A chip for fault detection experiments[†]

S.D. Brown and Z.G. Vranesic
Department of Electrical Engineering, University of Toronto, Toronto, M5S 1A4, Canada

We present the design and implementation of a chip to serve as a vehicle for test experiments. The chip implements a one-channel 8-bit DMA controller, and includes some special circuitry to act as faults. Each fault can be turned on or off by programming a set of pins and the experiment consists of finding a test set that can recognize a fault when it is turned on.

1. Introduction

Increasing complexity of VLSI chips is placing greater demands on the testing techniques needed to ascertain their operational correctness. We feel that it is fair to say that many of today's designers of integrated circuits are not as conversant with the testing techniques as they should be. In an attempt to correct this, universities are introducing courses on testing and fault tolerance of digital systems. These courses usually consist of lecture presentations and illustrative problem solving assignments, but seldom include practical hands-on experiments. Yet the latter aspect provides invaluable insight into the complexity of the task. To support such a course at the University of Toronto, we have designed a chip to serve as a vehicle for test experiments. The chip can be programmed to exhibit a number of different faults and it can be tested in the same way as standard integrated circuits.

For practical reasons, the chip to be tested must be a non-trivial circuit that is of some use, and has a reasonable complexity and pin-count. We chose a one-channel data-chaining 8-bit DMA controller, which is a good choice because it contains both data path and control circuits. The chip contains programmable faults which are controlled by four input pins whose permutations activate faults within the circuitry. Students can then be asked to find the single permutation that results in a fault-free chip.

The intent is for the student to create a set of test vectors by some method from the theory of fault detection [1]–[4], and then apply the vectors to the chip. The test set should be applied for each of the 16 possible permutations of the fault pins and should show a fault-free circuit for exactly one of the permutations. The student is thus faced with the challenge of deriving a thorough test set for this real non-trivial circuit, and is given a means of determining whether the set provides full coverage.

An even more challenging task may be attempted in which the student is told the location and type of each fault, and is then asked to determine which of the faults is activated by each permutation of the fault-pins.

In the remainder of this paper we will describe the types of faults created on the chip and their implementations. We will also discuss the reasoning behind the locations of each type of fault.

2. The types of faults

The types of faults we have chosen are intended to represent the most common faults that may occur in real VLSI circuits:

- stuck-at-0 fault
- stuck-at-1 fault
- short-of-two-lines fault
- spurious fault

The stuck-at-0 and stuck-at-1 faults, shown in Figs. 1(a) and 1(b), represent the general "stuck-at" fault model that is the most widespread model in use in fault detection today. The signal marked z is used to activate the fault (a logic level of 0 is used to indicate an active fault consistently in this paper). The normal signal is x and the faulty signal is y. Thus, in the Figure, if z is active, y is stuck-at-0, while if z is not active y is the same as x.

The short-of-two-lines fault probably occurs most often along the path of a data transfer bus that consists of several parallel lines with a minimum separation. In Fig. 1(c), when z is active, inputs x_1 and x_2 appear to be shorted together at y_1 and y_2, while if z is not active, y_1 is x_1 and y_2 is x_2.

Figure 1(d) shows the spurious fault, which is really a stuck-at-0 fault (controlled by an edge triggered flip-flop) that appears inconsistently in the circuit. This might represent the physical case

† This paper was first presented at the 1989 Canadian Conference on Very Large Scale Integration, CCVLSI '89, held in Vancouver, BC, on 22–24 October, 1989, and is reprinted here with the permission of the Conference organisers.

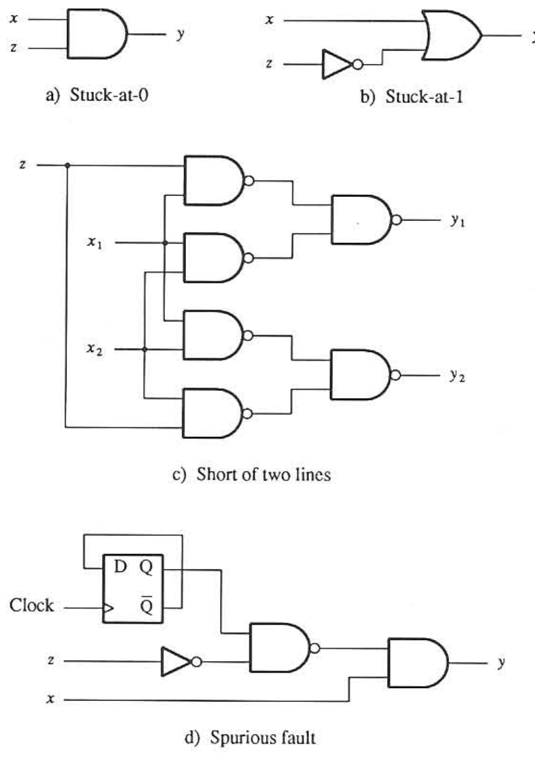

Fig. 1 The implementation of logic faults

of a race condition, where a node (presumably an active-high node for this spurious stuck-at-0) sometimes appears to behave correctly, and sometimes does not.

3. Fault locations

The four fault pins are decoded so that one permutation is fault-free, while the remaining permutations each control at least one fault. The fault locations were chosen to try to represent as many different scenarios as possible — this is important because it is obviously a very different matter to detect a fault in a piece of control circuitry than a fault on a data transfer line.

The stuck-at faults are mostly placed on control signals. Some of these signals are obvious choices, such as the Read/Write* line, and some are more subtle such as the interrupt request line (IRQ) which indicates that DMA is finished. Some control line faults would create chaos in the chip because they affect such critical circuitry as state machines, and these faults would probably be easy to detect. More subtle faults might only affect a small part of the chip's operation, and would be more difficult to spot. For instance, a fault on the IRQ line will not affect any of the chip's operation, except that it will not assert the IRQ when the DMA is finished. Subtle faults are of course the biggest challenge for the tester. As far as the *location* of a detected fault is concerned, it is not clear whether an obvious fault is easier to locate than a subtle fault. The obvious fault is likely to manifest itself in many parts of the chip, thus making location of its source nebulous, while the subtle fault is contained in a small part of the circuitry.

The short-of-two-lines faults are placed on data transfer lines because it is these that generally run in parallel with minimum separation for relatively long distances. Even so, there are different scenarios that should be represented here as well. For instance, the operation of the chip could be unaffected by a fault on two data lines except that a register that is written correctly cannot be read correctly (a short of two internal lines that are isolated from the data pins by buffers). Such a fault would be difficult to detect. By contrast, a fault directly in the path of the data pins would affect all register operations on the chip and would be easily detected. Shorted lines also pose a challenge because some combinations of inputs will not create errors (because the shorted lines were supposed to have the same values anyway), and some will.

The spurious faults will naturally be the most difficult to detect. This corresponds to the real world difficulties in detecting and especially locating spurious faults. Race conditions, for instance, can sometimes be almost impossible to locate even when they are detectable. The spurious faults in the DMA chip are placed on control signals because race conditions are more likely to exist there.

4. Implementation of the DMA chip

To shorten the design cycle the chip was designed with standard cells, either taken from the cell library at the University of Toronto or designed by the author. Since this chip is meant to be tested, all flip-flops that appear in the control logic must be of the LSSD (Level Sensitive Scan Design) type [5]–[7]. Flip-flops that are part of data registers are

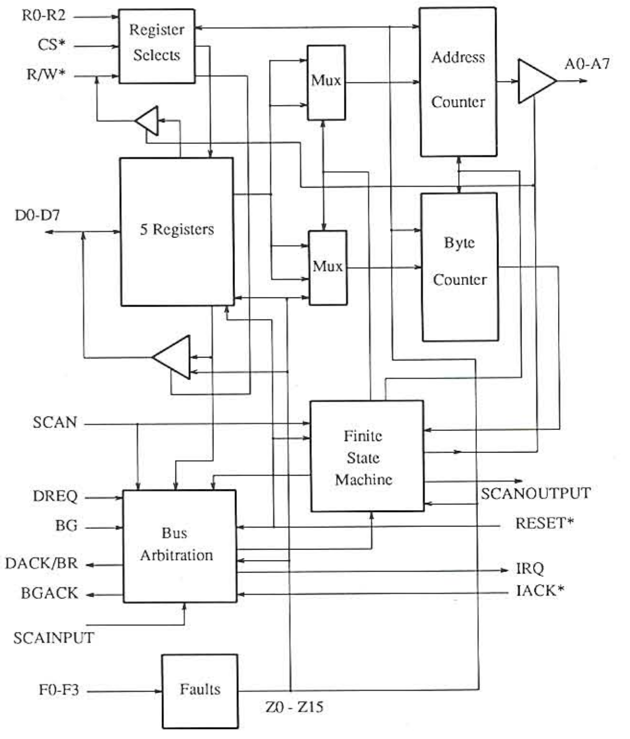

Fig. 2 The DMA block diagram

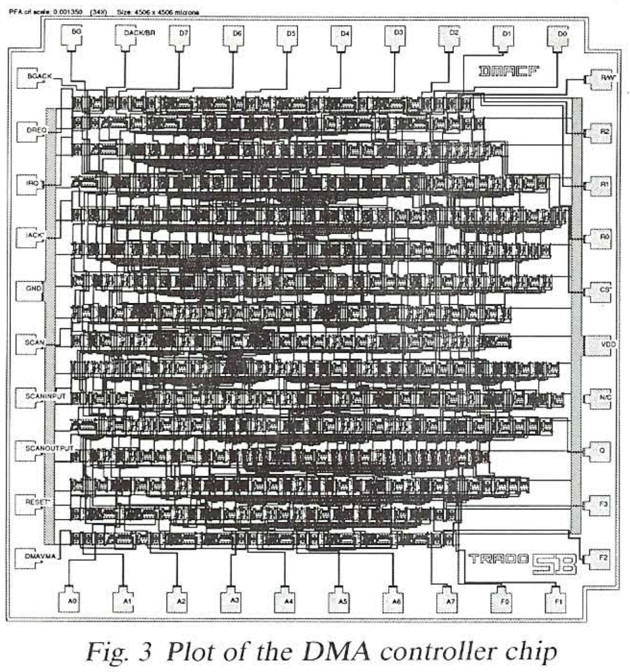
Fig. 3 Plot of the DMA controller chip

not LSSD because they can be loaded directly from external signals.

Figure 2 shows a block diagram of the chip and indicates where the faults have been placed. Figure 3 gives a plot of the final chip, which measures about 4500 microns square.

An alternative to creating a standard-cell chip is to use a Field Programmable Gate Array. This would provide a fast turnaround time and also allow the chip to be changed from year to year for different students. We firmly feel, however, that a chip must be built and that simulation of a chip is not a viable alternative. The philosophy of this project is to provide a hands-on testing experience and we do not feel this can be effectively simulated.

5. Concluding remarks

We have created a chip with known faults, so that the validity of a set of tests for its circuitry can be evaluated. Moreover, because the locations of the faults are known, the tester can assess the ability of the test set to locate the detectable faults. We hope that the availability of this chip will encourage the application of fault tolerant methods by supplying practical experience in testing a real circuit, with positive feedback on the fault coverage achieved. We intend to use this chip in a graduate level course on Fault Tolerant Computing at the University of Toronto.

Biographies

Zvonko G. Vranesic received his B.A.Sc., M.A.Sc., and Ph.D. degrees in electrical engineering from the University of Toronto, Canada, in 1963, 1966 and 1968, respectively. From 1963 to 1965 he worked as a design engineer for Northern Electric Co. Ltd., Bramalea, Ontario, Canada. In 1968 he joined the faculty of the Departments of Electrical Engineering and Computer Science at the University of Toronto, where he is now a Professor. During the academic years 1977/78 and 1984/85 he was a Senior Visitor in the Computer Laboratory at the University of Cambridge, England, and in the Institut de Programmation at the University of Paris 6, France.

His research interests include computer architecture, VLSI systems, fault tolerant computing, local area networks and many-valued switching systems. He is a member of the Association of Professional Engineers of Ontario and a Senior Member of IEEE.

Stephen Brown received a B.Sc.Eng. degree from the University of New Brunswick in 1985, and an M.A.Sc. degree from the University of Toronto in 1987, both in Electrical Engineering. Presently, he is a Ph.D. candidate in the Department of Electrical Engineering at the University of Toronto.

His research interests include Computer Aided Design and VLSI Systems. He is a member of the IEEE and of the ACM.

6. References

[1] Pradham, D.K. (Editor), *Fault-Tolerant Computing Theory and Techniques*, Prentice Hall, 1986.

[2] Lala, P.K., *Fault-tolerant and Fault-testable Hardware*, Prentice Hall, 1985.

[3] Miczo, A., *Digital Logic Testing and Simulation*, Wiley, 1986.

[4] Kuo, F.F., *Computer Applications in Electrical Engineering Series*, Prentice Hall, 1971.

[5] Williams, T.W. and Parker, K.P., "Design for testability — a survey", *IEEE Trans. on Comput.* pp. 2-15, January 1982.

[6] Eichelberger, E.B. and Williams, T.W., "A logic design structure for LSI testability", *Proc. IEEE 14th Design Automation Conference*, pp. 347-352, June 1978.

[7] Dasgupta, S. et. al., "A variation of LSSD and its implications on design and test pattern generation in VLSI", *Proc. IEEE Test Conf.*, pp. 63-66, 1982.