*The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Gora, Poland*

A SYSTEMATIC DEVELOPMENT OF VIRTUAL COMPONENTS COMPATIBLE TO STANDARD ICs (AN INDUSTRIAL EXPERIENCE)

Mirosław BANDZEREWICZ[*)], Wojciech SAKOWSKI[**)],
Włodzimierz WRONA[***)]

*) Evatronix S.A., ul. Dubois 16, 44-100 Gliwice, POLAND
tel./fax +48 32 231 11 71, e-mail: *mirek@gliwice.evatronix.com.pl*
http://www.evatronix.com.pl

**) Institute of Electronics, Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, POLAND
tel. +48 32 237 17 47, fax +48 32 237 22 25, e-mail: *sak@boss.iele.polsl.gliwice.pl*

***) Technical University of Łódz, Bielsko-Biała branch
Willowa 2, 43-308 Bielsko-Biała, POLAND

Abstract *Paper presents a proven methodology of development and productization of virtual electronic components. Methodology consists of rigorous approach to development of component specification, reverse engineering of behavior of reference circuits, application of industry-standard rules to coding of RTL model in a hardware description language and extensive testing and verification activities leading to (measured) high quality of hdl model and to FPGA prototype. In the final stage called productization a series of deliverables are produced to ensure effective reuse of the component in different (both FPGA and ASIC) target technologies.*

Key Words. *Virtual Components, IP Cores, Hardware Description Languages, high level design, quality assurance*

1. OBJECTIVE AND MOTIVATION

Objective of the effort described in this paper was to define a quality assurance policy for the development of the virtual components based on existing integrated circuits. At present our company specializes in cores compatible to 8-bit and 16-bit microcontrollers and microprocessors. Our methodology is based on the methodology recommended in [1], but it reflects to some extent peculiarities of our current profile as well as the fact that we have no access to certain EDA tools recommended in [1]. We found a lot of inspiration in the paper presented by SICAN company (now SCI-WORX) at the FDL'99 in Lyon [2].

The main motivation for the definition of a formalized methodology was to assure a high quality of the cores that we develop. Our first (bad) experiences in the development of a microcontroller core that was compatible to Intel 8051 chip [3] showed that lack of consistent and rigorous methodology results in a buggy core. Moreover, lack of a clear and complete specification turn the debugging of our first core into nightmare.

2. OVERVIEW OF THE METHODOLOGY

2.1. Design flow

Basic development steps in the creation of a virtual component include:

- Development of the macro specification,
- Partitioning the macro into subblocks,
- Development of a testing environment & test suite,
- Design of subblocks,
- Macro integration and final verification,
- Prototyping the macro in FPGA,
- Productization.

We will discuss these stages one by one below focusing on details related to our experiences.

2.2. Project management issues

At the beginning of a new project all the steps enumerated above are refined into subtasks and scheduled. Human and material resources are allocated to the project. Usually several projects are being realized in parallel. Therefore people, equipment and software have to be shared among these projects. We use MS Project software to manage scheduling of tasks and allocation of resources.

3. DEVELOPMENT OF THE MACRO SPECIFICATION

We use the documentation of an original device as a basis for specification of the core modelled after it. However, the documentation provided by the chip manufacturer is oriented towards chip users and it usually does not contain all details of chip behavior that are necessary to recreate its full functionality. Therefore analysis of the original documentation results in a list of ambiguities that have to be resolved by testing the original chip. The overall testing program is usually very complex, but the first tests to be written and run on a hardware modeler (see point 5) are those that provide resolve ambiguities in documentation.

At a later stage of specification we use an Excel spreadsheet to document all operations and data transfers that take place inside the chip. Spreadsheet columns represent time slots and rows represent communication channels. Such approach enables gradual refinement of scheduling of data transfers and operations up to the moment when clock cycle accuracy is reached. It reveals potential bottlenecks of the circuit architecture and makes easy to remove them at this early design stage.

4. PARTITIONING INTO SUBBLOCKS

Dataflow spreadsheet makes easier to define proper partitioning of the macro into subblocks. This first level of design hierarchy is needed to handle the complexity and to easier to divide design tasks between several designers. The crucial issue in this process is distribution of functions between the subblocks, definition of the structural interfaces and specification of timing dependencies between them.

5. TESTING ENVIRONMENT AND PROCEDURES

5.1. Testing the reference chip

As a reference for our virtual components we use hardware models that run on a (second hand) CATS hardware modeler (*Fig 1*). The hardware modeler interfaces over network to the CADAT simulator. The environment of the chip is modeled in C. Test vectors supplied from a file may be used for providing stimuli necessary to model interaction of the modeled chip with external circuits (e.g. interrupt signals).

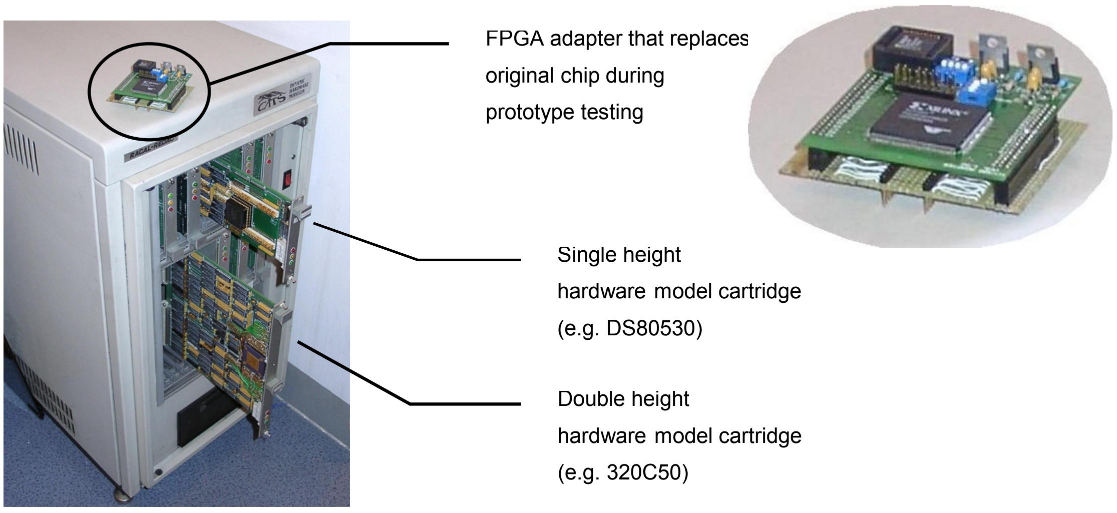
FPGA adapter that replaces original chip during prototype testing
Single height hardware model cartridge (e.g. DS80530)
Double height hardware model cartridge (e.g. 320C50)

Fig. 1 CATS hardware modeler

An equivalent testing environment is developed in parallel as a VHDL testbench to be run on VHDL simulator. We use Aldec's ActiveHDL simulator that proved to be very effective in model development and debugging phase. It enables us to import the testing results obtained with a hardware model into its waveform viewer in order to compare them against simulated behavior of the core under development.

5.2. Test suite development

Test suite development is based on the specification. Specification is analyzed and all the functional features of the core that should be tested against the original device are enumerated. Test development team (engineer) starts with development of tests that are needed to resolve ambiguities in available documentation of the chip to which a core has to be compliant.

Most of the functional tests are actually short programs written in assembly language of the processor that is modeled. Each test exercises one or several instructions of the processor. For instructions supporting several addressing modes tests are developed to check all of them. After compiling a test routine the resulting object code is translated to formats that may be used to initialize models of program memory in the testbenches (both in CADAT and VHDL environments). We have developed a set of utility procedures that automate this process.

In order to test processor interaction with its environment (i.e. I/O operations, handling of interrupts, counting of external events, response to reset signal) a testbench is equipped with stimuli generator.

5.3. Code coverage analysis

The completeness of the test suite is checked with code coverage tool (VN-Cover from TransEDA). The tool introduces monitors into the simulation environment and gathers data during a simulation run. Then the user can check what percentage of code statements was actually executed. More sophisticated measures like branch or path coverage may be also determined.

Incompleteness of the test suite may be a reason for leaving bugs in untested part of code [4]. Therefore we set a requirement to achieve 100% statement coverage during RTL simulation (i.e. each statement must be executed at least once during simulation of the complete test suite). Code coverage also helps to reveal redundancy of the test suite and sometimes the redundancy in the hardware design under test.

TransEDA State Navigator tool complements VN-Cover with special tools for verifying finite state machines. It may extract fsm from the VHDL source and present it graphically as state diagram. It also analyzes the results of simulation and shows what edges of the state diagram were taken or whether particular sequence of edges was exercised.

5.4. Automated testbench

Our cores are functionally equivalent to the processors they are compliant to, but they are not always cycle accurate. Therefore a strategy for automated comparison of results obtained with hardware modeler to those obtained by simulating RTL model was developed.

Scripts that control simulators may load the program memory with subsequent tests and save the simulation data into files. These files may serve as reference for post-synthesis and post-layout simulation. The testbench that is used for these simulation runs contains a comparator that automatically compares simulator outputs to the reference values.

6. SUBBLOCK DEVELOPMENT

The main part of the macro development effort is the actual design of subblocks defined during specification phase. For the moment we have no access to tools that check the compliance of the code to a given set of rules and guidelines. We follow the design and coding rules defined in [1]. We check the code with VN-Check tool from TransEDA to ensure that the rules are followed. Violations are documented.

For certain subblocks we develop separate testbenches and tests. However, the degree to which module is tested separately depends on its interaction with surrounding subblocks. As we specialize in microprocessor core development it is generally easier to interpret the results of simulation of the complete core than to interpret the behavior of its control unit separated from other parts of the chip. The important aspect here is that we have access to the results of the test run on the hardware model that serve as reference.

On the other hand certain subblocks like arithmetic-logic unit or peripherals (i.e. uarts and timers) are easy to test separately and are tested exhaustively before integration of the macro starts.

Synthesis is realized with tools for FPGA design. We use Synplify, FPGA Express and Leonardo. We realize synthesis with each tool looking for the best possible results in area-oriented and performance-oriented optimizations.

7. MACRO INTEGRATION

Once the subblocks are tested and synthesized they may be integrated. Then all the tests are run on the RTL model and the results are compared against the hardware model. As soon as the compliance is confirmed (which may require a few iterations back to subblock coding and running tests on integrated macro again) a macro is synthesized towards Xilinx and Altera chips and the tests are run again on the structural model.

8. PROTOTYPING

The next step in the core development process is building of a real prototype that could be used for testing and evaluation of the core.

For the moment we target two technologies: Altera and Xilinx. Our cores are available to users of Altera and Xilinx FPGAs through AMPP and AllianceCORE programs. In the near future we are going to implement our cores in Actel technologies, too. Placing and routing of a core in a given FPGA technology is realized with vendor specific software. The tests are run again on the SDF-annotated structural model. We developed a series of adapter boards that interface FPGA prototype to a system in which a core may be tested or evaluated.

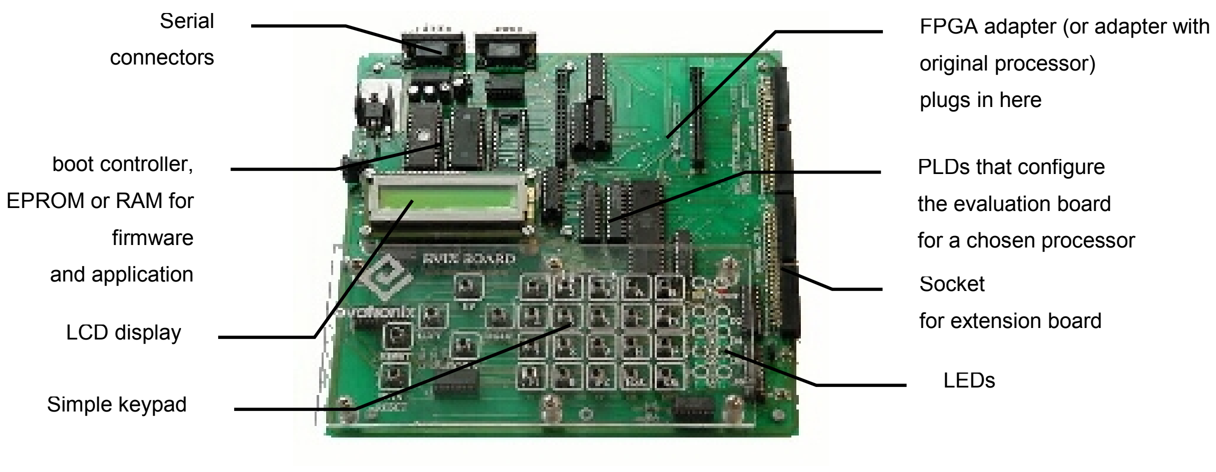
Serial connectors
boot controller, EPROM or RAM for firmware and application
LCD display
Simple keypad
FPGA adapter (or adapter with original processor) plugs in here
PLDs that configure the evaluation board for a chosen processor
Socket for extension board
LEDs

Fig. 2 Processor core evaluation board

The simplest way to test the FPGA prototype is to replace an original reference chip used in the hardware modeller with it. This makes possible to compare behavior of the prototype against the original chip. However for some types of tests even hardware modeller does not provide necessary speed. These tests can only be executed in prototype hardware system at full speed. Such approach is a must when one need to test a serial link with a vast amount of data transfers or to perform floating point computations for thousands of arguments. Our experience shows that even after an exhaustive testing program, some minor problems with the core remains undetected until it runs real-life application software.

For this reason we have developed a universal evaluation board (*Fig.2*). It can be adapted to different processor cores by replacement of on-board programmable devices and EPROMs. An FPGA adapter board (*see Fig. 1*) containing the core plugs into this evaluation board. An application program may be uploaded to the on-board RAM memory over a serial link from PC. Development of this application program is done by a separate design team. This team plays actually a role of an internal beta site, that reveals problems in using the core before it is released to the first customer.

The FPGA adapter board may also be used to test the core in the application environment provided that a prototype system exists. Such system should contain a microcontroller or microprocessor that is to be replaced with our core in the integrated version of the system. The adapter board is designed in such a way that it may be plugged it into the microprocessor socket of the target system. Using this technique we made prototypes of our cores run into ZX Spectrum microcomputer (CZ80cpu core) and SEGA Video Game (C68000 core), in which they replaced original Zilog® and Motorola® processors.

9. PRODUCTIZATION

The main goal of productization phase is to define all deliverables that are necessary to make the use of the virtual component in the larger design easy. We develop simulation scripts for Modelsim simulator and we run all the tests with this simulator to make sure that the RTL model simulates correctly with it. As we target FPGA market an important issue in productization phase is to develop all the deliverables for firm cores required by Altera and Xilinx from their partners participating in AMPP and AllianceCore programs.

Our foreign partners help us in developing synthesis scripts for Synopsys Design Compiler which we do not have access to. This deliverable is a must for customers targeting ASIC technologies. Synthesis scenarios for high performance and for minimal cost are developed.

We use VHDL during core development we but we translate our cores into Verilog, to make them available to customers that only work with Verilog HDL. The RTL model is translated automatically while the testbench have to be developed in Verilog manually (the translation tool is not able to translate all VHDL constructs into Verilog).

User documentation is also completed at this stage (an exhaustive, complete and updated specification is very helpful).

10. EXPERIENCES

The methodology described in this paper was defined over last few years during design of several versions of 8051-compatible microcontroller core [3].

It was then successfully applied to development of several virtual components compatible to Microchip PIC® 1657 microcontroller, to Motorola MC68000 16-bit microprocessor, to Zilog Z80 8-bit microprocessor and its peripherals, to TI® 32C025 digital signal processor as well as to VCs that implement controllers of standard serial links (I2C, SDLC and USB).

We continue to improve this methodology in order to turn it into a set of formal quality assurance procedures compliant to ISO 9000 requirements.

REFERENCES

[1] M. Keating, P. Bricaud, *Reuse Methodology Manual 2nd ed.*, Kluwer Academic Publishers, 1999

[2] J.Haase, *Virtual Components - from Research to Business*, Proceedings of the FDL'99 Conference, Lyon, 1999

[3] M.Bandzerewicz, W.Sakowski, *Development of the configurable microcontroller core*, Proceedings of the FDL'99 Conference, Lyon, 1999

[4] M.Stuart, D. Dempster, *Verification Methodology Manual*, Teamwork International, Hampshire UK, 2000