Defect-Oriented Testability for Asynchronous IC's

MARLY RONCKEN

Invited Paper

For a CMOS manufacturing process, asynchronous IC's are similar to synchronous IC's. The defect density distributions are similar, and hence, so are the fault models and fault-detection methods. So, what makes us think that asynchronous circuits are much harder to test than synchronous circuits? Because the effectiveness of best known test methods for synchronous circuits drops when applied to asynchronous circuits? That may very well be a temporal hurdle. Many test methods have already been reevaluated and successfully adapted from the synchronous to the asynchronous test domain. The paper addresses one of the final hurdles: I_{DDQ} testing. This type of test method, based on measuring the quiescent power supply current, is very effective for detecting (resistive) bridging faults in CMOS circuits. Detection of bridging faults is crucial, because they model the majority of today's manufacturing defects. I_{DDQ} fault effects are sensitized in a particular state or set of states and can only be detected if we stop the circuit operation right there. This is a problem for asynchronous circuits, because their operation is self timed.

In the paper, we quantify the impact of self timing on the effectiveness of I_{DDQ}-based test methods for bridging faults, and propose a Design-for-Test (DfT) approach to develop a low-cost DfT solution. For comparison, we do the same for logic voltage testing and stuck-at faults. The approach is illustrated on circuits from Tangram, the asynchronous design-style employed at Philips Research, but it is applicable to asynchronous circuits in general.

Keywords*—*Asynchronous circuits, bridging faults, DfT, I_{DDQ}, self timed, stuck-at faults, testability.

I. Introduction

From a design perspective, asynchronous circuits promise advantages over synchronous realizations in a number of application areas [1]. They can be designed for average-case rather than worst-case performance and have a higher degree of modularity because there is no global clock. Instead, there is a self-timed protocol that automatically puts inactive circuitry in standby mode, which results in lower power dissipation and better electromagnetic compatibility.

Manuscript received February 2, 1998; revised September 4, 1998.

The author was with Philips Research Laboratories, Eindhoven, The Netherlands. She is now with Strategic CAD Laboratories, Intel Corporation, Hillsboro, OR 97124–5961 USA (e-mail: mroncken@ichips.intel.com).

Publisher Item Identifier S 0018-9219(99)00889-0.

From a test perspective, the essential distinction between asynchronous and synchronous circuits is the self-timed computation paradigm in the first versus a lock-stepped, globally clocked computation in the latter. Common techniques for voltage and current testing, like scan and I_{DDQ}, are applied when the state is quiescent. Self timing can get in the way and reduce the effectiveness of these test techniques because it offers fewer quiescent states. But self timing can also work to the advantage, especially in a request-acknowledge implementation style such as handshake signaling, because of its self-checking properties [2], [3].

From a technology perspective, there is no distinction between manufacturing asynchronous or synchronous CMOS circuits, and hence the manufacturing defects are similar for both design styles. Most defects in modern CMOS processes can be categorized as shorts or opens, with a significantly smaller probability for opens [4]. The probability for shorts is highest in the metal and polysilicon interconnect layers, followed by gate oxide shorts. Research has shown that these shorts can be modeled as (resistive) interconnect bridging faults and transistor bridging faults [5], [6].

So, the fault models are the same, but the effectiveness of best known test methods may be different when shifting from synchronous to asynchronous circuits. Since the survey in [3], many best known test methods have been reevaluated and adapted from the synchronous to the asynchronous test domain. Testability for asynchronous circuits is getting mature. Synchronous partial scan solutions are built into asynchronous synthesis methodologies [7]–[10], and also self-timed solutions are available [11]. In addition to these externally controlled test structures, Built-In Self Test (BIST) has been applied to asynchronous micropipeline designs [12]. The logic redundancy in the speed-independent design style has been tackled [13]. And last but not least, automatic test pattern generation can be done for stuck-at faults [14], delay faults [15], [16], and bridging faults [17].

And yet, this paper will not be the follow-up survey to [3], because we feel that the most essential element in testing asynchronous circuits has not been addressed properly. That element is the self timing or autonomy which is present in the computation model of every asynchronous design methodology, and even in some of the new synchronous design styles that we see today. So, rather than giving a general overview of how best known test methods were successfully adapted from the synchronous to the asynchronous domain, we follow up on [17] and focus on self-timed test issues and solutions.

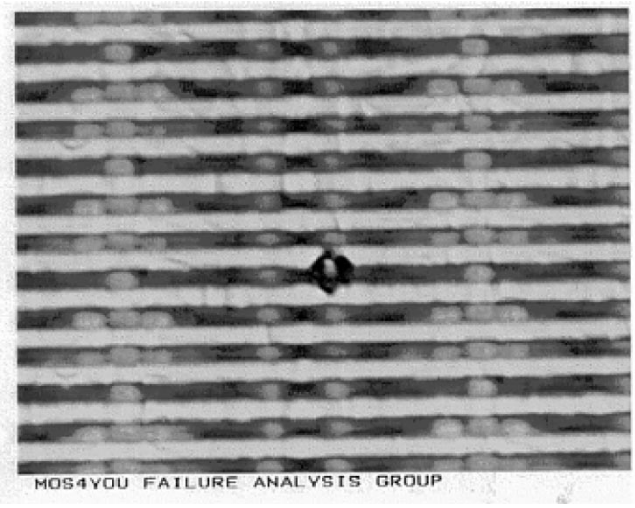
The organization of the paper is as follows. Section II is an introduction to defect-oriented fault modeling and testing, where we build a fault model for the defect types that are common in modern CMOS manufacturing processes, explain the fault effects and also the detection mechanisms using voltage and I_{DDQ} testing. In Section III, we analyze the characteristics of self-timed design and their implications on the test quality and costs. Section IV is the technical core of the paper. Here, we quantify the lack of quiescent states in terms of lost fault coverage. We classify the essential quiescent states by grading undetected faults, and develop a Design-for-Test (DfT) approach to make these states quiescent during test. We demonstrate this approach for a small circuit and three large-scale IC's developed in Tangram, the asynchronous design method used by Philips [18], [19].

II. Defect-Oriented Fault Modeling and Testing

The purpose of testing manufactured very large scale integration (VLSI) circuits is to check whether the expected quality and reliability of the circuit have been realized. Assuming that the correctness of the IC design and of the underlying physical processing models have been established, the origins of functional failures and of diminished performance and reliability are defects introduced during the manufacturing process. In a mature manufacturing process, the vast majority of such failures originate from so-called spot defects that are caused by local process disturbances, such as a short between metal wires caused by a contaminating particle sticking to the positive photoresist before the exposure step in the metal lithography: see photograph in [20, Fig. 3]. Fig. 1 shows another photograph of a spot defect in the form of a short between two metal interconnect lines. By targeting the fault effects originating from spot defects, we have a way to predict and control the effectiveness and efficiency of testing procedures.

A. Fault Modeling

Various methods and tools have been developed to model spot defects and analyze the resulting local deformations in the IC structure to derive the fault effects [20]–[23]. This approach, known as inductive fault analysis, is the best known method for defining realistic fault models for testing manufactured circuits. Inductive fault analysis for modern (i.e., positive photoresist) CMOS technologies has shown that most spot defects can be



(a)

(b)

Fig. 1. Scanning electron microscope photos of a short between conducting lines: (a) overview and (b) detail. (Philips Electronics Nijmegen, MOS4YOU.)

categorized as resistive shorts between conducting lines, some as gate-oxide shorts, and relatively few as opens in conducting lines and vias [4], [24].

In this paper, we focus on the first two categories, resistive shorts between conducting lines and gate-oxide shorts. In the first category, we consider resistive shorts between conducting nodes that, based on connectivity, have a high probability of being physically close to each other when fabricated. For gate-oxide shorts we use the six-transistor short model introduced in [25], which considers all (pairwise) resistive shorts between the four MOS transistor terminals: gate, source, drain, and substrate or bulk (V_{DD} for p-MOS and V_{SS} for n-MOS transistors). Although this is an oversimplification because gate-oxide shorts may exhibit nonlinear instead of simple ohmic behavior, it can be shown that I_{DDQ} tests targeted for the six-transistor model will also cover gate-oxide shorts [6]. We will discuss I_{DDQ} in Section II-B.

The resistances for shorted interconnect nodes are typically below 500 Ω, but measurements by [5] on process-

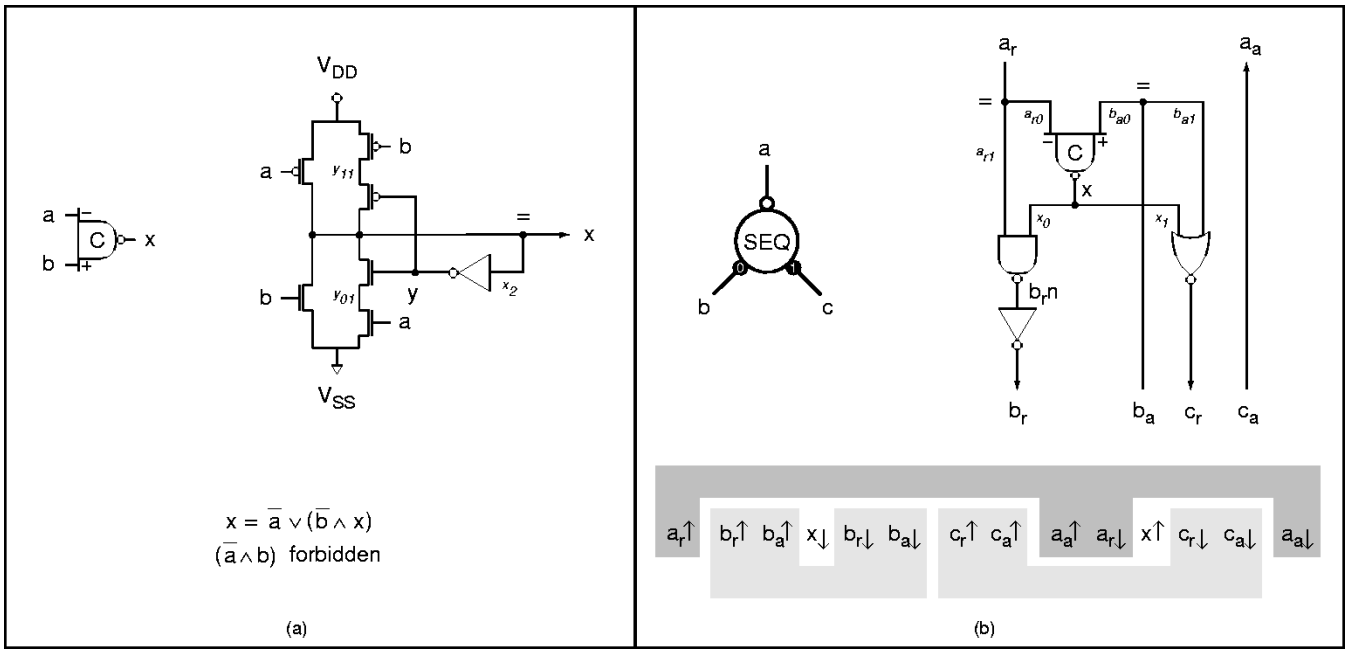


Fig. 2. Icon, schematics, and behavior for (a) asynchronous latch, or (generalized) C-element, and (b) asynchronous control circuit SEQ for sequential execution. The six transistors in the state-holding part (keeper) of the C-element are of minimal size. The asynchronous operations are controlled by handshaking, using request and acknowledge signals. The • and ◦ symbols in the SEQ icon identify, respectively, the requesting side and the acknowledging side of the handshake channels a, b, and c, and the numbers for b and c indicate the sequential order, i.e., first b then c. The implementation of SEQ is based on four-phase handshake signaling, as indicated by the grey-colored regions in the cycle behavior. The "=" symbols in both schematics indicate isochronic forks, with relative timing assumptions for the forked transitions. For the isochronic fork in b_a, the assumptions are: $b_{a0}\downarrow$ arrives at the C-element before $a_{r0}\downarrow$, and $b_{a1}\uparrow$ arrives at the NOR gate before $x_1\downarrow$. Both assumptions are easy to implement, because the transitions are separated by at least one internal gate delay. Similar assumptions hold for the isochronic fork in a_r.

related defect monitoring wafers also report small percentages in the range of 500 Ω to 20 kΩ that vary from batch to batch. Measurements by [26] on four gate-to-source transistor shorts show values ranging from 800 Ω to 4.7 kΩ. Compared to the on-impedance of an MOS transistor, 500 Ω is low and 20 kΩ is high. To make sure that our fault evaluation is representative for the complete range of short resistances, we will model both a low-resistive value (of 100 Ω) and a high-resistive value (of 10 kΩ) for each short. We call the resulting resistive short models bridging faults.

1) Bridging Faults: In our analysis, the bridging faults are generated from the circuit schematics. We first lump nodes and transistor terminals that alias the same signal into a single node, and then generate low and high resistive shorts for the lumped nodes. As an example, consider the asynchronous control circuit SEQ in Fig. 2(b). The nodes in this component are likely to be kept together on silicon because the connectivity to other components is very sparse. Therefore, we consider all bridging faults between the 14 (lumped) nodes:

- power, and ground or bulk: $V_{\rm DD}$, $V_{\rm SS}$;
- interconnect: a_r, a_a, b_r, b_a, c_r, b_rn, x;
- additional nodes inside the C-element: y, y_{01}, y_{11};
- two additional nodes inside the NAND and NOR.

This gives a total of 91 low-resistive 100-Ω bridging faults and 91 high-resistive 10-kΩ bridging faults.

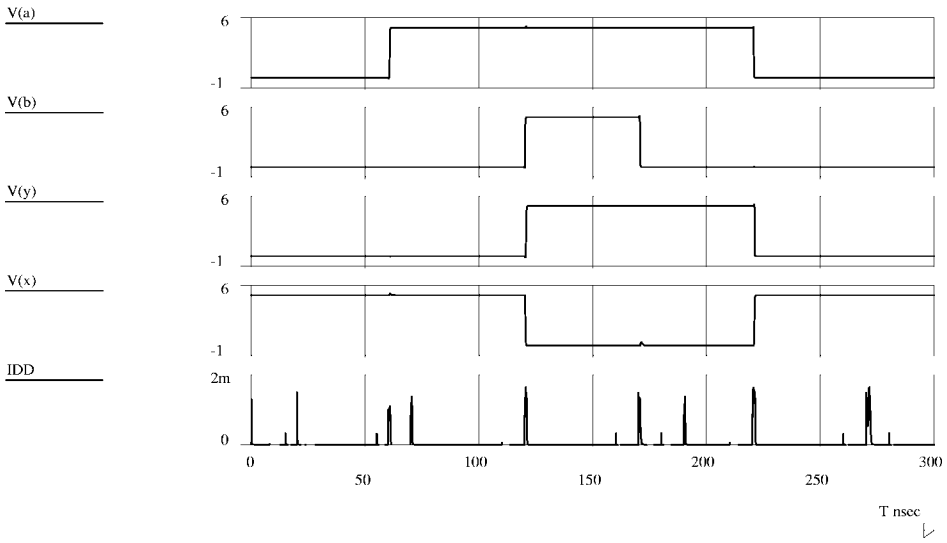
The probability of these bridging faults can vary significantly for different realizations of the module:

- power or ground bridges have a higher probability in technologies that use power and ground lines for shielding in interconnect layers;
- two nodes inside different (logic) gates are more likely to bridge when the gate transistor networks are combined into a single complex gate;
- a metal or polysilicon interconnect node outside a (logic) gate very unlikely bridges to an internal node in the diffusion layer, but the probability increases when the latter is routed inside the gate via metal or poly.

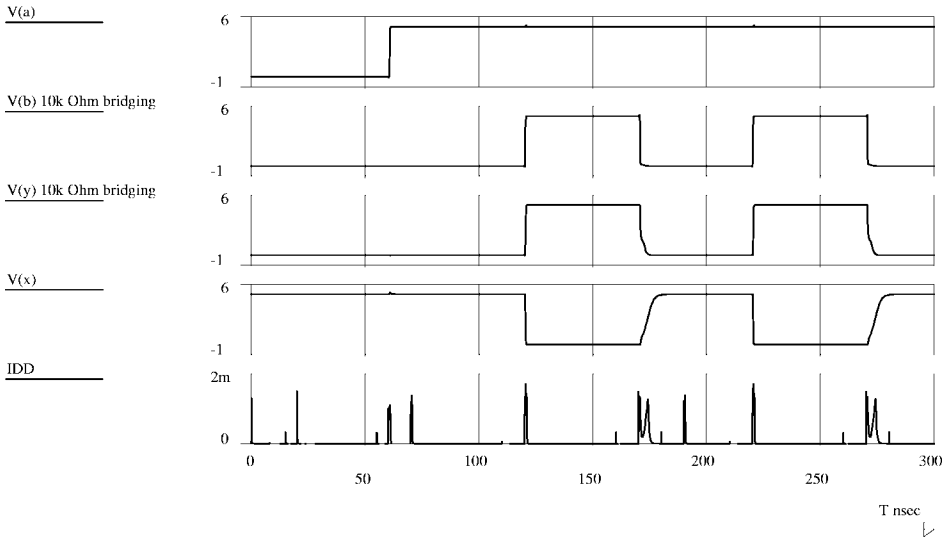
By considering all 91(×2) bridging faults, we get a better idea for the test implications of changes in the realization. Our objective is to develop a DfT strategy for bridging faults that are hard to detect due to the self-timed operation of the circuit.

B. Testing Bridging Faults

Bridging faults can affect the behavior of a CMOS circuit in several ways. The most salient fault effect for testing is a high quiescent power supply current, or $I_{\rm DDQ}$, measured for opposite (logic) voltage values across each of the bridged nodes. For today's technologies, the faulty $I_{\rm DDQ}$ value is typically several orders of magnitude higher than the value of the corresponding transistor leakage current in the fault-



(a)



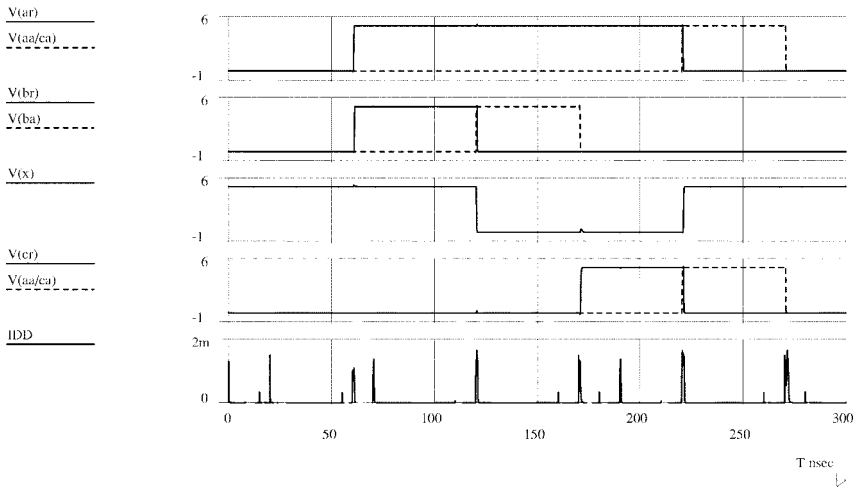
(b)

Fig. 3. Feedback fault effect for a 10-kΩ bridging fault in the C-element between interconnect node b and keeper node y [circuit schematics in Fig. 2(a)]. The operation of the C-element is as performed in the context of control circuit SEQ, with a quiescent state every 50 ns: (a) fault-free behavior and (b) for 10-kΩ bridging. Because node y is weakly driven by the keeper, the bridge to b easily overwrites y, causing the keeper forward drive to flip and invert the originally conflicting voltage driven onto node y. This happens, for instance, in the time interval 150–200 ns. In this way, any voltage conflicts across the bridge are resolved. The fault can be detected by logic voltage testing, but not by I_{DDQ} testing.

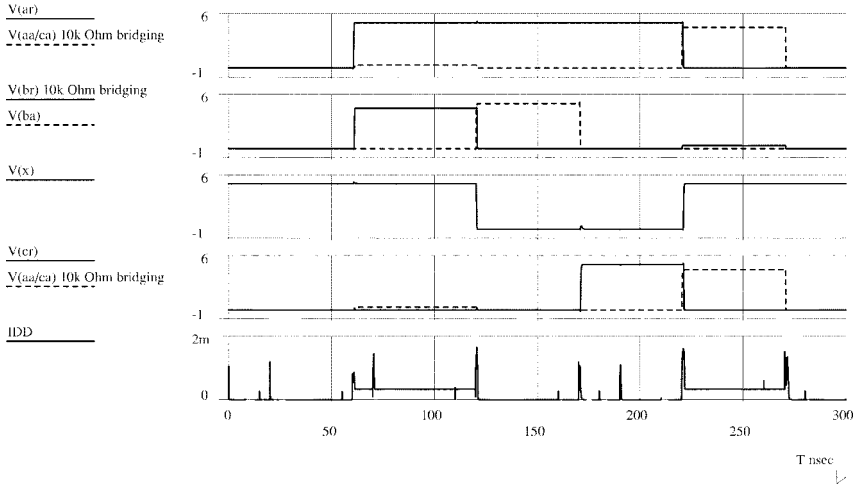
free circuit [27]. Whether or not this will still be the case for deep submicron CMOS technologies the next ten years is unclear [28], but various approaches are being developed to reduce the leakage current and preserve the ability for I_{DDQ} testing [29], [30].

Bridging faults in combinational circuits become I_{DDQ}-detectable by driving conflicting voltages across the bridge, even when the bridging fault creates a feedback path that causes the circuit to oscillate [31]. In sequential circuits, however, the bridging fault may create a feedback path that eliminates the conflict by overwriting or disconnecting the voltage drives across the bridge. In such cases the fault cannot be detected by I_{DDQ} testing, and voltage-based test methods are needed [32]. An example is shown in Fig. 3 for a bridging fault between the interconnect node b and keeper node y in the C-element. Because the keeper transistors are of minimal size, b easily overwrites the keeper state via the bridge, even for a bridge resistance as high as 10 kΩ. Thus, any voltage conflict across the bridge is resolved and I_{DDQ} testing will not detect the fault.

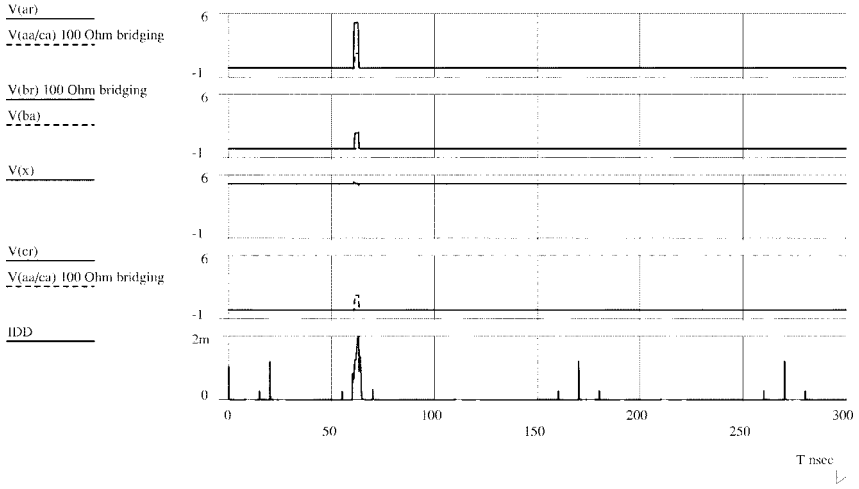
Self-timed execution, typical for asynchronous circuits, may lead to longer bridging-induced feedback paths than normally encountered in synchronous circuits. For example, a low-resistive bridging fault between the nodes b_r and a_a in circuit SEQ creates a feedback path from b_r rising to a_a rising (by the bridge) to a_r falling (by the self-timed



(a)



(b)



(c)

Fig. 4. Fault effects for a bridging fault between the interconnect nodes b_r and a_a in control circuit SEQ: (a) fault-free behavior and (b) for 10-kΩ bridging and (c) for 100-Ω bridging. The circuit [see Fig. 2(b)] is quiescent every 50 ns. The 10-kΩ bridge leads to slightly reduced voltage levels which can be detected by I_{DDQ} but not by voltage testing. The 100-Ω bridge creates a global feedback path that resolves the voltage conflict across the bridge, but changes the logic behavior. Similar to the local feedback created by the bridging fault in the C-element as shown in Fig. 3, the fault effect can be detected by logic voltage testing but not by I_{DDQ}. In conclusion, detection of a bridging fault between b_r and a_a requires logic voltage testing to catch the lower-resistive fault effects and I_{DDQ} testing to catch higher-resistive fault effects.

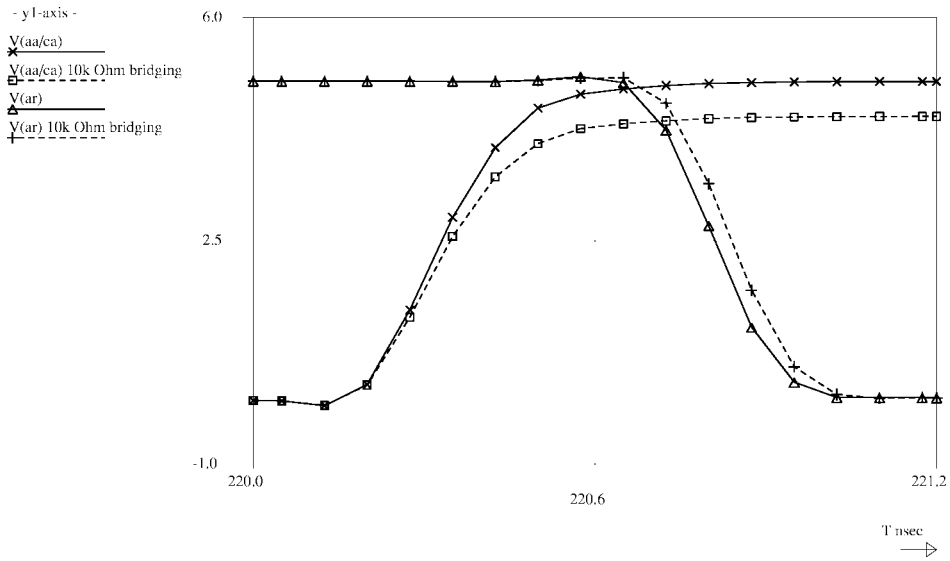


Fig. 5. Detail of the 10-kΩ bridging fault simulation in Fig. 4(b), showing the increased path delay from a_a rising to a_r falling compared to the fault-free situation. For high-resistive bridging faults, such as in this case, the changes in propagation delay typically are marginal, and the circuit will function correctly.

environment) to b_r falling (by SEQ) to a_a falling (by the bridge). This is reflected in Fig. 4(c). The fault can be detected by logic voltage testing, but not by I_{DDQ} because the feedback path resolves the original bridge conflict for b_r rising and a_a low by making both nodes low. For a high resistance, the conflicting voltage values across the bridge will slightly degrade, but the functionality of the circuit is not affected. In this case, the fault can be detected by I_{DDQ} but not by logic voltage testing, as is shown in Fig. 4(b).

1) Dynamic Fault Effects: Logic voltage failures and elevated I_{DDQ} are static fault effects. Bridging faults can also change the dynamic behavior of the circuit, for instance by increasing some propagation delays. Fig. 5 shows a small increase in the propagation delay from a_a rising to a_r falling due to the 10-kΩ bridging fault between nodes b_r and a_a in circuit SEQ. Studies of delay faults due to bridges show that large changes in gate delays are not very likely [33], [34]. Typically, the gate output transition slows down by only a limited amount before a logic voltage failure results, but minor speed ups are also possible. Test techniques for delay faults typically target path delays [35] which are usually larger and will therefore miss most of the high-resistive bridging fault delays. An exception is the delay test method proposed in [36], which enables detection of relatively small increments in gate delays, and also targets multiple gate-delay faults reflecting spill-over effects by resistive bridging faults from inside a gate into neighboring gates. Still, slightly increased or decreased delays that do not cause immediate failures cannot be detected by delay fault test techniques, while the corresponding current-drive fault effects usually can be detected by I_{DDQ} testing.

2) Reliability: The reverse side of the medal for being more successful than other test methods in detecting faults that do not visibly affect the functionality of the circuit is I_{DDQ}'s nomination for yield loss [37]. There is strong evidence, however, that these so-called benign I_{DDQ} failures can become living time bombs in the field. Experiments reported in [38] indicate that the resistances of gate-oxide shorts have a tendency to change with time, and the simulation results in [33] show that the circuit operation can be very sensitive to such changes. In other words, benign gate-oxide shorts can reduce the mean time to failure as well as the lifetime of an IC. In addition, degraded voltage levels due to bridging faults make the circuit more vulnerable to noise, which can lead to intermittent failures not anticipated by the noise calculations for the fault-free design. Therefore, in this paper we take the position that a benign bridging fault, I_{DDQ}-detectable or not, should be subjected to a reliability analysis to judge whether it needs detection or not.

III. Self-Timed Test Issues

From now on, we focus on the (logic) voltage and I_{DDQ} detection of static fault effects in asynchronous circuits. The Q in I_{DDQ} is for quiescent, and although both detection methods are applied when the circuit state is assumed to be stable, or quiescent, the detection characteristics for each method are different.

- I_{DDQ} failures can be observed directly, whereas logic failures usually need to be propagated to latches and output pins to become observable.
- I_{DDQ} failures typically do not propagate with state transitions, whereas logic failures typically do.

This suggests that the effectiveness of I_{DDQ} highly depends on the set of quiescent states available for detection, whereas this seems to be less the case for logic testing. Because self timing typically reduces the set of quiescent states, the effectiveness of I_{DDQ} may be too low for asynchronous design styles, unless adequate DfT measures

are taken. As such, DfT for bridging faults in asynchronous circuits may be more expensive than DfT for typical logic fault models, like stuck-at faults. To quantify how much more expensive, we evaluated the DfT requirements and costs for both stuck-at and bridging faults, which is presented in Section III-B.

The outline of this section is as follows. First, we give a more detailed explanation of why quiescent states are important for detecting bridging faults and why they are sparse in self-timed operations. Then, we illustrate why the obvious solution by elimination of self timing during testing, using full-scan DfT, is too expensive. The conclusion of this section is that we need a more tailored DfT approach to enable high-quality, low-cost bridging fault detection.

A. Quiescent States

For high-resistive bridging faults, often the only visible fault effect is an elevated I_{DDQ}. Typically the current-drive fault effect is sensitized in a particular state or set of states and can only be detected if we stop the operation right there. In a truly synchronous circuit the state transitions in the circuit operation are controlled by one or more external clocks, and stopping the operation in the current state is not a problem. With self-timed state transitions the granularity for stopping the operation externally becomes coarser, and the capability for detecting current-drive fault effects thus becomes weaker. Without DfT for creating additional quiescent states, the effectiveness of I_{DDQ} testing for asynchronous circuits may become too low to guarantee good test quality. Quantitative analysis for the Tangram design style has shown that the test quality without DfT is indeed too low: see [17] or Fig. 9 in the present paper.

B. Costs for Eliminating Self Timing Using Full-Scan DfT

A general DfT solution would be to eliminate all self timing when testing the circuit, which can be done by cutting all internal feedback loops using full-scan DfT. This works for stuck-at faults as well as for bridging faults, but it can be relatively expensive due to the use of distributed control.

- While clocked designs usually have centralized control, asynchronous design methods typically use distributed control. The number of latches is larger in a distributed implementation. Full-scan DfT techniques, in which all latches are modified, are thus more expensive for asynchronous designs.
- Asynchronous circuits typically use set-reset type latches (cf. Fig. 2) that need more expensive scan modifications than clocked data latches.

Table 1 shows how expensive this can become in practice. Presented are three asynchronous industrial benchmarks, designed and fabricated by Philips using Tangram [18], [19]. Design DDD is a low-power (single-rail) DCC error detector [39], [40]. The 80C51 microcontroller (originally from Intel) is a first prototype of the asynchronous version designed by Philips Research and the low-power division of Philips Semiconductors Zürich [41]. The third design is a low-power asynchronous ADPCM Speech Codec according to DECT standard.

Table 1
The Ratio of Latches in Self-Timed Control Escalates for Full-Scan DfT

IC	Transistors		Latches		Extra (scan)latches	
	total	control	total	control	data	control
DDD	20k	40%	894	53%	0%	53%-107% (477-954)
80C51	40k	40%	1231	68%	0%	68%-136% (840-1680)
ADPCM	45k	20%	1496	28%	0.1%	28%-55% (413-826)

The last column in Table 1 shows how the ratio of latches in the self-timed control escalates when full-scan DfT is used. The percentages are relative to the original total latch set in the third column. The extra (clocked) data-path latches in the full-scan solution are real percentages taken from [10]. The percentages for extra control latches are based on [7] and [13]. The solution in [7] with one additional scan latch per set-reset latch gives the lower bounds, because it cannot generate all test pattern combinations, and extra dummy latches may be needed to support a given test suite. In the worst-case scenario this may lead to one additional scan latch plus dummy latch for each set-reset latch, similar to the dynamic scan solution in [13], which explains the upper bounds.

Because the full-scan area overhead was unacceptable even for the lower-bounds, the designs were made partially scannable [8], [10], [17]. For the ADPCM design we used the tailored DfT approach discussed in Section IV, which is geared towards creating a small but sufficient collection of quiescent states to achieve high test quality.

IV. Tailored Testability for Self Timing

In this section, we develop a DfT method tailored to self-timed test issues in asynchronous circuits, and based on fault grading. This method will enable high stuck-at and bridging fault coverage for relatively low costs compared to standard full-scan methods.

We start with fault analysis to determine which faults can escape detection. Undetected faults are subjected to a reliability analysis to determine whether detection is really needed. If detection is needed, we do inductive fault analysis to find out whether the fault probability can be reduced by using an alternative realization. For those faults that still need detection, we find additional quiescent states for which the fault effect becomes visible and develop DfT to create these states.

To quantify the DfT needs for bridging faults in relation to stuck-at faults (which may need fewer quiescent states), we follow the above procedure for both fault models. We use circuit SEQ to illustrate this procedure and the resulting DfT method developed for Tangram. The analysis is supported by analog fault simulations using in-house tools based on SPICE [42] and a 0.8-μm CMOS cell library with 5-V power supply. Simulation setup and results are described in Section IV-A. Section IV-B focuses on

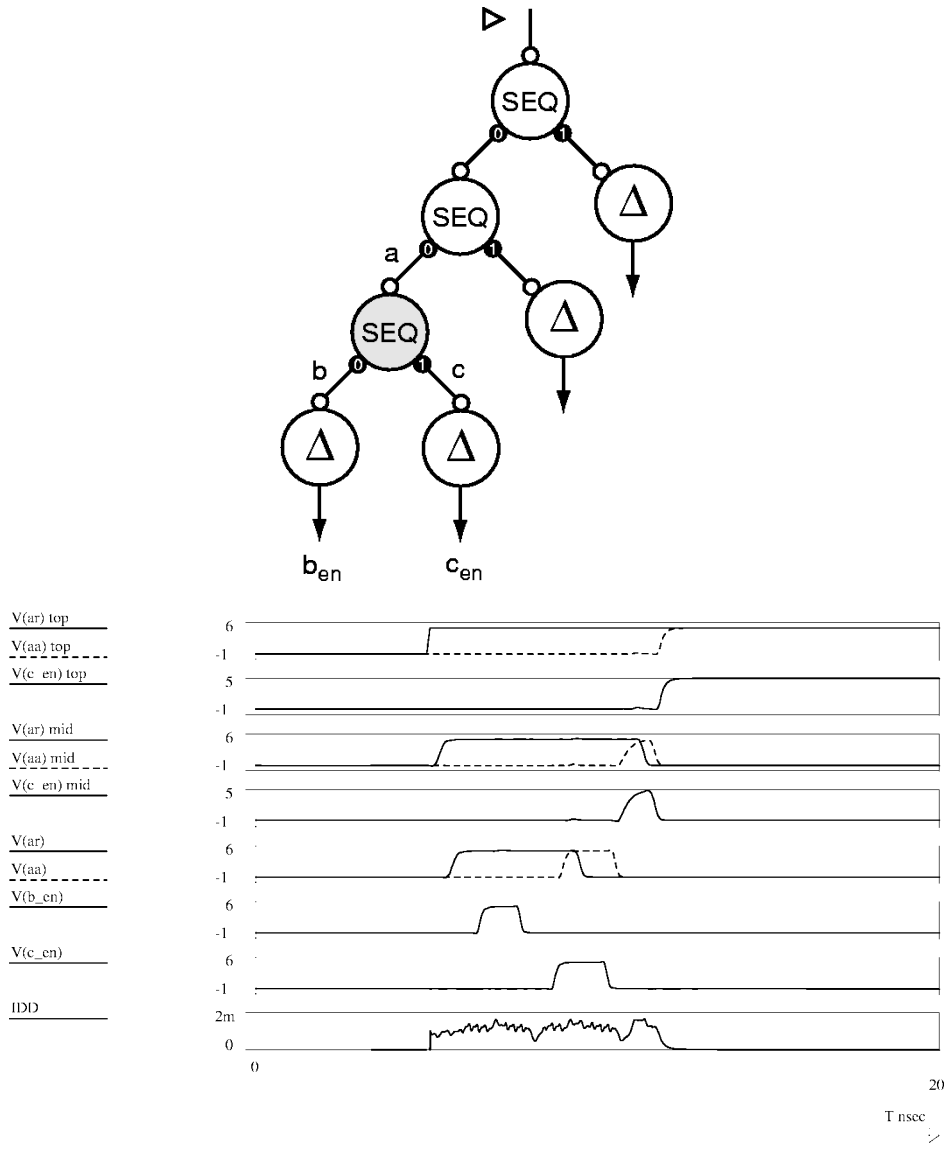


Fig. 6. Simulated configuration and fault-free behavior for three control components SEQ driving four data paths Δ (see also Fig. 7). Faults are injected in the grey-colored SEQ component, and the fault effects are tested in the two initial and final quiescent states.

the collection of faults that initially escape detection, and Section IV-C presents the resulting DfT solution and costs.

A. Fault Analysis Setup and Results for Circuit SEQ

As an example, we use the configuration in Fig. 6 to analyze the testability of the grey-colored SEQ component, for which we inject one fault at a time and compare the resulting behavior against the fault-free behavior. Each simulation starts at the topmost SEQ by raising the request for topmost channel a, and ends when the corresponding acknowledge rises. The start-up first causes the middle SEQ to complete its operation, before the topmost data-path is enabled. The middle SEQ first lets the grey-colored SEQ sequentially execute the data-path operations for b and c, before enabling the middle data-path operation.

1) Data-Path Enable Signals: The data paths in Tangram use single-rail data encoding, as in clocked data paths. Fig. 7 shows the interface between handshake control and

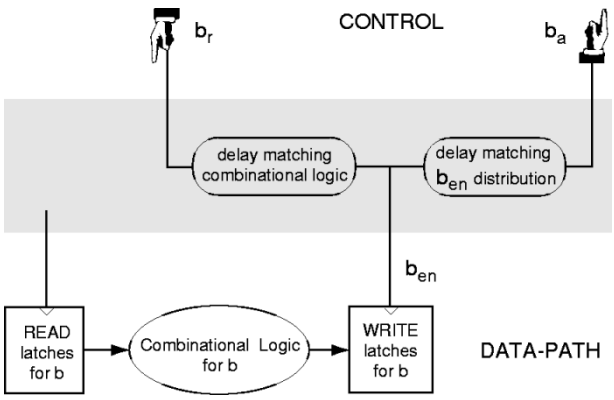


Fig. 7. Handshake interface between the control and a single-rail data path. A four-phase handshake over b_r and b_a generates a pulse on $b_{\rm en}$, which makes it possible to capture the results from the combinational logic in the level-sensitive latches in a way similar to clocked data paths. A circuit can have many data paths that may share combinational logic and latches but typically have their own handshake control channel.

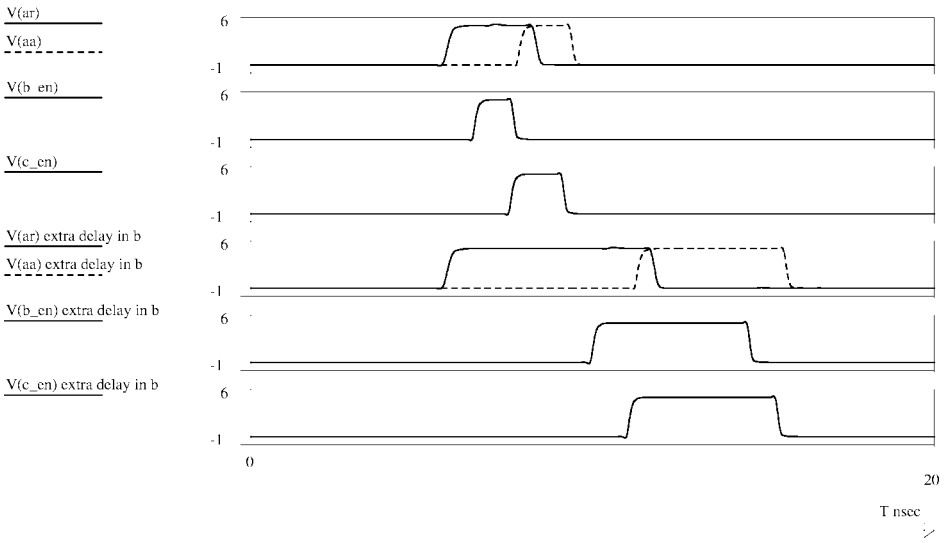


Fig. 8. Two possible fault effects for branch b_{a1} stuck-at-0 [see Fig. 2(b)]. This input stuck-at fault violates the isochronic fork assumption "$b_{a1}\uparrow$ arrives at the NOR gate before $x_1\downarrow$," which is an indication for potential hazardous behavior. The hazard in this case is interference between the data-path operations for b and c, which are supposed to be nonoverlapping. Depending on the delay for the operation in b versus c, the fault effect changes from "probably benign" when $b_{\rm en}$ and $c_{\rm en}$ are hardly overlapping (top three windows) to "likely hazardous" when there is significant overlap (bottom three windows).

data path for channel b in our simulation setup. The handshake request signal b_r is delayed to match the execution time for the combinational logic in the data-path operation. The delayed request signal, now referred to as data-path enable signal $b_{\rm en}$, latches the results of the operation into the data-path write latches. To safely generate the handshake acknowledge b_a, which is supposed to signal that the corresponding data-path phase has completed, there is an additional delay matching for the local distribution of $b_{\rm en}$. A four-phase handshake on b generates a pulse on $b_{\rm en}$ to open and close the level-sensitive data-path latches just like in a clocked data path.

2) Fault Detection: The overall simulation offers exactly two quiescent states: the initial state before start up (at 0 ns) and the final state (at 20 ns). Because faults may affect the initial state, we first analyze the possible initial states for the faulty circuit and start a separate fault simulation run for each of these (at most two runs per fault were needed). Detection is possible in both the initial and final state, where we test for elevated $I_{\rm DDQ}$ and inspect the logic voltage of the start-up acknowledge signal, data-path enable signals, and data-path latches. We assume that whenever the data paths are enabled in the wrong order or either one is skipped or executed more than once, then at least one of the data-path latches will have a faulty logic value. In Section IV-C2, we show how this assumption is realized.

3) Fault Injection and Results: Fig. 9 shows the simulated fault coverage, when injecting all 14 input and 16 output stuck-at-1/0 faults, plus a subset of 16($\times$2) bridging faults, from $V_{\rm DD}$ and $V_{\rm SS}$ to the set of output stuck-at nodes:

- input stuck-at nodes: a_{r0}, a_{r1}, b_{a0}, b_{a1}, x_0, x_1, x_2;
- output stuck-at nodes: a_r, a_a, b_r, b_a, c_r, b_rn, x, y.

A node stuck-at fault is modeled by disconnecting the node from the driving net and connecting it (via a 100-Ω bridge) to either $V_{\rm DD}$ in the case of a stuck-at-1 or $V_{\rm SS}$ in the case of a stuck-at-0. The explanation for the low coverage[1] is similar for both fault models: the self-timed behavior offers too few quiescent states to distinguish fault-free from fault-affected behavior.

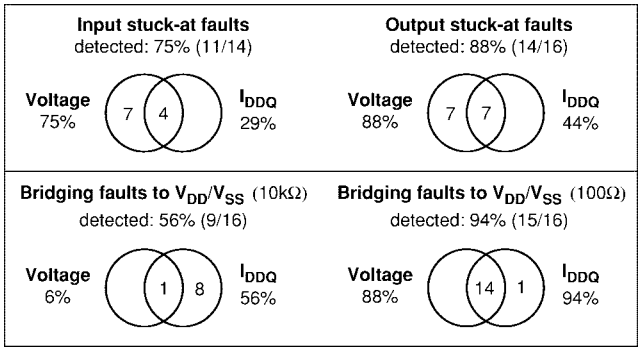


Fig. 9. Simulated fault coverage for control component SEQ.

B. Classification of Undetected Faults into Quiescent States

At first sight, it comes as a surprise that voltage detection also requires intermediate quiescent states. However, from earlier studies we know that stuck-at faults can lead to premature transitions [43] and thus xcreate races between signals that may or may not turn out hazardous. An example is given in Fig. 8, showing two possible fault effects for the same input stuck-at fault, one likely benign the other likely hazardous. Branch b_{a1} stuck-at-0 causes a premature rising transition $c_r\uparrow$, enabling data path c while data path b is still active. Depending on the data-path delays, the pulses for $b_{\rm en}$

[1]The figures are somewhat different from those presented in [17] because we use a slightly different library.

and $c_{\rm en}$ may or may not overlap. Although the fault effect is not *per se* visible when tested, it is noise sensitive and thus a reliability risk which needs to be targeted by DfT.

The fault can be detected independent of the data-path delays by creating an additional quiescent state between $c_{\rm en}$ rising prematurely as a result of the fault and $c_{\rm en}$ rising as intended by the design.

- The premature transition is excited for $b_a\uparrow$, which subsequently causes $x\downarrow$ (by design) $c_r\uparrow$ (by b_{a1} stuck-at-0) and hence $c_{\rm en}\uparrow$.
- The designed transition is excited for $b_a\downarrow$, causing $c_r\uparrow$ (see handshake expansion) and hence $c_{\rm en}\uparrow$.

The only candidate in the handshake expansion that separates the two excitation transitions $b_a\uparrow$ and $b_a\downarrow$ without changing SEQ is the state with b_r low and b_a still high. Voltage inspection in this new quiescent state shows that $c_{\rm en}$ is high in the presence of the fault, and low for the fault-free circuit. In this way, the fault is detected independent of the data-path delays.

Similar evaluations to nail the remaining undetected faults yield three classes of additional quiescent states in the handshake expansion of SEQ [see Fig. 2(b)].

Class 1: Hold handshake from request to acknowledge:

- b_r high and b_a still low;
- b_r low and b_a still high;
- c_r high and c_a still low;
- c_r low and c_a still high.

Class 2: Hold handshake from acknowledge to request:

- a_a high and a_r still high.

Class 3: Hold between internal transitions.

1) DfT Cost Perspectives for Classes 1–3: A hierarchical configuration of handshaking components, like the one we simulated, yields a hierarchy of handshake expansions which unfold like a telescope. A newly created quiescent state at the end of the unfolded hierarchy can be shared by all levels in the hierarchy. In Tangram, unfolding occurs typically between the request phase and the corresponding acknowledge phase [cf. dark-grey color scheme for the handshake expansion in Fig. 2(b)]. As a result, the following was observed.

- Class 1 has the lowest-cost DfT perspectives, because the extra quiescent states for the grey-colored SEQ also suffice for SEQ components higher up in the configuration. By way of illustration, note that "b_r high and b_a still low" implies that also "a_r high and a_a still low," and that "c_r low and c_a still high" implies "a_r low and a_a still high."
- Class 2 cannot take advantage of the unfolding and will require local DfT per component, which can be relatively expensive as we have seen in Section III-B.
- Class 3 is independent of the unfolding mechanism, and likely very expensive in DfT because it requires every single SEQ component to have additional states.

2) Getting Rid of Classes 2 and 3: From the cost perspectives in Section III-B, we conclude that Class 1 may be worth implementing, whereas Class 2 and 3 are better avoided. Here, we will show that the latter two can be ignored when testing SEQ, without impacting the reliability.

Class 2 targets just one fault, which is the bridging fault between metal interconnect node a_a and node y_{01} inside the C-element. With y_{01} in the diffusion layer, the probability of this fault is low enough to ignore Class 2.

Class 3 targets four bridging faults: between y and the internal NAND node, between y_{11} and the internal NOR node, between y_{11} and $V_{\rm DD}$, and between y_{01} and $V_{\rm SS}$. Only the last one gives a marginal increase in delay, namely from $a_r\downarrow$ to $c_r\downarrow$, because the C-element pull-up path controlled by a_r initially has to fight the keeper pull-down path via the bridge. With a keeper of minimal size, the conflict period will be very short and insignificant for aging. Thus, we can ignore Class 3.

C. Tailored High-Quality Low-Cost DfT Solution

Fig. 10 shows the new circuit configuration, with DfT to create the additional quiescent states of Class 1. The DfT circuitry, called HOLD, is injected in the leaf nodes of the control hierarchy, at the handshake interface between control and data path. A straightforward implementation of HOLD is shown in Fig. 10(b) and is explained in Section III-C1. Section IV-C2 shows how the data-path enable signals can be observed by scanning the data-path, and how HOLD and data-path scan cooperate. Section IV-C3 gives an overview of performance and costs.

1) HOLD Implementation for Class 1: The HOLD circuitry has three global signals h_0, h_1, and h_2, to lock-step the state transitions to and from the quiescent states in Class 1. In normal operation, their respective values are high, low, high, making the circuit transparent. In test mode all three values are initially low, forcing the circuit to halt in the first quiescent state, where b_r is high and b_a low for the fault-free circuit. Here, we raise h_0 to inspect the (supposedly high) voltage level of $b_{\rm en}$ and to measure $I_{\rm DDQ}$. Before stepping to the next quiescent state, with b_r low and b_a high for the fault-free circuit, we first close the set-reset latch for down-transitions by raising h_1. Then we raise h_2 to go to the next state, where we measure $I_{\rm DDQ}$ and inspect the (supposedly low) voltage level of $b_{\rm en}$. From here, we step to the first quiescent state in the next handshake by subsequently lowering h_0, h_1, h_2, etc.

Fault analysis of this HOLD implementation shows that all relevant stuck-at and bridging faults can be tested without further DfT needs [17].

Figs. 3–4 in the beginning of the paper are actual fault simulations for the new configuration with this HOLD DfT built into the leaf nodes, which explains the intermediate quiescent states.

2) Data-Path Scan: We can either directly inspect the voltage level of data-path enable signals, or indirectly by scanning the data-path latches. We do the latter, because it also enables us to test the data paths. Our scan DfT is synchronous [10] because of the lower costs, but self timed is also possible [11].

To see whether $b_{\rm en}$ is high, we check whether the write latches of the data path are being written (see Fig. 7). In

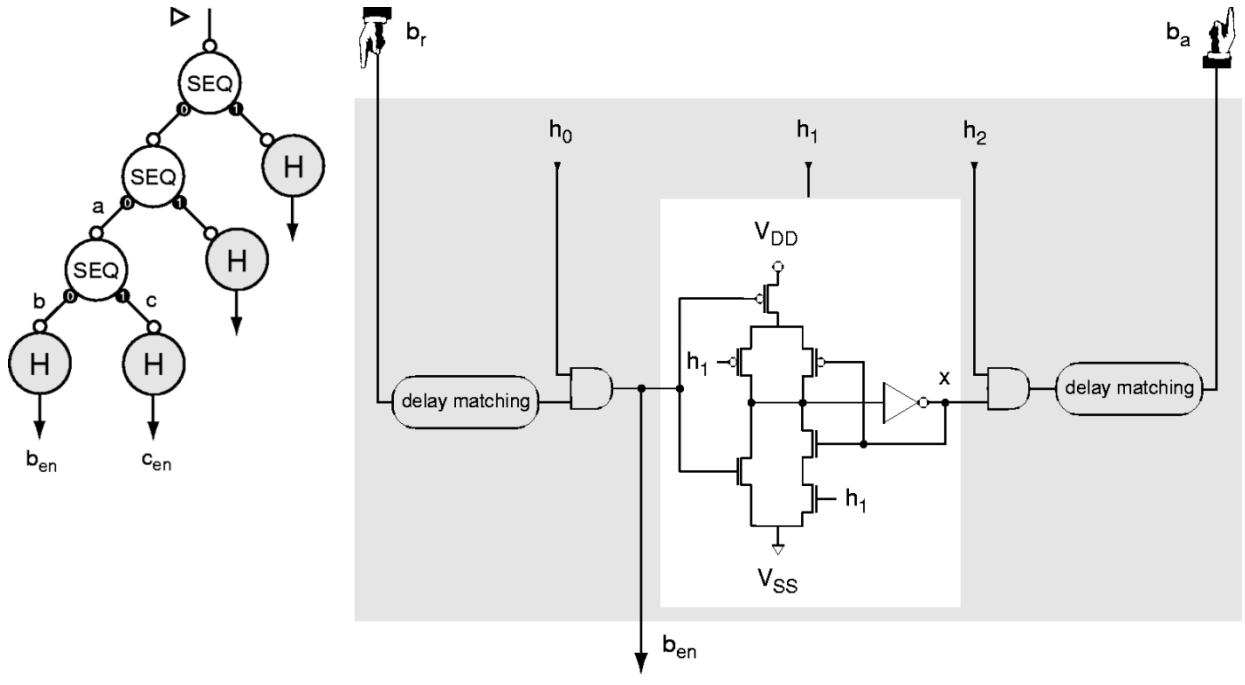


Fig. 10. New circuit configuration with (a) HOLD DfT and (b) a straightforward implementation for HOLD. The delay for the DfT circuitry is compensated in the delay-matchings for the data-path operation (see Fig. 7).

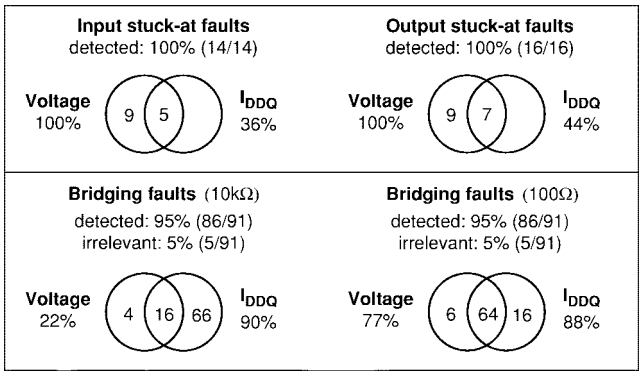


Fig. 11. Simulated fault coverage for SEQ with HOLD DfT.

order to see this, the old and new latch values must be different. A scan chain through the latches enables us to control the difference by shifting values into the latches when h_0 is still low. Then we pulse h_0 to pulse $b_{\rm en}$ and latch the new results into the scan chain, which are then shifted out for comparison. We conclude that $b_{\rm en}$ is high if and only if the old scan values are different from the new results. This can be done while we scan in and apply test stimuli for the data path and scan out the responses.

To see whether $b_{\rm en}$ is low, we check whether the write latches are not being written, which takes two more scan operations. We conclude that $b_{\rm en}$ is low if and only if the old and new values are the same.

3) Performance and Costs: Fig. 11 gives an overview of the stuck-at and bridging fault coverage for SEQ in the new configuration with HOLD DfT. The 5($\times$2) bridging faults that remain undetected correspond to the ones evaluated in Section IV-B2, which we judged to be either unlikely or benign.

Table 2
Costs of HOLD DFT for Self-Timed Control

IC	HOLD	Voltage tests		I_{DDQ}
		total	control	control
DDD	12%	508	35%	453
80C51	31%	2120	48%	1820
ADPCM	15%	1662	17%	763

Fault simulations for other Tangram control components using the same HOLD DfT to create quiescent states of Class 1 show equally high fault coverage [17]. For all these components, we could eliminate the need for quiescent states of Class 3, whereas Class 2 was only needed for specific multiplexer configurations.

Table 2 gives an overview of the HOLD costs and test performance for the three asynchronous IC benchmarks from Philips, introduced in Section III-B. The second column shows the percentage of HOLD's relative to the number of set-reset and data-path latches. Note that this is two to nine times lower than the latch percentage needed for full-scan control, which was given in Table 1. To get an idea of what this means in terms of cell-area overhead: the (nonoptimized) HOLD DfT led to an increase of 8% for the ADPCM, while the full-scan control solution (with optimized scan flipflops) would have added 24%.

The third column gives the number of voltage tests for checking the data-path enable signals. The percentage rela-

tive to the total number of voltage tests for control and (full-scan) data paths is proportional to the control area: 35%, 48%, and 17% versus 40%, 40%, and 20% (see Table 1).

The last column gives the number of I_{DDQ} tests for the self-timed control (including HOLD DfT), which is relatively high due to the distributed nature of the control that is kept intact by our DfT method. With a 100-kHz I_{DDQ} monitor [44] the test times will still be good, though.

One final remark on speed and power: the extra delay and activity introduced by the HOLD circuitry are usually negligible in comparison to the average handshake activity, and can often be compensated in the delay matchings for the single-rail data-path operations.

V. Conclusion

Self timing is one of the most essential elements in an asynchronous computation model. Besides offering low-power and average-case performance advantages, self timing can get in the way when the design needs to be tested because there are fewer quiescent states to observe. As a result, voltage and I_{DDQ}-based test methods become too ineffective, which we illustrated by fault simulation and analysis. We showed that standard full-scan solutions are too expensive, and we developed a tailored DfT approach to create a small but sufficient collection of quiescent states to restore the effectiveness of both test methods. We demonstrated this approach for asynchronous control circuits in Tangram, which resulted in the HOLD DfT solution that is needed for both stuck-at and bridging faults. While our development approach towards such a high-quality, low-cost DfT solution can be used for various styles of asynchronous circuits, the costs of the solution are tied to the class of missing quiescent states, which may differ per style.

Acknowledgment

The author gratefully acknowledges E. Bruls for his inspiring cooperation in the fault analysis and wishes to thank J. Kessels, A. Mettler, and E. Woutersen for the many constructive discussions on the test approach.

References

[1] G. Birtwistle and A. Davis, Eds., *Asynchronous Digital Circuit Design*. New York: Springer-Verlag, 1995.

[2] P. A. Beerel and T. H.-Y. Meng, "Semi-modularity and testability of speed-independent circuits," *Integr. VLSI J.*, vol. 13, no. 3, pp. 301–322, 1992.

[3] H. Hulgaard, S. Burns, and G. Borriello, "Testing asynchronous circuits: A survey," *Integr. VLSI J.*, vol. 19, no. 3, pp. 111–131, 1995.

[4] E. Bruls, "Quality and reliability impact of defect data analysis," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, pp. 121–129, Feb. 1995.

[5] R. Rodríguez-Montañes, E. M. J. G. Bruls, and J. Figueras, "Bridging defects resistance measurements in a CMOS process," in *Proc. Int. Test Conf.*, 1992, pp. 892–899.

[6] J. Segura, C. De Benito, A. Rubio, and C. F. Hawkins, "A detailed analysis of GOS defects in MOS transistors: Testing implications at circuit level," in *Proc. Int. Test Conf.*, 1995, pp. 544–551.

[7] M.-D. Shieh, C.-L. Wey, and P. D. Fisher, "A scan design for asynchronous sequential logic circuits using SR-latches," in *Proc. Midwest Symp. Circuits and Systems*, 1993, pp. 1300–1303.

[8] M. Roncken, "Partial scan test for asynchronous circuits illustrated on a DCC error corrector," in *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, Salt Lake City, UT, 1994, pp. 247–256.

[9] A. Khoche and E. Brunvand, "A partial scan methodology for testing self-timed circuits," in *Proc. IEEE VLSI Test Symp.*, NJ, 1995, pp. 283–289.

[10] M. Roncken, E. Aarts, and W. Verhaegh, "Optimal scan for pipelined testing: An asynchronous foundation," in *Proc. Int. Test Conf.*, 1996, pp. 215–224.

[11] V. Schöber and T. Kiel, "An asynchronous scan path concept for micropipelines using the bundled data convention," in *Proc. Int. Test Conf.*, 1996, pp. 225–231.

[12] O. Petlin and S. Furber, "Built-in self-testing of micropipelines," in *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, Eindhoven, The Netherlands, 1997, pp. 22–29.

[13] L. Lavagno, M. Kishinevsky, and A. Lioy, "Testing redundant asynchronous circuits," Dep. Comput. Sci., Technical Univ. Denmark, Lyngby, Tech. Rep. ID-TR:1993-124, 1993.

[14] A. Khoche and E. Brunvand, "Critical hazard free test generation for asynchronous circuits," in *Proc. IEEE VLSI Test Symp.*, NJ, 1997, pp. 203–208.

[15] M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Saldanha, and A. Taubin, "Partial scan delay fault testing of asynchronous circuits," in *Proc. ICCAD*, 1997, pp. 728–735.

[16] S. M. Nowick, N. K. Jha, and F.-C. Cheng, "Synthesis of asynchronous circuits for stuck-at and robust path delay fault testability," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 1514–1521, Dec. 1997.

[17] M. Roncken and E. Bruls, "Test quality of asynchronous circuits: A defect-oriented evaluation," in *Proc. Int. Test Conf.*, 1996, pp. 205–214.

[18] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, and F. Schalij, "The VLSI-programming language Tangram and its translation into handshake circuits," in *Proc. Europ. Conf. Design Automation*(*EDAC*), 1991, pp. 384–389.

[19] K. van Berkel, "Handshake Circuits: An asynchronous architecture for VLSI programming," *International Series on Parallel Computation*, vol. 5. Cambridge, U.K.: Cambridge Univ. Press, 1993.

[20] W. Maly, "Realistic fault modeling for VLSI testing," in *Proc. Design Automation Conf.* (*DAC*), 1987, pp. 173–180.

[21] J. P. Shen, W. Maly, and F. J. Ferguson, "Inductive fault analysis of MOS integrated circuits," *IEEE Design Test Comput.*, vol. 2, pp. 13–26, June 1985.

[22] H. Walker and S. W. Director, "VLASIC: A catastrophic fault yield simulator for integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. 5, pp. 541–556, Apr. 1986.

[23] A. Jee and F. J. Ferguson, "Carafe: An inductive fault analysis tool for CMOS VLSI circuits," in *Proc. IEEE VLSI Test Symp.*, NJ, 1993, pp. 92–98.

[24] F. J. Ferguson and J. P. Shen, "Extracting and simulation of realistic CMOS faults using inductive fault analysis," in *Proc. Int. Test Conf.*, 1988, pp. 475–484.

[25] W. Mao, R. K. Gulati, D. K. Goel, and M. D. Ciletti, "QUIETEST: A quiescent current testing methodology for detecting leakage faults," in *Proc. Int. Conf. Computer-Aided Design* (*ICCAD*), 1990, pp. 280–283.

[26] C. F. Hawkins and J. M. Soden, "Electrical characteristics and testing considerations for gate oxide shorts in CMOS IC's," in *Proc. Int. Test Conf.*, 1985, pp. 544–555.

[27] J. M. Soden, C. F. Hawkins, R. K. Gulati, and W. Mao, "I_{DDQ} testing: A review," *J. Electron. Testing: Theory Applicat.*, vol. 3, pp. 291–303, 1992.

[28] T. W. Williams, R. Kapur, M. R. Mercer, R. H. Dennard, and W. Maly, "I_{DDQ} testing for high performance CMOS—The next ten years," in *Proc. Europ. Design and Test Conf.* (*EDAC-ETC-EuroASIC*), 1996, pp. 578–583.

[29] M. Sachdev, "Deep sub-micron I_{DDQ} testing: Issues and solutions," in *Proc. Europ. Design and Test Conf.* (*EDAC-ETC-EuroASIC*), 1997, pp. 271–278.

[30] A. Keshavarzi, K. Roy, and C. F. Hawkins, "Intrinsic leakage in low power deep submicron CMOS IC's," in *Proc. Int. Test Conf.*, 1997, pp. 146–155.

[31] A. Rubio, M. Roca, and E. Sicard, "I_{DDQ} testing of oscillating

bridging faults in CMOS combinational circuits," *Proc. Inst. Elect. Eng. G*, vol. 140, no. 1, 1993.
[32] R. Rodríguez-Montañes and J. Figueras, "Analysis of bridging defects in sequential CMOS circuits and their current testability," in *Proc. European Test Conf.*, 1994, pp. 356–360.
[33] H. Hao and E. J. McCluskey, "Resistive shorts within CMOS gates," in *Proc. Int. Test Conf.*, 1991, pp. 292–301.
[34] H. T. Vierhaus, W. Meyer, and U. Gläser, "CMOS bridges and resistive transistor faults: I_{DDQ} versus delay effects," in *Proc. Int. Test Conf.*, 1993, pp. 83–91.
[35] A. K. Pramanick and S. M. Reddy, "On the computation of the ranges of detected delay fault sizes," in *Proc. Int. Conf. Computer-Aided Design* (*ICCAD*), 1989, pp. 126–129.
[36] P. Franco and E. J. McCluskey, "Delay testing of digital circuits by output waveform analysis," in *Proc. Int. Test Conf.*, 1991, pp. 798–807.
[37] P. Nigh, W. Needham, K. Butler, P. Maxwell, and R. Aitken, "An experimental study comparing the relative effectiveness of functional, scan, I_{DDQ}, and delay-fault testing," in *Proc. IEEE VLSI Test Symp.*, NJ, 1997, pp. 459–464.
[38] C. F. Hawkins and J. M. Soden, "Reliability and electrical properties of gate oxide shorts in CMOS IC's," in *Proc. Int. Test Conf.*, 1986, pp. 443–451.
[39] K. van Berkel, R. Burgess, J. Kessels, A. Peeters, M. Roncken, and F. Schalij, "Asynchronous circuits for low power: A DCC error corrector," *IEEE Design Test Comput.*, vol. 11, pp. 22–32, Summer 1994.
[40] K. van Berkel, R. Burgess, J. Kessels, A. Peeters, M. Roncken, F. Schalij, and R. van de Wiel, "A single-rail re-implementation of a DCC error detector using a generic standard-cell library," in *Proc. 2nd Working Conf. Asynchronous Design Methodologies*, London, 1995, pp. 72–79.
[41] H. van Gageldonk, K. van Berkel, A. Peeters, D. Baumann, D. Gloor, and G. Stegman, "An asynchronous low-power 80C51 microcontroller," in *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, San Diego, CA, 1998, pp. 96–107.
[42] B. Atzema, E. Bruls, M. Sachdev, and T. Zwemstra, "Computer-aided testability analysis for analog circuits," in *Proc. Workshop on Advances in Analog Circuit Design*, 1995.
[43] A. Martin and P. Hazewindus, "Testing delay-insensitive circuits," in *Proc. UC Santa Cruz Conf. Advanced Research in VLSI*, 1991, pp. 118–132.
[44] K. Baker and A. Hales, "Quality test action group (QTAG): Plug and play I_{DDQ} testing for test fixtures," *IEEE Design Test Comput.*, vol. 12, pp. 53–61, Fall 1995.

Marly Roncken received the M.Sc. degree in mathematics and computer science from the University of Utrecht, The Netherlands.

From 1985 to 1997, she was a Researcher at Philips Research Laboratories, Eindhoven, The Netherlands, in the area of VLSI design automation and test. She was responsible for the test operations and test research and development in the VLSI Programming and Silicon Compilation Project for asynchronous circuits (Tangram). In 1997, she joined Intel Strategic CAD Laboratories, Hillsboro, OR. Her main interests include VLSI fault grading and test methodologies, and design for test and manufacturability.