An Interactive Dependability Driven Design Space Exploration

Stefan Scharoba, Jacob Lorenz and Heinrich T. Vierhaus
Brandenburg University of Technology
Cottbus, Germany
Email: scharst1@b-tu.de

I. Introduction

Due to the downscaling of transistor feature sizes, today's integrated circuits are much more likely to be affected by transient or permanent faults during their operational lifetime. In order to still meet certain dependability requirements, many different fault tolerance techniques have been developed, which can handle these faults in the field. Naturally, each of these techniques has a different effect on the system's dependability and is at the same time associated with distinct costs in terms of hardware overhead, additional power dissipation and performance degradation. As a consequence, deciding which techniques are used and to which parts of the system they are applied represents a crucial step in the overall design process. It must be accompanied by a careful exploration of different design choices including an estimation of their costs and benefits in order to find an implementation that meets the actual dependability requirements at acceptable costs. Performing these analyses manually can be very time-consuming and error-prone.

In this demonstration we present a prototype of IDDD, a tool that enables an interactive dependability driven design space exploration. Its main purpose is the automatic estimation of costs and benefits for selected fault tolerance techniques. Moreover these techniques can be automatically applied to a given VHDL model.

II. Interactive Dependability Driven DSE

Figure 1 shows how IDDD can be used during the design process. The starting point is a preliminary VHDL model of the system whose dependability needs to be improved. The VHDL description is parsed by the tool and the needed information, mainly concerning the system's structure, is stored in an internal data structure. This model serves as a first design candidate for the following design space exploration. With the help of the tool's graphical user interface, the circuit designer can apply different fault tolerance techniques that can handle transient and permanent faults by means of hardware redundancy. When selecting the techniques to apply, the designer has to make several decisions, e.g. which parts of the system are protected, how many spare components are added, and which online fault handling mechanism is used. The internal model is automatically modified according to these choices, resulting in a set of different design candidates.

In order to identify the most suitable implementation, the individual candidates have to be evaluated regarding costs and benefits of the applied fault tolerance methods. These analyses are performed automatically by the tool. Costs can be determined by generating the corresponding VHDL model of a design candidate and synthesizing it with the help of an external EDA tool. For this purpose, generic VHDL models and script templates for available EDA tools have to be provided. As a synthesis of a large number of design candidates can be very time-consuming, it is also possible to perform fast but less accurate estimations based on experience values. The benefit of a certain fault tolerance scheme is evaluated by computing different dependability measures like the reliability, the mean time to failure and the mission time as well as the corresponding improvement factors. These computations are based on the assumption of certain failure rates for the individual components, which have to be specified in advance. The system's behavior in case of component failures is modeled by Markov chains, which are automatically constructed and solved during the dependability evaluation.

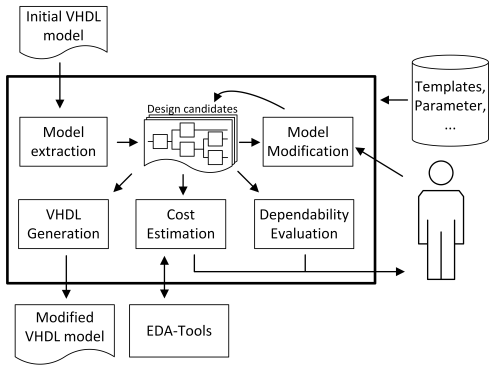

Fig. 1. The work flow of IDDD

Based on the results of the aforementioned evaluations, the designer can decide which implementation meets the actual requirements best or whether another iteration of modifying and evaluating the existing design candidates is necessary. Once the preferred implementation is found, the corresponding VHDL description can be generated by the tool to be processed in the following steps of the design flow.

Acknowledgment

This work was supported by the German Research Foundation (DFG) under the grant VI 185/16-1.