# CONSTRAINTS-BASED SPECIFICATION AND SYNTHESIS OF FUNCTIONAL ANALOG MODULES

Jacek Maitan
Lockheed Missiles & Space Company, Inc.
Research & Development Division
3251 Hanover Street
Palo Alto, CA 94304-1187

April 1989

Unlimited Distribution

Prepared for
Dr. Andre van Tilborg
Contract No. N00014-87-C-0513

Monitoring Organization
Office of Naval Research

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Lockheed Missiles & Space Company, Inc.<br>*Research & Development Division*<br>6b. OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION<br>Office of Naval Research |
| 6c. ADDRESS (City, State, and Zip Code)<br>3251 Hanover Street<br>Palo Alto, CA 94304-1187 | 7b. ADDRESS (City, State, and Zip Code)<br>800 North Quincy Street<br>Arlington, Virginia 22217 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>Contract N00014-87-C-0513 |
| 8c. ADDRESS (City, State, and Zip Code) | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. |

11. TITLE (Include Security Classification)
Constraints-Based Specification and Synthesis of Functional Analog Modules

12. PERSONAL AUTHOR(S)
Jacek Maitan

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Final | FROM 7/1/87 TO 9/30/88 | 1989, April | 28 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | design specifications, design synthesis, nonlinear analog circuits, distributed control, synchronization, scheduling, dining philosophers problem, neural networks (more) |
| | | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper describes a new methodology for synthesizing analog circuits based on declarative behavioral specifications. Behavioral specifications are translated into a convex energy function. Analog circuitry is used for the realtime search for a solution to system constraints. The uniqueness of the design procedure is due to the explicit synthesis of the design from its high-level specifications.

| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | Unclassified |

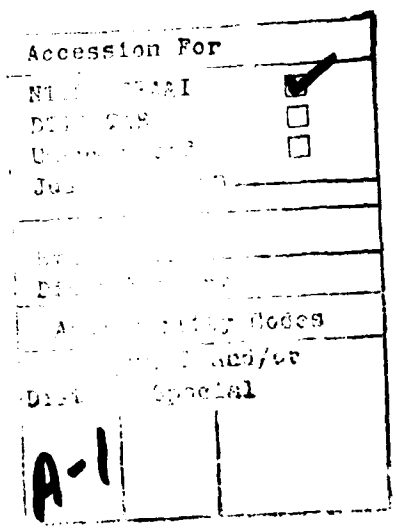
| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE (Include Area Code) | 22c. OFFICE SYMBOL |
|---|---|---|
| Raghu Raghavan | (415) 424-2114 | |

DD FORM 1473, 86 JUN

X90057_DD1473

## ACKNOWLEDGMENTS

The author thanks I. Forman for discussions on RADDLE; C. Tomlinson, W. Wilner, and all other colleagues from Microelectronic and Computer Technology Corporation (MCC) for the opportunity to participate in the KLEIN project and for their permission to include concepts of KLEIN in this paper; and Tom Huynh for comments and discussions. This research was supported by Office of Naval Research contract N00014-87-C-0513.




# CONTENTS

# ILLUSTRATIONS

# Section 1

## INTRODUCTION

Recently, there has been an upsurge in research on advanced languages that can be effectively executed in high-performance, concurrent architectures (1,2,3). The challenges of such research are to identify high-performance architectures and to design a language to write applications that run on these architectures. In order to be used, a language must have an effective implementation on the chosen architecture. A major problem with these concurrent languages is the lack of effective implementations.

This paper explores a link between the language used to code an application and its parallel operational semantics. Since the effectiveness of the execution environment is directly related to the match between application and the computational structure onto which the application is being mapped, we have explored the consequences of using a constraints-based description of communication and control of concurrent processes to synthesize computational structure. As a result, one may implement an analog circuitry that is functionally equivalent to a functional module to control concurrent processes. The general nature of the technique makes it possible to apply it to other types of functional modules.

Three languages influenced our research: RADDLE (4), UNITY (2), and KLEIN (3). They can be functionally characterized as supporting a concurrent programming methodology. In all three, the design process is a sequence of refinement steps, starting with a formal specification and ending with a program to be executed on a concurrent machine.

Synchronization and communication constraints are discussed in section 2. Constraints-based specifications of functional modules are expressed in terms of meet and choice primitives in the spirit of KLEIN constructs. Section 3 outlines the process of synthesis of functional modules using nonlinear elements (neurons). Structural and behavioral synthesis is uniformly expressed as an energy-function synthesis problem. The solution for the constraints by the minimum of the energy function is demonstrated graphically, in addition to two analytical methods. The realtime search for a minimum can

be aided by specialized hardware (5) designed using analog components. Preliminary experiments show rapid convergence of the search process. Problems and advantages of the approach are described in section 4. Section 5 is a summary, which includes brief descriptions of possible extensions. Section 5 also outlines applications of the technology for truly distributed control of computer networks.

## Section 2

## FUNCTIONAL MODULES—SPECIFICATIONS

We are interested in supporting a concurrent programming methodology based on a sequence of correctness-preserving transformations. In this methodology, a concurrent processing application is implemented starting from high-level specifications. The direct mapping from high-level specifications into analog-based functional modules reduces the number of levels of abstractions (6) into one equivalent level and results in faster execution.

This section describes constraint-based specifications of synchronization and resource-allocation problems. These constraints can later be used explicitly to synthesize the concurrency management hardware. The KLEIN semantics that inspire our work is found in (3).

A limited number of concepts from KLEIN are used to illustrate issues involved in distributed processing. The resulting mini-KLEIN is expressed in terms of operational semantics describing, for a logical machine (7), rules of legal state transitions. Thus, we are concerned about states and transition rules. In general, one can construe an application (system) as an ensemble of concurrent processes, each with its own state, progressing in time by making transitions from one state to another. A process can make a transition if it satisfies some conditions. KLEIN assumes a nondeterministic transition mechanism. It agrees with the philosophy of using the weakest set of assumptions about implementation mechanisms. Inspired by (8), we will focus on the rules of communication among multiple processing elements.

The description below is limited to simplified communication and process interaction semantics.

Assume that a concurrent system is built out of n processes and that each can participate in a set of synchronous transfers of data among themselves. A synchronous transfer is referred to as an interaction. (In the language of KLEIN, a synchronous transfer is known

as an "event." The term "interaction" is used here to stress the communication focus of our analysis.)

In an interaction, each sender must have at least one receiver. An interaction is mediated by a channel, identified by a unique label.

Each interaction involves input/output (I/O) requests and solving resource-allocation problems associated with the I/O requests. In this work, I/O requests are given names. A name is an identifier or a label bound to a unique resource such as a communication channel or an I/O device. An I/O request is satisfied when all the names associated with the send operations match those associated with receive operations. (The use of names enables us, in addition to tracing activities within an interaction, to build an internal data structure, a tableau, to compile control hardware.)

## 2.1 DISTRIBUTED CONTROL PRIMITIVES—MEETS AND CHOICES

This section contains a detailed description of the communication constraints in terms of the system's components described above. The focus is placed on the dynamics of the resource-allocation process. An example of cooperating processes bidding for the access to limited resources is provided to illustrate the problem.

A process is bidding for resources. A bid is represented as a Meet (M). Each M consists of two parts, Send (S) and Receive (R)

$$M(label) = \{S(label), R(label)\} = \{\{names\}, \{names\}\}$$

where S(label) = { names } and (label) = { names }.

Thus, an interaction may be encoded as an ordered set of names. The first set contains names associated with Send operations, and the second set contains names associated with Receive operations. Label is a unique identifier attached to M, S, and R. It indicates a logical link between these elements.

Two functions, send_names and receive_names, extract names associated with the S or R parts of an M. They are defined as:

send_names(M(label)) = S(label) = {names}

receive_names(M(label)) = R(label) = {names}

A more complex structure, Choice (C), is a disjoint set of bids and contains a multiplicity of meets. (In this paper, we do not discuss a guarded meet, which is used to select a meet predicating on the condition of the guard.) At any given moment, several concurrently active Meets and Choices may be waiting to be resolved within a system. Upon resolution, new meets and choices are generated until no further computation is required. As a result, each process can participate in several configurations of communication bidding. However, once bids are resolved and resources are allocated, a process can participate in only one interaction. Thus, in order to directly map these declarative specifications into effective implementation, one needs an effective constraints solver.

Mutually exclusive sets are denoted by {[set1]...[set i]}.

Example 1. An interaction is described as M(alfa) = {{a},{[b,c],[c,d]}}. In this configuration, M(alfa) sends a and may receive either [b,c] or [c,d]. For example, if [c,d] is selected, then [b,c] is inhibited.

A Meet is a Choice with single Meet, i.e., C = {M} = M = [ channels ].

Let a command A(label) be denoted as Alabel. A Choice then can be defined as

Ci = {M1, M2, ... Mn}

Functions send_names and receive_names are still valid:

send_names(C) = (send_names(M1) ... send_names(Mn)) = (U send_names(Mi))

receive_names(C) = (receive_names(M1) .. receive_names(Mn))
= (U receive_names(Mi))

where U is a sum of sets operator.

A system (SYS) is a list of all currently active bids and may be described as

SYS(label) = { C, M }

where C is a set of all Ci and M is a set of all Ms not associated with any Ci.

Implied in SYS(label) is a set of all currently active names c_names, a set of all names associated with C or M, or more formally:

c_names = (U send_names(C) receive_names(C) send_names(M) receive_names(M)) = all_names(SYS)

where all_names(an_operation) is a function returning a single set of all names associated with an_operation. (SYS is an operation describing system transition from an old state to the new state, i.e., sequence of SYS(current) → SYS(new)).

A similarly constructed N-party formalism is presented by Ira Forman (4) in RADDLE.

## 2.2 DYNAMICS OF DISTRIBUTED CONTROL

We now illustrate the dynamics of interaction by solving Dijkstra's dining philosophers problem.

Example 2. There are n spaghetti-eating philosophers who share n forks. The philosophers are seated around a table. Two forks must be used simultaneously to eat spaghetti; therefore, any two philosophers must share a fork placed between them. A philosopher can do one of the following two activities: (1) eating, then he uses both forks he has access to, and (2) thinking, when he has no need for forks.

The problem occurs if the act of picking up both forks by a thinking philosopher is not simultaneous. In the extreme case, if each philosopher takes a fork from the left, there will be no second fork available to his right. Since there is no fork-access protocol, and as a result a philosopher is not allowed to negotiate with his neighbors, no one can eat. This is referred to as a deadlock.

We show here the solution to the dining philosopher problem by providing a way to synthesize a functional module, a controller, and to manage the access to forks and avoid deadlocks. In the modified configuration that includes philosophers, forks, and the

controller, each philosopher bids for his forks using this controller as an arbitration agent. The controller decides who thinks and who eats.

A possible extension of the dining philosophers problem is to allow a philosopher to die from starvation if he waits too long for the requested forks. This is a proper analogy with a realtime system in which tasks must often be performed within a predefined period of time; otherwise, the waiting process is terminated. This is known as the fair-scheduling problem. If a demand for a shared resource is known then, one can apply structural partitioning to equalize the utilization pattern (balance the load).

During the execution of a distributed program, the controller dynamically binds senders with receivers in some optimal way. In the simplest case, the controller attempts to avoid deadlocks. In more complex, realtime situations, in order to avoid time-outs, some form of fair scheduling or load balancing must be included. The latter is especially important in a case of realistic control of high-speed computer networks.

A system SYS(label) can be represented by a dynamically reconfigurable graph G(label). For example, nodes represent processes, and channels and labels on links are associated with currently active send or receive requests.

Example 3. A system consists of two philosophers competing for two forks. It can be described by the following structure:

SYS(phil) = { Cforks, {Ma, Mb} } = { {$n_1$, {[a-$f_1$, a-$f_2$], [b-$f_1$, b-$f_2$]} }, { {{a-$f_1$, a-$f_2$}, *nil*}, { {b-$f_1$, b-$f_2$}, *nil*}} }

Philosopher a associated with Ma sends to forks f1 and f2 associated with Cforks. Similarly, philosopher b requests the same two forks. Graph G(phil) is depicted in Figure 1.

G(.) is, in a general case, a composition of disjoint subgraphs.

graph(SYS(label)) = G(label)

G(.) = G0(.) U G1(.) U ... U Gn(.)

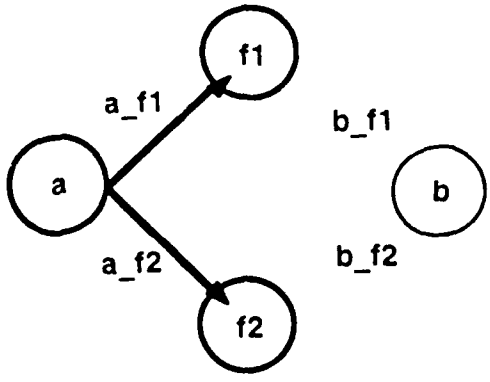



Figure 1. Simple graph describing two philosophers, a and b, sharing forks f1 and f2.

Since each process may participate only in a single active/resolved bid, the following must also be true:

for all i and j, j <>i, Gi(.) ∩ Gj(.) = O.

The last condition is simply the requirement that Gi(.) and Gj(.) do not overlap.

Each Gi(.) represents a fully satisfied set of bids. Since each graph Gk(.) can be decomposed into sending and receiving parts, both send_names and receive_names functions can also be extended to graphs:

*send_names(G) = {names}*

receive_names(G) = {names}

## 2.3 CONSTRAINT SOLVER (STATIC CASE SPECIFICATION)

Problem: For a given SYS, find its communication graph G such that: (1) send_names(G) = receive_names(G), and (2) | send_names(G)) | = max.

It is not required that during each interaction all channels be used or all bids be satisfied. Thus, the following condition

send_names(G) + receive_names(G) = c_names

is too strong, as is illustrated in the example below.

Example 3 (continued). There are two basic solutions to a dining philosophers problem with two philosophers and two forks:

$$= \{ \{nil, \{[a\text{-}f_1, a\text{-}f_2], [b\text{-}f_1, b\text{-}f_2]\}\} \quad \{\{a\text{-}f_1, a\text{-}f_2\} nil\} \}$$

$$= \{ \{nil, \{[a\text{-}f_1, a\text{-}f_2], [b\text{-}f_1, b\text{-}f_2]\}\} \quad \{\{b\text{-}f_1, b\text{-}f_2\} nil\} \}$$

graphs G(phil) consists of two disjoint graphs corresponding to two possible solutions with either philosopher a or philosopher b eating. All other solutions, in which a single fork is granted to each philosopher, violate the condition of satisfying simultaneously a request for both forks by a or b. It is observed that

(1) send_names(G(phil)) = receive_names(G(phil))

(2)| send_names(G(phil)) | = | receive_names(G(phil)) | = max = 2

In this configuration, represented by a single-node graph, at least one philosopher must always think.

## Section 3

## FUNCTIONAL MODULE SYNTHESIS

This section describes a general procedure to synthesize behavior encapsulated within functional modules. An objective of this research is to provide a technique to mechanize the procedure.

Recent reports on heuristic optimization techniques (5, 9) described a class of relaxation methods implemented using sigmoid functions. (tanh(x) is an example of a sigmoid function.) Components of functional modules with such characteristics are referred to as "neurons." Without going into a discussion of the degree of similarity between neurons of the brain and neurons of computer or neural scientists, we will use the term to identify components with the static characteristic in the form of a linear section bounded by saturation regions, as illustrated in Figure 2. Accidentally, this characteristic is similar to the characteristic of an operating amplifier, and this property will be used to synthesize VLSI implementation.

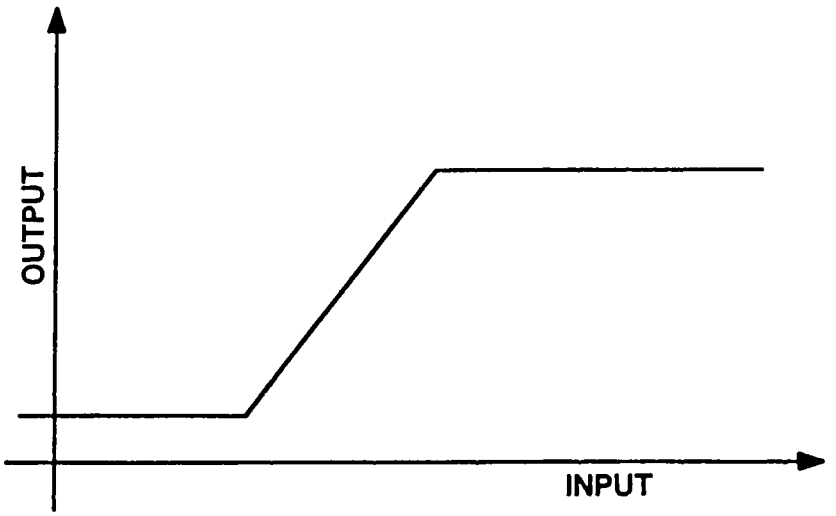



Figure 2. Nonlinear processing element (neuron).

In each SYS, we associate a set of neurons with each interaction and denote sets of neurons associated with M and C by NM and NC.

Neuron(M) = NM

Neuron(C) = NC

Since the number of neurons, NM, is equal to the number of "free" meets M (meets not associated with any choice), the following relation is also true:

| NM | = | M |

In general, a set of neurons associated with a choice C is equal to the number of alternative meets contained in it.

The desired solution to the distributed control problem is to find a correct activation pattern for both NC and NM. For the dining philosophers problem, an activation pattern is simply a pattern of 0s and 1s, where 0 deactivates a component and 1 activates it. An augmented vector of neurons NC and NM is referred to as control neurons (Nctrl).

Nctrl = [NM | NC]

Both the dimension of a vector and the vector will be denoted by the same symbol. For example, Nctrl serves both as a denotation of a vector and a denotation of its dimension, | Nctrl |.

### 3.1 FUNCTIONAL MODULE—STRUCTURE

Now we discuss how to connect neurons in order to obtain a desired behavior described using declarative communication constraints.

Interconnection constraints are captured by incidence matrices Ms for resources associated with send operations and Mr for resources associated with receive operations. For a system with Nctrl neurons and c_names resources, the matrices Ms and Mr are of dimension Nctrl x c_names.

A resource used by send or receive in each interaction is marked as "1"; otherwise, it is marked as "0" in the Ms and Mr matrices. Since Nctrl states of neurons denote active or

inactive control signals, multiplication of an incidence matrix by a neuron vector gives us a resource utilization pattern.

Ms * Nctrl = send_names(G)

Mr * Nctrl = receive_names(G)

Solving the distributed control problem is equivalent to solving the following relation, referred to as the Satisfaction Invariant (SI):

(SI) Ms * (Nctrl) == Mr * (Nctrl)

where the "==" operator satisfies the following conditions:

C1 For each C, there is at most a single active neuron representing an M.

C2 If send is part of the solution, it has at least one receive, and if receive is a part of the solution, it has only one send.

C3 Total number of active neurons at the fixed point must be maximal.

C4 A neuron at the fixed point takes the value of 0 or 1.

The fixed point of communication constraints can be found by applying a relaxation process to a set of communication constraints as shown in Figure 3.

Figure 3 helps to define the structure of the problem. To find a solution, one can connect neurons on the diagonal using MS and MR matrices as switchboxes, i.e., a solution line can change its direction only in the nonzero matrix entries. Any combination of inteconnected neurons satisfying C1-C4 is a solution.

Figure 4 illustrates the solution procedure for example 3. It shows a tableau for Example 1. The continuous lines are associated with a solution allowing philosopher a to eat, while the broken lines are associated with a solution allowing philosopher b to eat. The simplicity of the dining philosopher problem results in the sparsity of the incidence matrices.

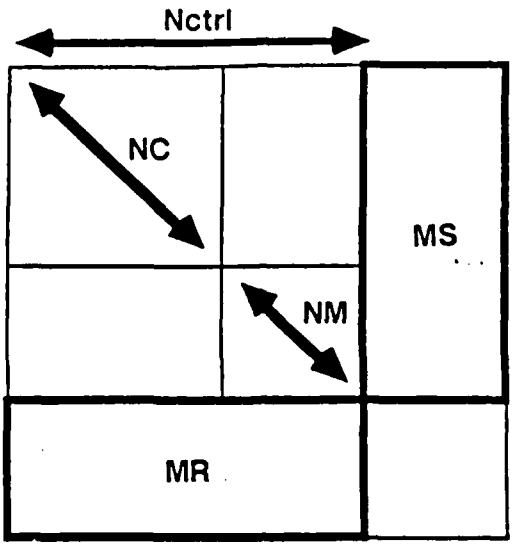



Figure 3. Distributed control table.

### 3.2 FUNCTIONAL MODULE—ENERGY FUNCTION FORMULATION

In the previous section, we discussed a graphical construct for solving communication constraints under C1 to C4. Let us now discuss two other approaches to solve communication constraints. Both are based on finding the minima of an energy function. They were implemented and used to simulate a controller for a dining philosophers problem.

Methods 1 and 2 described below are based on defining an energy function. The solution of this energy function is a desired controller decision. Both methods are based on explicit encoding of C1 to C4.

In the description below, a neuron N[i] is active if N[i] = 1; otherwise, it is inactive.

Method 1

C1 If in a choice there are n meets out of which only a single meet is requested, then the energy component due to C1 can be formulated as

$$E1 = \Sigma \Sigma \, NM[i] \, NM[j], \; (i=1..n, \; j>i)$$

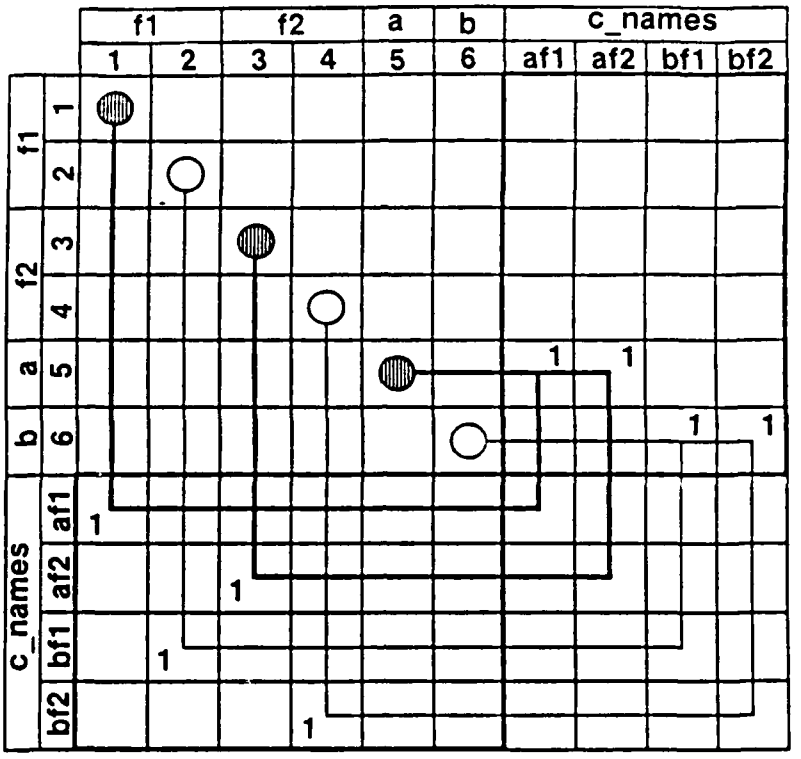



Figure 4. Distributed control table for example 2.

E1 is nonzero when two or more meets, each represented by a single neuron, are activated; E1 is zero if only one meet is selected.

C2 When a meet with a send request is selected, a matching receive request can be found using incidence matrix MR. The energy function component E2 due to C2 is:

$$E2 = \Sigma\, NM[i](NM[i] - (1/n)(\Sigma\, NM[k]))^2$$

$$i = 1..n\,, \quad k \in \{\, MR^{*}Nctrl[i]\}$$

E2 is equal to zero if for each active NM[i], a neuron associated with a send request, there are n NM of neurons associated with corresponding receive requests. (n > 1 is in the case of broadcasting.)

C3 The sum of all active neurons must be maximal; the E3 component of energy function has form

$$E3 = ((\Sigma\ Nctrl[i])-k)^2,\ i = 1\ ..\ n$$

E3 is equal to zero if there are k active neurons; k is equal to the maximal number of possibly active neurons.

C4 A neuron takes the value of 0 or 1; the corresponding E4 component has the form:

$$E4 = \Sigma\ Nctrl[i](1\text{-}Nctrl[i]),\ \text{for all } i$$

E4 is equal to zero if and only if NM[i] is either 0 or 1.

The total energy E is E = E1 + E2 + E3 + E4 is equal to zero if conditions C1 to C4 are satisfied. In the case of the dining philosophers problem, if the control value is equal to 1, then a philosopher is granted the use of forks; if 0, he must continue thinking.

Method 2

Method 2 differs from method 1, only in that a sigmoid function to normalize the value of the sum of active neurons is used in conditions C2 and C3.

C2 If a meet with a send request is selected, then a matching receive request can be found using incidence matrix MR; the following is a formula for an energy function component E2 due to C2:

$$E2' = \Sigma\ NM[i](NM[i]-s(\Sigma\ NM[k]))^2,$$

$$i \in \{\ MS^{*}Nctrl\ \},\ k \in \{\ MR^{*}Nctrl[i]|\}$$

E2 is equal to zero if for each active NMi, a neuron associated with an active send request, there are some NM[k] neurons associated with corresponding receive requests. To assure that all NM[k] are active when used for E2, one must add an additional constraint. Thus, total E2 has the form

$$E2 = E2' + \Sigma\Sigma\ NM[i]\ (1-NM[j])\ (i,\ j>i)$$

C3 The sum of all active neurons must be maximal; the E3 component of energy function has form

$$E3 = (s(\Sigma\ NM[i])-1)^2$$

the higher the number of active neurons, the closer E3 is to zero.

The major difference between methods 1 and 2 rests on the amount of knowledge required to satisfy C2 and C3. Since they use the sigmoid to normalize the number of receivers, both C2 and C3 in method 2 do not depend on exact knowledge of how many neurons participate in the interaction as receivers. On the other hand, when using method 2, one must include an additional constraint about the necessity of all receivers being in an active state.

The energy function is positive and convex; thus, it has a minimum. Elementary calculus will identify the minima at points with coordinates satisfying the constraints of the dining philosophers problem.

### 3.3 FUNCTIONAL MODULE—ANALYSIS AND IMPLEMENTATION

The model based on energy formulation was simulated and the results are now discussed.

A typical simulation run is illustrated in Figure 5. The behavior of each neuron is displayed in a separate panel labeled by its id. The last panel represents the energy level. Notice that energy is monotonically decreasing, and that in all panels, the signals stabilize shortly after the relaxation process begins. In both methods, the energy is used to trigger the control. In the currently implemented control model, if the system's energy falls below a certain threshold level, the current relaxation states are labeled as stable and they are used as control signals.

The uniqueness of the above approach lies in its simplicity. We have demonstrated that it is possible to map distributed control constraints using satisfaction invariant into an effective computational algorithm to control distributed processes. Since the constraints are formulated using a declarative programming language, this approach can be used to build realtime systems using traditional components in a fashion similar to that illustrated in (5).

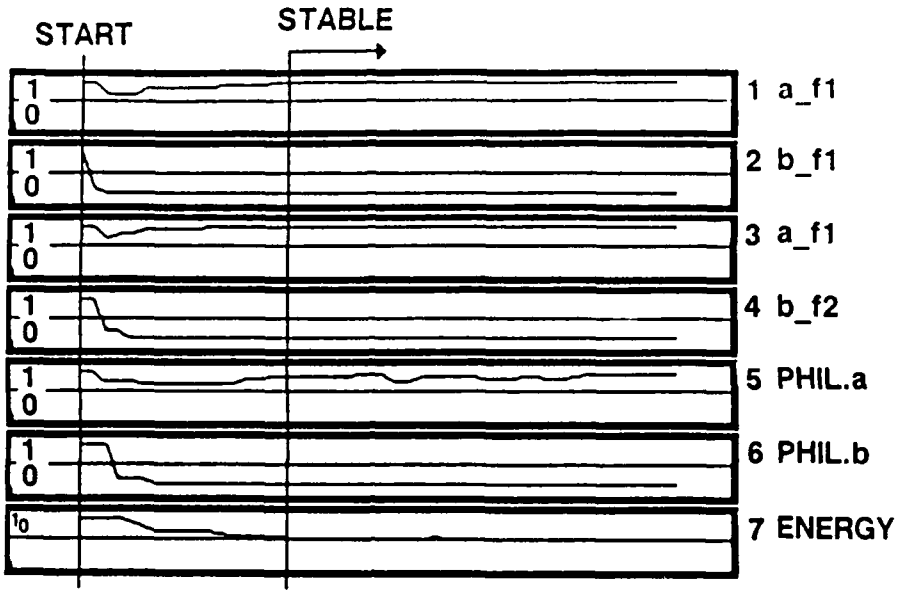



Figure 5. Dining philosopher problem–simulation.

In Hopfield's approach, which represents a large class of adaptive circuits, the sigmoid function is effectively implemented as an operational amplifier. In addition to the desired shape of static characteristics, an operational amplifier has the functionality of an adder.

The next section discusses the impact of recent progress in very large-scale integration (VLSI) technology in reducing device features and making analog technology increasingly attractive as an implementation vehicle. Approaches based on the use of optical components are also viable.

## Section 4

## FUNCTIONAL MODULE—VERIFICATION ISSUES

It is taken for granted that circuits become more complex and faster. Limiting factors due to verification and specification of these designs are usually overlooked. Problems with the design and synthesis of digital circuits are still far from being solved, and now may be the time to reevaluate basic assertions of the design process in order to minimize them.

There are basic differences in verifying digital circuits and analog circuits synthesized using the approach described in this paper. We will start to compare both by first evaluating the design methodology for digital circuits. The following description of problems associated with digital circuits is far from being complete, and the selection is constrained by personal experience; (10) contains a more complete review and bibliography.

A goal of the verification process is to ensure the correct behavior of a circuit. In all commercially used methodologies, correctness is defined with respect to some procedurally specified models of behavior. The verification technique is based on exhaustive simulation. This approach, commonly used in industry, will not be discussed in this paper.

Other, more experimental techniques, include the use of petri nets (11), theorem proving (12), attribute grammars (6), and temporal logic (13). With the notable exclusion of the temporal-logic approach based on "controlled exhaustive" simulation (14), all these techniques address the issues of correctness for large-scale circuits. These approaches were applied to digital circuits only.

The functional module synthesis approach proposed in this paper is based on the use of realtime evaluation of constraints. One can apply complex digital circuits to solve the problem, but analog technology, since it does not require accurate synchronization, is more natural for this type of problem. Improvements in the technology and sophisticated clocking schemes introduce problems in final timing analysis of digital circuits (15). The

use of self-clocked designs is an improvement, but such use has penalties: the increased area of the required silicon and the increased complexity of the circuit.

### 4.1 ADVANTAGES

Explicit analog implementation of a fixed-point evaluator makes the whole process not only potentially faster but also simpler. The lack of message-passing, a mechanism commonly used to synchronize processes (16), reduces the cost of firmware. Since specialized hardware can be synthesized from declaratively specified constraints, there is no top-down structural hierarchy to describe the levels of abstraction. In addition, subnanosecond digital devices mean shorter time constants for analog circuits.

The direct link between structure and behavior, which is achieved by constructing the Mr and Ms incidence matrices, is unique in its simplicity. The energy function is dynamically constructed. As a result, the correctness of a design depends on the correctness of its software specification. There is no need for timing verification at the transistor level and for all possible situations, both for data and control.

### 4.2 DISADVANTAGES

The lack of a technique to estimate the relaxation time for a large-scale system complicates the design process. In a general case, the convergence of the nonlinear dynamical systems is hard to predict but can be reduced to the stability analysis of such systems (17). If the analysis is not made correct, a problem can be easily translated into a set of unambiguous and incomplete specifications. The experience with various formulations indicates that the latter leads to systems with no fixed point in the required domain of admissive solutions.

Additional effort must be made in choosing parameters for the analog circuit. It has been observed that if parameters are not chosen well, the search process may lock itself into a local minimum or the relaxation process may be slow (9).

The way to avoid the above problems is to apply a formal verification technique which resembles classical calculus analysis of the existence of a solution. In the simplest case, for a given tableau, a full analytical formulation in terms of an energy function can be derived and simple analysis of its shape provides all necessary and sufficient conditions

for the existence of a solution. Further work on mechanizing the process and direct hardware compilation is currently under development by the author.

# Section 5

# SUMMARY

This paper has described a new methodology of synthesizing analog circuits based on declarative behavioral specifications. Behavioral specifications are translated into a convex energy function. The fixed points of such a system correspond to its minima. Analog circuitry based on a relaxation technique is used for the realtime solution of the system constraints. This solution is simply a minimum within a constrained area of admissible solutions. In the case of an example of communication management, the admissible solutions are values from {0,1}.

By using this novel technology, one can avoid all top-down decomposition of the solution. In the case of communication management, there is no need for message passing, protocols, I/O firmware, or interrupts.

Careful analysis of specifications is required to assure the existence of the solution to the desired set of constraints. If the energy function has certain properties globally or locally (continuity, monotonicity, or convexity), one can estimate the convergence of the whole process.

## 5.1 CURRENT RESEARCH

In parallel to the work in applying functional analog modules to computationally intensive problems, we investigated an application of this technique to management and realtime control of distributed computer networks. High-speed terrestrial networks for intercomputer communication require fast reconfiguration capability. The necessary reconfiguration and control speed can be achieved either by projecting existing algorithms into faster control circuits based on esoteric, high-speed digital techniques or by careful management of routing information. To support the latter, we propose to investigate the applicability of an array of electronic techniques to provide truly distributed network-control capabilities.

Networks based on fiber optics are characterized by a small ratio between packet transmission time to the time required to propagate a packet through the network. The

latter has two components: (1) transport delay, which is limited by the speed of the propagation of light in the fiber, and (2) the node-processing time required to route and resolve contentions at a node and, as a result (due to the distributed nature of the control scheme), in a whole fiber-optic network.

In networks based on current gigabits-per-second technology, a further increase in the speed of transceivers to decrease packet-transmission time will not increase the actual network throughput. One must reduce the packet-propagation time (routing and resolution of contentions) in order to improve network utilization. In addition, due to high data bandwidth, the traditional concept of management using store-and-forward techniques must be carefully reevaluated.

There are two types of information at each node: static information localizing a node in relation to other nodes, and dynamic information estimating the cost of routing to other nodes. Static information is true throughout the whole network, whereas dynamic routing information, due to the delay in its collection and distribution, can be applied only to link the nearest nodes. Therefore, we propose to use local routing maps, which are characterized by high accuracy of the cost estimates associated with access to local (nearest) nodes and only "rough" knowledge of both cost and routing information to remote nodes.

The maps are continuously exchanged between neighborhood nodes using some of the broad-bandwidth subchannels. With this continuous exchange, nodes that are close together can route data based on accurate knowledge of the local traffic; in the nodes that are far apart, the maps are used only for connectivity checks.

The use of local routing maps is in full compliance with the distributed nature of the control mechanism.

The relatively small size of the routing maps allows for direct VLSI implementation. A high-speed, asynchronous, associative-memory implementation of the proposed routing map can be evaluated as a result of this research.

Based on this shared local-routing information, nodes or channels bid for resource allocation by setting appropriate parameters in the cost functions. Minima of these functions computed by node solvers are the desired allocation of required resources. One or more subchannels, at separate nodes, are needed to build a path to connect nodes.

From a functional point of view, nodes or channels are bidding to connect to each other by firing requests. As a result of this bidding, a network, in this case a local resource allocation controller, responds with a minimum-energy configuration that encodes a channel allocation. Thus, a combination of minima solvers with locally maintained routing maps, used to set up cost-function parameters, gives a unique structure functionally equivalent to a data-driven, distributed associative memory.

We propose a relaxation-based solution to the problem of finding minima of a cost function. This class of solutions can also be implemented using high-speed VLSI circuits.

The allocation process can be illustrated using a simple case in which several input channels bid to transfer data through the same output channel. In this technique, instead of discrete requests being exchanged between channels, bids are continuously exchanged between nodes in the form of continuous data. Thus, the problem becomes similar to the classic resource-allocation problem known as the dining philosophers problem.

We want to study the applicability of the relaxation technique to a large class of synchronization and resource-allocation problems in network routing. If implemented using VLSI technology, a high-speed, small iterative solver with low power consumption can provide the necessary solution to the resource-allocation problem.

The use of functional modules to construct a distributed algorithm to control communication networks is listed here to illustrate the practical potential of the technique.

## 5.2 FUTURE RESEARCH

Problems with the growing complexity of designs using digital technologies, especially in the domain of realtime systems, can be reduced by using analog technology (18). As examples in this paper have demonstrated, the simplicity of the technology is especially visible in the case of implementing algorithms related to the semantics of concurrent processes.

In order to apply the technique to everyday design practice, one must test and verify techniques to synthesize hardware to implement both large-scale problems and other adaptable, multipurpose functional modules, e.g., image-processing algorithms, associative memories, and combinatorial optimization components. (A functional module

can be implemented using either digital or analog technologies.) In addition, the ideas presented in this paper show a manageable way of applying the growing field of neural network methodology to implement declaratively specified systems. Thus, this problem-specification-driven methodology may constitute an evolutionary path that effectively unifies both software and hardware approaches to system design.

## Section 6

## REFERENCES


1. Forman, I. R., Raddle: An Informal Introduction, STP-182-85, Microelectronics and Computer Technology Corporation, February 1986.

2. Chandy, K. M. and Misra, J., Parallel Program Design: A Foundation, Addison-Wesley Publishing Company, Inc., 1987.

3. C. Tomlinson, private communication—report on KLEIN to be published in 1989.

4. Forman, I. R., On the Design of Large Distributed Systems, STP-098-86, Microelectronics and Computer Technology Corporation, 1987.

5. Hopfield, J. J. and Tank, D. W., "'Neural' Computation of Decisions in Optimization Problems." Biological Cybernetics, Vol. 52, pp. 141-152, 1985.

6. Maitan, J., "Management of Constraints in VLSI Data Bases." Applied Artificial Intelligence: An International Journal, 1988.

7. Plotkin, G. D., A Structural Approach to Operational Semantics, Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

8. Milne, G. J., "CIRCAL and the Representation of Communication, Concurrency, and Time." ACM Transactions on Programming Languages and Systems, Vol. 7, No. 2, pp. 220-298, 1985.

9. Fox, G. and Furmanski, W., Load Balancing, Technical Report C3P 363, California Institute of Technology, 1987.

10. Camurati, P. and Prinetto, P., Formal Verification of Hardware Correctness: An Introduction." Proceedings of the IFIP WG 10.2 Eight International Conference on Hardware Description Languages and their Applications, Amsterdam, 1987.