The methods for verifying an implementation with respect to the specified behavioural VHDL description are not sufficiently developed yet. Therefore the correct behaviour of the implementation has to be validated by simulation. Since it is impossible to simulate the implementation completely suitable simulation patterns have to be selected for executing statements, operations, functions, etc. of the implementation. In practice many designer use simulation patterns that reflect the adequate behaviour of the specification. But often these patterns do not fulfil completeness criteria and some functions may remain untested. In order to achieve a systematic determination and selection of the simulation patterns it is suggested to apply test generation methods at behavioural level.

The test generation approach GESTE is characterized by the
• generation of symbolic tests by solving a system of equations and inequalities,
• internal model comprising control and data flow,
• application of structure-oriented and software testing methods.

The resulting symbolic tests have the following features:
• test symbols for the element (corresponding to a VHDL statement) to be tested
• parameter for the other variables
• fixed values only if required for the path sensitization

The symbolic test refers to one run through the internal model. Therefore, the variables carrying the test symbols have to be set before and observed after the run-through. However, these variables can only be set and observed via the input and output ports, respectively.

The further approach includes the following steps:
• generating sequences to set the sensitized variables from the inputs and to propagate them to the outputs,
• concatenating these sequences and the symbolic test,
• substituting the test symbols by definite values,
• embedding the obtained simulation patterns into a test-bench for the SYNOPSYS simulator.

The generated simulation patterns are applied for validating the implementation. It has been turned out that a lot of test generation effort is caused by setting and observing the sensitized variables. The reason is that it is rather difficult to set the internal variables and to observe them from outside. Moreover, the long paths for setting and observing the internal variables lead to the fact that only a limited range of values fulfil all restrictions on these paths. In order to increase the effectiveness of the validation perceptibly the debug mode of the simulator is exploited. This mode permits an access to the internal variables by assign and by evaluate. The test generation has got simplified:
• sequences to set and to observe the sensitized variables are not required,
• hence, the concatenation of these sequences and the symbolic test becomes no longer necessary,
• restrictions arising from the long paths for setting and observing the internal variables are weaker.

These points result in shorter and less restricted symbolic tests. The observability of the test increases clearly recognizable. A waveform window supports the localization of design faults.

The described approach has been applied to some examples (greatest common divisor, simple calculator, MARK1). The expected effects e. g. simplifying test generation and shortening test sequences could be achieved.

Reference