Analysis and Characterization of State Assignment Techniques for Sequential Machines

R.Z. MAKKI

University of North Carolina at Charlotte, Charlotte, NC 28223, USA

S. SU

Hewlett Packard, Fort Collins, Colorado 80523, USA

(Received January 11, 1990; Revised August 1, 1992)

In this paper, we study the problem of state assignment as it relates to silicon area, propagation delay time and testability of finite state machines. The results of a study involving various FSM benchmarks show that the simple technique of one-hot encoding often produces better results than those attained by complex state assignment algorithms.

Key Words: *Finite State Machine; One-hot encoding; Area; Delay time; Testability*

I. INTRODUCTION

Automated synthesis of finite state machines (FSMs) typically refers to a process by which a high level description of an FSM is transformed into a gate-level netlist. Beginning with a state transition table, a typical synthesis process is shown in Figure 1. An important step in the synthesis process is the state assignment. The state assignment has a direct influence, in part, on many parameters in the resulting circuit such as circuit area, circuit performance, and circuit testability [1]–[4]. Examples of state assignment tools include KISS [2], MUSTANG [5], NOVA [6], MUSE [7], and JEDI [8]. Two-level minimization seeks to reduce the number of terms in the Boolean equations that realize the FSM by reducing redundancy and exploiting don't care conditions. An example of a two-level minimization tool is ESPRESSO [9]. Multi-level minimization refers to factoring two-level Boolean equations into multiple levels. The objective is to maximize the sharing of common Boolean expressions in order to reduce the number of required logic literals. Examples of multi-level minimization tools include MIS [10] and DECAF [11]. Technology mapping of multi-level Boolean equations is performed in order to map the equations into standard cells. Consideration is given

to minimizing a cost function subject to design constraints such as speed and area. After technology mapping, a netlist is generated which can be interfaced to silicon compilation tools for placement and routing. Although each step in Figure 1 is shown as a separate entity, the steps are interdependent. For example, state assignment algorithms that target two-level minimization may not be efficient for multi-level minimization [5].

In the synthesis of an FSM, important criteria are considered which include in part:

- 1) silicon area,
- 2) propagation delay time, and
- 3) testability.

For each of the above criteria, the particular state assignment which is chosen plays a crucial and central role. In this paper, we study the problem of state assignment as it relates to the above criteria for the case of multi-level minimization. The study involves the analysis and characterization of various classes of state assignment techniques in relation to the above criteria. The result of a study involving various FSM benchmarks is presented. The results show that the simple technique of one-hot encoding [12] often produces better results than those attained by complex state assignment algorithms.

FIGURE 1 A typical logic synthesis process beginning with a state transition table.

II. PRELIMINARY DISCUSSION

In general, an FSM $F(NSD, M, OD)$ is composed of three sections as shown in Figure 2. The next state decoder (NSD) and output decoder (OD) are composed of combinational logic (and are often grouped into a single combinational logic block), and the memory (M) consists of a set of flip-flops. We shall restrict M to be composed of D-type flip-flops organized as a parallel-in parallel-out (PIPO) shift register in the regular mode of operation. An FSM can be described via a state transition diagram (STD) or via a state transition table (STT). The STD and STT describe the set of FSM states, the conditions that

cause a transition from one state to another, and the conditions that cause the assertion of output signals. For a given STD, a state assignment is made by assigning a unique binary code to each of its states. As measured by the number of bits, the length of the state assignment code, N , determines the size of the PIPO in M. An STD having P states requires a minimum length code of size $\lceil \log_2 P \rceil$ where $\lceil \log_2 P \rceil$ is the least integer greater than or equal to $\log_2 P$. However, utilizing a minimum length code does not necessarily result in an optimum solution relative to the area, speed, and testability of the FSM (see Section IV).

We shall assign a cost function $C(A, D, T)$ to a given FSM, where A is the silicon area, D is the worst-case propagation delay time, and T is the testability. Each of the parameters A , D , and T is a complex function and is interrelated to the other two. In the following, we shall examine the effect of state assignment on each of the above parameters.

Area

For the purpose of quantifying the silicon area, A , occupied by an FSM, we shall assume that the process technology and layout style are fixed. We shall further assume that a standard set of cells is used for the layout. Now, A can be separated into two parts: the first is the area occupied by the cells, and the second is the area occupied by routing. Thus, A depends on the number of required cells and the regularity of the circuit structure. In order to optimize the function of A , one has to study the effect of state assignment on NSD, M, and OD. It has been established (see above) that the code length N determines the size of M. In addition to N , the actual composition of the code also influences A . It has been shown that a significant improvement can be achieved by having the state assignment satisfy certain constraints which are derived by applying a number of rules [13]. Traditionally, the following three rules have been used to identify states which are to

FIGURE 2 General model of a finite state machine.

be given adjacent state assignments (i.e., two codes are adjacent if they differ in only 1 bit position):

Rule 1. states having a common next-state,

Rule 2. next-states of a given state, and

Rule 3. states having common outputs.

The use of the above rules can help to reduce the amount of logic in NSD (rules 1 and 2) and OD (rule 3). It has been shown that using variations of the above rules, which also include the input space of the FSM, to derive weighted constraints can be effective for multi-level minimization [5].

Delay Time

In general, OD is a function of the present state and the present input combination. The present state is computed from the previous state via NSD. Thus in order to compute the next outputs, logic signals are propagated through NSD, M, and OD. Hence, in general, the worst-case propagation delay path traverses a set of logic elements belonging to NSD, M, and OD. Since M consists of a PIPO, a state assignment that produces a wider PIPO (more bits) would not, at least to a first order, increase the propagation delay time through M (this does not take into account additional capacitance and power dissipation). However, if such a state assignment produces less logic in NSD and OD, the value of D would be reduced (everything else being equal). In other words, if increasing the size of M results in less logic in NSD and OD, we would expect a speed improvement. Indeed this was the case for most of the FSMs which were studied as presented in Section IV.

Testability

Testability refers to the degree to which internal circuit nodes can be controlled and observed. Test generation for sequential circuits has been recognized to be especially difficult [14]. For example, in order to detect a fault in NSD, the FSM has to be placed in an appropriate state. However, this is a difficult task for two main reasons. First, the flip-flops are a coded representation of the states and must be set to the appropriate code so as to propagate the fault from NSD through M and on to the outputs of OD. Second, the feedback paths pose a problem because they are not directly controllable. With proper Design-For-Test such as the use of scan paths [14]–[16], the test process can be simplified. In the scan path approach, the flip-flops in M are threaded together to form a serial shift register in the test mode of operation. The combinational portion of the FSM is

separated from M so that standard test algorithms for combinational circuits such as D[17], PODEM [18], and FAN[19] can be utilized. Unfortunately scan paths serialize the test process and can result in an unacceptable amount of overhead for some applications having tight performance and silicon area requirements.

Recently, it has been shown that there exists an intimate relationship between the synthesis approach chosen for an FSM and the testability of the FSM [3]. A set of codes of distance-2 was used to enhance the testability of FSMs at the cost of a 0% to 30% circuit area overhead for the benchmark samples which were presented in the study. Also, in a previous paper [20], it was demonstrated that the one-hot state assignment results in circuits that can be easily tested without the use of scan paths. In the next section we will briefly review the reasons for the positive effect that one-hot encoding has on testability. In Section IV, we compare the one-hot state assignment with other types of state assignment techniques relative to testability.

III. ANALYSIS

In the past, designers have sought to use a minimum length code in order to reduce the cost of M because flip-flops were expensive. However, in VLSI CMOS applications a D-type dynamic flip-flop can be realized with eight transistors [21]. Furthermore, the amount of logic in NSD and OD depends, in part, on the code length N. For many FSMs, as the value of N is increased the amount of logic in NSD and OD begins to decrease (to a certain limit) because: 1) of the availability of more don't care conditions which can be used in minimization, and 2) a larger N implies the existence of more state variables which can be used to satisfy more of the adjacency constraints. Thus optimizing N (i.e., a minimum length code) only guarantees an optimum M but does not necessarily imply that A is optimum because A is affected by M as well as NSD and OD. The issue is to determine the length and composition of the state code which optimizes A for the single entity composed of the parts NSD, M, and OD. Similar arguments can be made for D (Delay) and T (Testability). For example, the testability of an FSM is not only affected by the amount of sequential elements in M but also by the amount of logic in NSD and OD. We have found that a one-hot state assignment, which utilizes a maximum length code where N is equal to the number of states, results in easily testable circuits. The fact that each flip-flop in the FSM repre-

sents a unique state in one-hot encoding, and that only one flip-flop can be at logic “1” in any given clock cycle greatly simplifies the process of path sensitization. Once the FSM is initialized to a given state, a path from the state to an output point can be easily sensitized by using the state-to-state sequencing information given in the STD. Another advantage of one-hot encoding is that the STD maps directly into hardware. The resulting circuit has a one-to-one correspondence with the STD which allows for test generation at the STD level (for gate-level faults). A test algorithm called Branch Testing has been developed for one-hot encoded FSMs [18]. Branch Testing identifies a minimum set of paths (between the top-most state and the bottom state) containing all the edges in the STD and then proceeds to sensitize each path using the STD sequencing information. Branch Testing requires the use of only three test points and a 2-way multiplexer regardless of the size of the FSM. More detailed information on Branch Testing can be found in reference 18.

In order to quantify the effect of state assignment on A, D, and T, we developed three state assignment tools.

Tool 1

Only two rules are utilized to obtain a constraint matrix that specifies the states which are targeted to be adjacent:

Rule 1. states having a common next-state with the transition controlled by the same input

literal (i.e., a literal is a logic variable or its complement) as shown in Figure 3a, and

Rule 2. next states of the same state as shown in Figure 3b.

Once the constraint matrix is developed, each constraint is given a certain weight factor. Generally, the state that appears the most number of times in the constraint matrix is given some priority. A heuristic is used to assign a set of state codes that best satisfies the constraints. We used a technique similar to that of KISS where the vertices and faces of a Boolean hypercube are employed to encode states and groups of states respectively.

Tool 2

Herein, the term unconditional output refers to an output that depends only on the present state of the associated FSM, independent of the present input combination. The term conditional output refers to an output that depends on the present state and the present input combination.

A total of four rules are utilized:

Rule 1. same as that of tool 1,

Rule 2. same as that of tool 1,

Rule 3. states with common conditional outputs are targeted to be adjacent only if the outputs are conditional on the same input literal(s) as shown in Figure 3c, and

Rule 4. states with common unconditional outputs are targeted to be adjacent as shown in Figure 3d.

FIGURE 3 Examples of state assignment rules where state-a and state-b are given adjacent assignments.

The weight factors and state codes are assigned as in tool 1 except that rules 3 and 4 are given higher priority because tool 2 targets FSMs having many outputs.

Tool 3

One-hot encoding is used for an FSM of P states, where $N = P$. The one-hot assignment implies that the code for state i has a single "1" in the i th position and "0s" elsewhere.

Given an FSM of P states, tools 1 and 2 produce codes of length N where $\lceil \log_2 P \rceil \leq N \leq P$, and tool 3 produces a code of length P . In order to complete the study, we chose to use the tool MUSTANG which provides the option of producing a minimum length code. The results of the study are presented in the next section.

IV. RESULTS

The statistics of 13 practical FSMs are shown in Table I. The FSM samples were taken from various sources including: 1) the MCNC 1989 Logic Synthesis Workshop set of FSM benchmarks, 2) University projects, and 3) reference 10. The FSMs represent a wide range of samples which includes Moore machines (no OD) and Mealy machines. The "# outputs" column in Table I refers to the number of outputs from OD. In the case of a Moore machine, we specify the number of outputs to be zero.

The study involved a comparison between MUSTANG, tool 1, tool 2, and tool 3. The parameters of interest are: 1) number of state variables (Table II), 2) the number of transistors in NSD and OD (Table III), 3) the total silicon area after placement and routing (Table IV), 4) the worst-case prop-

TABLE II
Number of State Variables in State Code

EXAMPLE	mustang	tool1	tool2	tool3
mc	2	2	2	4
pop	3	3	3	5
opus	4	4	5	11
modulo12	4	4	4	12
ex4	4	4	5	14
log	5	5	5	17
tst4	5	5	5	17
fetch	5	5	5	26
nucpwr	5	5	5	29
rie	5	5	6	29
dvram	6	6	6	35
sync	6	6	9	52
ram_test	7	7	8	72

agation delay time (Table V), and 5) the test length (Table VI).

The silicon area results were obtained as follows. ESPRESSO was used for two-level minimization and DECAF was used for multi-level minimization. The VPNR (Vanilla Place aNd Route) [22] system was used for cell placement and routing. The area results are given in units of λ^2 where λ is the minimum feature length.

CRYSTAL [23] was used for finding the worst-case path. The results are given in units of nanoseconds and were obtained by using the circuit parameter values of the MOSIS SCMOS 3-micron p-well process.

The test results for tool 3 (one-hot) were obtained by applying the Branch Testing algorithm. In the other three cases, a scan path was employed to partition NSD and OD from M. Test patterns for NSD and OD were then generated using MIKE [24] and applied serially via the scan path. The scan path itself was tested by shifting in a sequence of alternating 1's and 0's. The fault simulation data were obtained from the CADAT [25] fault simulator. MIKE uses

TABLE I
Statistics of FSM Samples

EXAMPLE	# inputs	# outputs	# edges	# states
mc	3	5	10	4
pop	3	7	8	5
opus	5	6	22	11
modulo12	1	0	24	12
ex4	6	9	21	14
log	9	24	29	17
tst4	10	38	46	17
fetch	9	15	35	26
nucpwr	13	27	43	29
rie	9	26	49	29
dvram	8	15	47	35
sync	19	7	80	52
ram_test	16	24	117	72

TABLE III
Transistor Count

EXAMPLE	mustang	tool1	tool2	tool3
mc	56	56	78	127
pop	68	68	66	97
opus	302	274	264	347
modulo12	104	86	86	233
ex4	248	234	230	238
log	350	344	348	263
tst4	376	422	422	265
fetch	412	396	368	237
nucpwr	502	542	506	307
rie	522	568	616	459
dvram	608	592	582	397
sync	920	812	840	861
ram_test	1436	1484	1364	1235

TABLE IV
Silicon Area after Placement and Routing

EXAMPLE	mustang	tool1	tool2	tool3
mc	52,224	85,608	85,608	112,560
pop	73,920	80,976	73,920	81,792
opus	275,904	261,616	243,648	312,480
modulo12	109,440	88,000	88,000	221,232
ex4	222,376	232,256	241,424	256,896
log	370,240	361,080	366,792	350,192
tst4	428,800	479,520	479,520	417,312
fetch	410,592	426,496	401,128	398,400
nucpwr	568,800	633,440	644,800	505,400
rie	596,448	611,520	728,280	557,600
dvram	642,960	609,472	654,000	631,104
sync	1,159,584	1,235,430	986,168	1,170,432
ram_test	1,831,032	2,048,160	1,704,296	1,733,520

TABLE VI
Test Length for 100% fault coverage

EXAMPLE	mustang	tool1	tool2	tool3
mc	45	54	54	55
pop	71	75	71	62
opus	216	191	315	382
modulo12	116	106	106	151
ex4	196	186	225	275
log	273	285	249	311
tst4	291	303	309	263
fetch	321	285	321	381
nucpwr	357	351	411	433
rie	357	393	523	519
dvram	502	432	446	541
sync	642	621	1001	631
ram_test	1119	1103	1215	935

a combination of pseudo-random pattern generation and a deterministic PODEM-like approach.

We start our analysis by observing the number of state variables used in the state code for each of the FSMs in Table II. The number of state variables directly affects the size of M. The largest number of state variables was incurred by tool 3 and the smallest was incurred by MUSTANG. Table III shows that tool 3 incurs the least amount of combinational logic for most of the FSM samples. The data shown in Table IV represents the area occupied by the entire FSM. The area occupied by a single D flip-flop is $58 \lambda \times 64 \lambda$. It is clear from Table IV that one-hot encoding performs well for the larger FSM samples. It performs especially well for FSMs having many conditional outputs. However, it performs poorly for Moore machines. The reason is embedded in the fact that conditional outputs are directly available in the NSD portion of a one-hot encoded FSM. This allows the OD to share many common terms with the NSD.

In the case of propagation delay time, it is observed that one-hot encoding performs well for most of the FSM samples. This can be attributed to the fact that an increase in the number of state variables does not, to a first order, increase the delay time

because M is organized as a parallel-in-parallel-out shift register. In general, such an increase will result in less combinational logic which would explain the results shown in Table V.

The test data of Table VI also supports the case for one-hot encoding. The circuits resulting from MUSTANG, tool 1, and tool 2 utilize a scan-path to separate NSD and OD from M. Whereas, the circuits resulting from tool 3 utilize no scan-paths. Hence, we are really comparing a case where test pattern generation was performed for the combinational portion (separately from the flip-flop portion) with a case where test pattern generation was performed for the FSM regarded as a single entity with no separation between the combinational and flip-flop portions.

V. CONCLUSIONS

Table VII summarizes the experimental results by comparing the one-hot encoding data against the tool that performed best in each category (area, delay, and testability) for each of the FSM samples. Based on Table VII, we give one-hot encoding high marks for: 1) simplicity, 2) speed performance, 3) testability, and 4) area performance for Mealy machines that have a large number of outputs. We give one-hot encoding low marks for area performance for Moore-type and small FSMs.

The good performance of one-hot encoding relative to silicon areas is, in part, due to the degree of regularity of the resulting circuit structures. Regular structures lend themselves to more compact (denser) results. The speed performance results were due to shorter critical paths. Finally, the test results show that one-hot encoded FSMs do not require a scan-chain, but are inherently easily testable. In conclusion, for VLSI applications, the simple technique of one-hot encoding offers a competitive performance package for many FSM structures.

TABLE V
Worst-Case Propagation Delay Time

EXAMPLE	mustang	tool1	tool2	tool3
mc	210.70	115.96	192.03	133.11
pop	295.82	199.35	166.16	131.85
opus	480.02	484.37	271.90	300.62
modulo12	413.50	391.80	496.58	321.42
ex4	539.11	608.02	424.86	384.55
log	894.35	775.32	827.41	455.80
tst4	737.19	578.26	578.26	413.67
fetch	908.80	1260.26	836.32	424.61
nucpwr	1281.99	896.27	1160.92	527.61
rie	1137.55	1298.87	1440.67	756.54
dvram	1434.96	1135.35	1039.97	920.25
sync	1590.22	1931.32	1463.44	1360.71
ram_test	2018.22	1403.45	2309.33	1770.48

TABLE VII
Area, Delay, Test, and Transistor count Results in Comparison

EXAMPLE	Area		Critical path		Test length		# Transistors	
	best	one-hot	best	one-hot	best	one-hot	best	one-hot
mc	1	2.15	1	1.14	1	1.22	1	2.27
pop	1	1.11	1	0.79	1	0.83	1	1.43
opus	1	1.28	1	1.11	1	2.00	1	1.31
modulo12	1	2.51	1	0.82	1	1.42	1	2.71
ex4	1	1.15	1	0.90	1	1.47	1	1.03
log	1	0.96	1	0.58	1	1.24	1	0.76
tst4	1	0.97	1	0.72	1	0.93	1	0.68
fetch	1	0.99	1	0.50	1	1.33	1	0.64
nucpwr	1	0.88	1	0.58	1	1.23	1	0.61
rie	1	0.93	1	0.66	1	1.45	1	0.88
dvram	1	1.03	1	0.88	1	1.25	1	0.68
sync	1	1.18	1	0.92	1	1.01	1	1.03
ram_test	1	1.01	1	1.26	1	0.84	1	0.91

Acknowledgments

This work was funded, in part, by the Microelectronics Center of North Carolina.

References

- [1] J. Hartmanis, "On the State Assignment Problem for Sequential Machines I," *IRE Trans. on Elect. Comp.*, pp. 157-165, June 1961.
- [2] G. DeMicheli, R.K. Brayton, and A. Sangiovanni-Vicentelli, "Optimal State Assignment for Finite State Machines," *IEEE Trans. Computer-Aided Design*, pp. 269-285, July 1985.
- [3] S. Devadas, H-K T. Ma, A.R. Newton, and A. Sangiovanni-Vicentelli, "Synthesis and Optimization Procedures for Fully and Easily Testable Sequential Machines," *Proc. Int. Test Conf.*, Sept. 1988, pp. 621-630.
- [4] S. Su, R.Z. Makki, S. Chen, and S. Bou-Ghazale, "Automatic Synthesis of VLSI Control Paths from High Level Specifications," *Proc. IEEE Southeastcon*, April 1989, pp. 1009-1013.
- [5] S. Devadas, H-K T. Ma, A.R. Newton, and A. Sangiovanni-Vicentelli, "MUSTANG: State Assignment of Finite State Machines for Optimal Multi Level Logic Implementations," *Proc. Int. Conf. on Computer-Aided Design*, November 1987, pp. 16-19.
- [6] T. Villa and A. Sangiovanni-Vicentelli, "NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementations," *Proc. 26th Design Automation Conference*, June 1989, pp. 327-332.
- [7] X. Du, G. Hachtel, and P. Moceyunas, "MUSE: A Multilevel Symbolic Encoding Algorithm for State Assignment," *IEEE Trans. on CAD/ICAS*, Vol. 10, pp. 28-37, January 1991.
- [8] B. Lin and A.R. Newton, "Synthesis of Multiple Level Logic From Symbolic High Level Description Languages," *Proc. IFIP Int. Conf. on VLSI*, August 1991, pp. 187-196.
- [9] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vicentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Press, Boston, 1984.
- [10] R. Brayton, R. Rudell, A. Wang, and A. Sangiovanni-Vicentelli, MIS: A IEEE Trans. Computer-Aided Design, pp. 1062-1081, Nov. 1987.
- [11] R. Lisanke, G. Kedem, and F. Brglez, "DECAF: Decomposition and Factoring for Multi-Level Logic Synthesis," *MCNC Technical Report TR87-15*.
- [12] F.J. Hill and G.R. Peterson, *Digital Systems Hardware Organization and Design*, John Wiley & Sons, 1987.
- [13] J.P. Roth, *Fundamentals of Logic Design*, West Publishing Co., St. Paul, MN, 1979.
- [14] T.W. Williams and K.P. Parker, "Design for Testability—A Survey," *IEEE Trans. on Comp.* Vol. C-31, No. 1, pp. 2-15, Jan. 1982.
- [15] R.Z. Makki and C. Tianshang, "Designing Testable Control Paths with Multiple and Feedback Scan Paths," *Proc. Int. Test Conf.*, Sept. 1986, pp. 484-492.
- [16] E.B. Eichelberger and T.W. Williams, "A Logic Design Structure for LSI Testability," *J. Design Automat. Fault-Tolerant Comput.*, Vol. 2, pp. 462-468, May 1978.
- [17] J.P. Roth, "Diagnosis of Automat Failures: A Calculus and a Method," *IBM J. R&D*, Vol. 10, pp. 278-291, July 1966.
- [18] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Comp.*, pp.215-222, March 1981.
- [19] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. on Comp.* Vol. C-30, pp. 1137-1144, Dec. 1983.
- [20] R.Z. Makki, S. Bou-Ghazale, and C. Tianshang, "Automatic Test Pattern Generation with Branch Testing," *IEEE Trans. on Comp.*, Vol. 40, pp. 785-791, June 1991.
- [21] L.A. Glasser and D.W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, 1985.
- [22] P.R. Suaris and G. Kedem, "Quadrisection: A New Approach to Standard Cell Layout," *Proc. of Int. Conf. on Computer-Aided Design*, November 1987, pp.474-477.
- [23] W.S. Scott, G. Hamachi, J. Ousterhout, and R.N. Mayo, "1985 VLSI Tools: More Works by the Original Artists," Report No. UCB/CSD 85/225, University of California at Berkeley.
- [24] J. Calhoun, D. Bryan, and F. Brglez, "Automatic Test Pattern Generation for Scan-Based Digital Logic, MCNC Technical Report TR87-17.
- [25] HHB Systems, CADAT Version 5, User's Manual, 1986.

Biography

RAFIC MAKKI received his B.E and M.S. degrees in Electrical Engineering from Youngstown State University in 1979 and 1980 respectively. He received his Ph.D. degree in Electrical Engineering from Tennessee Technological University in 1983. He currently holds the position of associate professor and director of graduate programs in Electrical Engineering at the University of

North Carolina at Charlotte. His technical interests include logic synthesis, design for testability, power supply current testing, and fault modeling.

SHYANG-TAI SU received a B.S. degree in Electronics Engineering from Tamkang University, Taiwan, R.O.C., in 1982, a

M.S. degree in Electrical Engineering from the University of North Carolina at Charlotte in 1989, and a Ph.D. in Electrical Engineering from North Carolina State University in 1993. He is currently a member of the technical staff at Hewlett Packard. His research interests include design for testability, fault modeling, supply current testing, and VLSI CAD design.