George J. Milne Laurence Pierre (Eds.)

Correct Hardware Design and Verification Methods

IFIP WG10.2 Advanced Research Working Conference
CHARME '93
Arles, Frances, May 24-26, 1993
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
W-7500 Karlsruhe, FRG

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Volume Editors

George J. Milne
HardLab, Department of Computer Science, University of Strathclyde
Glasgow, G1 1XH, Scotland, UK

Laurence Pierre
Université de Provence
3 Place Victor Hugo, F-13331 Marseille Cedex 3, France

CR Subject Classification (1991): B.1.4, B.2.1-2, B.3.2-3, B.4.4, B.5.1-2, B.6.1, B.6.3, B.7.1-2

629,1

ISBN 3-540-56778-X Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-56778-X Springer-Verlag New York Berlin Heidelberg

Printed in Germany

Typesetting: Camera ready by author/editor
45/3140-543210 - Printed on acid-free paper

Foreword

These proceedings contain the papers presented at the "Advanced Research Working Conference on Correct HARdware Design MEthodologies" held in Arles (France) on 24–26 May, 1993 and organized by the ESPRIT Working Group 6018 "CHARME-2" and the Université de Provence (Marseille), in cooperation with IFIP WG10.2. The ESPRIT WG 6018 "CHARME-2" includes the following organizations:

- IMAG/Artemis, Grenoble (France)
- IMEC, Leuven (Belgium)
- Politecnico di Torino, (Italy)
- University of Frankfurt (Germany)
- Université de Provence, Marseille (France)
- University of Strathclyde, Glasgow (Scotland, UK).

Formal verification is emerging as a plausible alternative to exhaustive simulation for establishing correct digital hardware designs. The validation of functional and timing behaviour is a major bottleneck in current VLSI design systems, slowing the arrival of products in the marketplace with its associated increase in cost. From being a predominantly academic area of study until a few years ago, formal design and verification techniques are now beginning to migrate into industrial use. As we are now witnessing an increase in activity in this area in both academia and industry, the aim of this working conference was to bring together researchers and users from both communities.

The CHARME'93 working conference continued the series devoted to the development and use of formal techniques in digital hardware design and verification. Previous conferences have been held in Darmstadt (1984), Edinburgh (1985), Grenoble (1986), Glasgow (1988), Leuven (1989) and Torino (1991). The 1991 event was organized by the ESPRIT Basic Research Action 3216 "CHARME" and it gave the opportunity to present the CHARME results at the end of its second year. Similarly, presentations of the CHARME-2 achievements at the end of its first year were given at CHARME'93.

These proceedings reflect the structure of the conference, and are divided into sections which correspond to the conference sessions. Various research areas are represented by the 20 papers that were selected after review, thus covering many theoretical and practical aspects of formal hardware design and verification methods.

We are grateful to the Commission of the European Communities DG XIII, to the Université de Provence and to Sun Microsystems France S.A., who provided financial support for the conference.

March 1993

George Milne
Laurence Pierre

Program Committee

Dominique Borrione (IMAG/Artemis, France)
Paolo Camurati (Politecnico di Torino, Italy)
Luc Claesen (IMEC, Leuven, Belgium)
Hans Eveking (Univ. Frankfurt, Germany)
George Milne (Univ. Strathclyde, Scotland, UK)
Laurence Pierre (Univ. Provence, France)
Paolo Prinetto (Politecnico di Torino, Italy)
Bernard Soulas (EDF, Paris, France)

List of Referees

Einar Aas
Catia Angelo
Dominique Borrione
Paolo Camurati
Luc Claesen
Hélène Collavizza
Fulvio Corno
Solange Coupet
Hans Eveking
Mark Genoe
Stefan Hoereth
Peter Johannes
Enrico Maell
George McCaskill
George Milne
Jean-Luc Paillet
Laurence Pierre
Paolo Prinetto
Hans Samsom
Paul Shaw
Bernard Soulas
Jan Vandenbergh
Diederick Verkest
Eric Verlind

Contents

5. Asynchronous Circuit Design

6. Hardware Derivation

7. Use of Theorem Provers

A Graph-based Method for Timing Diagrams Representation and Verification

Viktor Cingel

Department of Computer Science and Engineering
Slovak Technical University, 821 19 Bratislava, Slovakia

Abstract. A graph-based approach to the verification of timing constraints in timing diagrams is described. Timing diagrams as a specification tool together with a specialized theorem prover for inequalities are used to support the timing design process. The method for automatically proving the consistency of a designed timing diagram works over a graph representing the timing diagram and constraints. It is based on the detection of cycles in this graph. A simple extension of this method enables the generation of a set of timing constraints which have to be fulfilled in the given timing diagram.

1 Introduction

In the design process of digital systems and circuits, a number of formal mathematical models for their description and the tools for automatic reasoning about their behaviour and implementation are being used currently. The functional verification seems to be the main task solved when designing a correct structural implementation from an external specification (usually expressed at the higher level). On the other hand, the timing and the way in which the system's interface has to operate in time are also very important design aspects [1,2]. Possible behaviours must satisfy both internal timing relations among building components and external timing requirements generated from the system's environment.

In general, the tools for the specification and verification of the system timing fall into four groups:

- tools supporting the timing analysis using critical paths detection [4,9,17],
- tools performing timing simulation over a set of predefined patterns [14],
- tools generating timing constraints directly from the structure of a circuit [5,6],
- tools producing mathematical proofs of certain timing properties [1,7,8].

The method proposed in this paper falls into the fourth group of the tools. Recently, different formal approaches are studied to be employed in the timing verification. For example, formalisms based on HOL or the Boyer-Moore logic [7,13], timed event structures [15], or timed CSP-based notations [11].

When designing and verifying timing, we usually abstract from the functional specification. As a result we obtain the worst case timing behaviour satisfying all the necessary timing constraints. Timing diagrams usually provided by manufacturers describe timing aspects of the behaviour of components (chips). To specify precisely the external system timing, ignoring functional relationships and internal details, we

first informally introduce a specification tool capable of describing timing diagrams. We use the min-max model for timing parameters which has proved to be sufficient for many practical applications. The formal reasoning on timing, which we apply in connection with a particular timing diagram, consists of two parts: proving theorems about the worst case behaviour in time, and the generation of timing constraints which must hold if the timing diagram is to be correct.

2 Timing diagram as a specification tool

2.1 Basic definitions

Time interval. The time interval T is defined as a finite interval of real numbers representing continuous time. Any event occurrence time falls inside T.

System variables. The digital circuit the timing of which is to be analysed is externally specified by a set of system variables $SV = \{x_1, x_2, ..., x_n\}$, each of them is of certain data type, denoted by Dx. Each Dx also contains an unknown (or unspecified) value u of a given data type. To denote the behaviour in time, each variable x is defined as a time function $x: T \rightarrow Dx$.

Events. An event is defined as an instantaneous value change which occurs in time at certain variable. For example, the rising and successive falling edge represent two successive events. Events can be of various types. The type of an event e for a variable x we define by $x<v_s, v_f>$, $v_s \neq v_f$, which denotes a value change from v_s to v_f of the x's data type. The most frequent event types are the following ones:

- $x<0,1>$ $up.x$ - the rising edge of a boolean variable x,
- $x<1,0>$ $dw.x$ - the falling edge of a boolean variable x,
- $x<u,V>$ $uV.x$ - the value change up to a valid numerical or symbolic value V,
- $x<V,u>$ $dV.x$ - the value change down from a valid value V.

In the sequel we use the abbreviated notation, e.g. $up.x$ instead of $x<0,1>$. By EV we denote the set of all events existing in a particular timing diagram we analyse.

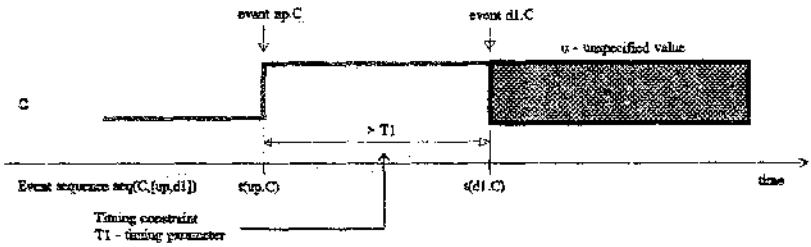

Fig.1. Basic items of a timing diagram