The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Gora, Poland

SYMBOLIC STATE EXPLORATION OF CONTROLLERS SPECIFIED BY MEANS OF STATECHARTS

Grzegorz ŁABIAK

Computer Engineering and Electronics Institute, Technical University of Zielona Góra,
ul. Podgórna 50, 65-246 Zielona Góra, POLAND, *G.Labiak@iie.pz.zgora.pl*

Abstract. *The FSM and Petri nets theories have elaborated many techniques and algorithms, which enable the employment of formal method in the fields of synthesis, testing and the verification. Many of them are based on symbolic state exploration. This paper focuses on the algorithm of the symbolic state exploration of controllers specified by means of statecharts. Statecharts are new technique for specifying behaviour of controllers, which, in comparison with FSM and Petri nets is enriched with notions of hierarchy, history and exception transitions. The paper presents the mathematical model of the diagram, its physical interpretation as a digital circuit and the characteristic function, which is the key notion in state exploration.*

Key Words. *Statechart, Logic Control, Symbolic Analysis, BDD*

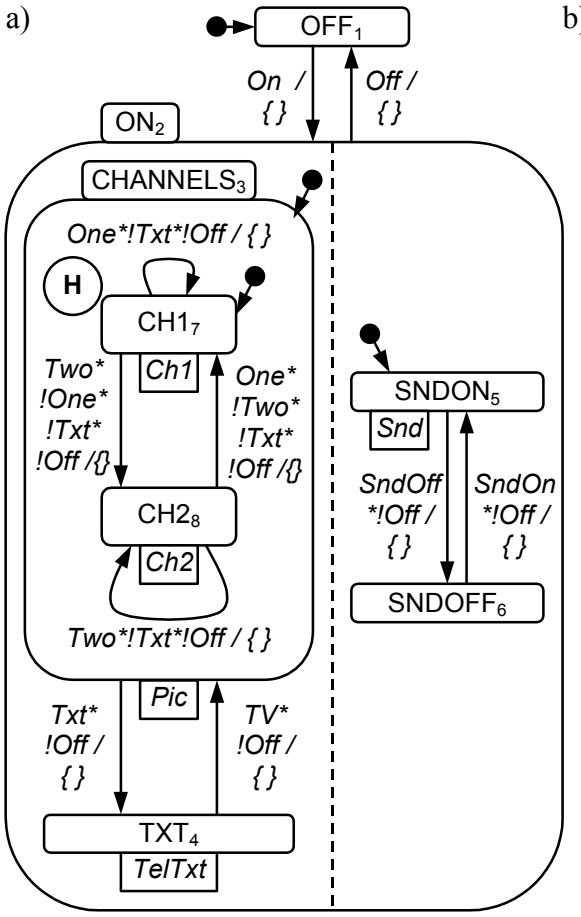
1. INTRODUCTION

Statecharts are a visual formalism for the specification of reactive systems, which is based on the idea of enriching state-transition diagrams with notions of hierarchy, concurrency and broadcast communication [6,7,8,10]. It was invented as a visual formalism for complex systems by David Harel [7]. Today, as a part of UML technology, it is widely used in many fields of modern engineering [11]. The presented approach features such characteristics as Moore's and Mealy's automata, history and terminal states. There are many algorithms based on a State Transition Graph traversal for finite state machines, which have applications in the area of synthesis, test and verification [2,3,4,5,10]. It seems to be very promising to use well developed techniques from FSM and Petri net theory in the field of synthesis [1], testing and the verification of controllers specified by means of statechart diagrams. These considerations caused the elaboration of the new algorithms of symbolic state space exploration.

2. SYNTAX AND DEFINITIONS

Based on the formalism contained in [8], the following definition of syntax can be given. Let S be the infinite set of states, T the infinite set of transition, E the infinite set of events. Symbols $s, s', s_1, s_2, s_3, \ldots$ are used to range over S, $t, t', t_1, t_2, t_3, \ldots$ to range over T, $e, e', e_1, e_2, e_3, \ldots$ to range over E.

a)



b)

$$[M_0\rangle = \{10000000,\ 10000010,\ 10000001,\ 01101010,\ 01100110,\ 01101001,\ 01100101,\ 01011010,\ 01010110,\ 01011001,\ 01010101\}$$

c)

$$[C_0\rangle = \{10000000,\ 01101010,\ 01100110,\ 01101001,\ 01100101,\ 01011000,\ 01010100\}$$

d)

$$\begin{aligned} X_{[C_0\rangle} = {} & s_1\bar{s}_2\bar{s}_3\bar{s}_4\bar{s}_5\bar{s}_6\bar{s}_7\bar{s}_8 \\ & + \bar{s}_1 s_2 s_3 \bar{s}_4 s_5 \bar{s}_6 s_7 \bar{s}_8 \\ & + \bar{s}_1 s_2 s_3 \bar{s}_4 \bar{s}_5 s_6 s_7 \bar{s}_8 \\ & + \bar{s}_1 s_2 s_3 \bar{s}_4 s_5 \bar{s}_6 \bar{s}_7 s_8 \\ & + \bar{s}_1 s_2 s_3 \bar{s}_4 \bar{s}_5 s_6 \bar{s}_7 s_8 \\ & + \bar{s}_1 s_2 \bar{s}_3 s_4 s_5 \bar{s}_6 \bar{s}_7 \bar{s}_8 \\ & + \bar{s}_1 s_2 \bar{s}_3 s_4 \bar{s}_5 s_6 \bar{s}_7 \bar{s}_8 \end{aligned}$$

Fig. 1. TV remote controller: a) statechart diagram, b) set of all global states, c) set of all reachable configurations, d) characteristic function $X_{[C_0\rangle}$

Definition 1 *Statechart*

A *Statechart Z* is a tuple consisting of the following elements:

$$(S_z, hrc_z, type_z, default_z, history_z, E_z, T_z, out_z, in_z, tlabel_z, saction_z)$$

where:

1. $S_z \subseteq \boldsymbol{S}$ is the finite non-empty set of states.
2. $hrc_z : S_z \to 2^{S_z}$ is the *hierarchy function*, which for every state $s \in S_z$ assigns the set of immediate sub-states of *s*.
3. $type_z : S_z \to \{AND, OR\}$ is the *state-type function*.
4. $default_z : S_z \to S_z$ is the *default function*.
5. $history_z : S_z \to \{true, false\}$ is the Boolean *history function*.
6. $E_z \subseteq \boldsymbol{E}$ is the finite set of *events*.
7. $T_z \subseteq \boldsymbol{T}$ is the finite set of *transition*.
8. $out_z : T_z \to S_z \setminus \{root_z\}$ is a total function, called *source function*, such that $out_z(t) = s$ if transition *t* originates from state *s*.
9. $in_z : T_z \to S_z \setminus \{root_z\}$ is a total function, called *target function*, such that $in_z(t) = s$ if transition *t* ends in *s* state.
10. For every transition $t \in T_z$, the following predicate holds:
    $parent(out_z(t)) = parent_z(in_z(t)) = s$ with $type_z(s) = OR$.

11. $tlabel_z : T_z \to 2^{E_z} \times 2^{E_z}$ is the *transition labelling function*. The first component of $tlabel_z$ is called $trigger_z(t)$, the second is called transition action and is denoted $taction_z(t)$.
12. $saction_z : S_z \to 2^{E_z}$ is the *state labelling* function, which gives the set of events associated to state *s*.

To use statecharts as a model for the specification of the digital controller it is necessary to give a real world interpretation of such notions as event, set of events or label. The following definition introduces the interpreted statecharts model. Based on this definition it is possible to use the statechart diagram as a mean of the specification of the digital controller or reactive systems.

Definition 2 *Interpreted Statechart*
An I*nterpreted Statechart* is a statechart as in Definition 1 where:

1. $X \subseteq E_z$ is a set of events coming from the environment, $Y \subseteq E_z$ is a set of events visible to the outside world
2. An event is a named signal that is either *present* or *absent*. *I* is a set of all signals in the system, both input, output and internal ones.
3. $input : X \to I_X$ where $I_X \subseteq I$ – is a function assigning the event coming from the environment to a signal. $output : Y \to I_Y$ where $I_Y \subseteq I$ – is a function assigning the event visible to the environment to a signal and $I_X \cap I_Y = \varnothing$. Signals related to events coming from the outside world and visible to the outside world are, respectively, the input and the output of the system. The sets of input and output events are disjoint.
4. Component $trigger_z(t)$ of the transition labelling function $tlabel_z$ called guard is a Boolean expression generated by the following grammar:
   `g ::= true | false | i | !g | g + g | g * g | (g)`
   where $i \in I$ is a signal associated to event $e \in E_z$. The evaluation of an event is either **true** or **false** when the event is either present or absent. The operators !, + and * correspond to the Boolean operators **not**, **or** and **and**, respectively.
5. Functions $taction_z(t)$ and $saction_z(s)$ lists a set of events $a \subseteq E_z$ associated with transitions and states respectively, according to the following rule:
   ```
   a ::= nil | b
   b ::= i | b, b
   ```
   where $i \in I$ is a signal associated to an event and "," distinguishes two events in an action.

It is essential from a symbolic technique point of view to express the concept of the set of states. The notion of characteristic function, well known in algebra theory, can be applied [2].

Definition 3 C*haracteristic function*
A characteristic function X_A of a set of elements $A \subseteq U$ is a Boolean function $X_A : U \to \{0,1\}$ defined as follows:

$$X_A(x) = \begin{cases} 1 \Leftrightarrow x \in A, \\ 0 \ otherwise. \end{cases} \quad (1)$$

The characteristic function is calculated as a disjunction of all elements of *A*. Operations on sets are in direct correspondence with operations on their characteristic functions. Thus:

$$X_{(A \cup B)} = X_A + X_B;\ X_{(A \cap B)} = X_A * X_B;\ X_{(\overline{A})} = \overline{X_A} \quad (2)$$

The characteristic function allows sets to be represented by BDDs. Fig. 1d presents the characteristic function of all possible configurations [2].

3. MODELLING SYNCHRONOUS INTERPRETED STATECHART BY BOOLEAN EQUATIONS

The modelling of statecharts is based on the assumption that for every state $s_i \in S_z$ one flip-flop is assigned, and then for every such flip-flop excitation function, as a Boolean expression, is produced, Fig. 2. The excitation function δ evaluates to 1 when the flip-flop associated with s_i will be active in the next iteration or remembers past activity, otherwise it equals 0. A state is said to be active when every state belonging to the path, carried from it to the root state, is active. Global state G of the system, called marking, is represented by the set of all states of flip-flops. A configuration C is a set of all active states. The excitation function $\delta_i(S_z, I)$ is defined on signals and current states of flip-flops. A detailed description of the creation of the functions is beyond the scope of this paper and the method developed by the author will be published soon.

Let Z be a synchronous interpreted statechart and Ω the set of all possible markings of Z. Each marking of Z can be coded as a vector $M_{1\times n} = (\mu_1, \mu_2, \ldots, \mu_n)$ where $\mu_i \in \{0, 1\}$ represents the activity of flip-flop representing a state $s_i \in S_z$ and n is a number of all states in S_z. The set of all reachable markings from default marking M_0 is denoted $[M_0\rangle$. Firing of a transition t_k transforms a marking M_i into marking M_j. This fact is denoted by $M_i[t_k\rangle M_j$. It is possible to fire a set of enabled transitions in a given moment of discrete time. Any set of markings can be represented using its characteristic function.

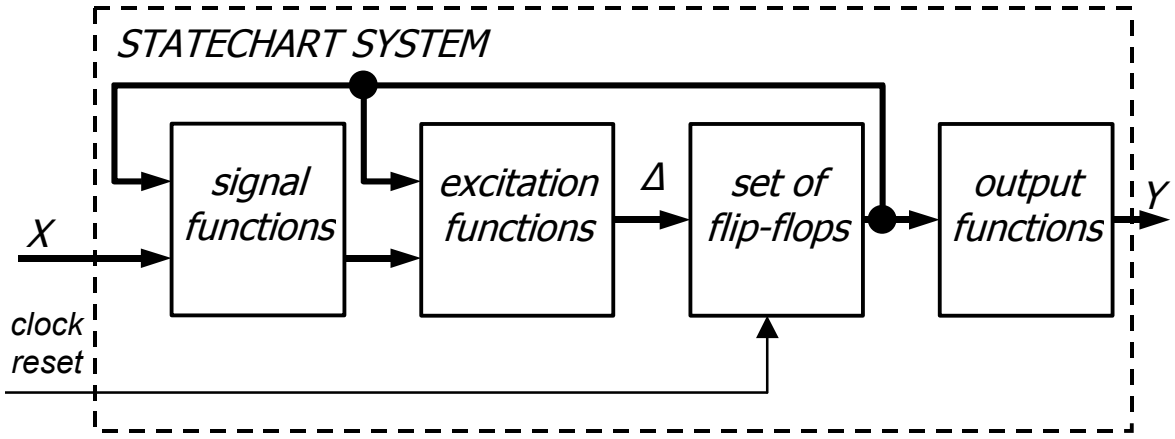


Fig. 2. Statechart system model

By the association of the excitation function with a state, a direct application of FSM and Petri Nets traversal algorithm can be used. The transition function in Fig. 2 $\Delta : \Omega \to \Omega$, is defined as a functional vector of a Boolean function: $\Delta : [\delta_1(S_z, I), \delta_2(S_z, I), \ldots \delta_n(S_z, I)]$, where $\delta_i(S_z, I)$ is an excitation function of the state s_i flip-flop and I is the set of signals in the system represented by their functions. In Fig. 1d symbol s_i denotes both a state in the diagram and a variable of characteristic function. Boolean expressions related to transition functions can be implemented by using topological information from the diagram.

4. SYMBOLIC STATES SPACE EXPLORATION OF STATECHARTS

Symbolic state space exploration techniques are widely used in the area of synthesis, testing, and the verification of finite state systems. Coudert *et al* were the first to realise that Binary Decision Diagram (BDDs) could be used to represent sets of states [4]. This led to the

formulation of an algorithm that traversed the State Transition Graph in breadth-first manner, moving from a set of a set of states to the set of its fan-out states. In this approach a set of states is represented by means of characteristic functions. The key operation required for traversal is the computation of the range of a function, given a subset of its domain [2]. The computational cost of these symbolic techniques depends on the cost of the operation performed on the BDDs and does not depend on the number of states and transitions. For example, from Fig. 1a BDD characteristic function for the set of all global states (Fig. 1b) consists of 21 nodes, and characteristic function for the set of all configurations (Fig. 1c) counts 20 nodes. The symbolic state exploration of statecharts relies on:

- association transition functions to states,
- association logic functions to signals,
- representation of Boolean function as BDDs,
- representation of sets of states using their characteristic functions,
- computation of a set of next states as an image of the state transition function on the current state set for all input signals.

Starting from the default configuration and the set of signals, symbolic state exploration methods enable the computation of the entire set of next states in one formal step. Burch *et al* and Coudert *et al* were the first to independently propose the approach to the image computation [4,5]. Two main methods are transition relation and transition function. The latter is the method implemented by the author. The symbolic state space algorithm of statechart *Z* is as follows:

```
symbolic_traversal_of_Statechart(Z, initial_marking) {
   X[M0⟩ = current_marking = initial_marking;
   while (current_marking != ∅) {
      next_marking = image_computation(Z, current_marking);
      current_marking = next_marking * ¬X[M0⟩;
      X[M0⟩ = current_marking + X[M0⟩;
   }
}
```

Fig. 3. The symbolic traversal of Statecharts

The variables in italics represent characteristic functions of corresponding sets of configurations. All logical variables are represented by BDDs. Several subsequent configurations are simultaneously calculated using the characteristic function of current configurations and transition functions. This computation is realised by the *image_computation* function. The set of subsequent configurations is calculated from the following equations:

$$next\_marking = \exists_s \exists_x \left(current\_marking * \prod_{i=1}^{n} \left[s_i' \odot \left(current\_marking * \delta_i(s, x) \right) \right] \right) \quad (3)$$

$$next\_marking = next\_marking \langle s' \leftarrow s \rangle \quad (4)$$

where s, s', x denote the present state, next state and input signal respectively; $\exists_s$ and $\exists_x$ represent the existential quantification of the present state and signal variables; symbols $\odot$ and * represent logic operators XNOR and AND respectively; equation (4) means swapping variables in expression.

Given the characteristic function of all reachable global states of a system, it is possible to calculate the set of all configurations. As mentioned earlier in section 3, a state is said to be active when every state belonging to the path, carried from it to the root state, is active. This

leads to the formulation of a state activating function. Let α_i be a Boolean function $\alpha_i : S_z \to \{0,1\}$ which evaluates to 1 when state s_i is active. The generation of a set of all configurations relies on the image computation of a characteristic function in transformation by activating functions:

$$X_{[C_0\rangle} = \exists_s \exists_x \left(X_{[M_0\rangle} * \prod_{i=1}^{n} \left[s_i' \odot \left(X_{[M_0\rangle} * \alpha_i(s,x) \right) \right] \right) \tag{5}$$

$$X_{[C_0\rangle} = X_{[C_0\rangle} \langle s' \leftarrow s \rangle \tag{6}$$

The example in Fig. 1a describes the behaviour of a remote controller. The remote can be in 11 global states (Fig. 1b) which correspond to the set of 7 possible configurations (Fig. 1c).

5. CONCLUSION

A visual formalism proposed by David Harel can be effectively used to specify the behaviour of digital controllers. Controllers specified in this way can subsequently be synthesised in FPGA circuits. In this paper it has been shown that state space traversal techniques from FSM and Petri nets theory can be efficiently used in the fields of statechart controllers design. The presented issues are a matter of the author's investigations. Within the framework of the research, a software system called *HiCoS* has been developed, where presented algorithms have been successfully implemented.

REFERENCES

[1] Adamski M., "SFC, Petri Nets and Application Specific Logic Controllers, *Proc. of The IEEE Int. Conf. on Systems, Man and Cybernetics* San Diego, USA Nov. '98, ss 728-733

[2] K. Biliński, *Application of Petri Nets in parallel controllers design*, PhD. Thesis, University of Bristol, Bristol, 1996

[3] J. R. Burch, E. M. Clarke, K. L. McMillan, and D. Dill. "Sequential Circuit Verification Using Symbolic Model Checking", *Proceedings of the 27th Design Automation Conference,* pp. 46-51, June 1990

[4] O. Coudert, C. Berthet, and J. C. Madre, "Verification of Sequential Machines Using Boolean Functional Vectors", *IMEC-IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, pp. 111-128, November 1989

[5] A. Ghosh, S. Devadas, A. R. Newton, *Sequential logic testing and verification*, Kluwer Academic Publisher, Boston 1992

[6] D. Harel, Statecharts, A Visual Formalism for Complex Systems, *Science of Computer Programming*, No 8, North-Holland, 1987, pp. 231-274

[7] G. Łabiak, Implementacja sieci Statechart w reprogramowalnej strukturze FPGA. *Mat. I Krajowej Konf. Nauk. Reprogramowalne Układy Cyfrowe*, Szczecin, pp.169-177 '98

[8] A. Magiollo-Schettini, M. Merro, *Priorities in Statecharts, Diparamiento di Informatica*, Universita di Pisa, Corso Italia

[9] E. Pastor, O. Roig, J. Cortadella, and R. M. Badia, "Petri Net Analysis Using Boolean Manipulation", *Proc. of 15th Int. Conference: Application and Theory of Petri Nets*, Vol. 815 of *Lecture Notes in Computer Science* pp. 416-435, Springer Verlag June 1994

[10] M. Rausch B. H. Krogh, *"Symbolic Verification of Stateflow Logic"*, *Proceedings of the 4th Workshop on Discrete Event System*, Cagliari, Italy, pp. 489-494 1998

[11] UML 1.3 Documentation, Rational Software Corp. '99, *http:// www.rational.com/uml*