Analysis and Identification of Speed-Independent Circuits on an Event Model

M. KISHINEVSKY  
R&D Coop TRASSA, St. Petersburg, Russia, and Technical University of Denmark, Lyngby, Denmark

A. KONDRATYEV, A. TAUBIN, AND V. VARSHAVSKY  
R&D Coop TRASSA, St. Petersburg, Russia, and The University of Aizu, Japan

Received January 17, 1992; revised June 22, 1993

Abstract. The object of this article is the analysis of asynchronous circuits for speed independence or delay insensitivity. The circuits are specified as a netlist of logic functions describing the components. The analysis is based on a derivation of an event specification of the circuit behavior in a form of a signal graph. Signal graphs can be viewed either as a formalization of timing diagrams, or as a signal interpreted version of marked graphs (a subclass of Petri nets). The main advantage of this method is that a state explosion is avoided. A restoration of an event specification of a circuit also helps to solve the behavior identification problem, i.e., to compare the obtained specification with the desired specification. We illustrate the method by means of some examples.

Keywords: speed-independent circuits, delay-insensitive circuits, event models, signal graph, analysis, identification, verification

1. Introduction

Speed-independent and delay-insensitive circuits seem to be very promising in a number of VLSI applications, for example, in hardware interfacing. Such circuits, which do not use clocks for the ordering of operations, have well-known advantages. They are more robust, they have self-checking properties with respect to stuck-at faults, and they are potentially faster than synchronous circuits in highly parallel and iterative computations, since their correct operation does not depend on worst-case delays of the components [1–3].

The design of these circuits appears to be an intellectual problem requiring skill and experience. In a top-down approach, we start the design from a formal specification and proceed to develop a logic circuit, which implements this specification. Such an approach requires verification of the initial specification followed by a formal method of synthesis [3–6]. Another, bottom-up, approach consists of generating standard library modules, which can be used as building blocks for large-scale circuits [2, 7].

In the top-down design, analysis methods are used for performance evaluation [8] and for dynamic modeling of the circuits. Even if methods of formal design are correct by construction, we still need tools to browse through alternative
solutions and to choose the best one. In the bottom-up approach, analysis is a kernel of a design method: there is no guarantee that the circuit implementation we have arrived at is correct in both the functional and dynamic sense.

The purpose of this article is to present a method of analysis that can be used both in the bottom-up and the top-down design. In [9] we gave a brief discussion of this method. Objects of analysis are asynchronous circuits, specified by netlists of logic functions of components. Analysis problems can be divided in two parts: first, to test the circuit with respect to speed independence and delay insensitivity, and second, to perform behavior identification, i.e., to construct and check “all circuit behaviors” in order to recognize whether the circuit operates as requested.

The problem of identification can be decomposed into two independent sub-problems: reconstructing a behavior specification from a netlist of circuit components, and matching this specification with the desired one.

In the present article, we check the distributivity property to ensure the correctness of circuits. We formally define the distributivity property in section 2.3. This property is sufficient to guarantee the speed independence of a circuit behavior [10]. Distributive circuits are the most popular in a design [2, 8], since both analysis and synthesis of such circuits are much easier than in the general case of speed-independent circuits. It was proved in [2] that any distributive circuit can be implemented without hazards using the minimal basis of NAND gates or, dually, NOR gates. To implement nondistributive speed-independent circuits, one needs either a basis of complex AND-OR-NOT gates or a basis of NAND and NOR gates together. The distributivity property is a necessary but not a sufficient condition for the circuit to be delay insensitive. To test for delay insensitivity, we explicitly introduce auxiliary “WIRE”-components (i.e., buffers) into all wire connections that have to be checked.

The behavior of distributive circuits can be specified with an event-based model, called signal graphs, with only one type of causal relations (AND type): an event can occur only if all its predecessors have occurred.

Distributive circuits do not include circuits with OR-causal relations between signal transitions. Any circuit with an OR gate (or, dually, AND gate) has an OR-causal relation, and, hence, is not distributive, if concurrent transitions of the type 0 → 1(1 → 0) can occur at the inputs of the gate. A small example of a circuit with an OR-causal relation is given in figure 2. The parallel compression circuit (Figures 4.6 and 11.2 in [2]) used to perform a code membership test for a double-rail 4-phase communication is a practical example of a nondistributive speed-independent circuit. Circuits with a nondeterministic choice, like the arbiters, are also excluded. In subsection 8.2 we give an example of the SELECT element, where the internal implementation of the circuit is not distributive, and not even speed independent, but its behavior at the external terminals appears distributive. This can be viewed as a useful technique for dealing with the limitations of the class of distributive circuits, based on hiding a nondistributive behavior inside circuit modules.