Hardware Cost Analysis for Weakly Programmable Processor Arrays

Dmitrij Kissler, Frank Hannig, Alexey Kupriyanov, and Jürgen Teich
Department of Computer Science 12, Hardware-Software-Co-Design,
University of Erlangen-Nuremberg, Germany

***Abstract*— Growing complexity and speed requirements in modern application areas such as wireless communication and multimedia in embedded devices demand for flexible and efficient parallel hardware architectures. The inherent parallelism in these application fields has to be reflected at the hardware level to achieve high performance. Coarse-grained reconfigurable architectures support a high degree of parallelism at multiple levels. In this paper technology-independent hardware cost analysis for a new class of highly parameterizable coarse-grained reconfigurable architectures called weakly programmable processor arrays is performed.**

I. Introduction

Due to the recent advances in semiconductor technology the integration of several complex modules like processors, peripheral devices, and memory in a single *System-on-a-Chip (SoC)* is a common practice today. Modern hardware systems demand for a high level of reconfigurability and parallelism. In particular, the modern embedded systems for digital signal processing implement different signal processing standards, like for example multiple wireless communication protocols.

For reconfigurable hardware architectures a classification can be made with the help of a single processing units functionality and the width and configurability of the interconnect [1]. *Look-up tables (LUTs)* constitute the basis of the currently available fine-grained reconfigurable architectures. These LUTs are implemented on the basis of SRAM cells (*Static Random Access Memory*). Together with an extremely flexible interconnect network of 1 bit granularity, look-up tables constitute such modern fine-grained reconfigurable architectures, like *Field Programmable Gate Arrays (FPGA)*. The advantage of high flexibility of the interconnect network on the other side turns out to be very inefficient in terms of area usage. Approximately 80-90 % of the total die area on a typical commercial FPGA is devoted to interconnect [2]. The computational complexity of placement and routing algorithms as well as the big reconfiguration data streams (up to several megabytes in modern FPGAs) are the other undesirable impact of fine-grained interconnect schemes. Furthermore, big reconfiguration data streams introduce long reconfiguration times and demand for large reconfiguration memory size [3].

Coarse-grained reconfigurable architectures can meet the growing demands on computational resources, fast reconfigurability, and flexibility, as well as high power efficency. Due to the big widths of reconfigurable interconnect signals in coarse-grained architectures (up to several words), much of the routing problems are alleviated [3]. As a consequence, the amount of memory for several reconfiguration data streams, as well as the reconfiguration time for coarse-grained architectures is reduced. Complex computations can be accomplished in several processing elements, since these can be complete processors with RISC architecture (*Reduced Instruction Set Computer*). The main contribution of this paper is the generic hardware cost analysis of a new class of massively parallel, reconfigurable, coarse-grained processor architectures called *weakly programmable processor arrays (WPPA)* that we introduced in [4].

This paper is structured as follows: in Section II a brief overview of existing coarse-grained architectures is given. Section III contains the high-level architectural description of the class of weakly programmable processor arrays. In Section IV technology-independent, parameterized hardware cost models for a WPPA are deduced. In Section V synthesis results for an example processor array synthesized on a specific FPGA platform are compared with theoretically estimated cost. Finally, in Section VI we give an outlook on further research subjects.

II. Related Work

Much research effort was spent on the field of coarse-grained architectures and resulted in valuable contributions and academic system prototypes as well as commercially available systems. A detailed overview of some academic coarse-grained architectures can be found for example in [3]. Well-known commercial architectures include the *PACT XPP* from *PACT XPP Technologies* [5], and the *Avispa*, *Moustique*, and *Bresca* IP-Cores (*Intellectual Property Cores*) from *Silicon Hive* [6]. Another examples are *D-Fabrix* from *Elixent* [7], the *Dynamically Reconfigurable Processor* (*DRP*) from *NEC* [8], or *QuickSilver Technology*'s *Adaptive Computing Machine* (*ACM*) [9]. Existing fine-grained and coarse-grained architectures both lack programmability as a result of paradigms, which are very different from the *von Neumann's*. Therefore our concept which we call *weak programmability* is to allow a limited, simple, and parameterizable instruction set for each processing element of the processor array.

Several levels of parallelism are utilized in coarse-grained architectures. Instruction level parallelism is usually accomplished with the help of the VLIW architecture, where the functional units of a single processing element are executing instructions in parallel. The next hierarchical level of parallelism consists of different parallel working processing elements. Potentially, even higher levels are possible, like multiple coarse-grained systems integrated in a high-level array. As an example the multi-core streaming arrays from *Silicon Hive* can be mentioned.

Since the interconnect scheme can significantly affect the type of algorithms running efficiently on the given architecture, it constitutes another interesting research subject. To the best of our knowledge only static interconnect schemes like for

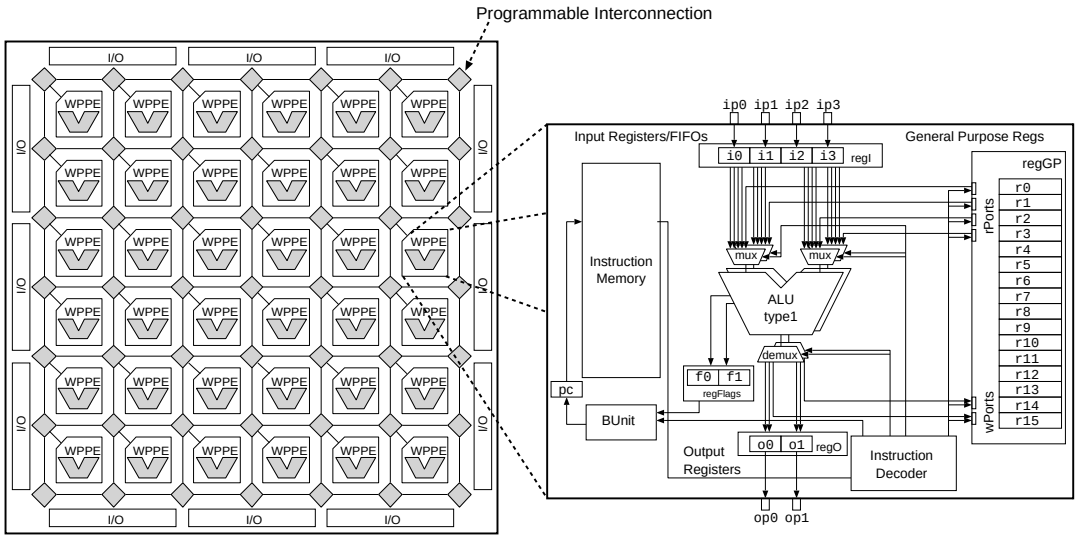


Fig. 1. General Processor Array and Processing Element Structure.

example meshes or trees are used so far for the coarse-grained architectures. Although in some architectures different interconnect schemes can be chosen at design time, they cannot be changed dynamically at run time, or the change only affects special connections, but the basic structure, e.g. bus architecture, remains. With the help of dynamically reconfigurable interconnect schemes a substantial gain in flexibility can be achieved. For this purpose, we introduce a generic concept of an *interconnect wrapper module*. It allows for dynamic reconfiguration of the interconnect scheme in coarse-grained architectures. Furthermore, we derive technology-independent hardware cost models for WPPA, which can be used for automatic design space exploration.

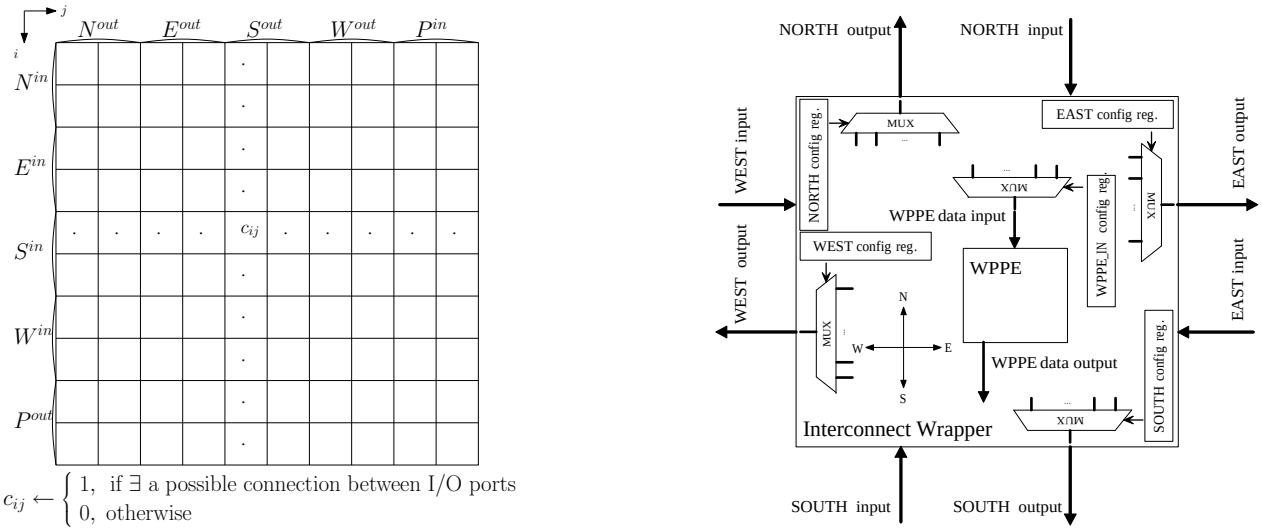
III. Architecture Description

The basic building blocks in our architecture [4], [10] are so-called weakly programmable processing elements (WPPEs), see Fig. 1. They are called weakly programmable because of the limited instruction memory in each processing element and the optimized control overhead, which is kept as small as possible. The instruction set of a WPPE is also kept small and specific to instructions commonly needed in digital signal processing. For each single WPPE the instruction set is parameterizable at compile time. Every WPPE is encapsulated by an *interconnect wrapper* module which is described later. At the first level, single interconnect wrapper modules are statically connected in a grid topology to form a corresponding processor array. A second, dynamic interconnection scheme is provided by the dynamic switching capabilities of each interconnect wrapper module. All possible interconnection schemes for a given processor array are defined at compile-time by means of a set of so-called *adjacency matrices* (exact definition follows).

General Structure of a WPPE. Each WPPE can be parameterized at compile time to contain several functional units like adders/subtractors, multipliers, shifters, and modules for logical operations. The number of specific functional units is also parameterizable at compile time. The possibility to add functional units which implement user-defined functions, like for example a FFT (*Fast Fourier Transformation*) is also provided. Furthermore, a WPPE contains a parameterizable register file for data and also a parameterizable register file for control signals. The parameters of both types of register files are for example number and width of general purpose and output registers and input FIFOs (*First In First Out Buffers*). For transfer of data and control signals between the different storage elements like registers and FIFOs a special data transfer unit is used.

Multiway Branch Unit. For VLIW architectures multiway branch units are very often used [11]. Since a WPPE has a VLIW architecture this approach was also chosen here. With the help of parameterizable number of branch flags a multiplexer for the selection of several branch target addresses is controlled. As a branch flag any control signal from the register file for control signals, input control signals or any status flag from any functional unit for addition/subtraction can be chosen. The different branch target addresses are given in the corresponding branch instruction. With the help of this concept n branch flags are analyzed in parallel and a total of 2^n different branch target addresses can be given by only one branch instruction in a cycle.

Dynamically Reconfigurable Interconnect Wrapper. An interconnect wrapper encapsulates the corresponding weakly programmable processing element and is used to describe and parameterize capabilities of switch networks. Different topologies between the single WPPEs in a weakly programmable processor array like torus, 4D hypercube, and others can be implemented and changed dynamically. To define all possible interconnect topologies, an adjacency matrix is given for each interconnect wrapper in the array at compile time. The structure of the adjacency matrix is exemplarily shown in Fig. 2(a). All input and output signals in the four directions *N* (north), *E* (east), *S* (south), and *W* (west) of the interconnect wrapper, as well as input P^{in} and output signals P^{out} of the encapsulated WPPE are organized in a matrix form. The input signals of the interconnect wrapper and the output signals of



(a) (b)

Fig. 2. (a) Adjacency Matrix Example and (b) Logical Structure of the Reconfigurable Interconnect Module.

the encapsulated WPPE build the rows of the corresponding adjacency matrix, and the output signals of the interconnect wrapper and the input signals of the encapsulated WPPE build the columns of the adjacency matrix. Two input and output signals in each direction of the interconnect wrapper are assumed, and two input and output ports in the encapsulated WPPE for the corresponding example interconnect wrapper and the adjacency matrix in Fig. 2(a). If an arbitrary input i and output signal j have to be connected, the variable c_{ij} in Fig. 2(a) is set to 1. Otherwise it is set to 0. If many input signals are allowed to drive a single output signal a multiplexer with appropriate number of input signals is generated. The inputs of this multiplexer are connected to the corresponding input signals and the output to the corresponding output signal. The logical structure of an interconnect wrapper module is schematically shown in Fig. 2(b). The select signals for such generated multiplexers are stored in configuration registers and can therefore be changed dynamically. By changing the values of the configuration registers in an interconnect wrapper component different interconnect topologies can be implemented and changed at run time. The number and width of the input and output signals of the interconnect wrapper component might be different. This is configured at compile time.

IV. Cost Analysis

In order to perform a technology-independent cost analysis of the highly parameterizable hardware modules they were hierarchically decomposed to the level of basic components given in Table I. These basic components are used in almost all hardware technologies. In Table I, the hardware cost for the basic components is given for a CMOS process (*Complementary Metal Oxide Semiconductor*). These costs are normalized to the complexity of an inverter C_{inv} and are taken from [12]. If our modules are to be implemented in another technology, e.g. FPGA, corresponding costs for the basic components have to be evaluated and can then be inserted into the cost equations. As an example, the technology-independent hardware cost of the adder/subtractor block of a WPPE with x adder/subtractor modules, n bit operands, m bit immediate operand width, and

TABLE I
Hardware Cost of the Basic Components.

Component	*Name*	*Cost*
NOT	$C_{Inv}(n)$	$1n$
AND, OR	$C_{and}(n), C_{or}(n)$	$2n$
XOR	$C_{xor}(n)$	$4n$
MUX_1^2	$C_{mux_1^2}(n)$	$3n$
Flip-Flop	$C_{ff}(n)$	$8n$
RAM-Cell	$C_{rcell}(n)$	$2n$

r bit register address field width in an instruction, is given in Eq. (1).

$$\begin{aligned} C_{\text{add\_BLK}}(x,n,m,r) \; \leq x \cdot \big[& C_{add}(n) + (n-2) \cdot C_{or} + C_{nor} + \\ & + C_{ff}(4) + 3 \cdot C_{xor} + 3 \cdot C_{mux_1^2} + \\ & + C_{mux_1^3}(2) + C_{mux_1^3}(m) + \\ & + C_{mux_1^2}(r) \; + 2 \cdot C_{and}(r) + \\ & + C_{mux_1^2}(n) \big] \end{aligned} \tag{1}$$

If instructions can be executed with both register and immediate operands, the corresponding equation for hardware cost estimation contains the width m in bit of the immediate operand as a parameter. In Eq. (1), the hardware cost for a) the underlying adder/subtractor module $C_{add}(n) \leq 8n$ for a carry look-ahead adder, as well as b) the corresponding instruction decoder modules and c) logic for status flag generation is included. Similar equations were derived for the technology-independent hardware cost estimations of all WPPE and WPPA components. Since for a design space exploration only the relation between the costs of the single components in a design is relevant, hardware cost of an ASIC process from Table I can for example be used to obtain numerical values. The resulting equations are given in Table II.

V. Case Study

A highly parameterizable template for the generation of weakly programmable processor arrays was implemented in

TABLE II
COST EQUATIONS FOR THE FUNCTIONAL UNITS.

Functional Unit Block	*Hardware Cost* $\leq$
Adder/Subtractor	$13n + 6m + 7r + 57$
Multiplier	$10n^2 - 9n + 6r + 13$
Logic	$9n + 3m + 7r + 27$
Shifter	$(6n + 1)\log_2(2n) + 3(n + m) + 7r + 45$

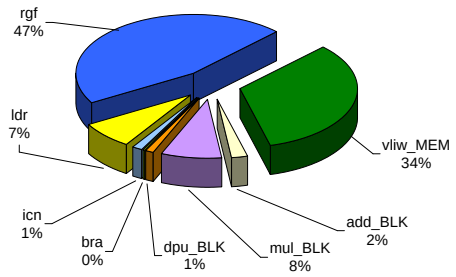
TABLE III
THEORETICAL HARDWARE COST ESTIMATIONS [GATE EQUIVALENTS/C_{Inv}] AND SYNTHESIS RESULTS [GATE EQUIVALENTS] FOR A CASE STUDY WPPE.

Module	*Estimation*	*Synthesis*
Register File (rgf)	13576	27084
Instruction Memory (VLIW_mem)	9906	16259
Adders block (add_BLK)	646	1663
Multiplier (mul_BLK)	2453	4078
Units for Data Transfer (dpu_BLK)	348	432
4-Way Branch Unit (bra)	111	108
Interconnect Wrapper (icn)	432	573
Configuration Loader (ldr)	2198	5179
Total (WPPE & ICN-Wrapper)	29670	56244

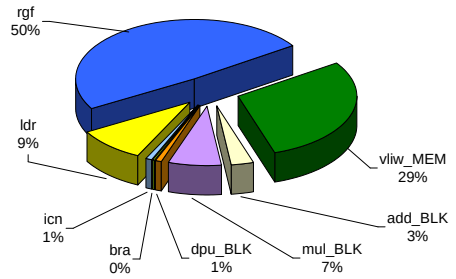
VHDL. An example array with four WPPEs was instantiated and tested on a *Xilinx Virtex-II Pro™ xc2vp30* FPGA. Each processing element was compiled to include two adder modules, one multiplier module, and three modules for the transfer of data. The data path width was chosen to be 16 bit. Each WPPE has 2 Kbyte of local memory. The maximum operating frequency for this design is 84 MHz. The whole SoC-Design uses 54% of the FPGA resources. Table III summarizes the estimated costs using the data from Table I and the FPGA synthesis results. It can be seen from Table III that the ratio between the estimated hardware cost (normalized to the cost of an inverter) and the synthesis result in gate equivalents is approximately $1 : 2$. In Fig. 3(a) the estimated relation between different components is shown graphically. Synthesis results can be seen in Fig. 3(b). The orders of magnitude between different implemented modules are quite accurate. This indicates that the comparison of different designs in a design space exploration is accurate enough to take the right calculations for searching cost-optimal architectures.

VI. CONCLUSION

In this paper a new class of massively parallel embedded processor architectures called weakly programmable processor arrays was introduced. A novel approach for dynamically reconfigurable interconnection schemes for coarse-grained reconfigurable architectures was also introduced by means of a concept called *"interconnect wrapper"*. Furthermore, we developed high-level technology-independent hardware cost models for single processing elements as well as for the whole processor array. On the basis of such abstract hardware cost models and mapping methodology we want to perform automatic design space explorations for weakly programmable processor arrays. Besides this architectural research we are currently working on retargetable compilation techniques. A preliminary overview of how to match architectural parameters and mapping methodology is given in [13].



(a)



(b)

Fig. 3. (a) Theoretical Estimation of Hardware Cost Relations in the Case Study WPPE Design and (b) Synthesis Results for a *Xilinx Virtex-II Pro™ xc2vp30* FPGA.

ACKNOWLEDGMENT

This project is supported in part by the German Science Foundation (DFG) in project under contract TE 163/13-1.

REFERENCES

[1] T. J. Todman, S. J. E. Wilton, O. Mencer, W. Luk, G. A. Constantinides, and P. Y. K. Cheung, "Reconfigurable Computing: Architectures and Design Methods," in *IEE '05: IEE Proceedings - Computers and Digital Techniques*, vol. 152, 2005, pp. 193–207.

[2] A. Singh and M. Marek-Sadowska, "FPGA Interconnect Planning," in *SLIP '02: Proceedings of the 2002 International Workshop on System-level Interconnect Prediction*. New York, NY, USA: ACM Press, 2002, pp. 23–30.

[3] R. Hartenstein, "A Decade of Reconfigurable Computing: A Visionary Retrospective," in *DATE'01: Proceedings of Design Automation and Test in Europe*, Mar. 2001, pp. 642–649.

[4] D. Kissler, F. Hannig, A. Kupriyanov, and J. Teich, "A Dynamically Reconfigurable Weakly Programmable Processor Array Architecture Template," in *Proceedings of the 2nd International Workshop on Reconfigurable Communication-Centric System-on-Chips*, Montpellier, France, July 2006.

[5] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt, "PACT XPP – A Self-Reconfigurable Data Processing Architecture," *The Journal of Supercomputing*, vol. 26, no. 2, pp. 167–184, 2003.

[6] Silicon Hive, *Product Overview*, http://www.siliconhive.com.

[7] Elixent Ltd., *Product Overview*, http://www.elixent.com.

[8] M. Motomura, "A Dynamically Reconfigurable Processor Architecture," in *Microprocessor Forum*, CA, 2002.

[9] Quicksilver Technology, *Product Overview*, http://www.qstech.com.

[10] D. Kissler, F. Hannig, A. Kupriyanov, and J. Teich, "A Highly Parameterizable Parallel Processor Array Architecture," in *Proceedings of the IEEE International Conference on Field Programmable Technology (FPT)*, Bangkok, Thailand, Dec. 2006.

[11] S. M. Moon and S. D. Carson, "Generalized Multiway Branch Unit for VLIW Microprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 8, pp. 850–862, 1995.

[12] S. M. Müller and W. J. Paul, *Computer Architecture: Complexity and Correctness*. Springer, 2000.

[13] H. Dutta, F. Hannig, and J. Teich, "Mapping of Nested Loop Programs onto Massively Parallel Processor Arrays with Memory and I/O Constraints," in *Proceedings of the 6th International Heinz Nixdorf Symposium, New Trends in Parallel & Distributed Computing*, ser. HNI-Verlagsschriftenreihe, F. Meyer auf der Heide and B. Monien, Eds., vol. 181, Paderborn, Germany, Jan. 2006, pp. 97–119.