Automated Functional Verification of Systems-on-Chip[1]

Zdeněk Přikryl, Marcela Šimková, Karel Masařík
Faculty of Information Technology, Brno University of Technology,
Czech Republic
{iprikryl, isimkova, masarik}@fit.vutbr.cz

Motivation

An increase of the complexity of systems-on-chip (SoC) induces an increase of the complexity of their verification as well. The reason is that we must verify not only the functions of separate logic blocks, but we need to check their interconnections, timing and functional collaboration as well.

The biggest amount of time in functional verification is consumed by:

- implementation of verification environments and testbenches,
- preparation of different test scenarios,
- preparation of reference models according to the specification,
- verification runs.

Therefore, there is still a great demand for verification tools, which are **TIME-EFFECTIVE**, **FAST** and as **AUTOMATED** as possible. Exactly these issues we target in our solution.

Proposed Solution

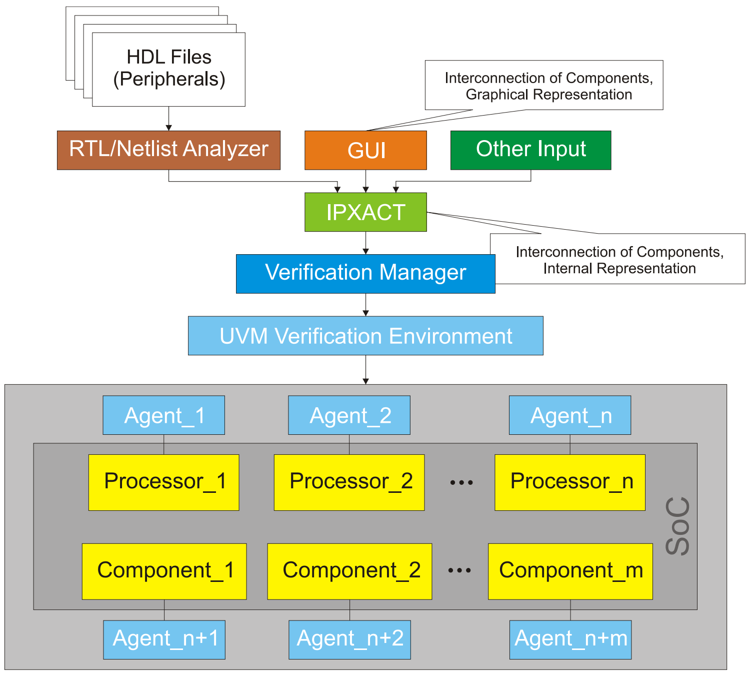
The aim of our work is to automatically generate up-to-date **UVM** (*Universal Verification Methodology*) verification environments [1] for SoCs, with simulation models serving as the reference models. The progress in verification runs is measured by the achieved level of functional, assertion and code coverage. For this purpose, we need specific information about the SoC under analysis, which we gain from one of the following approaches:

1) **Up-to-bottom** (orange part of the picture)**:** all components of SoC (processors, buses, peripherals) and their interconnections are specified at the higher level of abstraction, e.g. using Codasip Framework [2]. We have all information needed for verification at our disposal.

2) **Bottom-to-up** (brown part of the picture)**:** third-party components provided in RTL/Netlist form are automatically analyzed at first and then the important information for verification is extracted by our tools.

3) **Combination of above approaches or other user-defined approaches** (green part of the picture)**.**

The information needed for the generation of verification environments (performed in the *Verification Manager* block) is stored into the internal IP-XACT [3] representation.



References:

[1] Mentor Graphics Verification Academy. UVM/OVM. 2013. https://verificationacademy.com/topics/verification-methodology.
[2] Codasip Framework. Codasip. 2013. www.codasip.com.
[3] IEEE 1685. IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows. 2013. www.ieee.org.

[1] This work was supported by the European Social Fund (ESF) in the project Excellent Young Researchers at BUT (CZ.1.07/2.3.00/30.0039), the IT4Innovations Center of Excellence (CZ.1.05/1.1.00/02.0070), Brno Ph.D. Talent Scholarship Programme, the BUT FIT project FIT-S-11-2, and project no. MSM0021630528.