isis
SENSORIAL MATERIALS

Universität Bremen

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

Stefan Bosse
University of Bremen, Department of Computer Science, Workgroup Robotics, Germany, ISIS Sensorial Materials Scientific Centre, Germany(2)

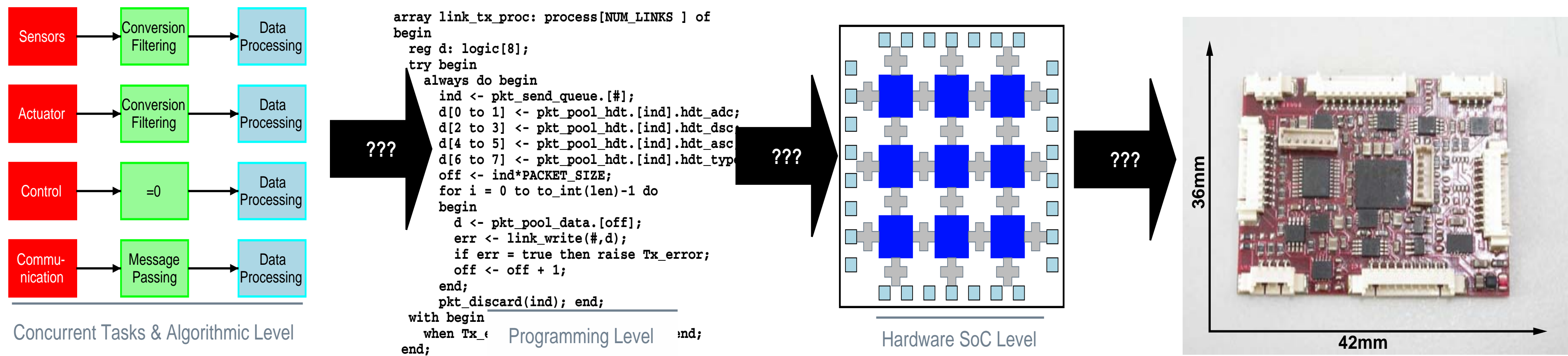
19.4.2011

SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 1 of 14

Overview

Goals and Requirements in Embedded System Design

1. **Design of parallel and distributed embedded systems from algorithmic level**
2. Behaviourial modelling on programming level using a *multi-process model* with interprocess-communication and atomic guarded actions
3. ConPro: *Concurrent programming* and design of parallel hardware and software systems
4. *Abstraction of operational hardware blocks* and access from programming level
5. Transition from parallel to distributed systems: Required communication architecture for distributed systems
6. *Design example*: **SLIP** - a robust and efficient communication protocol stack for multidimensional network topologies (HW/SW implementation)

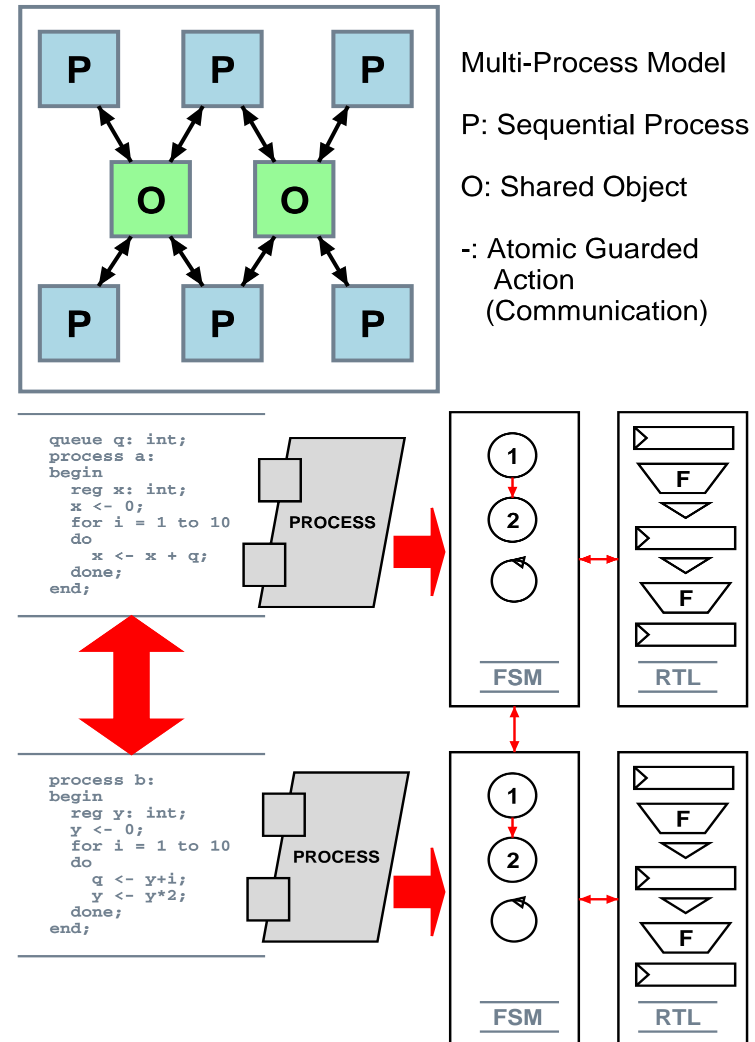


SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 2 of 14

Concurrent Programming with a Multi-Process Model

- Execution Environment: processes executing instructions in sequential (imperative) order ➠ *Finite State Machine*
- Interaction between processes: always using global *shared objects* ➠ *Interprocess-Communication (IPC)*
- Interprocess-Communication = Synchronization: Mutex, Semaphore, ...
- Access of shared resources is serialized: *guarded atomic actions*
- Access of shared resources is managed by a *scheduler*: processes blocked untill resource is available.
- *Hardware Implementation*: Mapping of processes to concurrently executing state machines and RTL
- *Software Implementation*: Mapping of processes to threads (simulated multiprocessing)

Figure 1. Multi-Process Model [mod. CSP/Hoare]



SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 3 of 14

ConPro: Language & Highlevel-Synthesis

Synthesis of massive parallel application specific SoC designs AND parallel software from algorithmic & behavioural programming level

Programming Model

- Communicating Sequential Processes
- Guarded shared objects

Concurrency Model

- *Control path*: concurrently executed processes
- *Data path*: bounded instruction blocks

Synchronization

- Interprocess-Communication ➠ directly implementable in hardware: *Mutex, Semaphore, Event, Timer, Queue, ...*
- Shared objects guarded by mutex scheduler (atomic guarded access)

Execution Model

- Process: strict sequential
- HW: Finite-State-Machine & RTL
- SW: light weighted process/thread

Objects

- Data storage: registers (CREW), variables (RAM,EREW), ...
- Object orientated programming: abstract objects accessed with methods (hardware blocks)

Programming Language

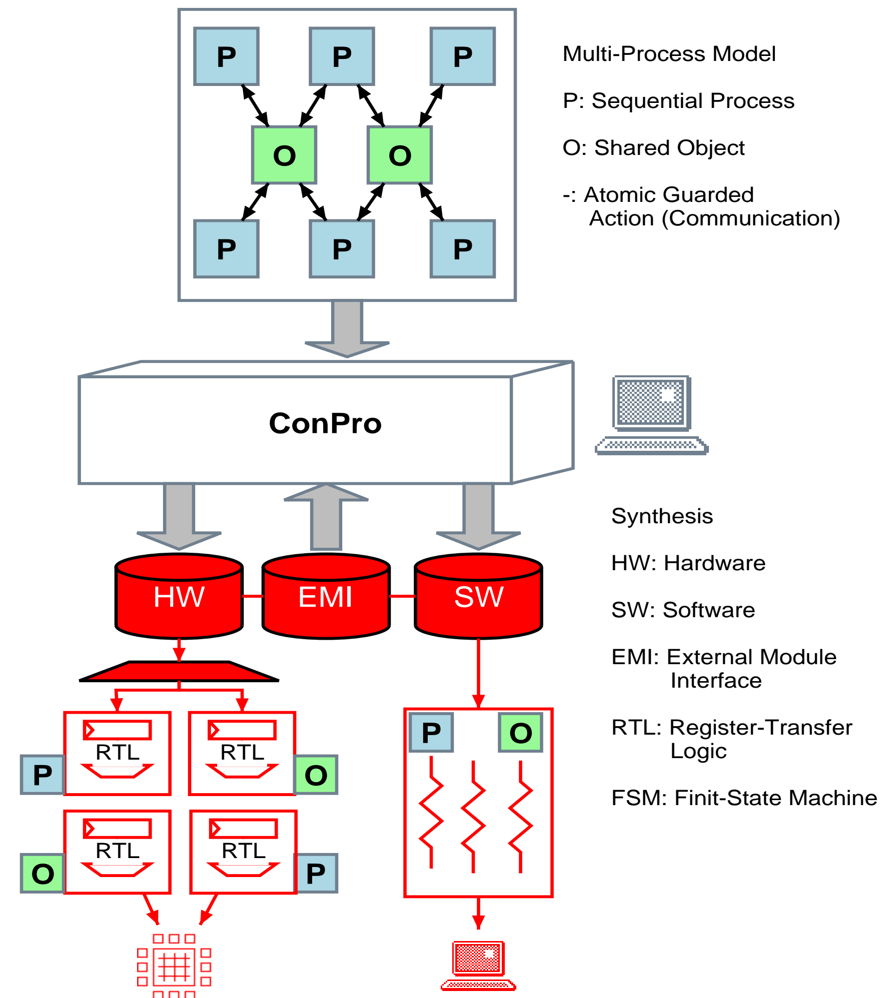
- Imperative with data and control statements
- Explicitly modelled parallelism
- Parameterization on block level: synthesis, scheduling, allocation, object parameters, ...

SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 4 of 14

ConPro Synthesis

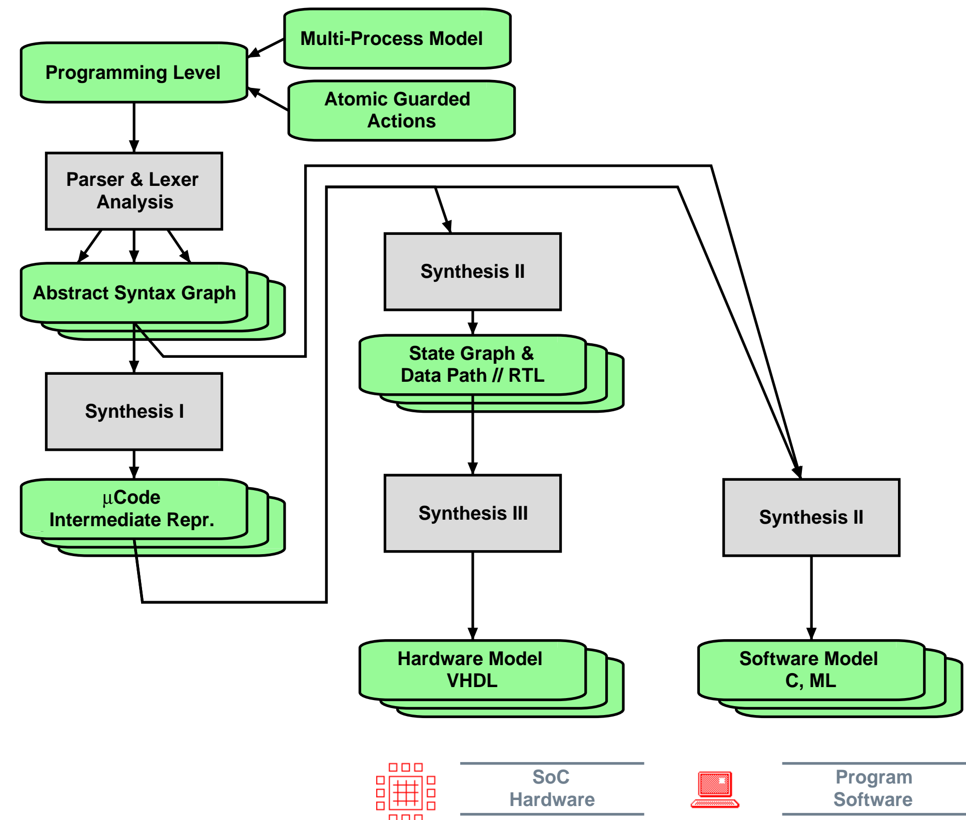
- One synthesis compiler is used for hardware- and software targets.
- *Hardware Implementation*: Mapping of processes to concurrently executing state machines and RTL
- *Software Implementation*: Mapping of processes to threads with *different abstraction levels* (high,mid,low)
- Central part for HW/SW co-design: External Module Interface
- Operational blocks like Interprocess-Communication are modelled with unified abstract objects
- Objects can be implemented directly in hardware or software

Figure 2. ConPro Synthesis with HW/SW targets



SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 5 of 14

- Multi-stage synthesis flow (HW/SW):
  - **I.** Parser, Lexer, Analysis
  - **II.** First Intermediate Representation (IR): Abstract Syntax Tree ($p_i \rightarrow AST_i \; \forall \; p \in P$)
  - **III.** Second IR: Compiling of AST to linear list of µCode using *parameterizable rule sets*
  - **IV.** Third IR: Mapping of µCode to state transition-graphs (STG) and data path: Register-Transfer Level Architecture
  - **V.** HW: Compiling of state transition-graphs to hardware model VHDL
  - **VI.** SW: Compiling of AST or µCode to software model C (imperative) or ML (functional)



SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 6 of 14

ConPro: Abstract Objects and External Module Interface

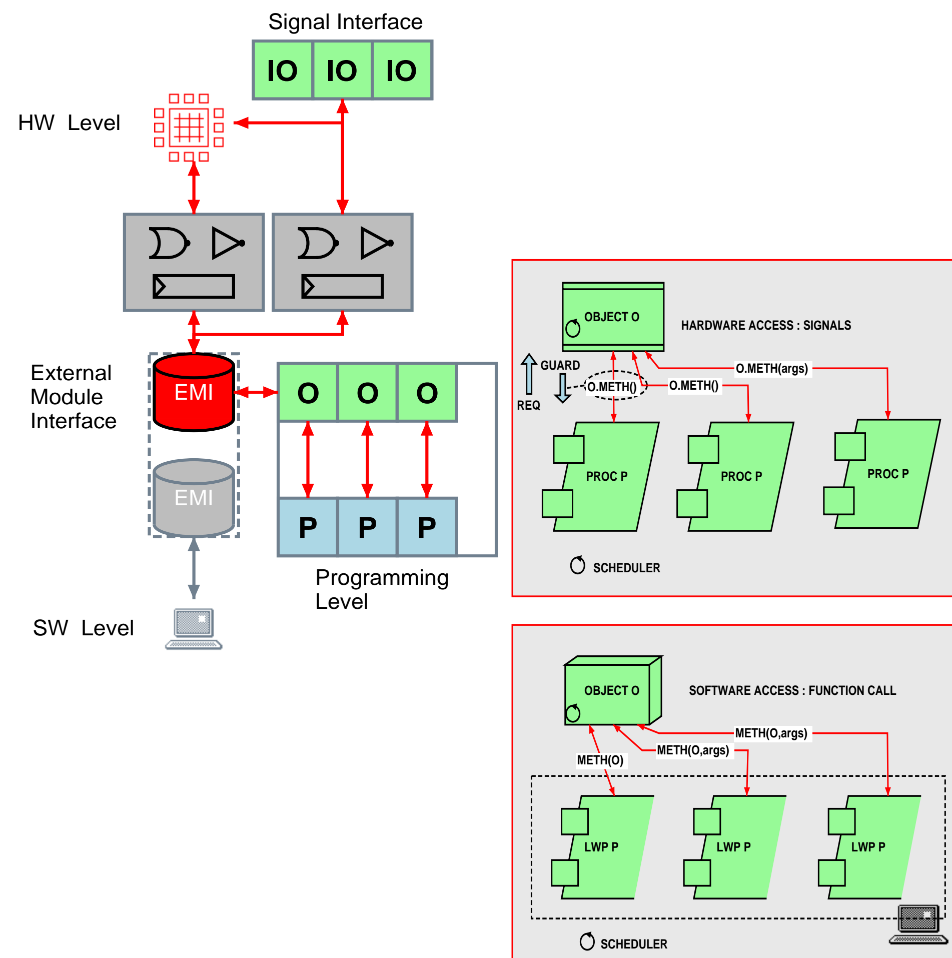
Abstract Objects

- **Abstraction & Interconnect** of hardware blocks to programming level
- A set of *methods* is used to access objects (read,write,control).
- Unified access interface (HW/SW)
- HW: method access is mapped to handshaked hardware signals
- SW: method access is mapped to function calls

External Module Interface EMI

- HW: Objects are modelled on hardware behaviour level (VHDL) and meta language statements (interpreted during synthesis)
- SW: For each object there is a software model, too.

Figure 4. Abstract Objects modelled with the External Module Interface



SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 7 of 14

Distributed Architecture and Communication

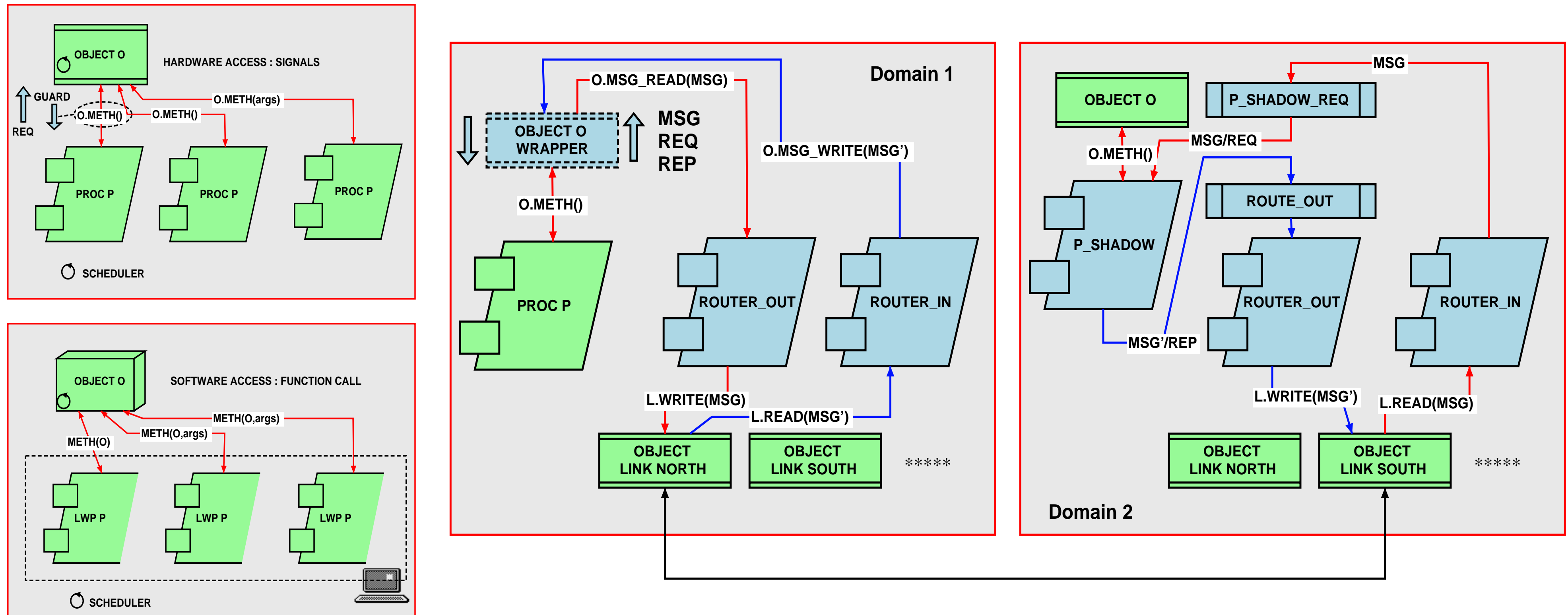
Distributed Architecture

- Processes and objects can be distributed in domains on algorithmic programming level

Communication Architecture

- Object access is mapped to message based communication and domain routing using serial links

Figure 5. Left: local access, Right: global access of objects using message based communication



SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 8 of 14

Communication

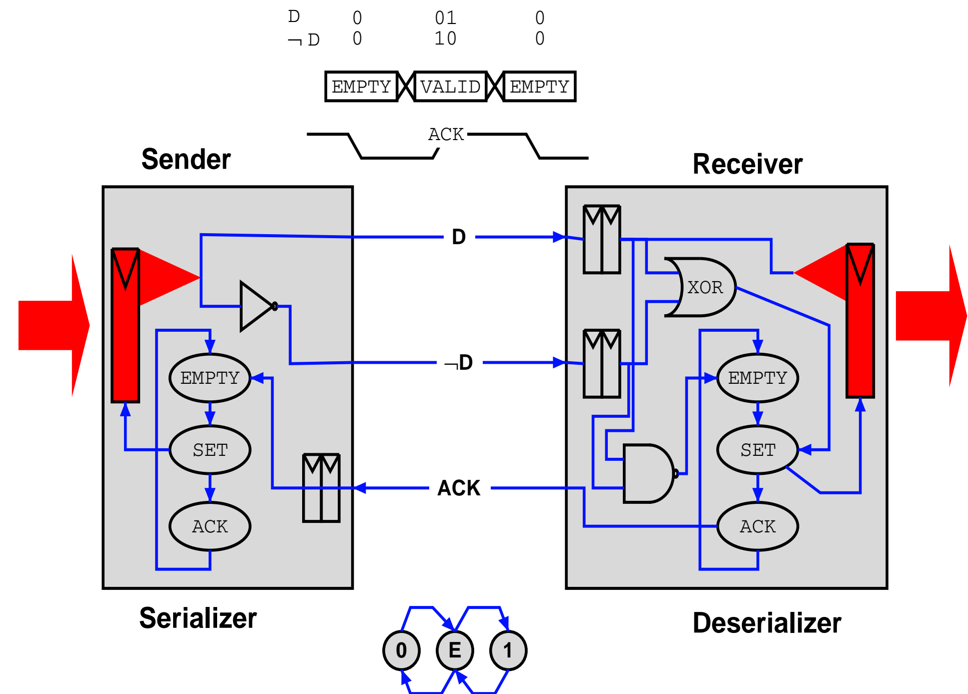
Network Topology

- Each node is connected in a two-dimensional mesh-grid with their direct neighbours
- Each node is a service end-point and a message router, too.

Delay Insensitive Link

- Problem: local: synchronous system, global: asynchronous system
- Solution: asynchronous serial links between nodes (GALS wrapper)
- But asynchronous links requires asynchronous logic or
- Asynchronous link is implemented with synchronous finite state-machine and input signal oversampling
- Dual-rail encoding and four-phase handshaked protocol is used

Figure 6. Asynchronous Link Architecture



SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 9 of 14

Synthesis of Distributed Systems

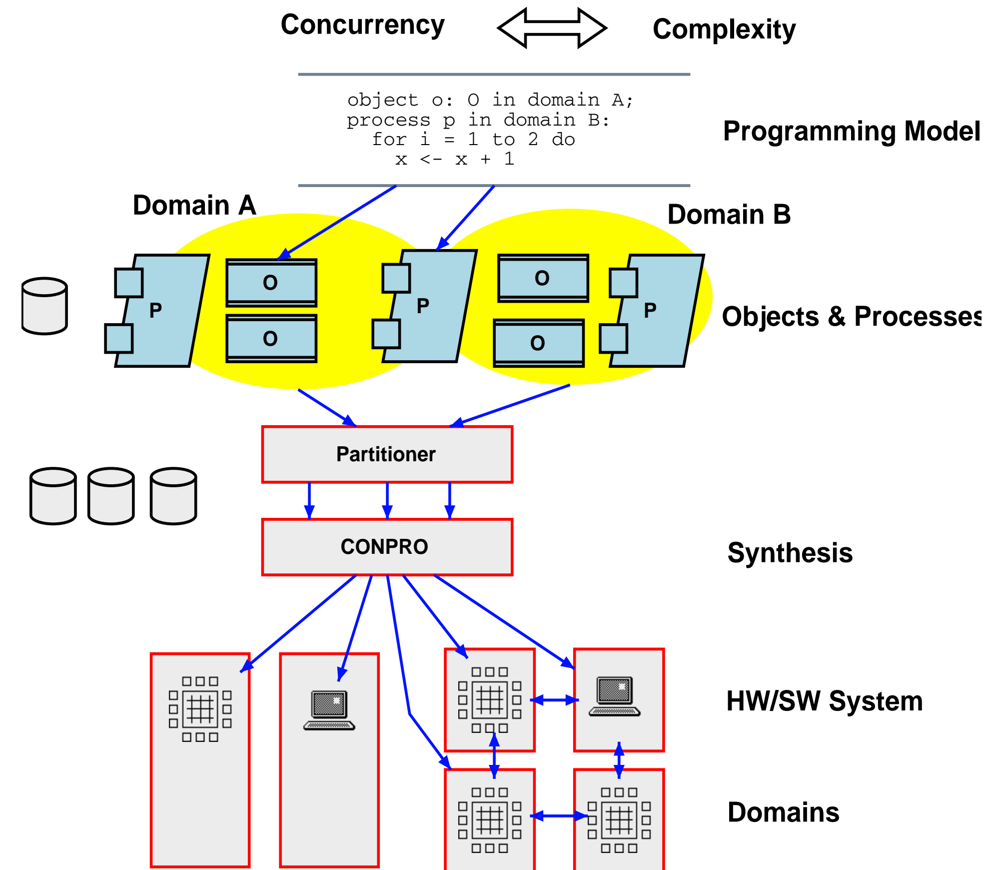
Programming Model

- Concurrently executing communicating processes
- Processes and objects are assigned to domains on programming level
- Communication: always using abstract object access (IPC...)
- Local object access: communication still with handshaked hardware signals (HW) or function calls (SW)
- Remote object access: hidden message based communication

Partitioner

- A common programming source is partitioned into independent domain sources and compiled independently by ConPro to HW/SW targets
- Add communication architecture

Figure 7. Extended Design flow using a partitioner

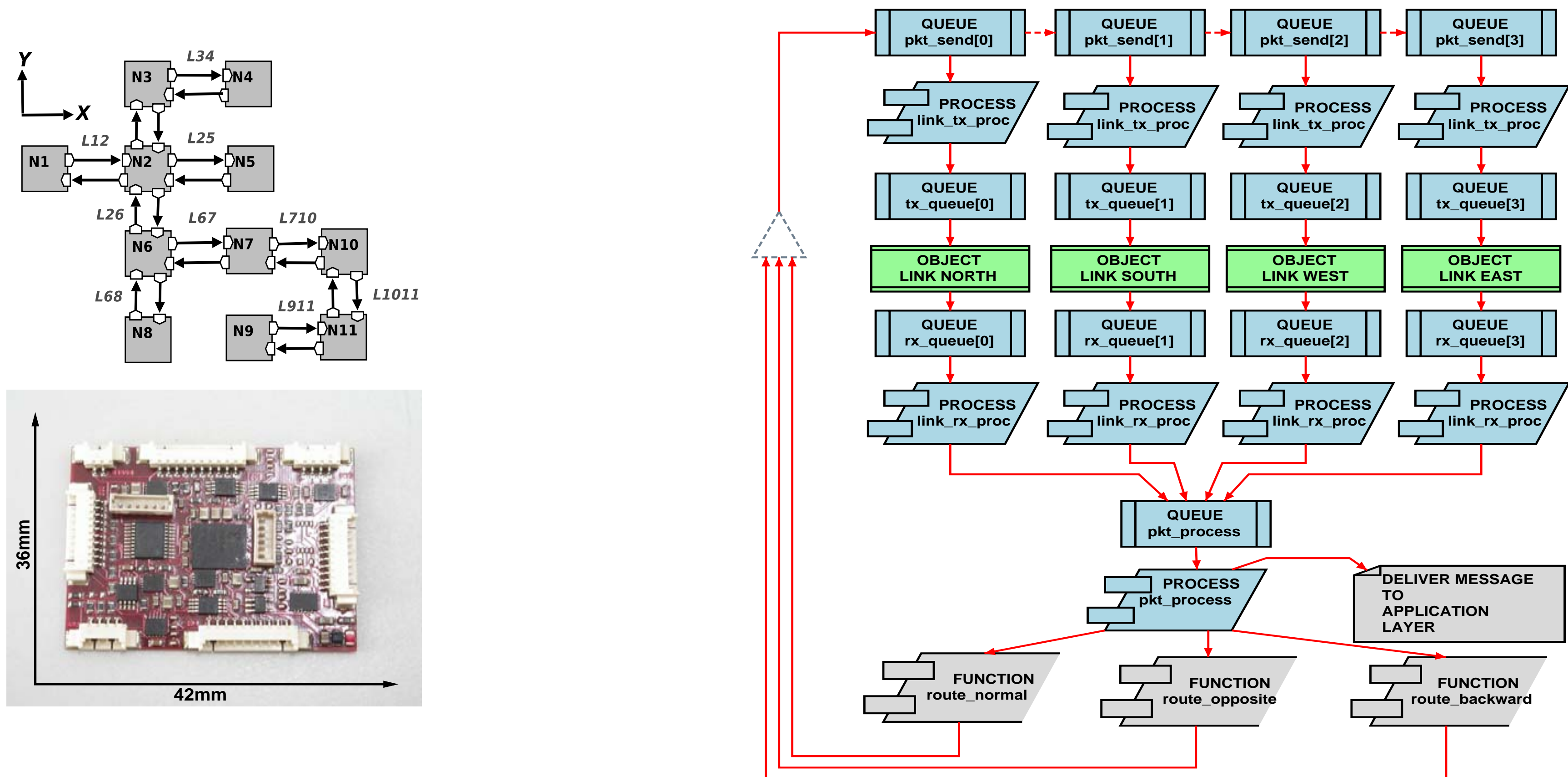


SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 10 of 14

Design Example: SLIP

- Smart and robust communication with Simple Local Intranet Protocol SLIP based on smart delta-routing
- Remote procedure call interface (RPC, application layer)

Figure 8. SLIP network topology (2-dim.) and massive parallel implementation of protocol stack



SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 11 of 14

- Mapping of algorithms and massive parallel data processing to SoC node with high-level synthesis using ConPro: ➠ ❶ low power ❷ miniaturization ❸ low latency✓
- Mapping of same sources to software (C) using ConPro, too: ➠ ❶ interfacing computers ❷ test/simulation✓

Table 1. Characteristics of SLIP implementation (HW: Hardware, SW: Software)

Parameter	Value
HLS source code, ConPro	~ 4000 lines, 34 processes 30 shared objects (16 queues, 2 timers)
HW: synthesized VHDL sources	~ 32000 lines
SW: synthesized C sources	~ 5500 lines
HW: FPGA, Xilinx Spartan III - 1000k	11261/15360 LUT (73 %), 2925 FF
HW: ASIC, standard cell library LSI_10K	~ 244k gates, 15k FF ≅ 2.5mm² | 0.18µm
HW: power consumption (FPGA board)	< 100mW (including analog electr.)
HW: performance benchmark R1*	82 clock cycles
SW: performance benchmark R1*	2305 unit machine instructions

*R1: Sequential part of message routing in SLIP

SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 12 of 14

Summary and Conclusions

Desgin of parallel hardware systems

- Complex SoC hardware devices with concurrency on control- and data path level can be efficiently designed from programming level
- The concurrent multi-process model with interprocess-communication and guarded atomic access of shared resources allows designing of complex parallel systems
- Hardware blocks are abstracted and accessed using a method based object-orientated programming style
- Access of objects is fast and efficient (at least 2 clock cycles)

Design of parallel software systems

- Parallel software can be synthesized using the same synthesis frame work and programming language
- Access of objects is fast and efficient (< 100 machine operations)

Design of distributed systems

- Distributed systems can be synthesized using the same synthesis frame work and programming language with an additional partitioner and message based communication
- Prelimenary results: automated synthesis of distributed systems using a partitioner results in significant increase of complexity, resources, and latency of remote object access (> 100 clock cycles).

SPIE Microtechnologies 2011 Conference, 18.4.-20.4.2011, Prague, Session EMT 102 VLSI Circuits and Systems, Proc. SPIE 8067, 80670G (2011); doi:10.1117/12.888122,Page 13 of 14

isis SENSORIAL MATERIALS

Universität Bremen

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

Stefan Bosse
University of Bremen, Department of Computer Science, Workgroup Robotics, Germany, ISIS Sensorial Materials Scientific Centre, Germany(2)

19.4.2011