Ferruccio Barsi was born in Lucca, Italy, on January 3, 1942. He received the Dr. Ing. degree in electronic engineering from the University of Pisa, Pisa, Italy, in 1969. Since 1969 he has been with the Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy. He has been engaged in research on minicomputer architecture, fault diagnosis, and arithmetic codes.

Piero Maestrini was born in Colle di Val d'Elsa, Italy, on January 9, 1938. He received the Dr. Ing. degree in electronic engineering (cum laude) from the University of Pisa, Pisa, Italy, in 1962, and the Libera Docenza in switching theory and electronic computers in 1971. Since 1962 he has been with the Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy. Since 1969 he has also been an Associate Professor with the Department of Electrical and Electronic Engineering, University of Pisa. He has been engaged in research concerning digital electronics, computer architecture, microprogramming, and arithmetic codes.

The calculation of satisfactory state assignments for asynchronous sequential circuits is one of the most difficult tasks in the synthesis procedure for such networks. This paper considers the state assignment problem for circuits operating in the normal fundamental mode and describes a new procedure especially suited to the automated synthesis of very large circuits.

Operation of the machine to be synthesized is assumed to be described by a flow table, which may have been previously simplified or minimized [1]. The reader is assumed to be familiar with this form of specification, illustrated in Fig. 1.

The complexity of the state assignment procedure for asynchronous sequential circuits stems from the necessity of avoiding critical races, which may cause the machine to malfunction. (A critical race occurs when, due to the asynchronous nature of state transitions, internal state variables may change values in an order which allows the circuit to reach a final stable state other than the intended one.)

Critical races may be avoided by using standard state assignments which do not depend on flow table characteristics [2]. However, it is often possible to code a given flow table with significantly fewer variables than required for standard assignments. A standard assignment may be rapidly generated for tables of a given size, since it may be used with any table of that size, regardless of behavior. The often smaller nonstandard assignment, however, depends on next-state specifications for a particular machine.

Generation of Internal State Assignments for Large Asynchronous Sequential Machines

ROBERT J. SMITH, II, MEMBER, IEEE

Abstract—Several algorithms have been proposed for generating satisfactory state assignments for normal fundamental mode asynchronous sequential circuits. Programmed versions of minimal and near-minimal techniques have been incorporated in automated synthesis systems, but are known to require computational efforts which increase exponentially with flow table size. Other methods, requiring much less effort, produce assignments with far more state variables than minimal assignments.

This table presents a practical method for generating near-minimal uncode single-transition-time state assignments for very large flow tables. The new technique has been programmed and thus used to generate assignments for tables of up to 400 cells. It is shown that resources required to generate practical assignments for very large tables increases almost linearly with flow table size using the procedure described.

Index Terms—Asynchronous sequential circuits, design automation heuristics, state assignments.
Nonstandard state assignment generation techniques have been a popular research topic in recent years. In an early paper, [3] C. N. Liu showed that a critical-race-free nonstandard assignment could be formed by combining individual column codes which are themselves critical race free; column codes are obtained by considering transitions from unstable to stable states within a column. Liu's technique is not computationally complex, but frequently produces assignments with many more variables than necessary; for tables with many columns, the number of variables may approach or even exceed the number required for a standard assignment.

Tracey later described a method for finding minimum variable nonstandard unicode assignments for normal, fundamental mode asynchronous sequential circuits [4]. These restrictions indicate that inputs change only while the machine is in a stable internal state, that a single code is applied to each internal state, and that when inputs do change, transitions are made directly to the destination stable state. Such assignments are also known as unicode single transition time (USTT) state assignments. Tracey noted that for "large" (8–12 row) flow tables, the effort required to calculate minimum variable USTT codes by his method becomes prohibitive. He therefore suggested two related algorithms which would require less effort, but would usually produce near minimum variable assignments.

In earlier work [5] the author demonstrated that for tables of more than about 50 cells (rows x columns), even automated generation of minimum-variable assignments is not practical. Furthermore, it was discovered that for very large flow tables (more than 150 cells), even Tracey's near-minimum variable methods required extremely large computation times. This paper presents an extension of Tracey's techniques which produces near-minimum variable assignments for very large tables, while requiring much less computational effort than previous methods.

**TRACEY'S METHOD**

The techniques described by Tracey are based on the idea that critical races can be avoided by devising state assignments which partition internal state transitions under each input (flow table column). (For more complete information on these methods, see [4].) Minimum variable assignments may be produced by first recording a Boolean transition constraint matrix, shown for machine 6 in Fig. 2. (Note that the first block of each partition has been arbitrarily coded with zeros.) Minimal reduction of the transition constraint matrix is achieved by calculating maximally intersectable classes of constraints, then covering all initial constraint with a set of maximal intersectables. One of several 4-variable assignments found in this manner for machine 6 is

\[
\begin{align*}
y_1 &= 000110 \\
y_2 &= 001101 \\
y_3 &= 010110 \\
y_4 &= 100111.
\end{align*}
\]

For tables significantly larger than machine 6, minimization of the transition constraint matrix becomes a very formidable task. One 12 row by 4 column table from the literature requires 50 transition constraints, and has more than 300 maximal intersectable classes. Even using very large computers, it has been economically impractical to calculate minimum variable assignments for tables with more than a few dozen constraints. (Since the number of
Fig. 2. Transition partitions and corresponding constraint matrix for machine 6. (a) Transition partitions. (b) Boolean constant matrix.

Fig. 3. Assignment generation for machine 6 using k-set constraints. (a) K-sets which must be partitioned. (b) K-set partitions. (c) K-set constraint matrix. (d) Assignment obtained using minimal reduction.

constraints is related to number of next state entries rather than number of rows, the number of cells [rows * columns] is used here as a rough measure of flow table complexity.)

Tracey proposed a nearly minimum constraint matrix reduction procedure to allow computation of assignments for larger tables. However, for very large tables, even this technique is unable to reduce the required transition constraint matrices.

To overcome these computational difficulties, Tracey proposed a further simplification of the constraint matrix reduction problem intended to significantly reduce computation effort while producing "nearly minimal" assignments for large flow tables. Consider the k-sets of each flow table column, each consisting of k-1 unstable next state entries in a single column which all specify the same stable next state, which is also included in the k-set. K-sets for machine 6 are shown in Fig. 3(a). Note that a direct transition in k-set k, does not race critically with direct transitions in any of other k-set k_j if at least one y-variable of the state assignment partitions elements of k_i and k_j into separate blocks (4). Thus, the number of constraints required to produce critical-race-free assignments can be reduced to the number of stable states plus the number of table row pairs, or less. Fig. 3(b) shows the k-set constraint matrix for machine 6, followed by the state assignment resulting from minimal reduction of that matrix.

This method usually decreases the size of the constraint matrix significantly, thereby reducing the effort required to obtain compact nonstandard assignments. Utilization of constraints based on k-set membership frequently precludes generation of minimum-variable assignments. Those assignments found are, however, almost always much smaller than corresponding standard assignments. Limited
experiments with tables of approximately 50 cells show that use of $k$-set rather than transition partitions may be expected to introduce less than 20 percent additional state variables (see Fig. 9).

For large tables of more than 75 cells, it can be shown that minimal reduction of even $k$-set constraint matrices is impractical. Fig. 4 shows a comparison of assignment generation times, using the following Tracey schemes.

1) Transition partition constraint matrix, reduced to a minimum state assignment using a minimum cover of maximal intersectables.

2) Transition partition constraint matrix, reduced to a near-minimum variable assignment.

3) $K$-set partition constraint matrix, minimized to produce a near-minimum variable assignment.

4) $K$-set partition constraint matrix, reduced (but not minimized) to produce a nonstandard assignment.

Data summarized in Fig. 4 were obtained using identical flow tables for each of the assignment methods indicated. Note that for tables with over 200 cells, even method 4) requires a nontrivial amount of computation. Indeed, this technique becomes impractical for very large problems, since Tracey's nonminimal reduction of constraint matrices requires computation effort proportional to the square of matrix size. (Effort required to generate necessary constraints has been found insignificant when compared with matrix reduction requirements.)

AN ASSIGNMENT TECHNIQUE FOR LARGE TABLES

Because of the exponentially increasing computation times for large problems, none of the Tracey assignment methods appear to be suitable for dense (mostly specified) flow tables of more than about 250 cells. The remainder of
this paper describes a modification of Tracey's $k$-set partition assignment algorithm which permits economical generation of internal state codes for extremely large tables.

The strategy employed is a simple one: $K$-set and row-pair partitioning constraints are generated, as previously described. However, the constraint matrix is only allowed to grow to a predetermined size limit, then is partially reduced. This procedure will, in general, produce assignments having at least as many variables as the Tracey methods. It will be shown, however, that the new method can be used to rapidly determine assignments for extremely large tables.

The new state assignment procedure begins by finding $k$-set partitions for each flow table column. When the partition matrix reaches the size limit, the matrix is partially reduced, yielding state variables and a small number of constraints not satisfied by the variables generated.

$K$-set partition generation then resumes. However, a $k$-set partition is not added to the constraint list if it can be satisfied by a previously calculated state variable. The constraint matrix thus contains only those $k$-set partitions which remain to be satisfied. When this constraint list again reaches the size limit, the partial matrix reduction procedure is repeated.

After all $k$-set partitions have been found, the state variable and partition lists must be checked to insure that each flow table row is partitioned by some variable from every other flow table row. Any constraints needed to satisfy this requirement are added to the partition matrix, and it is completely reduced.

Figs. 5 and 6 summarize the large flow table state
assignment procedure. The matrix reduction technique is derived from Tracey's algorithm 2 [4], modified to permit the maximum number of constraints not included in the partial assignment to be a passed parameter.

Fig. 7 shows a 12 row by 4 column flow table which will be used to illustrate the state assignment procedure. It should be noted that the algorithm described herein is not particularly well suited to tables as small as this example; it is presented primarily to illustrate the technique.

Assume that constraint matrix size limits are 10 rows maximum and 2 rows minimum; these are unrealistically low, but are chosen for illustrative purposes. As shown in Fig. 8, the matrix is reduced in two segments, producing a six variable assignment.

The state assignment method outlined above produces codes more economically than the Tracey algorithms because the amount of computation required to reduce a Boolean matrix is much greater for schemes which consider the entire matrix than for methods which reduce relatively small matrix segments. It is, for example, much simpler to reduce four matrix segments of 50 rows each than to reduce one 200 row matrix.

Two programmed implementations of the new state assignment procedure have been used to investigate the effects of varying matrix size and residue limits. Results are problem dependent, but programming considerations also appear to influence choice of optimum values for these parameters. An early PL/1 version [5] produced best results with a matrix size limit of 20 to 25 constraints and a residue maximum of two to five constraints. A much more efficient Fortran program recently developed appears to be fastest with a matrix size limit of 30–35 (33 was used in experiments summarized in Fig. 8), and a residue limit of five to eight.

Residue limits below five tend to produce assignments with slightly more variables, with no apparent savings in time; the same effect is observable for low constraint matrix size limits. Calculation times rise with increasing matrix size limits, typically producing slightly smaller assignments with 25–50 percent more effort using a 100 constraint limit.

Relatively efficient Fortran implementations of Tracey's techniques and the new method described here have been created. These computer programs have been used to calculate assignments for a large number of flow tables having similar properties. The state assignment experiments summarized in Fig. 9 used the flow tables characterized in Appendix I. Programs utilized were written in Fortran and run on a CDC 1604. Undoubtedly, faster hardware or assembly level coding would improve the execution time cited. Assignment methods are those used in the text, with "SR" designating the segmented reduction pro-

Fig. 6. State assignment procedure for large flow tables.
Fig. 7. Machine 4 flow table.

<table>
<thead>
<tr>
<th>Internal States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input States</td>
<td>00</td>
<td>01</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>000</td>
<td>001</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>001</td>
<td>6</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>1</td>
<td>--</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>010</td>
<td>6</td>
<td>--</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>01</td>
<td>6</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>01</td>
<td>10</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>01</td>
<td>100</td>
<td>5</td>
<td>--</td>
</tr>
<tr>
<td>8</td>
<td>01</td>
<td>7</td>
<td>10</td>
<td>011</td>
</tr>
<tr>
<td>9</td>
<td>01</td>
<td>7</td>
<td>10</td>
<td>011</td>
</tr>
<tr>
<td>10</td>
<td>01</td>
<td>7</td>
<td>10</td>
<td>011</td>
</tr>
<tr>
<td>11</td>
<td>01</td>
<td>7</td>
<td>10</td>
<td>110</td>
</tr>
<tr>
<td>12</td>
<td>01</td>
<td>7</td>
<td>10</td>
<td>110</td>
</tr>
</tbody>
</table>

Fig. 8. Constraint matrix segment reduction for machine 4.

procedure described herein. Fig. 9 shows a comparison of representative performance observed for very large tables, utilizing Tracey's method "D" and the constraint segment reduction procedure described here.

SUMMARY

This paper has described a technique for generating nonstandard USTT internal state assignments for very large normal, fundamental mode asynchronous sequential
circuits. The procedure avoids computational difficulties encountered using previous minimum or near-minimum variable assignment methods by reducing segments of the constraint matrix, as it is generated. Computer programs were developed to compare performance of the new technique with previous methods; these experiments verified the performance advantages of the new procedure.

APPENDIX I

FLOW TABLE CHARACTERISTICS

Flow tables used in the state assignment experiments described in this paper are characterized in Table I. Except where indicated, the tables were derived from pseudorandom transition sequences.

<table>
<thead>
<tr>
<th>Table Reference Number</th>
<th>Rows</th>
<th>Columns</th>
<th>Cells</th>
<th>K-Sets</th>
<th>Stable States</th>
<th>Unstable States</th>
<th>Unspecified Next-state</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>48</td>
<td>14</td>
<td>14</td>
<td>27</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>4</td>
<td>24</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>4</td>
<td>28</td>
<td>12</td>
<td>12</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>40</td>
<td>10</td>
<td>400</td>
<td>101</td>
<td>101</td>
<td>274</td>
<td>25</td>
</tr>
<tr>
<td>22</td>
<td>39</td>
<td>10</td>
<td>390</td>
<td>106</td>
<td>106</td>
<td>262</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>36</td>
<td>11</td>
<td>396</td>
<td>115</td>
<td>115</td>
<td>271</td>
<td>10</td>
</tr>
<tr>
<td>24</td>
<td>15</td>
<td>10</td>
<td>150</td>
<td>40</td>
<td>40</td>
<td>95</td>
<td>9</td>
</tr>
<tr>
<td>25</td>
<td>12</td>
<td>12</td>
<td>144</td>
<td>20</td>
<td>20</td>
<td>91</td>
<td>23</td>
</tr>
<tr>
<td>26</td>
<td>13</td>
<td>11</td>
<td>143</td>
<td>45</td>
<td>45</td>
<td>78</td>
<td>20</td>
</tr>
<tr>
<td>28</td>
<td>25</td>
<td>10</td>
<td>250</td>
<td>72</td>
<td>72</td>
<td>154</td>
<td>24</td>
</tr>
<tr>
<td>29</td>
<td>20</td>
<td>13</td>
<td>260</td>
<td>74</td>
<td>74</td>
<td>150</td>
<td>36</td>
</tr>
<tr>
<td>32</td>
<td>9</td>
<td>4</td>
<td>38</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>33</td>
<td>10</td>
<td>4</td>
<td>40</td>
<td>17</td>
<td>17</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>35</td>
<td>12</td>
<td>5</td>
<td>60</td>
<td>25</td>
<td>25</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>36</td>
<td>10</td>
<td>6</td>
<td>60</td>
<td>24</td>
<td>24</td>
<td>29</td>
<td>7</td>
</tr>
<tr>
<td>37</td>
<td>6</td>
<td>4</td>
<td>24</td>
<td>11</td>
<td>11</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>7</td>
<td>3</td>
<td>21</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
REFERENCES


Robert J. Smith, II (S’66–M’70) was born in San Francisco, Calif. in 1944. He received the B.S. degree in electrical engineering from Wichita State University, Wichita, Kans., in 1966, and M.S. and Ph.D. degrees in electrical engineering from the University of Missouri at Rolla, Rolla, in 1968 and 1970, respectively. In 1965 and 1966 he was a Tool Design Engineer with the Boeing Corporation Wichita, Kans. He joined the staff of the Missouri Institute of Psychiatry in 1970, where he worked on computer applications in psychiatric medicine, and became Director of the MIP computer center. In 1971 and 1972 he served as an EDP Plans and Operations Officer at Aberdeen Proving Ground, Aberdeen, Md. He is currently an Assistant Professor of Electrical Engineering and Computer Science in the Institute of Technology, Southern Methodist University, Dallas, Tex. His present research interests are in the areas of digital design automation, computer organization and biomedical computing applications.

Dr. Smith is a member of Eta Kappa Nu, Tau Beta Pi, the Association for Computing Machinery, and the American Society for Engineering Education.

Discretization Error Analysis in Linear DDA Connections

LAURI HAKKALA AND LEO OJALA

Abstract—In this paper mathematical models for digital integrators operating with multidigit increments are given, and the discretization error analysis of their linear connections is studied. The integration methods considered are linear multistep integration methods; some examples are studied in detail. Linear digital differential analyzer (DDA) connections are defined, and their use in connection with the numerical solution of the initial value problem is explained. Finally, the discretization error in these linear DDA connections has been analyzed in detail; the theoretical results have also been verified experimentally.

Index Terms—DDA, digital incremental integrators, discretization error analysis, linear DDA connections, linear multistep integration methods, multidigit increments.

I. INTRODUCTION

The digital differential analyzer (DDA) was developed to replace the analog computer in the solution of ordinary differential equations. The progress has been very rapid: the first electronic DDA "MADDIDA" (magnetic drum digital differential analyzer) was built in 1950; today there are several types of DDA's with improved techniques and increased speed and accuracy [1], [2].

The operation principle of the conventional fixed-point DDA is very simple [3], [4]. The basic unit is an integrator which usually uses the rectangular (Euler) or the trapezoidal integration rule, and the communication between these units is conveyed by binary or ternary transfer. The major disadvantage of the conventional DDA is its rather poor accuracy or computing speed. Thus the next step in the evolution of the DDA is to increase the amount of information conveyed between the DDA integrator units: the use of multidigit incremental integrators [5]. An additional improvement is to use more efficient and accurate integration methods [6].

This paper presents a mathematical model for a special type of DDA integrator: the model of the multidigit digital integrator using a general linear multistep integration method. The information between the integrator units is transferred by words consisting of at most M binary digits differing from zero, so-called M-increments.

Furthermore, the concept of a linear DDA connection is introduced. This is a special arrangement of DDA integrators which can be used to solve numerically a certain class of initial value problems, namely, the class...