設 計 自 動 化 46－12
（１９８９．２．２１）

グラフィカル言語を用いるシンセシスシステムについて

Doron DRUSINSKY

ソニー（株）半導体事業本部

Statechartsという新しい言語の応用について発表する。これはハイレベルの動作記述が可能なグラフィカル言語である。従来のオートマトンの概念を発展させたもので，コンカレンシー，コミュニケーション，同期機能表現，階層化表現等の諸機能を視覚的に表現できるという特徴を有している。

この言語の一応用として，動作記述レベルで入力を受けとり，EDIF形式のネットリストを出力する様なシンセシスシステムについて提案する。

On the Synthesis of a New Graphical Language.

Doron DRUSINSKY

Semiconductor Group, SONY Corporation

4-14-1, Asahi-cho, Atsugi, Kanagawa, 243 Japan

Abstract

A new high-level graphical language called *statecharts* is presented. It extends Finite State machines with flexible concurrency, broadcast communication, hierarchy, visual synchronization mechanisms, and other features. A synthesis tool for statecharts augmented with a graphical hierarchical data language is developed. It maps the behavioral onto net-list EDIF description, and also transforms it to Sonys Astro system for logic-simulation and layout.

1. Introduction.

Automated synthesis systems usually emphasize on data-path synthesis with control treated as a by-product [Dm]. Such systems typically use simplistic single state machines to model the control-path. The two major drawbacks of such an approach are its sequentiality and flatness (finite state machines do not enable concurrency and hierarchy).

Statecharts have been proposed recently [H] as a visual formalism for the behavioral description of complex systems. They extend classical state-diagrams in several ways, while retaining their formality and visual appeal. In [DH87, DH89] we argued that statecharts can be beneficially used as a behavioral hardware description language, and we presented an efficient synthesis methodology for them. In [DH88] we presented some formal arguments about the power of various types of statecharts. Their formal definition appears in [HPSS,D].

In this paper we describe some details of our on-going statecharts synthesis project. First, in Section 2, we present a brief overview of the language and its power. In Section 3 we summarize the synthesis methodology of [DH89], and finally, in Section 4, we describe SYNTH, our synthesis project.

2. Statecharts Via a Traffic-Light Controller.

We shall not present a full description of the statecharts language of [H], but rather illustrate it via an example (see also [DH87]).

Fig. 1 describes the behavior of a traffic-light controller whose I/O interface is described in Fig. 2. There are two sets of lights: one is positioned over the main road (MAIN) entering the cross-junction, and the other is over the secondary road (SEC). During day time (D/N=1) the controller operates according to one of two possible programs: program A (PROG=1) gives two minutes to the vehicles in MAIN, and half a minute to the vehicles in SEC, alternatingly, and program B (PROG=0) gives half a minute to the cars in SEC once the SEC_FULL signal goes high. During the night (D/N=0) the controller gives precedence to the cars in MAIN until one of the two possibilities occurs: (1) two minutes have passed since MAIN became green and either a pedestrian wants to cross MAIN (PD_MAIN=1), or a new car appeared in SEC (NEW_CAR_SEC=1); or (2) three cars have already appeared in SEC. Once one of these conditions occurs, vehicles in SEC are given half a minute. The controller can be operated manually as well (A/M=0). In this mode, whenever POLICE becomes 1 (a policeman pushing a button) a transition is triggered from MAIN to SEC or vice-versa. This manual operation, and any transition from day to night and vice-versa, starts with 5 seconds of flashing yellow lights and then MAIN receiving the green lights. A hidden camera can be operated by the controller when it is in AUTOMATIC mode only. The camera will take a photo of the MAIN entrance to the junction, by producing the FMAIN signal, when MAIN is in the red state and a car enters the junction from MAIN (ENTER_M=1), and similarly for the SEC entrance (using the ENTER_S signal, and producing the FSEC signal). The controller can receive an ambulance signal (AMB=1), notifying the controller that an ambulance is approaching the junction from MAIN (DMAIN=1) or from SEC (DMAIN=0). It then synchronizes the lights according to the direction of the ambulance, and ignores all other events. Once the ambulance enters the junction the controller is notified (AMBJ=1), and returns to its previous operation mode, namely DAY or NIGHT. The controller can receive an error message (ERRIN=1) and then flickers both yellow lights. Another possibility for an error occurs when the controller operates manually for more than fifteen minutes without the policeman pushing the POLICE button, in which case the ERROUT signal is produced. A RESET signal resets the controller to the AUTOMATIC state.

In Fig. 1, we have *exclusive* states (e.g. DAY and NIGHT), and *orthogonal* states (e.g. AUTOMATIC and CAMERA). We have default entrances (e.g. the entry to WAIT within MANUAL), and entrances by *history* (e.g. from AMBULANCE upon the event AMBJ, returning by history only one level backwards, i.e. to AUTOMATIC or MANUAL, and then by default).

We have time bounds on the duration of being in a state (e.g. precisely 5 seconds in six of the states in LIGHTS, and at most 15 minutes in two of the states in MANUAL). We use conditional connectors (e.g. the entrance to AUTOMATIC dependant upon D/N), and uparrows and downarrows to turn condition changes into events (e.g. D/N $\uparrow$ is the event occurring when D/N changes from 0 to 1).

Actions can appear along transitions as in Mealy automata (e.g. β is generated when making the transition between two states of MANUAL, triggered by the POLICE event). They can also appear in states as in Moore automata, in which case, by convention, they are carried out upon entrance to the state (e.g. the red and green lights are cleared upon entering ERROR).

Some of the informal features of the language are: hierarchy, flexible concurrency, timing constraints specification, and visual synchronization specification. See [DH87,DH89,D] for details.

We have a computerized graphical system for the language of statecharts that enables drawings and simulations to be done with ease. The system also supports a data-path language that shows the flow of information within the designed system.

3. The Statechart Synthesis Method.

The following is an overview of the methodology of [DH89]. The methodology extends any conventional FSM synthesis method in a way which is isomorphic to the way statecharts extend FSM's. More specifically, the methodology generates a tree of communicating FSM. The tree structure implements statecharts' hierarchy, whereas concurrent FSM's at any level implement statecharts' flexible concurrency. Hence, the basic idea of our synthesis methodology is to trade the concept of a single machine implementing an FSM for a tree of interconnected machines implementing a statechart. Every state at every nonatomic level of the statechart hierarchy is represented by a machine implementing the immediate FSM one level lower. Fig. 3 is the tree of machines for the traffic-light controller of Fig. 1. The details of timing, inter-machine communication, and intra-machine structure, that implement hierarchy correctly appears in [DH89]. In [DH89] we were interested in layout techniques for the resulting tree of machines. In SYNTH however, we are interested in a netlist level output, hence layout techniques are not relevant.

4. SYNTH: The Statechart Synthesis tool.

SYNTH generates an EDIF netlist level description of the tree of machines of [DH89]. Each machine is implemented as a PLA with a state register. We use Berkeleys' state assignment programs [B] to generate reduced sized PLA's. The main reason for choosing EDIF was the desire to use (almost) standard tools. For practical use within Sony we use Sony's Astro system which enables logic-simulation and layout. Sony has completed the construction of a set of tools that convert EDIF descriptions into Astro descriptions and visa-versa.

Currently, our main emphasis is on a powerful control-path synthesis tool. Hence, data-path implementation is rather straight-forward. We enable the user to specify data cells out of a predefined set of Astro cells.

Most of the code is written in Lisp whereas format conversion programs are written in C with the aid of Lex. We expect the project to to consume about one year for a single engineer.

5. Future Plans.

We will conduct a precise comparison of the efficiency of our tool with existing tools (such as FSM implementation tools in [B]). These results will be published separately. We are also

planning the following extensions to the project:
1. Data-path synthesis. This can be done in house as well as interleaving our tool with an existing data-path synthesis tool.
2. Multiple-level logic implementation rather than the current two-level logic implemetation.
3. Optimization.

References.

[B] Brayton R. K., *et-al,* Oct Tools Distribution 2.1, Electronic Research Laboratory, U. C. Berkeley, March 1988.

[Dm] De Micheli G. and D. Ku. Sequencing Control Synthesis in the Hercules System. *Proc. 1989 IEEE Workshop on VLSI.*

[D] Drusinsky D., On Synchronized Statecharts, Ph.d. thesis, The Weizmann Institute of Science, 1989.

[DH87] Drusinsky D. and Harel D., Using Statecharts for Hardware Description, *Proc. IEEE conf. on CAD,* Santa-Clara 1987, pp. 162-165. (Also, Weizmann Institute of Science, CS85-06, December 1985).

[DH88] Drusinsky D. and Harel D., On the Power of Cooperative Concurrency, *Proc. Concurrency '88,* Lecture Notes in Computer Science **335** pp. 74-103, Springer Verlag, Hamburg, FRG, 1988. Submitted to JACM. (Also, Weizmann Institute of Science, CS88-10.)

[DH89] Drusinsky D. and Harel D., Using Statecharts for Hardware Description and Synthesis, *IEEE Transactions on CAD/ICAS,* To appear. (Also, Weizmann Institute of Science, CS87-11, July 1987.)

[H] Harel D., Statecharts: A Visual Formalism for Complex Systems, *Science of Computer Programming* 8 (1987), 231-274.

[HP] Harel D. and Pnueli A., On the Development of Reactive Systems, in *Logics and Models of Concurrent Systems,* (K. R. Apt, ed.), Nato ASI Series, Springer-Verlag, Berlin, 1985, pp. 477-498.

[HPSS] Harel D., A. Pnueli, J.P. Schmidt, and R. Sherman, On the formal Semantics of Statecharts, *Proc. 2nd IEEE Symp. on Logic in Computer Science,* Ithace, NY, 1987, pp. 54-64.

[P] Pnueli A., Applications of Temporal Logic to the Specification and Verification of Reactive systems: A survey of Current Trends, in *Current Trends in Concurrency* (de Bakker et al. eds.), Lect. Notes in Comput. Sci., Vol. 224, Springer-Verlag, Berlin, 1986, pp. 510-584.

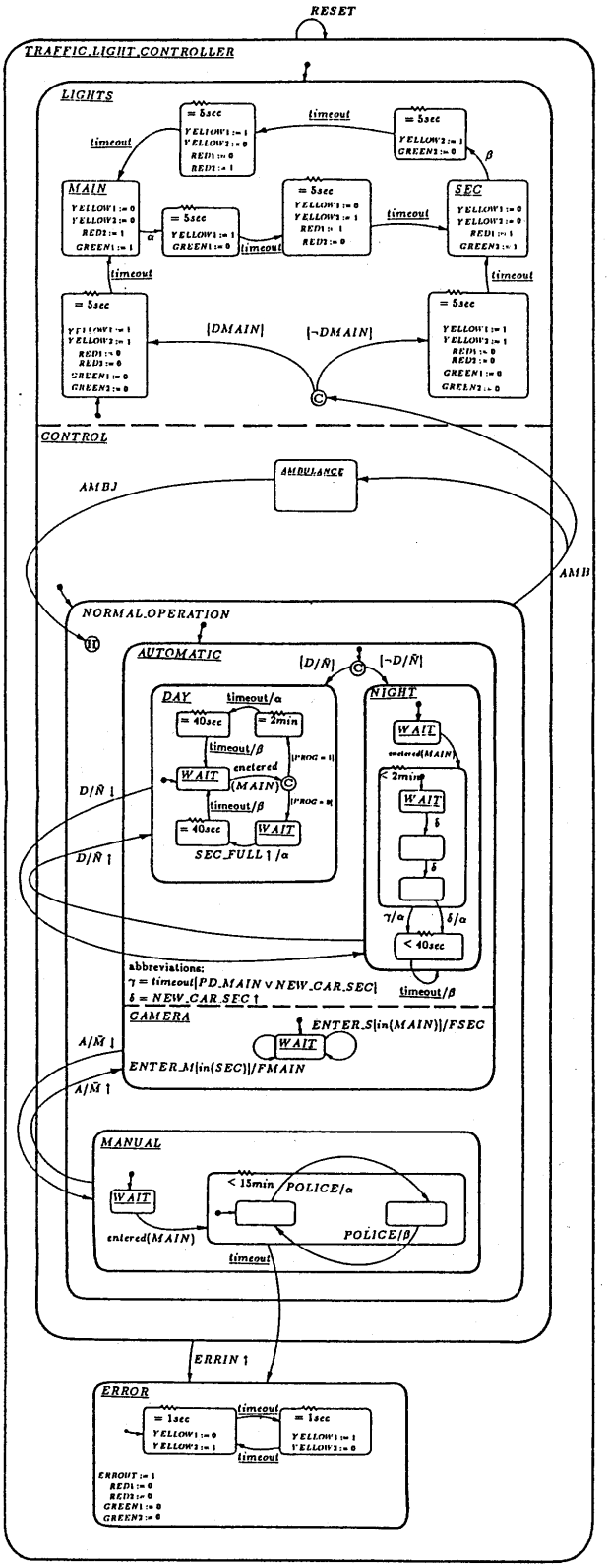

Figure 1. Statechart for the traffic-light controller

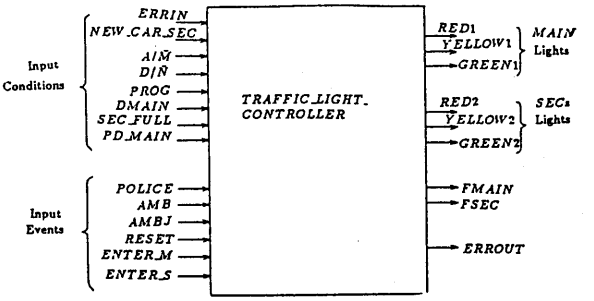

Figure 2. The I/O interface for the traffic-light

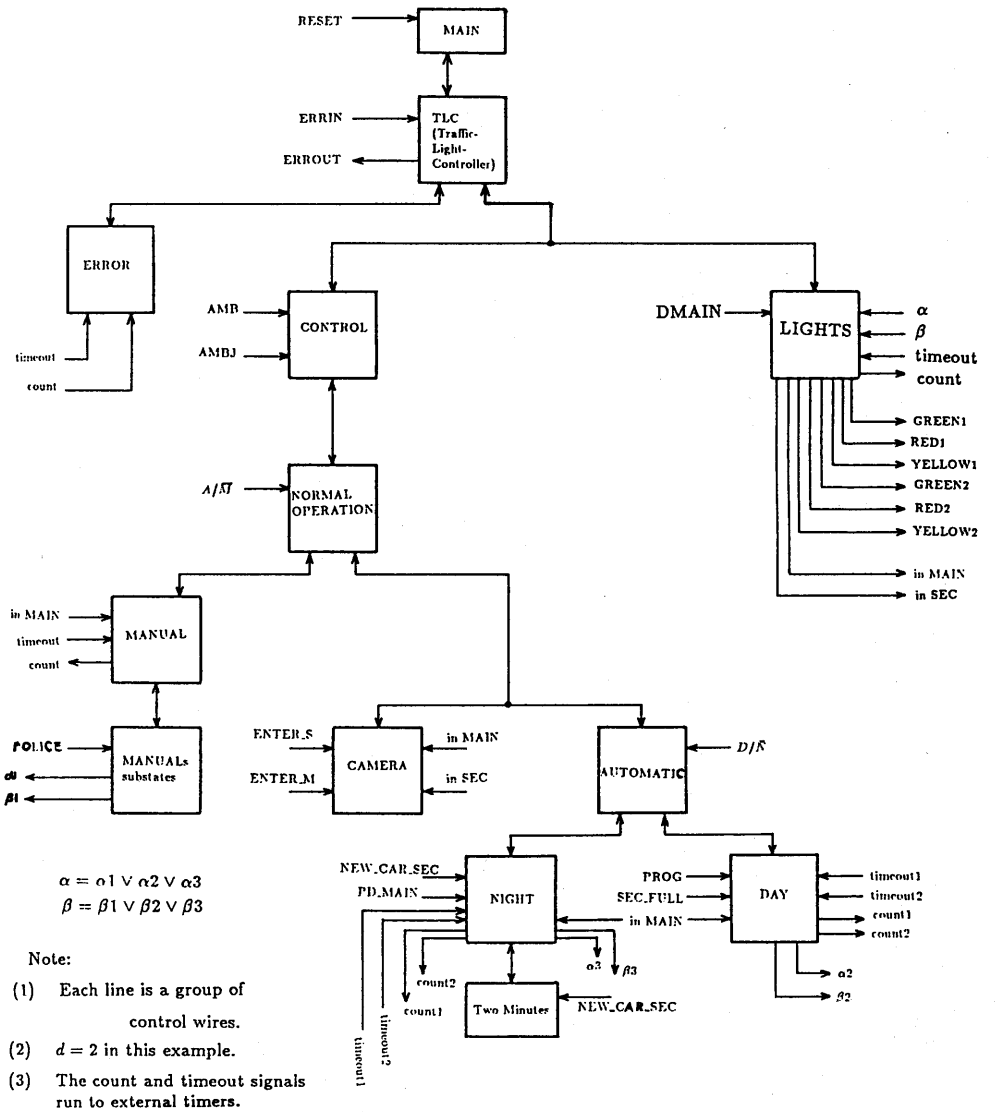

Figure 3. The machine-tree for the traffic-light controller of Figure 1