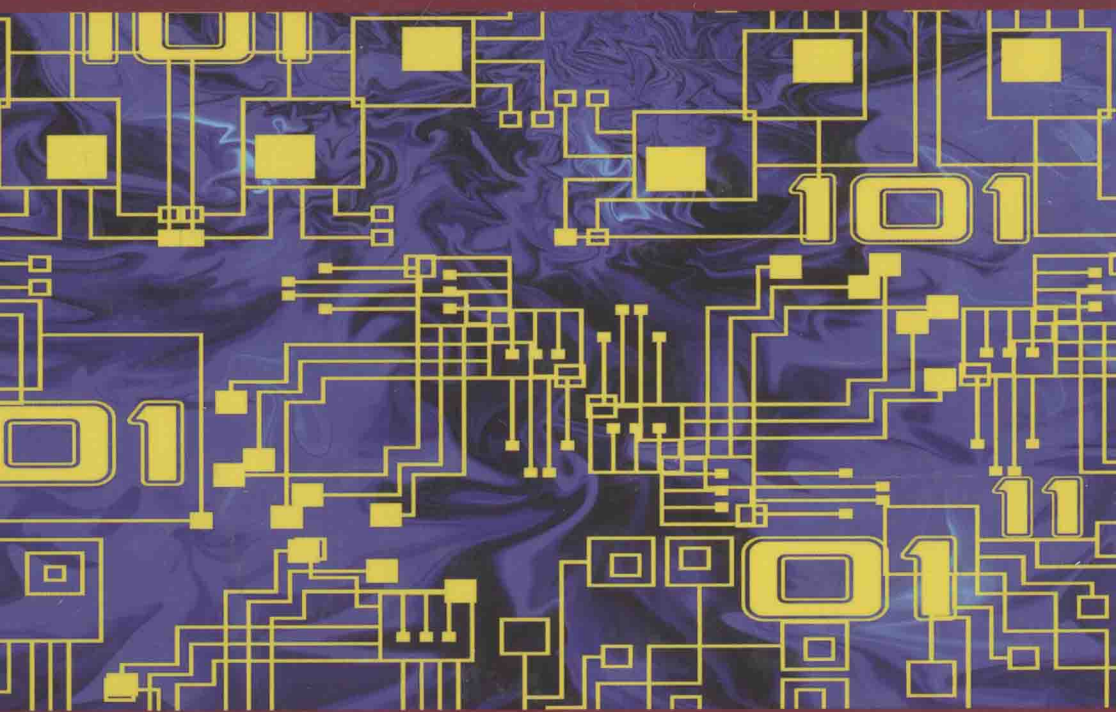
Safety of Computer Architectures

Edited by
Jean-Louis Boulanger



ISTE

WILEY

Safety of Computer Architectures

Edited by
Jean-Louis Boulanger

ISTE
WILEY

First published 2010 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.
Adapted and updated from *Sécurisation des architectures informatiques* published 2009 in France by Hermes Science/Lavoisier © LAVOISIER 2009

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Cataloging-in-Publication Data

Sécurisation des architectures informatiques. English
Safety of computer architectures / edited by Jean-Louis Boulanger.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-84821-197-1
1. Computer architecture. 2. Computer systems--Reliability. 3. Computer security. 4. Avionics--Safety measures. I. Boulanger, Jean-Louis. II. Title.
QA76.9.A73S4313 2010
005.8--dc22

2010016489

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN 978-1-84821-197-1

Printed and bound in Great Britain by CPI Antony Rowe, Chippenham and Eastbourne.

Safety of Computer Architectures

Introduction

In recent years, we have experienced an increase in the use of computers and an increase of the inclusion of computers in systems of varying complexity. This evolution affects products of daily life (household appliances, automobiles, etc.) as well as industrial products (industrial control, medical devices, financial transactions, etc.).

The malfunction of systems within these products can have a direct or indirect impact on integrity (injury, pollution, alteration of the environment) and/or on the lives of people (users, population, etc.) or an impact on the functioning of an organization. Processes in industry are becoming increasingly automated. These systems are subject to *dependability* requirements.

Today, dependability has become a *requirement*, not a concern (which was previously the case in high-risk domains such as the nuclear or aerospace industries), in a similar fashion to productivity, which has gradually imposed itself on most industrial and technological sectors.

Dependable systems must protect against certain failures that may have disastrous consequences for people (injury, death), for a company (branding, financial aspects), and/or for the environment.

In the context of systems incorporating "programmed" elements, two types of elements are implemented: hardware elements (computing unit, central processing unit (CPU), memory, bus, field programmable gate array (FPGA), digital signal processor (DSP), programmable logic controller, etc.) and software elements (program, library, operating system, etc.). In this book, we will focus on the safety of the hardware element.

Where the gravity and/or frequency associated with the risks is very important, it is said that the system is "critical". These "critical" systems are subjected to evaluations (assessment of conformity to standards) and/or certifications (evaluation leading to a certificate of conformity to a standard). This work is carried out by teams that are outside of the realization process.

This book aims to present the principles of securing computer architectures through the presentation of tangible examples.

In Chapter 1 the overall set of techniques (diversity, redundancy, recovery, encoding, etc.) for securing the hardware element of an architecture is presented.

For the railway transport field, Chapters 2, 3, 4, 5 and 11 present the applicable standards (CENELEC EN 50126, EN 50128, and EN 50129) as well as tangible examples (SACEM, SAET-METEOR, CSD, PIPC and the DIGISAFE XME architecture).

Chapters 6 and 7 will cover the field of aeronautics and outer space through three known examples, which are the aircraft from the AIRBUS Company, satellites and the ARIANE 5 launcher. The aviation field was one of the first to establish a referential standard that is currently composed of the DO 178 standard for embedded software development aspects, a trade referential consisting of a set of regulations FAR/JAR, applicable to all aircraft manufacturers and a set of methodological guides produced by the aviation community, ARP 45.74 and ARP 47.61. This referential has been recently complemented by the DO 254 standard, which applies to digital component aspects, such as FPGAs and other ASICs. The DO 278 standard applies to ground software aspects.

For automation-based systems, Chapter 8 presents examples of installations in the oil industry. The IEC 61508 standard allows for a definition and control of the safety objectives (SIL). Chapter 8 presents an opportunity to revisit this standard and its use. This chapter is supplemented by Chapter 10, which is a summary of the implementation of safety instrumented systems (SIS) in industry.

It should be noted that Chapter 12 provides an example of the implementation of a rather interesting automation-based system: the Large Hadron Collider (LHC).

Finally, in Chapter 9 we present examples in the automotive field. The automotive field is currently evolving. This development will result in the establishment of a variation of the IEC 61508 standard for the automotive industry called ISO 26262. This standard takes the safety level concept (called here the automotive safety integrity level, or ASIL) and identifies recommendations for activities and methodologies for implementation in order to achieve a given safety

objective. The automotive field is driven by different types of objectives (cost, place, weight, volume, delays, safety), which requires the establishment of new solutions (see Chapter 9).

It is hoped that this book will enlighten the reader as to the complexity of the systems that are used everyday and the difficulty in achieving a dependable system. It should be noted that this encompasses the need to produce a dependable system but also the need to guarantee the safety during the operational period, which can range from a few days to over 50 years.

Table of Contents

Chapter 1

Principles

1.1. Introduction

The objective of this chapter[1] is to present the different methods for securing the functional safety of hardware architecture. We shall speak of hardware architecture as safety can be based on one or more calculating units. We shall voluntarily leave aside the "software" aspects.

1.2. Presentation of the basic concepts: faults, errors and failures

1.2.1. *Obstruction to functional safety*

As indicated in [LAP 92], the functional safety of a complex system can be compromised by three types of incidents: failures, faults, and errors. The system elements are subjected to failures, which can potentially result in accidents.

DEFINITION 1.1: FAILURE – as indicated in the IEC 61508 [IEC 98] standard: a failure is the suspension of a functional unit's ability to accomplish a specified function. Since the completion of a required function necessarily excludes certain behavior, and certain functions can be specified in terms of behavior to avoid, then the occurrence of a behavior to avoid is a failure.

Chapter written by Jean-Louis BOULANGER.

1. This chapter is based on educational material produced together with M. Walter SCHÖN, Professor at the University of Technology of Compiègne, whom I can never thank enough.

From the previous definition, the need to define the concepts of normal (safe) and abnormal (unsafe) conduct can be removed, with a clear boundary between the two.

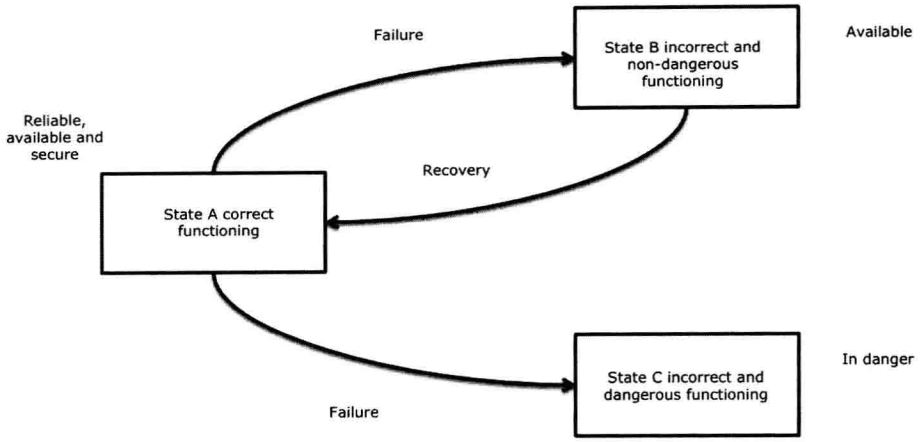


Figure 1.1. *Evolution of the state of the system*

Figure 1.1 shows a representation of the different states of a system (correct, incorrect) and the possible transitions between these states. The system states can be classified into three families:

– correct states: there is no dangerous situation;

– incorrect safe states: a failure was detected and the system is in a safe state;

– incorrect states: this is a dangerous, uncontrolled situation: there are potential accessible accidents.

When the system reaches a fallback state, there may be a partial or complete shutdown of service. The conditions of fallback may allow a return to the correct state after a recovery action.

Failures can be random or systematic. A random failure occurs unpredictably and is the result of damage affecting the hardware aspects of the system. In general, random failure can be quantified because of its nature (wear, aging, etc.).

A systematic failure is linked deterministically to a cause. The cause of the failure can only be eliminated by a reapplication of the production process (design, manufacture, documentation) or by recovery procedures. Given its nature, a systematic failure is not quantifiable.