

---

## TITLE

**Hardware-Software-Co-Design of Parallel and Distributed Systems Using a unique Behavioural Programming and Multi-Process Model with High-Level Synthesis**

---

## JOURNAL/CONFERENCE

SPIE 2010, Session EMT102 VLSI Circuits and Systems

---

## AUTHOR(S)

Stefan Bosse

---

## ABSTRACT

Embedded Systems used for control and data processing, for example in Cyber-Physical-Systems (CPS), perform the monitoring and control of complex physical processes using applications running on dedicated execution platforms in a resource-constrained manner. Either software-controlled processor or System-on-Chip hardware architectures on RT level are used to provide the execution platform. Required parallelism can only be partially implemented in traditional processor architectures, but complex algorithms can not be implemented on RT level. A compromise exists in traditional hardware-software-co-design, synthesizing multi-processor architectures, and software.

A new design methodology for parallel and distributed systems is presented using the behavioural hardware-software compiler ConPro providing an imperative programming model based on concurrent communicating sequential processes (CSP) with an extensive set of interprocess-communication primitives. The behavioural programming level describes the behaviour of the full de-

sign interacting with the environment using processes and (shared) objects. The programming language and the compiler-based synthesis flow enables the design of constrained power- and resource-aware embedded systems on RT level efficiently mapped to FPGA/ASIC technologies. Concurrency is modelled explicitly on control- and data path level. Additionally, concurrency on data-path level can be automatically explored and optimized by different schedulers.

The CSP programming model can be synthesized to software, too, primarily C and ML. A common source can be used for both hardware and software implementations with identical functional behaviour, providing real and simulated concurrency, respectively. The necessary abstraction of hardware blocks and IPC is performed by using an object-orientated model layer, part of the programming model.

Processes and objects of the entire design can be distributed on different hardware and software platforms, for example, several FPGA components and software executed on

several microprocessors, providing a parallel and distributed system. Intersystem-, inter-process-, and object communication is automatically implemented with serial links, not visible on programming level.

This presented design methodology has the benefit of high modularity, freedom of choice of target technologies, and system architecture. Algorithms can be well matched to and distributed on different suitable execution platforms and implementation technologies, using a unique programming model, providing a balance of concurrency and resource complexity.

An extended case study of a communication protocol used in high-density sensor-actuator-networks should demonstrate the design of a SoC and software implementation.