Exploitation of the External JTAG Interface for Internally Controlled Configuration Readback and Self-Reconfiguration of Spartan 3 FPGAs

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

Abstract

Field Programmable Gate Arrays, FPGAs, are increasingly often applied in various industrial applications as well as investigated in different research projects. Due to the possibilitiy for performing parallel computations, this kind of hardware architecture is especially interesting for high-performance applications. Dynamic and partial hardware reconfiguration, which is provided by several FPGA families such as the Xilinx Spartan 3 and Virtex 2/4 families, further increases the flexibility of these architectures. The Spartan 3 family was a less attractive choice for performing dynamic and partial reconfiguration due to the lack of an internal configuration port. However, a virtual internal configuration port, JCAP, has previusly been realized by using the external JTAG interface. This paper presents an approach for internal configuration readback for failure detection and task migration by extending the JCAP core functionality. The paper also presents the first results from implementing self-reconfiguration over JCAP.

Keywords: Low-power applications, reconfigurable architectures, failure detection, JTAG

1. Introduction

The application of Field Programmable Gate Array's, FPGAs, has mainly been investigated for highly computational applications e.g. signal processing or multimedia devices. The typical characteristics (e.g. the structure of programmable switch boxes for the routing and programmable Look- Up Tables for realizing the logic behaviour) of this kind of hardware architecture make it very suitable for high performance applications that can exploit the possibility for parallel calculations.

Furthermore, the possibility to perform dynamic and partial reconfiguration on some available FPGA families further increases the flexibility of systems implemented on this kind of architecture. Dynamic and partial hardware reconfiguration makes it possible to store the functional descriptions of tasks that are currently not required in an external memory and to exploit their resources for executing other tasks. Upon system request the functions may be dynamically configured. This might lead to the usage of a smaller chip size which decreases costs and power consumption.

The application of hardware reconfiguration is also highly interesting in the design of self-adaptive systems for future applications, which is major research area with many currently on-going projects. Since hardware reconfiguration enables an additional degree of flexibility, this also simplifies the realization of self-adaptive characteristics such as self-configuration, self-optimization and self-recovery. Self-recovery after the detection of system failures is becoming increasingly important, especially when considering the increased complexity of future embedded systems which makes it difficult to detect faults during the design phase.

Dynamic and partial hardware reconfiguration on FPGA has until now mainly been investigated on Xilinx Virtex 2/4 families [10], since they provide the Internal Configuration Access Port, ICAP, which allows the realization of internal self-reconfiguration.

The Xilinx Spartan 3 FPGA family [2] also provide the feature of dynamic and partial hardware reconfiguration, but unfortunately not the ICAP port. The advantage of the Spartan 3 FPGA family is the fact that this architecture has been developed for low-cost and low-power applications, which makes it highly interesting even for systems exploiting hardware reconfiguration. Therefore, a virtual internal core, JCAP (JTAG Configuration Access Port), has been implemented which makes it possible to perform internal self-reconfiguration even on Spartan 3 FPGAs [11]. This core exploits the external JTAG interface for reconfiguration, and includes a Finite State Machine implemented in VHDL for controlling the TAP controller in the JTAG interface.

This JCAP core can also be extended to include additional functionalities to the partial reconfiguration capability. It is also possible to read back configuration data in order to for example verify the configuration bitstreams or in order to extract register contents. This can be used to integrate techniques for on-line self-tests or task migration in dynamically and partially reconfigurable systems based on the Spartan 3 FPGAs. Since the JCAP core exploits the external JTAG interface, the core can also be exploited in systems designed for Virtex 2/4 and 5 FPGAs.

This paper presents an approach for extending the first version of the JCAP core to include readback capabilities, as well as how this information can be exploited by the system during run-time. Also, some results from the first investigations on partial self-reconfiguration on a Spartan 3 FPGA are presented. The paper is organized the following way: chapter 2 gives an overview of the low-power capacity-based level measurement application for Spartan 3 FPGAs. Section 3 discusses dynamic and partial reconfiguration on FPGAs and in addition to this, chapter 4 gives an overview how configuration readback can be controlled internally by the FPGA system. Chapter 5 presents the first results from performing self-reconfiguration on Spartan 3 FPGAs. Finally, the paper is concluded in section 6, which also presents the future steps for this work.

2. Overview of the Spartan 3-based low-power application

The target application is a low-power capacity-based level measurement system. The system measures the level of material in a tank by monitoring the change of capacity within the tank. The level of the material is then given by the calculated capacity. Normally this application is implemented on low-power microcontrollers, but the exploitation of FPGAs is investigated for increased system flexibility and adaptation to dynamically changing requirements.

For example, different interfaces are used for communication with the measurement system, for reading the data and for system calibration. FPGAs allow dynamic adaptation of the interfaces by performing hardware reconfiguration to switch between e.g. an Ethernet interface to a Profibus interface. This way, the user may very easily switch between local or remote level monitoring, while reducing the required hardware resources. Further more, FPGAs allow the implementation of efficient failure detection mechanisms as well as fast run-time adaptation of the data processing algorithms.

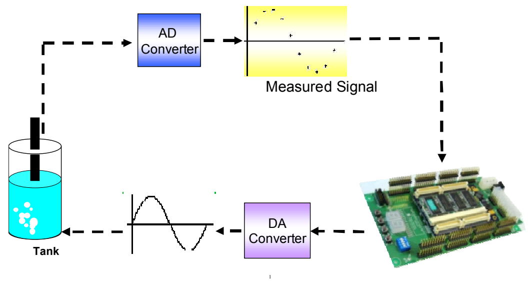

Figure 1. FPGA based level measurement system

The basic functionality of the measurement system can be described the following way: the system generates an 8-bit wide digital signal, which is converted into an analog sinus signal by an external DA converter. The signal is then applied to the tank in which the level measurement is performed. Every measurement cycle the current is detected back from the tank and converted into digital data by an external AD-converter. The digital data is then processed and the level of material in the tank is calculated. An overview of the complete FPGA- based system is given in Figure 1.

The system implementation has been optimized for FPGAs by realizing the data processing algorithms in hardware as reconfigurable modules. Originally, these algorithms were implemented in software and executed by a microcontroller, but previous work has shown that the software-based does not provide enough performance and requires a great deal of internal BRAM blocks. Since the system was implemented on a Spartan 3 FPGA (using the Xilinx Embedded Development Tools Kit [1][7] and the Xilinx Integrated Software Environment [4]), it was necessary to solve the problem caused by the lack of an internal configuration port, such as the ICAP [10] on Xilinx Virtex 2/4 and 5 FPGAs. Therefore, the JCAP core was implemented, as presented in [11]. This core allows the internal control of the partial reconfiguration process by the system itself. The described measurement application would benefit from the exploitation of readback for failure-detection purposes and for increased flexibility when performing reconfiguration.

3. Dynamic and partial hardware reconfiguration on Spartan 3 FPGAs

Some of Xilinx FPGA families (e.g. Virtex 2, Virtex 4 and Spartan 3) offer the possibility to perform dynamic and partial hardware reconfiguration. The exploitation of this feature has been investigated in several research projects for different kinds of applications, e.g. in [5] and [13].

The main parts of the earlier investigations where state-of-the-art FPGAs have been used are based on Virtex 2 or Virtex 4 FPGAs, which include the Internal Configuration Access Port, ICAP. This component enables the system to perform internal self-reconfiguration, so that no additional external components are required to control the reconfiguration process.

One limitation when performing dynamic and partial hardware reconfiguration on Virtex 2 FPGAs is the fact that reconfiguration can only be done over completely vertical areas of the FPGA architecture. The reason for this is the fact that the smallest addressable unit of the configuration memory is a so-called frame, which stretches completely vertically over the FPGA and has a width of 1 bit. For the Virtex 4 devices however, the frames have been further partitioned vertically in multiple addressable units which allows a more flexible reconfiguration. The restriction of vertical reconfiguration has also lead to the introduction of slice based busmacros, which are fixed implemented signal lines for communication over reconfigurable areas [9]. Furthermore, the reconfigurable system must be divided into static and dynamic areas, where the static area includes the functions that are constantly required by the system, e.g. controllers or interfaces. The dynamic area provides reconfigurable slots for the reconfigurable functions.

Dynamic and partial hardware reconfiguration on Xilinx Spartan 3 FPGAs has not been investigated as much as on Virtex 2 and Virtex 4. The reason for this is most likely the lack of an internal configuration port. Furthermore, the Spartan 3 FPGA family does not allow frame wise partial reconfiguration, only CLB column wise, which naturally decreases the flexibility of the reconfiguration process. Spartan 3 FPGAs are however still highly interesting even for reconfigurable applications, due to its cost-and-power optimized architecture.

In [11], the implementation of a virtual configuration access port for Spartan 3 FPGAs is presented. This core is implemented in VHDL and manages internal control of partial reconfiguration over the external JTAG interface. In [11], the basic implementation of this core is presented as well as some first results regarding performance and resource utilization. This paper will present results from performing the first tests of internal self-reconfiguration over the JCAP core, as well as an extension of the core to control readback of the configuration memory for additional functionalities such as self-tests, task migration and 2-dimensional hardware reconfiguration.

4. Internally controlled configuration readback

Xilinx Spartan 3 FPGAs also allow run-time readback of the configuration memory. This can be exploited for implementation of self-test mechanisms (based on so-called scrubbing), 2-dimensional hardware reconfiguration and task migration. “Scrubbing” [12] for testing the FPGA implementation during run-time, means that the configuration is read back and compared to the fault-free implementation. This technique has been investigated in detail, also by performing readback over the Internal Configuration Access Port, ICAP, on Virtex 2 FPGAs.

2-dimensional hardware reconfiguration has been investigated on Virtex 2 FPGAs for increasing the flexibility of partial reconfiguration [14]. The Virtex 2 architecture only allows completely vertical frame-wise reconfiguration, which means that the reconfigurable areas must be divided into reconfigurable slots that stretch completely vertically over the FPGA. This is naturally a constraint of flexibility when performing partial reconfiguration and also makes it difficult to maintain the timing constraints for the reconfigurable functions, since the router must implement the functions on a fewer number of CLB columns and completely vertically on the FPGA in order to save resources. Therefore read-modify-write-back was performed over the ICAP interface in order to realize 2-dimensional reconfiguration on Virtex 2 FPGAs. Virtex 4 FPGAs however, even allow partial reconfiguration of areas that do not stretch completely over the FPGA. The frames in the configuration memory have thereby been partitioned in multiple addressable areas, which can be dynamically reconfigured. Such a scenario has also been investigated [15].

The readback feature may even be exploited to perform task migration. When performing task migration of control flow oriented functions (migration of task execution from software to hardware or hardware to software, or from 2 different hardware implementations) the current state of the Finite State Machine, FSM, must be read back and written to the alternative implementation. This can be done by reading back the content of the flip-flops that define the state of the FSM, and either storing the state or immediately writing it to the alternative implementation.

In order to realize the 3 above briefly described features, it is necessary to perform readback. Readback is possible over either the SelectMap interface or over the JTAG interface. A dedicated register in the JTAG interface allows access to the configuration data, and can be used to write to and read from the configuration memory. This is straightforward when reading back the configuration over the JTAG interface from a host PC, or over the ICAP interface on Virtex 2 or 4 FPGAs. Since the Spartan 3 FPGA family does not provide the ICAP interface, the JCAP core may be extended and used to read back the configuration data over the external JTAG interface.

The following section will give a brief overview of the JTAG port on Spartan 3 FPGAs.

4.1. The JTAG port on Spartan 3 FPGAs

The Xilinx Spartan 3 FPGA family provides the JTAG port according to the IEEE 1149.1 standard [16] with some additional Xilinx specific functionalities. Figure 2 gives an overview of a typical JTAG interface.
The port has 3 input pins, TMS, TCK and TDI, and 1 output pin: TDO. TMS, Test Mode Select decides the state transition of the Test Access Port, TAP, controller. TCK is the input for the clock signal, TDI provides a serial input data pin and the TDO is the Test Data output pin. The standard interface provides the Instruction register, the BYPASS register, the IDCODE register and

the Boundary Scan register. Xilinx Spartan 3 FPGAs also provides some additional device specific registers [16]. The JTAG Configuration Register for example, is the register that provides access to the configuration data bus. The already implemented JCAP core exploits this register for controlling the dynamic and partial reconfiguration process. Here, an approach for extending the JCAP core with the functionality to perform readback is presented.

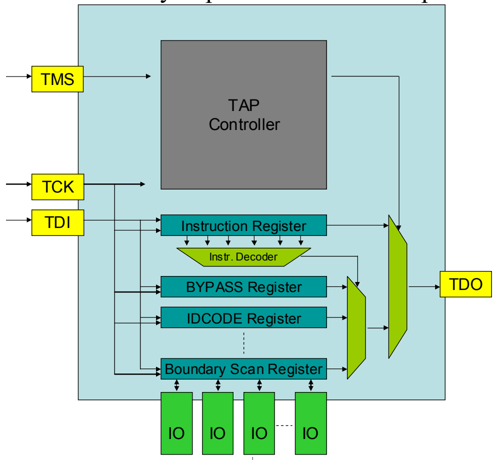

Figure 2. Typical JTAG interface

4.2. Internal control of the readback process

Figure 3 gives an overview of the TAP controller which is integrated in the JTAG interface.

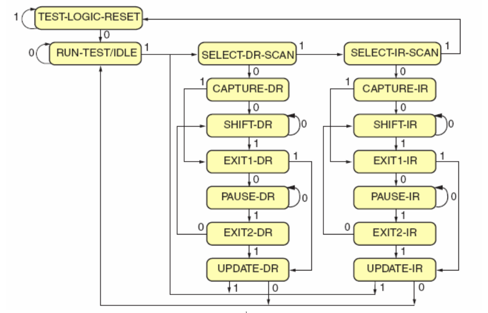

Figure 3. The Test Access Port (TAP) controller [16]

The state transitions are synchronized by the TCK input of the JTAG interface, and controlled by the TMS input. In order to control the readback process internally, the JCAP core must first write the Instruction Register, see Figure 2, with the CFG_OUT command [16]. The TMS input is sampled on the raising edge of TCK. After writing this command to the Instruction Register, the Data Register must be written with a pad frame which includes the commands for readback as specified in [17]. These commands are similar to the commands which are included in the pad frame for a partial bitstream, but specify readback instead of reconfiguration.

A similar process is executed when performing partial reconfiguration, with the difference that the Instruction Register must then be written with the command for configuration, CFG_IN and the Data Register is then written with the pad frame for reconfiguration and the configuration data.

4.3. Extraction of the readback data

After reading back the bitstreams from the configuration memory, the required information must be extracted from the configuration data. In order to include register content in the configuration data which is read back, the "CAPTURE" primitive from Xilinx must be instantiated in the design [17]. When this primitive is added to the design and its capture signal is activated before readback, the register content will be written to the configuration memory and thereby included in the data which is read back.

After the system has read back configuration data from a specific part of the configuration memory, it must also manage to extract the required information out of the bitstream. Depending on if the system will use the readback data to perform self-tests, or if it shall use it to perform task migration, different parts of the data is useful.

In the first case, the realization of self-tests by exploiting readback can be done in two ways. Either the actual hardware configuration can be compared to the original configuration by the system itself (or by host PC during run-time) in order to determine if bit kips have occurred during execution. Or a functional test can be performed by extracting the register states out of the configuration data to analyze the contents of those registers to verify functional behaviour. If a comparison between the current configuration and the correct configuration data is to be performed, there are some parts of the data that can not be included in this comparison, such as the content of LUT RAMs or BRAMs. The reason for this is that the data in the original configuration includes the initial data of these components, while the readback data includes LUT RAM/BRAM data which has been modified during run-time. Therefore, only the actual hardware implementation can be analyzed this way. When performing a functional test by extracting the register content, the Capture function which is described above must be exploited before readback. Then, after reading back the configuration, the data which describes the register contents must be extracted and compared to the expected values.

In order to realize task migration by performing readback, the states of the corresponding registers must be extracted from the readback data, as in the case with the functional tests. This means that the system must initialize readback from a specific part of the configuration memory, read back the data internally over JCAP, analyze the data and extract the data which corresponds to the specific registers.

The run-time decoding of bitstreams for the Virtex 2 family is presented in [18]. Such a decoding process in order to localize the register content in a Spartan 3 bitstream can also be performed by the system itself during run-time.

5. Implementation and test of self-reconfiguration on a Spartan 3 FPGA

Here, a brief overview of the JCAP core will be given as well as some new results from the first realization of self-reconfiguration on a Spartan 3 FPGA. A structural overview of the core is given in Figure 4.

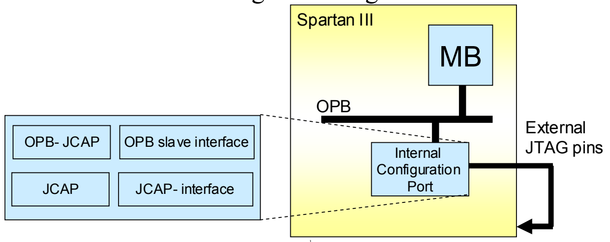

Figure 4. Structural overview of the JCAP core

There it can be seen that the outputs of the JCAP core are connected with the external JTAG interface. Here it can be mentioned that the JCAP and the JCAP interface includes the functionality for controlling the TAP controller of the external JTAG interface. These parts can be adjusted to provide the different functionalities of the external JTAG interface.

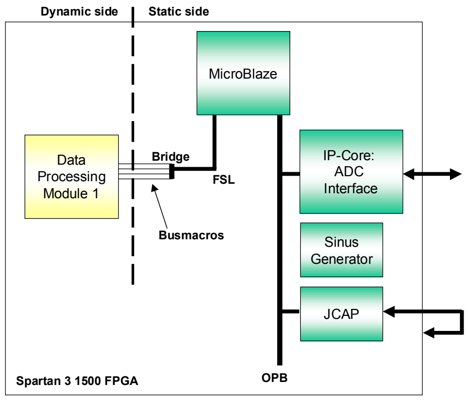

Figure 5. Self-reconfigurable system implemented on a Spartan 3 FPGA

In order to keep the resource utilization of the core low, the original implementation of the JCAP core only provided the functionality to perform partial reconfiguration.

In [11], the performance results of the core were presented and discussed. There it was clear that the reconfiguration rate of JCAP was lower than the one of ICAP because of the fact that ICAP allows byte-wise writes to the configuration memory, while the JCAP core must write the data serially to the JTAG interface.
Self-reconfiguration on a Spartan 3-1500 FPGA by exploiting the JCAP core has now been implemented in the reconfigurable measurement system. Figure 5 presents a structural overview of that system.
As can be seen there, the system is divided in static and dynamic parts and includes one reconfigurable slot for the partially reconfigurable function. The reconfigurable module which is seen in Figure 5 is the first data processing module for the dynamically reconfigurable measurement system.
The module communicates with the MicroBlaze-based [6] system over the Xilinx Fast Simplex Links, FSLs that are unidirectional links for direct communication between 2 components.

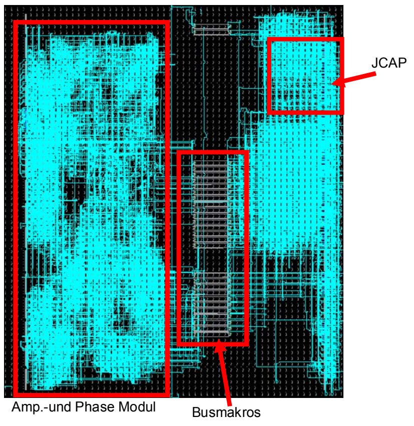

Figure 6. Spartan 3 self-reconfigurable system in FPGA Editor

MicroBlaze first reads the partial bitstream over the serial interface and then writes the bitstream to the JCAP core, which writes the configuration data over the external JTAG interface. The tests have that it is possible to perform self-reconfiguration even on Spartan 3 FPGAs. The additional design constraints that have already been discussed in Section 3 will however have to be considered. This includes the fact that only CBL wise reconfiguration is possible on Spartan 3 FPGAs. This further leads to the constraint that the reconfigurable area is reseted immediately after the reconfiguration, as has been discussed in section 3.
Figure 6 gives an overview of the reconfigurable system after implementation on the chip, viewed with the FPGA Editor tool [4]. There it is possible to see the partition between the static and dynamic sides of the system, as well as the reconfigurable function to the left on the FPGA. The static side includes the JCAP core which is

used for self-reconfiguration. The JCAP core can be seen marked in red in Figure 6.
The reconfiguration rate depends mainly on the rate with which MicroBlaze can write the data to JCAP over the IBM On-Chip Peripheral Bus, OPB [3]. The data processing module which is configured after the first module has a size of ca. 387 Kbytes. When the JCAP core is implemented with the maximum performance which can be achieved with the current implementation, the MicroBlaze would then require 1.58 s to perform reconfiguration, which is clearly too long. However, this reconfiguration time can be further reduced by pre-caching the bitstreams in an internal BRAM block, which allows JCAP to read the data directly from the BRAM. The bitstreams can be pre-cached since they are reconfigured according to a statically defined time schedule, and not on-demand. Furthermore, the final system will be based on a much smaller FPGA with more and smaller bitstreams.

6. Conclusions

This paper presents an approach for realizing internally controlled readback of configuration data on Spartan 3 FPGAs, by exploiting the internal JCAP core for controlling the external JTAG interface. Furthermore, this paper presents the first results from implementing self-reconfiguration over the JCAP core on a Spartan 3 FPGA system.

The JCAP core has some advantages over the fixed implemented ICAP core which is available on Virtex 2/4 and 5 FPGAs. Since the ICAP core is fixed implemented, it is normally more convenient to implemented the static part of the system on the left side on the FPGA, which may in some case lead to other design difficulties when considering the available I/O-pins on the FPGA. The JCAP core can be freely placed which simplifies the design of the static and dynamic parts and provide the designer an increased flexibility.

The readback technique which is presented here can be exploited for realization of system self-test (by verifying configuration data or by controlling register content) or for performing task migration by migrating register content between different functional implementations in the level measurement application. Future work will involve dynamic techniques for decoding the read back configuration data for Spartan 3 FPGAs and to integrate these techniques in the application.

7. References

[1] Xilinx; “Embedded Systems Tools Reference Manual”, UG111 (v6.0) June 2006.

[2] Xilinx; “Spartan III FPGA Family”, DS099 April 2006.

[3] IBM; “On-Chip Peripheral Bus, Architecture Specifications”, 2001.

[4] Xilinx; “ISE 7 Software Manuals”, 2005.

[5] C. Bobda, A. Ahmadinia, “Dynamic Interconnection of Reconfigurable Modules on Reconfigurabl Devices”, Design and Test of Computers, IEEE, 2005.

[6] Xilinx; “MicroBlaze Processor Reference Guide”, UG086 (v6.0) June 2006.

[7] Xilinx; „Platform Studio User Guide“,UG113 (v4.0) February 2005.

[8] M. Huebner, T. Becker, J. Becker: "Real-time LUT-based Network Topologies for dynamic and partial FPGA Self-Reconfiguration", SBCCI, Porto de Galinhas, Brasil, 2004.

[9] M. Ullmann, M. Hübner, B. Grimm, J. Becker, “An FPGA Run-Time System for Dynamical On-Demand Reconfiguration“, Proc. of the 11th Reconfigurable Architectures Workshop (RAW/IPDPS), April 2004.

[10] Xilinx: “Virtex-II Pro and Virtex-II Pro X FPGA User Guide”, ug012, v4.2, November 2007.

[11] K. Paulsson, G. Auer, M. Dreschmann, M. Hübner, J. Becker: "Implementation of a Virtual Internal Configuration Access Port (JCAP) for Enabling Partial Self- Reconfiguration on Xilinx Spartan III FPGAs", FPL 2007, Amsterdam, Netherlands.

[12] C. Carmichael, M. Caffrey, and A. Salazar, “Correcting single-event upsets through Virtex partial configuration”, Tech. Rep., Xilinx Corporation, June 1, 2000, XAPP216 (v1.0).

[13] B. Blodget, S. McMillan, P. Lysaght: "A lightweight approach for embedded reconfiguration of FPGAs", DATE 2003, Munich, Germany.

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

[15] P. Sedcole, B. Blodget, T. Becker, J. Anderson and P. Lysaght, "Modular Dynamic Reconfiguration in Virtex FPGAs", IEEE Proceedings Computers & Digital Techniques, May 2006.

[16] Xilinx; “Spartan-3 Generation Configuration User Guide“, UG332 (v1.3), November 2007.

[17] Xilinx, „Spartan-3 Advanced Configuration Architecture“, XAPP452 (v1.0), Dec. 2004.

[18] Michael Huebner, Lars Braun, Juergen Becker:"On-Line Visualization of the Physical Configuration of a Xilinx Virtex-II FPGA", ISVLSI 2007, Porto Alegre, Brazil.