Practical approach to asynchronous gate networks

Prof. J.A. Brzozowski, Ph.D., and Prof. M. Yoeli, D.Sc.

Indexing terms: Asynchronous sequential logic, Logic design, Logic gates

Abstract

In analysing sequential gate networks one must take into account the propagation delay of each gate. In conventional methods, state variables are associated with feedback loops, and a flow table based on these variables is constructed. This flow table may be incorrect if hazards are present, and an additional complex analysis is required to obtain the correct behaviour. In this paper, we describe a model in which a state variable is associated with each gate. This leads to a conceptually simple one-pass analysis procedure. The model is applied to the analysis of complex commercial flip-flops.

1 Introduction

This paper is concerned with the analysis of practical asynchronous sequential gate networks. Frequently, methods associating state variables with feedback loops are used for the analysis of such networks. This in itself is not correct, as we demonstrate in Section 2, because the method effectively takes into account the delays of some of the gates only. In the conventional theory, one must also analyse the network for various types of hazards.[6] The effect of each hazard must then be examined. The entire analysis is rather complicated and is not mathematically precise. Often the hazard analysis is treated as a 2nd-order effect and the primary stress is placed on races among the feedback variables. This is a misleading point of view because hazards may cause critical races in the overall network.

In this paper, we propose an alternative model in which we associate a state variable with each gate. This has the effect of taking into account all the gate delays uniformally, and avoids the need for a multipass analysis based on various hazards. We demonstrate the feasibility of the model by analysing an edge-sensitive *JK* flip-flop.

2 Conventional analysis

A large number of books and papers on the theory of asynchronous gate networks use the following analysis method. Given the logic diagram of a gate network, choose a certain set of signals to constitute the state variables; namely, choose a set of lines such that cutting each line removes all feedback loops in the network. We will call the set of corresponding variables a *feedback* set. For example, in Fig. 1*a* all the feedback sets are $\{v\}$, $\{w\}$ and $\{v, w\}$. Usually, minimal sets are used; in this case, one would choose either $\{v\}$ or $\{w\}$. If the line corresponding to w is cut as in Fig. 1*b*, there remains a combinational network. If we distinguish between the signals on the two ends of the cut line in Fig. 1*b* by calling them W and w, we can think of w as a new input, W as a new output, and we find

$$W = \{(wx)'x\}' = w + x'$$

The feedback variable w provides sufficient information to determine all the signals in the network of Fig. 1*b* (provided the input is also known), and $\{w\}$ is indeed an 'internal-state' variable set for Fig. 1*a*. Note that $v = (wx)'$.

The behaviour of a sequential network cannot be explained properly without taking into account the propagation delays of the gates. We assume here that line delays are negligible. In the conventional theory one delay is introduced for each line in the chosen feedback set. In our example we get Fig. 1*c*. This procedure is clearly equivalent to assuming that gate G_1 has no delay whereas gate G_2 has delay Δ. This assumption is not justified if we want a general analysis of Fig. 1*a*. In reality, every gate has a delay, and it is likely that both gates of Fig. 1*a* will have delays of similar magnitude. We now proceed to show that the approach of Fig. 1*c* yields misleading results.

In the conventional theory the network behaviour is explained with the aid of the excitation table, as in Fig. 2*a*, where we assume that v and w are the outputs. We show the outputs only for stable total states, for simplicity.

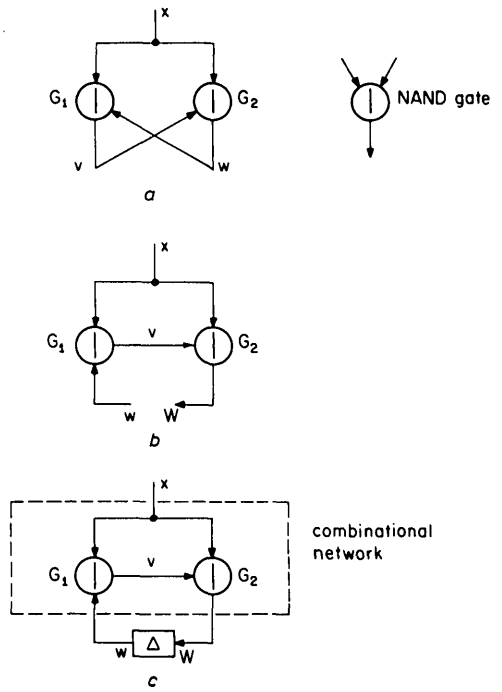

Fig. 1
Conventional analysis

(*a*) Network N_1 (*b*) Cutting a feedback line (*c*) Inserting a delay

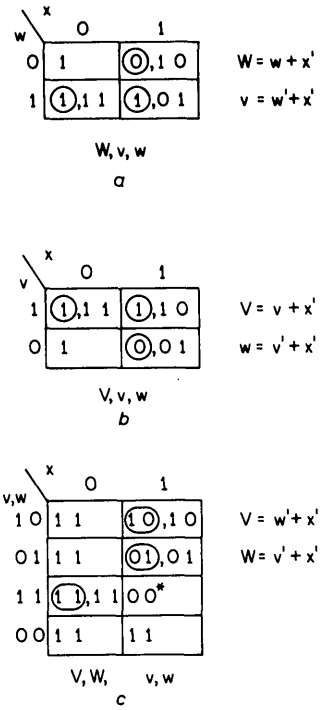

Fig. 2
Excitation tables for N_1:

(*a*) Using $\{w\}$ (*b*) Using $\{v\}$ (*c*) Using $\{v, w\}$

From Fig. 2*a* we find that, if N_1 is started with $x = 0$, it is forced to state $w = 1$ with $v = w = 1$. Next, when x changes arbitrarily w remains fixed at $w = 1$.

Paper 7659 E, first received 17th February and in revised form 19th December 1975

Prof. Brzozowski is with the Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Prof. Yoeli is with the Department of Computer Science, Technion, Israel Institute of Technology, Haifa, Israel

A similar analysis with $\{v\}$ as the feedback set gives Fig. 2*b*. Here $x = 0$ forces N_1 to $v = 1$, again with $v = w = 1$. However, as x changes subsequently, it is clear that $w = x'$.

We started with one network N_1, analysed it using two different sets of feedback variables, and obtained two different input/output behaviours. Obviously, the two analyses are not equivalent. In fact, both are generally incorrect. More realistically, we should associate a propagation delay with each gate, as mentioned earlier. When this is done we obtain Fig. 2*c*. Obviously we get the same stable states as before, but the transitions are quite different. In fact, when x has been 0 and then changes to 1, there is a critical race, indicated by *.

Discrepancies such as these between Fig. 2*a* and Fig. 2*b* are explained in the conventional theory with the aid of hazards. As an example, consider the network of Fig. 1*b*. When $w = 1, x = 0$, we have $W = 1$. Also, when $w = 1, x = 1$, we have $v = 0$ and $W = 1$. Thus, when x changes from 0 to 1, the output W should remain constant at $W = 1$. One easily verifies, however, that the network producing W has a '0-hazard'[6] during this transition. This is then interpreted as a warning that the corresponding transition in the flow table of Fig. 2*a* (from total state $x = 0, w = 1$ to total state $x = 1$, $w = 1$) may be incorrect.

In summary, in the conventional theory one first performs an analysis using a minimal set of feedback variables, and must then check whether that analysis is applicable by performing a complex hazard analysis. In contrast to this, the analysis using one variable per gate requires no further verification, thus avoiding the necessity of a complicated analysis of hazard phenomena and their effects. Also our method is conceptually very simple and mathematically precise.

3 Realistic analysis of gate networks

In our model we assign a state variable with each gate output. We then obtain a flow table, similar to the conventional one, but one which correctly describes the entire network behaviour. To achieve this we also introduce mathematically precise models of races (see Section 4).

Note that line delays may be represented in our model by 1-input, 1-output identity gates. However, for simplicity, we neglect line delays in our examples.

In using the 'one-variable-per-gate' model, one encounters several difficulties due to the size of practical networks, such as commercial flip-flops which typically consist of 8 - 16 gates.[4,5] First, excitation-table methods are completely out of the question for 2^{16} gate-states. However, the number of stable states is usually very small. Secondly, the number of inputs to such flip-flops is typically five: Again $2^5 = 32$ columns are undesirable. Often, sets of equivalent columns can be treated all at once, as we will show below. Thirdly, the computation of transitions can be rather tedious. We can reduce the amount of computation by (*a*) computing only those transitions that are necessary, (*b*) looking for 'forcing' inputs (to be explained), (*c*) taking advantage of symmetry in the given network, if present, and (*d*) using the suitable model for analysing complex races.

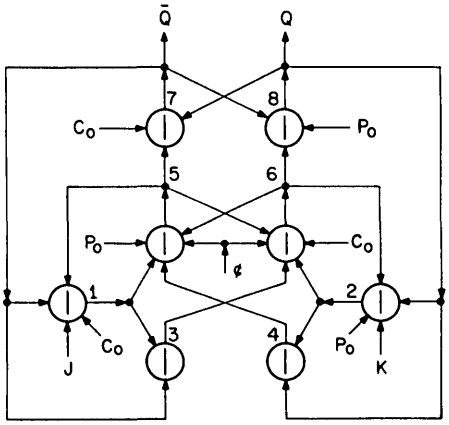

Fig. 3
Flip-flop to be analysed

We now illustrate some of these ideas by analysing the flip-flop of Fig. 3. It has eight gates, with outputs $y_1, \ldots, y_8$, two external outputs $\bar{Q}$ and Q, and five inputs as follows:

0-CLEAR	C_0
0-PRESET	P_0
CLOCK	ϕ
DATA INPUTS	J and K

We view the network of Fig. 3 as a proposed design for a positive-edge-triggered *JK* flip-flop, and our task is to verify that the designed network works properly.

The gate-excitation equations are

$$\begin{aligned} Y_1 &= (C_0 J y_5 y_7)' & Y_2 &= (P_0 K y_6 y_8)' \\ Y_3 &= (y_1 y_7)' & Y_4 &= (y_2 y_8)' \\ Y_5 &= (P_0 \phi y_1 y_4 y_6)' & Y_6 &= (C_0 \phi y_2 y_3 y_5)' \\ Y_7 &= (C_0 y_5 y_8)' & Y_8 &= (P_0 y_6 y_7)' \end{aligned} \qquad (1)$$

Note that these equations are listed in pairs, taking advantage of the symmetry.

Our first goal is to find all the stable states. We examine various inputs in turn. For example, if we set $P_0 = C_0 = 0$ and keep these inputs constant, we find

$$y_1 = y_2 = y_5 = y_6 = y_7 = y_8 = 1.$$

This in turn implies $y_3 = y_4 = 0$. Thus, for any input combination with $P_0 = C_0 = 0$ (eight input columns altogether), there is only one stable state $A \triangleq 11001111$.* Thus, $P_0 = C_0 = 0$ is 'forcing' to $A = 11001111$.

Similarly we find

$\underline{P_0 = 0, C_0 = 1, \phi = 0}$ is forcing to $B \triangleq 11101101$,

$\underline{P_0 = 0, C_0 = 1, \phi = 1}$ is forcing to $C \triangleq 11101001$,

and by symmetry

$\underline{P_0 = 1, C_0 = 0, \phi = 0}$ is forcing to $D \triangleq 11011110$,

$\underline{P_0 = 1, C_0 = 0, \phi = 1}$ is forcing to $E \triangleq 11010110$

When $\underline{C_0 = P_0 = 1}$, no gate output is forced. Specifying also $\phi = 0$ forces $y_5 = y_6 = 1$, but the other variables are not determined. Further, if we add $J = K = 0$, we find $y_1 = y_2 = 1$, but the remaining variables are still not determined. At this point, we may pick the value of one of the gate variables and see whether a stable state exists with that value. It is preferable to pick a value that forces as many gates as possible. For example, we can choose $y_7 = 0$. Then, in addition to $y_1 = y_2 = y_5 = y_6 = 1$, we find $y_3 = y_8 = 1$ and $y_4 = 0$. Thus, if there exists a stable state with $y_7 = 0$ for the input $C_0 = P_0 = 1, \phi = J = K = 0$, that stable state must be $11101101 = B$. Using the excitation equations we verify that this state is indeed stable.

We must determine next whether any stable state with $y_7 = 1$ can exist for the given input. Recall that y_1, y_2, y_5 and y_6 are all forced to 1. Thus, $y_8 = (P_0 y_6 y_7)'$ must be 0, forcing $y_4 = 1$ and $y_3 = 0$. Again we verify that this state $11011110 = D$ is stable. In summary, for the input just examined, there are two stable states B and D. Repeating this type of analysis for the remaining input columns, we find the flow table of Fig. 4, disregarding the unstable *-entries. Thus, we have a reasonably manageable 7 x 13 table instead of the 256 x 32 excitation table. We remark here that analytical methods for solving sequential Boolean equations[3] are, of course, applicable to the problem of finding stable states; however, these methods are quite laborious. Also, we may not dismiss this problem as easily programmable. One can enumerate all total states and check whether each is stable, but this would involve $2^8 \times 2^5$ computations for the network of Fig. 3. Consequently, this straight forward programming approach is not realistic. Of course, one can build heuristic short cuts into the program, in a way similar to what we are describing.

To complete the analysis we must now find the *-entries. For the present, we assume that the J and K inputs will not change while the clock input ϕ is changing. Thus, we postpone the computation of the entries marked '–' in Fig. 5. Also the P_0 and C_0 inputs are intended to be the 'asynchronous 0-PRESET and 0-CLEAR inputs', namely, $P_0 = 0, C_0 = 1$ 'presets' the flip-flop, forcing $\bar{Q} = 0, Q = 1$, and $P_0 = 1, C_0 = 0$ 'clears' the flip-flop forcing $\bar{Q} = 1, Q = 0$. The values $P_0 = C_0 = 0$ need not be used; this leads to the 'don't care' entries marked =.

There now remain 24 transitions to be evaluated. This problem will be discussed in the next Section.

4 Races

The evaluation of unstable entries in our flow table will involve the study of complex races. We have dealt with various race models in detail elsewhere,[1,2] and will adopt here the easily understood 'general single winner' (g.s.w.) model introduced there. We now define the model.

* '≙' stands for 'is by definition'

Let the network have n binary inputs $x_1, ..., x_n$, and s gates $G_1, ..., G_s$. The ordered s-tuple $y \triangleq (y_1, ..., y_s)$ is the *gate state*, the ordered n-tuple $x \triangleq (x_1, ..., x_n)$ is the input state, and the ordered s-tuple $Y \triangleq (Y_1, ..., Y_s)$ is the *gate excitation*, where Y_j is the value of the Boolean function f_j computed by G_j at (x, y), i.e.

$$Y_j = f_j(x_1, ..., x_n, y_1, ..., y_s), j = 1, ..., s.$$

Our problem is to determine the possible y-states that can result when the network starts in total state (x, y). We assume that x will be fixed. We use the following notation. For

$$y = (y_1, ..., y_{i-1}, y_i, y_{i+1}, ..., y_s),$$

$$y^{(i)} \triangleq (y_1, ..., y_{i-1}, y_i', y_{i+1}, ..., y_s).$$

In other words, $y^{(i)}$ is y with the ith variable changed.

present state		$P_0=0$ $C_0=0$	$P_0=0$ $C_0=1$		$P_0=1$ $C_0=0$		$P_0=C_0=1$ $\phi=0$ JK 00	01	11	10	$\phi=1$ JK 00	01	11	10
1 2 3 4 5 6 7 8			$\phi=0$	$\phi=1$	$\phi=0$	$\phi=1$								
1 1 0 0 1 1 1 1	A	(A),11	B	C	D	E	— *	— *	— *	— *	— *	— *	— *	— *
1 1 1 0 1 1 0 1	B	A	(B),01	C	D	E	(B),01	F *	F *	(B),01	C *	— *	— *	C *
1 1 1 0 1 0 0 1	C	A	B	(C),01	D	E	B *	F *	F *	B *	(C),01	(C),01	(C),01	(C),01
1 1 0 1 1 1 1 0	D	A	B	C	(D),10	E	(D),10	(D),10	G *	G *	E *	E	— *	— *
1 1 0 1 0 1 1 0	E	A	B	C	D	(E),10	D *	D *	G *	G *	(E),10	(E),10	(E),10	(E),10
1 0 1 1 1 1 0 1	F	A	B	C	D	E	B *	(F),01	(F),01	B *	— *	E *	E *	— *
0 1 1 1 1 1 1 0	G	A	B	C	D	E	D *	D *	(G),10	(G),10	— *	— *	C *	C *

forcing inputs

next state, $\bar{Q}, Q$

Fig. 4
Flow table for network of Fig. 3.

For each input $x \in \{0, 1\}^n$ define a binary relation R_x on the set $\{0, 1\}^s$ of gate states as follows. Given (x, y) compute Y. If $y = Y$, then yR_xy. Otherwise, for each i such that $y_i \neq Y_i$, we have $yR_xy^{(i)}$. For example, if $y = 110$, $Y = 011$ for some x, then $110\ R_x\ 010$ and $110\ R_x\ 111$.

We now consider the relation diagram for R_x. The nodes correspond to gate-states (elements of $\{0, 1\}^s$), and a directed edge leads from node y to node $\bar{y}$ if $yR_x\bar{y}$. If yR_xy then (x, y) is a stable total state. Otherwise consider all directed paths in the relation diagram for R_x that start at node y. Any such path reaches a cycle after a finite number of steps. If a cycle of length greater than 1 is found, there is an oscillation. *In this paper we assume that oscillations do not occur.* Thus, every cycle reached must be of length 1. If exactly one cycle (of length one) can be reached, the state corresponding to the node in the cycle is the unique stable state reached from y under input x. If more than one cycle can be reached from y, the situation represents a *critical race*. For further details see References 1 and 2. The main point here is that for each (x, y) we have a unique entry in the flow table. This entry may be (i) y if (x, y) is stable (ii) $\bar{y}$ if only state $\bar{y}$ can be reached from y under x, or (c) "?", indicating a critical race. The computation of these entries can be easily programmed.

Our flow-table of a gate network is a table with rows corresponding to those gate states y that are stable for at least one input. The columns correspond to input n-tuples or sets of equivalent input n-tuples. The entries in the table are computed using the GSW model. Outputs are shown for stable states only.

In Fig. 5 we illustrate the evaluation of unstable entries for two of the transitions. The unstable variables are underlined and edges are labelled with the subscript of the changing variable. The remaining unstable entries are obtained similarly.

In Fig. 4, the portion of the flow table in double lines provides a verification of the 'synchronous' behaviour of the flip-flop. When $\phi = 0$ and the flip-flop is originally set, it is in state B or F. When the clock arrives and $K = 0$, the network goes to state C and does not change its Q-value. If $K = 1$ when the clock arrives, the network goes to state E and Q changes. Once the clock goes on, the state becomes independent of the J and K inputs. (This feature is sometimes called 'data lockout', and because the J and K values present during the rising edge of the clock ϕ determine the future value of Q, the flip-flop is called 'positive-edge-triggered'.) When the clock falls, the new value of Q is remembered. Similar action takes place when the network was originally reset. Incidentally, the interested reader is referred to References 4 and 5 for a master-slave edge-sensitive flip-flop somewhat similar to the one we described above, the TTL unit 74110 with 16 gates.

To complete the analysis one should also evaluate the '—' entries in Fig. 4, using the same race model. Since double input changes may occur in practice, their consequences should be known. We will return to these entries in the next section.

5 Analysis with minimal feedback sets

As has been explained in Section 2, the stable states can be computed using a minimal set of feedback variables. We illustrate this now for the network of Fig. 3. One can verify that at least 3 variables are required to 'break' all the feedback loops. A possible

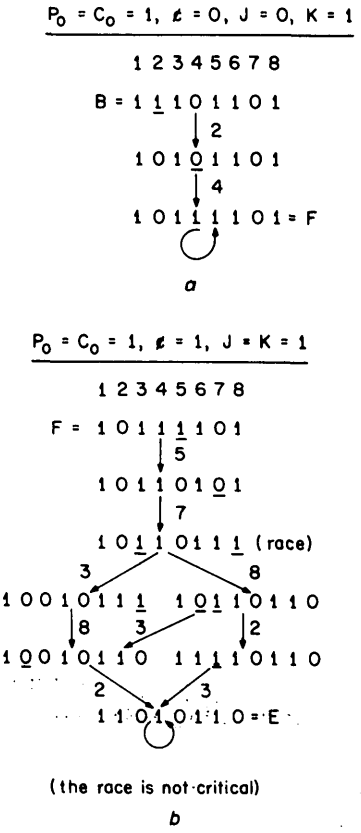

Fig. 5.
Examples of evaluating unstable entries.

minimal set is $\{y_1, y_6, y_7\}$. The other gate variables can be expressed as follows:

$$y_3 = (y_1 y_7) \qquad y_8 = (P_0 y_6 y_7)'$$
$$y_2 = (P_0 K y_6 y_8)' = (P_0 K y_6)' + (P_0 y_6 y_7)$$
$$y_4 = (y_2 y_8)' = (P_0 K y_6)(P_0 y_6 y_7)' + (P_0 y_6 y_7) \qquad (2)$$
$$= P_0 K y_6 + P_0 y_6 y_7$$
$$y_5 = (P_0 \phi y_1 y_4 y_6)' = (P_0 \phi K y_1 y_6 + P_0 \phi y_1 y_6 y_7)'$$

Thus, under stable conditions, the variables y_2, y_3, y_4, y_5 and y_8 can be expressed in terms of y_1, y_6 and y_7. After substituting and

present state			$P_0=0$ $C_0=0$	$P_0=0$ $C_0=1$		$P_0=1$ $C_0=0$		$P_0=C_0=1$							
				$\phi=0$	$\phi=1$	$\phi=0$	$\phi=1$	$\phi=0$ JK 00	01	11	10	$\phi=1$ JK 00	01	11	10
{A, D, E}	111	a	ⓐ,11	b	c	ⓐ,10	ⓐ,10	ⓐ,10	ⓐ,10	d	d	ⓐ,10	ⓐ,10	ⓐ,10	ⓐ,10
{B, F}	110	b	a	ⓑ,01	c	a	a	ⓑ,01	ⓑ,01	ⓑ,01	ⓑ,01	c	a	a	c
{C}	100	c	a	b	ⓒ,01	a	a	b	b	b	b	ⓒ,01	ⓒ,01	ⓒ,01	ⓒ,01
{G}	011	d	a	b	c	a	a	a	a	ⓓ,10	ⓓ,10	—	—	c	c

next state, $\bar{Q}, Q$

Fig. 6
Flow table using minimal feedback set $\{y_1, y_6, y_7\}$.

considerable manipulation and simplification, we obtain the following set of three equations:

$$y_1 = (C_0 J y_7)' + P_0 \phi y_1 y_6$$
$$y_6 = (C_0 \phi)' + P_0 K y_6 y_7' + y_1 y_7 \qquad (3)$$
$$y_7 = C_0' + P_0 y_6 y_7 + P_0 \phi K y_1 y_6$$

We can use our shortcut techniques as before, to compute the stable feedback states, or check which of the 8 triples (y_1, y_6, y_7) is stable. Either way we find that the following feedback states are stable:

$\underline{C_0 = 0}$: $a \triangleq 111$

$\underline{C_0 = 1, P_0 = 0}$, $\phi = 0$: $b \triangleq 110$

$\phi = 1$: $c \triangleq 100$

$\underline{C_0 = 1, P_0 = 1}$, $\underline{\phi = 0}, J = 0$: b and a

$J = 1$: b and $d \triangleq 011$

$\underline{\phi = 1}$: a and c

This of course agrees with the table of Fig. 4, except that gate states A, D, and E all correspond to feedback state a, (since we are examining only a subset of the gate variables), B and F correspond to b, C corresponds to c and G to d.

Now, for each stable total feedback state (i.e. input state together with feedback state) we can compute the values of y_2, y_3, y_4, y_5 and y_8, thus finding the complete stable gate state. Then we can proceed to the analysis of races, as in the previous section, to complete the construction of the flow table of Fig. 4.

One must be careful not to assume that the minimal feedback set is sufficient to describe the entire behaviour of the network, not just its stable states. Often the equations given above are used in the following form:

$$Y_1 = (C_0 J y_7)' + P_0 \phi y_1 y_6$$
$$Y_6 = (C_0 \phi)' + P_0 K y_6 y_7' + y_1 y_7 \qquad (4)$$
$$Y_7 = C_0' + P_0 y_6 y_7 + P_0 \phi K y_1 y_6$$

i.e. they are used as excitation equations. If we construct the flow table from these equations we find the table of Fig. 6. Now consider Fig. 4 assuming that all unspecified entries constitute don't cares. One easily checks that Fig. 6 happens to be a reduced version of Fig. 4. However, the reader can verify that, when the table of Fig. 4 is started in state F, with $C_0 = P_0 = 1, \phi = 0, J = 0$ and $K = 1$, and ϕ and K both change, the gate state model predicts a critical race to either C or E, but the minimal feedback set model predicts that the corresponding transition from b will always go to c. This again shows that the minimal feedback set model does not always yield the correct transitions.

6 Hazards reconsidered

In our direct approach to the analysis of a given gate network, hazards are taken into account implicitly. For a large network it may be more convenient to first decompose the network into a number of smaller parts, and then analyse the parts using our model. For example, we may have a combinational network driving a flip-flop. Then a hazard in the combinational network may cause the flip-flop to enter a wrong state. In this type of situation a hazard analysis would be needed.

We show how hazards[6] can be detected in our model. Let us

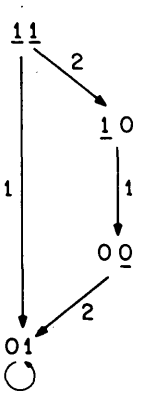

Fig. 7
Detection of a 1-hazard.

analyse the network of Fig. 1(*b*) as a sequential network, using our model. To avoid confusion of symbols, let y_1 and y_2 be the outputs of gates G_1 and G_2, respectively. The excitation equations under the condition $w = 1$ are:

$$Y_1 = x' \qquad Y_2 = (x y_1)'$$

When $x = 0$, we have the stable state $A \triangleq (1, 1)$. When x becomes 1, the excitation equations become:

$$Y_1 = 0 \qquad Y_2 = y_1'$$

There is a race as shown in Fig. 7. Observe that, if y_2 wins the race, the 'long' path is taken. During this transition the values of y_2 go through the sequence 1, 0, 0, 1, showing the presence of a 1-hazard in y_2. Thus hazards can be described and detected in our model. Furthermore, 'quantitative' information about the hazards can be obtained in the following sense. From Fig. 7 it is clear that the 0-pulse will occur only if $\Delta_2 < \Delta_1$ and that the duration of this hazard pulse is $\Delta_1 + \Delta_2$. Note that logic hazards, function hazards and dynamic hazards[6] are all detectable in our model

7 References

1 BRZOZOWSKI, J.A., and YOELI, M.: 'Digital networks' (Prentice-Hall, 1976)
2 BRZOZOWSKI, J.A., and YOELI, M.: 'Models for analysis of races in sequential networks' *in* BLIKLE, A. (Ed.): 'Mathematical foundations of computer science', pp. 26–32, and 'Lecture notes in computer science' (Springer-Verlag, 1975)
3 EVEN, S., and MEYER, A.: 'Sequential Boolean equations', *IEEE Trans.*, 1969, C-18, pp. 230–240
4 'Digital bipolar circuits' (Signets International Corporation, 1972)
5 'The TTL data book for design engineers' (Texas Instruments Inc., 1973)
6 UNGER, S. H.: 'Asynchronous sequential switching circuits' (John Wiley & Sons, 1969)