Component-based Design of Embedded Systems with FPGA Support

Dušan Kolář*

Zbyněk Křivka*

Rudolf Schönecker*

{kolar, krivka, schonec}@fit.vutbr.cz

Abstract: This paper introduces conceptual development based on interconnection between tools that work with components on the conceptual level and FPGA development tools. Basic goal is to allow the development of FPGA on the conceptual level by design with the usage of components and event-driven programming. Concretely, we work on extension of Processor Expert RAD tool that allows an effective usage of FPGA in embedded system design. As supportive tools, Xilinx EDK and ISE are used.

Key Words: embedded system, FPGA, component, Processor Expert, Xilinx tools

1 Introduction

General design of hardware-software (HW-SW) systems focuses on efficiency and performance (see [1]). In reality, the speed of development and the design flexibility are more important. Fast development of low-end product is crucial for success on the market.

Fast development provides many errors and defects and some of them are not discovered until the real usage. Producers have to be able to update their products in SW (automatic updates) even in HW (automatic reconfiguration).

These opposing requirements (fast design and generality of the design) have led to the component-based development model or architecture-driven model in SW development (see [6]). This approach is applied to HW design and to the combination of HW and control SW too (see [2, 3, 5]). The reusability of the tested components speeds up the development of new embedded systems. In addition, post-production updates are possible thanks to the general design and auxiliary interface for update and/or reconfiguration. FPGA (Field-Programmable Gate Array) brings the possibility of HW reconfiguration and therefore better flexibility.

The basic idea is to focus on the description of the model of mutually interconnected components with their properties, methods, and events. The interconnection is implemented by events known from SW RAD tools. This idea of SW or HW component encapsulation is implemented for instance in Processor Expert (see [4, 7]). The unified approach to SW and HW components allows to work with CPUs, MCUs, memories, and peripherals in the same way (inspiration by object-orientation). On the other hand, general support for the usage of FPGA in this approach has not been introduced yet.

We introduce extended development concept based on the interconnection of tools, which work with the components on conceptual level and the tools, which are focused on development of FPGA-based systems (for instance Xilinx EDK and Xilinx ISE [8]). Our goal

* Department of Information Systems, Faculty of Information Technology, Brno University of Technology, Božetěchova 1, CZ-612 66 Brno

is to allow development for FPGA on the conceptual level with the higher abstraction using component-based design (components are called Embedded Beans) and event-driven programming. In practical aspects, it means to extend the actual version of Processor Expert by the encapsulation of some Xilinx components (called *IP cores*) for the components into Processor Expert's Beans (special HW-based component with unified object-oriented SW interface). The part of this encapsulation is the ability of generating of hardware and software platforms to implement Processor Expert's Beans in Xilinx tools, respectively into FPGA.

2 Technologies

Technique and implementation of the conceptual modeling is presented with the respect to two development environments: (1) RAD IDE Processor Expert by Unis Inc. (see [7]) for the embedded system development and (2) Xilinx Embedded Development Kit (EDK) with Platform Studio, Integrated Software Environment (ISE) with Project Navigator and other Xilinx tools for logical design of FPGA and CPLD devices.

2.1 Processor Expert

Processor Expert (PE) offers very high level of abstraction because it is the component-based object-oriented development environment for embedded system design (with 8, 16, 32-bits CPUs). It encapsulates HW elements into the components called Embedded Beans (EB). EB is an object-like component with some functionality (methods, events) and properties (parameters). The large EB database and EB's properties enable to set the HW without deep knowledge about the implementation of EB. EB's events are used for the treating of interrupts from peripherals. The EB database contains information for the parameters verification and the unified interface based on inheritance.

2.2 Xilinx Tools

The design flow of Xilinx tools (ISE and/or EDK) is quite variable. Both environments use the common command-line tools for the synthesis and bitstream generation. The resulting bitstream is downloaded into FPGA.

The lower level design in ISE is created in HDL language (VHDL, Verilog). The design is composed from the HDL elements, so the structure can be absolutely driven by a developer. This choice is the best for the minimalist design in HDL and for instance for the creation of new IP cores.

EDK is more complex with the higher level of abstraction. Platform Studio supports few boards with FPGA and two CPUs (PowerPC, MicroBlaze); uses target compilers, libraries of IP cores, and corresponding drivers. Then, HW design is composed of the IP cores (components) and controlled by the code for CPU. EDK is halfway step between ISE and full RAD tool.

Xilinx products strictly separate the graphical user interface and the command-line utilities that do the job. The configuration files of every HW-SW design are text files with information needed for the whole production cycle (up to synthesis and bitstream download into the device).

Standard design flow for FPGA consists of four iterative main steps (see Xilinx Design Flow part of Figure 1 or [9]):

1. Design Entry – create your design using Xilinx-supported schematic editor, text-base entry by hardware description language (HDL), or both.
2. Synthesis – convert schematic or textual description into the logical design (EDIF).

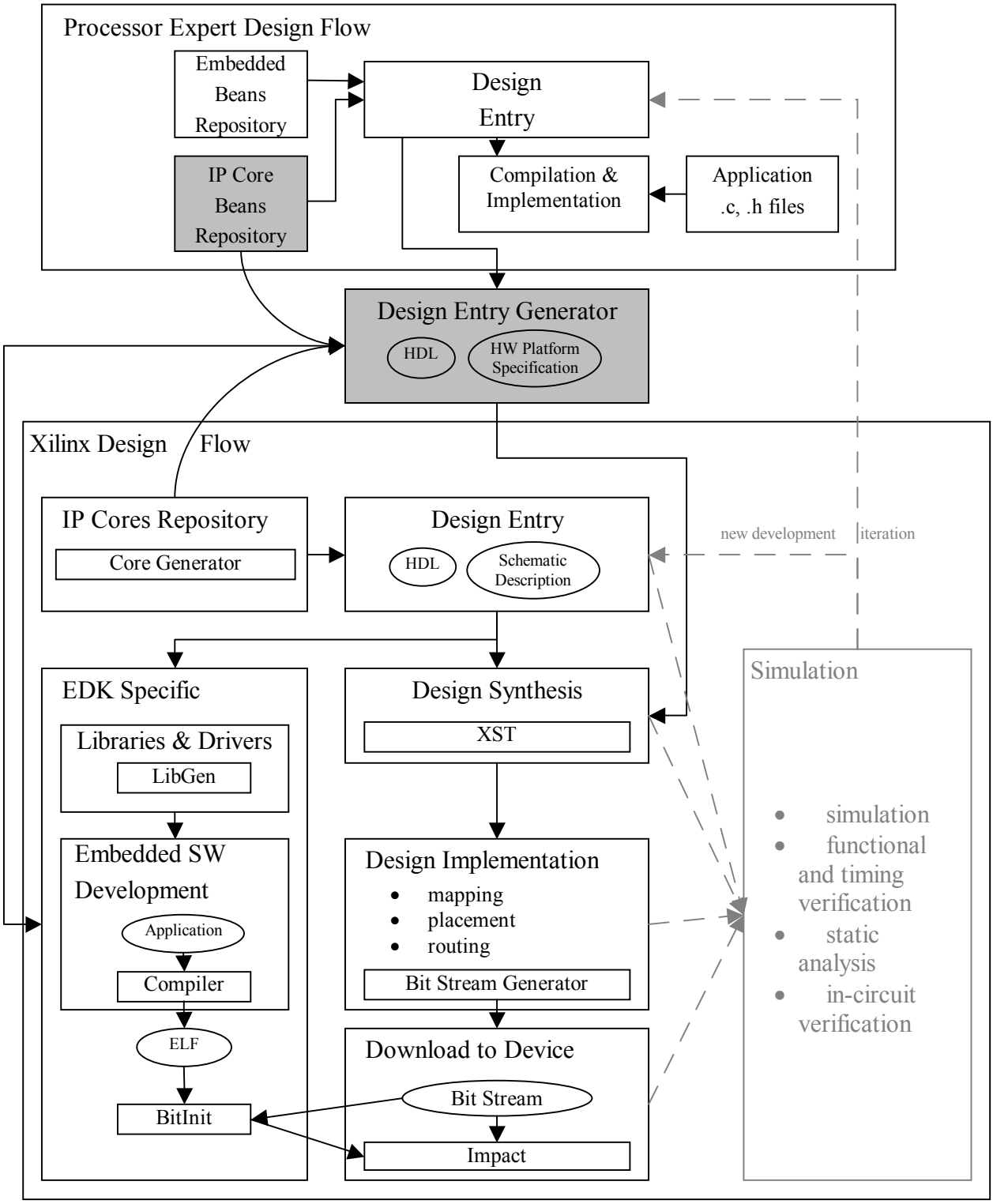

Figure 1: ***Combined Design Flow***

3. Design Implementation – transform logical design into physical format (native circuit description, NCD) specific for the concrete Xilinx architecture and generate bitstream for downloading into the device to configure it.

4. Design Verification – use the gate-level simulator or in-circuit debugging for the verification of your functional and timing requirements.

3 Conception and Draft of Practical Realization

There are four important challenges and requirements for the design of the new concept:

- Interconnect two design flows used by: (a) PE and (b) Xilinx ISE or EDK;
- Reuse of existing FPGA components (so called IP cores in Xilinx) in PE;
- Pick up PE as the master of the combined design and production flow to ensure encapsulation of the work with FPGA into EB;
- Enable parameters' modification and configuration before the usage of IP cores, or FPGA Embedded Beans, respectively.

The goal is to create the concept that uses (a) PE as primary (master) RAD development environment and (b) Xilinx tools (slave) for synthesis and implementation of some Embedded Beans from PE's design entry. In the concept, PE controls the design entry and implementation. Let us call EB with the target device in the FPGA implemented by IP cores in Xilinx tools as *IP Core Bean*. An addition of new functionality into the FPGA consists of the insertion of the corresponding IP Core Bean and the specification of its required properties (for instance EB for fast HW encryption, special mathematical operations, and DSP operations that require higher performance and other specific constraints).

3.1 Design Specification for Xilinx FPGA

Xilinx EDK design internal description is of form of the text-based configuration files (see MHS and MSS File Formats in [10]) that specify used IP cores and all parameters necessary for the interconnection of these IP cores in the FPGA implementation. In case of Xilinx ISE, the interconnection is not so highly abstract and HDL is used for it.

The main step of the realization of the concept is to create a command-line tool (*Design Entry Generator*) that is capable to assembly working FPGA system design based on the list of IP cores and their parameters given by IP Core Beans from PE part of the production cycle. The interconnection of the components is defined on high-level abstraction by the developer. In general, the full automatic connection of IP Cores is not possible. Therefore, the knowledge base with necessary information is introduced. For every supported IP core/component, it contains the lists of parameters, interconnection requirements, constraints, and other design conditions (for instance a mutually exclusive usage of some components, requirements on timers and clocks) to be able to achieve synthesizable and implementable design.

3.2 Processor Expert Design Flow with FPGA Support

The production cycle in PE has the following phases (see Figure 1; gray boxes are the new parts of the design flow and have to be implemented.):

A) Design entry (schematic or text-based, so called coding phase) by the developer who
   - chooses the target architecture (CPU, FPGA board);
   - adds required HW and SW Beans and sets their properties, generates (using PE)/writes the code for methods and event-handlers.

B) Incorporation of one or more IP Core Beans into PE Design Entry by developer who

- adds the Communication Bean (run-time bridge between PE's and FPGA's part of the design);
- adds one or more IP Core Beans that encapsulate the parameters of the corresponding IP Cores with the wanted functionality; sets the properties (some specify mutual interconnection in FPGA), generates/writes the code of methods executed before the computation in FPGA and of event handlers called after the FPGA computation.

C) Compilation and generation phase
- compiles the code of Embedded Beans for the target architecture;
- executes Design Entry Generator to transform (based on the knowledge base) settings of the Communication Bean and the IP Core Beans into Xilinx Design Entry (in case of EDK design flow into HW Platform Specification, and in case of ISE into HDL, respectively).

D) FPGA Design Synthesis, implementation and download to the device is made by Xilinx design flow; the obtained bitstream is downloaded into the physical FPGA board or used for the design simulation and verification.

In this moment, we have finished the generation of the program for the concrete CPU and downloaded the bitstream into the FPGA board. The application is ready to launch.

3.3 Run-Time Interconnection of Implementations by Combined Design Flow

The crucial design parts are the special components – FPGA Communication Bean in PE and IP Core in case of Xilinx FPGA device that arrange the run-time communication between the CPU (programmed by PE) and the FPGA (programmed by Xilinx tools) (see Figure 2). The communication is provided by the communication system that consists of a channel and a protocol chosen for the concrete design (e.g. UART).

To enable the design with more IP Core Beans implemented in an FPGA board at once, the communication components provide a simple communication multiplexing between the corresponding Beans and IP Cores in run-time.

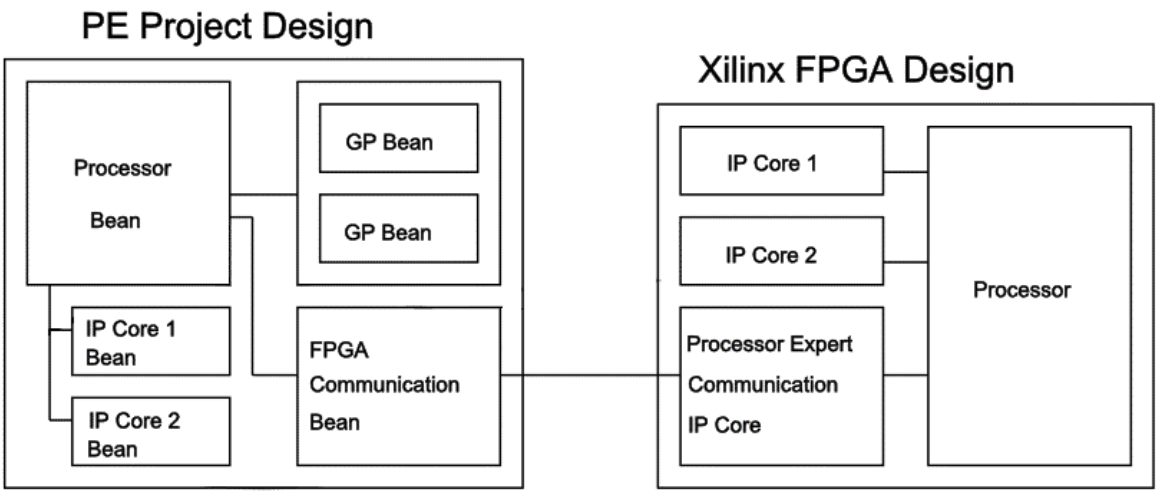

Figure 2: ***Run-time PE and Xilinx Interconnection***

4 Conclusions

In this contribution is introduced new basic concept of the interconnection of two technologies: (1) RAD tool Processor Expert and (2) Xilinx tools (EDK and ISE). The goal is to allow the usage of FPGA IP components with the special functionality in Processor Expert.

The general concept is designed for EDK and ISE tools. EDK is more suitable for the bigger designs that use mutual communication between the IP components driven by a full-featured processor (MicroBlaze, PowerPC). Then, we have to take account with the greater FPGA space demands due to the usage of buses and possibly a softcore processor. On the other hand, ISE is better for the smaller designs with simple functionality (e.g. the design with PicoBlaze, UART, shift-RAM-based register, GPIO) or for the usage with FPGAs with the lower number of logic cells.

The integration of the usage of FPGA IP components into Processor Expert provides the possibility of more effective designs. Without the modification of Processor Expert's design flow, the usage of this extension is absolutely transparent for a developer.

This work has been supported by the Czech Ministry of Education, Youth and Sports grant MŠMT 2C06008 "Virtual Laboratory of Microprocessor Technology Application".

References

1. Barr, M.: *Programming Embedded Systems*. O'Reilly, 1999, ISBN 1-56592-354-5.
2. Bližňák, M., Kolář, D.: Formal-method-based Software Development Applied on Embedded Systems: Basic concepts. In: 17*th International DAAAM Symposium* 2006, Vienna, AT, 2006, pp. 45-46, ISBN 3-901509-57-7.
3. Černý, S., Kolář, D., Stružka, P.: Processor Expert, Component Application Builder for Embedded Systems. In: *Proceedings of 5th IEEE Design and Diagnostics of Electronics Circuits and Systems Workshop*, Brno, CZ, FIT VUT, 2002, pp. 393-397, ISBN 80-214-2094-4.
4. Černý, S., Stružka, P., Kořenek, J., Martínek, T., Kotásek, Z.: FPGA Components in Simulink. In: *Proceedings of XXVIIIth International Autumn Colloquium* ASIS 2006, Ostrava, CZ, MARQ, 2006, pp. 158-163, ISBN 80-86840-26-3.
5. Kolář, D., Černý, S.: Evolution of Software for Embedded Systems in Processort Expert. In: *Proceedings of Eleventh IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, ECBS 2004, US, IEEE CS, 2004, pp. 419-422, ISBN 0-7695-2125-8.
6. Sommerville, I.: *Software engineering*, 7th Edition. Addison Wesley, 2004, 784 p., ISBN 0-3212-1026-3.
7. Unis, Ltd.: *Processor Expert On-Line Home Page* [online]. http://www.processorexpert.com, 2007.
8. Xilinx, Inc.: Xilinx: *The Programmable Logic Company* [online]. http://www.xilinx.com, 1994-2007.
9. Xilinx, Inc.: *Embedded System Tools Reference Manual:* 9.1i [online]. http://www.xilinx.com/ise/embedded/edk91i_docs/est_rm.pdf, 2007.
10. Xilinx, Inc.: *Platform Specification Format Reference Manual:* 9.1i [online]. http://www.xilinx.com/ise/embedded/edk91i_docs/psf_rm.pdf, 2007.