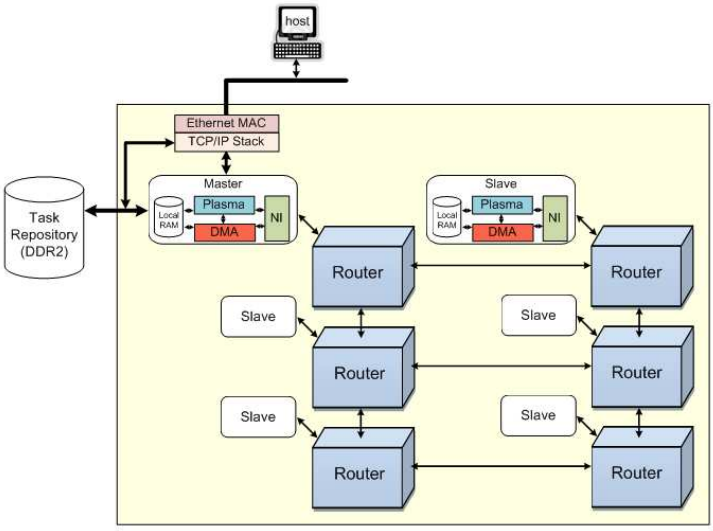



**Figure 1 – Hemps Station Environment**

## 3. Present HeMPS Environment

The HeMPS architecture is a homogeneous NoC-based MPSoC platform. Figure 2 presents a HeMPS instance using a 2x3 mesh NoC. The main hardware components are the HERMES NoC [MOR04] and the mostly-MIPS processor Plasma [PLA06]. A processing element, called Plasma-IP, wraps each Plasma and attaches it to the NoC. This IP also contains a private memory, a network interface and a DMA module.

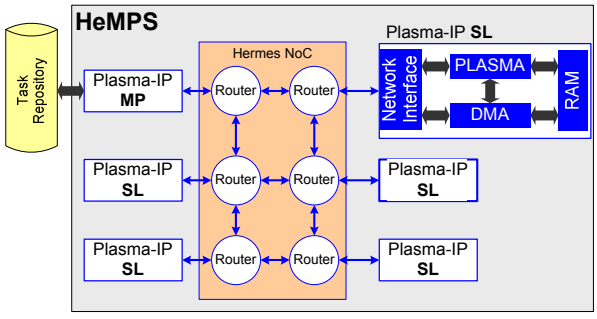



**Figure 2: HeMPS instance using a 2x3 mesh NoC.**

Applications running in HeMPS are modeled using task graphs, and each application must have at least one *initial task*. The *task repository* is an external MPSoC memory, keeping all task codes necessary to the applications execution.

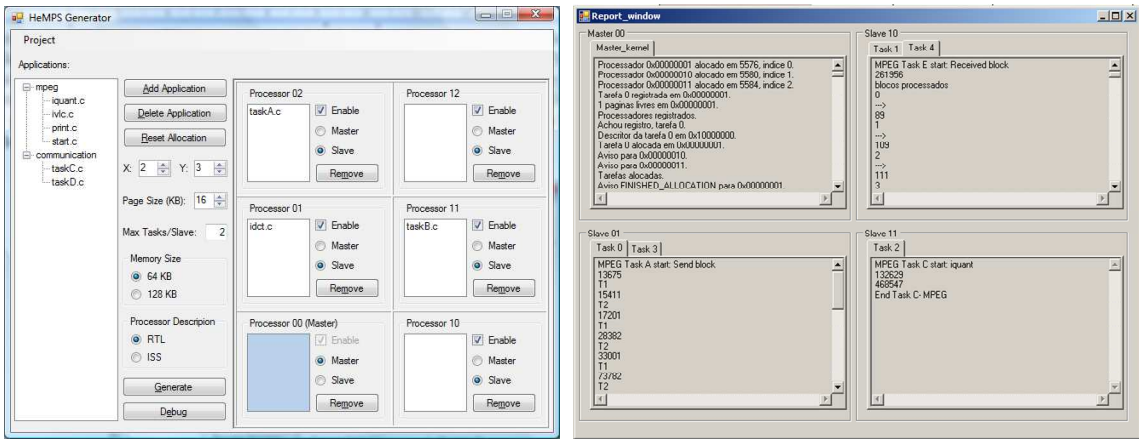
The system contains a *master processor* (Plasma-IP MP), responsible for managing system resources. This is the only processor having access to the task repository. When HeMPS starts execution, the master processor allocates initial tasks to the *slave processors* (Plasma-IP SL). During execution, each slave processor is enabled to ask to the master, on demand, dynamic allocation of the tasks from the task repository. Also, resources may

become available when a given task finishes execution. Such dynamic behavior enables smaller systems, since only those tasks effectively required are loaded into the system at any given moment.

Each slave processor runs a microkernel, which supports *multitasking* and *task communication*. The microkernel segments memory in pages, which it allocates for itself (first page) and tasks (subsequent pages). Each Plasma-IP has a *task table*, with the location of local and remote tasks. A simple preemptive scheduling, implemented as a *Round Robin Algorithm*, provides support to multitasking.

To achieve high performance in the processing elements, the Plasma-IP architecture targets the separation between communication and computation. The network interface and DMA modules are responsible for sending and receiving packets, while the Plasma processor performs task computation and wrapper management. The local RAM is a true dual port memory allowing simultaneous processor and DMA accesses, which avoids extra hardware for elements like *mutex* or cycle stealing techniques.

The initial environment to evaluate distributed applications running in HeMPS is named *HeMPS Generator* (Figure 3).




**(a) HeMPS generator main window** **(b) HeMPS debug tool**

**Figure 3: HeMPS environment.**

The *HeMPs Generator* environment allows:

1) Platform Configuration: number of processor connected to the NoC, through parameters *X* and *Y*; the maximum number of simultaneous running tasks per slave; the page size; and the abstraction description level. User may choose among a processor ISS or VHDL description. Both are functionally equivalent, with ISS being faster and VHDL giving more detailed debug data.
2) Insertion of Applications into the System: The left panel of Figure 3(a) shows applications *mpeg* and *communication* inserted into the system, each one with a set of tasks.
3) Defining the initial task mapping: Note in Figure 3 (a) *idtc* task (initial task of *mpeg*) allocated in processor 01, and tasks *taskA* and *taskB* (initial tasks of *communication*) allocated to processors 11 and 02 respectively.
4) Execution of the hardware-software integration: Through the *Generate* button, it fills the memories (microkernel, API and drivers) and the task repository with all task codes.
5) Debug the system: Through the *Debug* button, system evaluation takes place using a commercial RTL simulator, such as ModelSim. The *Debug HeMPS* button calls a graphic debug tool (Figure 3 (b)). This tool contains one panel for each processor. Each panel has tabs, one for each task executing in the corresponding processor (in Figure 3 (b) processors 10 and 01 execute 2 tasks each one). In this way, messages are separated, allowing to the user to visualize the execution results for each task.

# 4. Experimental Setup and Results

This section presents an application executing in the present HeMPS platform. As a result, features like task allocations and execution time can be observed. The application used as benchmark is a partial MPEG codification. There are five tasks, named as A, B, C, D and E, with a sequential dependence. It means that task B only can initiate after task A request B. The HeMPS used for this experimental has size 2x2, where each slave processor has a memory size of 64KB and a page size of 16KB. So, they can allocate at maximum two tasks per slave processor (two pages are reserved for the kernel). It is important to mention that the master do not allocate for himself tasks. Its function is to dynamically allocate tasks and control the communication with the host.

Initially, the application code is written, using C programming language, and five files are created, each one represents a task. Then, using the HeMPS Generator, the HeMPS configuration is set and the codes of the tasks are opened and mapped. The mapping is depicted in the Figure 4, and it shows that this experiment

executes dynamical allocation of the tasks. After that, the software generates all files for the simulation. After simulation, the software uses the debug tool to show results, shown in table 1.

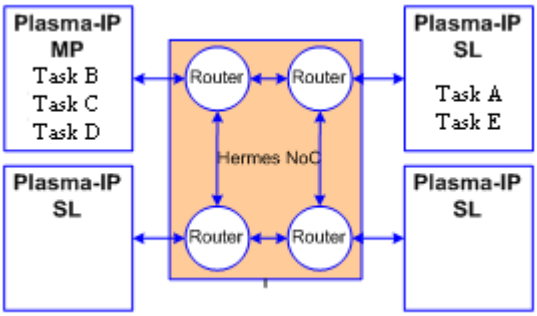



**Figure 4: Experiment Mapping**

| Tasks | Tick_Counter | |
|---|---|---|
| | Start | End |
| A | 3.791 | 39.608 |
| B | 10.762 | 426.678 |
| C | 58.214 | 435.848 |
| D | 79.660 | 442.303 |
| E | 41.939 | 532.715 |

**Table 1: Time, in clock cycles, of the start and end of MPEG application tasks.**

The results show that tasks B/C/D are correctly allocated by the master. It is important to mention that dynamic task allocation overhead is minimal [CAR09], enabling the use of such strategy (dynamic task allocation) in larger MPSoCs.

# 5. Conclusion and Future Works

This paper presented the features of the *HeMPS Station framework*, composed by software executing in a given host, and an MPSoC executing in FPGA, both interconnect through an internet connection. The basis for the HeMPS Station framework is the HeMPS MPSoC, which is open source, available at the GAPH web site (www.inf.pucrs.br/~gaph). Presented results shown the efficiency of dynamic task allocation, a key feature adopted in the HeMPS MPSoC. Present work includes: (*i*) development of the IPs responsible to effectively implement the framework (Ethernet communication and memory controller); (*ii*) integration of such IPs to the HeMPS MPSoC; (*iii*) adapt the present software environment to communicate with the FPGA platform through the internet protocol. Future works include the addition of hardware monitors to collect data related to power, latency and throughput.

# 6. References

[BEN02] Benini, L.; De Micheli, G. “Networks On Chip: A New SoC Paradigm”. Computer, v. 35(1), 2002, pp. 70-78.

[TIL07] Tilera Corporation. “TILE64™ Processor”. Breve descrição do produto. Santa Clara, CA, EUA. Agosto 2007. 2p. Disponível em http://www.tilera.com/pdf/ProBrief_Tile64_Web.pdf

[KIS06] Kistler, M.; Perrone, M.; Petrini, F. “Cell Multiprocessor Communication Network: Built for Speed”. IEEE Micro, Vol 26(3), 2006. pp. 10-23.

[CRIS07] Woszezenki R. C.; “Alocação De Tarefas E Comunicação Entre Tarefas Em MPSoCS”, Dissertação de Mestrado. Março 2007.

[CAR05] Carara, E.; Moraes, F. “MPSoC-H – Implementação e Avaliação de Sistema MPSoC Utilizando a Rede Hermes”. Technical Report Series. FACIN, PUCRS. Dezembro 2005, 43 p.

[CAR09] Carara, Everton Alceu, Oliveira, R. P., Calazans, N. L. V., Moraes, F. G. “HeMPS - A Framework for Noc-Based MPSoC Generation”. In: ISCAS, 2009.

[LIN05] Lin, L.; Wang, C.; Huang, P.; Chou, C.; Jou, J. “Communication-driven task binding for multiprocessor with latency insensitive network-on-chip”. In: ASP-DAC, 2005. pp. 39-44.

[MOR04] Moraes, F.; Calazans, N.; Mello, A.; Mello, A.; Moller, L.; Ost, L. “HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip”. Integration the VLSI Journal, Amsterdam, v. 38(1), 2004, p. 69-93.

[PLA06] PLASMA Processor. Captured on July 2006 at http://www.opencores.org/projects.cgi/web/mips/overview.

[SAI07] Saint-Jean, N.; Sassatelli, G.; Benoit, P.; Torres, L.; Robert, M. “HS-Scale: a Hardware-Software Scalable MP-SOC Architecture for embedded Systems”. In: ISVLSI, 2007, pp 21-28.