01 Jan 1971

A State Assignment Procedure For Asynchronous Sequential Circuits

Gary K. Maki

James H. Tracey
Missouri University of Science and Technology

Recommended Citation

G. K. Maki and J. H. Tracey, "A State Assignment Procedure For Asynchronous Sequential Circuits," *IEEE Transactions on Computers*, vol. C thru 20, no. 6, pp. 666 - 668, Institute of Electrical and Electronics Engineers, Jan 1971.
The definitive version is available at https://doi.org/10.1109/T-C.1971.223320

$$\begin{bmatrix} 0 & 1 & 0 \cdots & 0 \\ 0 & 0 & 1 \cdots & 0 \\ \vdots & & \ddots & \\ \vdots & & & 1 \\ 1 & 0 & & 0 \end{bmatrix}$$

representing the cyclic permutation

$$(0\ 1\ 2 \cdots (m-1)) \text{ on } \{0, 1, \cdots, m-1\}.$$

Recall the definition of the Kronecker product of two matrices [7]. If A is an $m \times m$ matrix, the *Kronecker* or *tensor* product $A \otimes B$ is the $mn \times mn$ matrix

$$\begin{bmatrix} a_{11}B \cdots & & a_{1m}B \\ & \ddots & \\ \vdots & a_{ij}B & \vdots \\ & \ddots & \\ a_{m1}B \cdots & & a_{mm}B \end{bmatrix}.$$

The Kronecker product is associative. Also, it is easy to see that the Kronecker product of permutation matrices is a permutation matrix.

Furthermore, the permutation represented by $Q_{n_1} \otimes Q_{n_2} \otimes \cdots \otimes Q_{n_k}$ can be described in "coordinatized" form as follows. It maps

$$j_1 + j_2 n_1 + \cdots + j_k n_1 \cdots n_{k-1}$$

into

$$(j_1 + 1) + (j_2 + 1)n_1 + \cdots + (j_k + 1)n_1 \cdots n_{k-1},$$

where $j_p + 1$ means addition modulo n_p. It is then straightforward to verify that

$$P^{-1} Q_n P = Q_{n_1} \otimes Q_{n_2} \otimes \cdots \otimes Q_{n_k}. \tag{2}$$

Let $A(x)$ be an $n \times n$ circulant. Then

$$A(x) = \sum_{j=0}^{n-1} x_j Q_n^j,$$

where $Q_n^0 = I_n$, the $n \times n$ identity matrix. Thus, applying (2),

$$P^{-1} A(x) P = \sum_{j=0}^{n-1} x_j (Q_{n_1} \otimes \cdots \otimes Q_{n_k})^j$$

$$= \sum_{j=0}^{n-1} x_j (Q_{n_1}^j \otimes \cdots \otimes Q_{n_k}^j) \tag{3}$$

$$= \sum_{j=0}^{n-1} x_j Q_{n_1}^{j_1} \otimes \cdots \otimes Q_{n_k}^{j_k}. \tag{4}$$

Equation (3) follows from the matrix identity $(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D)$, while (4) follows from the identity $Q_p^p = I_p$ for all p. If C_i is an $n_i \times n_i$ circulant for $i = 1, 2, \cdots, k$, then $C_1 \otimes \cdots \otimes C_k$ is an $(n_1, \cdots, n_k)$ supercirculant. Finally, a linear combination of $(n_1, \cdots, n_k)$ supercirculants is an $(n_1, \cdots, n_k)$ supercirculant. Thus $P^{-1} A(x) P$ is an $(n_1, \cdots, n_k)$ supercirculant and the lemma is proved. A more conceptual proof appears in [5]. As a consequence of the supercirculant lemma we obtain the following.

Speedup Lemma: Suppose there is a function $f: N \to N$, where N is the set of positive integers such that for any commutative ring R, the product of two $n \times n$ circulants can be computed with $f(n)$ multiplications. Then, if $n = n_1 \cdots n_k$, with the n_i's relatively prime in pairs, the product of two $n \times n$ circulants can be computed with $f(n_1) \cdots f(n_k)$ multiplications.

Proof: By the supercirculant lemma it suffices to consider multiplication of two $(n_1, \cdots, n_k)$ supercirculants. The proof is by induction on k. The assertion is trivially true for $k = 1$. Assume that it is true for k and let S_1, S_2 be two $(n_1, \cdots, n_k, n_{k+1})$ supercirculants. Let R_k be the set of all $(n_1, \cdots, n_k)$ supercirculants over R. It is easy to see that R_k is a *commutative* ring, under matrix addition and multiplication. S_1 and S_2 can be considered $n_{k+1} \times n_{k+1}$ circulants over R_k. Thus $S_1 S_2$ can be computed using $f(n_{k+1})$ multiplications in R_k. Further, by the induction hypothesis each multiplication in R_k requires $f(n_1) \cdots f(n_k)$ scalar multiplications. Thus the total number of scalar multiplications required is $f(n_1) \cdots f(n_k) f(n_{k+1})$. This proves the lemma.

By Lemma 1, we can take $f(n) = n(n+1)/2$; thus we get the following.

Proposition: Let $n = n_1 \cdots n_k$ with $(n_i, n_j) = 1$ for $i \neq j$. Then the product of two $n \times n$ circulants and thus the convolution of two n-vectors can be computed using $n(n_1 + 1) \cdots (n_k + 1)/2^k$ scalar multiplications.

Remark: It is easy to see that the factorization minimizing the number of multiplications by our method is the complete factorization of n into prime-power factors.

References

[1] W. M. Gentleman and G. Sande, "Fast Fourier transforms-for fun and profit," in *1966 Fall Joint Computer Conf., AFIPS Proc.*, vol. 29, Washington, D. C.: Spartan, 1966, pp. 563–578.

[2] J. W. Cooley, P. A. W. Lewis, and P. D. Welch, "The fast Fourier transform and its applications," *IEEE Trans. Educ.*, vol. E-12, Mar. 1969, pp. 27–34.

[3] P. J. Nicholson, "Algebraic theory of the finite Fourier transform," Ph.D. dissertation, Dep. Oper. Res., Stanford Univ., Stanford, Calif., Mar. 1969.

[4] R. R. Stoner, "A flexible fast Fourier transform algorithm," U. S. Army Electron. Command, Fort Huachuca, Ariz., Rep. ECOM-6046 (CFSTI AD 696 431), Aug. 1969.

[5] P. J. Nicholson and Y. Zalcstein, "Super-circulant matrices and the fast Fourier transform," to appear.

[6] J. Landin, *An Introduction to Algebraic Structures.* Boston: Allyn and Bacon, 1969.

[7] R. Bellman, *Introduction to Matrix Analysis.* New York: McGraw-Hill, 1960.

A State Assignment Procedure for Asynchronous Sequential Circuits

GARY K. MAKI, MEMBER, IEEE, AND JAMES H. TRACEY, MEMBER, IEEE

Abstract—This paper presents a new procedure for constructing nonuniversal shared-row internal state assignments for asynchronous sequential circuits. The method consists basically of establishing an initial code with the minimum number of variables required to dis-

Manuscript received September 12, 1969; revised October 1, 1970, December 2, 1970, and January 13, 1971. This work was supported in part by the National Science Foundation under Grants GK-2017 and GK-20190.

G. K. Maki is with the Department of Electrical Engineering, University of Idaho, Moscow, Idaho 83843.

J. H. Tracey is with the Department of Electrical Engineering, University of Missouri at Rolla, Rolla, Mo. 65401.

tinguish the states. Then variables are systematically added until a satisfactory assignment is obtained. State variables added in the expansion of an assignment are merely the EXCLUSIVE OR of state variables in the original assignment. This simple construction procedure terminates with a maximum of $m+[m/2]$ state variables for a 2^m-row flow table.

Index Terms—Asynchronous sequential circuits, internal state assignment, nonuniversal assignment techniques, shared-row assignments.

I. Introduction

The operation of an asynchronous sequential circuit can be described by means of a flow table (see Fig. 1). An important step in the synthesis procedure for these circuits consists of coding the internal states such that all state transitions can be accomplished without the introduction of critical races [1]. The internal state variables, $y_1, y_2, \cdots, y_n$, and corresponding code are called the internal state assignment. The type of assignment considered here is one in which a single y state is assigned to each row of the flow table and remaining y states are used as necessary to complete transitions between nonadjacent rows. The result of such an assignment is that, after an input change, the circuit may sequence through several internal states before becoming stable. Unger refers to these assignments as *shared-row state assignments* [2]. This should be contrasted with *single transition time* assignments where all internal state transitions go directly from unstable states to terminal stable states; no sequencing is permitted. Generally, fewer state variables are required for shared-row assignments.

A state assignment is said to be *universal* if it is valid for any flow table with a given number of rows; otherwise, it is *nonuniversal*. A nonuniversal assignment for a particular flow table frequently requires fewer state variables than a universal assignment. Procedures are well established for generating nonuniversal single transition time assignments for asynchronous sequential circuits [3], [4]. This note treats the more difficult problem of generating nonuniversal shared-row assignments [5]–[7].

The assignment procedure presented here does not guarantee a minimum-variable assignment but terminates with $m+[m/2]$ state variables for a 2^m-row flow table, where

$$[m/2] \triangleq \begin{cases} m/2, & m \text{ even} \\ (m-1)/2, & m \text{ odd.} \end{cases}$$

For flow tables where the number of rows are equal to or less than 2^m (and greater than 2^{m-1}), the method consists of first establishing an initial code of m state variables. Additional state variables are obtained, as necessary, by EXCLUSIVE-ORING variables from the initial code.

It has not been proven that $m+[m/2]$ state variables are always sufficient for the coding of a 2^m-row flow table by the procedures of this note. However, a search for counterexamples has been unsuccessful.

II. Development of the Assignment Procedure

A *k-set* of a flow table column consists of $k-1$ unstable states leading to the same stable state, together with that stable state. A *column partition* is a collection of the k-sets that appear in a column of a flow table. The column partitions for the flow table in Fig. 1 are

	I_1	I_2	I_3
1	(1)	3	5
2	1	5	(2)
3	1	(3)	2
4	(4)	5	2
5	4	(5)	(5)

Fig. 1. Flow table.

$$\alpha_1 = \{1, 2, 3; 4, 5\}$$
$$\alpha_2 = \{1, 3; 2, 4, 5\}$$
$$\alpha_3 = \{2, 3, 4; 1, 5\}.$$

If a transition is to be effected between a pair of states S_a and S_b, then the pair of states S_a, S_b is called a *transition pair*. A *transition path* is the set of n-tuples that the circuit assumes in undergoing a transition between states of a transition pair.

The following definitions are useful in the discussion of the assignment procedure.

Definition: A set of state variables $\{y_1, y_2, \cdots, y_k\}$ is an *independent set of state variables* if no y_i in the set is equal to the mod 2 sum $\oplus$ of a subset of the other state variables of the set.

Definition: Consider a set of two independent and distinct state variables $\{y_\xi, y_\delta\}$. If $y_\xi \oplus y_\delta = y_\eta$, then the complete set $\{y_\xi, y_\delta, y_\eta\}$ is called a *parity set*. State variable y_η is generated from the independent state variables and is therefore called *dependent*.

III. Statement and Illustration of the Assignment

A valid set of transition paths for a column partition must not have any n-tuples that are common to different k-sets. If the column partition consists of only two-sets, then there is a maximum number of transition paths with no elements in common. Therefore, it appears that the most difficult type of flow table to work with is one composed of column partitions with only two-sets. The assignment procedure will be discussed and illustrated for this case first. It will be assumed that the number of flow table rows is between 2^{m-1} and 2^m. Next, the procedure will be generalized to the case where k-sets are not restricted to two-sets. Details regarding the rationale for each step can be found elsewhere [6].

Step 1: Code the states with m state variables y_1, $y_2, \cdots, y_m$ in such a manner that a minimum number of transition pairs are a maximum distance apart. In general, best results are obtained if the states of transition pairs are assigned codes that are close together.

Step 2: Count the number of times each state variable y_i, $i=1, 2, \cdots, m$, must be excited to effect a distance two or greater transition. This forms the count list.

Step 3: Let y_j and y_k be two highest value state variables in the count list that are not yet in a parity set. Generate a new state variable y_1 such that $y_1 = y_j \oplus y_k$. Then y_1, together with y_j and y_k, forms a parity set.

Step 4: Attempt to find the transition paths for each column partition. If a satisfactory set of transition paths exists, then the assignment procedure is complete. If not,

then return to Step 3 and generate another state variable. The process continues until a set of transition paths is found.

The following is an example to illustrate the procedure. Consider the six following column partitions which correspond to a flow table called machine A:

$$\alpha_1 = \{0, 9; 7, 13; 3, 1; 10, 4; 2, 8; 11, 15; 6, 14; 5, 12\}$$
$$\alpha_2 = \{0, 6; 2, 7; 3, 13; 9,1; 10, 15; 4, 12; 14, 8; 11, 5\}$$
$$\alpha_3 = \{9, 4; 7, 10; 0, 12; 3, 2; 15, 6; 5, 13; 8, 14; 1, 11\}$$
$$\alpha_4 = \{0, 14; 7, 15; 9, 12; 3, 5; 4, 6; 11, 13; 10, 8; 1, 2\}$$
$$\alpha_5 = \{7, 10; 2, 4; 15, 12; 0, 6; 3, 13; 1, 9; 8, 14; 11, 5\}$$
$$\alpha_6 = \{0, 13; 7, 15; 3, 5; 9, 14; 4, 8; 1, 11; 10, 12; 6, 2\}.$$

The codes assigned to the states of the m-cube (four-cube) are shown in Fig. 2. It is possible in this case to avoid coding any transition pair with codes a maximum distance apart.

Next, the number of times each state variable is excited, in a distance two or greater transition, are tabulated as follows.

State Variable	Number of Times Excited
y_1	14
y_2	15
y_3	12
y_4	16

On the basis of this count list, a new state variable y_5 is generated from state variables y_2 and y_4 with $y_5 = y_2 \oplus y_4$. The Karnaugh map representation of this assignment is shown in Fig. 3.

Transition paths can be constructed within each column, and therefore the assignment is satisfactory. The amount of effort involved in finding this state assignment appears to be significantly less than other known techniques.

It is interesting to note that the assignment procedure just described and illustrated always produces the *universal* four-variable assignment given by Saucier [7].

For the more general case where k-sets are not restricted to two-sets, the above algorithm can be easily modified. Each k-set is converted to a set of transition pairs. For example, consider a three-set consisting of states 1, 2, and 3, with state 1 as the stable state. There are three possible sets of transition pairs for a three-set: (1, 2) (1, 3); (1, 2) (2, 3); (1, 3) (2, 3); where the parentheses denote the transition pair. It is apparent that more flexibility is associated with k-sets where k is greater than two. To account for this, in the initial coding attention is given primarily to keeping the codes of the two-sets as close together as possible. Little effort is expended in minimizing the distance between states of k-sets greater than two-sets. Transition pairs that consist of states that are adjacent are most desirable in that the resulting transition path contains just the two states. The means of determining which independent state variables are to be used in generating the additional state variables is identical to the procedure previously discussed for the all-two-set case.

The final code depends on the flow table characteristics and the initial encoding. Therefore, when all transitions

y_3y_4 \ y_1y_2	00	01	11	10
00	0	9	13	14
01	6	11	1	2
11	15	4	5	7
10	12	10	3	8

Fig. 2. Initial coding for machine A.

$y_5 = 0$

y_3y_4 \ y_1y_2	00	01	11	10
00	0			14
01		11	1	
11		4	5	
10	12			8

$y_5 = 1$

y_3y_4 \ y_1y_2	00	01	11	10
00		9	13	
01	6			2
11	15			7
10		10	3	

Fig. 3. Trial state assignment for machine A.

cannot be made with the constructed assignment, one may choose to experiment with a modification of the initial code before adding another state variable to the assignment.

References

[1] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, Mar./Apr. 1954, pp. 161–190, 275–303.

[2] S. H. Unger, *Asynchronous Switching Circuits*. New York: Wiley, 1969, p. 71.

[3] C. N. Liu, "A state variable assignment method for asynchronous sequential switching circuit," *J. Ass. Comput. Mach.*, vol. 10, Apr. 1963, pp. 209–216.

[4] J. H. Tracey, "Internal state assignments for asynchronous sequential machines," *IEEE Trans. Electron. Comput.*, vol. EC-15, Aug. 1966, pp. 551–560.

[5] B. Hazeltine, "Encoding of asynchronous sequential circuits," *IEEE Trans. Electron. Comput.* (Short Notes), vol. EC-14, Oct. 1965, pp. 727–729.

[6] G. K. Maki, "State assignments for non-normal asynchronous sequential circuits," Ph.D. dissertation, University of Missouri, Rolla, July 1969.

[7] G. Saucier, "Encoding of asynchronous sequential networks," *IEEE Trans. Electron. Comput.* (Short Notes), vol. EC-16, June 1967, pp. 365–369.

Serial Adders with Overflow Correction

R. O. BERG AND L. L. KINNEY, MEMBER, IEEE

Abstract—A method of implementing two single-bit adders is discussed. These adders can be used individually to realize the conventional functions of serial addition and serial multiplication on a pair of operands, or they can be cascaded to allow the serial addition of three operands for forming the product of complex numbers. In either case, the circuits will detect the occurrence of an overflow or the generation of the number minus one, and they will allow an addition to be rescaled by outputting the correct bits during the additional shifts, whether the addition overflowed or not.

Index Terms—Cascaded adders, overflow detection and correction, parallel processing, serial addition.

Manuscript received June 16, 1970; revised January 4, 1971.

R. O. Berg is with Honeywell Systems and Research Center, Minneapolis, Minn.

L. L. Kinney is with the Department of Electrical Engineering, University of Minnesota, Minneapolis, Minn.