Smart Communication in a Wired Sensor- and Actuator-Network of a Modular Robot Actuator System using a Hop-Protocol with Delta-Routing

Dr. Stefan Bosse[1,2], Dr. Dirk Lehmhus[1,3], University of Bremen, Dept. 3, Workgroup Robotics[2] + Dept. 4[3] & ISIS Scientific Centre[1], Bremen, Germany

1 Introduction and State of the Art

Communication gains impact with increasing minaturization and densities in sensor- and actuator networks, especially in the context of robotics and sensorial materials. System-On-Chip design on Register-Transfer-Logic (RTL) level using FPGA and ASIC technologies enables 1. small sized integration of sensors and data processing units (DPU) scaled down to one-chip designs using ASIC- and MEMS technology, and 2. satisfies low power requirements, required especially for high density sensor networks sourced by environmental energy (energy harvesting).
But using resource and power aware designs constraint algorithm complexity significantly, concerning control, data processing, and communication. Software based communication approaches, like TCP/IP based, can not be implemented entirely in hardware, and they are still too complex. Another important quality of communication in sensor- and actuator networks is reliability and robustness against link failures Thus, communication protocols aligned to low resource and low power designs are required. Most actual work in communication focus on wireless networks [4]. But sensorial materials and highly integrated robotics systems require basically wired networks [3].
The network topology of sensor- and actuator networks is in generally distributed and decentralised, and nodes of such a network have different computing power and storage, resulting in the following constraints for protocol design: 1. message based point-to-point communication, 2. application specific scalable protocol regarding network and data sizes to satisfy A. low power design, and B. reducing computing and storage requirements, 3. no unique node addressing (not usable in high density sensor networks), 4. simple routing strategies, but finally 5. robustness related to alternative path finding.
This article gives an overview of a new developed modular robot actuator embedded in sensor and actuator networks connected to sensorial materials and focus on communication and implementation issues using SoC-design methodologies.

2 ModuACT: a Modular Robot Actuator with Integrated Sensorial Materials

The design of the modular robot actuator is based on previous projects designing bio-inspired multi-legged autonomous (walking) robots [1,2]. In the german scientific research centre ISIS (Integrated Solutions In Sensorial Structure Engineering), research focuses primarily on 1. sensorial materials and sensor networks, 2. new technologies for sensors, integration, signal processing and interconnect, 3.

communication, 4. distributed data processing, and 5. energy management and sources.
The robot actuator consists of a rotational joint drive unit using a brush-less DC motor and a skew-less Harmonic-Drive gear, power, control and data processing electronics using a FPGA, and different sensors (angular position, current, temperature). There are four bidirectional communication links (physical layer: LVDS). Several actuators are connected in a chain building a robot manipulator arm or a leg of a walking robot [2]. The control and data processing system is implemented entirely in digital logic and RTL as a SoC-design using a communicating sequential multi-process programming model and the ConPro high-level synthesis framework [5] on behavioural level (results explained later).
Between and beneath each actuator pair different sensorial materials can be arranged. Actually planned is an array of strain gauge sensors printed on distance elements and connected each to a DPU using functional printing process technology [6]. Each DPU implements data processing and communication using SLIP, explained later. The DPUs are connected to each other using printed wire technology and finally connected to the network together with the actuator nodes.

3 SLIP: a Scalable Local Intranet Network Protocol

The Scalable Local Intranet Protocol (SLIP) and a communication controller design was developed. The goals were: 1. low power design and efficiency, 2. low resource design, SoC and RTL capable, and 3. adaptable to local communication requirements. SLIP is scalable with respect to network size (address size class ASC, ranging from 4 to 16 bit), maximal data payload (data size class DSC, ranging from from 4 to 16 bit length) and the network topology dimension size (address dimension class ADC, ranging from 1 to 4).

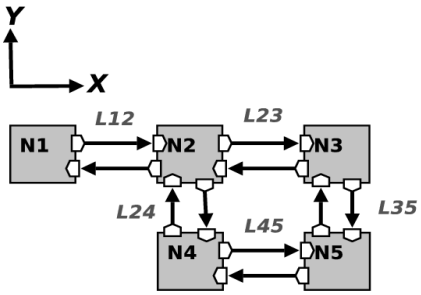

Figure 1: *An example network (ADC=2) with five nodes connected by bidirectional point-to-point links.*

Network nodes are connected using (serial) point-to-point links, and they are arranged along different metric axes of different geometrical dimensions: a one dimensional network (ADC=1) implements chains and rings, a two-dimensional network (ADC=2) can implement mesh grids, a three-dimensional (ADC=3) can implement cubes, and so on. Both regular (complete) and irregular networks (with missing nodes and links) are supported for each dimension.

The main problem in message-based communication is routing and thus addressing of nodes. Absolute and unique addressing of nodes in a high-density sensor-actuator-network is not suitable. An alternative routing strategy is delta-distance routing, used by SLIP. A delta vector Δ specifies the way from the source to a destination node in node hopping count units for each dimension.
An example network (ADC=2) with five nodes connected by bidirectional point-to-point links is shown in figure **1**. Suppose a message should be sent from Node 1 to Node 5. The relative distance is Δ=(2,-1).
A message packet contains a header descriptor specifying the type of the packet and the scalable parameters ASC, DSC and ADC.
A packet descriptor follows the header descriptor, containing: the actual delta-vector Δ, the original delta-vector Δ^1, a backward-propagation vector Γ, a preferred routing direction ω, an application layer π, and the length of the following data part. The total bit length of the packet header ranges from 39 up to 228 bits depending on {ASC,DSC,ADC} scalable parameter tuple setting, which optimises application specific the overhead and energy efficiency (spatial & temporal).
Each time a packet is forwarded (routed) in some direction, the delta-vector is decreased (magnitude) in the respective dimension entry. For example, routing in x-direction results in: $\Delta_1=\Delta_1-1$. A message has reached the destination if Δ=0 and can be delivered to the application layer π.
There are different smart routing rules, applied in order showed below until the packet can be routed (or discarded):

route_normal
This is the main routing rule with the goal to decrease the distance from the actual node to the destination node, finding the shortest path: min|Δ|.
Each i-dimension of the Δ-vector is checked: if $\Delta_i>0$, and there is a link in i-direction, route packet with $\Delta_1=\Delta_1-1$, else if $\Delta_i<0$, and there is a link in -i-direction, route packet with $\Delta_1=\Delta_1+1$, else try next rule. The starting dimension is specified in the packet descriptor ω-field.

route_opposite
Try to send the packet on any other link (but not the direction from which the packet has arrived), resulting in a partial increase of distance. The opposite travel is marked in the Γ-entry.

route_backward
The packet is trapped, no further way to route the packet. Send back the packet on the direction from which it has arrived. The backward travel is marked in the Γ-entry.

For example, the path N1→N5 can be directly routed using normal routing, first x-direction, finally y-direction. Now assume the link L35 is down, and the packet arrives at node N3, but must be send back to node N2, to change the routing direction, and finally arrives N5 over link L45. The above set of smart routing rules solve such backend-trap problems, supporting irregular network topologies and robustness against link failures. The way back from N5→N1 requires opposite routing at node N4!

4 Hardware Implementation and Synthesis

The full SoC-design consisting of the actuator controller, data processing and communication protocol stack (with ADC=2, ASC=8, DSC=8, 4 links, packet pool size=16 packets) was implemented with the behavioural multi-process programming model and the imperative programming language ConPro [5], about 3400 source code lines. The SoC-design is a massive parallel system partitioned in 46 behavioural processes communicating and synchronising over queues (15), guarded and scheduled shared resources, e.g. registers (62) and RAM blocks (21), mutual exclusions, semaphores, events and timers (7). It was synthesised and tested for two target technologies using two different design flows: 1. XilinX-FPGA, Spartan III-1000, Xilinx-ISE synthesis with place & route backends, and 2. standard-cell-library ASIC using the lsi_10k library and the Design Compiler from Synopsys. The second one was only used to get a good area estimation of the SoC-design. The design requires about 99% of the FPGA LUT resources, and about 300k equivalent gates of the ASIC design. The SLIP part requires about 50% of the resources (150k gates). But there is also a small SLIP implementation (without a packet pool and reduced concurrency) only requiring about 20k gates, best suited for small sensor nodes sourced by low power energy harvesting technologies.

5 Conclusion and Summary

The design and implementation of a scalable communication protocol and stack in a SoC design with smart routing enables the design of large scale integrated and high density sensor- and actuator networks, while reducing resource and energy requirements.

Bibliography

[1] M. Eich, F. Grimminger, S. Bosse, D. Spenneberg, F. Kirchner
ASGUARD: A Hybrid Legged Wheel Security and SAR-Robot Using Bio- Inspired Locomotion for Rough Terrain. In IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance, Benicassim, Spain, January 7-8/2008.

[2] J. Hilljegerdes, P. Kampmann, S. Bosse, and F. Kirchner
Development of an Intelligent Joint Actuator Prototype for Climbing and Walking Robots
12th CLAWAR, 09-11 September 2009, Istanbul, Turkey

[3] A. Roenna, T. Kersche, M. Ziegenmeyer, J.M. Zoellner, R. Dillmann
Six-legged walking in rough terrain based on foot point planning.
12th CLAWAR, 2009, p. 591-598

[4] K. Römer, F. Mattern
The design space of wireless sensor networks.
IEEE Wireless Communications 11 (2004), Dezember, Nr. 6, p. 54-61

[5] S. Bosse
ConPro: Rule-Based Mapping of an Imperative Programming Language to RTL for Higher-Level-Synthesis Using Communicating Sequential Processes
Technical Paper, BSSLAB, Bremen, 2009

[6] 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