An Ad-hoc IP Functional Verification Procedure

A. K. Oudjid, D. Benamrouche, R. Tiar, M. Goudjil and A. Liacha
Centre de Développement des Technologies Avancée (CDTA), Algiers

This paper presents an ad-hoc IP-verification procedure that is used to verify the conformity between the IP specifications and their corresponding HDL-code implementation. The verification procedure is a general purpose solution offering an automatic validation to any IP design. The purpose of this paper is to provide a full description of the verification procedure and how to tailor it in order to fit any particular needs. The Ad-hoc verification procedure has been used to validate several designed IPs, notably: FIFO, Transceiver and I^{2}C-salve. The whole procedure code is implemented in both Verilog 2001 (IEEE 1365) and VHDL (2002).

1. Introduction

To ensure that the implemented design fully meets the initial specifications, the design must undergo two types of verification: at core level (cycle accurate test bench) and at board level (C-software test bench.)

1.1. Core level verification

Each unit of the architecture is tested separately. First, each unit is challenged against a set of severe special cases, and then against a very large number of random patterns. Once all units tested successfully, the same test process is repeated for the whole IP core.

For the ease of verification, a fully automated verification procedure (self-checking HDL test bench) is used (Fig. 1) For this purpose, we used the Unix Gawk tool to generate a parametrizable number of random-pattern files, which are submitted to both the synthetisable RTL code and to the behavioural test bench code for simulation. The simulator (Modelsim,), which runs in batch mode, performs a comparison between the delivered results and reports error if there is any. In case of an error, the Tcsh process is stopped and a visual simulation (wave mode) is performed on the responsible pattern-file to localize the bug. In the case when there is no error, the whole process is reiterated using Tcsh (Unix shell tool.)

1.2. Board Level Verification

The hardware (evaluation board) used for the SoC application is represented by the PCB of [1]. The system includes an ARM9TDMI as central 32-Bits CPU and different standard IO ports like 10/100 Ethernet and USB 1.1 together with a standard memory bus connecting SRAM, SDRAM and FLASH to the CPU. A free programmable FPGA for the implementation of approx. 30,000 gates allows any additional digital I/O interface. The system runs under a mini Real Time Operating System. A standard ARM based software development toolkit (assembler, compiler, debugging tools etc.,) is applied to the system. Concerning simulation of the total system, all modules are available in a common data base as HDL file or PLI file (for ARM9TDMI) as well as at the gate net list level in a readable or an encrypted version.

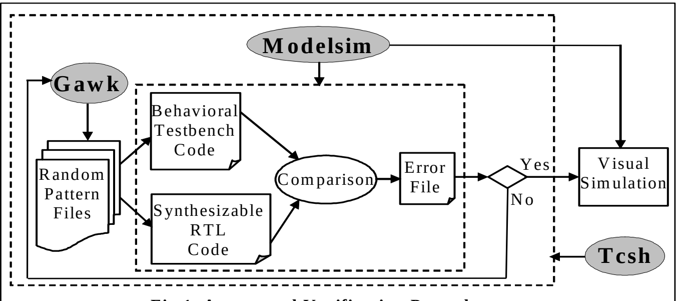

Fig 1. Automated Verification Procedure

This system has been used to validate our designed IPs (FIFO, Transceiver, I^2C) core at board level. We wrote a C interrupt–driven application using n-empty and n-full interrupts that verifies that the frame-transfers between a software FIFO and our I^2C transceiver FIFO are correct. To make the C application independent from the hardware, drivers have been developed

2. Concluding remarks

FV is indispensable. To be marketable, a design must be functionally correct and provide features required by its customers. But FV always takes at least twice as much effort as the design itself. This is why FV is currently the target of new tools and methodologies, which attempt to reduce the overall verification time by enabling parallelism of effort, higher levels of abstraction and automation [2].

Throughout this paper, we presented a self test ad-hoc FV procedure mainly based upon automation, allowing faster and predictable results. However, despite the general-purpose character of our FV procedure, automation requires standard processes with well defined inputs and outputs. Not all processes can be automated because of the variety of functions, interfaces, protocols and transformations. In such a case, it is always possible to use our ad-hoc FV procedure to automate some portion of the verification process, especially when applied to a narrow application domain.

References

[1] A. K. Oudjida et al., "Master-Slave Wrapper Communication Protocol: A Case Study," Proceedings of the 1st IEEE International Computer Systems and Information Technology Conference ICSIT'05, pp. 461-467, 19-21 July 2006, Algiers, Algeria.

[2] Janick Bergeron, *Writing Testbenches, Functional Verification of HDL Models* (Kluwer Academic Publisher, Copyright 2001).