

Development of IP cores compatible to standard ICs

by Miroslaw Bandzerewicz, Wojciech Sakowski and Wlodzimierz Wrona, Evatronix

The methodology described in this article has been developed to create virtual components compatible to once-popular chips such as: 8051, MC68000, Z80, TMS32C025 and 80186.

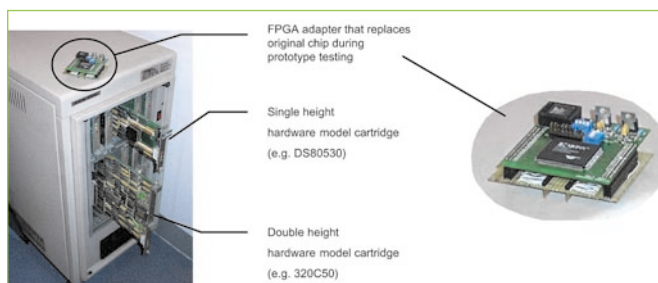


Figure 1. CATS hardware modeler

■ The dramatic increase in complexity of integrated circuits challenges the IC designer's productivity. The only way to take full advantage of what semiconductor technology offers is to reuse existing designs of functional blocks while designing more complex systems. The data format for description of such reusable designs may be different, but they share a common characteristic: from the point of view of their users they are components. But as they are sets of design data rather than actual pieces of hardware they are referred to as virtual components (macros). And as they represent their authors' intellectual property they are also called IP cores. This article presents a development methodology that Evatronix successfully applied to numerous cores compatible to once-popular chips. Basic development steps in the creation of a virtual component include:

- Development of the macro specification,
- Partitioning the macro into sub-blocks,
- Development of a testing environment and test suite,

- Design of sub-blocks,
- Macro integration and final verification,
- Prototyping the macro in FPGA,
- Productization.

We from Evatronix use the documentation of an original device as a basis for specification of the core modeled after it. The documentation provided by the chip manufacturer is chi-users-oriented and it usually does not contain all the 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. At a later stage of specification development we use an Excel spreadsheet to document all operations and data transfers that take place inside the core.

A dataflow spreadsheet makes it easier to define proper partitioning of the macro into sub-blocks. The crucial issue in this process is distribution of functions between the sub-blocks, definition of the structural interfaces and speci-

fication of timing dependencies between them. The main part of the macro development effort is the actual design of these sub-blocks. We check the code with VN-Check tool from TransEDA to ensure that the rules are followed. Violations are documented. Once the sub-blocks are written, tested and checked for acceptable synthesis results, they are integrated. Then the tests are run on the RTL model and the results are compared against the reference chip.

As a reference for our virtual components we use hardware models that run on a (second-hand) CATS hardware modeler. 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). An equivalent testing environment is developed in parallel in VHDL. Aldec's ActiveHDL simulator that we use 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. The test-bench that is used for simulation contains a comparator that automatically compares simulator outputs to the reference values.

Test suite development is based on the specification. The first tests to

be written and run on a hardware modeler are those that resolve ambiguities in documentation. Most of the functional tests are actually short programs written in the assembly language of the processor that is modeled. 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). 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.

The completeness of the test suite is checked with a code coverage tool (VN-Cover from TransEDA). The tool introduces monitors into the simulation environment and gathers data during a simulation run. Incompleteness of the test suite may be a reason for leaving bugs in untested part of the code. Therefore we pursue 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 under test. We also use more sophisticated coverage measures like branch or path coverage.

The next step in the core development process is building a real prototype that could be used for testing and evaluation of the core. For the moment we target two technologies: Altera and Xilinx. The tests are run again on the SDF-annotated structural model. We developed a series of adapter boards that interface the FPGA prototype to a sys-

a universal evaluation board. It can be adapted to different processor cores by replacement of on-board programmable devices and EPROMs. An FPGA adapter board 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

Main goal of the productization phase is to define all deliverables that are necessary to make use of the virtual component in larger designs easy. The scripts for different synthesis and simulation tools are developed in this phase. The core (that is normally developed in VHDL) is translated into Verilog. The deliverables for firm cores required by Altera and Xilinx from their partners participating in AMPP and AllianceCore third party IP programs are developed. User documentation is also completed at this stage.

The methodology described in this paper was developed over the last few years during design of several versions of a 8051-compatible microcontroller core. It was then successfully applied to development of other virtual components compatible to such devices as: Motorola MC68000 16-bit microprocessor, to Zilog Z80 8-bit microprocessor and its peripherals, to TI 32C025 digital signal processor and recently to Intel 80186 16-bit microcontroller and 80187 numeric coprocessor. With slight modifications we used it also while developing serial link controller cores (I2C, SDLC and USB). We are working now on formal documentation for this methodology and the design process itself in order to get prepared for ISO 9000 certification. ■

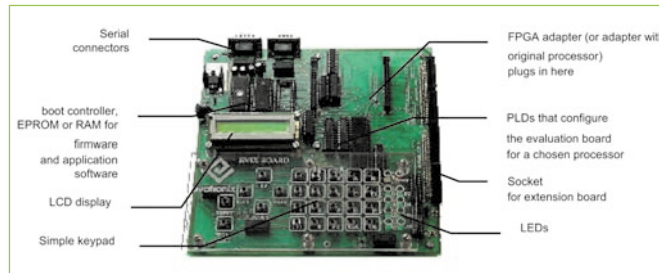


Figure 2. Evaluation board

tem in which a core may be tested or evaluated. The simplest way to test the FPGA prototype is to replace an original reference chip used in the hardware modeler with it. This makes possible to compare behavior of the prototype against the original chip. However, for some types of tests even the hardware modeler does not provide necessary speed. These tests can only be executed in the prototype hardware system at full speed. Such an approach is a must when one needs to test a serial link with a vast amount of data transfers or to perform floating point computations for thousands of arguments.

For this reason we have developed

PC. Development of this application program is done by a separate design team (in an other town). This team plays the 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 real application environment. The adapter board is designed in such a way that it may be plugged 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.