

# EVALUATION AND DESIGN OF PROCESSOR-LIKE RECONFIGURABLE ARCHITECTURES

*Tobias Oppold*

Department of Computer Engineering, Prof. Dr. W. Rosenstiel  
University of Tuebingen, Sand 13, 72076 Tuebingen, Germany  
email: oppold@informatik.uni-tuebingen.de

## 1. INTRODUCTION

Traditionally, FPGAs are deployed because of their flexibility to change the application over time. Newly developed architectures can be reconfigured within one clock cycle so that components of a device can be re-used within a single application. The reconfiguration keeping pace with the execution yields an additional degree of freedom that constitutes a new principle of reconfiguration. This principle is called *processor-like reconfiguration*.

In the presented work, it is evaluated how processor-like reconfiguration can be exploited by a high-level compiler and which architectural resources are needed for an efficient mapping of applications. For an assessment of the benefits and costs imposed by those resources, a synthesizable model for coarse-grained reconfigurable architectures is used.

## 2. RELATED WORK

In order to take advantage of reconfigurability, a proper temporal partitioning of an application must be found. Various projects deal with dynamic reconfiguration of FPGAs. Due to the long reconfiguration times of those devices, such a partitioning is hard to find. Processor-like reconfiguration allows it to partition an application on the basis of clock cycles and thus to make use of well known techniques from C-based hardware synthesis and from compilers for VLIW processors. E.g. in [1], modulo scheduling is applied to map loops of C descriptions onto the ADRES architecture.

An example for the design space exploration of reconfigurable architectures is the Totem project [2]. The exploration is based on compiled netlists for the RaPiD architecture, whereas the presented approach starts at the level of C descriptions.

Only few works present results on the physical implementation of coarse-grained reconfigurable arrays or on power dissipation. [3] is an example that presents details on a custom implementation including power estimations.

Advances in physical synthesis and power estimation tools enable evaluation at the gate-level at reasonable time and effort. In the presented work, performance, area, and power dissipation are estimated using such commercial tools.

## 3. ARCHITECTURE DESIGN

To evaluate different architectural options, the CRC model (Configurable Reconfigurable Core) was developed as a general model for processor-like reconfigurable architectures. It consists of an array of processing elements (PE) that are connected by a reconfigurable interconnect network. Each PE consists of a functional unit (FU) for word-wide arithmetic and logic operations, a register set, and a context memory that defines several configurations for the PE. At the beginning of each clock cycle, an entry of the context memory is selected by a control unit that implements a finite state machine (FSM). The CRC model can be configured with a variety of parameters in order to create different architecture instances. More details on the CRC model can be found in [4].

The features of the CRC model are modified according to the requirements imposed by mapping applications onto the model. These modifications comprise the adaptation of simple parameters like the width of the data path or the number of PEs in the array, as well as the creation of new features like an improved interconnect network.

For the application mapping, a combined control and data flow graph (CDFG) is derived from the C description of an application. It is assumed that the input data of the application arrives at a certain rate (data input rate, DIR) appointed by the surrounding environment and/or memory accesses. For a variety of DIRs, the operations of the CDFG are scheduled for execution so that the required throughput is achieved using a minimal number of FUs.

Minimizing the number of FUs can be accomplished by assigning the operations to different contexts if the DIR allows distributing the operations over multiple clock cycles. In addition to that, branches in the control flow can be handled by context switches rather than incorporating them in

---

This work is funded by DFG under RO-1030/13 within the 'Priority Program 1148' which is focused on reconfigurable computing systems

the data path [5]. Such a multi-context execution does not only allow it to re-use the FUs but also to re-use the interconnect network in different clock cycles.

After scheduling, the execution of the application is pipelined, registers are allocated, and an FSM that controls the execution is extracted. The result of all these steps defines a so called *execution scheme* and hence how many FUs, registers, I/O ports, states, and contexts are needed to execute an application under a given DIR constraint. The execution scheme also defines the features required for the control unit, for which various options are evaluated.

For each execution scheme, an architecture instance that fits its requirements is generated and the execution scheme is mapped onto the instance. This can be impossible due to insufficient interconnect resources since these are not defined by an execution scheme. In this case a new instance is generated with either more PEs or an extended interconnect network.

#### 4. RESULTS

Besides identifying beneficial compilation techniques and architectural resources, a comparison to other reconfigurable architectures is an important part of the research.

The presented flow has been applied to the luminance calculation ( $xy = (c1 * xr + c2 * xg + c3 * xb + c4) >> c5$ ) of the RGB to YIQ conversion which is part of the EEMBC Consumer Benchmark [6]. Details on this example and results of mapping it for different DIRs onto instances of the CRC model are presented in [4]. To demonstrate the benefits of processor-like reconfiguration, it is assumed in the following that the input values  $xr$ ,  $xg$ , and  $xb$  are provided by the surrounding environment in 3 different clock cycles of a continuous stream and that the clock speed is 200 MHz. This DIR constraint can be imposed, for example, if there is only one 8-bit memory in the surrounding system to read the input values from.

For processor-like reconfigurable architectures, the 7 operations of the example can be distributed over 3 clock cycles as described in the previous section. A 1x3 array of PEs each featuring 4 states for the control unit, 4 contexts, and 4 data registers provides sufficient resources to satisfy the DIR constraint (one state/context is used for initialization).

If the example is executed under the same DIR constraint on a coarse-grained architecture that can only be reconfigured statically, 7 FUs must be provided for the 7 operations. At the output of each FU a register is required to enable pipelined execution. 2 more registers are needed to store intermediate values until they can be consumed by subsequent operations. A 3x3 array of PEs without control unit or context memory and one data register in each PE provides sufficient resources to satisfy the DIR constraint.

The following table summarizes the gate-level results for

two instances of the CRC model featuring the minimal resources described above, targeting a 130 nm standard cell technology. Execution time and power/energy estimations are based on processing 100 input samples.

<i>reconfiguration</i>	<i>PEs</i>	<i>area</i> [mm <sup>2</sup> ]	<i>exe</i> [μs]	<i>power</i> [mW]	<i>energy</i> [nJ]
proc-like	1x3	0.138	1.5	17.8	26.7
static	3x3	0.262	1.5	33.5	50.2

#### 5. CONCLUSIONS AND FURTHER WORK

The comparison of the two architectures providing only minimal resources demonstrates how processor-like reconfiguration can be used to optimize area and power for a given performance constraint by almost factor 2. The results of the static architecture can certainly be optimized by adding resources to reduce unnecessary switching activity or to enable operator sharing. But by doing so, features of the general concept of processor-like reconfiguration are actually added motivating further research on finding the optimal architectural resources.

Ongoing work includes applications with different kinds of branches in the control flow as well as comparisons to other architectures including FPGAs.

The feasibility of the proposed execution schemes has been validated using a commercial architecture [7].

#### 6. REFERENCES

- [1] B. Mei, S. Vernalde, D. Verkest, H. DeMan, and R. Lauwereins, "Exploiting loop-level parallelism on coarse-grained reconfigurable architectures using modulo scheduling," in *Design, Automation and Test in Europe (DATE)*, 2003.
- [2] K. Compton and S. Hauck, "Totem: Custom reconfigurable array generation," in *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2001.
- [3] F.-J. Veredas, M. Scheppeler, W. Moffat, and B. Mei, "Custom implementation of the coarse-grained reconfigurable ADRES architecture for multimedia purposes," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2005.
- [4] T. Oppold, T. Schweizer, J. Oliveira Filho, S. Eisenhardt, T. Kuhn, and W. Rosenstiel, "Execution schemes for dynamically reconfigurable architectures," in *Workshop on Synthesis and System Integration of Mixed Information Technologies (SASIMI)*, Nagoya, Japan, 2006.
- [5] T. Oppold, T. Schweizer, T. Kuhn, W. Rosenstiel, U. Kanus, and W. Straßer, "Evaluation of ray casting on processor-like reconfigurable architectures," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2005.
- [6] Embedded Microprocessor Benchmark Consortium (EEMBC), <http://www.eembc.org>.
- [7] M. Motomura, "A dynamically reconfigurable processor architecture," in *Microprocessor Forum*, 2002.