135-141

78756348

STATUS: PENDING 20110613 **OCLC #:** 79077197

REQUEST DATE: 20110610 **NEED BEFORE:** 20110710 **SOURCE:** ILLiad

BORROWER: AZU **RECEIVE DATE:** **DUE DATE:**

RENEWAL REQ: **NEW DUE DATE:** **SPCL MES:**

LENDERS: CUY, CUY, *ZAP, BRI, BRI

TITLE: Problem solving by simulation : IMACS European Simulation Meeting, Esztergom, Hungary, 28-30 August 1990 /
ISBN: 9789633725108
IMPRINT: Budapest : Scientific Society of Measurement and Automation 1990.
DISSERTATION: [1] Proceedings -- [2] Posters.
ARTICLE: Rozenblit, J.W.: Simulation Modeling in Design Generation and Solution
ISSUE DATE: August 1990
PAGES: 135-141
VERIFIED: <TN:952031><ODYSSEY:150.135.238.6/ILL> OCLC
SHIP TO: ILL/UNIVERSITY ARIZONA LIBRARIES/1510 E UNIVERSITY/TUCSON AZ 85721-0055

BILL TO: same//FEIN 74-2652689//BLLD acct#51-105
SHIP VIA: Ariel, Odyssey or Library Mail
MAXCOST: IFM - $50
COPYRIGHT COMPLIANCE: CCL
ODYSSEY: 150.135.238.6/ILL
FAX: (520) 621-4619 //ODYSSEY PREFERRED/IL... ARIEL 150.135.238.50
EMAIL: askddt@u.library.arizona.edu
AFFILIATION: AZNET ; GIF ; GWLA ; SHRS
BORROWING NOTES: For Book Chapter requests, scan the chapter only, do not lend book. If over your
page limit, please CONDITIONAL request. Thank you.

PATRON: Valenzuela, Michael L

try B4 371 760

DTD/ Ariel
NRLF Access Service:
Date 6/14/11
Initials TC
PS A
Pages 9

AZU.14A

UNIVERSITY OF CALIFORNIA

Northern Regional Library Facility

ELECTRONIC DELIVERY COVER SHEET

WARNING CONCERNING COPYRIGHT RESTRICTIONS

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted materials.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use," that user may be liable for copyright infringement.

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

PROBLEM SOLVING BY SIMULATION

IMACS EUROPEAN SIMULATION MEETING

Esztergom, Hungary
28-30 August 1990

PROCEEDINGS

Edited by A. JÁVOR

III. THEORY

Sensitivity Analysis of Simulation Experiments: Regression Analysis and Statistical Design
J.P.C. Kleijnen 111

Estimates of Reliability by Analytical and Simulation Methods
V.V. Kalashnikov 119

Conceptual Modelling in Discrete Event Simulation Using Diagrammatic Representations
R.J. Paul, V. Ceric 127

Simulation Modelling in Design Generation and Solution
J.W. Rozenblit 135

Partial Functions in Simulation: Formal Models and Calculi
V.M. Antimirov, D.E. Naidich, V.N. Koval 143

A New Optimization Method for Combined Simulation
I. Molnár, S. Hardhienata 149

Criteria for Comparison of Software Systems for Continuous Simulation
P. Cerny 157

A Semantic Open-Ended Approach to Special Simulation Systems
M. Frank, S. Strohmeyer 165

A Study of Deductive Types of Automatic Reasoning by Means of Simulation
F. Capkovic 171

IV. TRAFFIC AND CONTROL

Solving Highway Safety Problems Using Simulation Model NARD
S. Basu 181

Train Driving Simulator
Zhang Dazhang, Wang Qianhua, Wen Li 189

Computer Simulation Study on Performances of Permanent Magnet Levitation Vehicle with Mechanical Air-Gap Controller
M. Abe 195

Control Design Using DYNAMO
M.S. Mahmoud, S.M. El-Said 203

The Computer Simulation of Automatic Control System Including Power Electronic Circuits
Wang Zhaoan, Zhuo Fang, Li Min 213

SIMULATION MODELLING IN DESIGN GENERATION AND SOLUTION

Jerzy W. Rozenblit

Dept. of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona 85721
U.S.A

ABSTRACT

The paper discusses a simulation modelling-based system design methodology. The methodology, termed Knowledge Based Simulation Design (KBSD), employs simulation and artificial intelligence concepts to support design model development and performance evaluation. A formal representation scheme called *system entity structure* is employed to structure the model of a system to be designed. The generation of design solutions consist in a goal driven, rule-based selection and synthesis of model static structures. More specifically, a model is synthesized from components identified through the system entity structure and stored in the model base. Performance of design models is evaluated through computer simulation. The best model (with respect to trade-off criteria over the set of performance measures) constitutes a design solution.

1. INTRODUCTION

Knowledge-based frameworks consider design as a technological activity in which knowledge about a specific domain is used to represent design artifacts, constraints, and requirements. It is an activity that seeks all relevant knowledge and combines it to produce a design solution. Design is often considered as a search process in which a satisfactory design solution is produced from a number of alternatives [1,2]. The search proceeds in a design space whose elements are design objects (components) and attributes (parameters).

Design frameworks differ mainly in the underlying knowledge representation scheme and the search methods employed for solution generation. However, all the methodologies attempt to capture and enumerate alternative solutions in a design domain. Design knowledge should be organized in such ways that it can be manipulated effectively and efficiently. The system design approach proposed by Rozenblit [3,4], termed Knowledge-Based Simulation Design, focuses on the use of modelling and simulation techniques to build and evaluate models of the system being designed. It treats the design process as a series activities that include the following phases: specification of design levels in a hierarchical manner (decomposition), classification of system components into different variants (specialization), selection of components from specializations and decompositions, development of design models, experimentation and evaluation via simulation, and choice of design solutions.

In the ensuing sections, we shall define the design problem and demonstrate how the above phases of the methodology can effectively support generating design solutions.

2. DESIGN PROBLEM FORMULATION

As we have pointed out in the introduction, the design process is often considered as a search problem. In our approach, we generate a target design model (a goal state) which best satisfies design constraints and requirements. Thus, the problem can be formulated as follows:

Given a set of design objectives, constraints, and requirements OCR, find a design model DM^* such that: $DM^* = best\{DM_i, i = 1, ..., n\}$, where each DM_i is a design model that satisfies the OCR, i.e., $OCR(DM_i)$, (a set of objectives, constraints, and requirements satisfied by design model DM_i) is a subset of OCR, and *best* is a function ranking and selecting the design alternatives DM_i.

The Knowledge–Based Simulation Design methodology provides a set of methods for generating a solution to the above problem. The solution process consists is constructing a set of alternative design models $DM_i s$, simulating their behavior, and selecting the model DM^*.

3. DESIGN SOLUTION GENERATION

The design model construction process begins with developing a representation of design components and their variants. To appropriately represent the family of design configurations, we have proposed a representation scheme called the *system entity structure* (SES) [4,5]. The scheme captures the following three relationships: decomposition, taxonomy, and coupling. Decomposition knowledge means that the structure has schemes for representing the manner in which an object is decomposed into components. Taxonomic knowledge is a representation for the kinds of variants that are possible for an object, i.e., how it can be categorized and subclassified. The synthesis (coupling) constraints impose a manner in which components identified in decompositions can be connected together. The selection constraints limit choices of variants of objects determined by the taxonomic relations.

Beyond this, procedural knowledge is available in the form of production rules. They can be used to manipulate the elements in the design domain by appropriately selecting and synthesizing the domain's components. This selection and synthesis process is called *pruning* [3,4]. Pruning results in a recommendation for a *model composition tree*, i.e., the set of hierarchically arranged entities corresponding to model components. A composition tree is generated from the system entity structure by selecting a unique entity for specializations and a unique aspect for an entity with several decompositions.

The final step in the framework is the evaluation of alternative designs. This is accomplished by simulation of models derived from the composition trees. Discrete Event System Specification (DEVS) [5] is used as a modeling formalism used for system specification in the methodology. DEVS provides a formal representation of discrete event systems. It is closed under coupling. This property facilitates the construction of hierarchical DEVS network specifications.

Performance of design models is evaluated through computer simulation in the DEVS-Scheme environment [6]. DEVS-Scheme is an object–oriented simulation shell for modeling and design that facilitates construction of families of models specified in the DEVS formalism. Alternative design models are evaluated with respect to experimental frames that reflect design performance questions. Results are compared and traded off in the presence of conflicting criteria. This results in a ranking of models and supports choices of alternatives best satisfying the set of design objectives.

3.1 Design Model Structure Representation

The interaction of decomposition, coupling and taxonomic relations in an SES affords a compact specification of a family of models for a given domain. In a system entity structure, *entities* refer to conceptual components of reality for which models may reside in a model base. Also associated with entities are slots for attribute knowledge representation. An entity may have several *aspects*, each denoting a decomposition, and therefore having several entities. An entity may also have several *specializations*, each representing a classification of possible variants of the entity.

Figure 1 illustrates a high level system entity structure representation a robot. A robot can be classified into a mobile or a fixed type through the Motion Specialization. The Subsystem Decomposition defines major components of the robot, namely, the cognition, control, mechanical and communication subsystems. The latter can be further classified with respect to the link type. Some of the entities have variable type attached to them (denoted by a the "–" prefix), which characterize their properties.

The first subproblem in generating a design solution is to find a set of design model structures

(composition trees) that conform to static design constraints and requirements. By *static*, we mean the constraints and requirements whose satisfaction can be accomplished prior to simulation of a design model's behavior.

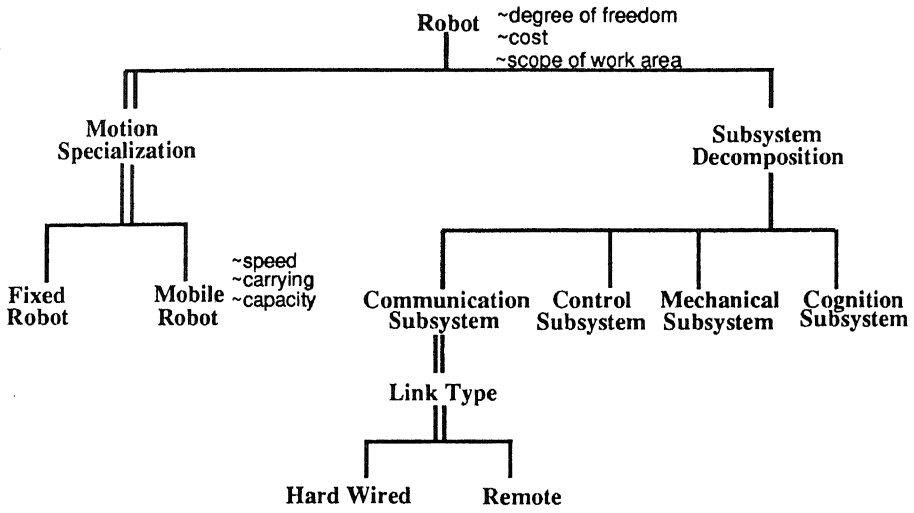

Fig. 1 System Entity Structure Example

Let SCR denote a set of static design constraints and requirements. Let $Gen(DES) = \{CT_1, CT_2, ..., CT_n\}$ denote all composition trees generated by the system entity structure DES. Then, the design structure generation subproblem is defined as follows:

Find a set: $CT_D = \{CT_1, CT_2, ..., CT_k\}$ such that $SCR(CT_i) \subseteq SCR$ for i = 1,2,..., k, i.e., CT_D is a set of composition trees that satisfy the static design constraints and requirements.

Rozenblit [3] defined a procedure for generating the $Gen(DES)$ and CT_D sets. The procedure, called generic attribute driven pruning consists in: 1.) defining a set of design performance attributes (e.g., throughput, average queue length, component utilization), and 2.) traversing the system entity structure and selecting entities from specializations. This process is computationally very expensive for large system entity structures; it can only be solved by a non-deterministic polynomial time algorithm [3,4]. Therefore, instead of the combinatorial enumeration of the composition trees spawned the system entity structure and searching for admissible trees, we apply the production rule framework to generate only the trees that satisfy the static constraints.

3.2 Rule-Based System Entity Structure Pruning

Rule based pruning requires that a knowledge base that contains rules for selection and configuration of the entities represented by the system entity structure be specified. Production rules are used to represent design objectives, constraints, user's requirements and performance expectations. The pruning process can be interpreted as a search directed by constraints through the search space consisting of the entities, their aspects and specializations.

The following steps are required to provide the rules that guide pruning of the system entity structure: 1) for each specialization, specify a set of rules for selecting an entity; 2) for an entity with several aspects, specify rules for selecting a unique aspect; 3) for each aspect specify synthesis rules that ensure that the entities selected from specializations are configurable, i.e., the components they

represent can be validly coupled. Each rule can be assigned a certainty factor indicating the rule's degree of applicability. Recall Figure 1 with the Robot system entity structure. Below, we give an illustrative selection and a synthesis rule for this entity structure. A full scale Robot synthesis example is given in [7].

Example of a selection rule:

If budget is relatively high and
    working area is usually greater than 25 square feet or
    requirement of arm carrying capacity is not heavy ($\leq$ 1000 lbs)
The recommended–motion–type is mobile (1.0)

Example of a synthesis rule:

If recommended robot is mobile and
    type of motion is mobile–free–flier and
Then robot system is configurable (1.0) and
    motion subsystem is free flier (1.0) and
    remote communication is required (0.9) and
    engine motion mechanism is strongly recommended (1.0)

An expert system shell MODSYN (MODel SYNthesizer) to generate model structures was developed and implemented [8]. MODSYN's inference engine uses the backward chaining strategy to recommend a composition tree. The goal of pruning is given by a configuration defined through the top level synthesis rule(s). The engine attempts to verify the hypothetical configuration by checking lower level selection and synthesis rules and/or asking the designer for facts not available in the data base. By providing answers, the designer instantiates attributes of the design constraints and thereby expresses design preferences and expectations. For details, concerning the inferencing mechanism, we refer the reader to [8].

Rule–based pruning is an effective means of generating recommendations for composition trees that satisfy static design constraints. It generates the set CT_D that constitutes the solution to the design subproblem defined in Section 3.1. Figure 2a depicts two composition trees derived for the entity structure of Figure 1. After the model composition trees are generated, we proceed to define model specifications. In our modelling environment (DEVS-Scheme), we can automatically generate a model ready to simulate.

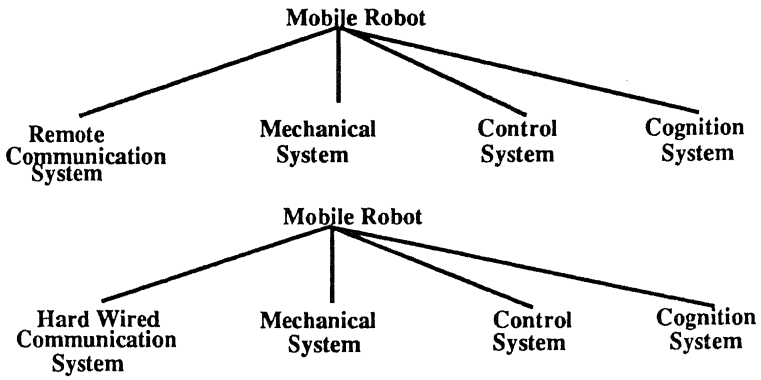

Fig. 2a Composition Trees for Robot Design

3.3 Design Modelling and Simulation Environment

The Discrete Event System Specification (DEVS) formalism introduced by Zeigler [5] provides a means of specifying a mathematical object called a system. Basically, a system has a time base, inputs, states, and outputs, and functions for determining next states and outputs given current states and inputs. DEVS-Scheme, an implementation of the DEVS formalism in Scheme (a Lisp dialect), supports building models in a hierarchical, modular manner. This is a systems oriented approach not possible in popular commercial simulation languages such as Simscript, Simula, GASP, SLAM and Siman (all of which are discrete event based) or CSMP and ACSL (which are for continuous models).

In the DEVS formalism, one must specify 1) basic models from which larger ones are built, and 2) how these models are connected together in hierarchical fashion. In this formalism basic models are defined by the structure: $M =< X, S, Y, \delta, \lambda, ta >$, where X is the set of external input event types, S is the sequential state set, Y is the set of external event types generated as output, δ_{int} (δ_{ext}) is the internal (external) transition function dictating state transitions due to internal (external input) events, λ is the output function generating external events at the output, and ta is the time-advance function. Rather than reproduce the full mathematical definition here, we refer the reader to [5].

Given a composition tree CT, we specify basic models for each leaf entity and coupling descriptions governing the hierarchical connection of basic models. Coupled basic models form coupled models that may in turn be connected to form a higher level coupled model. The coupled model specification is accomplished automatically in the DEVS-Scheme environment. Figure 2b illustrates the models resulting from the composition trees of Figure 2a. In the context of design solution generation, DEVS-Scheme is used as means of design model behavior specification and evaluation by simulation.

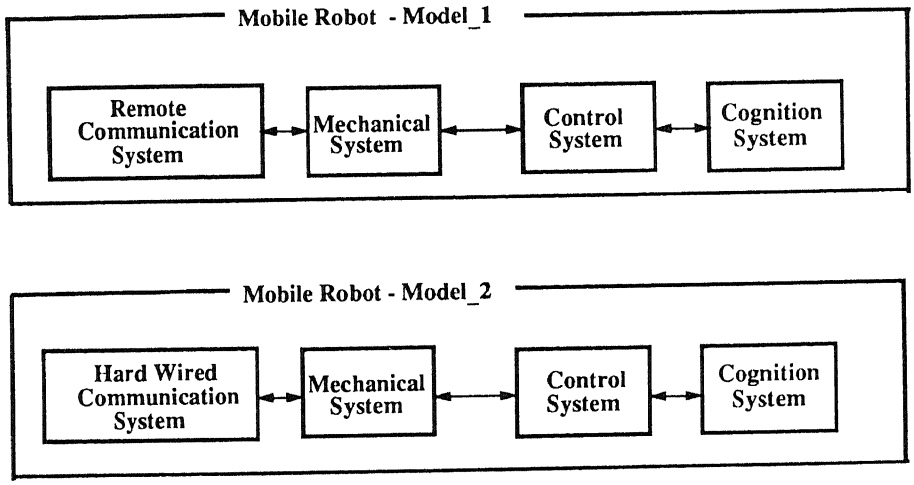

Fig. 2b Model Structures for Composition Trees of Fig. 2b

3.4 Simulation and Evaluation

We separate the model description from a simulation experiment under which the model is observed. This facilitates much greater flexibility in design model ranking and relieves the models from the burden of collecting data about themselves.

A set of circumstances under which a model system is to be observed and experimented with is called *an experimental frame*. Zeigler [5] has shown that an experimental frame can be realized as a coupling of three components: a generator (supplying a model with an input segment reflecting the effects of the external environment upon a model), an acceptor (a device monitoring a simulation run), and a transducer (collecting and processing model output data). The specification of experimental frames in the DEVS-Scheme environment is equivalent to that of specifying basic models and their corresponding couplings.

Experimental frames reflect I/O performance design requirements. For example, an experimental frame for evaluating the average task processing time by a robot could have the following constituents: a generator producing a workload of tasks for the robot model, an acceptor monitoring the observation interval and the length of the queue of tasks at the robot's workcell, and a transducer recording the times of task completions, and computing the average task execution time.

The data collected from alternative design models may now be compared in order to select the best design solution. Clearly, design performance measures may conflict with one another. Therefore, we propose to use Multiple Criteria Decision Making (MCMD) as a means for ranking candidate designs. The concept of design selection using the experimental frame is depicted in Figure 3. Some MCMD criteria and simple examples of design trade-offs are presented in Hu [9].

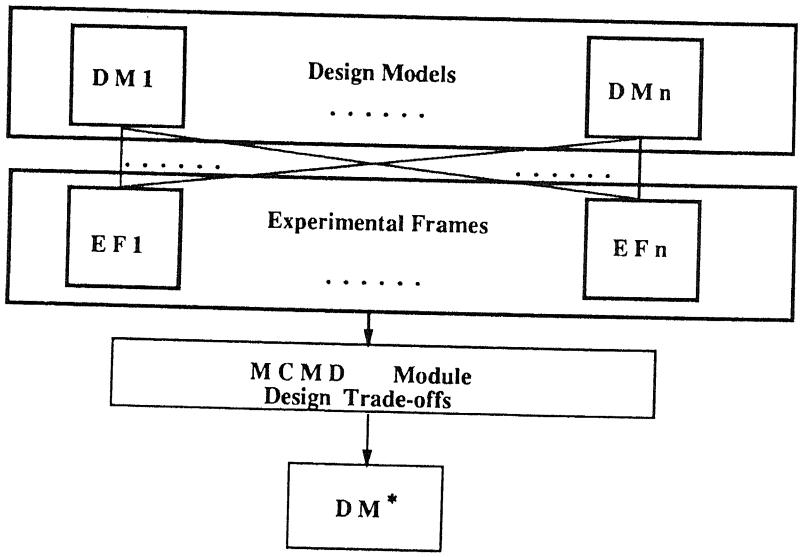

Fig. 3 Design Evaluation and Selection

To gather up the strands, we now summarize the design solution generation phases.

4. DESIGN GENERATION PHASES REVISITED

1. We conceptualize decompositions and specializations of components of the system being designed using the system entity structure. Design models associated with atomic entities must be developed and placed in the model base. The system entity structure generates the set $Gen(DES)$ of all composition trees underlying possible design models.

2. We develop a rule base to be used in the pruning process.

3. We invoke the pruning engine to generate recommendations for candidate solutions to the design problem in the form of model composition trees. This results in the set $CT_D = \{CT_1, CT_2, ..., CT_k\}$ such that $SCR(CT_i) \subseteq SCR$ for i = 1,2,..., k.

4. We invoke the transformation procedure that synthesizes models from the composition trees obtained in phase 3 using the DEVS-Scheme modelling environment.

5. To carry out a simulation experiment, we specify an experimental frame. We accomplish this by defining DEVS-Scheme components that: a.) generate input stimuli to the model, i.e., discrete event input segments; b.) observe model output; and c.) control the simulation experiment by observing model variables.

6. We evaluate simulation results and rank models with respect to the performance measures that express design objectives and requirements. MCMD criteria are used to define the *best* function which generates $DM^* = best\{DM_i; i = 1, ..., n\}$.

5. CONCLUSION

We have described a systematic approach for generating simulation models of systems. In the general system design context, the presented framework affords the following benefits: the formal concepts offered by our methodology integrate the design steps, facilitate a uniform treatment of design at different levels of abstraction. By providing performance evaluation mechanisms, the methodology facilitates comparative studies of design alternatives. This improves the decision making process, and eventually increases the chances of getting managerial approval for installing the proposed solution.

REFERENCES

[1.] Gero, John S. et. al. An Approach to Knowledge-Based Creative Design, *Proc. of NSF Engineering Design Research Conference*, Amherst, pp. 333-346, June 1989

[2.] Yang, J. and J.W. Rozenblit, Case Studies of Design Methodologies: A Survey, *Proc. of the International Conference on AI, Simulation and Planning in High Autonomy Systems*, IEEE Computer Press, pp. 136-141, 1990

[3.] Rozenblit, J.W., A Conceptual Basis for Integrated, Model-Based System Design, Ph.D. Dissertation, Department of Computer Science, Wayne State University, Detroit, Michigan, 1985

[4.] Rozenblit, J. W. and Zeigler, B. P. Design and Modelling Concepts, in: *International Encyclopedia of Robotics, Applications and Automation*, (ed. Dorf, R.) John Wiley and Sons, New York, pp. 308-322, 1988

[5.] Zeigler, B. P., *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, London, 1984

[6.] Zeigler, B. P., Hierarchical, Modular Discrete Event Modelling in an Object Oriented Environment. *Simulation Journal*, vol 49:5, pp. 219-230, 1987

[7.] Rozenblit, J. W. and Y. M. Huang, Rule-Based Generation of Model Structures in Multifacetted Modelling and System Design, *ORSA Journal on Computing* (submitted)

[8.] Huang, Y. M., Building An Expert System Shell for Design Model Synthesis in Logic Programming, Master Thesis, University of Arizona, Tucson, Arizona, 1987.

[9.] Hu, J., Knowledge-Based Design Support Environment for Design Automation and Performance Evaluation, Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona, 1989