Physical 2D Morphware and Power Reduction Methods for Everyone

Jürgen Becker, Michael Hübner, Katarina Paulsson
Universität Karlsruhe (TH), Germany
http://www.itiv.uni-karlsruhe.de/
{becker, huebner,paulsson}@itiv.uni-karlsruhe.de

Abstract

Dynamic and partial reconfiguration discovers more and more the focus in academic and industrial research. Modern systems in e.g. avionic and automotive applications exploit the parallelism of hardware in order to reduce power consumption and to increase performance. State of the art reconfigurable FPGA devices allows reconfiguring parts of their architecture while the other configured architecture stays undisturbed in operation. This dynamic and partial reconfiguration allows therefore adapting the architecture to the requirements of the application while run-time. The difference to the traditional term of software and its related sequential architecture is the possibility to change the paradigm of brining the data to the respective processing elements. Dynamic and partial reconfiguration enables to bring the processing elements to the data and is therefore a new paradigm. The shift from the traditional microprocessor approaches with sequential processing of data to parallel processing reconfigurable architectures forces to introduce new paradigms with the focus on computing in time and space.

Keywords: Dynamic and partial reconfiguration, Communication primitives, run-time adaption, FPGA

1. Introduction

1. Tabellen

The term of Morphware was coined from a participant of the Polymorphous Computing Architectures (PCA) program in the early 90th [6]. The idea behind this term is to change hardware, which is no longer “hard”, while run-time and “Morph” the integrated architecture while run-time in order to adapt to requirements of the application. The benefit here is to work with parallel processing architecture in comparison to the traditional approach of sequential architecture like microprocessors in von Neumann architectures. Reiner Hartenstein classified the terms of the novel paradigm for reconfigurable computing [7]. As described in figure 1, the term Morphware is used for fine- and coarse-grained reconfigurable architectures. Configware is the term for the structural code for programming these reconfigurable platforms.

platform			program source running on it	machine paradigm
hardware			(not programmable)	none
morphware	fine grain	rGA (FPGA)	configware	
	coarse grain	rDPU, rDPA		
		reconfigurable data stream processor	flowware & configware	anti machine
data stream processor (hardwired)			flowware	
instruction stream processor			software	von Neumann machine

Figure 1. Classification of terms. Source: Reiner Hartenstein [7]

This Configware is now used to run on the “Morphware” FPGA in order to enable a fine-grained adaption to the requirements of applications. The presented approach allows the adaptation of the hardware with a novel 2D-placement and routing approach. The benefit to the traditional 1D approach (see [5] and [4]) is the improved utilization of hardware with a high flexibility. Another important aspect is the possibility to adapt the internal wiring to performance / power consumption requirements of the application. This on-line routing approach enables a filigree adjustment of hardware and allows to have an impact to power / performance trade-off while run time. The following sections introduce the ideas and already integrated applications.

The paper is organized in the following sections and sub-sections: Section 2 describes the basic idea of the novel 2D placement approach. In section 3 one hardware integration is presented which is the basis of further investigation described in section 4. Section 5 introduces the challenging task to exploit online-routing as an

Dagstuhl Seminar Proceedings 06141
Dynamically Reconfigurable Architectures
http://drops.dagstuhl.de/opus/volltexte/2006/739

approach for power optimization issues. The paper is closed in section 6 with the conclusions and future work.

2. 2D-Online Placement and Routing

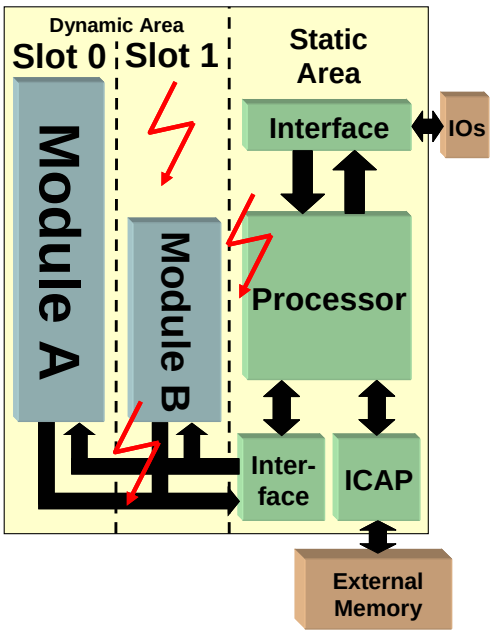

Figure 2. 1D approach for reconfigurable system on FPGA

Figure 2 shows the traditional approach for a dynamic and partial reconfigurable system. The system consists of a static area with a controlling element (Soft-IP Microcontroller) and interface IP-Cores. On the left side a dynamic area includes the modules with application dependent functionality. While run-time the modules on the left side will be replaced on-demand by exploiting the dynamic and partial reconfiguration method which is available for Xilinx FPGAs. A system working with this method was presented in [5]. In the figure restrictions of this system approach are marked with red signs. These restrictions can be found e.g. in Slot 1 where the utilization of Module B does not occupy the complete area. This waste of reconfigurable area has its cause of the fixed slot width which is defined by the module with the highest resource utilization. Another restriction is the bus-based communication mechanism which does not allow for modules to communicate in parallel with each other. The static area is not included as dynamic configurable area which also leads to a suboptimal utilization of the architecture. Also there precious reconfigurable resources may be usable if the approach isn't partitioned stringently. Therefore a new 2D approach will be introduced to overcome these restrictions. Figure 3 shows the schematic view to the 2D approach for a dynamic and partial reconfigurable system. The system is no longer slot based in order to utilize the configurable area more optimized. A more detailed view on the 2D approach is described in [8].

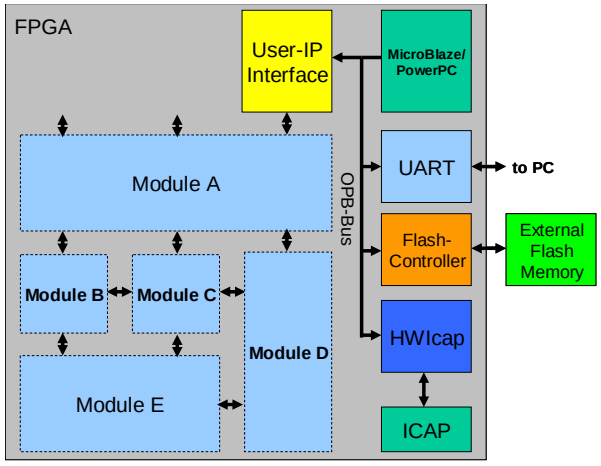

Figure 3. Schematic of 2D-system Approach

3. Hardware Integration

The 2D approach which was described in the previous section was implemented on Xilinx Virtex-II and Virtex-II Pro Architectures.

Figure 3 shows a schematic view of the modular reconfigurable system. As a control element the soft-core processor MicroBlaze or the hard-IP PowerPC 405 in a Virtex-II Pro architecture can be used. The system is connected via a Flash controller to an external Flash memory. The controller enables both read or write access to the memory device. Another device integrated into the FPGA is the Hardware-ICAP module which provides the access to the FPGA internal ICAP-interface for reading or writing configuration data to or from the devices' configuration memory. This Xilinx IP-Core is described more in detail in [2]. The software drivers provide several functions to access the configuration memory. As a buffer, an internal block-RAM is used to store the data, which has been read from the device. The software to handle the data has access to this block-RAM for further processing while run-time. First implementation uses the connection via RS232 (Uart-IP) to read the configuration data from an external PC. For this purpose, the software running on the internal processor only provides the GUI to handle external commands for accessing the hardware ICAP IP-core. As shown in Figure 3, rectangular shaped modules are connected via a user-IP interface to the controlling element. The size of these modules and their interfaces are described in the next section.

After receiving the configuration data with the PC, JBits was used to modify the bitstream and write it back to the device. This first approach was developed for

debugging purpose. To establish the data-base of modules, the complete content of the FPGA is read for extracting the relevant data of the modules. This is achieved by using the Java based functions of JBits. After extraction of the modules' configuration data, the Flash is programmed with the different configurations of the modules. Important here is, that the stored data does not contain any placement information. This is the initialization phase of the system. While run-time, it is possible to place the stored module to an area by including the position information. The process to place a

module, starts by reading back the columns of the area, which is dedicated for the placement. The data which were read-back contains also the configuration information of the upper and lower areas. Now the stored module data were merged into this bitstream. Only the positions of the bitstream concerning the module which has to be placed were substituted. Therefore, configuration outside of the dedicated area is not affected.

The implemented system enables to position a module variable to a vertical position. This is due to the fact that the Virtex-II architecture is homogeneous in vertical direction. Positioning a module to any horizontal coordinate is not possible. The cause here is the heterogeneous architecture which includes block-RAM and multiplier blocks. Routing resources crossing these blocks are different to those which can be used in an area without BRAM elements. However, placement of modules to an area with same horizontal conditions is possible. Figure 4 shows the integrated system on a Virtex-II Pro FGPA. More detailed information about the approach can be found in [12].

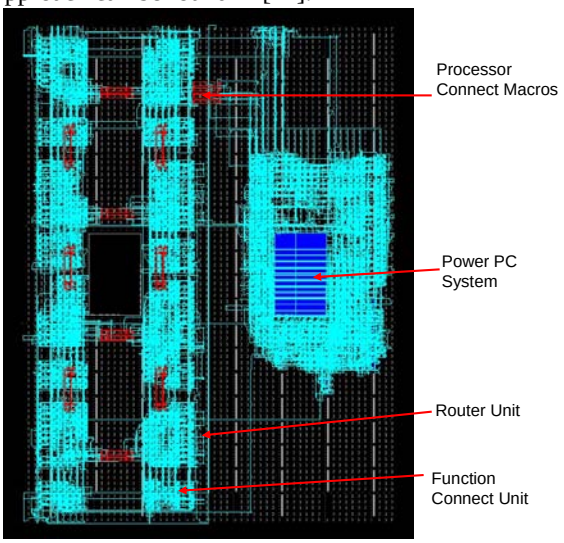

Figure 4. Hardware Integration of 2D Placement Approach

4. FPGA communication primitives

FPGA architectures normally have different kinds of communication primitivs to efficiently cover the entire surface of the FPGA. Figure 5 shows the different signal lines on a Xilinx Virtex II FPGA. The long lines span the entire FPGA vertically and horisontally. They provide a higher performance than the shorter lines, but therefore consume more power.

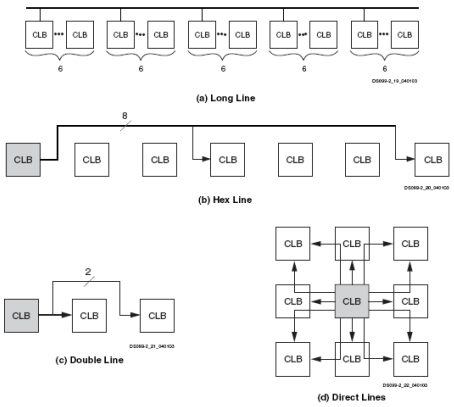

Figure 5. Communication primitives on Virtex II FPGAs [9]

The hex lines span 3 or 6 Configurable Logic Blocks, CLBs, and also provide a higher performance but are shorter than the long lines. The double and direct lines are shorter and less performant, but also have lower power consumption due to a decreased capacitance.

One important question is if an optimized selection of the used signal lines during the routing phase can lead to a decreased power consumption while maintaining the required performance. And if this can be done statically, can the routing be adapted during run-time to adjust to changing requirements on performance or power consumption, by exploiting dynamic reconfiguration? In order to further investigate this, it was necessary to now the power consumption and performance of the different signal lines.

The different communication primitives on a Xilinx Virtex II FPGA can be implemented separately as so called bus macros (see [11]). A bus macro can be placed on a specific position on the FPGA, and the length of the signal line as well as the resources used to implement it can also be specified. Figure 6 shows an overview of the implementation of the four different signal lines available on a Xilinx Virtex II FPGA, in the FPGA Editor.

By generating a so called VCD (Value Change Dump) file during a timing simulation, the power consumptions of the signal lines can be estimated in the XPower tool from Xilinx (see [10]). The VCD file contains all signal transactions during the entire simulation, and by using this file together with the complete implementation

information of the signal line, a very exact estimation of the power consumption can be performed.

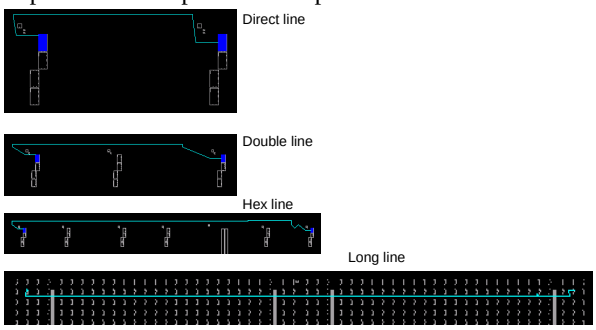

Figure 6. FPGA basic signal lines on a Virtex II FPGA, FPGA Editor

Figure 7 presents the result from the investigation of the power consumptions and worst case delays of the signal lines. The worst case delays were extracted from the timing report generated by Xilinx Integrated Software Evironment, ISE, tools.

In order to determine if the power consumption can be decreased by optimizing the routing, multiple shorter signal lines can be connected to span the same length as one longer line. For example, three double lines can be used instead of one hex line, or multiple hex lines can be connected instead of one long line. This is however only possible if the timing constrints are fulfilled.

To evaluate the power/performance comparison between one long line and multiple shorter lines, the same

Macro	Max. delay	Power Consumtion	Frequency
Direct	0.581 ns	0.125 mW	17 MHz
Double	0.590 ns	0.126 mW	17 MHz
Hex	0.755 ns	0.134 mW	17 MHz
Long	1.271 ns	0.280 mW	17 MHz

Figure 7. Power consumption/ maximum delay times for Virtex II communication primitives

tool flow with timing simulation and estimation of the power consumption in XPower was used. First one long line was implemented by connecting multiple hex lines, and then one hex line was realized with three double lines. The results of the power consumptions of the different signal lines can be seen in Figure 8. There it is shown that the usage of multiple hex lines instead of one long line reduces the dynamic power consumption with approximately 20%, while the worst case delay is only slightly increased. Using three double lines instead of one hex lines also reduces the power consumption.

Macro	Max. delay	Power consumtion	Frequency
1 Hex	0.755 ns	0.134 mW	17 MHz
3 Double	0.798 ns	0.125 mW	17 MHz
1 Long	1.271 ns	0.280 mW	17 MHz
Mult. Hex	1.828 ns	0.223 mW	17 MHZ

Figure 8. Routing / power comparisons

Even though the gain in power consumption may seem very low, it must be emphasized that a typical FPGA design contains many thousands nets. By optimizing the most important nets where signal transactions are frequently occurring, the efficiency of this method is increased.

5. On-line power/performance trade-off

The above presented results from power consumption estimations and timing predictions can be used to optimize the routing in a FPGA to decrease the power consumption caused by the routing. Optimization possibilities can be identified in many types of designs, for example in the implementation of the digital waveform generator which is shown in Figure 9. In this implementation many nets can be rerouted to decrease the power consumption, and one possibility is marked in red in Figure 9. The specific net is a hex line which can be implemented with multiple double lines as long as it i smade sure that the timing constraints are kept.

It is also easy to see that the most important nets to reroute are the ones with frequent signal transactions. The VCD file, which can be generated during simulation, can be used to provide information about on which nets most of the signal transactions will occur in the system. This way, the proposed technique can be focused on the nets which will have greater effect on the complete power consumption.

In an adaptive system, it is also possible that the requirements on performance will change during run-time. One example where such a scenario is exploited is clock scaling, which is used to put systems into sleep modus when no service is required. But it is also possible that a future self-adaptive system can adjust its performance requirements according to its workload and energy resources. For example, the frequency could be dynamically decreased for some tasks in order to save power. In such a scenario, the system could reroute the

tasks with decreased frequency and adjust the routing resources according to the timing constrains and save power. This would naturally require an on-line rerouting process.

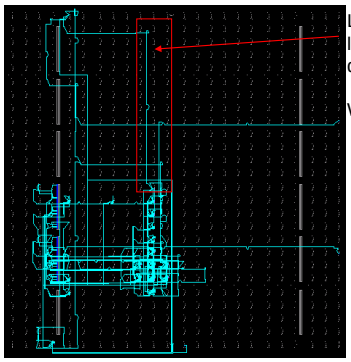

Figure 9. Routing optimization possibilities

It would also be possible that the system uses integrated monitors that observe the signal transactions on some signal lines. At a certain point, when the number of signal transactions is very high, the routing on those nets could be optimized in order to save power. This would be useful in self-adaptive systems where it is not possible to predict when signal transactions will occur, or on which nets the number of signal transactions will be high and at what time. By integrating monitors, this can be discovered by the system itself during run-time and passed on to the run-time decision making mechanisms.

6. Conclusions and Future Work

This paper presents the first approach for 2D placement and routing for dynamic and partial reconfigurable systems on Xilinx Virtex-II and Virtex-II Pro FPGAs.

The paper shows the promising investigation by exploiting 2D dynamic and partial reconfiguration for optimizing FPGA utilization and power consumption while run-time. Actual results show, that optimization of signal lines provide up to 10% improvement of power consumption. Next steps are the integration of a complete analysis of general systems and the online adaptation coupled with a run-time environment in order to have a filigree possibility of adjustment for performance and power trade-off.

The additional degree of freedom for adapting reconfigurable systems by introducing the 2nd dimension for placement and the availability of online-routing enables to increase the adaptivity of embedded system. Next steps are to introduce an online-routing mechanism which is based on netlists stored in external memory. The benefit here is to be architecture independent which allows transferring very easily IP from one architecture to another while design- and run-time. Further on, well established methods for scheduling and optimization from informatics have to be extended to the idea of computing in time and space. Graph theory and the related algorithms will be extended and support the paradigm of reconfigurable computing.

2. Tabellen 1.

7. References

3. Tabellen

[1] L. Benini, G. De Micheli: “Networks on Chip: A New Paradigm for Systems on Chip Design”, Date 02, March 3~7, Paris France

[2] B. Blodget, S. McMillan: “A lightweight approach for embedded reconfiguration of FPGAs”, Date03, Munich Germany

[3] P. Lysaght: “Future Design Tools for Platform FPGAs”, Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI’03), Brasil

[4] J.C. Palma, A. Vieira de Melo, F. G. Moraes, N. Calazans, "Core Communication Interface for FPGAs", SBCCI02, Porto Alegre BRAZIL

[5] M. Ullmann, M. Huebner, B. Grimm, J. Becker: “An FPGA Run-Time System for Dynamical On-Demand Reconfiguration”, RAW04, Santa Fee

[6] Daniel P. Campbell; Mark A. Richards; Dennis M. Cottel; Randall R. Judd, Kenneth M. McKensie;: “The Morphware Stable Interface: A Software Framework for Polymorphous Computing Architectures”, GEORGIA INST OF TECH ATLANTA, August 2004

[7] Reiner Hartenstein: “Embedded Architectures: Configurable, Re-configurable, or what?”, Panel on Embedded Architectures, CASES 2002 (International Conference on Compilers, Architecture, and Syntheses for Embedded Systems), October 8-11, 2002, Grenoble, France

[8] M. Hübner, C. Schuck, M. Kühnle, J. Becker: "New 2-Dimensional Partial Dynamic Reconfiguration Techniques for Real-Time Adaptive Microelectronic Circuits", ISVLSI2006, Karlsruhe, Germany

[9] Virtex-II Platform FPGAs: Complete Data Sheet, Xilinx DS031, March 2005

[10] XPower Analyzer FAQ, Xilinx, 2006

[11] M. Huebner, T. Becker, J. Becker: "Real-time LUT-based Network Topologies for dynamic and partial FPGA Self-Reconfiguration", SBCCI04, Porto de Galinhas, Brasil

[12] M. Huebner, C. Schuck, J. Becker: "Elementary Block Based 2-Dimensional Dynamic and Partial Reconfiguration for Virtex-II FPGAs", RAW2006, Rhodos, Griechenland

[13] K. Paulsson, M. Hübner, J. Becker: "On-Line Optimization of FPGA Power-Dissipation by Exploiting Run-time Adaption of Communication Primitives", SBCCI2006, Brazil