HAZARDS, CRITICAL RACES, AND METASTABILITY

Stephen H. Unger

Computer Science Department

Columbia University

9/7/93

Abstract

The various modes of failure of asynchronous sequential logic circuits due to timing problems are considered. These are hazards, critical races and metastable states. It is shown that there is a mechanism common to all forms of hazards and to metastable states. A similar mechanism, with added complications, is shown to characterize critical races. Means for defeating various types of hazards and critical races through the use of one sided delay constraints are introduced. A method is described for determining from a flow table situations in which metastable states may be entered. A circuit technique for defeating metastability problems in self timed systems is presented. It is shown that the use of simulation for verifying the correctness of a circuit with given bounds on the branch delays cannot be relied upon to expose all timing problems. An example is presented that refutes the conjecture that replacing pure delays with inertial delays can only eliminate glitches. Key Words asynchronous, critical race, delays, dynamic hazards, essential hazards, inertial delays, metastability, pure delays, sequential logic, timing problems, timing simulation.

1. Introduction Combinational logic circuits can operate correctly in the sense that their steady state outputs are correct, while generating spurious pulses, often termed glitches, when the input states change. If, depending on the values of relative delays along various paths, such behavior is possible, then we say that there are combinational hazards in the design. A hazard is manifested, in the operation of a physical instance of such a circuit, if a glitch actually occurs. A particular circuit with hazards may consistently manifest them, or occasionally the hazards may be manifested (depending perhaps on variable factors such as temperature or power supply voltages that can cause delay magnitudes to fluctuate), or the delays may be so related that, for the input state changes that occur in practice, the hazards are never manifested. In some situations these transients are of no consequence, as is the case for some circuits embedded in synchronous systems. But often they can lead t! o significant malfunctions, for example where the combinational logic generates signals controlling internal variables of sequential circuits, in which case the transient errors may be converted into steady state errors. Combinational hazards are classified as either static or dynamic, depending upon whether the output is specified to remain constant after the input change, or to change, respectively. A static 1-hazard is one where the output both before and after the input change is supposed to remain constant at 1 and a negative going pulse may be generated. Static 0 hazards are analogously defined where the output is supposed to remain constant at 0. A dynamic hazard is one where a glitch may appear prior to an output change from 0 to 1 or from 1 to 0. It is well known [7, 24] that combinational logic functions can always be realized with circuits that have no hazards for single input change (SIC) operation. Stated in another way, combinational hazards for SIC operation can always be eliminated. When more than one input may be changed simultaneously,

i.e. for multiple-input-change (MIC) operation, certain hazards, called function hazards, are inevitable [5]. If there is a hazard in a circuit that is not a function hazard (i.e. the function can be realized without this hazard), then it is a characteristic of the logic design, and is referred to as a logic hazard. Where it is not possible or not desirable to eliminate a hazard by means of logic design, it may be possible to defeat it, that is to prevent it from being manifested. This can be done by ensuring that the delays along certain critical paths are not be so related as to cause the glitches to occur. Another way to deal with hazards is to allow them to be manifested, but to filter them out through the use of inertial delays of sufficient magnitude to suppress the glitches. This is usually considered to be a last resort since it may be costly and may significantly slow down circuit operation. In many cases, defeating hazards is not feasible because it may necessitate guaranteeing that the delays along two (or more) different paths be constrained to differ by no more than some very small amount. This is generally not achievable. On the other hand it is generally possible to enforce one-sided constraints, i.e. to ensure that the total delay along one path exceeds that along another path by some minimum amount. In some cases, particularly where only certain input changes are possible, one- sided constraints may be sufficient. Other types of timing malfunctions are endemic to sequential circuits. If a sequential function is realized by a circuit with more than one internal variable, and if, for certain situations, an internal state change entails the simultaneous change of more than one state variable, then this is called a race condition. If the stable state ultimately reached depends on the outcome of the race, then it is referred to as a critical race. Generally we would consider critical races as being design defects, since it is difficult to control path delays so as to ensure particular outcomes of races. It is always possible to eliminate critical races by appropriate choices of state assignments. "Fixing" races by means of one sided constraints, on path delay values, is sometimes an acceptable option. Another class of timing problems in sequential circuits are those arising from situations where, if circuit delays are such that an internal state change resulting from an input change is perceived somewhere in the circuit as having occurred before the precipitating input change, then the system generates an output glitch or goes to the wrong internal state. When such behavior is inherent in the function being realized, we say that the function has an essential hazard. If the resulting malfunction is an output glitch, then it is a transient essential hazard. If the system might wind up in the wrong stable state, then the problem is designated as a steady state essential hazard. Both types of essential hazards can always be defeated by one sided delay constraints. In particular, delays in the internal variable branches (y branches) will always do the job [24]. Normally, we expect that, except during very brief, bounded, time intervals, each of the internal variables of a sequential circuit has one of two values (usually designated as 0 or 1). However, under certain conditions, any nontrivial sequential circuit can enter a mode where one or more of its internal variables takes on a value "in between" 0 and 1, possibly fluctuating in this range, for an indefinite period of time. A system in this condition is said to be in a metastable state (MSS). It has been shown that there is no way to eliminate the possibility of this occurring (although the probability can be made arbitrarily small), and that here is no way to put an upper bound on the interval during which the metastable state persists. Suppose that, in some logic circuit, a signal (input, output, or internal) undergoes two consecutive changes. Then we might refer to this pair of changes as constituting a pulse. (It is a positive pulse if the first change is 0->1, and is negative if the first change is 1->0.) The width of the pulse is the time interval between the changes. If this interval is short, we would say we have here a short pulse. In any real circuit, there exists a time value such that if the width of a short pulse is below that value, the pulse will definitely be ignored, due to the inertial properties of real systems. If the width is increased from this value, the pulse is marginally recognized, in the sense that it may or may not have an effect at different gate inputs. Finally, if the pulse width exceeds some larger value, then it will definitely be recognized by the circuit. A pulse whose width is in the marginal range, is referred to as a runt pulse. (Pulse amplitude is also at f! actor here. Runthood is in general a property of both width and amplitude, the basic idea being that a runt pulse is a pulse whose existence is marginal.) As is shown later, runt pulses are related to metastability. Any sequential circuit will malfunction if input changes are spaced too closely together. Such failures may be characterized as misuses of the circuit or failures to meet data rate specifications, rather than as timing faults. An objective of this paper is to show how a particular circuit feature, namely the possibility of a gate receiving contradictory signals simultaneously on different inputs, is common to all of the timing problems mentioned above. It is also shown that the various problems may be usefully viewed from different vantage points: boolean logic expressions, Karnaugh maps, flow tables, and logic circuit diagrams. This is helpful in enhancing our understanding of the mechanisms involved, identifying problem transitions, and devising means to eliminate or defeat problems. Techniques are presented for defeating any dynamic hazard and many critical races with one sided delay constraints. A simple procedure is described for determining by inspection of a flow table if a particular input transition can lead to metastability. A variation on a previously known circuit technique for filtering out the effects of metastability is presented which can cope with a broader class of problems.! It is shown that timing simulators cannot be relied upon to test for the presence of timing problems. Finally, it is demonstrated that replacing pure delays with inertial delays can, under certain circumstances, actually introduce timing problems not previously present. The systems discussed here are assumed to be constructed of AND, OR, and INVERTER gates. Extensions of the results to cover systems including NAND, and NOR gates are straightforward. Delays, with given upper and lower bounds, are associated with all wires. (Gate delays may be absorbed in the delays of wires at the outputs.) In order to simplify the discussions, it is assumed that each delay is pure, and that the delays for rising and falling signals are the same. It is not difficult to extend the results to models with mixed pure and inertial delays and with different delay values depending upon the direction of a value change. Where multiple input changes (MICs) are discussed, it is assumed that all of the input variables involved change simultaneously. Variations from simultaneity can easily be taken into account by delays in the wires leading from the inputs. A basic introduction to asynchronous sequential circuits consistent with the approach taken in this paper is in [26] , and a more detailed treatment of hazards and critical races can be found in [24]. The origins of the theory of combinational hazards are mainly in [7, 14], [5] (multiple input change static hazards), [24], [1, 2] (multiple input change dynamic hazards). The basic work on critical races and state assignments free of them is in [6, 11, 23]. More recent approaches to some of these problems are in [10, 17]. Methods for evading the problem by using locally generated clock pulses were introduced in [4, 25]. A more sophisticated approach is in [16]. The origins of the basic ideas pertaining to metastability are in [3, 8, 18, 21, 12]. More recent work on this topic is in [9, 19]. Static and dynamic combinational hazards are treated in the next two sections. In Sections 4 and 5 two types of essential hazards are shown to be associated with sequential functions that cannot be eliminated, but which can be defeated. They are shown to be related to static combinational hazards. In Section 6 it is shown that sequential dynamic hazards also exist, but that these are not inherent in the functions. Metastability is treated in Section 7, where it is shown that situations where it can be initiated by either runt pulse inputs or the internal generation of runt pulses following multiple input changes can easily be identified from the flow table point of view. It is also shown there that the effects of metastability can be prevented from reaching outputs by a circuit filter. Critical races are examined from a circuit as well as flow table point of view in Section 8. The resemblance, with a twist, between critical races and essential hazards is pointed out, ! and a technique for using one sided delay constraints that goes beyond "fixing" races is presented that allows the use of some state assignments with critical races. A consequence of the technique for defeating dynamic hazards is shown in Section 9 to provide a counterexample to the idea that if a circuit works properly with some delay element assigned either of two specific values, then it will also work for all intermediate values. Finally, it is shown in Section 10 that pure delays are not always more troublesome than inertial delays with respect to timing problems. 2. Static Combinational Hazards Assume we restrict ourselves for the moment to logic circuits composed only of AND, OR and INVERTER gates. How can a spurious 0 pulse be generated in such a circuit as a result of an input change in which no variable changes value more than once? That is, how can a 1 hazard be produced? One way would be for a spurious 1 pulse to be generated, and then passed through an inverter. Putting aside that possibility, we might ask first how a spurious 0 pulse can be produced at the output of an AND gate, if none of its inputs are pulses (i.e. if none of its inputs change more than once). The output of the AND gate must be 1 prior to the (possibly multiple) input change, and must also be 1 in the steady state after the input change. Then all of the inputs must be 1 prior to the change, and they must all be 1 after the input change. But if none of the inputs to the AND gate change more than once, it follows that all of the inputs to the AND gate must remain constant at 1. In t! hat event, assuming the gate is not defective, it is impossible for a spurious 0 pulse to be generated at the output of an AND gate. Next, consider how a spurious 0 pulse can be generated at the output of an OR gate (this is a variation of an argument in [2]). Four conditions are clearly necessary: 1. For the initial state at least one input must be 1. (The initial output must be 1.) 2. For the final state at least one input must be 1. (The final output must be 1.) 3. All of the inputs equal to 1 in the initial state must change to 0. (If any input remained fixed at 1, then the output would remain fixed at 1 and no glitch could occur.) 4. At least one of the inputs initially equal to 0 must change to 1. (Otherwise, given condition 3, condition 2 would not be satisfied.) Suppose all of these conditions are satisfied. Then if all of the changes described by condition 4 are delayed until after all of the changes described by 3 have occurred, then a 0 pulse will be produced. Hence the listed conditions are both necessary and sufficient for the existence of a 1 hazard when there is no 0 hazard at the output of any gate (which could then be inverted to produce a 1 hazard). The dual of this argument demonstrates that no 0 hazard can be produced at the output of an OR gate, and that a 0 hazard can be produced at the output of an AND gate if and only if: 1. For the initial state at least one input must be 0. (The initial output must be 0.) 2. For the final state at least one input must be 0. (The final output must be 0.) 3. All of the inputs equal to 0 in the initial state must change to 1. (If any input remained fixed at 0, then the output would remain fixed at 0 and no glitch could occur.) 4. At least one of the inputs initially equal to 1 must change to 0. (Otherwise, given condition 3, condition 2 would not be satisfied.) It is not difficult to show that the above results, with input values complemented, apply when ANDs are replaced with NORs and ORs replaced by NANDS. One way to summarize the ideas expressed above is to say that a static hazard is generated at the output of a gate (AND, OR, NAND, or NOR) if and only if there are inputs to the gate changing in opposite directions, while none of the fixed inputs have values that desensitize the gate to the other signals (i.e. no input to an OR or NOR gate is fixed at 1, and no input to an AND or NAND gate is fixed at 0). Consider now the important special case of single-input change (SIC) operation. The first result reduces to a requirement that, in order for a 1 hazard to be generated at the output of an OR gate, one input must change from 0 to 1, while another input must change from 1 to 0 (with no inputs fixed at 1). Since only one input variable, call it X, is changing, it follows that there must be one path from X to the OR gate with an even number of inverters, and another path from X to the OR gate with an odd number of inverters. Similar statements may be made with respect to AND, NOR, and NAND gates, and with respect to 0 hazards. The basic model for static hazard generation is thus the configuration shown in Fig. 2.1a, for 1 hazards, with the dual shown in Fig. 2.1b, for 0 hazards. (It is easy to show that a NAND gate could replace the OR gate, and that a NOR gate could replace the AND gate.) In the case of a 1 hazard for example, if the delay through the lower path to the OR gate (Fig. 2.1a) were sufficiently long relative relative to the delay through the inverter,in a particular physical circuit, then no hazard would be manifested for a change of X from 1 to 0. But then, for a change of X from 0 to 1 the hazard would definitely be manifested. The hazard would not be manifested for either change if the delays are so related that the opposite changing signals arrive sufficiently close together in time. An intermediate situation would be if the time interval between the signal arrivals is marginal, in which case a runt pulse might be produced. This point will be referred to again later in the discussion of metastable states. Consider the logic function specified in the 4 variable Karnaugh map shown as Fig. 2.2. A sum of products (SOP) expression for this function is: $Z=A\ B+AC+BC+B\ C\ D$ (1) This can be factored to yield: $Z=(A+B)(A\ +C)+B\ C\ D$ (2)

The circuit shown as Fig. 2.3 corresponds to this expression. Examining the circuit in the light of the preceding discussion, note that, from the A input, there are 2 paths to AND gate-7, one with 1 inverter, and the other with no inverters. Both paths are sensitized when B=0 and C=0. Thus, under these constraints, when A changes, there is a 0 hazard at the output of gate-7. However, if D=1, this output does not propagate to the output Z, since Z will be held at 1 by the 1 input from gate-6. But if D=0, the output of Z is same as that of gate 7; that is there can be a negative glitch at the circuit output. Thus, we have a 0 hazard when A changes while B=C=D=0. An essentially equivalent process can be carried out on expression (2), which mirrors the circuit. Observing the appearance of both A and A in the expression, we attempt to fix the other variables so as to reduce the expression to AA . This can be done by setting B=0 to convert the first parenthesized subexpression to A, and then, setting C=0 converts the second parenthesized subexpression to A . All that remains is to set D=0 thereby eliminating the B C D term. A third way to arrive at the same result is a bit different. Multiply out the product in expression (2), a hazard preserving operation, and do not eliminate the resulting AA term (that would not be hazard preserving). The result is: Z=A B+AC+BC+AA +B C D (3) Setting B=C=D=0 reduces (3) to , AA again revealing the 0 hazard.

Now consider variable B. There are two paths from B converging at final OR gate 8, one path with a single inverter on it, and the other with no inverters. Both paths are sensitized if A=0, C=0, and D=1. This corresponds to a 1 hazard at the output. The same conclusion can be reached from expression (2) by noting that setting A=C=0 and D=1 reduces that expression to B+B . A different way to see this, using expression (3), is to note that the consensus term, between A B and B C D, namely A C D, is missing from the expression. Thus, within the cube defined by A C D there is a hazardous transition. Finally, an examination of the K map of Fig. 2.2 shows that none of the chosen subcubes covers the transition under discussion (the missing cube is of course A C D), confirming again that there is a 1 hazard for transitions between 0001 and 0101. The hazard can be eliminated by adding to the circuit a path that corresponds to A C D. Note that the same techniques used to identify the 1 hazard could be used on the 0 hazard, by complementing the map, expressions, and circuit. (complementing the circuit would mean swapping AND gates and OR gates and complementing all inputs.) There is also a 1 hazard for transitions between 1001 and 1011. It can be detected using the same techniques illustrated for the 1 hazard between 0001 and 0101. Consider next a multiple input change. Suppose that with A and B both fixed at 0, C and D are both changed from 0 to 1. Paths from C (with 1 inverter) and from D (with no inverters) converge at AND gate 6. With B set at 0, we have the conditions for a positive glitch at the output of gate 6. With A=B=0, the output of gate 4 is 0, which forces the output of gate 7 and hence the lower input to gate 8 to be 0, so that the glitch will be propagated to the output Z. Thus we have a MIC hazard for transitions between 0000 and 0011. Note that, with A=B=0, expression (2) reduces to C D. This term (corresponding to an AND gate) has its 2 inputs changing in opposite directions when C and D both change in the same direction. An examination of this transition on the K map of Fig. 2.2, shows that, between the initial and final states of this transition, both of which are 0 points of the function, there is a minimal length path that passes through the point 0001, where Z=1. This i! s the condition for a function 0 hazard. There is no way to eliminate the function hazard. But, as noted above, it can be handled in one of two ways. One is simply to place an inertial delay element at the output of gate 6 or gate 8. If its magnitude exceeds that of the maximum possible width of the glitch, which is the maximum difference in the delays along the two critical paths, then the spurious pulse will be filtered out. Suppose that instead of this approach, which delays the circuit output, we attempt to defeat the hazard, i.e. to prevent the glitch from being generated. This might be done by ensuring that, for the 0000->0011 transition, the signal from C, which would hold the gate 6 output at 0, arrives before the signal from D, which would turn on the output of gate 6. This could be done by inserting a sufficiently large delay element (pure or inertial) in the path from D to gate 6. But now observe what would happen for the inverse transition, i.e. 0011->0000. Again, the effect of the C change woul! d reach gate 6 before the effect of the

D change. The result is that the output of inverter 2 goes to 1 before the signal from D goes to 0 so that the hazard is definitely manifested. Thus this approach is useful only where the input sequences are restricted so that such inverse pairs of transitions do not occur. For some technologies there may be a way around this problem. Suppose that for the current example, inverter 2 responds with relatively little delay to a change in C from 0 to 1 (i.e. the inverter output falls quickly) while taking much longer to respond to an 1->0 change in C (i.e. the inverter output rises slowly). This indeed corresponds to the behavior of an NMOS inverter. (Differences in logic gate thresholds can also produce similar effects.) Then, if the delay in the path from D to gate 6, for signal changes in both directions, is constrained to be between the two delay values for the path from C, then it may be possible to defeat the hazard for both transitions. This approach, which entails a two sided delay constraint, is applicable to all kinds of static hazards. 3. Dynamic Combinational Hazards For input changes that cause the output of an OR gate to change from 0 to 1, a dynamic hazard can be generated at (as opposed to being propagated to) the output of this gate, only if a positive glitch occurs at one of its inputs and then terminates before the arrival at another input to the OR gate of a signal changing monotonically from 0 to 1. Such a situation is depicted in Fig. 3.1. If g0 and g1, and m are simultaneously turned on, then complementary input changes occur at the inputs to the AND gate, which, as shown in Fig. 2.1b, is the basic mechanism for producing a glitch at the output of the AND gate (the signal labelled G). Note that g1 turns on the glitch and g0 turns G off. The 1 hazard is manifested if the effect of g1 is felt at the AND gate before the effect of g0. The monotonically increasing signal is m. Thus, if the signal from m arrives at the OR gate after the G signal at the input to the OR gate has gone on and then off, we would have a spurious pu! lse at Z preceding the steady state change of Z from 0 to 1. Assume now that there is a MIC involving m, g0 and g1. The dynamic hazard would be manifested if the path delays in the circuit are such as to cause events to occur in the order specified above. That is, if the order of the 3 input changes was perceived at the the inputs to the OR gate as having been g1, g0 (required to produce a glitch at G), and then m. It will now be shown that, unlike the case of static hazards, all dynamic hazards (both logic and function hazards) can be defeated for changes in both directions, by one sided delay constraints. (There are actually two different sets of constraints that can be used in each case.) Suppose that the path delays from g0, g1, and m to the OR gate inputs are respectively d0, d1, and dm. Then there are 2 ways to defeat the dynamic hazard for the case where Z is to change from 0 to 1 (i.e. for the MIC in which all 3 inputs go on). One way is to prevent the glitch at G from occurring. This can be done by making d1>d0. The second way is to make the glitch overlap the turning on of m (i.e. forcing the effect of the m going on to be felt while the glitch is still on). This will occur if d0>dm. Now consider the reverse transition, i.e. where all 3 input variables are switched off. Now the generation of the glitch at G is prevented if d0>d1, which constitutes one constraint for defeating the dynamic hazard. The dynamic hazard can also be defeated by allowing the G glitch, but overlapping it with the m signal (i.e. by ensuring that the effect of m going off is delayed until the effect of g0 going off is perceived by the OR gate). This is achieved by imposing the constraint dm>d0. Now observe that if d0>d1, dm, (i.e. d0>d1 and d0>dm), then the dynamic hazard is defeated for transitions in both directions. The same is true if d0<d1, dm. The simplest example of such a hazard is illustrated in Fig. 3.2, a SIC dynamic hazard. Here all 3 of the basic inputs are derived from the single input variable X. Corresponding to this circuit, is the logic expression: Z=XX +X. The first X symbol corresponds to g1, the second X to g0, the product XX to G, and the third X to m. The branch delays are denoted as a (which includes the inverter delay), b, c, and d. The dynamic hazards for changes of X in either direction are both defeated by the constraints a+d>b+d, and a+d>c, or alternatively by a+d<b+d, and a+d<c. Consider the circuit of Fig. 2.3 (used above to illustrate static hazards). There is a 0 >1 MIC dynamic logic hazard for the transition 1101 >1011, i.e. for the input state change B/C- when A=D=1. For this situation, m=C, g1=B , g0=C, G=B C (the output of gate 6). The hazard and the reverse hazard can both be defeated by constraining the g0 path delay to either exceed or be less than the delays in the g1 and m paths. For example, we could require the delays in the path from C through gates 2 and 6 to the input of gate 8 (this is the g0 path) to exceed the delays in the path from C through gates 5 and 7 to the input of gate 8 (this is the m path) and the delays in the path from B through gates 1 and 6 to the input of gate 8 (this is the g1 path). Alternatively, we could alter the above constraints by replacing the phrase "to exceed" by the phrase "be less than". The circuit (not shown) corresponding to Z=ABC+A B C illustrates a dynamic function hazard for input changes between 000 and 111. The G signal is generated at the output of the A B C gate, and the m signal is generated at the output of the ABC gate. Both A and B produce g0 signals, C produces the only g1 signal, and A, B, and C all contribute to m. If we delay the paths from both A and B to the A B C gate by an amount larger than the other delays, then, during the 000 >111 transition, that gate will not turn off the glitch until after the m signal has turned on Z, which defeats the hazard for that transition. For the reverse transition, the A B C gate will not see the A and B signals go on until after it sees the C signal go off, so that no glitch is produced, thereby defeating the reverse hazard as well. A similar argument shows that minimizing the delays from A and B through the A B C gate to the input of the final OR gate will also defeat both hazards. As a practical matter, when this route is chosen, the action taken would be to increase the delays in either or both of the other relevant paths. In fact, the best strategy might be to choose the solution that entails the least cost in terms of delay padding. The above discussion is easily extended to cover circuits terminating in AND gates by simply using dual arguments. Of course NAND and NOR gates are also included in the discussions of AND and OR gates respectively (the outputs are simply complemented). In some cases there may be several OR gate inputs at which G signals may appear, and perhaps several inputs at which m signals appear. Sufficient conditions for defeating the dynamic hazards in such cases are easily specified by, for example, constraining d0 values for all the G signals to exceed all d1 values and all dm values. Weaker sets of constraints may be found in some cases by more careful analyses taking into account various glitch overlaps. The results are extendable in a straightforward manner to cases where the delays for rising and falling signals are different, and it makes no difference whether the circuit delays are pure or inertial. In a particular circuit with several distinctly different dynamic hazards, there may be contradictions among the delay constraints required to defeat certain subsets of these hazards. The importance of the above result follows from the fact that, in many cases involving MIC operation, it has been shown, by Bredeson [2], that not all dynamic logic hazards can be eliminated. That is, for some functions, while any particular dynamic hazard can be eliminated by a redesign of the logic circuit, the new circuit will have some other MIC hazard. In fact, the function described in Fig. 2.2 and realized by the circuit of Fig. 2.3 is an example of such a function. This is an example in which there are a number of potential dynamic logic hazards in the function. It is possible to eliminate some, but not all (without introducing static logic hazards) by logic design using Bredeson's method, and then to defeat the rest by the method described above. It is not clear whether this can be done for all functions. In practical cases it is important to understand that, due to input constraints, transitions corresponding to many of the MIC hazards may never occur, so tha! t some hazards need not be dealt with at all. This issue is treated, from the point of view of eliminating (as opposed to defeating) hazards in 2 stage logic circuits, by Nowick and Dill [17]. 4. Sequential Circuits: Transient Essential Hazards Consider next those hazards that are inherent to sequential circuits. To begin with, let us examine those hazards that are inherent in certain sequential functions. That is, they cannot be eliminated, although they can be defeated by suitably constraining relative delays along certain paths. The subsequent discussion is restricted to single output change (SOC no output signal is specified to change more than once as a result of a single change in any input signal) sequential functions and operations are assumed to be SIC. An example of such a hazard is embedded in the sequential function described by the flow table of Fig. 4.1a. With the given encoding of the rows in terms of the single internal variable y, it is simple to obtain the K maps of Fig. 4.1b for Y and Z. From these the logic expressions below are easily generated: Y=Ay+B, Z=AB y The classic block diagram circuit model for realizing such a function is depicted in Fig. 4.1c and the detailed logic circuit diagram corresponding to the above expressions is shown in Fig. 4.1d. Clearly there are no critical races or combination logic hazards (for SIC operation). But now consider carefully (referring to Fig. 4.1a) what might happen when B is turned on with the system initially in the state 1 10. According to the flow table specification, the first event is a state change to 1 11, corresponding to a lateral move in the flow table to the first row of the 11 column. The output remains at 0. Since the next state entry is 2, the next specified event is a change in the internal state to row 2, which brings the system to state 2-11, a stable state. Note that in all three of the states involved in this transition, the output, Z, is specified to remain fixed at 0. In terms of the block diagram that represents any circuit realizing the function, the above process begins with a change at the B input, which propagates through the logic block to the Z and Y terminals (in general, several Y variables might be involved). Next, the change in the Y variable(s) causes a change in the corresponding y variable(s) at the input end of the logic. The subsequent y change(s) then propagate through the logic again to the Z and Y terminals. Suppose that, due to a relatively long delay in the path from the B input to the Z output, the signal from the y terminal reaches the Z terminal before the signal from B? Then it would appear at the Z terminal as though the internal state change had occurred before the input change that precipitated it. From the point of view of the flow table, it is as though the vertical transition occurred before the horizontal transition, i.e. that the sequence of states was 1 10, 2 10, 2 11. But if this is the way things! look from the Z terminal, the output would change to 1 for a brief interval during the transition, corresponding to the difference in arrival times at Z of the signals from B and from y. This constitutes a spurious pulse, or glitch at Z. The possibility of such a malfunction is referred to as a transient essential hazard. In the current example, the hazard would be manifested if the total path delay (see Fig. 4.1d) from B to the input of the AND gate via the inverter exceeds the delay in the path from B to the input of the the AND gate via the OR gate. This type of hazard is mentioned in [24], but is not treated in detail. Before analyzing this situation further, a formal definition of a transient essential hazard is presented below. First define Z(r,c) as the output for the total state r c, i.e. the state corresponding to row r , column c. If X is an input variable, and if the stable state reached after a single change in X with the system initially in stable state r c is r' c', and if the stable state reached after a second change in X is r''-c, and r''-r (r'' might be equal to r') then a transient essential hazard exists if Z(r,c)=Z(r',c')-Z(r'',c). This is equivalent to saying that a transient essential hazard exists if, for some stable state s, and for some input variable X, the output Z is specified to remain unchanged when X changes once with the system initially in s, but Z is specified to change if X is changed a second time. No change in the state assignment or the logic design can eliminate the possibility of a malfunction if the path delays are such as to reverse the apparent ordering of the input and internal state changes as seen at the Z terminal. But simply inserting a sufficiently large delay between the Y and y terminals does defeat the hazard by ensuring that no such misperception can occur anywhere in the circuit. Now let us examine the situation more closely from the circuit point of view. First consider Fig. 4.1e, which is identical to Fig. 4.1d except for a minor topological re arrangement. Focussing on the part of the circuit that generates Z, note that there are two paths from B to the inputs of the rightmost AND gate. One path includes an inverter, and the other does not. If we fix A at 1 (which is its value in the scenario under consideration, and fix the lower input to the OR gate at 0 (which it might very well be during all but the very end of the scenario) then the circuit reduces to that of Fig. 4.1f. This is precisely the model for the static 0 hazard shown in Fig. 2.1b. (Of course if the nature of the transient essential hazard was to produce a negative glitch, then the dual circuit, Fig.2.1a, for a static 1 hazard would apply.) The problem is also evident from an examination of the expression for Z (i.e. AB y). The A variable is fixed at 1, reducing the expression to B y, in which the B -term goes on while the y term goes off. >From the point of view of the K map description of the Z function (Fig. 4.1b), what is happening is that a transition is occurring from the point 100 to 111. For this MIC transition, there is a function 0 hazard. Even though there is only a single input change to the overall sequential circuit, the internal state transition introduces a second variable change, namely y. It is no coincidence that the combinational hazard is a function hazard, since, as the previous arguments have shown, the order in which the variables change determines whether or not a pulse is produced, which is in accordance with the definition of a function hazard. 5. Sequential Circuits: Steady-State Essential Hazards A mechanism very similar to that described in the previous section is responsible for a related type of hazard that can bring a circuit to the wrong stable state after certain transitions. The problem is illustrated in Fig. 5.1. A very simple sequential function is specified by the flow table in Fig. 5.1a, where a race free state assignment is shown. Using this assignment, K maps for Y1 and Y2 are produced in Fig. 5.1b (the output Z is not a factor in this situation). Logic expressions are: Y1=X y2+y1, Y2=X+y2, Z=y1 A circuit realization in standard form is shown in Fig. 5.1c. There are no race conditions (therefore no critical races) and all logic expressions are free of hazards for SIC operation. Consider now what might happen when X is turned on with the system initially in total state 1 0. According to the flow table, the total state should change from 1 0 to 1 1 (an unstable state), and then the internal state should change, bringing the system to 2 1. That is, following the input state change, there is an internal state change. With the given state assignment, following the X change, y2 should change. But suppose now that at the Y1 terminal the y2 change is perceived as having occurred first. Then the system would appear to be in state 2 0, where Y1 is supposed to change to 1. The result would be a pulse at the Y1 terminal, which could cause the system to end up in state 3 1, the wrong stable state. The logic circuit has been redrawn in Fig. 5.1d to clarify the situation. From this diagram, it is clear that there are two paths from X to Y1. One of these is through the inverter and the AND gate, while the other is through the Y2 OR gate and the AND gate. With y2 initially 0, the lower OR gate acts as a wire, and we have again the situation of complementary inputs being fed to an AND gate, the condition for generating a transient 1 pulse. The situation is very similar to that shown in Fig. 4.1e for the transient essential hazard, except that in this case the glitch is fed to the feedback circuit generating Y1, instead of directly to the output. Thus, the glitch affects not just the output, but also the internal stable state. Looking at the expression for Y1, the problem clearly reveals itself. With y1=0, we have Y1=X y2, so that, when X and y2 both go on, the components of the AND expression change in opposite directions, thus opening the way for a spurious pulse t! o be generated. >From the circuit point of view, the mechanisms for static combinational hazards and for both transient and steady state essential hazards are the same: signals changing in opposite directions reach inputs of a gate at about the same time. This problem is inherent in the sequential function being realized. Such a steady state essential hazard exists for any transition in a SOC sequential function involving a change in one input variable X if, starting in some stable state, a single change in X is specified to bring the system to a different stable state than the one specified after 3 changes of X. It has been shown [24] that there is no way to eliminate steady state essential hazards by manipulating the state assignment or the circuit logic. The only solution is to defeat the hazard by controlling the relative delays along the critical paths. As in the case of transient essential hazards, this can always be accomplished by a delay between the Y and y nodes of the y variable(s) that are supposed to change in the course of the transition. This ensures that the precipitating input change will be perceived everywhere as having occurred before any resulting internal variable changes. 6. Sequential Circuits: Dynamic Hazards Consider next the flow table of Fig. 6.1. There are no essential hazards in the function described. It is a simple matter to derive hazard free SOP expressions for the internal variable and for the output. These are: Y=ABC+Ay, Z=ABy+AC y. There are 3 paths from C to Z: A direct path with one inverter, corresponding to the C in the AC y term of the Z expression, and 2 uninverted paths through Y, both through the C in the ABC term of the Y expression, and then via the appearances of y in each of the 2 terms of the Z expression. These constitute a dynamic hazard, since they are all sensitized with A=B=1 and y initially 0. The transition involved is the one starting in state 1 110 of the flow table, with C changing from 0 to 1. If the path to the y input to the AND gate realizing AC y has the shortest delay of the 3 paths, then Z goes on. If the next event is that the change in C turns off the C in the AC y term, then Z goes off again, and remains off until the change in y gets through to turn on Z via the ABy term. What has happened is that turning on C turns on Y, but the change in y is received first by the AC y AND gate, which sees the system in 2 110 and therefore goes on. This same gate then sees t! he C change, thereby perceiving the system to be in 2 111, and hence goes off. Last of all, the ABy gate sees the system in internal state 2 (i.e. it sees y go on) and therefore it goes on. Unlike the essential hazards of the previous two sections, this hazard can be eliminated. It is a logic hazard, not a function hazard. All that is necessary in this case is to factor the Z expression to obtain Z=Ay(B+C). Since B is fixed at 1 throughout the events of interest, the parenthesized term (B+C) is also fixed at 1, and so Z goes on (and remains on) as soon as the AND gate in the Z circuit receives the y signal. While any particular dynamic sequential hazard can be eliminated, for a given state assignment, which fixes the logic functions to be realized, it is not always possible to eliminate all dynamic logic hazards (as pointed out in Section 3 above). The Bredeson method can be used to eliminate hazards to the maximum extent possible. There is always the option of defeating, rather than eliminating dynamic sequential hazards. This can always be done by simply delaying the appropriate y variables so that the events following an input change are perceived everywhere in the correct order (i.e. the input changes are seen to occur before the internal state changes that they initiate). The other approach described in Section 3, biasing the various path delays, can also be used. There are no "essential" dynamic hazards. Since the mechanism for SIC operation of SOC functions is such that the total number of total states involved in a transition (including the initial and final states) is at most 3, there is no way such an output sequence as 0101 to occur as a result of false perceptions of the ordering of states. 7. Metastability: Detection and Defeat The flow table of Fig. 6.1 will be used to illustrate a discussion of metastability. Suppose the initial total state is 1 101, and that input B is switched on. Then the immediate transition is to 1 111, followed by an internal state change leading to the stable state 2 111. If B is then turned off, the system goes to the stable state 2 101. The assumption thus far is that B is held at the 1 value long enough for state 2 111 to be reached before it is switched back to 0. Requiring that input changes be constrained to occur only when the system is in a stable state is called the fundamental mode assumption. But what if this constraint is violated and the 1 pulse at the B terminal is shorter than required to complete the transition? First suppose that the pulse is very short. Then the inertial delays in the wiring and in the gates may be sufficient to filter out the pulse altogether, so that the system simply remains in state 1 101, with no detectable change in the output. It is as though nothing at all happened. If, however, the pulse width falls into some narrow range between these two cases, then something rather different can happen. If the B signal goes off while the y signal is somewhere near the midpoint between 0 and 1, it may remain in this vicinity for an indefinite period of time before eventually going to either 0 or 1. This "in between" state is called a metastable state. The kind of signal on B described in the example is precisely what was described earlier as a "runt pulse". One of the ways in which a sequential circuit can get into a metastable state (MSS) is if a runt pulse is applied to an input terminal X when the system is in a stable state such that changing X twice is supposed to bring the system to a stable state other than the initial one. Thus, in our example, a runt pulse on A when the system is in 1 000 could not bring the system to a MSS. But a (negative) runt pulse on A with the system in 2 100 might do so. It is also possible for a circuit to get into a MSS as a result of a multiple input change. What may happen here is that the sequential circuit generates a runt pulse internally, which then behaves as did the runt pulse in the previous case. In the Fig. 6.1 example, suppose that starting in 1 110, B and C are both changed. If the circuit delays were so balanced that, at the various gates involved, it appeared that the changes were simultaneous (regardless of whether or not they actually did change at exactly the same time), then the state changes to 1 101, another stable state. If it appeared as though B changed first, then the system would go to state 1 100 (a stable state) and then to 1 101, i.e. the effect would be the same. But suppose that it appeared that C appeared to change first. Then the system would visit state 1 111, an unstable state. If the apparent difference between the times of the changes were sufficiently large, then the system would remain in the 1! 11 input state long enough for y to change. This would send the system to 2 111, and then when the effect of the B change got through, the system would move to state 2 101, another stable state. Thus, depending on the path delays and the actual time of th input changes, the system could end up in either 1 101 or 2 101. The key is the relative arrival times of the B and C signals at the terminals of the AND gate generating the ABC term in the expression for Y presented above in connection with Fig. 6.1. The

A input to that gate is constant at 1, while the B and C inputs are changing in opposite directions. The result may range from a constant 0 output to a relatively long 1 pulse at the output. If the output is a runt pulse of an appropriate width, then the system can enter a MSS. The situation in which a MIC can lead to a MSS can be defined as follows. A MSS may be entered if, starting at some stable state, a MIC is applied such that, depending on the order in which the individual variable changes appear to arrive, one of several different final stable states are reached. In our example table, in addition to the case just described, a MSS may be reached if the initial state is 1 011 and A and C are both changed, or if all 3 inputs are simultaneously changed with the system initially in 1 010. But it is not possible to enter a MSS if, starting in 1 100, B and C are both changed. No MIC change can bring the system to a MSS when the initial state is in row 2 of the flow table. The mechanism for generating the MSS is once again the arrival at the inputs to a gate, of signals changing in opposite directions, while the constant inputs are all of type that do not desensitize the gate (i.e. 0s for ORs and NORs, and 1s for ANDs and NANDs). The immediate! result may be the generation of a runt pulse, which, if it gets into a feedback loop, produces a MSS. Note that all of the y variables for a circuit realizing a SOC function are in feedback loops with an even number of inversions [24]. It has been pointed out above that hazards and critical races can produce runt pulses that, if fed to feedback circuits can also produce metastability. It is well known that, where multiple input changes are produced from independent sources, there is no way to prevent metastable states from being entered. Careful design can minimize the probability of such occurrences, but there is no way to reduce to 0 the probability of a MSS persisting beyond any finite interval. This is a most annoying problem in synchronous systems, where, if the MSS persists for an interval of the order of a clock period, the results can be serious. One solution for synchronous systems that is applicable where the clock source can be controlled, is the "pausible clock" method of Pechoucek [18], whereby the clock pulses are temporarily stopped as long as a detecting circuit indicates that the system is in a metastable state. Another technique relying on a metastability detector has been applied to specific circuits (principally arbiters) [22, 21, 20, 13]. The output of the device is filtered to prevent it from changing while the device is in a MSS. This is a satisfactory solution for self timed systems, since there is no danger of missing a clock pulse. What follows is a discussion of this technique and a generalization that can be applied to any sequential circuit in a self timed system. Fig. 7.1a depicts an arrangement used by Martin [13] in which the output of an arbiter is filtered to protect against metastability. (In Martin’s circuit a similar filter with inverter is also attached to Q to produce the second output of the arbiter.) In this situation, the MSS can be entered only when the system is initially in a stable state in which P=Q=1. This would be the case if inputs A and B are both at 0. Under this condition, R is clamped at ground potential (0 volts, which corresponds to logic value 0) as the NMOS transistor is active. This of course, as a result of the action of the output inverter, makes Z=1. If a transition is made to a state where P=1 and Q=0 (i.e. if B becomes 1), then no change occurs in the output. If instead A is set to 1, with B remaining at 0, then P will change to 0, with Q remaining at 1. Under this condition the NMOS transistor is cut off and the PMOS transistor is turned on, which causes R to rise to the value of Q. Hence Z! switches to 0. If, from the initial condition P=Q=1, both A and B are turned on simultaneously, then we have the possibility of the circuit going into a MSS. This would mean that the voltages at P and Q might both decrease by almost equal amounts and then either remain at some intermediate value between 0 and Vdd, the high voltage corresponding to logic value 1, or oscillate in phase within this range. Eventually, the situation would be resolved either with P returning to 1 and Q falling to 0, or vice versa. In either event, we would not wish the output Z to reflect this condition by fluctuating. The desirable situation is that Z remain at its original value of 1 until resolution occurs, and then it should either remain at 1 or change to 0, depending upon the outcome. This in fact is what happens with the given circuit. As long as VP, the voltage at P, remains greater than Vnt, the threshold voltage for the NMOS transistor, that transistor will continue to hold R at! 0. If VP falls below Vnt, then the NMOS transistor is cut off. If i n addition, VQ rises to at least Vpt (the threshold voltage for the PMOS transistor) above VP, then the PMOS transistor goes on and R will rise to the value of VQ. At this point the system may be assumed to be leaving the MSS with P headed toward 0 and Q toward 1. Thus R will soon go to 1 and hence Z will switch to 0. Thus this circuit filters the P signal to obtain a version at Z that is "clean" even if metastability should occur. The Z signal is, at worst, delayed by the occurrence of a MSS. (A dual of the filter can be applied to circuits where the P and Q signals are both 0 prior to the onset of metastability. such a circuit is shown in Fig. 7.1b. It resembles a circuit embedded in the Q flop resolver [20]. Earlier versions of this general idea, implemented in NMOS technology, are by Stucki [22] and Seitz [21].) This scheme and its dual are intended for, and work well for situations where metastability begins with P=Q. It does not work for other situations. For example, suppose that, starting with P=0, Q=1, with A=B=1, a runt pulse is fed to input A, and that this results in a MSS that is resolved by P returning to 0 and Q to 1. Then it is likely that, during metastability, VP will spend time above Vnt, which would cause VR to be pulled down toward 0 and therefore a glitch would appear at Z. The following variation complements the other schemes in that it can be used whenever metastability begins with P-Q. Since any asynchronous sequential function can be realized with its y variables (i.e. internal variables) generated by cross coupled NAND or NOR gates (as shown in parts a and b of Fig. 7.1) with P-Q in all stable states, this makes it possible to deal with metastability problems in the realizations of all asynchronous functions. The circuit is shown in Fig. 7.1c. First assume that P=0 and Q=1, with A=B=0. The PMOS transistor p will be on , holding R at the value of Q, namely 1, so we have Z=0. Similarly, with P=1 and Q=0, also with A=B=0, the NMOS transistor n will be on, so that R is again connected to Q, making Z=1. Hence in the 2 stable states (we assume that A and B are never both equal to 1), Z=P as desired. Now suppose that with P initially 0 (and Q=1) B is turned on. This causes VP to rise and VQ to fall. VR (initially equal to Vdd less the diode threshold voltage) does not change as long as VP is less than VQ+Vnt since transistor n is cut off and diode 2 prevents current flow out of the node R capacitance through transistor p. It is reasonable to assume, on the basis of various studies of metastability [9, 19], that once the values of VP and VQ have changed to the point where VP<VQ+Vnt, then either there will be no metastability or the MSS will have terminated, and P will quickly bec! ome 0 and Q will become 1. Transistor n will be turned on, allowing VR to fall to the diode threshold voltage as VQ falls to 0. If metastability does occur, then while it lasts, |VP-VQ | will remain below the threshold values of p and n, so that both transistors will remain cut off, leaving VR essentially constant at its initial value. A similar analysis can be made of the situation where, in the initial state, P=1 and Q=0. Thus, as long as we do not allow the input state A=B=1, this circuit should operate properly, preventing the consequences of metastability from reaching the output Z. Note that care must be taken in specifying the electrical parameters of the filter elements. It appears to be feasible to specify diodes on CMOS chips although this does not seem to be a common practice. The operation of this filter has been verified by means of a SPICE simulation An example of the application of this approach is the design of a D latch, shown in Fig. 7.2. Even if the setup or hold time constraints for this latch are violated, which might lead to metastability, no spurious output signals will occur at the Q output. The only effect of getting into a MSS is a delay at the output of uncertain duration. This does not solve the problem for synchronous systems, but it does essentially defang metastability for self timed systems. 8. Critical Races Consider the flow table of Fig. 8.1 (outputs are not shown as they play no role in this discussion). If, starting in state 1 00, B is turned on, then both state variables become unstable, i.e. both Y1 and Y2 change (in this case to 1). If y1 changes first, then the state becomes 4 01. Since this is a stable state, the excitation for Y2 reverts to 0. If the delay in the y2 branch is inertial, then the y2 change is aborted and the system remains in 4 01. If the delay in that branch is pure, then, eventually y2 does change. But then, since Y2 was 0 during the time that the system was in 4 01, y2 reverts later to 0, and so we have oscillatory behavior between states 3 01 and 4 01. It is also possible for a runt pulse to be generated, which might take the system into a metastable state. All of these outcomes must be regarded as faulty behavior. The situation is characterized as a race condition, since several y variables are simultaneously changing values. It is a

critical race, because the outcome is dependent on which y variable "wins the race", i.e. changes first. As was just shown, if y1 wins, then malfunctioning results. If y2 wins, then the system goes to state 2 01, where the excitations on both Y’s remain unchanged at 1, so that the correct final destination, state 3 01, will be reached. If the race ends in a draw, then the system goes directly to the correct final state. Clearly this is an undesirable situation. There are many ways to generate state assignments for the flow table that are free of critical races [24]. Since these may entail an increase in circuit complexity, we might first explore the possibilities of defeating the critical race for the given assignment. Several approaches are possible. First, we might impose a one sided delay constraint, requiring that the delays in the y1 branch exceed those in the y2 branch so as to ensure that the system takes the path through 2 01, which leads to the correct stable state. Or, we could change the next state entry in 1 01 from 3 to 2, so that the logic circuitry is altered to take the system to 2 01 first. This makes the transition a 2 step process. The penalty is some slowdown in operation since there is now no overlap of the time it takes to change the 2 state variables; they change in strictly sequential order. Because the problem for the transition discussed above is solvable ! by either of the two methods outlined above, without changing the state assignment, such a critical race is said to be removable. As is shown next, not all critical races can be handled so easily. Still referring to Fig. 8.1, suppose that, again starting in state 1 00, A is turned on instead of B. Then, once more, both y variables become unstable. We have another race condition, and it is certainly critical, since, if y1 wins, the situation is identical to the one considered previously: the system may fail in one of 3 ways. But now, even if y2 wins the race, the situation is essentially the same, since the intermediate state is 2 10, another stable state. Only if both y variables change simultaneously would the system go to 3 10, the specified final state. Neither of the solutions suggested for the previous example are applicable. This critical race is not removable. Of course, the overall problem is solvable, namely by changing the state assignment to one that is free of critical races,but in this case it would be necessary to use 3 state variables. It is interesting to consider these situations from a circuit point of view. Logic expressions corresponding to the given state assignment are: Y1=A B+Ay1+By1+AB y 2, Y2=A y2+B y2+A By 1+AB y 1 For the removable critical race transition, where A is fixed at 0, these reduce to: Y1=B, Y2=y2+By 1 During this process, while Y2 is a function of y1, Y1 is independent of y2. The corresponding reduced circuit is shown in Fig. 8.2. (Delay elements are shown between each Yi signal and the corresponding yi signal. These do not necessarily represent actual elements, but may simply designate inherent wiring delays.) Observe that y1 changing to 1 blocks y2 from changing to 1. Delaying the change of y1 sufficiently gives y2 time to change and "lock up" the change via the feedback path through the OR gate. Note the resemblance to the corresponding circuit for a steady state essential hazard (Fig. 5.1d). In both circuits there are two paths from the input variable to an AND gate, one direct and uncomplemented, and the other, through a y variable and complemented. The output of the AND gate feeds a second state variable. In the essential hazard case, malfunctioning occurs (i.e. the hazard is manifested) if a pulse is generated at the output of the AND gate. In the removable critical race case, the absence of a sufficiently long pulse at the output of the AND gate causes malfunctioning, since the second y variable is supposed to be turned on and stay on. As indicated above, by making the delay between Y1 and y1 sufficiently long, an adequate 1 pulse can be produced by the AND gate thereby ensuring correct operation. In both cases, if the logic controlling the second state variable reacts to the change in the first y variable before reacting to the input change that caused t! hat change, then improper behavior results (a spurious y change in the case of the essential hazard, and a failure to change in the case of the critical race). Delaying the change in the first y variable by a sufficiently large amount is thus a remedy in both cases. The situation is a bit more complex with respect to nonremovable critical races. For the second critical race discussed above, B remains fixed at 0 and A changes from 0 to 1. The logic expressions are reducible to: Y1=Ay1+Ay 2=A(y1+y 2), Y2=y2+Ay 1 The corresponding reduced circuit is shown in Fig. 8.3. Here we see that, as in the case of Fig. 8.2, there is an inverted and an uninverted path from the input A to AND gate 5 in the circuit generating Y2, the former path passing through y1. But there is also feedback from y2 to the circuit generating Y1, and there are two paths from A to AND gate 1 in the Y1 circuit: one direct and uncomplemented, the other complemented via y2. As in the previous case, y1 changing to 1 blocks y2 from changing to 1. But now, the converse is also true, i.e. y2 changing to 1 blocks y1 from changing to 1. Therefore, delaying the y1 change, which solved the problem for the removable critical race, will not work here because it does not solve the converse problem; once y2 goes on, it prevents y1 from turning on. However, this analysis does point to a solution. What matters is not when each of the y signals actually changes, but when the circuit generating the other y sees the effect of the change. Thus, if the news of the y1 change is prevented from reaching the y2 circuit until y2 has already changed and that change has been locked up, and, conversely if the news of the y2 change is prevented from reaching the y1 circuit until the y1 change has already been locked up, then correct operation will be ensured. The key is to delay not the generation of the y signals, but the propagation of those signals to the other y signal involved in the race. No effort need be made to "fix" the race, only to delay the reporting of the results to specific points. In the current example (referring to the logic expressions for Y1 and Y2) a delayed version of y2 can be used to produce the y 2 term in the Y1 expression, and a delayed version of y1 would be used to produce the y 1 term in the Y2 exp! ression. Specifically, in terms of the circuit , y1 will become stable at the correct value if the delay in the path from A through AND gate 1, delay 2 to the input of OR gate 3 is less than the delay in the path from A through AND gate 5, OR gate 6, delay 7, INVERTER 8, to the input of OR gate 3. Letting di represent the delay in branch i, and assuming wiring delays are included with the associated gate delays, this leads to the constraint: $d1+d2<d5+d6+d7+d8$ or $d8>(d1-d5)+(d2-d7)-d6$ (4) In a similar manner, y2 is guaranteed to stabilize at the specified value if the delay from A on the path through gate 5, gate 6, and delay 7 to the input of gate 6 is less than the delay from A through gate 1, delay 2, INVERTER 4, and gate 5 to the input of gate 6. This generates: $d5+d6+d7<d1+d2+d4+d5$ or $d4>(d6-d1)+(d7-d2)$ (5) (Note that the cancelling out of the d5 terms would not occur if we were distinguishing between the delays in transmitting signal changes of opposite polarities.) The right hand sides of 4 and 5 specify how much we must delay the y2 signal to Y1 and the y1 signal to Y2 respectively. Clearly there is no conflict between these constraints, as d4 appears only in 5 and d8 appears only in 4. Returning to the flow matrix (Fig. 8.1) the process can be examined from another viewpoint. Starting in state 1 10, both Y1 and Y2 change to 1. If y1 changes first, the danger is that Y2 will see that change, i.e. see the system in 4 10. Since Y2=0 in 4 10, the change in y2 is thereby blocked. But, if Y2 sees the y2 change occurs first, it sees the system in 2 10, where Y2=1, so that there is no problem. Similarly, if Y1 sees the transition as going from 1 10 to 4 10 (i.e. if it sees y1 change first), it too will continue to stay at the correct value and the overall operation will be as specified. Unfortunately, this method does not work for all critical race situations. The flow table in Fig. 8.4 is a modified version of the Fig. 8.1 table in that the next state entry in state 2 01 is 4 instead of 3. Now there are two critical races in the 01 column: from 1 to 3 and from 2 to 4. Consider the first of these, where y1 and y2 are both supposed to change from 0 to 1. The technique presented above fails because in both of the intermediate states, Y2=0. We could however defeat this race by delaying the y 1 and y 2 signals to Y2 (with respect to the y1 and y2 signals) in the reduced expression for the 01 column shown below: Y1=1, Y2=y 1y 2+y1y2 But, for the race from 2 to 4, different constraints are required, namely that the y1 and y 2 signals be delayed. It does not appear possible to defeat both races simultaneously with one sided delay constraints. Complications can also occur with constraints arising from the need to defeat essential hazards. The above results can be generalized to races involving more than two variables. A careful inspection of the flow matrix is necessary to determine if delaying the propagation of y signals to circuits generating other y signals will ensure correct excitations for all transitions. This will always be the case for flow table columns with only one unstable state, regardless of how many y variables are contestants in the race. 9. Simulation and Timing Problems Suppose a circuit has been designed that is believed to be free of the kinds of problems addressed here. Before proceeding to the fabrication stage, it is generally considered wise to verify this by carrying out a series of tests with a simulator that can deal with timing as well as logic. This would be done by repeatedly applying to the simulated system a set of tests calculated to check out its behavior in a reasonably thorough manner. If the circuit is not too complex, the test set might be designed to cause the execution of each transition in the flow table describing the desired circuit behavior. Each repetition of the test set would be carried out with a different combination of values for the delays in the various branches. It is generally assumed that upper and lower bounds are available for the delays in each branch. How should the delay configurations be chosen for each run through the test set? Clearly, since each branch delay lies somewhere in a continuous range, it is not even theoretically possible to test for all possible delay values in even one of the circuit branches. An approach that seems plausible is to try all combinations of branch delays in which either the maximum or minimum delay is assigned to each branch. For a large circuit, this would be a formidable task, since, if there are n branches, the test set would have to be applied 2n times. (If we allow both pure and inertial delays in our model for each branch or allow different delay values depending on whether the branch output is increasing or decreasing, then the exponent would have to be multiplied by 2 or 4, compounding the problem significantly.) It is obvious that this approach would be feasible only for circuits of modest size. But the situation is even worse than it would appear from this analysis. At first it might seem as though if a circuit worked properly when, with all other delays fixed, di, the delay in branch i were dim or diM, then it would also work properly for any other value of di between dim and diM. Thus if the above described series of tests were actually carried out, we could be sure that the circuit would have no timing problems as long as the branch delay values were confined to the given ranges (and, of course, assuming that our general model was realistic). But consider now the result presented in Section 3 above with respect to dynamic hazards. It was shown that a dynamic hazard (of any subclass) could be defeated if the delays along a particular path, p1 either exceeded the delays in each of two other paths, p2 and p3 or was exceeded by the delays in each of these other two paths. Suppose now that the branch i is in p1 and that when di is in the neighborhood of its maximum value that the total delay in p1 exceeds the delays in both p2 and p3,! and that when di is in the neighborhood of its minimum value that the total delay in p1 is less than the delays in both p2 and p3. Then the circuit would work properly with di at either its maximum or minimum value, but the dynamic hazard would be manifested for some range of values of di contained within the bounds defined by dim and diM that satisfied neither requirement. Thus, it is not sufficient to simulate using only maximum and minimum values of branch delays. An important consequence of this result is that, while simulation may be a useful tool for checking out circuits, we cannot rely on it to expose all timing problems. It is necessary to understand the various causes of timing failures and to take explicit steps to eliminate or defeat all hazards, critical races or metastable state conditions. For each transition associated with a problem condition, it is necessary to identify various critical paths and to ensure that the total delays along those paths are properly related to one another. 10. Pure and Inertial Delays It is assumed here that all signals are strictly binary, i.e. 2 valued. A pure delay does nothing more to a signal than delay it by the magnitude of the delay. It does not alter the wave form. An ideal inertial delay of magnitude D does not respond to any input change of duration less than D, and its response is delayed by D. Thus, an inertial delay element suppresses positive or negative pulses whose widths are less than D, and otherwise behaves in the same way as a pure delay. Neither type of delay can be realized physically in an exact manner. Even for close approximations to ideal inertial delays, trouble occurs with pulses whose widths are close to the value of D. The outputs in such cases may be runt pulses. Assume for the purposes of our present discussion, that such pulses do not occur. In the preceding material dealing with the generation of glitches associated with hazards of various types, the simplest assumption about the circuit delays is that they are pure. If, however, we assume that some or all of the delays are inertial, then, while the mechanism for glitch generation would not be changed, there would be situations in which glitches would be filtered out by inertial delays in subsequent stages of logic. One might be tempted to infer from this that replacing pure delays with inertial delays can only eliminate malfunctions due to hazards, never causing additional manifestations of hazards. This inference is not valid. The difficulty is that situations can exist in which glitches can cancel out other glitches. If an inertial delay filters out a glitch that cancels another glitch, the result can be an output malfunction or a transition to an incorrect internal state. Such a situation is illustrated in Fig. 10.1 Assume delays: d1=9,d2=7,d3=5,d4=1,d5=5,d6=1. If the delays are all pure, then, following a change of X from 0 to 1 at t=0, a 0 hazard is manifested at the output of the AND gate starting at t=1 and ending at t=5. That is, a 1 pulse of width 4 appears at the point p in the circuit at t=1. It is delayed by branch delay d3, so that it begins at t=6 at the lowest input to the OR gate. If the signal q were fixed at 0, then a 1 hazard would be manifested at point r, the output of the OR gate in the form of a negative pulse of width 2 beginning at t=7. Observe now that the 1 pulse at q overlaps the 0 pulse completely, so that the signal at r remains fixed at 1, as does the circuit output Z. Thus, if all the delays are pure, no hazard is manifested at the circuit output. But now suppose that d3, the branch delay between p and q, is changed from a pure to an inertial delay (with the same magnitude, namely 5). Then the glitch produced by the conflicting inputs to the AND gate, which is of width 4, is filtered out by d3. The q input to the OR gate remains fixed at 0, and so the 1 hazard produced by the OR gate is manifested at r as a 0 pulse of width 2. It is transmitted to Z with a delay of 1 unit of time. Thus, changing a pure delay in the circuit to an inertial delay introduces an output glitch that was not previously present. (Note that the result would have been the same if all of the delays were made inertial.) It follows then that, for an overall circuit, changing pure delays to inertial delays may sometimes introduce faulty signals not previously generated. 11. Conclusions Timing problems in logic circuits are generally caused by different input signals to a gate changing in opposite directions. Depending on path delay values, this may or may not produce a pulse (positive or negative) at the gate output. In some situations (hazards, both combinational or sequential), such a pulse constitutes a faulty response. If transmitted to the circuit output, this results in a transient error. Or such a spurious pulse might be captured in a feedback path associated with a state variable, in which case the system might enter an incorrect state, thereby producing a steady state error. The pulse may be of a marginal nature (with respect to width and or amplitude) i.e. a runt pulse. If such a pulse gets into a feedback path as mentioned above, the result can be that the system becomes metastable for an uncertain time interval. In the case of a critical race, several gates may have such conflicting inputs, each output entering a feedback path, and there! may be an overall feedback path involving all of them. Here faulty behavior results if the pulses at the outputs of the gates receiving oppositely changing inputs are not sufficiently wide. There exist sequential dynamic hazards, which are always MIC logic hazards. They are not, as is the case for essential hazards, inherent in the sequential functions being realized. A method was presented above for defeating all types of dynamic hazards for transitions in both directions through the use of one sided delay constraints. It was further shown that for many critical race situations, one sided delay constraints can prevent malfunctioning. This can sometimes simplify circuit design by allowing the use of simpler state assignments. Asynchronous sequential circuits serve as building blocks for synchronous systems as well as for self timed systems. One usually has methods for estimating upper and lower bounds on branch delays (which include wiring as well as gate delays). Since, essential hazards are associated with almost all nontrivial sequential functions, there is no escaping the need to consider various critical path delays and, where necessary, to insert extra delays in particular paths to ensure that certain constraints are satisfied. One sided constraints on these added delays specify that they must exceed certain sums and differences involving the upper and lower bounds on various branch delays. Since such constraints are inevitable, it is reasonable to consider additional constraints of the same type that may be traded off in return for reducing logic complexity. An example would be the use described in Section 8 of delay constraints that allow the use of state assignments with critical races. They may also be used sometimes to defeat logic hazards rather than to eliminate them by adding logic elements. As shown in Section 3, the same approach can be used to defeat dynamic hazards with delay constraints. Note that sometimes within sequential circuits, input changes to a combinational logic circuit may be possible in only one direction (for example, in the flow table of Fig. 6.1, a transition from 1 110 to 1 111 is possible, but there can be no transition in the reverse direction), so that using delays to defeat static logic hazards is an option. In some cases, although a transition may be possible in the reduced flow table, an inspection of the primitive flow ! table that describes exactly what transitions are possible from any stable state may indicate that in fact the transition cannot actually occur. A flow table inspection procedure was described here for identifying situations in which multiple input changes can lead to a MSS, and where a runt pulse input can lead to a MSS. Although there is no way to prevent the occurrence of metastable states, the consequences in self timed systems can be reduced to the slowing down of the appearances of output changes. This is accomplished by the insertion of output filters. Previous researchers have devised such filters that are applicable under certain circumstances. A variation has been presented here that makes it possible to gain the benefits of such MSS filtering in the realizations of arbitrary sequential functions. Two plausible sounding conjectures about the effects of delay elements have been shown to be invalid. First, although the presence of inertial as opposed to pure delays tends to reduce the presence of spurious pulses, this is not always the case, since a spurious pulse may on occasion cancel out another such pulse of opposite polarity. Second, the fact that a circuit has no timing problems for 2 given values of a particular branch delay does not always mean that there will be no problems for all values of that delay between those values. This latter point has fundamental implications with respect to the possibility of using simulation to verify the absence of timing faults. Rather than relying on simulation, it is better to consider systematically each flow table transition that might involve a critical race or hazard, or which might lead to metastability. Appropriate delay paths should be identified for each case, and it should be determined if, on the basis of a worst! case analysis using given branch delay bounds, the addition of added delays is necessary to satisfy the constraints that ensure proper operation, or if the use of MSS filters is necessary. Acknowledgments My understanding of metastability was enhanced by discussions with Henry Li. I am grateful to Luis Plana for his work in evaluating the MSS filter via SPICE simulations, and for determining how to integrate diodes into CMOS technology. Many valuable comments on a draft of this paper were supplied by Steven Nowick. The research on which this paper is based was partially supported by a grant from the AT&T Foundation.

References [1] Bredeson, Jon G., and Paul T. Hulina, "Elimination of Static and Dynamic Hazards for Multiple Input Changes in Combinational Switching Circuits", Information and Control, V. 20, pp. 114 124, 1972. [2] Bredeson, Jon G., "Synthesis of Multiple Input Change Hazard Free Combinational Switching Circuits without Feedback", Int. J. Electronics, V. 39, No. 6, June 1975, pp. 615 624. [3] Chaney, Thomas J., and Charles E. Molnar, "Anomalous Behavior of Synchronizer and Arbiter Circuits," IEEETC, V. C-22, No. 4 (April 1973), 421-422. [4] Chuang, Henry Y. H., and Santanu Das, "Synthesis of Multiple Input Change Asynchronous Machines Using Controlled Excitation and Flip Flops", IEEETC, V. C 22, No. 12, Dec. 1973, pp. 1103 1109. [5] Eichelberger, E. B., "Hazard Detection in Combinational and Sequential Switching Circuits", IBMJ, V. 9, No. 2, March 1965, pp. 90 99. [6] Huffman, David A., "The Synthesis of Sequential Switching Circuits," J. Franklin Institute, V. 257, No. 3 (March)161-90, and No. 4 (April 1954), 275-303. [7] Huffman, David A., "Design of Hazard Free switching Circuits", JACM, V. 4, January 1957, pp. 47 62. [8] Hurtado, M., and D. L. Elliott, "Ambiguous Behavior of Logic Bistable Systems", Proc. 13th Ann. Allerton Conf. on Circuit and System Theory, Oct. 1975. [9] Kacprzak, Tomasz, "Analysis of Oscillatory Metastable Operation of an RS Flip Flop", IEEE J. Solid State Circuits, V. 23, No. 1, Feb. 1988, pp. 260 266. [10] Lavagno, L., K. Keutzer, and A. Sangiovanni Vincentelli, "Algorithms for Synthesis of Hazard Free Asynchronous Circuits", 1991 Design Automation Conference, June 1991, pp. 302 308. [11] Liu, C. N., "A State Variable Assignment Method for Asynchronous Sequential Switching Circuits," JACM, V. 10 (April 1963), 209-216. [12] Marino, Leonard R., "General Theory of Metastable Operation", IEEETC, V. C 30, No. 2 (Feb. 1981), 1082 1090. [13] Martin, Alain, "From Communicating Processes to Delay Insensitive Circuits", UT Year of Programming Institute on Concurrent Programming, C. A. R. Hoare, (ed), Addison Wesley, 1989. [14] McCluskey, Edward J., "Transient Behavior of Combinational Logic Networks", in Redundancy

Techniques for Computing Systems, Spartan Books, 1962, pp. 9 46. [15] Mead, Carver, and Lynn Conway, Introduction to VLSI Systems, Reading, MA: Addison-Wesley, 1980. [16] Nowick, Steven M., and David Dill, "Synthesis of Asynchronous State Machines Using a Local Clock", International Conference on Computer Design, 1991. [17] Nowick, Steven M., and David Dill, "Exact Two Level Minimization of Hazard Free Logic with Multiple Input Changes", 1992 Int. Conf. on Computer Aided Design, (ICCAD 92). [18] Pechoucek, M., "Anomalous Response Times of Input Synchronizers", IEEE-TEC, V. C 25, Feb. 1976, pp. 133 139. [19] Reyneri, Leonardo M., and Dante Del Corso, "Oscillatory Metastability in Homogeneous and Inhomogeneous Flip Flops", IEEE J. Solid State Circuits, V. 25, No. 1, Feb. 1990, pp. 254 264. [20] Rosenberger, Fred U., Charles E. Molnar, Thomas Chaney, Ting Pien Fang, "Q Modules: Internally Clocked Delay Insensitive Modules", IEEETC, V 37, No. 9, September, 1988, 1005 1018. [21] Seitz, Charles, "System Timing", Chapter 7 in [Mead 1980]. [22] Stucki, M. J., and J. R. Cox, Jr., "Synchronization Strategies", Proc. Caltech Conf. VLSI, Pasadena CA, Jan. 1979, pp. 375 393. [23] Tracey, James H., "Internal State Assignments for Asynchronous Sequential Machines," IEEE-TEC, V. EC-15, No. 4 (Aug. 1966), 551-560. [24] Unger, Stephen H., Asynchronous Sequential Switching Circuits, New York: Wiley-Interscience, 1969, (Reissued by Krieger, Malabar, Florida, 1983.) [25] Unger, Stephen H., "Self Synchronizing Circuits and Non Fundamental Mode Operation", IEEETC, V. C 26, No. 3, March 1977, pp. 278 281. [26] Unger, Stephen H., The Essence of Logic Circuits, Prentice Hall, 1989.