Many industrial products are based on the use of embedded computer systems. Usually, these systems have to fulfill real–time requirements, and correct system functionality depends on their logical correctness as well as on their temporal correctness. Therefore, the developed systems have to be thoroughly tested in order to detect existing deficiencies in temporal behavior, as well as to strengthen the confidence in temporal correctness.

Existing test methods are not specialized in the examination of temporal correctness. For this reason, new test methods are required which concentrate on determining whether the system violates its specified timing constraints. Normally, a violation means that outputs are produced too early, or their computation takes too long. The task of the tester therefore is to find the input situations with the longest or shortest execution times, in order to check whether they produce a temporal error. It is virtually impossible to find such inputs by analyzing and testing the temporal behavior of complex systems manually.

However, if the search for such inputs is interpreted as a problem of optimization, evolutionary computation can be used to find the inputs with the longest or shortest execution times. We have developed and examined a new approach for testing temporal behavior which is based on the use of evolutionary algorithms, namely evolutionary testing [2].

For the optimization we employed extended Evolutionary Algorithms. This includes the use of multiple subpopulations, each using a different search strategy. Additionally, competition for limited resources between these subpopulations have taken place [1].

However, the use of Evolutionary Algorithms alone is not sufficient for a thorough and comprehensive test of real–time systems. A combination with existing test procedures is necessary to develop an effective test strategy for embedded systems which examines functional as well as temporal system behaviour. A combination of systematic testing and evolutionary testing is promising [2].

References
