Chapter 13

FORMAL MODELLING OF ELECTRONIC CIRCUITS USING EVENT-B
Case Study: SAE J1708 Serial Communication Link

Yann Zimmermann\textsuperscript{1,2}, Stefan Hallerstede\textsuperscript{1}, Dominique Cansell\textsuperscript{2}
\textsuperscript{1}KeesDA, Centre Equation, 2 Avenue de Vignate, 38610 Gières; \textsuperscript{2}LORIA, BP 239, 54506 Vandoeuvre-Lès-Nancy Cedex

1. INTRODUCTION

This chapter presents a study of the SAE J1708 Serial Communication link described in [1]. The study is carried out in Event-B, an extension of the B method. The system is implemented and decomposed using step-wise refinement. We present how to derive with this method a cycle-accurate hardware model. The model of the communication link system is composed of an arbitrary, finite, number of identical components that run concurrently. The model contains synchronization of these components required to control access to the communication link. At the end of the refinement we obtain an implementable model of the components which is translated into VHDL. The generated VHDL design is synthesizable, meaning that the implementable B model is synthesizable as well.

The system is first described in an abstract way. Then, step by step, it is described in more detail by means of refinement. The refinement process permits to develop the system step by step. Each step introduces an aspect of the system. It allows us to cope better with the complexity of a model than if a model was obtained at once. Refinement adds detail to a model, distributes complexity of the system and of its proof. It also makes it easier to explain, and communicate to achieve a step by step validation of the model. The last refinement has to be close to an implementation. Particularly, we aim to obtain a model of the system in which the description of the behavior of each

\textcopyright 2004 \textit{Kluwer Academic Publishers}.
controller only refers to its local state. We say that the components are independent.

The system is described in terms of a collection of events. Events permit to model concurrency. When more than one event may occur at any one time, the choice of what event occurs is non-deterministic. The independence of components ensures concurrency between the controllers because it does not impose any order on their execution. For this study we have made the assumption that a component writing to the bus immediately stops writing to it, if it registers a conflict in correspondence with ([1, 5.2.2]: "or sooner if possible"). In this case, bus contention is always resolved. We refer to this idealized model as the "ideal model" and to the other model as the "probabilistic model". From the initial specification of the ideal model we have developed a bit-level implementation of the bus-access controller. We prove that this model is safe, in that bus contention is always resolved (without delay). The advantage of this approach is that we will understand the implementation of the (ideal) controller before taking into account the technical difficulties involved in the probabilistic model. Furthermore, we consider the ideal model to be of interest by itself since it is implementable and may be used in future systems (where all components comply with the idealistic requirement above).

2. GLOSSARY

The glossary defines the concepts that are modeled formally. It is an important step in analyzing the intent of a specification. While creating the glossary we remove inconsistencies in the use of technical terms in the standards model. These inconsistencies usually arrive in natural language specifications. Formalizing such a specification forces us to define all terms precisely, thus revealing inconsistencies.

2.1 The system

The system consists of a network of components which communicate via a bus. All components are connected to the bus by a local controller. They may send (or receive) messages to (from) the bus.

**Network**: The network interconnects all components of the system via a global bus.

**Node**: Components are connected to the bus by a node. The node is the controller part of a component. A node is a receiver or a transceiver. A transceiver is a transmitter or a receiver.

Etc…