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

STATE ASSIGNMENT OF ASYNCHRONOUS PARALLEL AUTOMATA

Ljudmila CHEREMISINOVA

Institute of Engineering Cybernetics of National Academy of Sciences of Belarus,
Surganov str., 6, 220012, Minsk, BELARUS, *cld@newman.bas-net.by*

Abstract. *A problem of race-free state assignment of asynchronous parallel automata is considered. The goal is to encode partial states of parallel automaton using minimal number of coding variables and excluding critical races during automaton operation. Requirements imposing on the partial states codes to eliminate the influence of races are formulated. An exact algorithm to find a minimal solution of the problem of race-free state assignment for parallel automata is suggested. The algorithm provides reducing the computational effort when searching state encoding.*

Key Words. *Parallel Automaton, Partial State Assignment, Races in Asynchronous Automata,*

1. INTRODUCTION

The success of the control of a multiple component system depends greatly on the efficiency of the synchronization among its processing elements. The functions of a control of such a system are concentrated in one block - logic control device that should provide a proper synchronization of interaction between the components. In order to represent clearly the interaction involved in concurrent engineering system it is necessary to describe formally its functional and structural properties.

As a functional model of a discrete control device to be designed a model of parallel automaton is proposed [1, 8, 9]. This model can be considered as an extension of a sequential automaton (finite state machine) to represent parallel processes. The parallel automaton is more complicated and less studied model in contrast with classical sequential automaton model. An essential difference from sequential automaton is that a parallel automaton can be in more than one state simultaneously. That is why the states of a parallel automaton were called as partial ones [8]. All partial states a parallel automaton is in at some moment form its global state. In that case any two of these partial states (forming a global state) are called parallel [8]. Any transition of automaton defines the partial state changes that cause the global state changes. The most of transitions (and all for asynchronous parallel automaton) are forced by changes of external signals.

The design of asynchronous automata has been an active area of research for the last 40 years. There has been a renewed interest in asynchronous design because of their potential for highperformance and avoidance of clock. However, design of asynchronous automata remains a cumbersome problem because of difficulties to ensure correct dynamic behavior.

The important step on the way to control device hardware implementation is the state assignment. It is at the heart of the automaton synthesis problem (especially for its asynchronous mode of realization). Despite large effort devoted to this problem no satisfactory solutions have been proposed. A difference of this process for parallel automaton in comparison with the sequential one is that there are parallel states in the first one (they are compatible in the sense that the automaton can find itself in them at the same time). That is why it was suggested in [8] to code partial states with ternary vectors which should be non-orthogonal for parallel partial states but orthogonal for non-parallel ones. After having coded partial states it is possible provide them with their codes. In such a way an initial parallel automaton is transformed from its abstract form into a structural one – a sequent parallel automaton or a system of Boolean functions that can be directly hardware implemented.

The problem of state assignment becomes harder when asynchronous implementation of a parallel automaton is considered. The mentioned condition imposed on codes is necessary but is not enough for that case. The additional condition to be fulfilled is to avoid the influence of races between memory elements (flip-flops) during hardware operation. One of the ways to avoid that is to order switches of memory elements so as to eliminate critical races.

A problem of race-free state assignment of asynchronous parallel automata is considered. The goal is to encode partial states of parallel automaton using minimal number of coding variables and to avoid the critical races during automaton operation. An exact algorithm to find a minimal solution of the problem is suggested. The algorithm allows reducing the computational effort when searching state encoding. The same problem is considered in [5] where another approach was suggested. The method is based on covering a family of complete bipartite subgraphs defining constraints of absence of critical races by minimal number of maximal complete bipartite subgraphs of the state non-parallelism graph.

2. CONSTRAINTS OF ABSENCE OF CRITICAL RACES

The asynchronous sequential automaton behaves as follows. Initially, the automaton is stable in some state. After the input state changes the outputs change their values as specified in automaton description. An internal state change may be concurrently with the output change. After automaton achieving a new stable state it is ready to receive a new input. Throughout this cycle output and inner variables should be free of glitches. In summary asynchronous designs differ from those synchronous since state changes may pass through intermediate states.

The sequence of these intermediate states must be preserved in the case of multi-output change (when intermediate states involve the output change). It can be done with the proper state assignment. The 1-hot encoding [4] can ensure such a behavior, but it demands too many coding variables. That is why the methods of race-free state assignment are of interest.

In [6] the constraints to ensure hardware implementation of sequential automaton to be race-free are given. These constraints allow avoiding interference between automaton transitions that take place for the same input state. The codes satisfying these constraints ensure race-free implementation of the automaton. The encoding constraints can be represented in the form of dichotomies. A dichotomy is a bipartition $\{S_1; S_2\}$ of a subset $S_1 \cup S_2 \subseteq S$ ($S_1 \cap S_2 = \varnothing$). In considered state encoding a binary variable y_i covers dichotomy $\{S_1; S_2\}$ if $y_i = 0$ for every state in S_1 and $y_i = 1$ for every state in S_2 (or vice versa). A pair of transitions taking place at the same input is called below as competitive transitions. In [6] the following constraints of critical race-free encoding are given that are induced by competitive transitions of different types:

1) $s_i \to s_j$, $s_k \to s_l$ (i, j, k, l are pair-wise different) give rise to $\{s_i, s_j; s_k, s_l\}$;
2) $s_i \to s_j$, $s_j \to s_l$ (i, j, l are pair-wise different) give rise to $\{s_i, s_j; s_l\}$ and $\{s_i; s_j, s_l\}$;
3) $s_i \to s_j$, $s_k \to s_j$ (i, j, k are pair-wise different) give rise to $\{s_i, s_j; s_k\}$ if the output on the transition from s_k is different than that on the transitions from s_i and s_j (at the input considered).

A parallel automaton is described by a set of generalized transitions $(X_{kl}, S_k) \to (S_l, Y_{kl})$ between the subsets of partial states. Such a transition should be understood as follows: if the global state of the parallel automaton contains all the partial states from S_k and the variables in the conjunction term X_{kl} assume values at which $X_{kl} = 1$, then as the result of the transition the automaton goes to a new global state that differs from initial one by that it contains partial states from S_l instead of those from S_k. More than one generalized transition may take place in some moment when parallel automaton functions. These transitions define changing different subsets of parallel partial states. There are no races on such a pair of transitions.

In the case of parallel automaton we have generalized transitions instead of elementary ones. A generalized transition t_{kl}: $S_k \to S_l$ consist of $|S_k|\cdot|S_l|$ elementary transitions $s_{ki} \to s_{lj}$, where $s_{ki} \in S_k$ is nonparallel to $s_{lj} \in S_l$. Let us introduce the set $T(t_{kl}, t_{pq})$ of pairs of elementary transitions $s_{ki} \to s_{lj}$ and $s_{pi} \to s_{qj}$ between pair-wise nonparallel partial states taken from S_k, S_l, S_p and S_q generated by the pair of competitive transitions t_{kl}: $S_k \to S_l$ and t_{pq}: $S_p \to S_q$. For compatible pair t_{kl}, t_{pq} of generalized transitions we have $T(t_{kl}, t_{pq}) = \varnothing$.

In [2] it is shown that in order to avoid the influence of races on competitive generalized transitions t_{kl} and t_{pq} it is sufficient to avoid it on one pair of elementary transitions from the set $T(t_{kl}, t_{pq})$. Thus this statement gives the way of a parallel automaton partial states encoding. Besides this statement ensures any dichotomy constraint consists of pair-wise nonparallel partial states that implies the absence of a constraint forcing a coding variable to have orthogonal values in codes of parallel partial states.

Let distinguish elementary u_{ij}, simple u^p_{ni} and generalized U_n constraints. The first one is a single dichotomy constraint. The second one is associated with a pair of elementary transitions and can consist of one (cases 1, 3 of constraints) or two (case 2) elementary constraints. To avoid critical races on a pair of elementary competitive transitions one has to satisfy an appropriate simple constraint (one or two elementary ones). A generalized constraint U_n induced by a pair P_n of competitive generalized transitions consists of the simple constraints induced by pairs of elementary transitions from its generated set $T(P_n)$. To avoid critical races on P_n it is sufficient to satisfy one of the simple constraints from U_n.

Example 1. Let us consider the following parallel automaton in the form X_{kl} $S_k \to S_l$ Y_{kl}:

1.	$'x_1$	$s_1 \to s_2 \cdot s_3$	$y_1 y_2$	5.	x_3	$s_3 \to s_6$	y_4
2.	$'x_2 x_3$	$s_2 \to s_9$	$'y_2 y_3$	6.	$x_1 'x_2$	$s_4 \to s_7$	$y_1 'y_2$
3.	$'x_3$	$s_9 \to s_2$	$y_2 'y_3$	7.	$'x_2 x_3$	$s_5 \to s_8$	$'y_3$
4.	x_2	$s_2 \to s_4 \cdot s_5$	$'y_1 y_3$	8.	$'x_3$	$s_6 \cdot s_7 \cdot s_8 \to s_1$	$'y_1 'y_4$

The partial states from $\{s_2, s_4, s_5, s_7, s_8, s_9\}$ and $\{s_3, s_6\}$ are pair-wise parallel as well as partial states from $\{s_4, s_7\}$ and $\{s_5, s_8\}$. One can see, for example, that the pair t_1, t_8 of generalized transitions is competitive. The generalized constraint U_{18} induced by that pair consists of 3 simple constraints: $u^p_1 = (\{s_1, s_2; s_7\}$ and $\{s_1, s_7; s_2\})$, $u^p_2 = (\{s_1, s_2; s_8\}$ and $\{s_1, s_8; s_2\})$ and . $u^p_3 = (\{s_1, s_3; s_6\}$ and $\{s_1; s_3, s_6\})$.

By analogy with the case of sequential automaton [7] the algorithm of critical race-free partial states assignment of parallel automaton has two steps: 1) generate and compress a set of encoding constraints; 2) solve these constraints to produce a partial state assignment.

3. GENERATING AND COMPRESSING A SET OF ENCODING CONSTRAINTS

Now an encoding problem formulation is presented that is based on a matrix notation similar to that used in [7] for sequential automata. A dichotomy constraint $\{s_i, s_j; s_k, s_l\}$ can be presented as a ternary (3-valued) vector called a constraint vector. Its length equals to the number of partial states, i-th and j-th entries are 1, k-th and l-th entries are 0 (or vice versa), and the other ones are "-" (don't care). For example the dichotomy $\{s_1,s_7; s_2,s_9\}$ corresponds to the vector "1 0 - - - - 1 - 0".

The constraint matrix $\boldsymbol{U}$ is a ternary matrix with as many rows as critical race-free constraints exist (for a given automaton) and columns as partial states. The matrix $\boldsymbol{U}$ has a complex structure – it consist of submatrices $\boldsymbol{U}_i$ defining generalized constraints the last ones are in turn 1 or 2 line sectioned (separating simple constraints).

Now we give some definitions having in view ternary vectors of the same length. A ternary vector $\boldsymbol{a}$ covers a ternary vector $\boldsymbol{b}$ if, whenever the i-th entry of $\boldsymbol{b}$ is $\sigma \in \{1,0\}$ i-the entry of $\boldsymbol{a}$ is σ too. $\boldsymbol{b}$ is an inversion of $\boldsymbol{a}$ ($\boldsymbol{b}$ = $'\boldsymbol{a}$) if, whenever the i-th entry of $\boldsymbol{a}$ is 1, 0, "-" the i-the entry of $\boldsymbol{b}$ is 0, 1, "-" respectively. Vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are orthogonal if for at least an index i the i-th entries of $\boldsymbol{a}$ and $\boldsymbol{b}$ are orthogonal (1 and 0 or vice versa). An elementary constraint u_i implicates an elementary constraint u_j if u_j as a ternary vector covers u_i or its inversion.

A simple constraint u^p_n implicates:
– an elementary constraint u_j if u_j is implicated by one of the elementary constraints from u^p_n,
– a simple constraint u^p_m if every $u_{mj} \in u^p_m$ is implicated at least by one of $u_{nj} \in u^p_n$,
A generalized constraint U_k implicates:
– a simple constraint u^p_j (elementary constraint u_j) if every $u^p_{kj} \in U_k$ implicates it,
– a generalized constraint U_n if every $u^p_{kj} \in U_k$ implicates at least one of $u^p_{ni} \in U_n$.

For computational efficiency of procedure of searching an optimal encoding it is important to reduce the number of rows of constraint matrix $\boldsymbol{U}$ to the minimal number that represent an equivalent set of constraints on the encoding. It is trivial that duplicate generalized constraints can be deleted. Then the number of rows of $\boldsymbol{U}$ can be compressed further by discarding generalized constraints that are implicated by any other generalized constraint.

Example 2. For considered automaton we can see that generalized constraint ($\{s_1, s_7; s_2, s_9\}$ or $\{s_1, s_8; s_2, s_9\}$) induced by the pair t_2, t_8 of competitive transitions implicates the elementary constraint $\{s_1; s_2, s_9\}$ from the simple constraint ($\{s_1, s_2; s_9\}$ and $\{s_1; s_2, s_9\}$) induced by the pair t_1, t_2 of competitive transitions. Thus we have the following irredundant set of generalized constraints U_k (in the form of dichotomies) for this automaton:

1. $\{s_1,s_2; s_9\}$,
2. ($\{s_1,s_2; s_4\}$ and $\{s_1; s_2, s_4\}$) or $\{s_1; s_2, s_5\}$,
3. $\{s_1,s_2; s_5,s_8\}$,
4. ($\{s_1,s_2; s_7\}$ and $\{s_1, s_7; s_2\}$) or $\{s_1, s_8; s_2\}$,
5. $\{s_2,s_9; s_4,s_7\}$,
6. $\{s_2,s_4; s_9\}$ or ($\{s_2, s_5; s_9\}$ and $\{s_2, s_9; s_5\}$),
7. $\{s_1,s_7; s_2,s_9\}$ or $\{s_1,s_8; s_2,s_9\}$,
8. $\{s_1,s_7; s_2,s_4\}$ or $\{s_1,s_8; s_2,s_5\}$,
9. $\{s_1; s_4,s_7\}$,
10. ($\{s_1,s_3; s_6\}$ and $\{s_1; s_3,s_6\}$),
11. $\{s_4; s_7\}$,
12. $\{s_5; s_8\}$.

The last two constraints U_{11} and U_{12} are introduced since nonparallel partial states should be encoded with orthogonal codes (but constraint U_8 does not implicates neither U_{11} nor U_{12}).

4. FINDING ENCODING OF PARTIAL STATES

One can see that the matrix $\boldsymbol{U}$ is an encoding matrix $\boldsymbol{V}$, but the number of coding variables (equaled to the number of rows) is too big. The encoding matrix $\boldsymbol{V}$ is grown from an initial seed constraint matrix $\boldsymbol{U}$ by its compressing at the expense of combining some constraints and substituting them for one constraint implicating them.

Now we give some definitions and derive some useful properties from them. A constraint u is called an implicant of a set of rows of constrained matrix $\boldsymbol{U}$ if it implicates each of them taken separately. A set $U_j \in U$ is considered as compatible if there exists its implicant having no orthogonal entries associated with parallel states. For example the single $u_{11} \in U_1$ is compatible with $u_{31} \in U_3$ (its implicants are $\{s_1,s_2;\ s_5,s_8,s_9\}$, $\{s_1,s_2;\ s_4,s_5,s_8,s_9\}$, $\{s_1,s_2;\ s_4,s_5,s_7,s_8,s_9\}$), but not compatible with $u_{11} \in u^p{}_1 \in U_{10}$.

We can simply find whether two rows $\boldsymbol{v}_k \in \boldsymbol{V}$ and $\boldsymbol{u}_l \in \boldsymbol{U}$ (or both from $\boldsymbol{U}$) are compatible when using the notion of a boundary vector suggested in [3]. The boundary vector for any row $\boldsymbol{u}_k \in \boldsymbol{U}$ (or $\boldsymbol{v}_k \in \boldsymbol{V}$) is - 4-valued vector that gives an upper bound of its grows (extension) i.e. it determines the potential of verifying the components of $\boldsymbol{u}_k$. In [3] the operations over 3- and 4-valued vectors are given that help simply to find implicants.

When concatenating two rows $\boldsymbol{v}_k$ and $\boldsymbol{u}_l$ (constructing their implicant) we do minimal extension of $\boldsymbol{v}_k$ to implicate $\boldsymbol{u}_l$. In this way any i-th entry of the result of concatenation is equal to that of $\boldsymbol{v}_k$ and $\boldsymbol{u}_l$ (or $'\boldsymbol{u}_l$). $\boldsymbol{v}_k \in \boldsymbol{V}$ is an implicant for generalized constraint $\boldsymbol{U}_l \in \boldsymbol{U}$ if it is implicant for some $\boldsymbol{u}^p{}_{li} \in \boldsymbol{U}_l$. An implicant of a subset U' of generalized constraints is maximal if it is incompatible with all those others (it cannot implicate any more generalized constraints besides those from U'). For example the implicant $\{s_1,s_2;\ s_4,s_5,s_7,s_8,s_9\}$ is maximal, but $\{s_1,s_2;\ s_5,s_8,s_9\}$ is not.

An exact algorithm to find a minimum solution of the problem of race-free state assignment is based on building a set C of all maximal implicants for constraint matrix $\boldsymbol{U}$ and then searching a subset of $V \subseteq C$ of minimal cardinality such that for any generalized constraint $U_i \in U$ there exists an implicant in V implicating it. The second part of the problem is reduced to covering problem of Boolean matrix [7], as in the case with Quine's table.

4.1. Search of maximal implicants

We use branch and bound algorithm to build all maximal implicants. Constraints are processed one by one in predefined order choosing (at each step) one compatible with the current state of the implicant formed. If we exhaust such constraints we would start backtracking to a previous step to modify the solution and repeat searching.

The computational efforts can be reduced using a previously generated compatibility relation on the rows from $\boldsymbol{U}$. Taking into account that any maximal implicant may satisfy only one of the simple constraints from each generalized constraint they all can be regarded as pair-wise incompatible. At each step of the algorithm it is enough to consider as candidates for concatenating only those rows compatible with all concatenated in the current implicant. Further search reduction can be received by sorting the constraints according to the degree of their incompatibility: the greater it is the less is branching.

For the automaton considered there exist 17 maximal implicants.

4.2. Covering problem statement

Once a set C of maximal implicants is found the task is to extract from it a subset that satisfy all generalized constraints $U_k \subset U$. Every $U_k = \{u_{k1}{}^p, u_{k2}{}^p, \ldots, u_{kn}{}^p\}$ is satisfied as OR (though one of $u_{ki}{}^p$ should be satisfied) and $u_{ki}{}^p$ consist of one or two u_{ij} that are satisfied as AND (both u_{i1} and u_{i2} should be satisfied). These statements can be expressed logically (as it is suggested in [5]) by the formulas: $U_k = u_{k1}{}^p \vee u_{k2}{}^p \vee \ldots \vee u_{kn}{}^p$ and $u_{ki}{}^p = u_{i1} \cdot u_{i2}$. Substituting expressions $u_{ki}{}^p$ into U_k and using the distributive low one can receive conjunctive normal form $U_k = U_k{}^1 \cdot U_k{}^2 \cdot \ldots \cdot U_k{}^m$. Any $U_k{}^i$ is a union of separate elementary constraints. For example generalized constraint U_2 (from example 2) is represented as $(\{s_1,s_2;\ s_4\}$ or $\{s_1;\ s_2, s_5\}) \cdot (\{s_1;\ s_2, s_4\}$ or $\{s_1;\ s_2, s_5\})$.

Now the problem is stated in the form of Quine’s table Q. Its rows correspond to implicants $C_i \in C$ and columns to conjunctive members U_k^i for all U_k. An entry (ij) of Q is 1 (marked) if C_i implicates j-th conjunctive member. The task is to find the minimal number of rows covering all columns (every column should have 1 at least in one position corresponding to rows chosen) [7]. The cover presenting encoding for automaton considered (examples 1,2) contains 5 rows. So we find 5 components codes of partial states that provide the absence of critical races when the automaton operates:

$$\boldsymbol{V} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & - & \mathbf{0} & \mathbf{0} & - & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & - & - & \mathbf{0} & - & - & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & - & - \\ \mathbf{1} & - & \mathbf{1} & - & - & \mathbf{0} & - & - & - \\ \mathbf{1} & \mathbf{1} & - & \mathbf{1} & - & - & \mathbf{0} & - & \mathbf{0} \end{bmatrix}$$

It should be noted that some entries of matrix equal to 0 or 1 can be substituted with value don’t care because of usage of maximal implicants.

5. CONCLUSION

Unfortunately the problems considered are computationally hard ones. The growth of the computation time as the size of the problem increases is a practical limitation of the method suggested to computer-aided design systems. It can be used for solving encoding problems of moderate size obtaining after decomposing the whole big problem. Besides the method can be useful for estimation of efficiency of heuristic encoding techniques [3].

REFERENCES

[1] M.Adamski, M.Wegrzyn, “Field programmable implementation of current state machine”, *Proc.of the Third Int. Conf.on Computer-Aided Design of Discrete Devises* (*CAD DD’99*), Minsk, Institute of Engineering Cybernetics of the of Belarus Academy of Sciences, Vol. 1, pp. 4-12, 1999

[2] L.D.Cheremisinova, “Implementation of parallel digital control algorithms by asynchronous automata”, *Automatic Control and Computer Sciences*, Vol. 19, No. 2, pp. 78 – 83, 1985

[3] L.D.Cheremisinova, “Race-free state assignment of a parallel asynchronous automaton”, *Upravlyajushchie sistemy i mashiny*, No 2, pp. 51-54, 1987 (in Russian)

[4] L.D.Cheremisinova, “PLC Implementation of concurrent control algorithms”, *Proc.of the Int. Workshop “Discrete Optimization Methods in Scheduling and Computer-Aided Design”*, Minsk, Republic of Belarus, Sept. 5-6, pp. 190-196, 2000

[5] Yu.V.Pottosin, “State assignment of asynchronous parallel automata with codes of minimum length”, *Proc.of the Int. Workshop “Discrete Optimization Methods in Scheduling and Computer-Aided Design”*, Minsk, Republic of Belarus, Sept. 5-6, pp. 202-206, 2000

[6] S.H.Unger, *Asynchronous sequential switching circuits*, Wiley-Interscience, New York, 1969

[7] A.D.Zakrevskij, *Logical synthesis of cascade networks*, Nauka, Moscow, 1981 (in Russian)

[8] A.D.Zakrevskij, “Parallel automaton”, *Doklady AN BSSR*, Vol. 28, No. 8, pp. 717 – 719, 1984 (in Russian)

[9] A.D.Zakrevskij, *Parallel algorithms for logical control*, Minsk, Institute of Engineering Cybernetics of NAS of Belarus, 1999. (in Russian)