ON THE PROPERTIES OF EVENT ORIENTED LOGIC SIMULATION ACCORDING TO SIGNIFICANT TIMING MODELS

por

FLÁVIO RECH WAGNER

RT nº 022 CPGCC-UFRGS SET/85

Trabalho desenvolvido com o apoio do CNPq

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Av. Osvaldo Aranha, 99
Caixa Postal 1501
90.000 - Porto Alegre - RS - Brasil
Telex (051) 2680 Tel. (0512) 21.8499

Simulação lógica

Estrutura: Dados

ABSTRACT

This report discusses properties of event oriented logic simulators that employ the technique of selective searching the active gates. Properties of interest are mainly related to the data structures needed by the simulator and associated operations. The report considers four different timing models for explaining the behavior of gates.

KEYWORDS: logic simulation, timing models, data structures.

RESUMO

Este relatório discute propriedades de simuladores lógicos orientados a eventos, que utilizam a técnica de procura seletiva das portas lógicas ativas. Propriedades de interesse estão principalmente relacionadas com as estruturas de dados necessárias no simulador e as operações associadas. O relatório considera quatro diferentes modelos temporais para explicar o comportamento das portas lógicas.

PALAVRAS-CHAVE: simulação lógica, modelos temporais, estruturas de dados.

SUMMARY

1. INTRODUCTION

In event oriented logic simulation signal transitions are modeled as events. An event notice for each event is stored in a list [WA 84]. Considering that to each gate in the network is associated a positive and finite delay, a transition at the gate inputs will possibly cause an output event for this gate after this delay. In general the list contains events scheduled for many future times. When the simulated time is advanced, signals must be updated according to the events scheduled in the list for this new current time. Event oriented logic simulators use in general the selective trace technique [ST 75]. This technique allows that at each simulated time only those gates are evaluated, whose outputs can potentially change, so that a more efficient simulation is achieved. The objective of this paper is to determine, for a set of significant timing models employed in logic simulation, which are the possible operations to be performed on the event list, which are the properties of this list and these operations, and how these properties and operations affect the selective trace technique. Four timing models are considered: two-valued logic with nominal delay; three-valued logic with min-max delay; inertial delay; and differing delays according to the transition direction. It will not be discussed here neither the validity nor the application of these models. It shall be proved that, besides the search and removal of the next event to be processed (i.e., the event notice in the list with the minimum schedule time), only two operations are needed: insertion of an unique new event notice and cancellation of an unique event notice (necessarily the event notice with greatest schedule time associated with the gate in question). The consequences of these properties for a logic simulator which uses selective trace are then considered.

2. TIMING MODELS

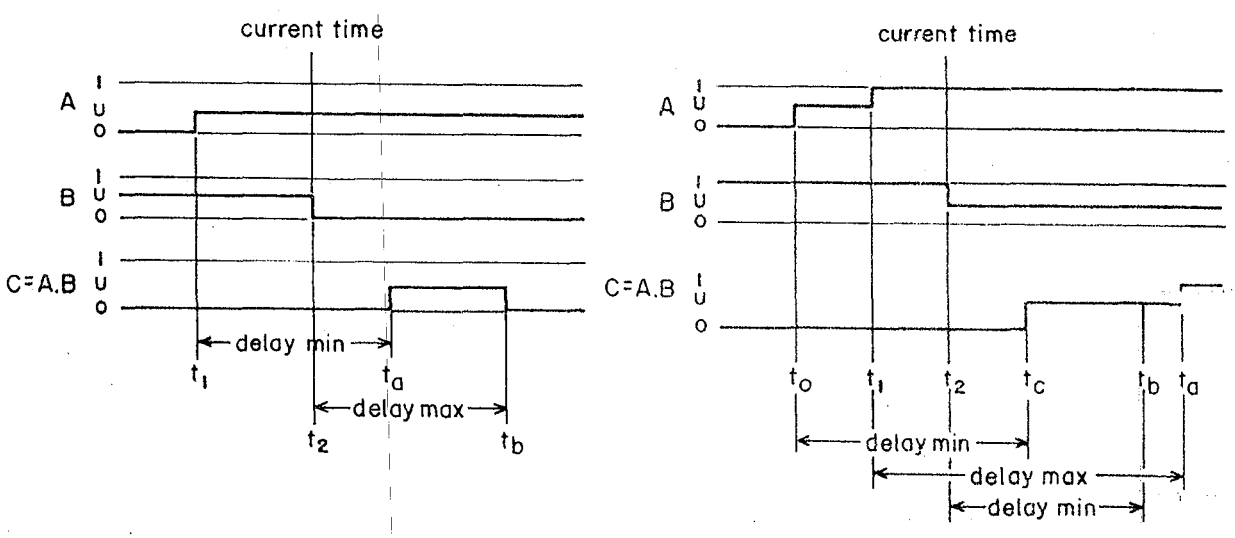
2.1 Two-Valued Logic with Nominal Delay

In this model a gate behaves like a pure function followed by a pure delay line, which doesn't introduce distortion. Any number of sinal transitions can be simultaneously travelling through this line, so that any number of future events can be scheduled for a gate at each time. A new event for the gate will have always a greater scheduling time (equal to the current simulated time plus the nominal delay for the gate) than all other events yet scheduled for this gate. Since only two logic values are present, the event notice doesn't need to contain the new logic value, because this will be surely the complement of the previous value at the moment of the event occurrence. The event notice contains only the gate name and the scheduled time for the event.

2.2 Three-Valued Logic with Min-Max Delay

In this model every signal makes a transition from o to 1 or from 1 to 0 going always through the intermediate state U ("unknown"). There is a dominance hierarchy [BF 76] between the logic values: U dominates 0 and 1, and these are in turn hierarchically equivalent. If there is an event from a logic value "a" to a logic value "b", then: 1) if "b" dominates "a", we must schedule the event to the earliest possible time (i.e., current time + minimum delay); 2) if "b" is dominated by "a", we must schedule the event to the latest possible time (i.e., current time + maximum delay). Using this rule the simulation always gives the worst possible case, that is, the signal will remain in the state U as long as possible due to the gate input conditions.

As in the previous model any number of future events can be scheduled for a gate at each moment. However, as a corollary of Theorem 5 of Eichelberger [EI 65], the gate output always goes through a sequence "non-dominating value → dominating value → non-dominating value" (i.e., 0 → U → 1 or 1 → U → 0) if the gate inputs also go through such a sequence. Let us define the gate inputs "at rest" when all of them have values 0 or 1 (that is, they have a non-dominating value). Let us say that at t=t1 an input event for a gate occurs. As we see in Fig. 1a, if at t=t2 > t1 a new input event appears, and the gate inputs didn't are at rest, then we have necessarily ta= t1+mindelay and tb (time for the new output event) = t2 + maxdelay (the input must have done at t2 a "dominating → non-dominating" transition), and so tb > ta, that is, the new output event has a greater scheduling time than all other events scheduled for this gate.



(a) no future event must be cancelled (b) a future event at t_a must be cancelled

Figure 1 - Possible Timing Relationships between Future Events in the Min-Max Delay Model.

If the gate inputs were at rest at t2, as in Fig. 1b, then ta = t1 + maxdelay and tb = t2 + mindelay. It can happen that tb < ta, that is, the new event can have a

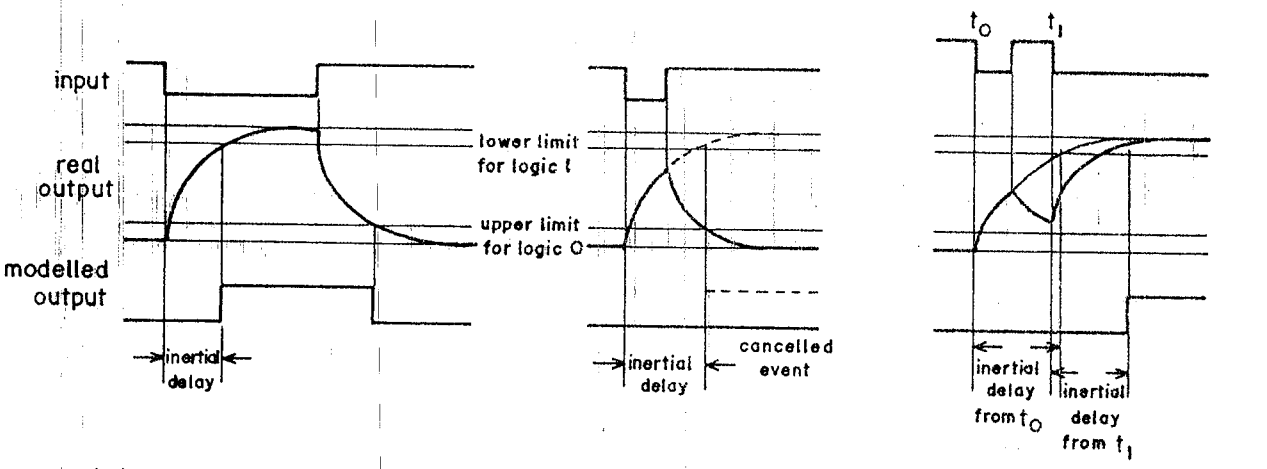
smaller scheduling time than the current event with greatest scheduling time for this gate. This later event must then be cancelled. However, this new event cannot have a smaller scheduling time that a second yet scheduled output event at tc < ta and due to an input transition at t0 < t2, because this second event must be of the type "non-dominating → dominating" transiton. We have then tc = t0 + mindelay, tb = t2 + mindelay, and so tc < tb.

We conclude that only a single event, and that with the greatest scheduled time among those scheduled for the gate, can be due to cancellation as a result of a new input transition, and that a new output event will always be the event with greatest scheduled time for the gate (after a possible event cancellation). Furthermore, after a cancellation, a new cancellation will never be needed before at least one new event is scheduled for the same gate.

2.3 Two-Valued Logic with Inertial Delay

Fig. 2 shows what really happens when an input transition occurs for a gate. The gate output capacitance is loaded (or discharged) according to an exponential curve. We say that the output reaches a logic value when this curve passes through a certain boundary. Another input transition occurring before the curve has reached the boundary can discharge (or load) the gate capacitance, so that the logic value is not really reached. We say then that the gate has an inertial delay, [BF 76] equal to the time interval the gate output needs to reach the opposite boundary. Within this interval the input values cannot be altered, if the modeled gate output really has to respond to the input transition. As algorithm which correctly handles this situation has to schedule an

output event, due to an input transition, for the "current time + inertial delay". If a new input transition occurs before the time scheduled for the output event is reached, and the output value calculated with the new input configuration is different from that predicted in the scheduled event (i.e., the new value is equal to the current value, in the case of two-valued logic), then this event must be cancelled, as in Fig. 2b.



(a) output reaches the other logic value (without initial charge)

(b) output doesn't reach the other logic value

(c) output reaches the other logic value (with initial charge)

Figure 2 - Timing Diagrams for an Inverter with Inertial Delay

A more exact algorithm would consider if the gate output capacitance has yet some charge due to previous input transitions. A smaller inertial delay has to be added to the current time at this new transition, as in Fig. 2c.

If this procedure is followed, and we have two--valued logic, then we will never have more than one future event scheduled for a gate. A reference for this event will be needed in the gate evaluation routine, because depending on new input transitions the event can be due to cancellation.

2.4 Two-Valued Logic with Different Delays for each Transition Direction

Modeling different delays for each transition direction can be considered with or without inertial delay. If we are considering inertial delay, different delays for each transition direction can be easily obtained if we have different curves for the charge and discharge of the gate output capacitance. We don't need to make any additional considerations about event scheduling and cancellation.

If we are modeling different delays for each transition direction without considering inertial delay, then Fig. 3 shows what could happen. An output event $0 \rightarrow 1$ is scheduled for td due to an input transition at ta=td-tplh. A later input transition at tb creates an input configuration that makes the output value equal to 0, and the corresponding output event should be scheduled at tc=tb+tphl. If tplh < tplh, it can happen that tc < td. Clearly the event at td must

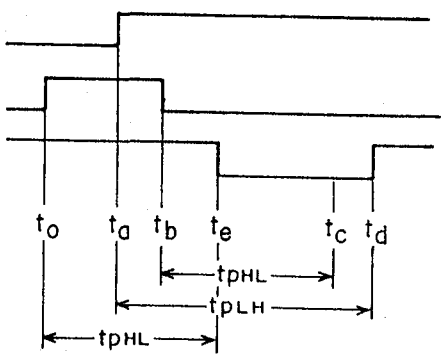


Figure 3.- Different Delays for each Transition Sense

be cancelled and the new event at tc doesn't need to be scheduled due to the selective trace. If another output event is scheduled for a time te < td, it can be easily proved that the time tc for the new event cannot be smaller than te. Times te and tc must both be calculated as the sum of the input transition time and tphl, so that only one event can be due to cancellation.

3. SELECTIVE TRACE AND THE FUTURE EVENTS

Selective Trace is a technique which evaluates at each simulation step only those gates which potentially can have an output transition, i.e., those which have an input transition at this time. As we see in Fig. 4, the existence of many future events must be considered when selective trace is to be applied. At t=0 the current value of C is 0 and the new calculated value is 1, so that an event is scheduled for t=10. At t=5, the current value of C is still 0 and the new calculated value is also 0, so that we could think that no future event would be needed due to this input transition. However, the new calculated value should be compared with the value just before t=15 (which is 1), not with the current value, because only at t=15 the new value will appear at the output, due to the delay. Then a future event must be scheduled at t=15. So, by using the selective trace technique we must always compare the new calculated value with the value stored in the event notice for this gate which has the greatest scheduled time.

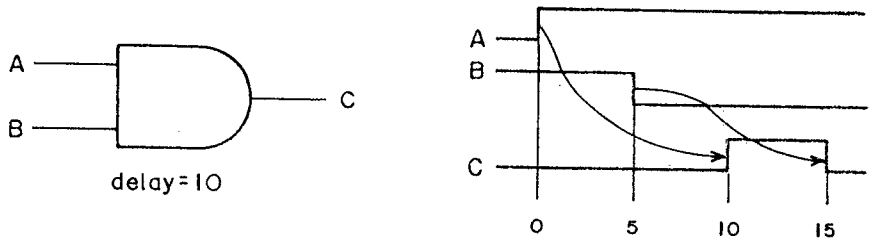


Figure 4. - Selective Trace: an Example of Comparison with a future Value.

4. CONSEQUENCES OF THE TIMING MODELS FOR THE SIMULATION ALGORITHM

We have deduced some properties about the event list and the operations on it, which are of course valid only for the timing models considered. They will be used now to derive the information which must be maintained by the simulation algorithm and how it must be used and updated when the basic operations on the event list are executed (execution, scheduling and cancellation of events), in order that the selective trace can be correctly applied. Two different cases must be considered, namely, simulation with or without inertial delay processing.

4.1 Simulaton with Inertial Delay Processing

This is the easiest case, because only one future event can be scheduled for each gate. The only information needed by the simulation algorithm, for each

gate, is a pointer to its unique scheduled event notice LEP (Latest-Event-Pointer). Be

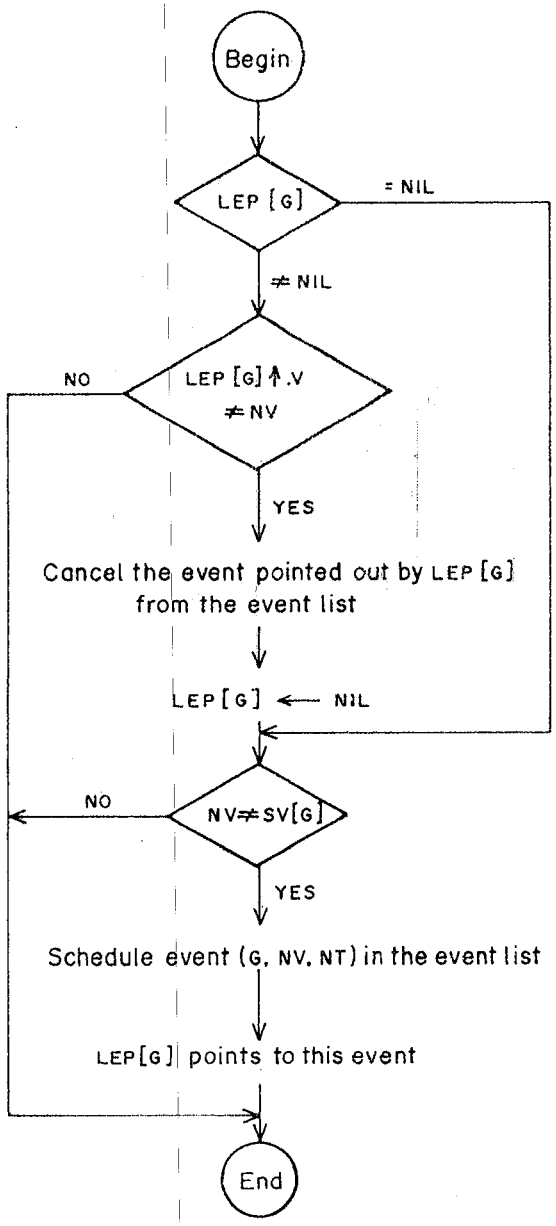


Figure 5.- Processing wich Inertial Delay: Actions after an Input Transition (Gate Evaluation Routine has given the values NV - New Logic Value - and NT - New Time -)

SV the current logic value at the gate output,

LEP↑.V the output logic value corresponding to the event notice pointed out by LEP,

LEP↑.T the time at which the event notice pointed out by LEP is to be processed,

NV the new gate output value calculated as a result of an input transition, and

NT the calculated time (according to the timing model), at which the value NV is to be assigned to the gate output.

When an event notice [gate G, value V, time T] is taken from the event list as the next to be processed, three actions must be performed by the simulation algorithm:

1. SV(G):= V

2. LEP(G):= NIL, because this was certainly the only event scheduled for this gate

3. Remove the event notice from the list.

When a gate has an input transition, and its function accordingly gives NV and NT, the actions shown in Fig. 5 must be executed.

4.2 Simulation without Inertial Delay Processing

In this case, at some point t1 a gate can have many future events for $t_i, \ldots t_j > t_1$, so that more information is needed. In addition to LEP, the following variables will be used:

COUNT, a count of the number of events scheduled for each gate at some moment during the simulation run;

LV, the logic value stored in the latest event notice for

each gate;

LT, the time at which the latest event notice for each gate is to be processed; and

VLV, the logic value stored in the latest but one event notice for each gate.

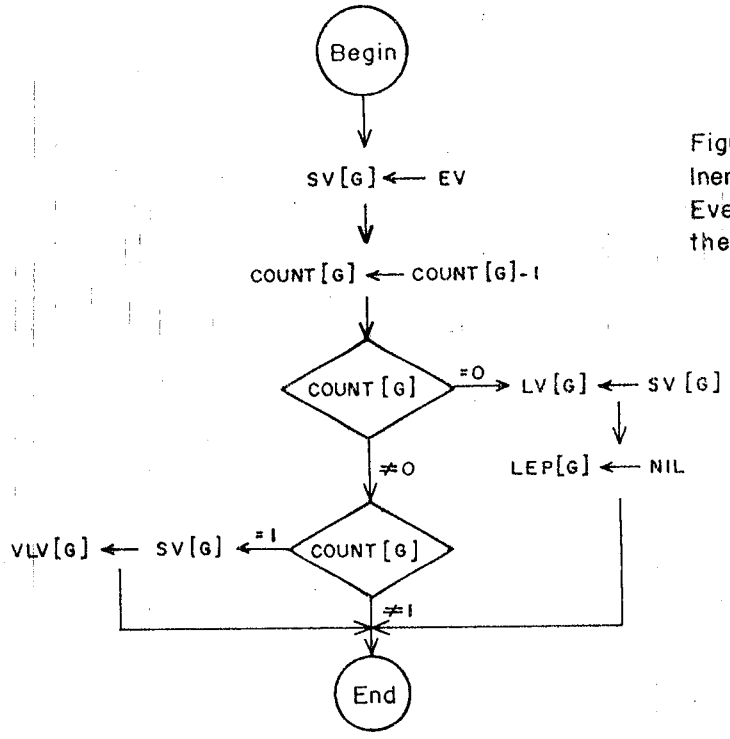


Figure 6.- Processing without Inertial Delay: Actions after an Event (G, EV, ET) was taken from the Event List to be processed.

Figures 6 and 7 show respectively the actions to be taken when an event notice has to be processed and after the gate function has evaluated new values NV and NT. The pointer LEP can assume the value NIL in two situations: 1) when there is really no event scheduled for the gate, and 2) when the latest event was cancelled, there are other events for the gate, but a new event was not yet scheduled for it since the cancellation. As we remember, two consecutive event cancellations for the same gate never occur, so that the knowledge of the latest event is not necessarily needed after a cancellation, until another scheduling is done.

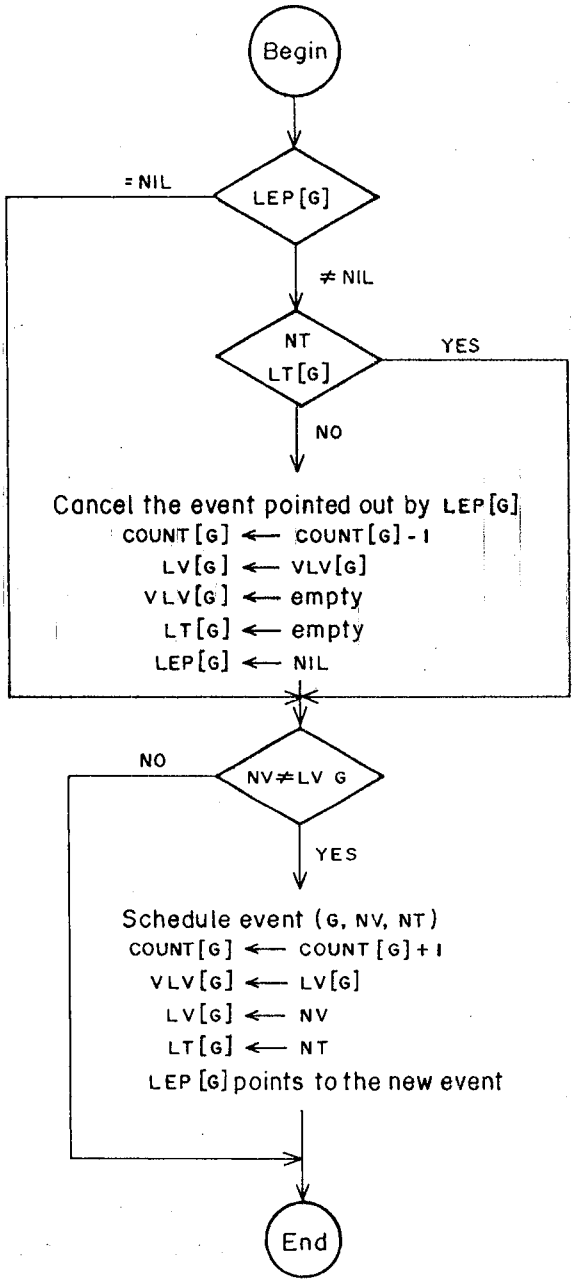


Figure 7.- Processing without Inertial Delay: Actions after an Input Transition (Gate Evaluation Routine has given the values NV- New Logic Value- and NT - New Time -)

Because of this possibility the logic value and the time corresponding to the latest event cannot be accessed always through LEP, and must be maintained in another variables. LV and VLV are initialized with the same logic value assigned to the gate output at the beginning of the simulation (i.e., SV). Always when there is only

one event scheduled for the gate, VLV has again the current gate output logic value SV. If there is no event scheduled for the gate, both VLV and LV have again this value SV.

5. CONCLUSIONS

We have shown that very simple algorithms, which use a very simple data structure besides the event list, namely a pointer to the latest event notice associated with each gate and some few variables, are needed in order that an event oriented logic simulator correctly processes four different significant timing models and implements the selective trace technique. The organization of the event list was not considered in this paper, and it can be deduced from the conclusions that it has no influence over the presented models and algorithms.

REFERENCES

[BF 76] BREUER, M.A. and A.D. FRIEDMAN. Diagnosis & Reliable Design of Digital Systems, Computer Science Press Inc., California, 1976, p.199

[EI 65] EICHELBERGER, E.B. "Hazard Detection in Combinational and Sequential Switching Circuits", IBM Journal of Research & Development, vol. 9, March 1965, pp 90-99

[ST 75] SZYGENDA, S.A. and E.W. THOMPSON. "Digital Logic Simulation in a Time-Based, Table--Driven Environment, part 1: Design Verification", Computer, Vol. 8, March 1975, pp 24-36

[WA 84] WAGNER, F.R. "Basic Techniques of Gate Level Simulation: A Tutorial". Porto Alegre, CPGCC-UFRGS, Out. 1984. (Relatório Técnico nº 012)

RELATÓRIOS TÉCNICOS MAIS RECENTES

RT-002: TELICHEVESKY, R., ICHIARA, A., AZEREDO, D. F. G., COITINHO, C. C.
MINUANO - Um Simulador de Logica.
Porto Alegre, CPGCC/UFRGS, dez/82.

RT-003: PRONDZYNSKI, P. R., FREITAS, C. M. D. S., TODESCO, A. R. W.
Sistema de Exibicao Grafica: utilizacao dos dispositivos.
Porto Alegre, CPGCC/UFRGS, nov/83.

RT-004: LASCHUK, A. PRONDZYNSKI, P. R., CALAZANS, N. L. V., ALBECHE, K. S.
Sistema de Exibicao Grafica: extensao do barramento para o HP2100.
Porto Alegre, CPGCC/UFRGS, out/83.

RT-005: PRONDZYNSKI, P. r.
Arquiteturas para videos graficos com varredura fixa.
Porto alegre, CPGCC/UFRGS, nov/83.

RT-006: WAGNER, F. R.
Modelamento de processos digitais com redes de instancias
Porto Alegre, CPGCC/UFRGS, mar/84.

RT-007: COSTA, A. C. R.
Caracterizacao dos conhecimentos e da arquitetura de um sistema especialista em projeto logico de circuitos digitais.
Porto Alegre, CPGCC/UFRGS, jun/84.

RT-008: FREITAS, C. M. D. S.
Programacao grafica interativa com o PGE/UFRGS.
Porto Alegre, CPGCC/UFRGS, jun/84.

RT-009: FREITAS, C. M. D. S.
Descricao do pacote grafico PGE/UFRGS.
Porto Alegre, CPGCC/UFRGS, jul/84.

RT-010: SAYAO, M. & TOSCANI, S. S.
Sistema multiprogramavel HP2100S - manual de referencia.
Porto Alegre, CPGCC/UFRGS, out/84.

RT-011: SAYAO, M. & TOSCANI, S. S.
Sistema multiprogramavel HP2100S - manual de usuario.
Porto Alegre, CPGCC/UFRGS, out/84.

RT-012: WAGNER, F. R.
Basic techniques for gate level simulation - a tutorial.
Porto Alegre, CPGCC/UFRGS, out/84.

RT-013: WAGNER, F. R.
Hazard detection in logic simulation.
Porto Alegre, CPGCC/UFRGS, nov/84.

RT-014: COSTA, A. C. R.
Clause machines.
Porto Alegre, CPGCC/UFRGS, jan/85.

RT-015: COSTA, A. C. R.
Especificacao das tarefas do sistema especialista em projeto logico de circuitos digitais.
Porto Alegre, CPGCC/UFRGS, jan/85.

RT-016: TOSCANI, L. V. & outros.
Laboratorio de Matematica Computacional - manual do usuario.
Porto Alegre, CPGCC/UFRGS, dez/84.

RT-017: TOSCANI, L. V. & outros.
Laboratorio de Matematica Computacional - manual de programas.
Porto Alegre, CPGCC/UFRGS, dez/84.

RT-018: COSTA, A. C. R.
Introducao aos sistemas especialistas e descricao informal do sistema muÖRóâÖéçTóR.
Porto Alegre, CPGCC/UFRGS, mai/85.

RT-019: FRIEDRICH, L. F. & COSTA, A. C. R.
Descricao da implementacao do montador MC68000 escrito em Pascal Sequencial.
Porto Alegre, CPGCC/UFRGS, jun/85.

RT-020: COSTA, A. C. R.
Processando linguagens naturais em PROLOG - Parte 1: Formalismo gramatical basico.
Porto Alegre, CPGCC/UFRGS, jul/85.

RT-021: WAGNER, F. R.
Algoritmos de simulacao de hardware no nivel RT.
Porto Alegre, CPGCC/UFRGS, set/85.