HeMPS
A Framework for NoC-Based MPSoC Generation

Markus Schoetz

FAU Erlangen

02.09.2025

Overview I

Overview II

Terminology

HeMPS	Hermes Multiprocessor Systems
IP	Intellectual property
PE	Processing element
RTL	Register-transfer level
NoC	Network-on-Chip
SoC	System-on-Chip
MPSoCs	Multi-Processor Systems-on-Chip
ISS	Instruction set simulator

HeMPS

Introduces ...

- a scalable NoC-based homogeneous MPSoC RISC architecture
- a microkernel which supports a dynamic workload
- design tools (configure MPSoC dimension, initial state, and debug simulation)
- the code for the above on github (open source)

Aim: Flexible architecture with fast design space exploration via SystemC simulation

Design flow

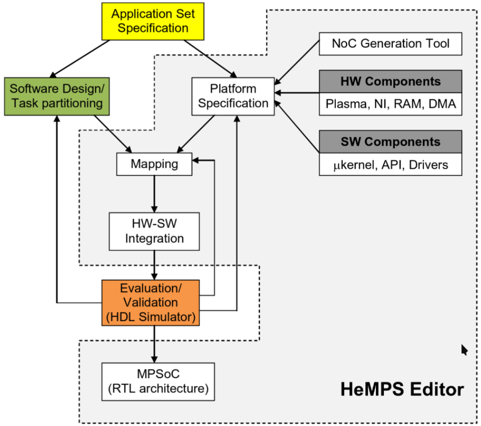


Figure: HeMPS[1] Design Flow.

Architecture

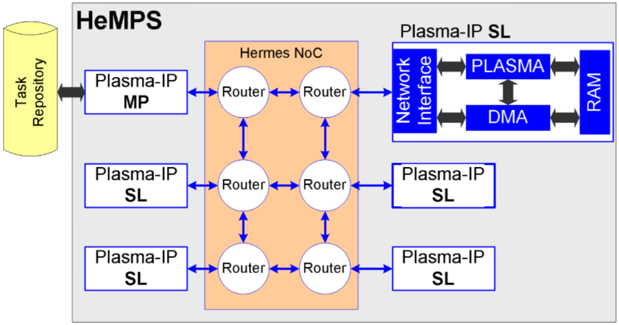


Figure: HeMPS[1] instance using a 2x3 mesh NoC.

Processing element

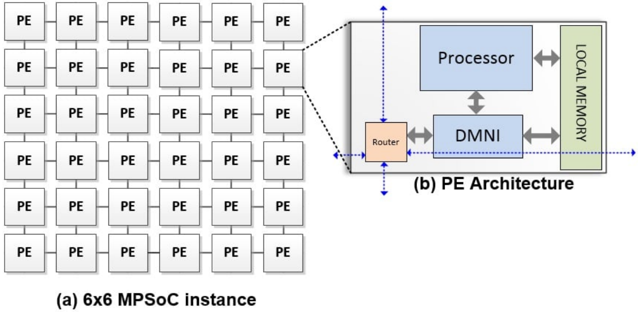


Figure: HeMPS[2] instance using a 6x6 mesh NoC.

NoC

- a 2D mesh topology of routers
- routers have: input buffers, control logic shared by all router ports, an internal crossbar and up to five bi-directional ports
- single round-robin arbitration schedule
- a deterministic distributed XY routing algorithm

Cluster-based organization

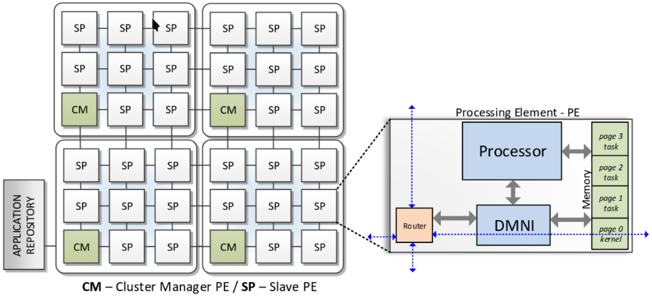


Figure: HeMPS[2] PE organization.

Communication layers

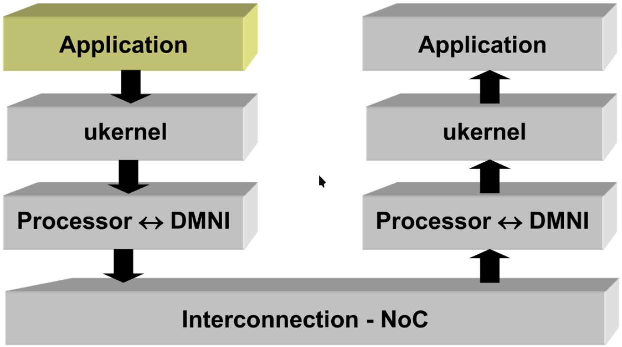


Figure: HeMPS[2] communication layers.

Overview

- small operating system that runs on each processor
- responsible for real-time task scheduling and communication between tasks
- multitask capable
- with inter-task communication primitives
- and support for dynamic workloads
- communication between PEs via message pipe

API

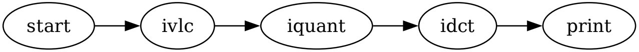
```
Receive(Message *msg, unsigned int target_task_id);
Send(Message *msg, unsigned int source_task_id);
unsigned int GetTick(void);
void Echo(char *string);
void Exit(char *string);

typedef struct {
  int length;
  int msg[MSG_SIZE];
} Message;
```

Application mapping

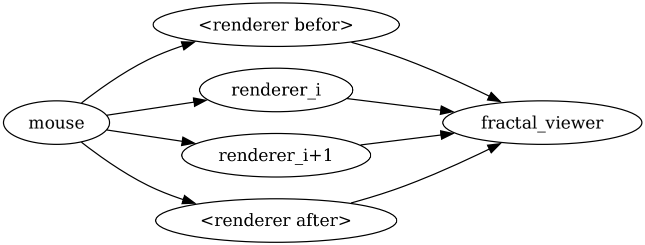
- applications are modeled as a set o communicating tasks (Communicating Task Graph)

MJPEG example:


$$\begin{tikzcd}
\text{start} \arrow[r] & \text{ivlc} \arrow[r] & \text{iquant} \arrow[r] & \text{idct} \arrow[r] & \text{print}
\end{tikzcd}$$

Application mapping

Communicating task graph example for fractal accelerator:



Inter-task communication

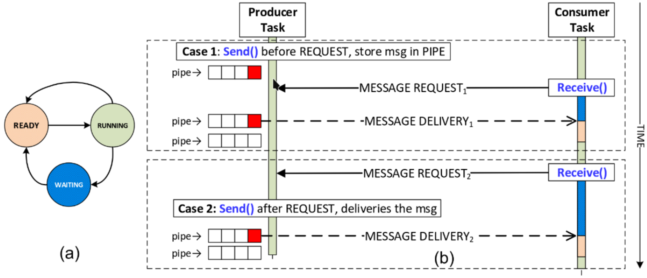


Figure: HeMPS[2] Inter-task communication.

Task synchronization

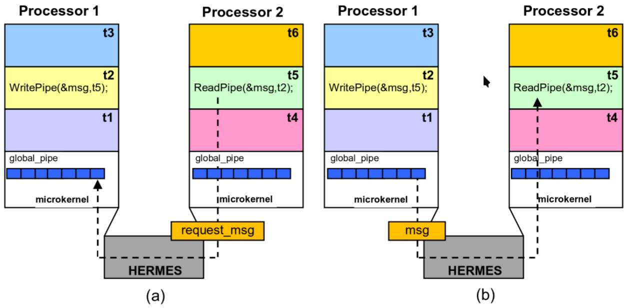
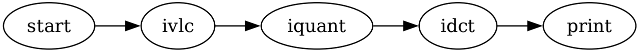


Figure: HeMPS[1] reading of an available message.

Showcase application

Showcase of code[3] for mjpeg



Master processor

Responsible for ...

- task allocation
- broadcasting of control messages such as placement of allocated tasks and release of finished tasks
- reception of control messages, as end of task and debug packets

Design tools

- configure the hardware and applications mapping
- generate clock-cycle accurate simulations in form of
  - synthesizable RTL VHDL
  - SystemC simulation model for processors
- debug and verify the system

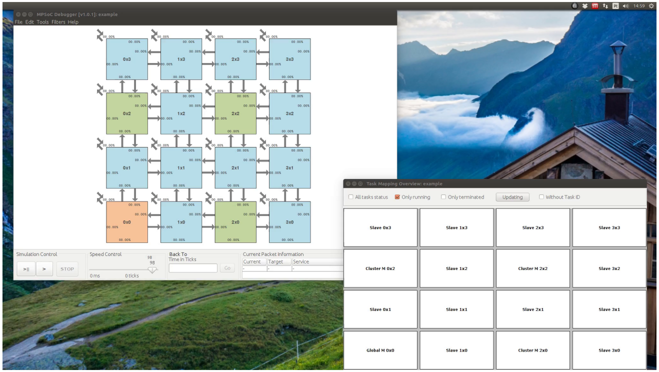
Design tools

Quick look at SystemC module[3] for hemps...

Configuration options

Configured via yaml, lets have a look at a configuration[3]...

Demo



Slave 0x3	Slave 1x3	Slave 2x3	Slave 3x3
Cluster M 0x2	Slave 1x2	Cluster M 2x2	Slave 3x2
Slave 0x1	Slave 1x1	Slave 2x1	Slave 3x1
Global M 0x0	Slave 1x0	Cluster M 2x0	Slave 3x0

Figure: HeMPS[2] Debugging demo.

References I

[1] Everton A. Carara et al. "HeMPS - a framework for NoC-based MPSoC generation". In: *2009 IEEE International Symposium on Circuits and Systems*. 2009, pp. 1345–1348. DOI: 10.1109/ISCAS.2009.5118013.

[2] Fernando Gehm Moraes. *HeMPS Multiprocessor System on Chip*. 2010. URL: https://web.archive.org/web/20240414090143/https://www.inf.pucrs.br/hemps/index.html.

References II

[3] Fernando Gehm Moraes. *HeMPS github source code*. 2019. URL: https://web.archive.org/web/20240414012042/https://github.com/gaph-pucrs/hemps.