# Reconfigurable Processing Units vs. Reconfigurable Interconnects

*A. Herkersdorf, C. Claus, M. Meitinger, R. Ohlendorf, T. Wild*

*Abstract: In this paper we discuss different aspects of system reconfiguration and their relation to the specific requirements from the application domain. Two projects – one from the video processing the other one from the IP networking domain – are introduced that make different use of runtime reconfiguration by either changing individual processing units or the logical interconnect structure of the system. We demonstrate that the requirements of the particular application domain are decisive concerning the design quality and performance of the reconfigurable system architectures.*

With the availability of sophisticated FPGAs ([1], [2]) that allow reprogramming parts of the HW resources while the rest is operational, runtime reconfiguration has become an increasingly relevant research topic ([3], [4]). This approach enables to dynamically provide different functionalities comparable to SW-programmable architectures, however with HW-level performance. It allows a much broader adaptation of system architectures to varying processing requirements during system runtime, especially when resources are limited. Reconfiguration concepts mostly deal with the reconfiguration of processing units by reprogramming HW resources on FPGA platforms. However, as we will show with the example from the network processing domain, specific requirements prohibit processor unit type reconfiguration in this application, while it nevertheless can benefit from reconfiguration of logical interconnects.

We are working on reconfigurable system architectures in the focus program “Reconfigurable Computing”, which is funded by the German Research Foundation ([5]). The Autovision project develops a processing architecture for an automotive driver assistance application ([6]). Key elements of the corresponding system architecture are different HW accelerator modules (coprocessors) that are optimized for the recognition of traffic participants in different driving environments. As these modules are not needed simultaneously, it is intended to exchange them during runtime of the system according to the particular driving situations. The HW reconfiguration is achieved by updating the internal configuration memory of the underlying FPGA platform. I.e. the physical circuit structure defined by CLBs, routing resources and BRAM content is changed. Compared to a solution where all coprocessors are implemented in parallel and are permanently available in the system architecture, the main focus of Autovision is to utilize partial dynamic reconfiguration to reduce the consumption of resources such as CLBs, BRAMs, etc. The available reconfiguration time is mainly determined by the frame rate of 25 fps and the maximum processing latency of the coprocessors if no unprocessed frames might be tolerable. Hence, timing requirements are in the range of several milliseconds in this application.

The FlexPath project uses reconfigurability to increase network processor performance ([7]). In this case the number and physical interconnection of architecture resources (CPU cores, packet processing accelerators, memory blocks) are fixed, but the logical interconnect structure is modified at system runtime. Precisely, the processing paths, i.e. the sequence of architecture blocks to be traversed for the processing of packets that belong to specific traffic flows, are reconfigured. The reconfiguration actually consists in modifying memory contents of the rule base that is used for the path decision. The objective is to guide the packets through the system with minimum resource usage of the internal communication infrastructure and especially of the embedded processors. Thus, higher performance than passing all packets by default through the processor cluster shall be achieved. In FlexPath, reconfiguration is heavily constrained by the packet interarrival time of tens to hundreds of nanoseconds (Gbps links, minimum sized packets) and the fact that packet losses are not acceptable.

Looking at these very diverse approaches, it is necessary to explain in more detail what is actually meant when talking about runtime reconfiguration of systems. The basic commonality is that suchlike systems are kept fully operational during reconfiguration phases. However, fundamental differences exist concerning the abstraction level of the reconfiguration, the timing behavior, and the underlying HW platform.

The abstraction level is mainly related with the resources that are involved in the reconfiguration. At one end of the spectrum, a system may be modified by altering the circuitry that determines functionality or connectivity on physical level. This type of reconfiguration is only possible on FPGA platforms that allow modification of the HW resources at runtime. In contrast to this low level reconfiguration, systems may also be reconfigured on a logical level. In this case the HW structure of the system is unchanged, but the usage of the resources is altered by modifying the rule base (mainly memory contents) that is used for guiding data through the system. In essence, in both variants memory contents are exchanged, however, with a fundamentally different impact: In one case the memory content determines the status of transistors and the functionality of logic gates, in the other case a fixed HW functionality on application level interprets the contents of the memory according to a certain convention, resulting e.g. in a modified usage of the system architecture. The latter type of reconfiguration is feasible on both FPGA and ASIC platforms.

Timing requirements of the application are decisive for the applicability of a reconfiguration approach of the underlying architecture. The reconfiguration time, i.e. the time period needed for carrying out the reconfiguration process, is a key factor in this context, as it makes up the duration for which the reconfigured part of the system is not operational. In any case, it has to be guaranteed that the system processes the requests correctly and in a consistent way. Therefore, only idle times of modules may be used for their reconfiguration. Thus, the maximum processing time for a request plus reconfiguration time always have to be less than the interarrival time of requests.

The reconfiguration time of parts of an FPGA is determined by the throughput of the programming interface and the size of the reconfigured area that in turn determines the amount of the associated programming information (partial bitstream) to be written into the configuration memory. Reconfiguration times around 1 – 100 milliseconds are common. However, there are also theoretical boundaries that determine the shortest reconfiguration time. Considering an input data width of 8 bit to access the configuration memory, the maximum theoretical throughput is 1 Byte per clock cycle. If the clock is set to 100 MHz, the maximum throughput is 100 kBytes per millisecond. Even tiny partial bitstreams (e.g. 10 kBytes) that can theoretically reconfigure a small fraction of the device would already consume 100 microseconds.

These times are by far not acceptable in applications like the FlexPath network processor, where reconfiguration has to be completed in a time frame much shorter than reconfiguration times at the physical level. Therefore, an approach is followed that encompasses only reconfiguration of the logical interconnect and no reconfiguration on the physical level, neither of the functionality nor of the interconnect circuitry. This is feasible by update mechanisms of memory contents that allow an atomic activation of the new configuration. Nevertheless, it has to be mentioned that functional reconfiguration is feasible also in this type of application if a configuration update is not required in the time frame of packet interarrival times. An example could be the migration of a specific functionality like de- or encryption between SW and a dedicated coprocessor ([8]). In such a case, however, it has to be guaranteed that the functionality has to be available permanently and that the arriving packets are always assigned unambiguously to the correct resource that provides the function.

The major difference between the two types of runtime reconfiguration used in the applications and projects mentioned above are summarized in Table 1.

| **Characteristics** | **Reconfiguration of processing inits (typical video applications)** | **Reconfiguration of interconnect (Gbps network processor application)** |
|---|---|---|
| Reconfiguration time | ~10-100 ms | ~200 ns |
| Real-time requirements | More relaxed, circuit can be (partially) stopped during reconfiguration | Very hard, circuit must remain fully operational during reconfiguration |
| Generator of reconfiguration information | Pre-calculated bit streams, online modified bit streams | Online control point SW |
| What is reconfigured | HW circuit: CLBs, routing resources, BRAM content (configuration memory) | Memory (lookup) contents |
| Possible HW platforms | FPGA | FPGA, ASIC |
| Main focus | Reduce consumption of chip resources | Performance |
| Initiator of reconfiguration | Control Plane Software | |

*Table 1: Major differences between Reconfiguration of function units and Interconnects*

Constrained by the different application specific (timing) requirements, different reconfiguration processes, either reconfiguration of processing units or reconfiguration of logical interconnects, are advantageous in a particular system architecture.

[1] Xilinx, Inc. “Virtex-II Pro and Virtex-II Pro X Platform FPGAs, Complete Datashet”, downloadable from http://www.xilinx.com/bvdocs/publications/ds083.pdf

[2] Xilinx, Inc. “Virtex-4 Family Overview”, downloadable from http://www.xilinx.com/bvdocs/publications/ds112.pdf

[3] K. Compton, S. Hauck, “Reconfigurable Computing: A Survey of Systems and Software“, ACM Computing Surveys, Vol. 34, No. 2, June 2002,

[4] J. Becker, M. Hübner, K. Paulsson, A. Thomas, “Dynamic Reconfiguration On-Demand: Real-time Adaptivity in Next Generation Microelectronics”, ReCoSoc2005, Montpellier, France, 2005

[5] DFG Schwerpunktprogramm “Rekonfigurierbare Rechensysteme”, http://www12.informatik.uni-erlangen.de/spprr/home.php

[6] W. Stechele, “Video Processing using Reconfigurable Hardware Acceleration for Driver Assistance”, DATE 2006, Future Trends in Automotive Electronics and Tool Integration Workshop, Munich, Germany, 2006

[7] R. Ohlendorf, A. Herkersdorf, T. Wild, “FlexPath NP – A Network Processor Concept with Application-Driven Flexible Processing Paths”, CODES+ISSS 2005, Jersey City, USA, 2005

[8] C. Albrecht, J. Foag, R. Koch, E. Maehle, “DynaCore – A Dynamically Reconfigurable Coprocessor Architecture for Network Processors”, Euromicro Conference on Parallel, Distributed and Network-based Processing, Montbeliard-Sochaux, France, 2006