Abstract

In this work, a logic design assistance system using Tokio is presented. Tokio is a logic programming language which is based on Interval Temporal Logic. Therefore, Tokio can specify both concurrency and sequentiality accurately and easily. In this system, the behaviors at both the algorithmic level and the register transfer level are given in the same language: Tokio. This is one of the most outstanding characteristics. In this paper, we mainly present a data path verifier at the register transfer level. This verifier is a core part of the assistance system.

1 Introduction

Presently, we are constructing a logic design assistance system using Tokio [1][2][3]. Tokio is a temporal logic programming language, adopted as a hardware description language in our assistance system.

We regard the logic design flow as follows.

First, the designers specify the algorithm of the behavior to be designed. Then, the behaviors at the register transfer level is derived from this abstract form by the designers. This derivation is assured by simulating the descriptions. The designers also give the structure of the data path to be designed. Although there have been many reports on synthesizing the data path automatically from the behavioral description [4], the derived data paths are not yet as satisfactory as those which are designed manually, are difficult to improve. On the other hand, the designers have images of the data path to be designed even at an early stage in the actual design process, because they have designed many similar circuits. The designers are constructing the frames of the data path while they are deriving the behavior at the register transfer level. Our strategy is to utilize positively the designers' experience. Then, the consistency of the two descriptions is checked automatically. Even though a good data path can be synthesized automatically from the behavior in the near future, the process of improvement will still have to be done manually. Therefore, an automatic consistency-check process is important.

In this design flow, the following three points are important for the behavior description language.

- The behavior is specified in an executable form.
- The behavior at both the algorithmic level and the register transfer level are given in the same language.

---

*This work is partially supported by Grand-in-Aid for Encouragement of Young Scientists, The Ministry of Education, Science, and Cultures, No.01700381
• Sequentiality and concurrency can be specified accurately and simply.

Tokio meets these requirements, and therefore provides smooth logic design assistance.

Structure of the Assistance System

The structure of the assistance system is shown in Figure 1.

The designers, at first, specify the algorithm of the behavior in Tokio. Then, the register transfer level description is derived. This derivation is assured by the already-developed simulator [3]. The register transfer level description is specified in RTL-Tokio which is a constrained form of Tokio. Both behaviors in Tokio and RTL-Tokio (Register Transfer Level Tokio) can be simulated on the same simulator.

The designers also give the structure of the data path in Prolog. Then, the “Data Path Verifier” verifies whether the behavior can run on the given structure. If there exists an error, the structure and the behavior are modified to rectify that error, and then they are verified again. Otherwise, the design proceeds to the lower level such as logic synthesis. In the process of verification, both the interval transition table and the facility usage table are derived, from which the control part is synthesized. As for the control part, this system is connected to our already-developed verification system [7].

Contents

Temporal logic programming language Tokio and RTL-Tokio is introduced in Section 2. Then the data path verifier is explained in Section 3. This verifier is the core part in the proposed assistance system. In section 4, this verifier is applied to an example and its result is presented. This paper is concluded in Section 5.

2 Temporal Logic Programming Language Tokio

2.1 Tokio

Tokio [1][2][3] is a logic programming language which is based on first-order Interval Temporal Logic [5]. Intuitively, Tokio is an extension of Prolog with an addition of the notion of time sequence. Therefore, Tokio can describe an algorithm as flexibly as in Prolog.

Tokio operator

The expression

\[ \text{head} :- p, q. \]

denotes that the goals p and q are executed in the same interval where the predicate head is defined. Concurrent execution is represented in this expression.

\[ \text{head} :- p \&\& q. \]

The chop operator \&\& illustrates the sequential execution of the two goals p and q. This operator divides the interval of head into the two