85/608

108188-0

HAZARD DETECTION IN LOGIC SIMULATION

Flávio Rech Wagner

RT nº 013 CPGCC-UFRGS Novembro 1984

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Av. Osvaldo Aranha, 99 Fone:(0512) 21.8499
Caixa Postal 1501 - Telex (051) 2680
90.000-PORTO ALEGRE-RS-BRASIL

ABSTRACT

This paper shows the influence of two factors on the hazards detection capability of a logic simulator: the delay model (zero, nominal or min-max) and the number of logic levels (2,3,5 or 8). Hazards are briefly defined. It is shown that a 2-valued, nominal delay model cannot detect hazards within a reasonable efficiency. It is also shown that a 3-valued, zero delay model (Eichelberger's algorithm) can detect hazards in their broadest sense, while definite hazard pulses in particular implementations need min-max delay models to be detected. Static hazards can be definetely detected only with 5-valued logic, while 8-valued logic is needed to detect dynamic hazards. It is assumed a previous knowledge of event-driven logic simulation by the reader /ST 75/ /WA 84/.

Key Words: Hazard detection, static hazards, dynamic hazards, logic simulation, multi-valued logic signals.

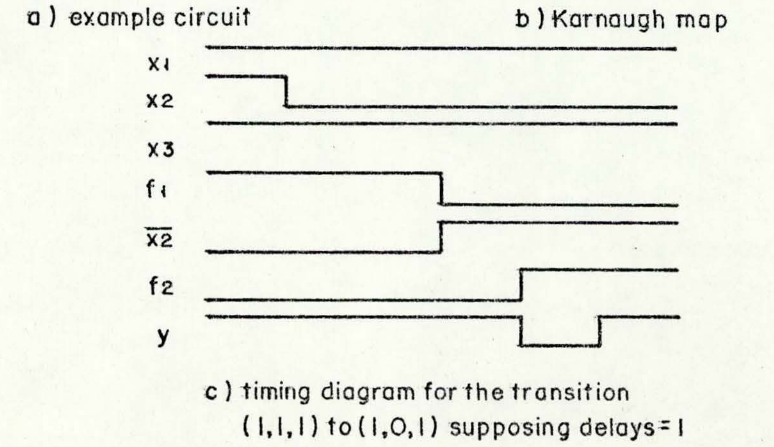
RESUMO

Este artigo mostra a influência de dois fatores na capacidade de deteção de hazards de um simulador lógico: o modelo de delay (zero, nominal ou min-max) e o número de níveis lógicos (2,3,5 ou 8). Hazards são definidos brevemente. É mostrado que um modelo de delay nominal com 2 níveis lógicos não pode detetar hazards com uma eficiência aceitável. É mostrado também que um modelo de delay zero com 3 níveis lógicos (algoritmo de Eichelberger) pode detetar hazards no sentido mais amplo da definição, enquanto pulsos definidos de hazards em implementações particulares necessitam modelos de delay min-max para serem detetados. Hazards estáticos so' podem ser definitivamente detetados com lógica de 5 valores, enquanto lógica de 8 valores é necessária para a deteção de hazards dinamicos. É presumido que o leitor tem um conhecimento prévio de simulação lógica dirigida por eventos /ST 75/ /WA 84/.

Palavras-Chave: Deteção de hazards, hazards estáticos, hazards dinâmicos, simulação lógica, sinais lógicos multivaluados.

1. DEFINITIONS

We will define hazards based on a Karnaugh map representation of the logic function. A static hazard exists when: 1) the circuit makes a transition from the present state to an adjacent state (i.e., there is a transition in an input signal); 2) the function value is the same in both states (i.e., there is no predicted output transition due to the input transition); 3) both states are not covered by a same function term. If these three conditions are present, it can happen that, due to the particular delays in a particular realization of the function, a spurious pulse appears at the output during the transition. We will call this spike a "hazard pulse". It can occur because the function term which maintains the output in the initial state can turn to 0 before the term which will maintain the output in the final state turns to 1. This short difference in time will be noted as a spurious output pulse. It must be noted that the hazard is a condition of the function implementation, and exists independently of particular delays, while the hazard pulse exists in a particular function realization, i.e., for specific delay values in the gates. Figure 1 shows an example.



a) example circuit

b) Karnaugh map

c) timing diagram for the transition (1,1,1) to (1,0,1) supposing delays=1

Figure 1 - Static hazard

A hazard pulse can always be eliminated by the insertion of appropriated delays in the circuit. The static hazard itself, however, can be eliminated if we include in the function implementation a redundant term which covers both the initial and final state, such that this term maintains the output during the transition. In the example of figure 1, the term $X1.\overline{X3}$ eliminates the hazard. This hazard elimination means that no hazard pulse can appear, no matter what values have the delays in the circuit.

M-hazards/EI65/ occur for simultaneous multiple input transitions. We define two types of M-hazards: function hazards and logic hazards. A function hazard exists when: 1) the circuit makes a transition from the present state to a non-adjacent state; 2) the function value is the same in both states; 3) for at least one of the possible intermediate states between the initial and the final state the functions has a different value. If these three conditions are present, it can happen that, due to the arriving order of the input transitions and to the particular delays in a particular realizations of the function, the circuit goes momentaneously through this intermediate state, such that a spurious pulse appears at the output. Figure 2 shows an example. This hazard cannot be eliminated: it is impossible to avoid the circuit to go through this intermediate state with different output value. Of course it is possible to avoid the hazard pulse in a particular realization by properly selecting the gate delays.

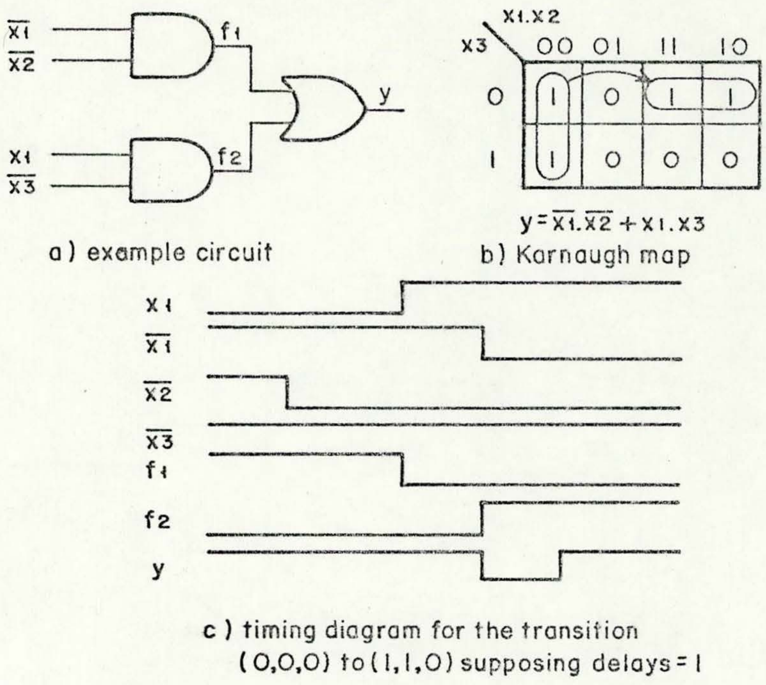


Figure 2 - Function hazard

A logic hazard exists when: 1) the circuit makes a transition from the present state to a non-adjacent state; 2) the function value is the same in both states; 3) for all possible intermediate states between the initial and final state the function has also the same value; 4) both initial and final states are not covered by a same function term. If these four conditions are present, it can happen that, due to the partic ular delays in the circuit, the term which maintains the output in the initial state turns to 0 before the term which will maintain the output in the final state turns to 1. Due to this time difference, a spurious pulse can appear at the output. It is easy to see that the logic hazard corresponds to the static hazard in the case of multiple input transition. As with the static hazards, logic hazards can be eliminated by the insertion of a redundant term in the function. Figure 3 shows an example. There is no function hazard in this case because for both possible intermediate states between states d and a (state b, if input z changes before w, and state c, if input w changes before z) the function output has also the value 1. State a is covered by term f1 and state d by term f4, such that a logic hazard exists. The inclusion of term x$\bar{y}$ eliminates this hazard.

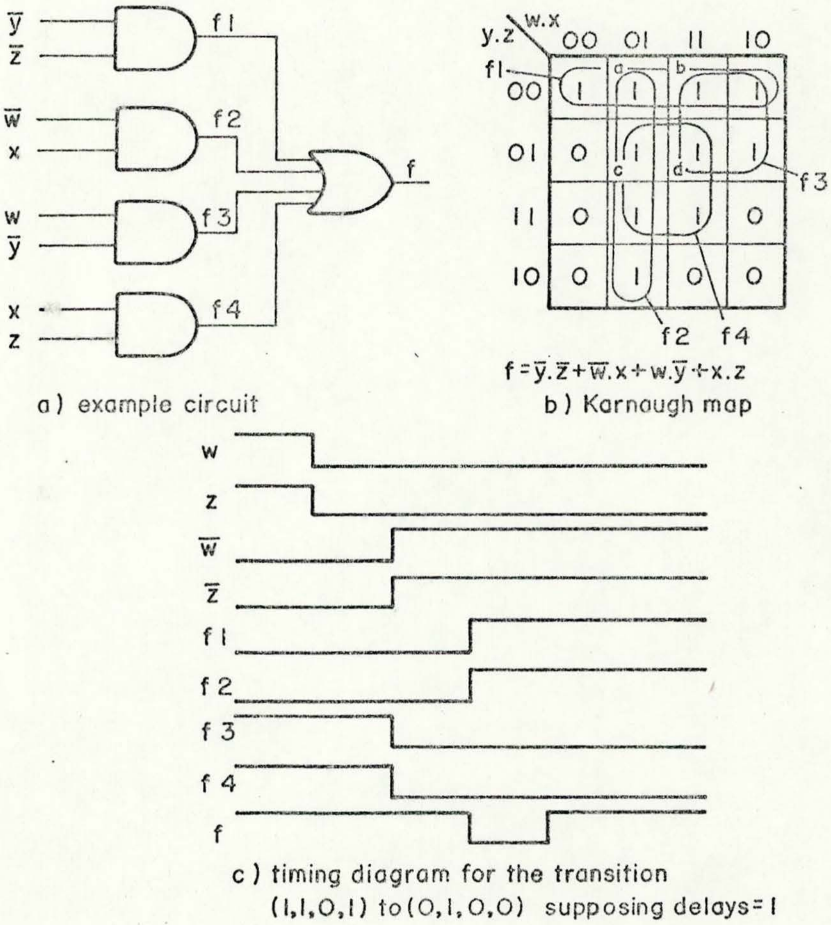

a) example circuit
b) Karnaugh map
f = $\bar{y}.\bar{z}+\bar{w}.x+w.\bar{y}+x.z$
c) timing diagram for the transition (1,1,0,1) to (0,1,0,0) supposing delays = 1

Figure 3 - Logic hazard

A dynamic hazard exists when the circuit has an input transition such that the function output has different values in the initial and final input states. The response is a single transition from the initial to the final output value. A dynamic hazard is the possibility that the circuit makes more than one transition (a minimum of three) before the output stabilizes in the final value. Figure 4 shows an example. Signal X1 has a static hazard as a result of the input transition propagation through some combinational circuit, while signal X2 makes a single transition from 0 to 1. Depending on X1 and on the time relationship between X1 and X2, the output signal can make a single transition from 0 to 1 or pass through a sequence 0-1-0-1. A dynamic hazard is always consequence of the combination of a static hazard with a single transition.

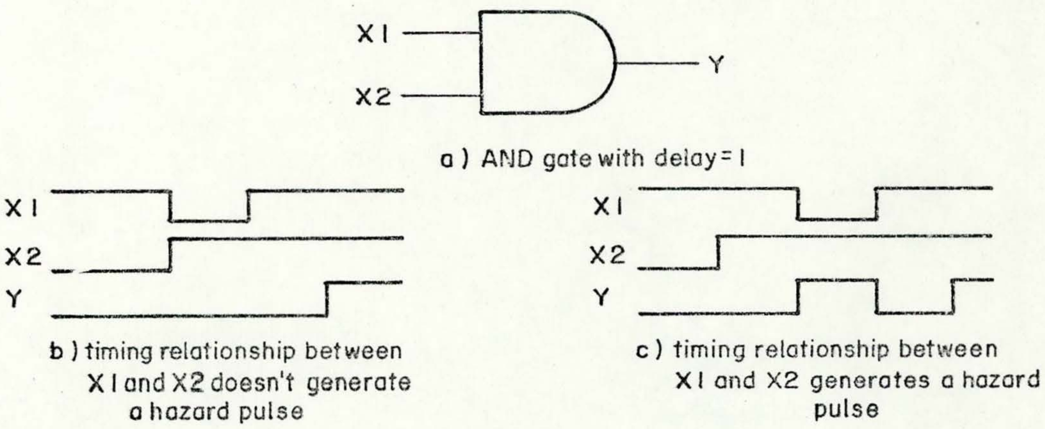
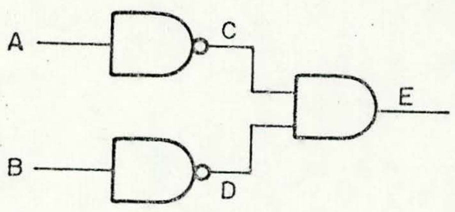

X1
X2
Y
a) AND gate with delay=1
X1
X2
Y
b) timing relationship between X1 and X2 doesn't generate a hazard pulse
X1
X2
Y
c) timing relationship between X1 and X2 generates a hazard pulse

Figure 4 - Dynamic hazard

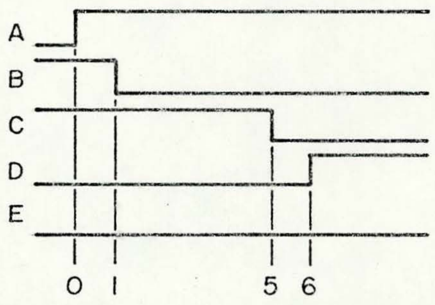
2. TWO-VALUED LOGIC AND NOMINAL DELAY SIMULATION

Let us suppose that we have a logic simulator with 2-valued logic and where a nominal delay can be specified for each gate in the circuit. The example of Figure 1 could be simulated with this tool. A delay = 1 would be assigned to each gate, and we could observe the hazard pulse in output Y. If we consider all AND gates with the same delay, no matter what value, the simulator would give us always a response with the hazard pulse: the lower path would be always longer than the upper path, due to the inverter (assuming the inverter with a non-zero delay). This is however a very specific case, in which a hazard exists because one path is certainly longer than the other. In general a hazard exists because one path can be longer than another, but this is not known before the circuit is realized.

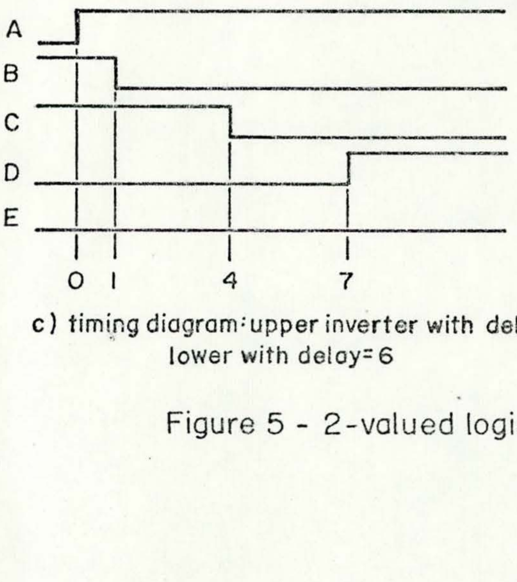
Figure 5 shows a part of a circuit. Let us suppose that the changes in A and B are originated from the same input transition, that propagated through different paths, such that the change in A occurs 1 time unit before that of signal B. Figures 5b to 5d show three timing diagrams for different values of delays in the inverters. Only for the third case a spike appears at the output. A hazard is predicted for a function only because we can consider the gates capable of having any possible delay value. Our simulator however can only consider concrete cases of delays, such that the hazard could or could not be detected in a specific simulation. As we cannot be sure of the delay values in the physical realization of the circuit, we would need Y^x simulations to definetely exclude the possibility of a hazard error, for a circuit with x gates, each with y possible delay values.



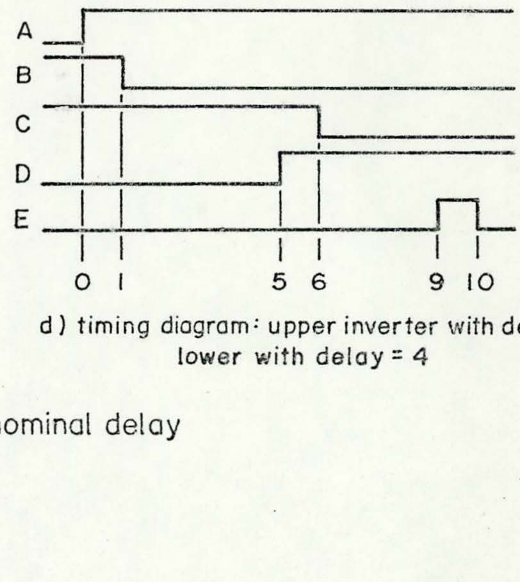
a) example circuit



b) timing diagram: both inverters with delay = 5



c) timing diagram: upper inverter with delay = 4, lower with delay = 6



d) timing diagram: upper inverter with delay = 6, lower with delay = 4

Figure 5 - 2-valued logic and nominal delay

3. THREE-VALUED LOGIC AND EICHELBERGER'S ALGORITHM

A three-valued logic function has three possible logic values instead of two. The third value "U" denotes an undefined value, which can be 0 or 1, and is used to represent a signal value during a transition. A three-valeud logic function can be obtained from a corresponding two-valued function in the following way: If the inputs of the 3-valued function are 0's and 1's, then the output value is the same as in the 2-valued function; if one of the inputs is U, then we take the two possible corresponding 2-valued input configurations; if for both we have the same 2-valued output value, this value is taken for the 3-valued function; if for them we have different 2-valued output values, then the 3-valued function output will be U. Figure 6 shows the 3-valued truth tables for the AND and OR logic functions.

AND	0	1	U
0	0	0	0
1	0	1	U
U	0	U	U

OR	0	1	U
0	0	1	U
1	1	1	1
U	U	1	U

Figure 6 - Truth tables for 3-valued logic

Eichelberger /EI65/ derived a method to detect hazards in combinational and sequential circuits, considering multiple input transitions. The method assumes no delays associated with the gates, such that it allows hazard detection according to the hazard definition, and not the detection of hazard pulses in a specific realization. For combinational circuits the method is very simple: 1) evaluate the function f(A) before the input transition: 2) assign to each changing input the value U and evaluate the function f(A/B) during the transition: 3) evaluate the function f(B) after the input transition. If f(A)=f(B)=f, and f(A/B)≠f, then the circuit has a hazard. Figure 7 shows an application of the method. Looking to the Karnaugh map in Figure 7e we see that in f1 we have a logic hazard, while in f2 we have a function hazard. The method doesn't allow a distinction between them.

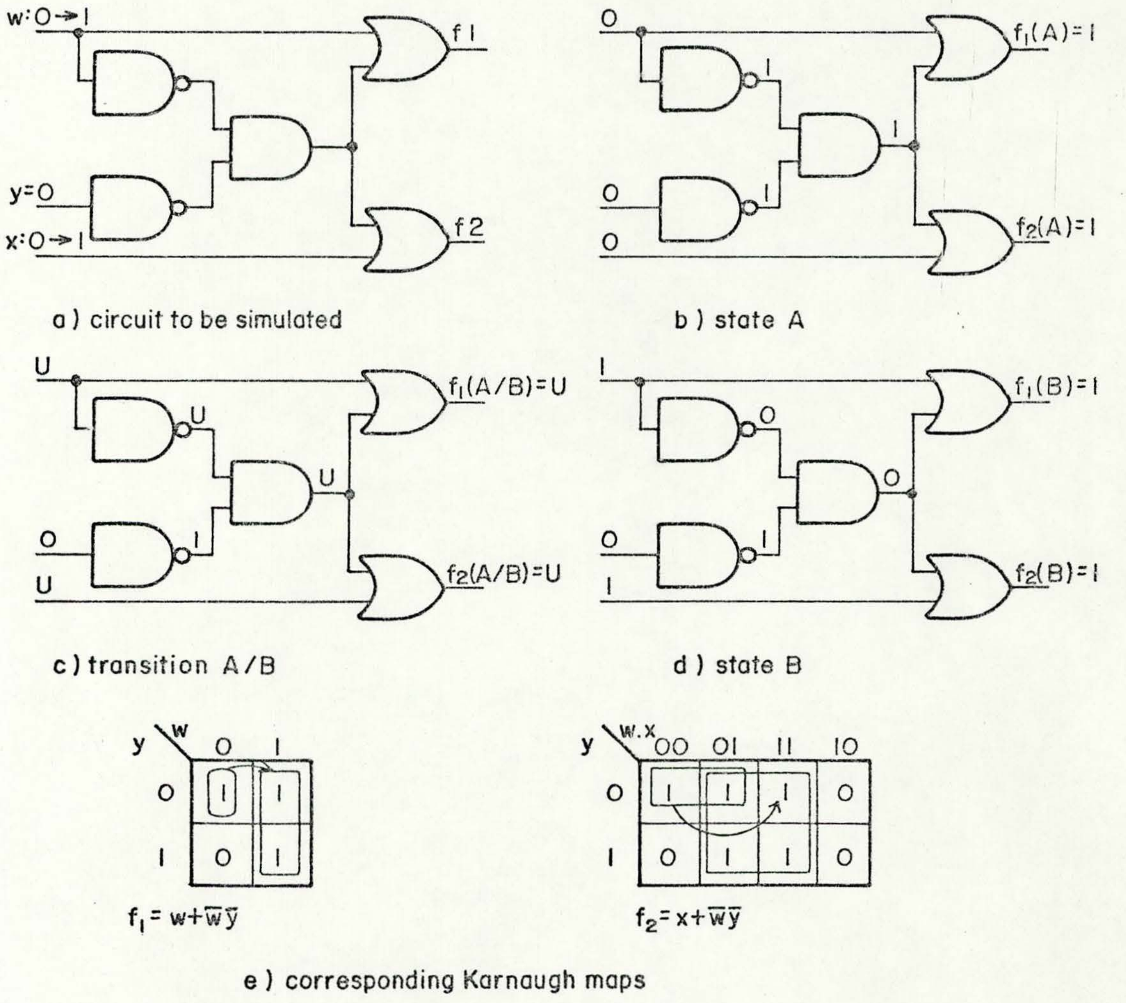


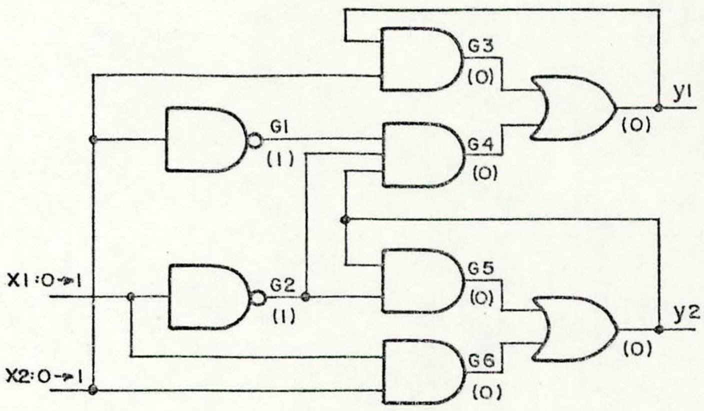
Figure 7 - Eichelberger's method for hazard detection in combinational circuits

For sequential circuits the method is a little more elaborated:

1) Procedure A - determine the feedback signals which can change due to the input transition: a) make each changing input

signal equal to U; b) evaluate the circuit until all feedback signals are stabilized (for each evaluated gate, if it has its output changed, put the gates connected with it in the A-List, but only if they have an output value different of U; Eichelberger has proved that every gate which output is yet U cannot be more changed in this pass).

2) Procedure B - determine the feedback sighals which stabilizes: a) make each changing input equal to its final value; b) evaluate the circuit until all feedback signals are stabi̲lized (for each evaluated gate, if it has its output chan̲ged, put the gates connected with it in the B-List, but only if they have an output value equal to U; Eichelberger has proved that every gate which output is yet 0 or 1 cannot be more changed in this pass). If after the procedure B a signal has stabilized in the value U, then a hazard is detected for it. Of course, if this is a feedback signal, then we have detected a critical race. Figure 8 shows an application of the method. Y1 and Y2 are the feedback signals. Procedure A is executed until the A-List is empty, what indicates that no more gates can change its output values in this pass. The result is Y1 = Y2 = U, i.e. both feedback signals can change due to the input transition. Procedure B is then executed until the B-List is empty. The result is Y1= U and Y2 = 1. This means that Y2 will certainly stabilize with value 1, while Y1 can stabilize with any value, depending on the relative delays of the circuit.



a) circuit to be simulated (indicated is the initial value of each signal)

A={ }		
X1 ← U, X2 ← U	A={G1, G2, G6}	
G1=$\overline{X2}$ ← U	A={G2, G6, G4}	
G2=$\overline{X1}$ ← U	A={G6, G4, G5}	(1)
G6=X1.X2 ← U	A={G4, G5, y2}	
G4=G1.G2.y2 ← O	A={G5, y2}	(2)
G5=y2.G2 ← O	A={y2}	
y2=G5+G6 ← U	A={G4, G5}	
G4=G1.G2.y2 ← U	A={G5, y1}	
G5=y2.G2 ← U	A={y1}	(3)
y1=G3+G4 ← U	A={G3}	
G3=X2.y1 ← U	A={ }	

b) Procedure A

B={ }		
X1 ← 1, X2 ← 1	B={G1, G2, G6}	
G1 ← $\overline{X2}$ ← O	B={G2, G6, G4}	
G2=$\overline{X1}$ ← O	B={G6, G4, G5}	(1)
G6=X1.X2 ← 1	B={G4, G5, y2}	
G4=G1.G2.y2 ← O	B={G5, y2, y1}	
G5=G2.y2 ← O	B={y2, y1}	(4)
y2=G5+G6 ← 1	B={y1}	(5)
y1=G3+G4 ← U	B={ }	(6)

c) Procedure B

Notes

(1) G4 is yet in the A(B)- list due to G1. it doesn't need to be inserted again.

(2) G4 has not changed, no gate comes to the A-list.

(3) y2 doesn't come to the A-list, because its output is yet U.

(4) y2 is yet in the B-list due to G6. It doesn't need to be inserted again.

(5) G4 and G5 don't come to the B-list, because their values are yet O.

(6) y1 has not changed, G3 doesn't come to the B-list.

Figure 8 - Eichelberger's method for hazard detection in sequential circuits

4. THREE-VALUED LOGIC AND MIN-MAX DELAY

The problem with Eichelberger's method is that it is too pessimistic. Although a function can have a hazard accord ing to the theoretical definition, a definite hazard pulse can never occur in a physical realization of this function. This is because the actual delays in the circuit cannot assume any values. Indeed they are restricted to a certain range determined by the technology. This means that a zero-delay simulation like that of Eichelberger (i.e., where no specific delay is assigned to the gates, so that the method in reality presumes that a gate can assume any delay value) is not a realistic approach. In Section 2 we have yet seen that assigning a nominal delay to each gate is also an unrealistic assumption, because before the circuit realization it is impossible to known the exact delays. Furthermore the nominal delay simulation is a very inefficient method of detecting hazards.

Assigning a range of delay values to each gate seems to be needed in order to obtain a realistic result. We associate this with 3-valued logic, and obtain a simulator which is capable in some extent to detect hazards, as we shall see. The delay range is specified for each gate through the minimum and maximum values. Every signal can only make a tran sition from 0 to 1 (or 1 to 0) if it passes through the intermediate value U. The simulation is of course event oriented We have four possible events for a signal: $0 \rightarrow U$, $1 \rightarrow U$, $U \rightarrow 1$, $U \rightarrow 0$. Now we have a problem: if an event is predicted for a signal, to what future time must this event be scheduled? I.e., what delay must we add to the present time to find the occurrence time of the event? Minimum or maximum? The problem is solved with the concept of dominance /BF76/. There is a dominance hierarchy between the logic values (we shall use this concept also for 5- and 8-valued logic). In the case of 3-valued logic, we say that the value U dominates 0 and 1, and that 0 and 1 are hierarchically equivalent. Is there is an event from a logic value "a" to a logic value "b", then: 1) if "b" dominates "a", we must make the event scheduling to the earliest possible time; 2) if "b" is dominated by "a", we must make the event scheduling to the latest possible time. Using this rule, we are sure that we take always the worst possible case. Fig. 9 shows the application of this method to the example of Fig. 5. If we allow primary inputs to go direct from 0 to 1 (or 1 to 0), it is needed a special treatment for those gates which receive these primary inputs: an input transition at the gate must cause the scheduling of two output events, one for the minimum and another for the maximum delay. The worst case result is clearly seen in the hazard predicted for signal E. The earliest combination which makes E=1 is C=1, D=1 at t=5, what causes E=1 at t=5+minimum=8 (U dominates 0). The latest combination which makes E=1 is C=1,D=1 at t=6, what causes E=1 at t=6+maximum=11 (1 is dominated by U).

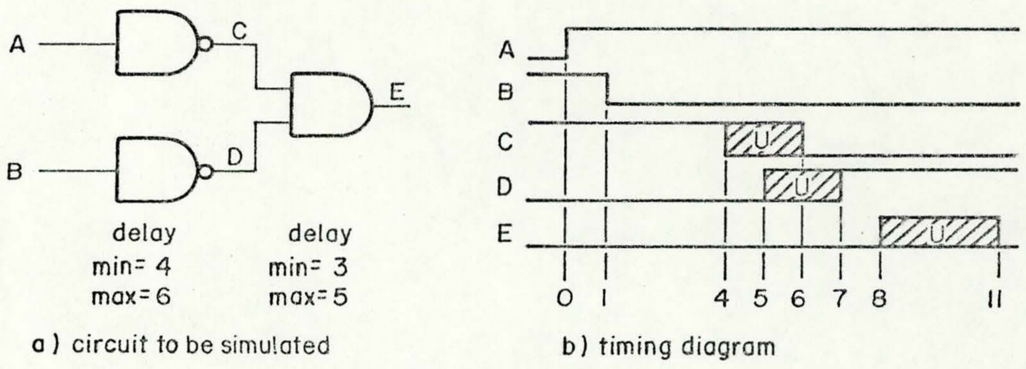
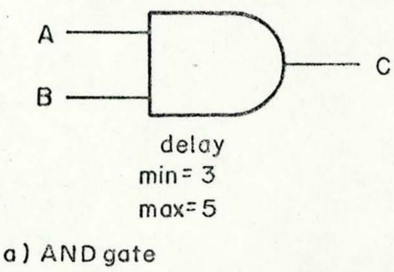


Figure 9 - 3-valued logic with min-max delay

Figure 10 shows another example. An U-interval is predicted for signal C. Here we see that an interpretation of the value U is needed. In the last example the U-interval of signal E was of course a hazard pulse, because the signal had the same value before and after the interval. In this case, signal C has different values before and after the interval, so that we must interpret the U value as a transition between 0 and 1. The interpretation is of course easy to make, due to the values out of the interval. Figure 11 shows a third example, where an U-interval is predicted for signal C as in Figure 10.

However it can be seen that the static hazard in signal A can cause a dynamic hazard in C, as in Figure 11c. The interpreta tion od signal C cannot give us however any idea about the existence of this hazard.



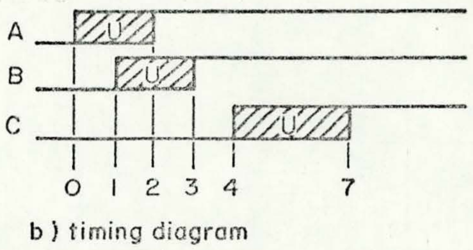


Figure 10 - A clean transition modeled by 3-valued logic with min-max delay

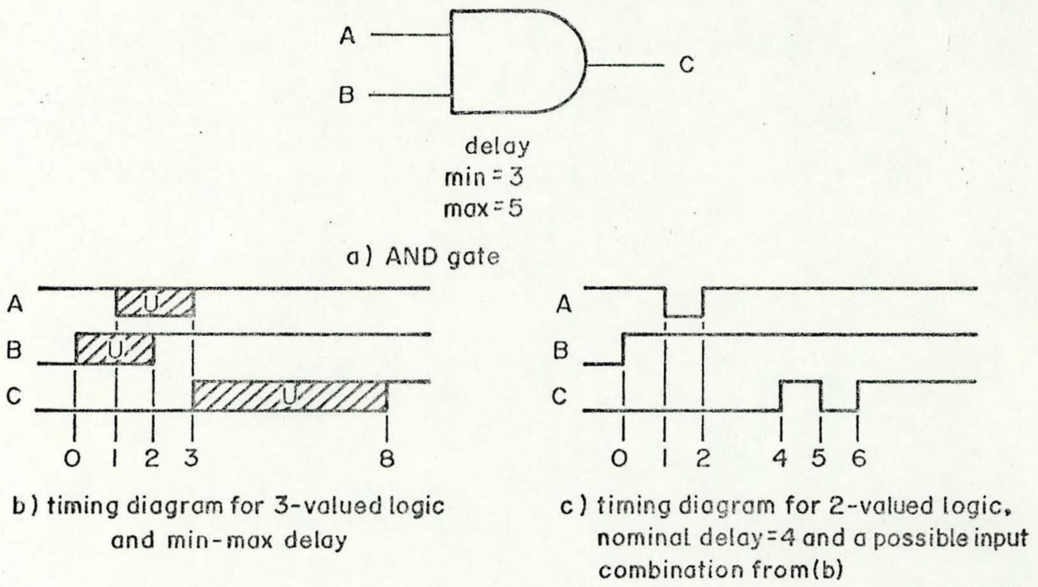
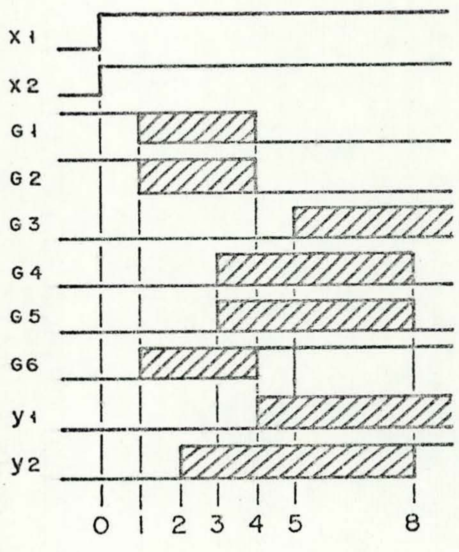

A
B
C
delay
min=3
max=5
a) AND gate
U
U
U
0 1 2 3 8
b) timing diagram for 3-valued logic and min-max delay
0 1 2 4 5 6
c) timing diagram for 2-valued logic, nominal delay=4 and a possible input combination from(b)

Figure 11 - A dynamic hazard modeled by 3-valued logic with min-max delay

Event oriented simulation can be applied to combinational as well as to sequential circuits, without any special considerations. Figures 12 and 13 correspond to the example of Figure 8, where the Eichelberger's method is applied to sequential circuits. In Figure 12 we assume delays min = 1 and max=4 for all gates, while in Figure 13 we have delays min=1 and max=2. It can be seen that in Figure 12 the hazard in signal y1 is detected as in Figure 8, while in Figure 13 y1 stabilizes with value 0. As mentioned, hazard pulses depend on the delay values, while the Eichelberger's method detects hazards, which are delay independent.

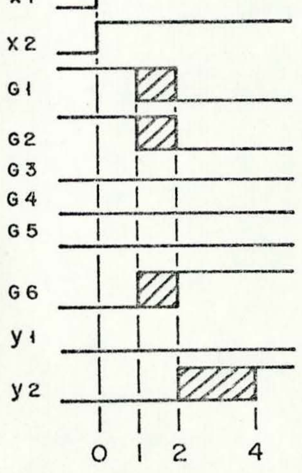


a) timing diagram

```
t=0  X1:0→1  ⇒ G2:1→U at t=0+1=1, G2: U→0 at t=0+4=4
     X2:0→1    G1:1→U at t=0+1=1, G1: U→0 at t=0+4=4
               G6:0→U at t=0+1=1, G6: U→1 at t=0+4=4
t=1  G1:1→U     without effect
     G2:1→U     without effect
     G6:0→U  ⇒  y2:0→U at t=1+1=2
t=2  y2:0→U  ⇒  G4:0→U at t=2+1=3
                G5:0→U at t=2+1=3
t=3  G4:0→U  ⇒  y1:0→U at t=3+1=4
     G5:0→U     without effect
t=4  G1:U→0  ⇒  G4:U→0 at t=4+4=8
     G2:U→0  ⇒  G5:U→0 at t=4+4=8
     G6:U→1  ⇒  y2:U→1 at t=4+4=8
     y1:0→U  ⇒  G3:0→U at t=4+1=5
t=5  G3:0→U     without effect
t=8  G4:U→0     without effect
     G5:U→0     without effect
     y2:U→1     without effect
```

b) event sequencing and scheduling

Figure 12 - Same example of Fig.8, now with delays min=1 and max=4



a) timing diagram

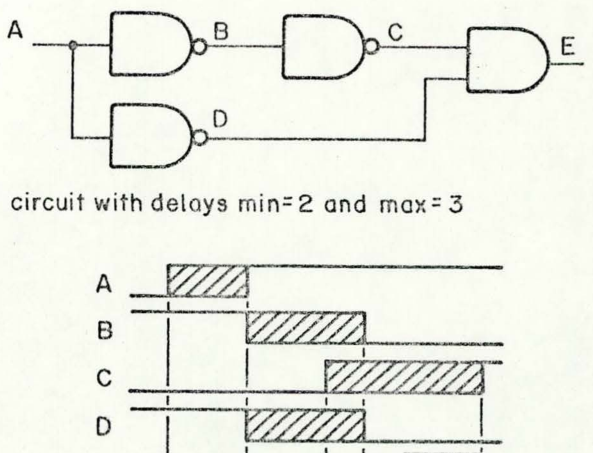
```
t=0  X1:0→1  ⇒ G2:1→U at t=0+1=1
     X2:0→1    G2:U→0 at t=0+2=2
               G1:1→U at t=0+1=1
               G1:U→0 at t=0+2=2
               G6:0→U at t=0+1=1
               G6:U→1 at t=0+2=2
t=1  G1:1→U     without effect
     G2:1→U     without effect
     G6:0→U  ⇒  y2:0→U at t=1+1=2
t=2  y2:0→U                      | signal changes occur
     G1:U→0  ⇒  without effect   | simultaneously, the simulator
                                 | first actualizes all variables,
     G2:U→0                      | later verifies the effect
     G6:U→1  ⇒  y2:U→1 at t=2+2=4
t=4  y2:U→1     without effect
```

b) event sequencing and scheduling

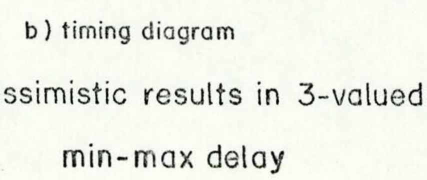
Figure 13 - Same example of Fig.8, now with delays min=1 and max=2

We conclude about 3-valued logic with min-max delay: 1) it allows detection of static hazards, but this detection can be made only by interpretation of the timing diagrams, and not by a direct information (as we shall see with 5- and 8-valued logic); 2) it doesn't allow the detection of dynamic hazards.

Multiple-valued logic gives always pessimistic results. In extreme cases, it can predict a hazard where this is impossible to occur. Figure 14 shows an example. If the input transition occurs at t=0, signal C can go to 1 earliest at t=4. If the input transition occurs at t=2, signal D can go to 0 latest at t=5. The interval from t=4 to t=5 is thus predicted with C=D=U, i.e., it is possible C=D=1, what causes a hazard pulse at E between t=6 and t=8. This prediction is made assuming that the transition occurs at t=0 for the upper path and at t=2 for the lower path, what of course cannot occur.



a) circuit with delays min=2 and max=3



b) timing diagram

Figure 14 - Pessimistic results in 3-valued simulation, min-max delay

5. FIVE-VALUED LOGIC

It was shown that 3-valued logic cannot detect dynamic hazards because the value U has two possible interpretations: a normal transition or a hazard. To avoid this problem and allow dynamic hazard detection it is needed a differentiation between these two cases, what is done in 5-valued logic. The value U remains to indicate a hazard, while two new values are introduced to represent normal transitions. We will represent them by ↑ and ↓. Figure 15 shows the truth table for the AND function. Since we maintain the min-max delay model, the concept of dominance is needed to determine the scheduling time of events. We have: U dominates ↑ and ↓ , which are hierarchically equivalent; they in turn dominate 0 and 1, which are again hierarchically equivalent.

AND	0	1	↑	↓	U
0	0	0	0	0	0
1	0	1	↑	↓	U
↑	0	↑	↑	U	U
↓	0	↓	U	↓	U
U	0	U	U	U	U

Figure 15 - Truth table for the 5-valued AND function

Figures 16 to 18 show the same examples from Figures 9 to 11, but now for 5-valued logic. It can be seen that: 1) a static hazard is predicted as an U-interval, like in Figure 16; 2) a transition without dynamic hazard is predicted as a ↑ (or ↓), as in Figure 17; 3) a transition with dynamic hazard is predicted as an U-interval, like in Figure 18. We conclude that a hazard (static or dynamic) will be always predicted as an U-interval, while a transition without hazard will be predicted as a ↑ or a ↓ .

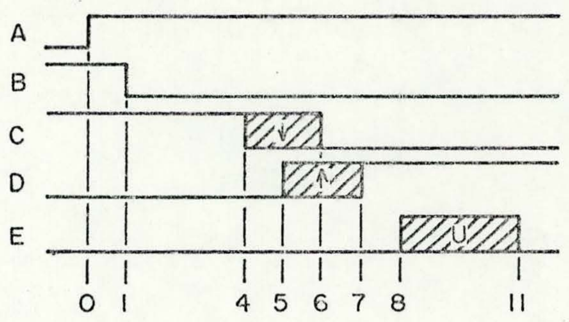


Figure 16 - Same example of Fig. 9, now for 5-valued logic

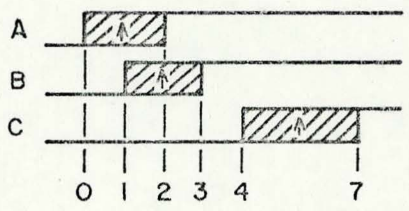


Figure 17 - Same example of Fig. 10, now for 5-valued logic

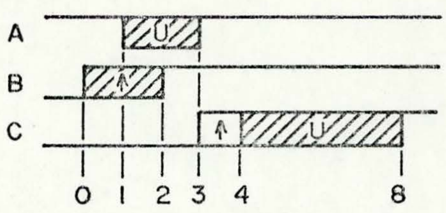


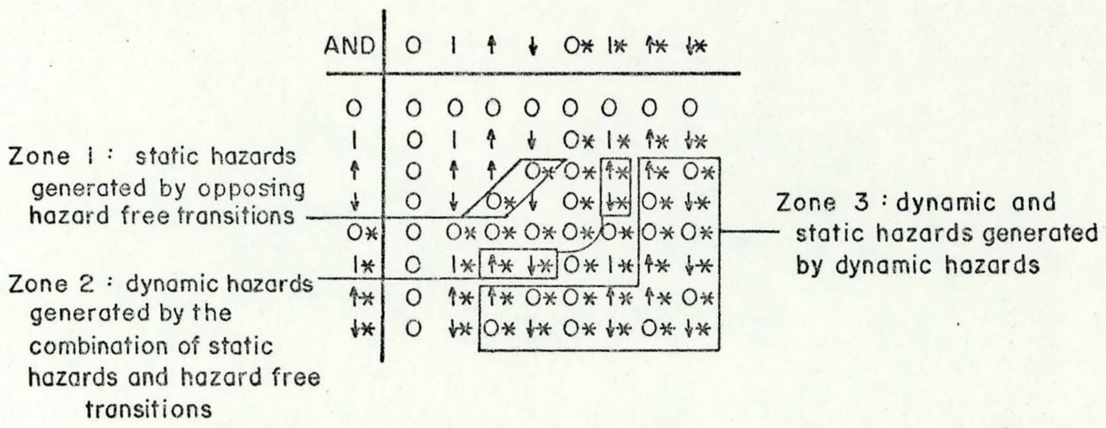
Figure 18 - Same example of Fig. 11, now for 5-valued logic

6. EIGHT-VALUED LOGIC

Although 5-valued logic allows the detection of both static and dynamic hazards, the distinction between them is not automatic, but must be made by interpretation of the timing diagrams: a static hazard is recognized when the signal has the same value before and after the U-interval, while a dynamic hazard exists when these values are different. An automatic distinction can be obtained only if we introduce different logic values for the two cases. The table below gives the 8 possible values in 8-valued logic. Besides the values 0,1, ↑ and ↓ , four values are introduced to distinguish between situations, which were all represented by U in the 5-valued logic: 0* for a static zero hazard, 1* for a static one hazard, ↑* for a 0 a 1 transition with dynamic hazard, and ↓* for a 1 to 0 transition with dynamic hazard.

1	static one value
0	static zero value
↑	hazard free 0 to 1 transition
↓	hazard free 1 to 0 transition
0*	static zero hazard
1*	static one hazard
↑*	0 to 1 transition with dynamic hazard
↓*	1 to 0 transition with dynamic hazard

Figure 19 gives the 8-valued truth table of an AND function. Three zones are marked in the table: in the first one we have static hazards being generated by hazard free transitions in opposite directions; in the second one we have dynamic hazards being generated by the combination of static hazards with hazard free transitions; in the third one we see that dynamic hazards can generate static and other dynamic hazards.



AND	0	1	↑	↓	0*	1*	↑*	↓*
0	0	0	0	0	0	0	0	0
1	0	1	↑	↓	0*	1*	↑*	↓*
↑	0	↑	↑	0*	0*	↑*	↑*	0*
↓	0	↓	0*	↓	0*	↓*	0*	↓*
0*	0	0*	0*	0*	0*	0*	0*	0*
1*	0	1*	↑*	↓*	0*	1*	↑*	↓*
↑*	0	↑*	↑*	0*	0*	↑*	↑*	0*
↓*	0	↓*	0*	↓*	0*	↓*	0*	↓*

Zone 1 : static hazards generated by opposing hazard free transitions

Zone 2 : dynamic hazards generated by the combination of static hazards and hazard free transitions

Zone 3 : dynamic and static hazards generated by dynamic hazards

Figure 19 - Truth table for the 8-valued AND function

Since we maintain min-max delay, dominance is again needed to solve the scheduling problem. The scheme below gives the dominance relationship between all logic values (values in the same line are hierarchically equivalent).

↑ growing dominance	↑*	↓*
	0*	1*
	↑	↓
	0	1

Figure 20 gives some examples for an AND gate, supposed with delays min=3 and max=5. It can be seen that for many cases output sequences are generated, which must be carefully interpreted. For example: a ↑ between t=3 and t=4 followed by a ↑* between t=4 and t=7, and followed by a 1* between t=7 and t=8. This means that a 0 to 1 transition will certainly occur between t=3 and t=7, but not necessarily between t=3 and t=4. This transition will be possibly followed by a negative hazard pulse between t=4 and t=8, but not necessarily between t=7 and t=8. We could say that the dynamic hazard region absorbs the intervals in its immediate vicinity, because these intervals give redundant information in relation to that given by the dynamic hazard interval. Using these ideas of redundance and absorption, we can derive the following rules:

for static hazards	0* absorbs an earlier ↑
	0* absorbs a later ↓
	1* absorbs an earlier ↓
	1* absorbs a later ↑
for dynamic hazards	↑* absorbs earlier ↑ and 0*
	↑* absorbs later ↑ and 1*
	↓* absorbs earlier ↓ and 1*
	↓* absorbs later ↓ and 0*

Figure 21 clarifies these rules, showing each hazard case and the redundant information which can be predicted by the simulation before and after the hazard intervals.

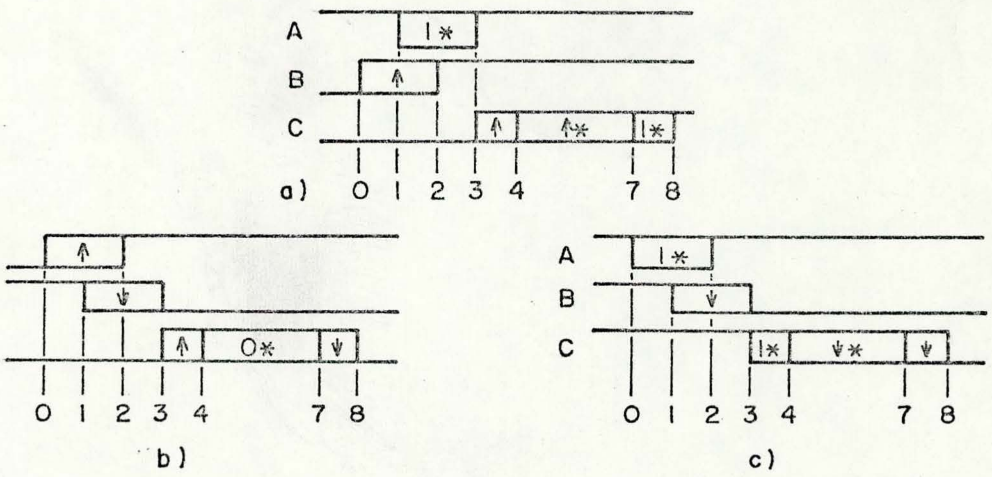


Figure 20 - Some timing diagrams for and AND gate, with delays min=3, max=5, evaluated with 8-valued logic

timing diagram	code	possible redundant information			
		before	code	after	code
_|‾|_	0*	_|‾	↑	‾|_	↓
‾|_|‾	1*	‾|_	↓	_|‾	↑
		_|‾	↑	_|‾	↑
_|‾|_|‾	↑*	_|‾|_	0*	‾|_|‾	1*
		‾|_	↓	‾|_	↓
‾|_|‾|_	↓*	‾|_|‾	1*	_|‾|_	0*

Figure 21 - Redundance in 8-valued simulation results

Acknowledgment

This work was done in 1982 during a 3 years stay at the University of Kaiserslautern,Germany. I appreciated the support of the University of Kaiserslautern and of the CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico.

REFERENCES

/BF 76/ Breuer, M.A. and A.D.Friedman, Diagnosis & Reliable Design of Digital Systems, Computer Science Press, Inc., California, 1976.

/EI 65/ Eichelberger, E.B., "Hazard Detection in Combinational and Sequential Switching Circuits", IBM Journal of Research & Development, Vol.9, No. 2, Mar 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, No. 3, Mar 1975, pp 24-36

/WA 84/ Wagner, F.R., "Basic Techniques of Gate Level simulation - A Tutorial", Internal Report No. 012, Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Nov 1984