Modeling and simulating optical computing architectures

I. R. Jones Jr. and V. P. Heuring
Department of Electrical and Computer Engineering
University of Colorado, Boulder, CO 80309-0425, USA.
Fax: 303-492-2758.
E-mail: jonesi@colorado.edu, heuring@colorado.edu

Abstract

The major issues in system design are timing, synchronization, and control. In designing free-space optical computing architectures, the application of CAD tools is necessary because of the high degree of system complexity, parallelism and concurrency; in conjunction with the high cost and lack of availability of devices. Current CAD tools lack the expressiveness to model system structure and behavior of parallel and concurrent architectures. Thus, making them inefficient and ineffective.

Petri nets, in comparison to other system modeling methodologies, are shown to be more efficient and effective at expressing the functional, behavioral, and structural properties of parallel and concurrent architectures. This paper shows how an extended version of the standard Petri net, a timed-colored Petri net (TCPN), is used to model and simulate free-space optoelectronic computing architectures.

Keywords

Optical computing, Petri nets, timed Petri nets, colored Petri nets, system modeling, simulation, discrete event systems.

1 INTRODUCTION

1.1 Free-Space Optical Computing Architectures

The attractiveness of applying optics to computing paradigms is its inherent advantage of speed, parallelism, and immunity to electromagnetic interference (EMI). Computing at the speed of light can push data rates to 100s of Gb/s. With immunity to EMI, optical signals are not affected by electronic noise. The parallelism of optics can be exploited by simultaneously transmitting optical signals of different frequencies through the same medium without mutual interference; or by broadcasting (fanout) an optical signal to several receptors [1].

There are two classes of optical computing architectures -- guided wave and free-space. A guided wave system is analogous to an electrical system: optical signals propagate through wave guides like electrical signals through wires. These systems are viewed as two dimensional (2D) systems. In free-space architectures, the propagation of optical signals is not constrained in the path perpendicular to the direction of propagation. These systems are three-dimensional (3D) systems. The advantages of 3D over 2D architectures are higher-density in connectivity, no physical contact for interconnections, high spatial and temporal bandwidth, low signal dispersion (high speed data transfer), and massively parallel communication [1][11].

The basic architecture used in many free-space optical computing systems consists of a 2D array of optoelectronic logic devices (smart pixels) followed by holographic or diffractive elements that serve as interconnects to direct the signals [8][11]. Figure 1 shows an architecture with two stages.

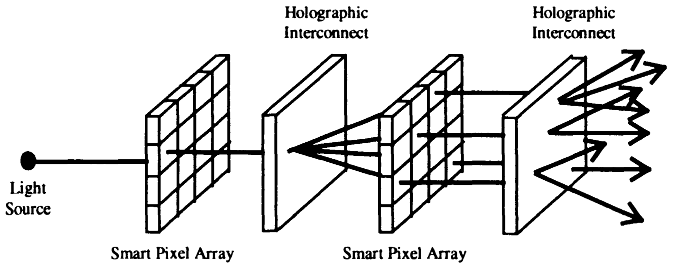


Figure 1 A two stage free-space optoelectronic architecture with 2D smart pixel arrays and holographic interconnects.

1.2 System Design Issues

The major functional issues in system design are alignment, power budgeting, timing, synchronization, and control. The timing behavior is defined as the effect that the location of a particular signal has on the functioning of a system. Signal degradation, propagation path length, delays along the propagation path, and the duration of the asserted value can cause variations in the behavior of the system. Synchronization defines the precise timing relationship between signals. The control behavior of a system is the order of occurrence of synchronized groups of signals. For optical architectures, predicting the timing, synchronization, and control behavior becomes increasingly difficult as the size and complexity of the system grows.

1.3 Modeling Methodologies

There are various modeling paradigms that span the graphical and mathematical domains. Mathematical models provide functional, behavioral, and temporal information, but do not show any structural details. Graphical models can provide functional, behavioral, structural, and temporal information, but can become large and unmanageable. Of the various system modeling methodologies, control flow diagrams (CFD), data flow diagrams (DFD), finite state machines (FSM), and state charts (SC), the Petri net (PN) is the best choice. The Petri net remains manageable in size compared to the growth rates for representation of parallel and composite systems for state machines and control flow diagrams. State based (CFD, FSM, and SC) and event driven (DFD, and FSM) methods lose structure due to the abstraction of state. For event driven models, timing is difficult to emulate [6][12]. This is summarized in Table 1.

Table 1 Comparison of system design models

Model	# of Nodes	Parallel System	Composite System	Shows Structure	Time	Synchron'zn	Control
PN	p	p	$p_1 + p_2$	Yes	Yes	Yes	Yes
DFD	q	q	$q_1 + q_2$	No	No	Yes	No
CFD	r	2^r	$r_1 * r_2$	No	Yes	Yes	Yes
FSM	s	2^s	$s_1 * s_2$	No	No	Yes	Yes
SC	t	t	$t_1 + t_2$	No	Yes	Yes	Yes

2 PETRI NETS

2.1 Basic Petri Nets

Petri nets are a graphical and mathematical modeling tool for describing and studying information systems characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. Formally [3][10], a Petri net is a 5-tuple, PN = (P, T, F, W, M_0) where:

$P = \{p_1, p_2, ..., p_m\}$ is a finite set of places,

T $= \{t_1, t_2, ..., t_n\}$ is a finite set of transitions,
F $\subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
W: $F \rightarrow \{1, 2, 3, ...\}$ is a weight function,
M_0: $P \rightarrow \{0, 1, 2, 3, ...\}$ is the initial marking,
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

Places denote a state or parameter of the system -- a pre or post condition, a signal, input or output data, or a resource. A transition represents an event, a processing step, or a logical clause. Tokens are used to mark places in the net. Transitions simulate the system's behavior by removing and reassigning tokens to places. A distribution of the tokens in the net designates a state in the system.

A Petri net structure N = (P, T, F, W) without an initial marking is denoted by N. A Petri net with a given initial marking is denoted by (N, M_0). The weight of an arc from a transition to a place, denoted as w(p, t), represents the number of directed arcs in the net from place p to transition t. Likewise, w(t, p) denotes the weight of a directed arc from transition t to place p. An unlabeled arc has a weight of 1.

The behavior of a Petri net is characterized by a change of state, or marking of the graph, by following a transition (firing) rule. The firing rules for the system are as follows:

1. A transition is enabled if each input place, p_i, of transition, t, is marked with at least $w(p_i, t)$ tokens where w is the weight of the arc from p_i to t.
2. An enabled transition may or may not fire.
3. The firing of a transition removes $w(p_i, t)$ from each input place and places $w(t, p_o)$ tokens in each output place, p_o.

An example Petri net is shown in Figure 2.

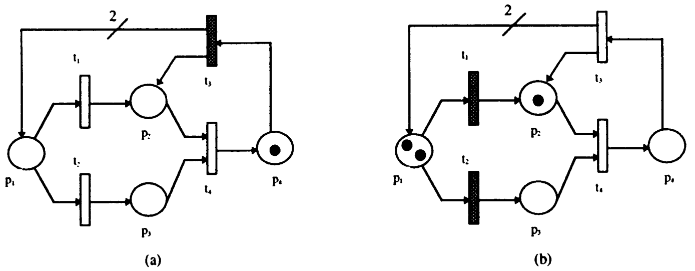


Figure 2 An example Petri net illustrating a transition (firing) rule: (a) The marking before firing the enabled transition t_3. (b) The marking after firing t_3.

2.2 Colored Petri Nets

A colored Petri net (CPN) is a type of high-level Petri net [9]. As shown in Figure 3, the structure of a colored Petri net is identical to that of the standard

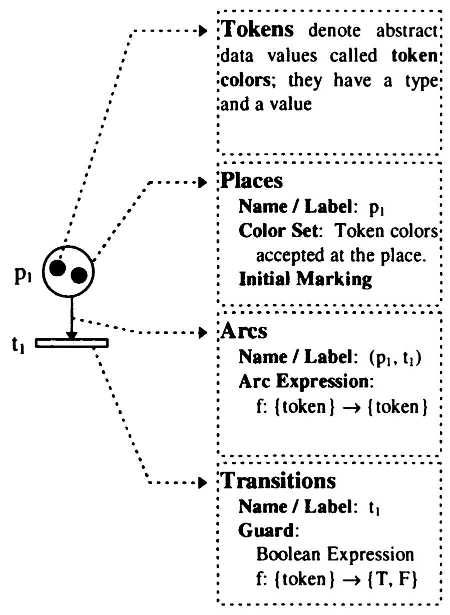


Figure 3 Colored Petri net

net, with the addition that *tokens have an attached attribute called a token color; and the places, arcs, and transitions have become token operators.* A token color can be viewed as an abstract data type with values assigned to the variables. The color set of a place is the type or attribute of a token that may reside in the place. The transition guard is a Boolean expression that when it evaluates to TRUE, enables the transition for firing. The arc expression for arcs(p_i, t_i) consumes tokens. The arc expression for arcs(t_i, p_i) generates tokens.

The action of a transition firing in a colored net is called a binding. A binding assigns a color (value) to each variable of a transition. The binding rules are as follows:

1. A binding is enabled if and only if enough tokens of the correct colors are on each input place and the guard evaluates to true.
2. A binding, when enabled, may or may not occur.
3. When a binding occurs, a multiset of tokens is removed from each input place and a multiset of tokens is added to each output place, depending on the evaluation of the arc expression.

2.3 Timed Petri Nets

Time and determinism are not explicit concepts in the definition of the firing/binding rules in the preceding Petri nets. Yet, they are important in evaluating the performance of a dynamic system and in simulating systems that have definite temporal dependencies. By definition, *a timed Petri net* (TPN) *is a net where time delays are associated with transitions, places, arcs, and/or tokens within the net model.* The delays can be specified deterministically, or stochastically [3][10][13]. Transition delays specify the amount of time a transition takes to fire. Place delays specify the amount of time a token must wait in a place before it becomes active and can therefore enable a transition. The delay value of a timed token specifies the waiting period of a token at a place. Timed arcs assign delay values to tokens.

Our Petri net incorporates each timing method, and abandons non-determinism in favor of urgency semantics: *when a transition is enabled, it is immediately fired.* Modifying the binding rule in this manner enables the model to process all enabled tokens concurrently at the earliest time period when the tokens are active. This rule expresses the parallelism and the dynamic nature of optical architectures.

3 FREE-SPACE OPTOELECTRONIC SYSTEM MODELING

3.1 Device Modeling

Tokens in our system model represent signals. In optical computing systems, a signal can be one of three types -- acoustical, electrical, or optical. The token color is a data attribute associated with its type.

There are two types of devices: passive and active. Passive devices such as mirrors, lenses, beamsplitters, and holograms, behave reactively. Active devices such as emitters, detectors, spatial light modulators (SLMs), acousto-optic devices, and smart pixels (optoelectronic logic gates) [7][5], behave dynamically. The arbitrary device model introduces hierarchical structure to the system model. The arbitrary device model can be an embedded system or some complex device. Figure 4 shows the model structures.

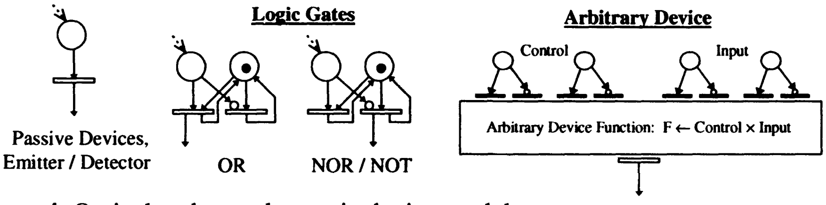


Figure 4 Optical and optoelectronic device models.

The token color and device attributes are data structures that incorporate the three dimensional structure, the functional and behavioral characteristics of the object. The data structure templates are shown in Figures 5 and 6.

3.2 Connectivity (Flow Relationship Between Devices) and Timing

Devices in the system model are interconnected by performing an optical ray-trace. A directed arc is added from the output transition of a source device to the input place of destination device if a directed beam light from the source device is incident to the input of the destination device.

```
<Device_Name>
{
   Attributes
       Geometrical Attributes -- physical shape of the device.
       Functional Attributes -- device operation parameters.
   Color Set -- the types of token accepted at the input place.
   Guard -- Boolean expression to activate the transition.
   Arc (p_i, t_j) Expression
       All tokens in place p_i that satisfy a function input condition.
   Arc(t_i, p_j) Expression
       For each token input token, apply the function
           f: token → { token }        (for passive devices)
           f: { token } × { token } → { token }    (for active devices)
}
```

Figure 5 Device data structure.

```
token_j  (signal)
{
  type = { acoustical, electrical, optical }
  acoustical  / optical                         electrical
  {                                             {
     direction vector,  d = (x, y, z)                power          e
     origin,  l = (x, y, z)                       frequency,     ω
     radius,          r                           wavelength,    λ
     power,           e                         }
     polarity,        a
     frequency,       ω
     wavelength,      λ
  }
  time stamp, s
  event time, τ
}
```

Figure 6 Token data structure.

The timing behavior of a system is reduced to three variables: the input latency of a device, the output latency of a device, and the propagation delay between devices. Place delays model device input latency, transition delays model device output latency, and arc delays model the propagation delay between devices.

The temporal behavior of the system is based on timing information within the arcs that connect devices in the architecture. Let T_b denote the time period when a binding is to occur. A token created at this time will have *time stamp* T_b. The distance that light travels from one device to another is called the optical path length, abbreviated as OPL. Light propagates linearly at a definite speed. The *event time*, τ, of a token to a device is the cumulative sum from T_b of the propagation and device delays along the optical path between devices. The arc expression assigns the event time to the generated token in the following formula, where *c* is the speed of light in air:

$$\tau = output\_device\_latency + \left(\frac{OPL}{c} \right) + T_b + input\_device\_latency \qquad (1)$$

4 SIMULATING OPTICAL COMPUTING ARCHITECTURES

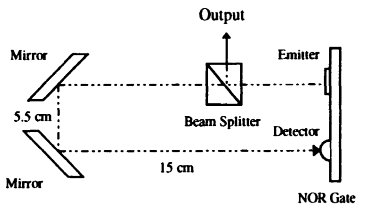
4.1 Discrete Event System

The type of simulator which best suits the defined Petri net model of an optoelectronic computing architecture is a discrete event, block oriented, deterministic, dynamic system [4][3]. An *event* is defined as a change in the state of the system. An event in the TCPN model occurs at the firing of a transition or the activation (the arrival) of a token at a place. The method for generating successive markings in a TCPN is algorithmic, and is defined as follows:

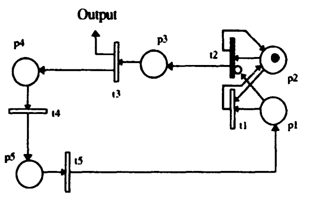
```
Repeat
{
    Find the set of enabled transitions.
    Fire all transitions concurrently.
}
```

4.2 Simulation Example

Figure 7 shows the optoelectronic diagram and Petri net model with an initial marking for a simple oscillator, a NOR gate with a feedback loop from output to input [7][5]. Figure 8(a) shows the timing diagram generated by simulating the oscillator. Figure 8(b) shows the marking of the net as each event occurs.

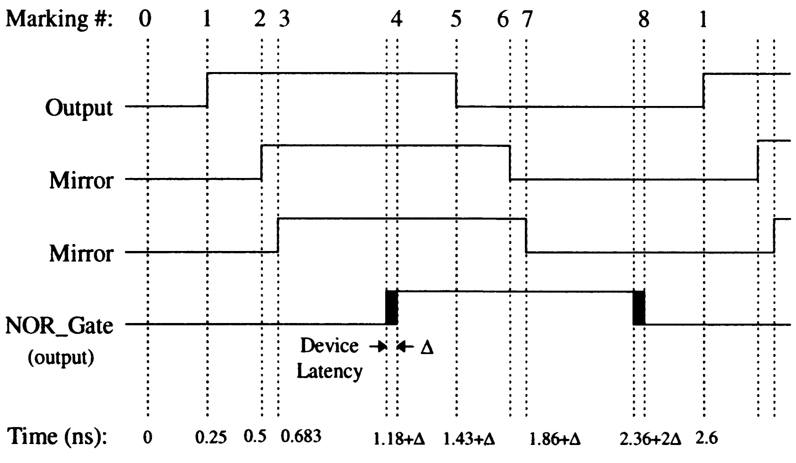


(a)



(b)

Figure 7 Models for a simple optoelectronic oscillator. (a) optoelectronic model, (b) Petri net model showing an initial marking (place p_2) and the enabled transitions = { t_2 }.



Period = 2.36+2Δ ns Frequency = 424 ~ 230 MHz (0 ≤ Δ ≤ 1)

Figure 8(a) Oscillator timing analysis. The occurrence and sequence of events, indicated by the marking of the net is shown.

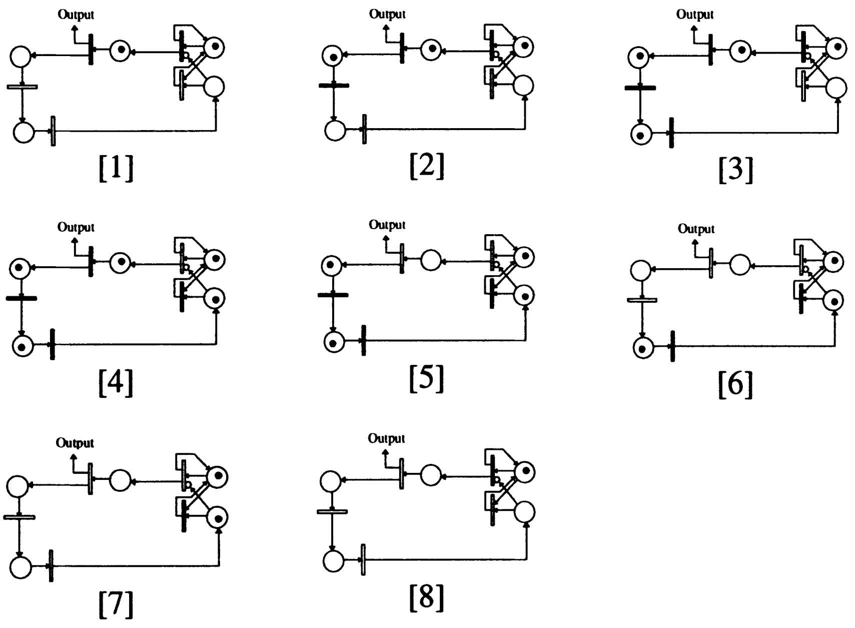


Figure 8(b) Petri net simulation of a simple optoelectronic oscillator. Each diagram shows the current marking obtained from the firing of transitions in the preceding net; and the transitions enabled by the current marking.

4.3 Analysis Techniques for Timing, Synchronization, and Control Behavior

Most of the behavioral properties of the system that are expressed by the model can be analyzed by graphical analysis or by the concept of reachability in Petri nets [3][10]. A Petri net, N, with a given initial marking M_0 is denoted (N, M_0). A marking is denoted by M, an m-vector, where m is the total number of places in the net. The pth component of M, denoted by M(p) is the number of tokens in place p.

Reachability: A marking M_n is said to be reachable from a marking M_0 denoted as $M_0[>M_n$ or $M_0[t_i,..., t_j>M_n$ (reachable via a firing sequence), if there exists a sequence of firings (or bindings) that transforms M_0 to M_n. The notation $M_n \in R(M_0)$ states that M_n is in the set of reachable markings from M_0.

Device placement and component alignment are behavioral aspects of reachability in the network model. Two components are aligned if the output token generated by one becomes part of the color set of the other. The system is aligned if all input places are reachable from their respective output transitions in the network. The placement of a device can be verified by checking its alignment within the system.

Functional verification implies that the sequence of markings generated by the model, when given an initial marking, corresponds to the sequence of states the actual system exhibits. A reachability path for a system is a simulation trace. The functional behavior of an architecture is verified if there is a sequence of markings of the system that corresponds to the defined behavior for the architecture.

Timing and performance measures of the architecture can be analyzed through reachability and/or graphical analysis. The Petri net model can be viewed as a weighted directed graph, where the weights correspond to signal delays (propagation and device latency) in the system. Questions regarding the timing of a signal and system performance are answered by formulating the question into a network flow problem and solving the flow relationship. For example, the minimum and maximum propagation delay of a signal from source to destination corresponds to the shortest and longest path in a weighted network. Measuring system performance becomes an optimization problem where the goal is to find the maximum marking (the maximum flow) in the network given the constrained behavior of the devices in the system [2].

5 SUMMARY

The modeling methodology described in this paper shows how to use a timed-colored Petri net to simulate a free-space optoelectronic computing architecture. The model synthesizes the behavior of an optical device into a functional mapping of an input to an output. This synthesis of behavior is then mapped to the place, transition, and arc components of a Petri net. By associating delay values with the arcs, the places, and the transitions of the Petri net, propagation delay and device latency are incorporated into the structure of the model. The transmission property of light (straight path and constant speed) provides the means to specify the time and place of an optical signal in free-space. This was modeled in the consumption and production of tokens. By observing the process of consumption and production of tokens within the net, we can follow the propagation of signals throughout the system, analyze the timing, synchronization, and control behavior of the system, and synthesize parallel and concurrent activities within the system.

This methodology of modeling and simulating optical computing architectures opens up possibilities for future research in several areas. These are the characterization and description of optical computing architectures via Petri net languages, design automation and optimization of optical computing architectures, and analysis of parallel and concurrent optical systems such as communication systems, network topologies, network switching systems, time multiplexed, and multithreaded architectures.

6 REFERENCES

[1] Cathey, W.T. (1993) Promises and Prospects of Optoelectronic Computing. *LEOS Conference Proceedings*, IEEE Lasers and Electro-Optics Society Annual Meeting, 69-70.

[2] Chen, W-K. (1990) Graphs and Networks, Maximum Flows in Networks., *Theory of Nets: Flows in Networks*, John Wiley & Sons.

[3] David, R. and Alla, H. (1992) *Petri Nets and Grafcet -- Tools for Modeling Discrete Event Systems*. Prentice-Hall.

[4] Denham, M. J. (1988) A Petri-Net Approach to the Control of Discrete-Event Systems. *Advanced Computing Concepts and Techniques in Control Engineering*. NATO ASI Series, **F47**, Springer-Verlag, 191-214.

[5] Ferrarini, L. (1992, May/June) An Incremental Approach to Logic Controller Design with Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics*, **22**, 3.

[6] Gajski, D., Dutt, N., Wu, A.. and Lin, S.. (1992) Architectural Models in Synthesis, Design Representation and Transformations, *High-Level Synthesis -- Introduction to Chip and System Design*, Kluwer Academic Publishers.

[7] Heuring, V.P. Ji, Lian H. Feuerstein, R.J. and V. Morozov. (1994, November 10) Toward a Free-Space Parallel Optoelectronic Computer: A 300-Mhz Optoelectronic Counter using Holographic Interconnects. *Applied Optics*, **33**, 32, 7579-7587.

[8] Hinton, H.S., Cloonan, T.J., McCormick, F.B., Tooley, F.A.P.and Lentine, A.L. (1994, November) Free-Space Digital Optical Systems. *Proceedings of the IEEE -- Special Issue on Optical Computing*, **82**, 11, 1632-1649.

[9] Jensen, K. and Rosenberg, G. (1991) Coloured Petri Nets: A High Level Language for System Design and Analysis, *High-Level Petri Nets: Theory and Applications*, Springer-Verlag Berlin, Heidelberg.

[10] Murata, T. (1989, April) Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, **77**, 4.

[11] Neff, J.A. (1992) Optoelectronic Arrays for Hybrid Optical/Electronic Computing. *Proceedings of SPIE -- Advances in Optical Information Processing V*, **1704**, SPIE, 44-54.

[12] Waxman, R., Bergé, J-M., Levia, O., and Rouillard, J. (1996) *High-Level System Modeling: Specification and Design Methodologies*, Kluwer Academic Publishers.

[13] Zuberek, W. M. (1991) Timed Petri Nets: Definitions, Properties, and Applications. *Microelectronics and Reliability*, **31**, 4, 627-644.

7 BIOGRAPHIES

Irvin R. Jones Jr. received the B.S. degree in electrical engineering from Stanford University in 1982, and M.S. degrees in computer engineering and in computer science from the University of California at Santa Barbara in 1986 and 1988 respectively. He is currently pursuing his Ph.D. in computer engineering at the University of Colorado at Boulder. His interests are computer architecture, optical computing, system modeling and simulating.

Vincent P. Heuring received the B.S. degree from the University of Cincinnati in 1966, and the Ph.D. degree from the University of Florida in 1969. He held various industrial and academic positions until 1984, when he joined the University of Colorado at Boulder. He is an Associate Professor in the Department of Electrical and Computer Engineering. His research interests include optoelectronic computing, software engineering, and software in support of training and education.