Asynchronous, Distributed Event Driven Simulation Algorithm for Execution of VHDL on Parallel Processors

Peter A. Walker & Sumit Ghosh
Division of Engineering, Brown University, Providence,Rhode Island 02912
{paw,sg}@lems.brown.edu

Abstract

This paper describes a new **A**synchronous, **P**arallel, **E**vent **D**riven **S**imulation algorithm with inconsistent event **P**reemption, P^2EDAS. P^2EDAS represents a significant advancement in distributed conservative digital circuit simulation algorithms in that it permits the use of any number of non-zero propagation delays for every path between the input and output of every hardware entity. P^2EDAS permits, accurate, concurrent, asynchronous, and efficient, i.e. deadlock free and null-message free, execution of sequential and combinatorial digital designs on parallel processors. It is a conservative algorithm in that only those output transitions, if any, are asserted at the output of a model following its execution, that are guaranteed correct. In addition, preemption of inconsistent events are allowed. P^2EDAS extends to any simulator based on high-level hardware description language. This paper presents a detailed description of the algorithm.

1 Introduction

Debenedictis, Ghosh, and Yu successfully introduced a novel algorithm for asynchronous, distributed discrete event simulation, YADDES [4] that is mathematically proved to be deadlock-free and accurate. In discrete event simulation, a behavior model C^1, referred to as component or entity, executes when stimulated by an input event or cause. Upon execution, C^1 may generate an output event which is then propagated to subsequent components, C^i,...,C^j, connected to the output of C^1. Subsequently, one or more of the components, C^i,...,C^j, may be executed. The process continues until the number of outstanding events is nil. In general, execution of the components, C^i,...,C^j, connected to the output of C^1, are initiated by output events generated from the execution of C^1 ; the execution of components proceed concurrently as conditions allow. To increase concurrency in the simulation and thereby improve efficiency, YADDES utilizes a scheme referred to as the data-flow network, that quickly generates the "earliest time of next event" at the output of C^1 prior to the generation of the output event of C^1. The "earliest time of next event," also referred to as the *predicted event time*, represents a conservative estimate on the time of the subsequent event at the output of C^1. Thus, components C^i,...,C^j may concurrently execute, without violating any data dependency, input events with times less than earliest time of next event to arrive at its inputs. The underlying assumption is that the computation cost of predicting the earliest time when an event E^j may be generated as a result of the execution of C^1, corresponding to an input event E^i, is less than the computation cost of executing C^1. For simulations with complex VHDL models [1], this assumption is, in general, true. An added problem when simulating digital designs with feedback loops, utilizing the traditional distributed discrete event simulation algorithm, is that, often, an output event may not be generated following the execution of a component, C^1. Thus, other components connected to its output, cannot execute. If the output of one of these components is, in turn, connected to the input of C^1, then deadlock occurs [3]. The data-flow network of YADDES successfully addresses this problem. The need for null messages [2] is also eliminated.

In YADDES , corresponding to every simulated component the data-flow network consists of pseudo-primed and pseudo-unprimed components. These pseudo components are purely mathematical entities that execute specific functions to generate W (or W') values at their outputs. They are interconnected in an acyclic manner that is detailed subsequently in this paper. The data-flow network executes concurrently with the actual simulation. The W (or W') value asserted at the input of a pseudo component reflects an estimate of the earliest time that an event may be propagated along the input. For complete definitions of W (or W'), the reader is referred to [4].

Consider a behavior model C^n with inputs, {1,2,...,i}. The *window of temporal independence*, t^n_{win}, is defined as the minimum over all the W (or W') values over inputs $i \in \{1 \ldots I\}$ of the pseudo-unprimed component(s) corresponding to a component C^n. Input events with time t, on an input of C^n may be executed, without violation of data dependency, provided, t is less than t^n_{win}.

32nd ACM/IEEE Design Automation Conference ®

Correctness is guaranteed through accurate values for W (or W'). Formally,

$$t^n_{win} = minimum(W_i \quad or \quad W'_i) \qquad \forall i \qquad (1)$$

While successful as an asynchronous, distributed discrete event simulation algorithm, YADDES is inappropriate for simulating behavior model such as VHDL models since it is limited to a single, input-independent, propagation delay between the input and output of a component. Further, for correctness in the simulation, preemption or descheduling of inconsistent events [1][5] is necessitated. The YADDES algorithms lacks any mechanism to achieve inconsistent event descheduling.

2 The Role of Event Prediction Network in P^2EDAS

A key concept in P^2EDAS is the *event prediction network*. It is analogous to the data-flow network in YADDES and is responsible for asynchronous and concurrent yet accurate simulation of the behavior models. The proceeding discussions provides motivation for, and explanation of the function, of this network along with theory of its synthesis.

2.1 Challenges to Parallel/distributed circuit simulation

In uniprocessor-based simulation, the centralized scheduler maintains a queue of all events in the system. For accuracy, the event with the smallest time is first executed. Correctness and accuracy are preserved by the strict chronological execution of the events. A well documented limitation of the uniprocessor approach is that it precludes the concurrent execution of two or more events that lack any data dependency.

To address this problem, the asynchronous distributed discrete event simulation algorithm, P^2EDAS partitions each component of a system under simulation to a unique processor. Components may be executed independently and concurrently by processors as soon as events are available at their input ports. Given the absence of the global event queue, to ensure correctness of simulation P^2EDAS requires the following. A *simulation clock*, $clock^N$, is computed for every component N, and is defined as the time up to which the model has been simulated. Thus, the simulation clock value is identical to the minimum of all input events at all input ports. A difficulty in the simulation of digital systems is that, often events are not generated at an output port of a component following its execution. Under such circumstance, events are not propagated over to the inputs of subsequent components that are connected to the output and the simulation clock value stagnates periodically.

A more serious problem occurs with sequential subcircuits. Often, an output event may not be generated following the execution of a component, C^1. Thus, other components connected to its output, cannot execute. If the output of one of these components is, in turn, transitively connected to the input of C^1, then deadlock may occur.

2.2 Event Prediction

P^2EDAS addresses both problems through the use of the *event prediction network*. The network is synthesized for the digital system under simulation and executes concurrently with the simulation of the behavior descriptions. It consists of mathematical entities, termed *pseudo components*, that generate *predicted event time values.* A predicted event time, defined at an output port of a pseudo component and associated with the corresponding output of the component, is the time at which an event is expected to be generated at that output and it is guaranteed that no events with assertion times less than the predicted event time will be generated. The predicted event time is computed separately, yet concurrently, from the actual simulation of the behavior descriptions. The simulation utilizes the predicted event times, generated by the event prediction network, to efficiently, yet correctly, schedule executions of the models for appropriate events. Figure

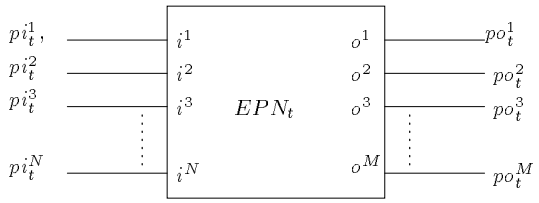

Figure 1: Event Prediction Network

1 presents a simple event prediction network for the behavior description of a circuit component. Although it is constituted by pseudo component(s), the network is conceptually represented through a single black box, EPN_t with inputs i^1, ... i^N and outputs o^1, ... o^N that correspond to the inputs and outputs of the simulation model. The black box EPN_t receives predicted event times, represented by pi^1_t, ... pi^N_t, at its input ports from the corresponding output ports of the preceding EPN_ts. EPN_t generates two quantities – (1) the predicted event times at its outputs that are, in turn, propagated to the inputs of subsequent EPN_ts, and (2) t_{win} that defines which input events may trigger the execution of the corresponding behavior model. EPN_t does not require detailed knowledge of the behavior expressed in the model from which it is constructed. The behavior of EPN_t is detailed subsequently.

2.3 Synthesis of Event Prediction Network

In constructing the event prediction network for a digital system under simulation, first the combinational and sequential subcircuits are identified. Next, for a subcircuit containing feedback loops, a feedback arc set

S given by $S = \{E_1, E_2, \cdots E_n\}$ of a directed graph corresponding to the subcircuit is identified such that the graph may be rendered acyclic following the removal of all of the edges in S. The correctness is not affected by the identification of the minimal feedback arc set which is difficult and time consuming. For each $E_i \quad \forall i \in \{1, 2, \ldots, n\}$, in the original directed graph, a new acyclic directed graph is constructed by replacing E_i with two unconnected edges E_i^{in} and E_i^{out} as illustrated through Figure 2.

Assume that the feedback arc set for the circuit is given by $S = \{E_1\}$. The graph is rendered acyclic in Figure 2b through the removal of E_1 and replacing it by E_1^{in} and E_1^{out} associated with the input of **A** and the output of **B** respectively.

The event prediction network for the circuit, is synthesized from connecting two identical copies of the acyclic circuit through a crossbar switch. The two acyclic circuits to the left and right of the crossbar switch are referred to as the *tail network* and *head network* respectively. The entities in the event prediction network are termed *pseudo components* and identified individually as X_t and X_h respectively, where X refers to the corresponding simulation model. Pseudo components in the tail network are identified through X_t while those in the head network are expressed through X_h. Every input port of a pseudo component X_t that has a label of the form E_i^{in} is permanently held at an infinitely large number represented by the symbol ∞. Since inputs to pseudo components are times of events, the symbol ∞ represents the fact that no finite time values will be propagate through such inputs. An output port of every X_t that has a label of the form E_j^{out} is linked to the input port of any pseudo component Y_h in the head network that has a label of the form E_j^{in}.

For the cyclic graph in Figure 2a, the corresponding event prediction network is shown in Figure 3.

3 The P^2EDAS Algorithm

In P^2EDAS, the behavior models and event predictions network execute asynchronously and concurrently. The dependence between them is explained as follows. When external events are asserted at the primary input ports of the models, the event times are made available to the corresponding inputs of the pseudo components. In general, a model is permitted to execute appropriate events, limited by the value of the t_{win}. Pseudo components use the T_i values at its primary inputs, the predicted event times at all other inputs, the event times associated with the input ports of the behavior models, and the propagation delay values, in their computation. The head pseudo components, generate t_{win} values for the corresponding behavior models which are used by the simulation system to initiate execution of appropriate events. Upon execution of a behavior model, events may be generated at the output port(s). However, the events are not immediately asserted at the output ports. Only those output events are asserted at the respective outputs at appropriate times that are not preempted based on the rules established in [1][5]. To preserve causality and thereby guarantee correct simulation of the digital system, events must be executed in their causal order. This implies that, at every head pseudo component, the t_{win} value must increase monotonically.

Corresponding to every behavior model C^n is a $clock^n$. C^n has access to the t_{win}^n value associated with the corresponding head pseudo component. Define S_o^n as the logical value of the most recent event asserted at the output o of C^n. Given that P^2EDAS employs preemption, events generated as a consequence of execution of C^n are stored in an output queue and not immediately asserted at the output port of the behavior model. Define $t_{e_o}^n$ as the time of the earliest event in the output event queue of output o. Where the output event queue is empty, $t_{e_o}^n$ is set to ∞. If one or more input ports of C^n are primary, T_i defines the assertion time of next earliest event at that input. Assume also that t_{e_i} represents the time of the earliest event at non-primary inputs, i, of C^n. P^2EDAS is explained for cyclic subcircuits since their simulation is most complex. Thus, assuming that C^n is included in a feedback loop, the event prediction network will consist of C_t^n and C_h^n, representing the tail and head pseudo components respectively. Corresponding to every head and tail pseudo component of C^n, the quantities W_i^n and W_o^n are associated with every input port i and output port o respectively. While W_i^n signifies the predicted event time at the input, W_o^n represents the predicted event time at the output.

Initialization consists of the following steps:

- For each behavior model define, C^n, $clock^n$ (at the model) and t_{win}^n (at the corresponding head pseudo component) are set to 0.
- For every pseudo component C_t^n and C_h^n, the predicted event times at all output ports are set to zero. Thus,
$$W_o^n = 0 \qquad \forall n, \quad \forall o. \tag{2}$$
- Input ports of the tail pseudo components, C_t^n, that are associated with the feedback arc set, the W_i^n values are set to ∞. The W_i^n values at all other input ports of pseudo components are initialized to zero.
- Fourth, the S_o^n values at every output of C^n, and its head and tail pseudo component, are user initialized.

Simulation Process: The simulation system ensures the correct execution of the models triggered by appropriate events, using the following steps.

- Model execution: For a given component, C^n, identify input events with time t_{e_i}, such that $t_{e_i} < t_{win}^n$. The behavior model is executed for all such events, starting with the earliest event. For every event executed, the $clock^n$ is advanced, wherever possible. The value of $clock^n$ will always be less than t_{win}^n. The newly generated output events, if any, are included in the output event queue.

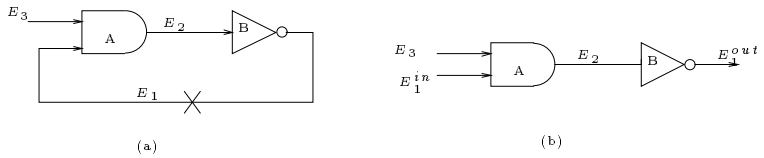

Figure 2: Reducing a Cyclic Directed Graph to an Acyclic Directed Graph

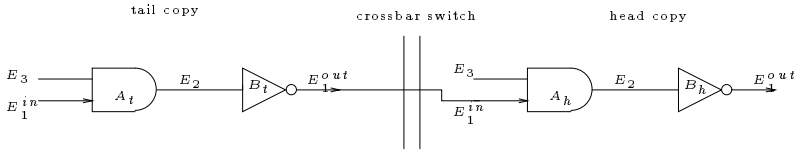

Figure 3: Event Prediction Network for a Cyclic Circuit in Figure 2a

- Identification of Inconsistent Output Events: Due to the nature of the timing semantics of event driven simulation, one or more output events generated as a result of model execution, may turn out to be inconsistent. Such inconsistent events are first identified and then preempted.

- Propagation of Correct Output Events: For each output o of C^n, mark the event with time $t^n_{e_o}$, in the event queue, if the following relationship is true.

$$t^n_{e_o} \leq clock^n \tag{3}$$

  The marked event is noted as correct and is asserted at the output port of C^n. S^n_o is immediately updated. These events are no longer subject to preemption. Whenever an input or output event queue is observed to become empty, a dummy event with assertion time ∞ is inserted in the queue.

- Updating Event Prediction Network: Whenever a new event is asserted at an input port of C^n, the assertion time is propagated to the pseudo components. In addition, whenever the output event queue for an output of C^n is updated, the assertion time of the earliest event is propagated to the head and tail pseudo components. The new logical value, S^n_o, and $t^n_{e_o}$, at the output of the behavior model is propagated to the pseudo components.

- The above three steps are continually executed until either all outstanding events are processed or $clock^n$ exceeds the maximum simulation time of interest.

Event Prediction Network operation: Upon execution, a pseudo component generates a predicted event time, W^n_o, at the outputs. Where the output differs from its previous value, it is propagated right to trigger the execution of subsequent pseudo components in a form of a chain reaction. Whenever any of the input arguments of a pseudo components' change value, the pseudo component is executed. Corresponding to every head or tail pseudo component, a lookup table, $L^n[i, o, s]$, is constructed. The dimensions of the table are determined by the number of input ports, i, output ports, o, and number of logical values, s, that can be realized at o. For an event at i, and with output o at current logical value s, entry $L^n[i, o, s]$ is the minimum transition time to other logical values at o by the event at i. P^2EDAS defines a function, $min\_inertial\_delay()$ which accepts three arguments and performs the required lookup operation. Although VHDL [1] proposes the use of inertial and transport delays in hardware descriptions, in this paper we focus on inertial delays only.

Execution of Pseudo Components:

- First, a pseudo component read the input the predicted event times, W^n_i, from preceding pseudo components and assertion times of earliest input events $t^n_{e_i}$ at the corresponding behavior model. It also receives the new logical values S^n_o at the output ports of the behavior model and the new assertion times $t^n_{e_o}$ of the earliest events in each output event queue.

- The pseudo component uses the accessed values to compute the predicted event time W^n_o at every output o, using the following equation.

$$W_o = min(t_{e_o}, (min(W_i, t_{e_i}) + min\_inertial\_delay(i, o, S_o))) \quad \forall i. \tag{4}$$

  Note there are nil events at an input port of the behavior model, t_{e_i} is ∞. Also, where there are nil events in the output event queue of a specific output, t_{e_o} is also ∞. The head pseudo component of each component computes t^n_{win} values as,

$$t^n_{win} = minimum(W_i) \quad \forall i. \tag{5}$$

Where a given input port, i, is primary, W_i is replaced by T_{max} in the computation of t^n_{win}. T_{max} represents the maximum simulation time of interest and its default value is infinity.

- The newly computed W_o and t^n_{win} values are propagated to the subsequent pseudo components and behavior model, respectively when the values differ from their previous values.
- The above three steps are repeated until the simulation process terminates.

3.1 An Example to Illustrate P^2EDAS

To illustrate the working of P^2EDAS, consider a cyclic subcircuit In Figure 4(a), each of the two NAND gates, **A** and **B**, have two inertial delay values given by $T_{pLH} = 10ns$ and $T_{pHL} = 5ns$ that correspond to low to high and high to low switching at the output. The function $min\_inertial\_delay(i, o, S_o)$ (represented as $delay(i, o, s_o)$ in figures) returns the value for T_{pLH} corresponding to $s_0 = 0$ and the value of T_{pHL} corresponding to $s_0 = 1$. Initially, the window of temporal independence and the local clock are set: $t^A_{win} = t^B_{win} = clock_A = clock_B = 0$. In addition, the event queues associated with the output ports of models A and **B** are empty, i.e. $t^A_o = t^B_o = \infty$. For the circuit in Figure 4(a), the corresponding event prediction network is shown in Figure 4(b). For the pseudo components, the predicted event time values are set to 0, i.e. $W^A_{t_o} = W^B_{t_o} = W^A_{h_o} = W^B_{h_o} = 0$. Corresponding to a change in a predicted event time value in Figure 4(b), the value is propagated towards other pseudo components to the right, and a chain reaction is initiated. In this section, the symbol "$\rightarrow$" represents the propagation of a predicted event time value. The event with the earliest assertion time at an input i of a model C is represented through t^C_i. Thus, for the event prediction network in Figure 4(b),

$\infty \rightarrow W^A_{t_2}$ $\quad t^A_1 \rightarrow W^A_{t_1}$ $\quad W^A_{t_o} \rightarrow W^B_{t_2}$ $\quad t^B_1 \rightarrow W^B_{t_1}$

$W^B_{t_o} \rightarrow W^A_{h_2}$ $\quad t^A_1 \rightarrow W^A_{h_1}$ $\quad W^A_{h_o} \rightarrow W^B_{h_2}$ $\quad t^B_1 \rightarrow W^B_{h_1}$

The execution of P^2EDAS is organized through the following steps:

Step 1: Following the assertion of the external transitions at the primary inputs E_1 and E_2, the pseudo components in the event prediction network are initiated. New predicted event times (W values) are computed, utilizing equation (4), and are shown on Figure 5; the window of temporal independence values for models **A** and **B** are computed utilizing equation (5).

Step 2: The Input events to models **A** and **B** with assertion times less than 9ns and 10ns, respectively, may be executed. **A** and **B** may execute concurrently. Figure 6 describes the state of the subcircuit prior to execution. First, the high to low transition at time 0ns at the input of model **A** is executed. The execution of **A** generates an output event: $A_o = (0 \quad NAND \quad 1) \Longrightarrow 10 \uparrow$. Since the current logical value at **A** is 0, the output event queue of **A** will contain a low to high transition for time $t^A_o = 10$. Simultaneous with the execution of **A**, the high to low transition at the input of B is executed. The execution of **B** generates an output event: $B_o = (0 \quad NAND \quad 0) \Longrightarrow 14 \uparrow$. Thus, the newly generated event is a low to high transition at $t^B_o = 14$. Since the current logical value at the output of model **B** is already high, the generated event is deleted. The output event queue of **B** is empty and t^B_o is reset to ∞.

The processed input transitions are deleted. For the input event queue of **A**, $E_1 = 5 \uparrow$ defines the subsequent transition. For the input event queue of B, $E_2 = 7 \uparrow$ defines the subsequent transition. Both of these transitions may be executed, concurrently, since their times are defined within the respective t_{win} values. The execution of **A** generates an output event: $A_o = (1 \quad NAND \quad 0) \Longrightarrow 10 \downarrow$. The previously generated event, stored in the output event queue of **A**, is thus rendered inconsistent. The newly generated output event of **A** is also discarded since its logical value is indistinguishable from the current logical value at the output of **A**. Therefore, the output event queue of **A** is empty and t^A_o is set to ∞. The execution of **B** generates an output event: $B_o = (1 \quad NAND \quad 0) \Longrightarrow 17 \uparrow$ which is also indistinguishable from the current logical value at the output of **B**. This event is deleted and t^B_o remains at ∞.

Step 3: The processed input transitions are deleted. For the input event queue of A, $E_1 = 15 \uparrow$ defines the subsequent transition. For the input event queue of B, $E_2 = 16 \uparrow$ defines the subsequent transition. Neither of these transitions may be executed since they exceed the respective t_{win} values. For further execution of the models, the event prediction network must execute and update the t_{win} values. The state of the subcircuit is shown in Figure 7.

Step 4: The event prediction network is initiated which utilizes the fact that the transitions $E_1 = 15 \uparrow$ and $E_2 = 16 \uparrow$ are asserted at the respective input ports of the models, i.e., $t^A_1 = 15 \rightarrow W^A_{t_1}, W^A_{h_1}$ $\quad t^B_1 = 16 \rightarrow W^B_{t_1}, W^B_{h_1}$. The execution of the event prediction network is shown in Figure 8.

In the event prediction network, the predicted event times, $W^A_{t_o}$, $W^B_{t_o}$, $W^A_{h_o}$, and $W^B_{h_o}$ are recomputed, as shown on Figure 8. As a result, the window of temporal independence values for models **A** and **B** are computed utilizing equation (5) and also shown on Figure 8. Thus, $t^A_{win} = 21 \rightarrow$ **A** *and* $t^B_{win} = 25 \rightarrow$ **B**.

Owing to space constraints were unable to show all steps of execution. It is easy however, to follow the algorithm discussed and eventually show that the output waveforms of the circuit will be as shown in figure 9.

4 Conclusions

This paper has presented an asynchronous, distributed discrete event simulation algorithm for behavior simulation, P^2EDAS, that permits the use of any number of non-zero propagation delays for every path

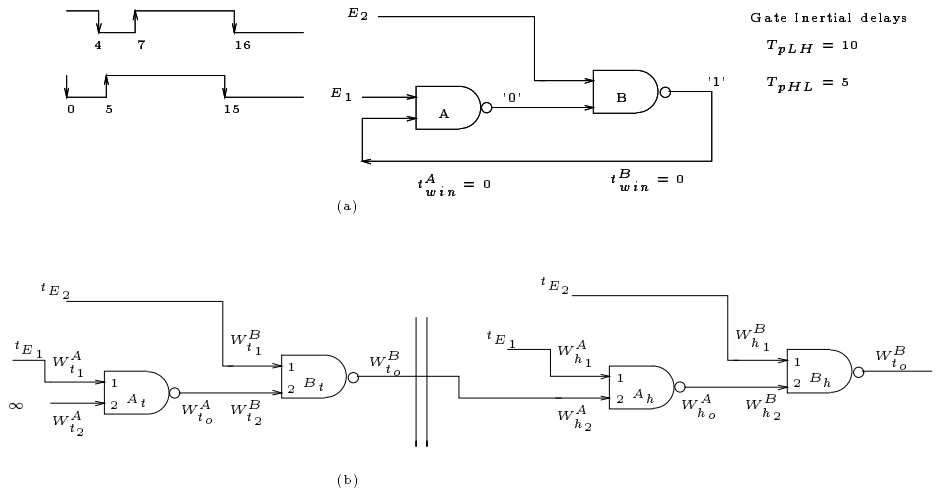

Figure 4: Example Circuit and Event Prediction Network

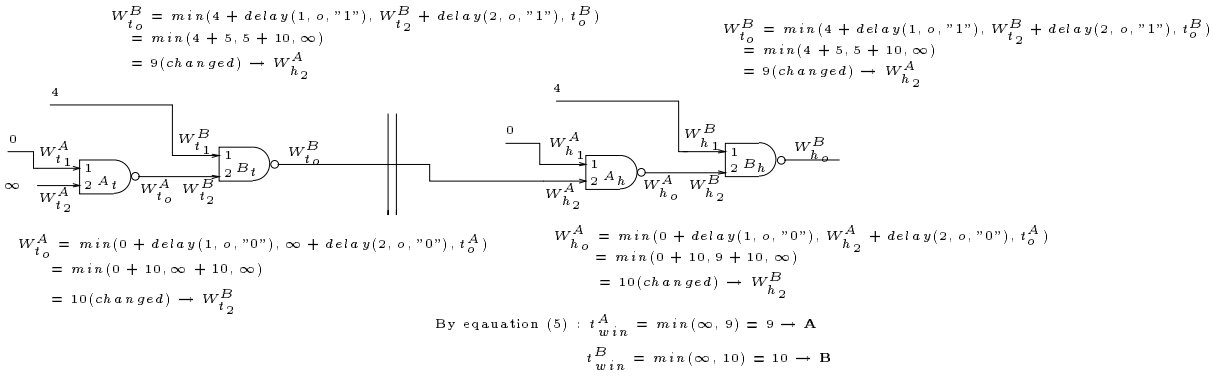

Figure 5: [Step 1] Execution of pseudo components

between the input and output of every hardware entity, while allowing detection and preemption of inconsistent events. P^2EDAS is capable of concurrently executing VHDL models and hardware descriptions in any hardware description language, on parallel processors. This paper has presented a detailed description of the algorithm. An implementation of P^2EDAS on a network of workstations, configured as a loosely-coupled parallel processors, is currently under progress.

References

[1] The Institute of Electrical and Electronic Engineers, IEEE Standard VHDL Language Reference Manual, ANSI/IEEE Std 1076-1993, IEEE, New York, NY, April 14, 1994.

[2] R. DeVries, "Reducing null messages in Misra's distributed discrete event simulation method," IEEE Transactions on Software Engineering, Vol. 16, No. 1, January 1990, pp. 82-91.

[3] D. A. Reed and A. Malony, "Parallel discrete event simulation: The Chandy-Misra Approach", Proceedings of the SCS Multiconference on Distributed Simulation, 3-5 February 1988, San Diego, California, pp.8-13.

[4] E. Debenedictis, S. Ghosh, and M.-L. Yu, "An Asynchronous Distributed Discrete Event Simulation Algorithm for Cyclic Circuits using Data-flow Network," IEEE Computer, Vol. 24, No. 6, June 1991, pp. 21-33.

[5] S. Ghosh and M.-L. Yu, "A Preemptive Scheduling Mechanism for Accurate Behavioral Simulation of Digital Designs," IEEE Transactions on Computers, Vol. 38, No. 11, November 1989, pp. 1595-1600.

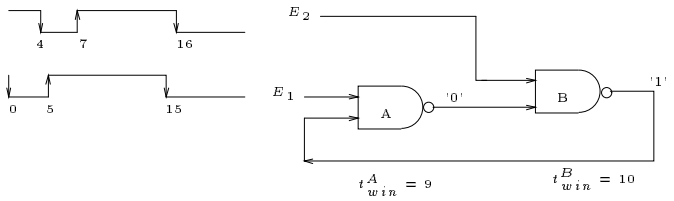

Figure 6: [Step 2] Models **A** and **B** receives new window of temporal independence values

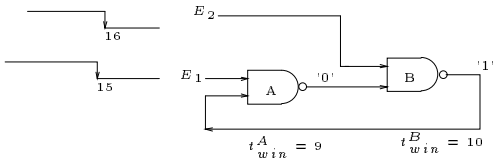

Figure 7: [Step 3] State of subcircuit following parallel execution of models **A** and B.

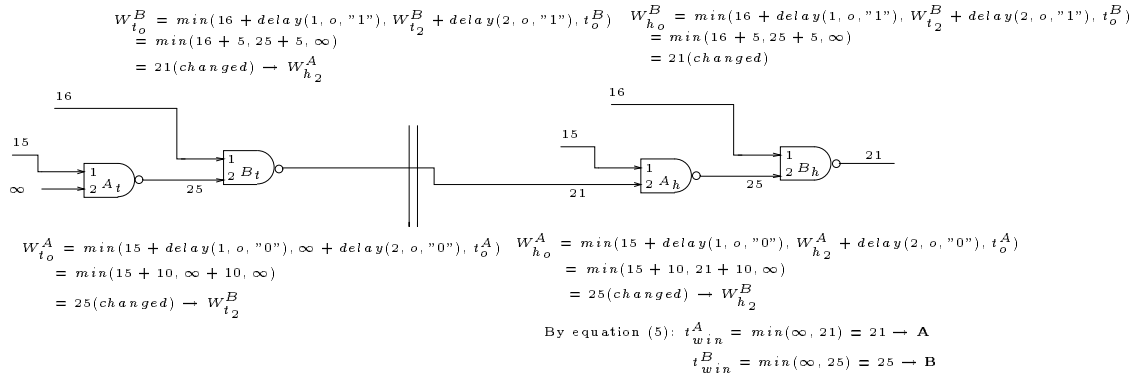

Figure 8: [Step 4] Execution of the event prediction network

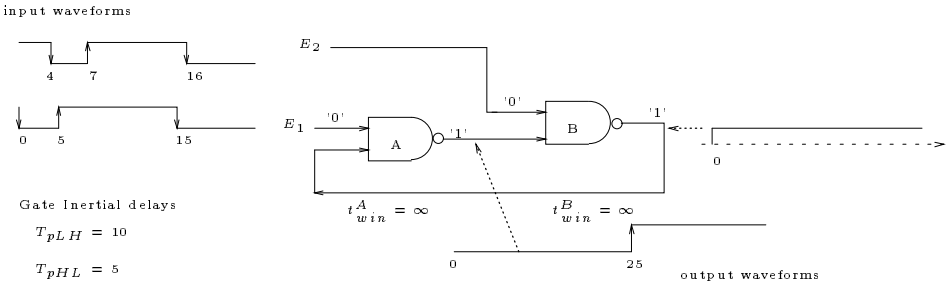

Figure 9: Final state and input/output waveform generated by subcircuit execution