

On Distributed Reconfigurable Systems: Open Problems and Some Initial Solutions

Apostolos Dollas, Dionissios Efstathiou, Georgios Vernardos,
Elias Polytarchos, Konstantinos Kazakos

*Department of Electronic and Computer Engineering
Technical University of Crete
Chania, 73100, Greece
Contact e-mail: dollas@mhl.tuc.gr*

1. The Driving Problem(s)

By “distributed reconfigurable systems” we mean systems of distributed resources, which include dynamically reconfigurable FPGA components. There already exist systems such as wearable computers with distributed reconfigurable resources [2], but the broader area is rather unexplored. In sensor networks we already have the notion of nodes getting added to or deleted from a sensor network, re-routing of information among sensors based on live/dead sensor status, a communication medium (generally wireless), and communication protocols [4]. It is a matter of time until the sensor network community will desire to take advantage of reconfigurable computing as a means to reconfigure or partially reconfigure individual nodes in order to alter system behavior, or circumvent non-fatal errors through partial reconfiguration. In cooperating robotic systems [5], many of the functions migrate to reconfigurable resources (possibly in combination with general-purpose computers) so that control algorithms will be directly mapped onto hardware for higher speed and efficiency. In biologically inspired systems we already have distributed processes mapped on reconfigurable resources. It is a natural extension to develop “communities” of interacting elements made of reconfigurable resources. Many more examples can be found but the trend is quite evident already.

Reconfigurable distributed systems have some unique characteristics or desired behavior, which merit further study. To illustrate, any attempt for a full implementation of TCP/IP in hardware leads to prohibitively expensive solutions for low-cost sensor networks. Therefore, lighter protocols are needed. However, whereas data exchange can be effectively done through unreliable communication channels, reconfiguration assumes that (at least at the protocol level) a reliable data stream has to be provided for purposes of reconfiguration. Whereas a Bit Error Rate (BER) of 10^{-5} due to noise, etc. can pose no problem in data communication, it would effectively lead to inability to reconfigure just about any present day reconfigurable device, unless accounted for at the protocol level. And yet we just dismissed TCP/IP as a protocol, although it would have provided for the desired robustness.

2. Problems and Challenges in Distributed Reconfigurable Systems

Flexibility: A distributed system has to cope with highly dynamic tasks. Each node must have the ability to perform a variety of services. The dynamic transition between these tasks is determined either by the needs of the entire system or by a

centralised scheduler. It is more efficient to use small-range microcontrollers for control and communication and to execute data processing in reconfigurable hardware [1].

Network Communication: In an *ad-hoc* network with dozens of separate nodes, each performing different tasks, with data and configuration information transferred through the network, we have these challenges:

- 1) A unified protocol assuring secure and compatible communication is imperative.
- 2) A flexible protocol with very large addresses will waste potentially valuable channel bandwidth in small scale systems whereas a small addressing capacity may make system expandability difficult.

Data and Configuration Transmission through a Common Medium: There needs to be a protocol that supports multi-vendor configuration file transfers, *plus* there needs to be a node which communicates through a common medium on one side and through vendor-specific programming ports on the other side (e.g. a wrapper).

Reliability, Testability and Fault Tolerance: Whereas a large body of knowledge exists in this area, the specific characteristics of distributed reconfigurable systems mean that the theory needs appropriate extensions. To illustrate, any notion of checkpointing in a distributed system in which a node may reconfigure itself without knowledge of its peers means that (at the very least) a global notion of time is available, and yet this may be an expensive feature to implement in a sensor network, and thus undesirable.

Some other problems can be the **Energy efficiency** of a node, which is closely related to its computational task [1,2] and the **Communication channel capacity demand** of the system, which depends on the level of data processing and control in the nodes [2].

3. A Case Study

The key features that a distributed node must have in order to cope with current trends are: reconfigurability, and flexibility. By taking into account the challenges mentioned earlier, the close coupling of hardware and software in a communication – demanding environment is a key factor to the development of such a system.

In a multi-FPGA node the inner communication between the FPGAs of the specific node is accomplished in a fixed manner. The main issue is the communication between the different nodes. There must be a medium between the wireless link and the reconfigurable section of the system. One solution

would be to partition each node into three inner-communication layers. Each layer only communicates with its adjacent layer. Layer 1 “talks” only with layer 2, while layer 2 communicates with both other layers. Layer 2 is the communication medium. It collects the incoming data from the network and, according to their use, forwards them to the 3rd layer or keeps them for its own use. It also collects processed data from layer 3 and forwards them to layer 1 (wireless link). Layer 3 (reconfigurable section) communicates in a fixed way with the middle layer. The “line of sight” of the 3rd layer is the medium layer (layer 2) as depicted in the Figure 1. It receives requests only from it and responds accordingly. This actually brings some level of transparency between the different layers of the node, thus ensuring a reasonably low cost and reasonably robust communication. Even if the FPGA logic is dynamically reconfigured, the communication protocol remains unaffected since the 2nd layer contains the necessary information for re-establishing the communication of the node with the network.

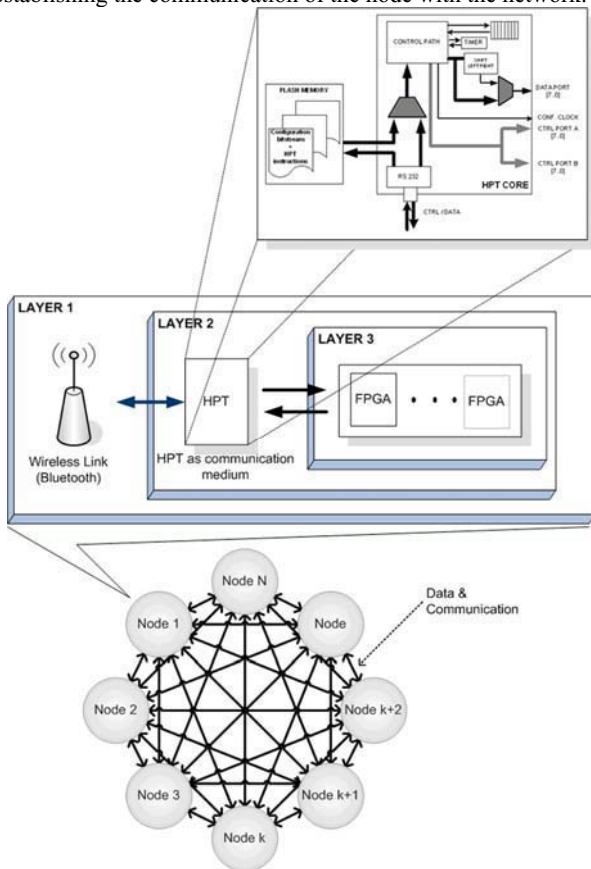


Figure1: Layers of transparency in a node of a distributed reconfigurable system – HPT architecture – Node Communication

In our system [5,6], which functionally resembles the detailed node in Figure1, the host microcontroller has a twofold role. It acts as (a) a centralized source for run-time re/configuration of the reconfigurable processing section, (b) an intermediate between the reconfigurable logic and the communication media. The Hardware Programmer & Tester, developed in our lab is capable of performing these tasks. The Hardware Programmer & Tester (HPT) is a low-cost universal vendor independent FPGA programmer and tester [3,5]. It can

support any known method of FPGA (or CPLD) configuration. The HPT can be integrated in a reconfigurable network node as the host microcontroller, providing run – time re/configuration. In terms of hardware, it is no more complex or costly than a low-cost 8-bit microcontroller. The HPT was originally developed as a download and testing apparatus of the Reconfigurable Run – Time Environment (RTE) **ReRun**[3] developed also in our lab. **ReRun**'s driving force is PTL (Programming and Testing Language) [3], a formal language used for writing scripts to describe the configuration or testing process. PTL has a BNF grammar and well-defined semantics. Thus, the behavior of the system as a whole is unambiguous. As an in–system host processor the HPT is connected to a FLASH memory which stores the configuration bitstreams necessary for the FPGA(s) reconfiguration, and the appropriate PTL scripts [3]. The HPT acts as the medium of layer 2, redirecting data between the reconfigurable processing section of the node and the outer network and maintaining the communication infrastructure of the system.

The BluMiu module acts as the wireless link of layer 1 and processes all the layers of the Bluetooth protocol stack, thus disengaging the HPT from implementing the protocol. The communication protocol that was designed is a lightweight packet switched protocol that enables a variable packet size of up to 64K bytes and supports a large number of target devices. When two BluMiu devices get connected, the external devices can immediately start to exchange packets in a connectionless manner. In its present version, the BluMiu project has the following characteristics:

- 1) Reliable data communication.
- 2) File transfer between 2 computers connected on BluMiu devices.
- 3) Support for point-to-multipoint connections including data transfer to every connected device.
- 4) Remote initialization of the HPT device, through the BluMiu connectivity, towards the wireless FPGA programming. Experiments were conducted and it programmed a FPGA.

A cooperating robot experiment is underway to test this approach in developing distributed, reconfigurable systems

4. References

- [1] Manfred Glesner, Thomas Hollstein, Leandro Soares Indrusiak, *et al.*, “Reconfigurable platforms for ubiquitous computing”, Conf. Computing Frontiers 2004: 377-389.
- [2] Christian Plessl, Rolf Enzler, Herbert Walder, *et al.*, “Reconfigurable Hardware in Wearable Computing”, Nodes. ISWC 2002: 215-222.
- [3] Apostolos Dollas, Dionissios Efstathiou, Thomas Kyriakides: “A Universal Low Cost Run-Time and Programming Environment for Reconfigurable Computing”, Proceedings, IEEE International Workshop on Rapid System Prototyping, 2003: pp. 2-8.
- [4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, “Wireless sensor networks: a survey”, Computer Networks 38, Elsevier, 2002, pp 393-422.
- [5] Paul E. Rybski, Sascha A.Stoeter, Michael D.Erickson, *et al.*, “A Team of Robotic Agents for Surveillance”, Proceedings of the 4th International Conference on Autonomous Agents, 2002, pp 9-16.