finite state machines," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 779-781, July 1990.
[6] D. Drusinsky, "A simple single-block implementation and efficient state-assignments for statecharts," presented at SASHIMI-90 Workshop on Synthesis, Japan, Oct. 1990.
[7] D. Drusinsky and D. Harel, "Using statecharts for hardware description and synthesis," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 798-807, July 1989; also, registered U.S Patent 4 799 141.
[8] D. Drusinsky and D. Harel, "On the power of cooperative concurrency," in *Proc. Concurrency '88*, 1988, pp. 74-103.
[9] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Prog.*, vol. 8, pp. 231-274, 1987.
[10] D. Harel and A. Pnueli, "On the development of reactive systems," in *Logics and Models of Concurrent Systems*, K. R. Apt, Ed. (Nato ASI Series). Berlin: Springer-Verlag, 1985, p. 477-498.
[11] D. Harel, A. Pnueli, J. P. Schmidt, and R. Sherman, "On the formal semantics of state charts," in *Proc. 2nd IEEE Symp. Logic in Computer Science* (Ithaca, NY), 1987, pp. 54-64.
[12] C. Huizig, R. Gerth, and W. P. deRoever, "A compositional semantics for statecharts," Tech. Rep., Eindhoven University of Technology, The Netherlands, 1987.
[13] C. Huizig, R. Gerth, and W. P. deRoever, "Modeling statecharts behavior in a fully abstract way," in *Proc. Colloq. Trees in Algebra and Programming*, 1988, pp. 271-294.
[14] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, pp. 541-580, Apr. 1989.
[15] J. D. Ullman, *Computational Aspects of VLSI.* Rockville, MD: Computer Science Press, 1984.

Decision Problems for Interacting Finite State Machines

Doron Drusinsky-Yoresh

***Abstract*—Given a system of n interacting finite state machines (FSM's) and a state configuration, the reachability problem is to examine whether this configuration is reachable within the system. We investigate the complexity of this decision problem and three of its derivatives, namely 1) verifying system determinism, 2) testing for the existence of unspecified inputs to any FSM within the system, and 3) testing for the exclusiveness of two intra-FSM signals. We prove that these problems are all PSPACE-complete. We show the effect of these problems on the state assignment process for concurrent systems of interacting FSM's.**

I. Introduction

Logic synthesis of sequential finite state machines (FSM's) is a well-developed field of knowledge. There is a massive body of research in this area, originating in the fundamental research of Stearns and Hartmanis [7], [8]. See, for example, [1] and [6].

In this paper we examine logic synthesis of concurrent FSM's. It is our belief that concurrent FSM's will be increasingly used in the future, as exemplified by the following three scenarios:

- Consider a real-time control system; here, the real-time constraints impose hardware concurrency of several, perhaps communicating, sequential controllers where, typically, each controller is modeled as an FSM.
- Consider a design task which is divided between design groups, where each group designs a designated subsystem. When several such subsystems are individually controlled by a finite state mechanism, the whole system is conceptually controlled by a system of communicating concurrent FSM's.
- Finally, consider the process of silicon compilation of a behavioral hardware description language (HDL). Conventionally, such a compilation output is an FSM for a control mechanism that generates the sequencing instructions for the controlled data path. Naturally, with the advent of higher level HDL's (e.g., VHDL), and for increasingly sophisticated designs, the controlled data path might be inherently concurrent and hierarchical, requiring a network of concurrent FSM's to control it efficiently. Also, it seems quite natural to expect a concurrent behavioral description to be implemented by many controlling FSM's, perhaps one for each sequential process in the high level specification.

Hence, it is important to examine existing, predominantly sequential logic synthesis methodologies in the concurrent realm. To date, this has not been thoroughly done.

In this paper we reexamine the well-known state assignment methodology in this context. In Section II, we examine three implicit assumptions made by conventional state assignment tools and review them in the light of concurrency. We describe appropriate decision problems and analyze their complexity in Section III.

II. Motivation: State Assignments for Sequential and Interacting FSM's

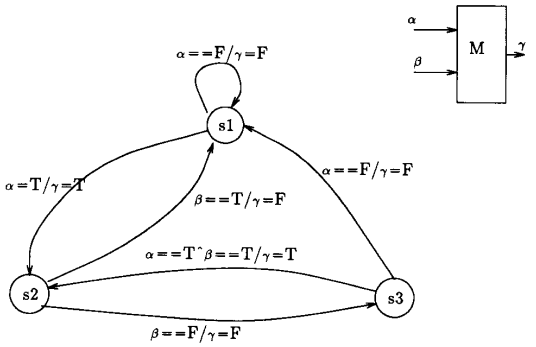
Consider the FSM of Fig. 1(a) and its PLA implementation of Fig. 1(b). This PLA was generated by NOVA [13], a well-known state assignment program, and embodies three typical assumptions made by such a program:

1) Determinism (DET): the FSM is assumed to be deterministic; i.e., for every state and every input there is at most one next state. This assumption enables the state assignment program to implement an n-state FSM with as few as $\log n$ state variables.
2) Unspecified input (UT): if at some state there is an input configuration that causes no next state (it triggers no transition; e.g., $\langle \alpha, \beta \rangle \equiv \langle$T, F$\rangle$ in state s3 of Fig. 1(a)), then it is assumed to be an "impossible" input for this state. In other words, it is assumed to be the designer's responsibility to verify that such an incident does not occur. This enables the state assignment program to exploit the free space for optimization; hence, the PLA of Fig. 1(b) generates s2 as the next state in the above case. Note that in the conventional FSM implementation scheme, where a state register is connected to a combinational logic block, the combinational logic always produces a (perhaps erroneous) next state.
3) Input and output orthogonality (IOO): the FSM is assumed to have orthogonal (independent) inputs and outputs, i.e., all combinations of signal values over the IO wires, except those found to be impossible earlier, are possible. Hence, in Fig. 1(a) all four input configurations of $\langle \alpha, \beta \rangle$: $\langle$T, T$\rangle$, $\langle$T, F$\rangle$, $\langle$F, T$\rangle$, and $\langle$F, F$\rangle$, are possible, except for $\langle \alpha, \beta \rangle \equiv \langle$F, T$\rangle$ in state s3. This assumption is self-evident in the single-FSM case, where the IO signals are connected to an unex-

Manuscript received October 5, 1989; revised April 5, 1990, and August 7, 1990. This paper was recommended by Associate Editor R. K. Brayton.
A preliminary version of this paper was presented at the IFIP Workshop on Applied Formal Methods for Correct VLSI Design (1989) and the Synthesis and Simulation Meeting and International Interchange (1989).
The author was with the LSI-Logic Development Department, Sony Corporation, Atsugi-shi, Kanagawa-ken 243, Japan. He is now with SSL, Sony, 611-B River Oaks Parkway, San Jose, CA 94087.
IEEE Log Number 9100303.

0278-0070/91$01.00 © 1991 IEEE

Authorized licensed use limited to: Naval Postgraduate School. Downloaded on October 14, 2008 at 13:56 from IEEE Xplore. Restrictions apply.



(a)

α	β	P.S.	N.S.	γ
-	0	1-	01	0
1	-	0-	11	1

(b)

Fig. 1. (a) An FSM M. (b) PLA and state assignment for M. Note that the PLA term representing a transition to state 00 and asserting output 0 is not required. State assignment: s1 = 00, s2 = 11, s3 = 01.

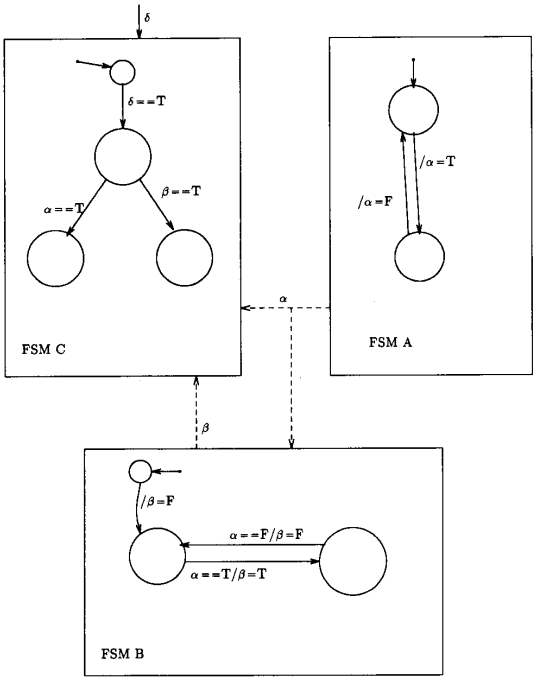


Fig. 2. Note that a transition without a trigger is always enabled. Here solid arrows represent state transitions, whereas dashed arrows represent signal flow.

pected environment. However, when one wants to economize on the communication area consumed by a network of communicating FSM's, it is tempting to find out which communication signals are timewise exclusive.

In the single-FSM case, the nontrivial assumptions are DET and UI. Verifying DET, namely, verifying that an FSM is deterministic, is easy [9]. UI, on the other hand, is not as easy. In fact, UI is NP-complete in the number of input wires to the FSM, because it is actually the problem of testing whether there is an input assignment such that all transitions that are outgoing from a given state are disabled. This condition can be formulated as finding a truth assignment for a disjunctive normal form (DNF) formula such that the formula evaluates to *false* or as the famous CNF satisfiability problem [9]. Nevertheless, heuristics are expected to solve NP-complete problems to a certain degree.

Now, consider the network of Fig. 2. Here, three synchronous FSM's communicate by sending and receiving two binary symbols, α and β. Semantics for such a network are available in the next section. As in the single-FSM case, a typical state assignment program, when applied to each FSM separately, tries to minimize the size of the implementation and assumes this input FSM to be deterministic. FSM C, however, is not deterministic in its own right. Viewed as an independent FSM, it must certainly consider the possibility that sometime α and β might be received simultaneously, which induces nondeterministic behavior. NOVA simply aborts; a different state assignment program might choose a 1-hot implementation or some other implementation that enables nondeterminism [12]. In any event, the basic flaw here is that the system of Fig. 2 will never produce $\alpha == T$ and $\beta == T$ simultaneously, thus ensuring a deterministic behavior of C, and of the whole system for that matter. Note that when each FSM is deterministic the whole system is also deterministic, but not visa versa. For example, in Fig. 2 FSM C is not deterministic when considered separately, but the whole network is.

To understand UI in a concurrent environment, consider a variant of the network of Fig. 2 where FSM C is substituted by FSM M of Fig. 1(a) (ignore M's output γ). Once again, using the same straightforward synthesis approach, the implementation of FSM M is given by the PLA of Fig. 1(b). This approach, however, because it assumes UI, assumes that the input $\langle \alpha, \beta \rangle \equiv \langle T, F \rangle$ cannot be received at state s3. This however, is incorrect; such an input can indeed be received, and when it is received the next state will be s2 without any real semantic reason. In fact, the "impossible" configuration in this example is $\langle T, T \rangle$ because $\alpha == T$ and $\beta == T$ can never be received simultaneously.

Finally, IOO has a very practical motivation, exemplified by Fig. 2. Can we save communication area by time multiplexing some of the inter-FSM communication signals, such as α and β? In the case of Fig. 2, the answer is affirmative, because configurations $\langle \alpha, \beta \rangle == \langle T, T \rangle$ and $\langle \alpha, \beta \rangle == \langle F, F \rangle$ are never transmitted. Also, because it is always the case that either $\alpha \equiv T$ or $\beta \equiv T$, one binary signal suffices.

Hence it is clear that for a system of interacting FSM's, these three implicit assumptions often need better verification. For example, a designer might indeed try to design such a system so that DET holds, i.e., so that the system is deterministic. But, as expected for a complex design, he would like to verify that indeed his design is deterministic, i.e., he needs to verify DET. Similarly, a designer might try to time multiplex two exclusive intra-FSM signals on one electrical line. He too is expected to verify his design in this respect, i.e., he needs to verify IOO. In the following section, we define the appropriate decision problems, and prove them all to be PSPACE-complete.

We believe these results have the following effect on logic synthesis. Given a system of n communicating FSM's, a naive synthesis approach is to separately synthesize each FSM, and then combine the resulting implementations into a physical network. Our

Authorized licensed use limited to: Naval Postgraduate School. Downloaded on October 14, 2008 at 13:56 from IEEE Xplore. Restrictions apply.

results show that, given existing tools, which typically assume one of the three assumptions discussed earlier, such an approach is not necessarily correct.

III. Related Theory[1]

We use the conventional notation. An FSM is a 6-tuple $M = (y_0, X, Y, Z, \delta, \lambda)$, where X, Y, and Z are finite sets of primary inputs, states, and primary outputs, respectively; $\delta: X \times Y \rightarrow Y$ is the next state function; and $\lambda: X \times Y \rightarrow Z$ is the output function. X and Z typically consist of multiple "channels"; thus $X = X_1 \times \cdots \times X_n$ and $Z = Z_1 \times \cdots \times Z_m$, where Z_i and X_i are sets of input and output symbols, respectively. Hence FSM inputs and outputs are tuples of symbols. We consider a network of r interacting FSM's, $M^1, \cdots, M^r$. FSM's communicate by exchanging inputs and outputs with other FSM's without any restriction. Hence a symbol α can be sent on channel i of FSM M^k (i.e., $\alpha \in Z_i^k$) and received on channel j of FSM M^l (i.e., $\alpha \in X_j^l$). Our model is completely synchronous, so α will be a valid input for M^l in the cycle that follows the transmission. Given a sequence of external input symbols (i.e., symbols that originate in the environment), each FSM carries out a *computation* (*run*) in a conventional way, considering its inputs as the external inputs together with the inputs received from other FSM's. A newtork configuration $\{q_1, \cdots, q_r\}$, $q_i \in Y^i$, is called a state configuration.

Given a network and an initial state configuration, the reachability problem is to define whether the configuration is reachable from the initial state configuration. We shall first investigate the complexity of this general problem, and then show how DET, UI, and IOO are derived.

Proposition 1: The reachability problem is PSPACE-complete in r, the number of FSM's.

Proof: First we show that the problem is solvable in PSPACE. We construct a nondeterministic polynomial space algorithm for the problem and then use a theorem by Savitch [11] that proves such an algorithm to be in PSPACE too. The nondeterministic algorithm iteratively guesses next states in all FSM's and verifies that these next states indeed compose a legal next-state configuration. It does so until the desired next-state configuration is reached. Verifying that a configuration is a legal next-state configuration is done by recording the outputs generated by every FSM M^i, guessing an external input, and verifying that all FSM's can indeed advance to their desired next state upon the reception of their composite inputs. This process is also done nondeterministically, and consequently in PSPACE. This entire process is carried out in polynomial space.

We prove that the problem is PSPACE-complete using a reduction from the *finite automata intersection problem* (INT) [10]. Let $F_1, \cdots, F_r$ be r deterministic finite state automata (i.e., acceptors) with a common input alphabet Σ such that L_i is the set accepted by F_i. The problem INT is to determine whether the automata accept a common element of Σ^*. We reduce INT to the reachability problem as follows. First we modify the automata F_i so that they produce an output 1 within every accepting state, and 0 otherwise. We denote the new FSM's as M^i. Clearly a word x is accepted iff every FSM M^i produces an output sequence that ends with a 1. Now we add communication to the system; each FSM sends its outputs (0 or 1) to all other FSM's. Also, we add a special state S to each FSM, and add transitions to the transition functions so that M^i moves to its S state (from any previous state) iff it receives 1's from all FSM's simultaneously. Clearly the state configuration s that consists of the special states of all FSM's is reachable iff there is a word x that belongs to all L_i in the original INT problem. Q.E.D.

Given a network of communicating FSM's as defined above, the determinism problem (DET) is to find whether the system has a sequence of external inputs for which the system run is not unique. The following proposition and proof show that this problem is very similar in essence to the reachability problem.

Proposition 2: DET is PSPACE-complete in r, the number of FSM's.

Sketch of Proof: Solving DET in PSPACE is done with a nondeterministic algorithm that guesses a nondeterministic configuration, generally following the footsteps of the algorithm of Proposition 1.

The reduction is very similar to that used in Proposition 1. Given an instance of the INT, we modify the automata F_i to be FSM's M^i, as in Proposition 1. Also, we add two contradicting transitions from the special state configuration, namely, we add two different next-state configurations over the same external input. Clearly there is a sequence of external inputs for which the system has two different runs (those that reach the special state configuration and then split) iff there is a word x that belongs to all L_i in the original INT problem. Q.E.D.

A naive algorithm for DET would be to test each FSM individually for nondeterminism and to conclude that the system is deterministic iff all FSM's are. Such a *local test* is only partially correct. Indeed, if all FSM's are deterministic then so is the system, but it might be the case that some FSM's seem to be nondeterministic while the system is deterministic. Consider Fig. 2. FSM C is nondeterministic whereas the system as a whole is deterministic because α and β are never received simultaneously.

Given a network of communicating FSM's as defined above, the unspecified reception (UI) problem is to find whether there exists an (external) input sequence for which the run is not precisely defined; namely, somewhere within the run the system will not have a next-state configuration. As discussed in Section II, for a single FSM with k input channels this problem is NP-complete in k. However, when the system consists of more than a single FSM, the problem is more difficult.

Proposition 3: UI is PSPACE-complete in r, the number of FSM's.

Sketch of Proof: Solving UI in PSPACE is done with a nondeterministic algorithm that guesses a configuration with unspecified transitions and then verifies this property.

A reduction from the reachability problem or from INT follows the techniques presented earlier, where a special configuration s is created such that s has no next-state configuration. As before, there is a reachable configuration with unspecified receptions iff INT is positively solved. Q.E.D.

Given a network of communicating FSM's as defined above, an FSM within the network, and two inputs to this FSM, the I/O code assignment (IOO) problem is to check whether the system has a sequence of external inputs such that the two inputs here are received simultaneously by the FSM during its run on the input sequence. Consider Fig. 2; where α and β are exclusive, which is in fact the reason for the network being deterministic. This is true although they are transmitted on different physical lines. In fact FSM C needs only one input channel for α and β, where $\alpha \equiv \neg\beta$.

Proposition 4: IOO is PSPACE-complete in r.

Sketch of Proof: Once again we modify INT so that two states,

[1]Full proofs appear in [2] and [5].

Authorized licensed use limited to: Naval Postgraduate School. Downloaded on October 14, 2008 at 13:56 from IEEE Xplore. Restrictions apply.

q^1 and q^2, $q^i \in Y^i$, within the special state configuration s, will produce two newly defined outputs, α and β, which will be consumed by a special, new FSM, M^s. Clearly, these inputs are received simultaneously by M^s iff INT is positively solved. Q.E.D.

ACKNOWLEDGMENT

The author wishes to thank T. Inoue, T. Nakamura, Y. Furui, and T. Sato for their assistance in this research and the participants of the SASHIMI-89 and IFIP workshops for their helpful comments.

REFERENCES

[1] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal state assignment for finite stte machines," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 269–285, July 1985.
[2] D. Drusinsky, "On synchronized statecharts," Ph.D. thesis, Weizmann Institute of Science, 1989.
[3] D. Drusinsky, "State assignments for extremely concurrent finite state machines," in *Proc. IFIP Workshop on Applied Formal Methods for Correct VLSI Design*, 198-205, Nov. 1989.
[4] D. Drusinsky, "State assignments for extremely concurrent finite state machines," presented at Synthesis and Simulation Meeting and International Interchange (SASIMI), Osaka, Japan, May 1989.
[5] D. Drusinsky and D. Harel, "On the power of cooperative concurrency," in *Proc. Concurrency '88*, 1988, pp. 74–103.
[6] T. A. Dolotta and E. J. McCluskey, "The encoding of internal states of sequential machines," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 549–562, 1984.
[7] J. Hartmanis, "On the state assignment problem for sequential machines 1," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 157–165, 1961.
[8] J. Hartmanis and R. E. Stearns, "On the state assignment problem for sequential machines 2," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 593–603, 1961.
[9] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computations.* Reading, MA: Addison-Wesley, 1979.
[10] D. Kozen, "Lower bounds on natural proof systems," in *Dig. 18th IEEE Symp. Foundations of Computer Science*, Oct. 1977, pp. 254–266.
[11] W. J. Savitch, "Relationship between nondeterministic and deterministic tape complexities," *J. Comput. Syst. Sci.*, vol. 4, pp. 177–192, 1970.
[12] T. D. Ullman, *Computational Aspects of VLSI*, Rockville, MD: Computer Science Press, 1984.
[13] T. Villa and A. Sangiovanni-Vincentelli, "NOVA: State assignment of finite state machines for optimal two-logic implementation," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 905–924, Sept. 1990.

A Generalized Scharfetter–Gummel Method to Eliminate Crosswind Effects

Yie He and Guoxiang Cao

***Abstract*—A generalized Scharfetter–Gummel discretization method is proposed for semiconductor device modeling. Motivated by the classical Scharfetter–Gummel approach and the SUPG (streamline upwind/Petrov–Galerkin) method, an optimal nonlinear artificial diffusion term is added to the standard Galerkin finite element formula within a weighted residual form to construct the generalized Scharfetter–Gummel method, which exhibits better properties in precluding numerical oscillation and crosswind effects than the classical Scharfetter–Gummel method and the SUPG method. The generalized Scharfetter–Gummel method is more general and is applicable to complicated problems.**

Manuscript received July 11, 1990; revised October 8, 1990 and December 14, 1990. This work was supported by the Chinese Education Committee. This paper was recommended by Associate Editor J. G. Fossum.

The authors are with the Microelectronics Center, Southeast University, Nanjing 210018, People's Republic of China.

IEEE Log Number 9102367.

I. INTRODUCTION

It is often necessary to solve the convection–diffusion equation in such fields as aerodynamics, fluid dynamics, and microelectronics. If the convection term dominates the diffusion term, numerical oscillation can result when the classical finite element method or the central finite difference method is used. To control the nonphysical oscillation in the solution, Scharfetter and Gummel proposed a discretization method for the one-dimensional current continuity equation in 1969 (usually referred to as the S–G method) [1]. This method was later extended to two- and three-dimensional problems (e.g., MINIMOS [2]). However, the S–G discretization method has a loss of accuracy in the streamline-normal direction which can be avoided without sacrificing stability. Usually, the loss of accuracy or numerical dissipation that appears in the streamline-normal direction is referred to as a crosswind effect. By adding a linear artificial diffusivity to the Galerkin formula in a weighted residual form, Sharma applied the SUPG (streamline upwind/Petrov–Galerkin) method to semiconductor device simulation to eliminate both numerical oscillation and crosswind effects [3]. Unfortunately, the linear artificial diffusivity is not optimal in most cases and is not applicable to complex current continuity equations other than the drift-diffusion model, e.g., the current continuity equation containing a magnetic field. In this paper, by adding a nonlinear artificial diffusion tensor instead of a linear artificial diffusivity in the streamline direction which satisfies the weighted residual formulation, we derive a generalized S–G method to achieve good stability and accuracy properties simultaneously.

II. DISCRETIZATION METHODS OF CURRENT CONTINUITY EQUATION

A. A Generalized Expression of the S–G Method

The electron current continuity equation for semiconductor devices can be written as

$$\nabla \cdot \vec{J}_n = qR \tag{1}$$

where R is the net recombination rate and the electron current density, $\vec{J}_n$, can be described by a drift-diffusion model:

$$\vec{J}_n = q\mu_n n\vec{E} + qD_n\nabla n. \tag{2}$$

Here μ_n is the electron mobility, D_n is the electron diffusion coefficient, $\vec{E}$ is the electric field, and q is the elementary charge. Introducing a function Q and setting

$$Q\nabla \cdot (\mu_n n\vec{E}) - \nabla Q \cdot D_n\nabla n = 0 \tag{3}$$

we can transform the current continuity equation (1) into the following form:

$$\nabla \cdot (QD_n\nabla n) = RQ. \tag{4}$$

There is no convection term in (4). Because of this, we can use the classical Galerkin finite element method in the discretization of (4) without producing any numerical oscillation. Assuming that the whole domain of the solution is divided into many elements and that μ_n, D_n, and $\vec{E}$ are constant in any element, from (3) we can

0278-0070/91$01.00 © 1991 IEEE

Authorized licensed use limited to: Naval Postgraduate School. Downloaded on October 14, 2008 at 13:56 from IEEE Xplore. Restrictions apply.