Chapter 6
An Aspect-Oriented Approach to Hardware Fault Tolerance for Embedded Systems

David de Andrés
Universitat Politècnica de València, Spain

Jaime Espinosa
Universitat Politècnica de València, Spain

Juan–Carlos Ruiz
Universitat Politècnica de València, Spain

Pedro Gil
Universitat Politècnica de València, Spain

ABSTRACT

The steady reduction of transistor size has brought embedded solutions into everyday life. However, the same features of deep-submicron technologies that are increasing the application spectrum of these solutions are also negatively affecting their dependability. Current practices for the design and deployment of hardware fault tolerance and security strategies remain in practice specific (defined on a case-per-case basis) and mostly manual and error prone. Aspect orientation, which already promotes a clear separation between functional and non-functional (dependability and security) concerns in software designs, is also an approach with a big potential at the hardware level. This chapter addresses the challenging problems of engineering such strategies in a generic way via metaprogramming, and supporting their subsequent instantiation and deployment on specific hardware designs through open compilation. This shows that promoting a clear separation of concerns in hardware designs and producing a library of generic, but reusable, hardware fault and intrusion tolerance mechanisms is a feasible reality today.

INTRODUCTION

Current embedded VLSI (Very Large Scale Integration) systems are widespread and operate in multitude of applications in different markets, ranging from life support, industrial control, or avionics to consumer electronics. Benefits of current manufacturing capabilities, in terms of attainable logic density, processing speed and power consumption, become threats to systems dependability, causing higher temperatures, shorter timing budgets and lower noise margins (Narayanan & Xie, 2006). In addition, deep-submicron technologies have both decreased the probability of manufacturing defect-free devices, and increased the likelihood of wear-out related problems and

DOI: 10.4018/978-1-4666-6194-3.ch006

the susceptibility to radiated particles (Constantinescu, 2003). Likewise, communications among devices expose hardware embedded systems to a number of external threats, especially when they are manufactured as an aggregation of off-the-self (OTS) Intellectual Property (IP) cores developed by third, and sometimes untrusted, parties. Nonetheless, reusing these components offers a reduction in time-to-market costs and a rapid integration of technology innovations while minimizing the risk of designs that integrate millions of gates (Rosenstiel, 2004; Vörg, 2003). It is unquestionable that critical systems require different degrees of fault and intrusion tolerance, given the human lives or great investments at stake. However, nowadays, the consideration of resilience in modern VLSI designs, understood as the ability of the system to ensure acceptable levels of dependability and security despite changes, is a requirement even in the industry of non-critical applications, as the occurrence of unexpected failures in consumer products may negatively affect the reputation of manufacturers and undermine the success of new products in the market.

The dependability and security communities widely accept that involving unskilled designers in the development of non-functional strategies (such as fault- and intrusion-tolerance and security) may actually have a negative impact on the global resilience of the deployed solution (Fabre & Pérennou, 1998). There is therefore an emerging requirement for frameworks supporting the separate design of non-functional and system core (functional) mechanisms, and their subsequent integration. In other words, fault and intrusion tolerance mechanisms must be developed by experts, but hardware designers with limited expertise in dependability and security must be able to integrate such mechanisms in their designs to make them resilient to faults and attacks.

How to support such separation of concerns during the design of dependable VLSI solutions remains an open challenge today. Aspect orientation (Kiczales, et al., 1996) provides interesting means to cope with this issue from the first steps of the system design flow, when integrated circuit models become available. The vast majority of modern solutions to digital circuit design revolve around the use of HDL (Hardware Description Language) models. Using such languages, hardware designers program circuits in a modular and hierarchical way. By modifying such models, related circuits can be accordingly adapted and evolved. The notion of metaprogramming, defining programs that automatically reason about and customize the structure of other programs, encompasses this idea. If this customization is specialized for fault tolerance (Taïani, Fabre, & Killijan, 2005), metaprogramming becomes a valuable technique to develop dependable strategies, which can be later (automatically and transparently) deployed onto HDL models following an open compilation process.

This chapter explains how an open compilation process can be established to support i) the implementation of fault tolerance and security techniques as metaprograms, and ii) their subsequent application to HDL-based models of integrated circuits. Additionally, this process must be seamlessly integrated into the regular design flow typically followed for HDL-based hardware systems, thus offering a great potential to increase the productivity of designers and reduce their error proneness. Other asset is that it can be applied as soon as a model is ready to simulate, even if it is not synthesizable yet. Hence, this opens the chance to study the impact of the applied modifications in the early stages of the design cycle, thus reducing the costs associated to late corrections. By enabling the automated integration of non-functional mechanisms and system functional mechanisms, the old idea of providing libraries of dependability and security mechanisms that could be reused in different contexts and deployed on different components could become a reality.

The rest of the chapter introduces first the basic concepts about aspect orientation and metaprogramming, and existing approaches to translate