Vol. 35 No. 2 Transactions of Information Processing Society of Japan Feb. 1994

Regular paper

Minimum Single Transition-Time Assignments for Asynchronous Sequential Circuits Using BDD

YONG-JIN KWON [†] and Shuzo YAJIMA [†]

We propose a new method of the single transition-time (STT) assignments for asynchronous sequential circuits, in which propositional calculus, or Boolean algebra is adopted. Exactly minimum solutions of the STT assignments are obtained by our method. In order to handle huge propositional formulas, the shared binary decision diagrams (SBDD's) are used as an internal representation of the formulas which denote the STT assignment for a given normal flow table. Moreover, as an application of the method, an algorithm for the constrained encoding problems is also proposed, for which so far only heuristic algorithms are known for solving large and practical problems. However, our algorithm always guarantees minimum solutions. Experimental results show that our methods are effective to obtain minimum solutions at significantly reduced computation cost. The STT assignments are said to be very practical so that the method is available at the design of asynchronous sequential circuits.

1. Introduction

While every state assignment, which has a unique binary vector for every internal state of a synchronous sequential circuit, may yield a correct circuit realization, it is not a sufficient condition for asynchronous sequential circuit realizations. The state assignment for asynchronous sequential circuits must ensure that no critical races exist among the state variables. We are concerned with the state assignment for asynchronous sequential circuits in this paper.

As for state assignments of asynchronous sequential circuits, many methods have been proposed, including a class of the single transition-time (STT) assignments. In order to find solutions having a minimum number of state variables for the STT assignment problems, we have to conquest NP-complete factors included in the problems, so that it is said to be difficult to calculate the solutions for problems having large and practical size.

In this paper, we propose a method that minimizes the number of state variables used for the STT assignments. Propositional calculus is employed for expressing the STT assignments with introduced Boolean variables here, and the shared binary decision diagrams (SBDD's)[1] are used as an internal representation of propositional formulas denoting the state assignments. Thus solutions having a minimum number of state variables are obtained efficiently for large and practical problems, where the solutions are called minimum solutions.

Besides, as an application of the idea that dichotomies of a given flow table are denoted by propositional formulas by means of newly introduced Boolean variables for obtaining the minimum solutions, we also propose a method for constrained encoding problems [2] whose importance have grown with the recent advances in the synthesis of sequential machines, where Ref. 2) proposed a heuristic algorithm with no guarantee of its results being minimum. However our method minimizes the number of state variables necessary to satisfy all constraints of the problems.

We have implemented the methods for the STT assignments and constrained encoding problems, and conducted experiments to evaluate the performance. The results show that our methods are very effective to obtain minimum solutions at the significantly reduced computation time and memory. The STT assignments are said to be very practical so that our newly proposed method is available at the design of

† Department of Information Science, Faculty of Engineering, Kyoto University

asynchronous sequential circuits.

The rest of the paper is organized as follows. Section 2 covers unicode STT assignments. A minimum unicode STT assignment is described in section 3. In section 4, A minimum constrained encoding is proposed, and we present in section 5 results of our experiments. Finally, we have concluding remarks and point to directions for future work.

2. Unicode STT Assignments

We consider a class of single transition-time (STT) assignments.[3] State assignments for which a single transition time is always sufficient for any transition, are called *STT assignments* in which transitions may occur by means of noncritical races among all of the state variables distinguishing the internal and final states. Speed of operation is often an important consideration in the design of switching circuits, and it is not unusual to pay for it with significant increases in the number of components used. A key factor influencing the speed of operation of asynchronous sequential circuits is the maximum number of transition times required for an interstate transition. Thus the STT assignments are said to be very practical at the designing of asynchronous sequential circuits. In this paper, we limit our discussion to the following: Only normal flow tables are treated here, and the assignments are restricted to those with one binary vector composed of state variables per state of machines, which we refer to as *unicode STT assignments*.

The fundamental conception of the STT assignments is that of dichotomies. Letting U and V be disjoint subsets of states of flow tables, we define a *dichotomy* as the unordered pair $(U; V)$. For example, given a pair of transitions $i \rightarrow j$ and $k \rightarrow m$, where i and j are each different from k and m, we say that the dichotomy associated with the pair of transitions is $(U; V)$, where $U=\{i, j\}$ and $V=\{k, m\}$ or vice versa. This dichotomy is generally written as $(ij; km)$(or $(km; ij)$). In the following, we will consider that a dichotomy $(ij; km)$ denotes a dichotomy $(ij; km)$ or $(km; ij)$. In cases in which one of the transitions is degenerate (say $i=j$), we obtain a dichotomy such as $(i; km)$, and if both transitions are degenerate, the dichotomy reduces to $(i; k)$. A state variable y_i in a particular state assignment is said to *cover* a dichotomy $(U; V)$ if $y_i=0$ for every state in U and $y_i=1$ for every state of V (or vice versa). Using the above notion, we can now state a basic theorem for the STT assignments as follows:

Theorem[3]: A state assignment for a normal flow table is a valid unicode STT assignment iff, for every pair of transitions $i \rightarrow j$ and $k \rightarrow m$ that appear in the same column and such that $j \neq m$, the dichotomy $(ij; km)$ is covered by at least one state variable of the state assignment. □

A symple and general procedure finding minimal unicode STT assignments for a normal flow table using the above theorem, is simply summarized as follows.

1. For each column, form the dichotomy $(ij; km)$ for every pair of transitions $i \rightarrow j$ and $k \rightarrow m$ where $j \neq m$.
2. Find maximal compatibles corresponding to the dichotomies.
3. Find a minimal set of the maximal compatibles that covers every dichotomy.

The above procedure contains some factors so that it is said to be difficult to obtain the minimum solutions for large and practical problems. The factors are that the number of maximal compatibles may be exponential in the number of states, and the covering problem is NP-complete. Thus efficient solving methods have been expected and have been proposed so far.

3. Minimum Unicode STT Assignments Using SBDD

We propose a method for the unicode STT assignments that minimizes the number of state variables used. We express the state assignments in propositional formulas which are specified with SBDD's as internal representations. The solutions are obtained by searching the SBDD's in time linear in the number of Boolean variables appeared in the SBDD's. We call this method a minimum unicode STT assignment using SBDD.

3.1 Introduction of Boolean variables

We introduce Boolean variables which are used for denoting the unicode STT assignments problem with propositional calculus.

We will let:

S = a set of n states

B = a set of binary vectors $(b_0 \cdots b_i \cdots b_{k-1})$, $b_i \in \{0, 1\}$

When a codeword $b(\mathrm{b}_0 \cdots \mathrm{b}_i \cdots \mathrm{b}_{k-1}) \in B)$ is assigned to a state $X \in S$, we introduce a Boolean variable $D^X_{b_i} \in \{0, 1\}$ in the following:

$$D^X_{b_i} = \begin{cases} 1 & \text{if the codeword } b_0 \cdots b_1 \cdots b_{k-1} \text{ is assigned to the state } X \in S \text{ and } b_i = 1 \\ 0 & \text{otherwise.} \end{cases}$$

For example, if a codeword $101\,(b_0 b_1 b_2)$ is assigned to a state s_i, then b_0 is 1, b_1 is 0 and b_2 is 1 so that $D^{s_i}_{b_0}$ is 1, $D^{s_i}_{b_0}$ is 0 and $D^{s_i}_{b_2}$ is 1. Using these Boolean variables, we can show that a dichotomy is covered by a state variable. Namely, if we have a propositional formula Q given as follows, then $Q = 1$ denotes that "A dichotomy $(ij\,;\ km)$ is covered by a state variable b_0."

$$Q \equiv D^i_{b_0} D^i_{b_0} \overline{D^k_{b_0}}\ \overline{D^m_{b_0}} + \overline{D^i_{b_0}}\ \overline{D^j_{b_0}}\ D^k_{b_0} D^m_{b_0}.$$

3.2 Propositional representations of the unicode STT assignments

We denote the assignments with propositional formulas based on the Boolean variables introduced above.

In order that the unicode STT assignment of a given flow table is expressed by propositional calculus, we have to calculate necessary dichotomies for the assignment of the given flow table. Given a normal flow table, we can derive a set of dichotomies with ease. We show an example in the following. We assume that a normal flow table is given and three state variables y_0, y_1 and y_2 are used for the assignment of the flow table, where each variable is binary. For building propositional formulas denoting the unicode STT assignment of the flow table, suppose that we obtain the following dichotomies from the flow table.

$(S_0 S_1; S_4 S_5)$, $(S_4 S_5; S_3)$, $(S_1 S_2; S_3 S_4)$, $(S_2 S_4; S_1 S_3)$, $(S_2 S_4; S_3 S_5)$

Then, a proposition that "The dichotomy $(S_0 S_1; S_4 S_5)$ is covered by at least one state variable among y_0, y_1 and y_2" is denoted by $\mathscr{D}_{(01;45)} = 1$, where the propositional formula $\mathscr{D}_{(01;45)}$ is given as follows:

$$\begin{aligned} \mathscr{D}_{(01;45)} \equiv\ & D^{S_0}_{b_0} D^{S_1}_{b_0} \overline{D^{S_4}_{b_0}}\ \overline{D^{S_5}_{b_0}} + \overline{D^{S_0}_{b_0}}\ \overline{D^{S_1}_{b_0}}\ D^{S_4}_{b_0} D^{S_5}_{b_0} \\ & + D^{S_0}_{b_1} D^{S_1}_{b_1} \overline{D^{S_4}_{b_1}}\ \overline{D^{S_5}_{b_1}} + \overline{D^{S_0}_{b_1}}\ \overline{D^{S_1}_{b_1}} \\ & D^{S_4}_{b_1} D^{S_5}_{b_1} + D^{S_0}_{b_2} D^{S_1}_{b_2} \overline{D^{S_4}_{b_2}}\ \overline{D^{S_5}_{b_2}} \\ & + \overline{D^{S_0}_{b_2}}\ \overline{D^{S_1}_{b_2}}\ D^{S_4}_{b_2} D^{S_5}_{b_2} \end{aligned}$$

Similarly, for the dichotomy $(S_4 S_5; S_3)$ we have the following formulas $\mathscr{D}_{(45;\,3)}$:

$$\begin{aligned} \mathscr{D}_{(45;3)} \equiv\ & D^{S_4}_{b_0} D^{S_5}_{b_0} \overline{D^{S_3}_{b_0}} + \overline{D^{S_4}_{b_0}}\ \overline{D^{S_5}_{b_0}}\ D^{S_3}_{b_0} + D^{S_4}_{b_1} \\ & D^{S_5}_{b_1} \overline{D^{S_3}_{b_1}} + \overline{D^{S_4}_{b_1}}\ \overline{D^{S_5}_{b_1}}\ D^{S_3}_{b_1} + D^{S_4}_{b_2} D^{S_5}_{b_2} \\ & \overline{D^{S_3}_{b_2}} + \overline{D^{S_4}_{b_2}}\ \overline{D^{S_5}_{b_2}}\ D^{S_3}_{b_2} \end{aligned}$$

For all the dichotomies $(U;V)$, we make propositional formulas $\mathscr{D}_{(U;V)}$ like above, and let $\mathscr{D}$ be the product of all the formulas $\mathscr{D}_{(U;V)}$:

$$\mathscr{D} \equiv \prod_{\mathrm{all}(U;V)} \mathscr{D}_{(U;V)}$$

Then, the unicode STT state assignment is denoted by $\mathscr{D}$. A minimum solution having the minimum number of state variables is obtained when the $\mathscr{D}$ that is not contradiction is acquired for the first time, while the number of state variables is increased one by one each step, and the minimum solution is associated with a product term which is involved in the $\mathscr{D}$ when the $\mathscr{D} = 1$.

Good representation of propositional formulas is a key to efficient implementation. In our implementation we use shared binary decision diagrams (SBDD's)[1] as an internal representation of the $\mathscr{D}$ in order to handle huge propositional formulas. The SBDD's, improved binary decision diagrams, are graph representations of Boolean functions, and have the following desirable properties;

- Many functions are represented compactly and simultaneously by sharing isomorphic sub-graphs.
- Logic operations between functions can be carried out much efficiently.

These two advantages are very much suited to our purpose. Since in our method, we need to represent many propositional formulas to express a set of dichotomies, the former property is very favorable. The latter one is also favorable for making an SBDD denoting the STT assignments. The propositional formulas appearing in our method are produced by logical operations. We can expect efficient implementation because SBDD's are known to have good affinity for the operations.

When the $\mathscr{D}$ is represented by an SBDD, the minimum solution of the unicode STT assignment for the flow table is associated with a path which is involved in the SBDD when the path goes to node "1" (the "1" leaf). Thus there are many minimum solutions in the SBDD. Namely, by searching out all the paths going to the "1" leaf, we can have all the minimum ones.

The minimum solution, or path going to the leaf is searched out in time linear in the number of Boolean variables appeared in the SBDD. The number of Boolean variables B is $B=n\times k$ if using the Boolean variables introduced in this paper, where n is the number of states and k the number of state variables allowed to the assignments. Using coded Boolean variables, it can be expected that the figure is decreased. See Ref. 4) for details. There are methods7)–10) that represent given problems with cost as a SAT and uses BDD to solve it. A method10) aims at minimizing logic circuits representing sequential machines. However we are interested in the state assignment and minimizing state variables being used for it. Moreover the method denotes constraints necessary for solving given problems with formulas as it is so that long formulas may be derived. However in our method, the constraints are transformed into dichotomies and then we make formulas for the dichotomies so that we obtain more simple and regular ones which may derives SBDD's having smaller sizes.

The algorithm is summarized as follows.

Algorithm[STTBDD]

```
(Inputs)
A directed graph $G$ consists of $n$ vertices.
$k$ state variables allowed to the unicode STT assignment.
(Output)
If a solution exists, a minimum solution.
(Algorithm)
step 1  $\mathcal{D}:=TRUE$;
step 2  while (all dichotomies)
        begin
          Make a propositional formula
          $D_{(U;V)}$;
          $\mathcal{D}:=\mathcal{D}\cdot D_{(U;V)}$;
          If $\mathcal{D}$ is a contradiction, then FAIL;
        end
step 3  Search the minimum solution and output it;
```

4. Complexity of the STT Assignment

We hope to discuss about a theoretical bound of computation time and memory space of our approach in the worst case. Let be the number of states n. As for the universal STT assignment, a method is kwown to assign n-state machines with n state variables. If we allow to use n state variables for assigning n-state machines in our method, the number of Boolean variables appeared in the SBDD's is $O(n^2)$ so that the size of the SBDD is $O(2^{n^2})$ in the worst case. The number of the dichotomies is $O(n^2)$ and the size of a formula denoting the covering condition of one dichotomy by state variables is $O(n)$. Thus the complexity of computation time is $O(2^{n^2})$ and that of memory space is also $O(2^{n^2})$.

If we make an STT state assignment for sequential machines by the method described simply in section 2, the computation time is $O(2^{n^2})$. This complexity is of course same with our method. However as proved by using the SBDD's as internal representations in many researchs of the CAD field, many functions which appear in practical extent of logic circuits, are represented compactly, i. e. the polynomial size of the number of Boolean variables, and logic operation between function can be carried out much efficiently. Moreover the procedure of searching out the path associated with the minimum solution is performed very fast, i. e. the time linear with the number of Boolean variables, so that it is expected practically that the solutions are obtained by much shorter time than that of the theoretical complexity.

As using the manipulator,1) the maximum number of available Boolean variables is 1024. If we use the Boolean variables introduced in our method, it may be possible to assign up to 32-state machines in the best case.

5. An Application: Constrained Encoding Problems

We propose a method for constrained encoding problems as an application of the minimum unicode STT assignment using SBDD described in the previous chapter, in which constraints denoted by dichotomies, are expressed by propositional formulas with introduced Boolean variables.

Given a set $S=\{s_0, s_1, \cdots, s_{n-1}\}$ of n states, *dichotomy-based constrained encoding*2) aims at finding an assignment m of S into a set $\{m(s_0), m(s_1), \cdots, m(s_{n-1})\}$ of n binary k-tuples (bit vectors), in such a way that k is minimized and a set of dichotomy constraints are satisfied. A dichotomy constraint DC on S is a pair $DC=(S^+;S^-)$ of disjoint subsets of S; to satisfy such a constraint the encoding must have at least one state variable that has the value 1 for all the states in S^+ and 0 for all the states in S^-, or vice versa.

The importance of finding minimum constrained encodings has grown with the recent advances in the synthesis of synchronous sequential machines. It is now accepted as a general rule that, in order to find an economic logic realization, it is desirable to assign certain groups of states to groups of neighboring vertices in the hyper cube. In particular, if a programmable logic array (PLA) is used to implement the combinational part of the circuit, such groups of states can be identified by a technique called symbolic minimization. Each group is to be embedded into a face in the hyper cube; such constraints are therefore called *face-embedding* constraints. They can also be reformulated as dichotomy constraints.[2)]

For the constrained encoding problems, Ref. 2) proposed a heuristic algorithm. However, the algorithm basically consists of the greedy step so that, in general, the minimum solutions are not guaranteed. We propose here a method of the constrained encoding problems, by which for practical and large problems, the minimum solutions are always calculated. We define the minimum constrained encoding problem as follows: We assume that a set S of n states, and a set DC of d nontrivial constraints on S are given. The minimum constrained encoding problem is to find an encoding m of S so that all constraints of DC are covered by k state variables, where k is a minimum integer.

For obtaining the minimum constrained encodings, it is sufficient to build propositional formulas for all given constraints that are given by means of the type of dichotomies. Then by representing the propositional formulas with SBDD's, we may obtain the minimum solutions at significantly reduced computation time and memory cost. Other steps of the algorithm for the constrained encoding problems are similar with them of that for the unicode STT assignments.

6. Experimental Results

We implemented a unicode STT assignment on UNIX operating system in C language on the basis of the method described in the preceding sections. This implementation was linked with an SBDD manipulator.[1)] We conducted experiments to evaluate its performance on SUN SPARC station 2 workstation (64MByte). We assigned several normal flow tables by giving them a number "k" which is the number of state variables allowed to the state assignment. The results are shown in **Table 1**. The first column shows the names of the flow tables; for example, S04D05 specifies a normal flow table which has 4 states and 5 dichotomies. The column # m sv shows the minimum number of state variables. The column # M nodes contains the number of nodes of SBDD's used for representing the flow tables (maximum values). The last column shows CPU times for initialization of SBDD's and the state assignment per flow table in seconds. The number of nodes of SBDD's was limited to 1,000,000 (2^{20}), and the implementation (including the SBDD manipulator) used approximately 20 MByte storage at most. It is clear from Table 1 that our method is very efficient in a point of view of computational time. In order to prove the usefulness of our method, it must be necessary of comparison with other methods. However to the best of our knowledge, it is difficult to find experimental results of minimum STT state assignments.

Table 1 Experimental results.

name	# m sv	# M nodes	CPU(sec)*
S04D05	3	43	2.5
S05D05	3	65	2.6
S05D12	4	522	2.7
S06D16	5	2922	3.2
S07D20	4	3678	3.7

* Time necessary to initialize 1,000,000 nodes (about 1.7 sec) is included.

7. Conclusion and Future Work

We have proposed a new method of the unicode STT assignments which leads to exactly minimum solutions. In the proposed method, dichotomies of normal flow tables are denoted by propositional formulas which are written with Boolean variables introduced here. By the use of SBDD's as an internal representation of the propositional formulas, we can reduce the storage requirement required for representing the formulas. With experiments, the new method may proves efficient. The STT assignments are said to be very practical so that our newly proposed method is useful at the design of asynchronous sequential circuits. In parallel, we showed an application of the proposed method to the minimum constrained encoding problems.

Our future work includes:

1. More sophisticated algorithm which reduces the size of nodes of SBDD's representing the minimum unicode STT assignment.
2. New method which introduces Boolean variables used for propositional formulas should be found out in order to handle more lager sequential circuits.

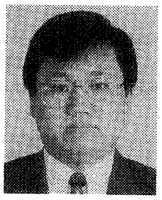
Besides, when we have an SBDD which is not contradiction and denotes the minimum unicode STT assignment for a flow table, there are a lot of the minimum solutions in the SBDD. Though our method now output any one among them as the minimum solution, it is worth considering for example that the minimum solution which is associated with what we call the simplest state transition functions, is searched out. This is another our future work.

Acknowledgments We would like to express our appreciation to Mr. Minato (now with NTT) for using of the SBDD manipulator. We would also like to thank the members of Yajima Laboratory at Kyoto University for their valuable discussions and comments.

References

1) Minato, S., Ishiura, N. and Yajima, S.: Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation, *Proc. 27th Design Automat. Conf.*, pp. 52-57 (1990).
2) Shi, C. J. and Brzozowski, J. A.: Efficient Constrained Encoding for VLSI Sequential Logic Synthesis, *European Design Automation Conference*, Germany, pp. 266-271 (1992).
3) Unger, S. H.: *Asynchronous Sequential Switching Circuits*, John Wiley & Sons, Inc. (1969).
4) Kwon, Y. J. and Yajima, S: Minimal One-Shot State Assignment Using Binary Desicion Diagrams, *IEICE Technical Report*, COMP92-27 (1992).
5) Bryant, R. E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677-691 (1986).
6) Kwon, Y.-J.: One-Shot State Assignment for Asynchronous Sequential Machines, Master Thesis, Kyoto University (1990).
7) Kitajima, M., Takagi, N. and Yajima, S.: Symbolic Simulator Using a Graphical Representation of Boolean Functions, *36th National Convention Record of IPSJ*, pp. 51-52, March (1988).
8) Ishiura, N., Deguchi, Y. and Yajima, S.: Coded Time-Symbolic Simulation Using Shared Binary Decision Diagram, *Proc. ACM/IEEE 27th Design Automation Conference*, pp. 130-135 (Jun. 1990).
9) Takahashi, N., Ishiura, N. and Yajima, S.: Fault Simulation for Multiple Using Shared Binary Decision Diagrams, *Proc. Synthesis and Simulation Meeting and International Interchange*, pp. 157-164 (Oct. 1990).
10) Lin, B. and Somenzi, F.: Minimization of Symbolic Relations, *Proc. IEEE Int. Conf. on Computer-Aided Design*, pp. 88-91 (1990).

(Recievd February 15, 1993)
(Accepted November 11, 1993)



Kwon, Yong-Jin was born in Seoul, Korea, in 1964. He received the B. E. in electronic engineering from Hankuk Aviation University, Seoul, Korea, and M. E. in information science from Kyoto University, Kyoto, Japan, in 1986 and 1990, respectively. He is working for the degree of Ph. D. in information science from Kyoto University. His current interests include logic synthesis of sequential circuits.

Shuzo Yajima was born in Takarazuka, Japan, on December 6, 1933. He received the B. E., M. E., and Ph. D. degrees in electrical engineering from Kyoto University, Kyoto, Japan, in 1956, 1958, and 1964, respectively. He developed Kyoto University's first digital computer, KDC-I, in 1960. In 1961 he joined the faculty of Kyoto University. Since 1971 he has been a Professor in the Department of Information Science, Faculty of Engneering, Kyoto University, engaged in research and education in logic circuits, switching, and automata theory. Dr. Yajima was a Trustee of the Institute of Electronics and Communication Engineers of Japan and Chairman of the Technical Committee on Automata and Languages of the Institute. He served on the Board of Directors of the Information Processing Society of Japan.