Graphical Entry of FSMDs Revisited: Putting Graphical Models on a Solid Base

Thomas Müller-Wipperfürth Richard Hagelauer

Research Institute for Integrated Circuits
Johannes Kepler University Linz, Austria

Abstract

This paper discusses issues of graphical modelling of Finite State Machines with Datapath (FSMDs). Tools supporting the graphical entry of state based systems are usable by intuition, but need to be based on an exact definition of semantics of graphical elements. This paper proposes to define semantics of graphical models based on the hardware description language VHDL.

1 Introduction

The advantages of graphical design representations are widely accepted. This is shown, e.g., by recent EDA tools performing Text-to-Graphics ([1]) conversions. Nevertheless, graphical models of state based systems are often lacking a well defined semantics. This is due to apparently intuitive extensions to classical state diagrams. Section 1.1 illustrates that intuitive graphical models may lead to surprising results. The poster presents a definition of semantics of graphical models of FSMDs based on VHDL.

1.1 Related work

Classical state diagrams [2] do not provide means to calculate output values using arithmetic expressions, to immediately react to asynchronously changing inputs or to express hierarchy and concurrency. The Statecharts [3] visual formalism comprises a comprehensive definition for hierarchical and concurrent models. However, there are reasons why Statecharts is not widely available and accepted for EDA tasks. Firstly, it is admitted in [3] that concurrency under some circumstances may cause problems:

> *... difficult problem arise with the introduction of events that are generated within the Statecharts itself and are sensed in orthogonal components ...*

Secondly, applying the Statecharts formalism for the specification of synchronous FSMDs is too unrestrictive, as Statecharts regards state transitions be triggered by arbitrary (asynchronous) events.

Experiences with SpeedCHART [4] showed that graphically designed concurrent state machines are not translated into concurrent HDL code. SpeedCHART represents concurrent FSMDs within a single VHDL process! It disregards the consistency of a graphical model and the final system's behavior.

As a consequence the following major shortcomings motivate to put graphical FSMD models on a solid base: a) Semantics of graphical representations of concurrent state diagrams and accompanying communication mechanisms is not exactly defined. b) Semantics of hierarchical models is left unclear, especially when moving up and down within hierarchy. c) Automatically generated HDL is always optimized for synthesis. This may even modify the (graphical) design's behavior. d) Graphical models are sometimes inconceivably restricted. e) Generated HDL code is hard to read and to map to the original graphical model.

2 VHDL based semantics

VHDL is standardized by the IEEE [5] defining syntactical issues of the language as well as the exact simulation algorithm. Therefore a VHDL model has a standardized simulation semantics [6]. This paper proposes to construct a synthesizable VHDL model from a graphical representation. This VHDL code is not optimized in any sense but provides a clear semantics of what has been graphically designed and serves as a reference model for further optimization efforts.

2.1 Transition actions

State transitions and attached datapath actions are executed synchronously. Figure 1 shows conditions and actions in the graphical state diagram as well as in the corresponding VHDL[1] code. `C` denotes a condition expressed in VHDL, not containing any side effects, i.e., assignment statements. `TA` represents arbitrary VHDL code, containing sequential VHDL statements only. Thus no concurrency may be defined within transition actions. `TA` allows to assign values to registers which are computed from datapath registers R, input values X, and values of outputs of internal combinational blocks D. Entry- and Exit-Actions

[1] For the sake of brevity `begin/end` statements are omitted but are indicated by indentations.

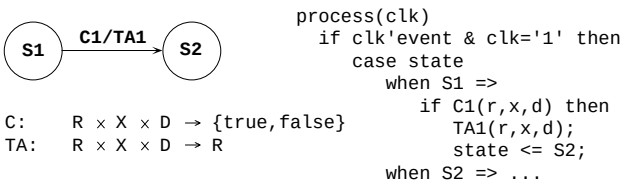

Fig. 1. Labels of transitions are composed of conditions (C) actions (TA) implementing register assignments.

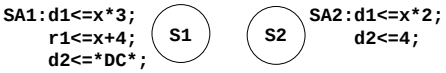

Fig. 2. State actions for asynchronous Mealy-type behavior.

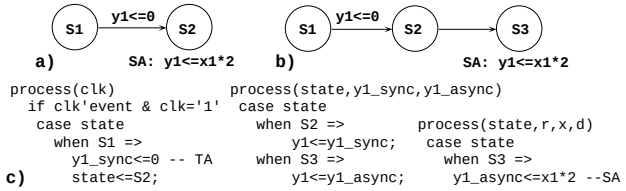

Fig. 3. Combined asynchronous and synchronous assignments: a) erroneous definition, b) valid combination, c) conflict resolution defined in VHDL-code.

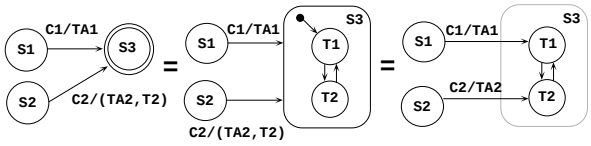

Fig. 4. Semantics of transition labels and hierarchical representations of transitions entering a hierarchical state.

of states are executed immediately before and after of transition actions, respectively. Synthesis tools will instantiate registers for assignments within transition actions. If an output should be recalculated on changing input values, a state action has to be defined.

2.2 State actions

State actions are asynchronously executed and define combinational behavior. Figure 2 illustrates that a signal value has to be preserved by means of a register (r1) if it is not specified for all possible states. However, d1 and d2 are purely combinational signals, as their values are explicitly defined for all possible states. *DC* denotes a don't-care value which achieves additional optimization potential as no register has to be instantiated for d2. A synthesizable don't-care value is not available for arbitrary VHDL data types. Hence the *DC*-value in the proposed graphical environment extends the possibilities of modelling.

2.3 Combining asynchronous and synchronous assignments

The combined application of asynchronous and synchronous assignments to a single signal may result in a conflicting situation as illustrated in Figure 3a. Figure 3b shows a valid assignment, because no conflict exists at any time. Experiments have shown that CAE tools may support synchronous as well as asynchronous assignments to a system's output but not their combined usage. As this is not a problem for hand-coded designs it should also be possible to graphically express this situation. Figure 3c shows the proposed solution and gives an exact definition of graphical semantics.

2.4 Concurrency and hierachy

Concurrent operation of components is an intrinsic feature of hardware and thus it is fundamental to any hardware description language to cope with parallelism. The proposed semantics of concurrent state machines heavily relies on this strength of VHDL.

Hierarchical models are well suited to achieve clearness and compactness of design representation. However, most hierarchical graphical representations have unclear semantics due to artificial entry/exit states which may only eventually consume state machine time depending on attached conditions. SpeedCHART considers flattening to be an optimization step which even may change a system's timing behavior. In our approach flattening serves as a definition of hierarchical representations. Hierarchical and flattened representations have to have identical behavior. The semantics of a hierarchical state is, as it is in Statecharts, the *exclusive-or* of its substates (Figure 4).

References

[1] Summit Design, Inc. *The Pinnacle*, 1997. Spring.

[2] J. Hartmanis and R. Stearns. *Algebraic Structure Theory of Sequential Machines.* Prentice Hall, 1966.

[3] D. Harel. STATECHARTS: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(4):403–414, 1987.

[4] Speed S.A. *SpeedCHART Reference Manual*, 1996.

[5] IEEE. *IEEE Standard VHDL Language Reference Manual*, 1994. ANSI/IEEE Std 1076-1993.

[6] Klaus Ten Hagen. *Abstrakte Modellierung Digitaler Schaltungen.* Springer, Berlin, 1995.