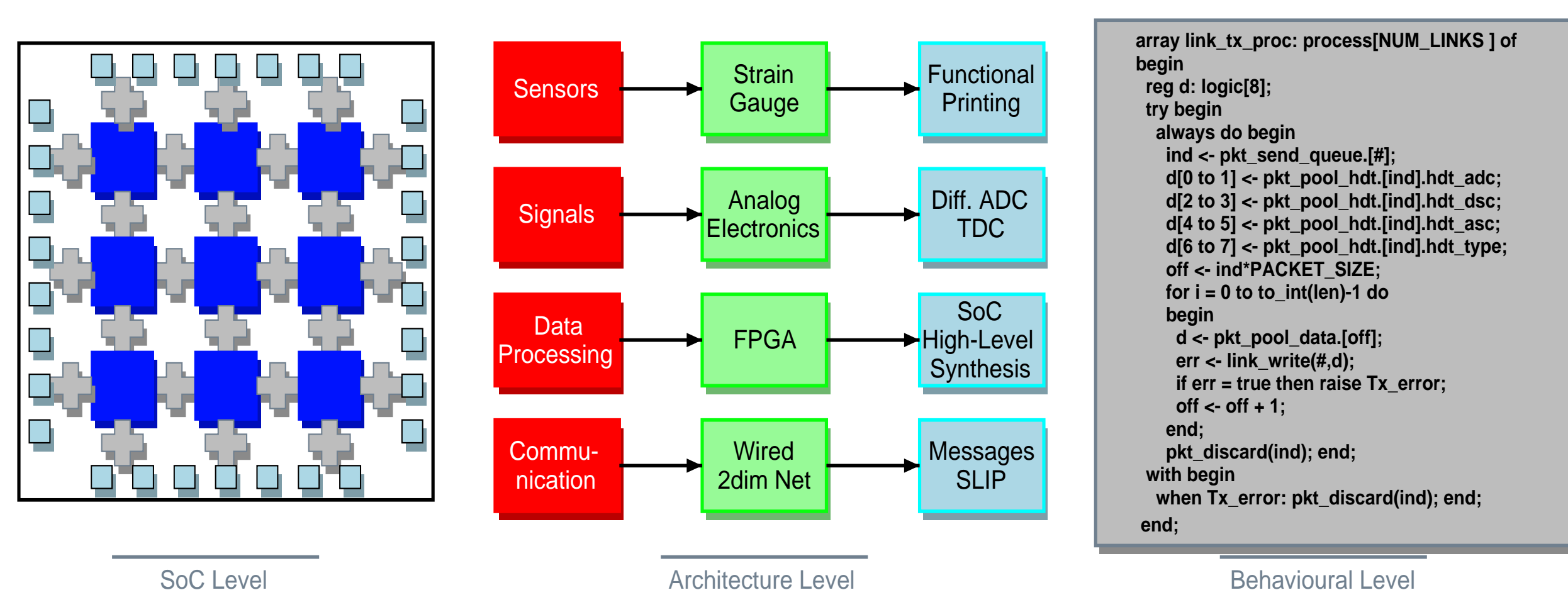


SENSOR NETWORKS AND COMMUNICATION

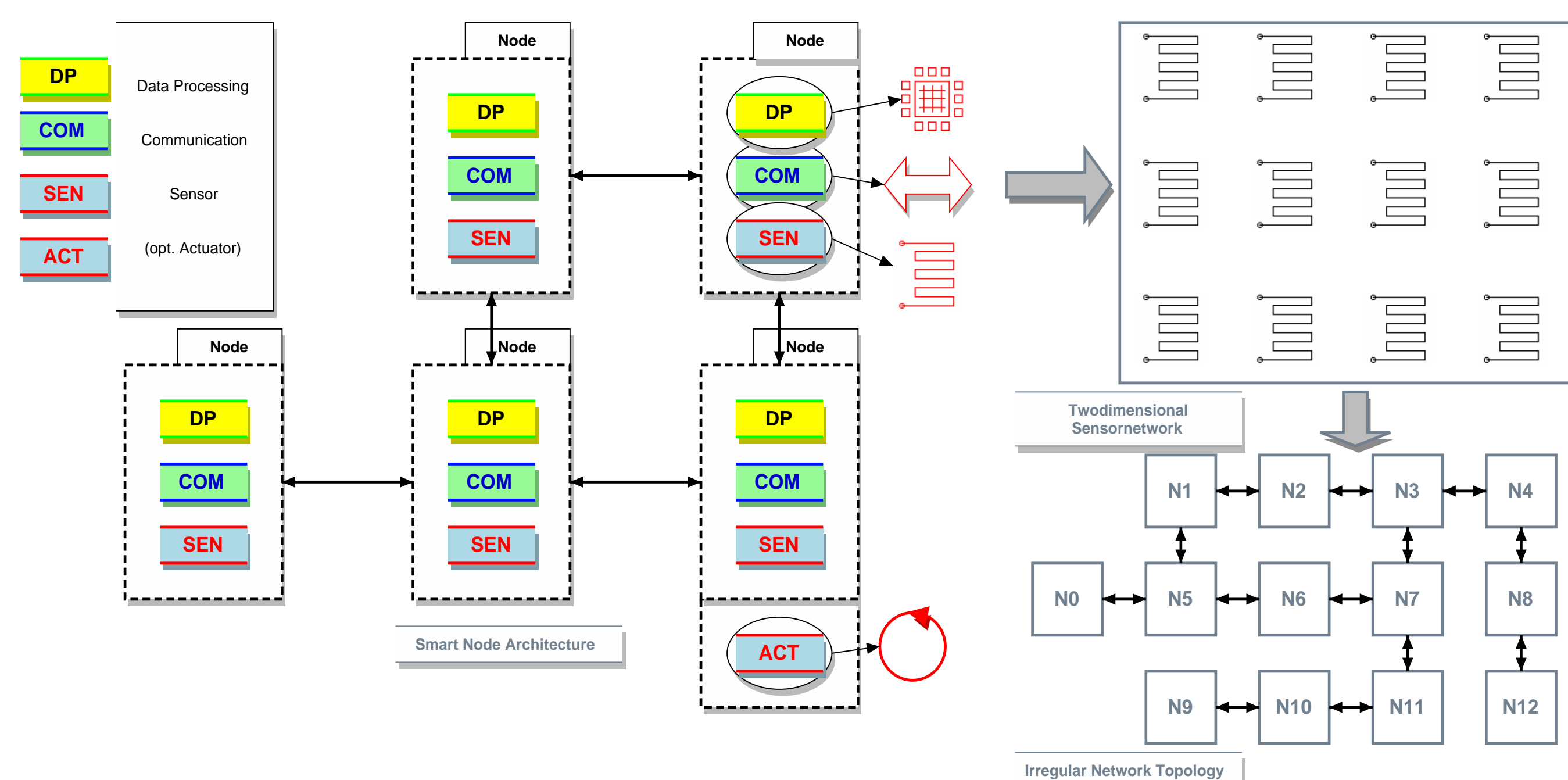
- Communication gains impact with increasing miniaturization and densities in sensor- and actuator networks, especially in the context of robotics and sensorial materials - they require basically wired networks [1].
- The sensor network is a massive parallel computer performing data fusion with smart nodes, requiring a functional system design flow (see Figure 1).

Figure 1. Functional system design of a sensor network requires different technologies from material and computer sciences. On the right side there is a source code fragment of the behavioural concurrent programming language ConPro [2].



- In general, the **network topology** of high-density sensor- and actuator networks is **distributed, irregular and decentralised**, and nodes of such a network have different computing power and storage, **requiring a dedicated protocol design**.
- Development of **Simple Local Intranet Protocol (SLIP)** featuring:
 - message-based point-to-point communication
 - protocol is scalable: network topology, network size, data size
 - no unique node addressing, instead delta distance vector addressing in Cartesian coordinates (1- to n-dimensional)
 - simple and **smart routing strategies**, required for irregular network topologies (for example in Figure 2.)
 - reliability and robustness against link failures
 - available both in software & hardware implementation targeting FPGA/ASIC technologies and SoC-Design
- Development of high-density and miniaturized strain-gauge **sensor networks** consisting of printed sensors [3] with directly coupled smart nodes using decentralized data processing architecture and advanced SoC design methodologies (see Figure 2.)

Figure 2. Sensor network with decentralized data processing architecture. Each smart node consists of data processing, communication, and strain gauge sensors. Data processing is implemented with a SoC design (FPGA), and strain gauge sensors with functional printing technologies [3].



REFERENCES

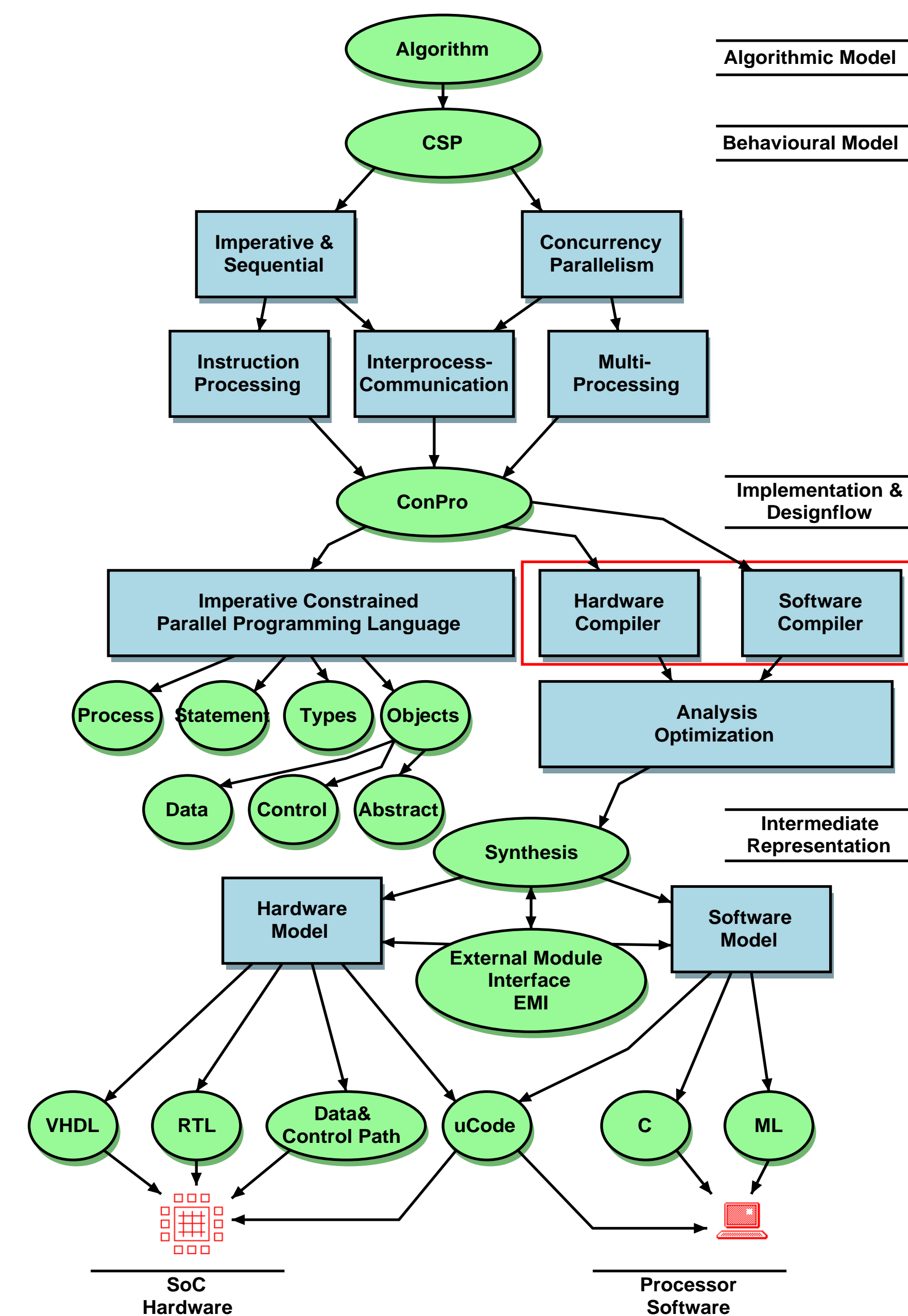
- [1] D. Lehmhus, M. Busse, H.-W. Zoch, W. Lang, S. Bosse, F. Kirchner, "Sensor usage in the transport industry - a review of current concepts and future trends", Euromat 2009, 7-10.9.2009, Glasgow, Session E51
- [2] S. Bosse, "ConPro: Rule-Based Mapping of an Imperative Programming Language to RTL for Higher-Level-Synthesis Using Communicating Sequential Processes", Technical Paper, Bremen, 2009
- [3] V. Zöllmer et al., "Printing with aerosols - A maskless deposition technique allows high definition printing of a variety of functional materials," European Coatings Journal, vol. 7-8, pp. 46 -- 50, 2006.

SYSTEM-ON-CHIP DESIGN USING A BEHAVIOURAL MODEL APPROACH AND HIGH-LEVEL-SYNTHESIS

- Traditionally, there are two different ways to model and implement System-on-Chip designs (SoC) used in highly miniaturized sensor- and actuator networks: ❶ using a structural and/or ❷ using a behavioural model level.
- Parallelism is mostly required to satisfy system latency time, resource, and power constraints.

- ❶ **Structural level:** decomposes a SoC into independent submodules interacting with each other using centralized or distributed networks and communication protocols.
- ❷ **Behavioural level:** describes the functional behaviour of the full design interacting with the environment.

Figure 3. SoC design flow using high-level synthesis framework ConPro. Concurrency is modelled explicitly with a concurrent multi-process model and using interprocess-communication for synchronization. Both SoC hardware and processor software models can be synthesized.



- Development of **High-Level-Synthesis Programming Language and Compiler ConPro** featuring (see Figure 3.) [2]:
 - Constrained compiler-based approach of SoC design from algorithmic programming level to register-transfer-level (RTL).
 - Concurrency** is modelled and implemented using the **Communicating Sequential Processes (CSP, Hoare 1985)** model. It describes concurrency explicitly by a multi-process model, with processes sequentially executing an instruction sequence.
 - Processes interacting with each other (and synchronizing) using different levels of **interprocess-communication (IPC)** primitives like mutual exclusions, semaphores, events, and queues. Global resources \equiv IPC, too.
 - Procedural imperative programming language - **building blocks** are: *Processes, statements, types including true-bit types, objects: data path \oplus control path, and abstract objects providing object orientated programming with hardware blocks using the External Module Interface (EMI).*
 - Programming model can be synthesized to different behavioural models from same sources: Pure **software models** (C, ML, μ Code), **state machine model** (data- & control-path, FSM), RTL, pure **hardware behaviour model** VHDL.