Simulating free-space optical computing architectures

Irvin R. Jones, Jr., and Vincent P. Heuring

Major issues in optoelectronic system design include timing, synchronization, and control. Designing free-space optical computing architectures is difficult because of the high degree of system complexity, parallelism, and concurrency in conjunction with the high cost and lack of availability of devices. Current simulation tools lack the expressiveness to model the system structure and behavior of parallel and concurrent architectures, thus making them inefficient and ineffective. We show that Petri nets, compared with other system-modeling methodologies, are more efficient and effective at expressing the functional, behavioral, and structural properties of parallel and concurrent architectures. We show how an extended version of the standard Petri net, a timed-colored Petri net, is used to model and simulate free-space optoelectronic computing architectures. We also present methods for analysis of system timing, synchronization, and control behavior. © 1998 Optical Society of America


1. Introduction

Because there is already a significant infrastructure for modeling and simulating electrical, optical, and optoelectronic devices, this paper focuses on the issues of optical—optoelectronic system-level modeling and simulation that are not addressed within the current infrastructure. Our goals in this paper are to delineate the significant issues in system design as they apply to the design and the development of free-space optical computing architectures and to present a method for modeling and simulating these architectures. The architectures for which the modeling and simulation method apply can include all optical as well as optoelectronic components (e.g., spatial light modulators, vertical-cavity surface-emitting lasers, and optoelectronic integrated circuits). This paper also shows how the structural and the functional properties of the model can be used to analyze system properties such as power loss and cross talk.

A. Free-Space Optical Computing Architectures

The attractiveness of applying optics to computing paradigms comprises the inherent advantages of speed, parallelism, and immunity to electromagnetic interference. Computing at the speed of light can push data rates to hundreds of gigabits per second. With immunity to electromagnetic interference, optical signals are not affected by electronic noise. The parallelism of optics can be exploited by the simultaneous transmittance of optical signals of different frequencies through the same medium without mutual interference or by the broadcasting (fan-out) of an optical signal to multiple destinations.1,2

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 waveguides like electrical signals through wires. We define these systems as two-dimensional (2-D) 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 defined as three-dimensional (3-D) systems. The advantages of 3-D over 2-D architectures are higher density in connectivity, no physical contact for interconnections, high spatial and temporal bandwidths, low signal dispersion (high-speed data transfer), and massively parallel communication.1,3

The basic architecture used in many free-space optical computing systems consists of a 2-D array of optoelectronic logic devices (smart pixels) followed by holographic or diffractive elements that serve as interconnects to direct the signals.3,4 Figure 1 shows a model architecture with two stages.
B. System-Design Issues

The major functional issues in optical 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, the 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 is defined as the precise timing relation between signals. The control behavior of a system is the order of occurrence of synchronized groups of signals. For optical architectures, predicting and computing the timing, synchronization, and control behaviors become increasingly difficult as the size and the complexity of the system grow.

C. System-Design and System-Simulation Tools

Recent advances in optoelectronic technology and system design have prompted the need for development tools. Optical computer-aided-design packages have evolved primarily in the area of design of imaging systems. They offer a variety of features that support coherent and incoherent design but lack the capability to support optoelectronic system design. A few of the standard computer-aided-design packages that are part of this group are OSLO, CODE V, GENI, ACCOS, and SYNOPTIS.

The trend in developing tools for optoelectronic system design is either to integrate electrical and optical design tools into a single package or to enhance the electrical or the optical design tools with features for optoelectronic system design. One such integration effort proposes to use the general laser analysis and design (GLAD) tool as a simulation component in an optical processor design system. OE-SPICE and ISMILE are enhanced SPICE-like applications. The optoelectronic system simulator (OSS) is a VHDL-based tool.

The drawback to using these tools is that the pertinent issues of system design, timing, synchronization, and control are disguised in the discrete analysis of device behavior. SPICE-based models become computationally intense because they cannot efficiently handle parallel and concurrent behavior. A VHDL-based tool has provisions for modeling parallel and concurrent behavior but loses structural visibility. The method that we propose - modeling the system as a timed-colored Petri net (TCPN) - efficiently handles parallel and concurrent behavior and maintains a visible system structure.

D. System-Modeling Methodologies

There are a variety of modeling paradigms that span the graphical and the 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's) data-flow diagrams (DFD's), finite-state machines (FSM's), and state charts (SC's) - the Petri net (PN) is the best choice. Table 1 shows that the PN remains manageable in size compared with the growth rates for representation of parallel and composite systems for state machines and CFD's. State-based (CFD, FSM, and SC) and event-driven (DFD and FSM) methods lose structural description capability because of the abstraction of state. For event-driven models timing is difficult to simulate.

2. Petri Nets

A. Basic Petri Nets

PN's are graphical and mathematical modeling tools for describing and studying information systems characterized as concurrent, asynchronous, distributed, parallel, nondeterministic, or stochastic. Formally, a PN is a 5-tuple, \( PN = (P, T, F, W, M_0) \), where

- \( P = \{p_1, p_2, \ldots, p_m\} \) is a finite set of places.
- \( T = \{t_1, t_2, \ldots, t_n\} \) is a finite set of transitions.
- \( F \subseteq (P \times T) \cup (T \times P) \) is a set of arcs (flow relation).
- \( W : F \rightarrow \{1, 2, 3, \ldots\} \) is a weight function.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of Nodes</th>
<th>Parallel System</th>
<th>Composite System</th>
<th>Structure</th>
<th>Timing</th>
<th>Synchronization</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN</td>
<td>( p )</td>
<td>( p )</td>
<td>( p_1 + p_2 )</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DFD</td>
<td>( q )</td>
<td>( q )</td>
<td>( q_1 + q_2 )</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CFD</td>
<td>( r )</td>
<td>( 2^r )</td>
<td>( r_1 \times r_2 )</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FSM</td>
<td>( s )</td>
<td>( 2^s )</td>
<td>( s_1 \times s_2 )</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SC</td>
<td>( t )</td>
<td>( t )</td>
<td>( t_1 + t_2 )</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1. Comparison of System-Design Models
Fig. 2. Example PN illustrating a transition (firing) rule: (a) The marking before firing the enabled transition $t_3$. (b) The marking after firing $t_3$.

- $M_0 : P \rightarrow \{0, 1, 2, 3, \ldots \}$ is the initial marking (a distribution of tokens to places).
- $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

Places denote a state or parameter of the system: a precondition, a postcondition, 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 the places in the net. The transitions simulate the behavior of the system by means of removing and reassigning tokens to places. A distribution of the tokens in the net designates a state in the system.

A PN structure without an initial marking is denoted by $N = (P, T, F, W)$. A PN with an initial marking given 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 PN is characterized by a change of state or a marking of the graph when a transition (firing) rule is followed. 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 might or might not fire.
3. The firing of a transition removes $w(p_i, t)$ from each input place and places $w(t, p_0)$ tokens in each output place $p_0$.

An example PN is shown in Fig. 2. In this example Fig. 2(a) shows a system with an initial state marking (place $p_4$ marked with a single token) and the transition enabled by the marking ($t_3$). The next state obtained by the system from the firing of transition $t_3$ is shown in Fig. 2(b). Note that, because of the weight of arc $(t_3, p_1)$, two tokens are added to place $p_1$.

B. Colored Petri Nets

A colored Petri net (CPN) is a standard net with the addition that tokens have an attached attribute called a token color, and the places, arcs, and transitions have become token operators (see Fig. 3). 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 could reside in the place. The transition guard is a Boolean expression that, when evaluated to true, enables the transition for firing. The arc expression for arcs $(p_i, t_i)$ consumes tokens. The arc expression for arcs $(t_i, p_i)$ generate 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 there are enough tokens of the correct colors on each input place and the guard evaluates to TRUE.
2. A binding, when enabled, might or might 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.

C. Timed Petri Nets

Time and determinism are not explicit concepts in the definition of the firing–binding rules in the PN’s described in Subsection 2.B. 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 PN is a net in which time delays are associated with transitions, places, arcs, or tokens within the net model. The delays can be specified deterministically or stochastically. 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 PN incorporates each of the timing methods and abandons nondeterminism in favor of urgency semantics: When a transition is enabled, it is immediately fired. Modifying the binding rule in this manner allows 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 by Use of Petri Nets

A. Device Modeling

Tokens in our system model represent signals. For the purposes of this paper, we assume that a signal can be one of three types: acoustical, electrical, or optical. The token color comprises the data attributes associated with its type.

Figure 4 shows the model structures. There are two types of device: passive and active. Passive devices, such as mirrors, lenses, beam splitters, and holograms, behave reactively; active devices, such as emitters, detectors, spatial light modulators, acoustooptic devices, and smart pixels (optoelectronic logic gates), behave dynamically. The arbitrary device model introduces a hierarchical structure to the system model. The arbitrary device model can be an embedded system or some complex device.

The token color and the device attributes are data structures that incorporate the 3-D structure and the functional and behavioral characteristics of the object. The data-structure templates are shown in Figs. 5 and 6. An example of the correspondence between the device attributes and the device model is shown in Fig. 7, representing a smart pixel. This model is an interconnection of a detector and an emitter. Only a single-input detector (place $p_d$) is needed for multiple-input fan-in because the device input
signals are optical.\cite{2} Note that this model uses an inhibitor arc, arc \((p_1, e_2)\). In the generic sense of a PN an inhibitor bubble on an arc to a transition disables the transition when the input place has a token and enables the transition when the input place has no token. For high-level PN's the behavior of the inhibitor bubble is relaxed so that the presence of a token at the input place will not necessarily disable the transition. This is illustrated in the guard expression for transition \(t_2\) of the smart pixel shown in Fig. 7. The smart pixel will produce an output token only when the power present at place \(p_2\) is above the defined threshold value and when place \(p_1\) either has no tokens or the combined power of the tokens present at place \(p_1\) does not exceed the threshold value for turning off the device. The output of the smart pixel is the logical NOR of the input signals at place \(p_1\).

B. Connectivity (Flow Relation Between Devices) and Timing

We interconnect devices in the system model by performing an optical ray trace. A directed arc is added from the output transition of a source device to the input place of a destination device if a directed beam light from the source device is incident upon the input of the destination device.

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 the timing information within the arcs that connect devices in the architecture. Let \(T_b\) denote the time period when binding is to occur. A token created at this time will have a time stamp of \(T_h\). The distance that light travels from one device to another is called the optical path length (OPL). Light propagates linearly at a definite speed. The event time \(\tau\) of a token to a device is the cumulative sum from \(T_h\) of the propagation and the device delays along the optical path between devices. The arc expression assigns the event time to the generated token according to

\[
\tau = \text{output-device-latency} + \left(\frac{\text{OPL}}{c}\right) + T_h + \text{input-device-latency},
\]

where \(c\) is the speed of light in air.

4. Simulating Optical Computing Architectures

The methods of execution in optical architectures are gate and strobe and time of flight.\cite{18,19} In the gate-and-strobe method the data are gated from the storage element to the processing element and strobed into the next storage element. If the pulse rise or fall time or the signal skew is an appreciable fraction of the propagation time between or through processing elements, the pulse duration can be made to equal the propagation time from storage element to processing element and back to storage element. In the time-of-flight method signals are allowed to flow from storage element to storage element according to their propagation speed. If the signal skews and pulse rise and fall times are short compared with the propagation time between devices, information flows through the system at a speed governed strictly by the propagation time through the system.

The gate-and-strobe paradigm is modeled by the synchronization of transition firings. The time-of-
flight paradigm is modeled dynamically by the event times of the tokens.

A. Discrete-Event System

The type of simulator that is best assimilated into the defined PN model of an optoelectronic computing architecture is a discrete-event, block-oriented, deterministic, dynamic system. An event is defined as a change in the state of the system. An event, the TCPN model, occurs at the firing of a transition, the activation (the arrival) of a token at a place, or the departure of a signal from the model boundary. 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.
}

B. Analysis Techniques for Timing, Synchronization, and Control Behaviors

CPN's can be analyzed in three different ways. The first analysis method is simulation. This implies executing the net model to derive statistics about the behavior of the system. The second analysis method is occurrence graphs (also called state spaces or reachability graphs). An occurrence graph is a directed graph construct that has a node for each reachable system state and an arc for each possible state change. The third analysis method is place invariance whereby a set of equations is used to derive properties of the modeled system that are known to be satisfied for all reachable system states. Simulation provides a means of understanding system behavior through observation of system activity. Occurrence graphs predict system behavior and are a means of proving state-dependent properties of systems. The graphs show the set of possible states and the transitional path between states. Place-invariant analysis is useful in proving static properties of systems, such as the maximum power consumption and performance bounds.

Pulse degradation derived from pulse dispersion, pulse jitter, long pulse rise and fall times of processing elements, or loss of pulse amplitude or energy caused by loss in passive elements of the system can cause timing and synchronization problems in optoelectronic circuits. Pulse degradation is indicated by a reduction in the power value or in the duration value of a token. If these values fail to fall within a predefined range, appreciable losses in the system have compromised the integrity of the signal. We can determine the losses by checking the power and the duration values of the tokens between each binding iteration of the simulation. Signal restoration is accomplished by means of pulse stretching and clock gating.

Signal skew is another cause of timing and synchronization problems in optoelectronic circuits. Signal skew is defined as the difference in arrival time between the earliest arriving pulse and the latest arriving pulse at a point of interaction. A major cause of signal skew in free-space optical systems is the difference in OPL between a set of signals. In the system model, we detect the signal skew by finding the difference between the maximum and the minimum event times of all tokens at a point of interaction with the same time stamp.

Cross talk and interference are defined as undesired leakage of a signal from one channel to another or different signals meeting at a point of interaction. The result is a superposition of behavior. This is due to device fabrication flaws or signal jitter. If multiple tokens are assigned to a place, there is a possibility that cross talk or interference has occurred. Also, the possibility of cross talk or interference exists if the ray trace of the system generates a connection between components that is not designed in the actual system. These aspects of cross talk and interference can be detected and verified by means of simulation.

Device placement and component alignment are behavioral aspects of control and 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. We can verify the placement of a device by checking its alignment within the system. The reachability (occurrence) graph verifies device placement and component alignment if, for all valid system states between out-
put and input devices, there is an arc between the state nodes in the reachability graph.

Functional verification, a control property, 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. 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. An analysis of the occurrence graph for the model will show all function sequences for the system.

Timing and performance measures of the architecture can be analyzed through reachability or invariance analysis. The PN model is viewed as a weighted directed graph in which 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 when the question is formulated into a network flow prob-

Fig. 9. (a) PN 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. (b) Oscillator timing analysis. The occurrence and the sequence of events is indicated by the marking of the net.
lem and the flow relation is solved. For example, the minimum and the maximum propagation delays of a signal from source to destination correspond to the shortest and the longest paths in a weighted network. Measuring system performance becomes an optimization problem in which the goal is to find the maximum marking (the maximum flow) in the network, given the constrained behavior of the devices in the system.21

5. Simulation Results
The described PN modeling method was used to develop a system-level simulation tool for optical computing architectures. The current capabilities of the tool that is under development are 3-D viewing and animation of system architectures, PN discrete-event simulation control, system function verification, and system timing analysis. To prove the correctness of the simulator, we carried out a simulation test of a simple oscillator and a signal-stretch circuit. The simulation results for the oscillator circuit were compared for function and timing correctness with the experimental test results of the oscillator-circuit design used to test a 300-MHz optoelectronic logic gate.16 A second test of the simulator, the signal-stretch circuit, was a designed experiment. The simulation results for this test allowed us to verify function and timing correctness with the expected results of the design.

The basis of the test assumed binary token-power values, unspecified token-duration values, and no loss in signal integrity during propagation and device interaction. A dynamic execution mode of simulation was used. In dynamic execution, when an event time is reached, the event itself is processed, and the scheduling of future events is carried out. Execution in this manner allows the system to run freely and to determine token-duration (signal-duration) values dynamically. It should also be noted that event duration does not affect transition-firing behavior because only a single firing of a transition is allowed for each event occurrence.

Figure 8 shows the optoelectronic diagram and the PN model with an initial marking for a simple oscillator, a NOR gate with a feedback loop from output to input.16,17 Figure 9(a) shows the marking of the net as each event occurs. Two experiments were done: the first with active devices with no latency and the second with active devices with a latency of 0.5 ns. The oscillation period is approximately 2 times the OPL from emitter to detector, resulting in a frequency of between 297 and 424 MHz. These results match the expected system behavior, given the setup.
in Fig. 8. The timing diagram generated from the simulation of the oscillator is shown in Fig. 9(b).

In time-of-flight design optical signals are timed to arrive at a point of interaction simultaneously. Timing errors caused by signal skew have an additive effect as signals propagate through the circuit. For restoring synchronization, the asserted value of signals that arrive at the point of interaction earlier are stretched to accommodate the latest signal arrival. The circuit that accomplishes signal stretch is shown in Fig. 10(a). By adjusting the distance from the mirrors to the beam splitters, we can obtain a maximum stretch of 1.81 ns. Figure 10(b) shows the PN model of the circuit. The resulting timing diagram is shown in Fig. 10(c).

6. Summary

The modeling methodology described in this paper has shown how to use a TCNP 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 PN. By the association of delay values with the arcs, the places, and the transitions of the PN, 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 of specifying the time and the place of an optical signal in free space. This has been modeled in the consumption and the production of tokens. By observing the processes 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.

The outcome of our research in modeling and simulating optical computing architectures has resulted in the development of a system-level simulator. The simulator provides the means to visualize and analyze the behavior of complex systems through analysis of PN's. This has been demonstrated in the simulation of two systems: a simple oscillator and a signal-stretch circuit. The simulation test results matched the predicted behavior of the actual system in the areas of function and timing.

This methodology of modeling and simulating optical computing architectures can open up prospects for future research in the areas of the characterization and the description of optical computing architectures by means of PN 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, and time-multiplexed and multithreaded architectures.

References
