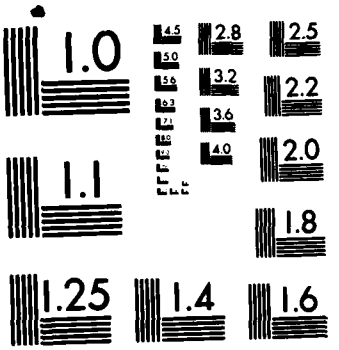
AD-A161 929 FUNCTIONAL TESTING OF LSI/VLSI BASED SYSTEMS WITH MEASURE OF FAULT COVERAGE(U) STATE UNIV OF NEW YORK AT ALBANY RESEARCH FOUNDATION S Y SU 03 AUG 83 1/1

UNCLASSIFIED DAAB77-82-K-J056 F/G 9/2 NL



SCIENTIFIC AND TECHNICAL REPORT

FOURTH QUARTERLY STATUS REPORT

4th

Prepared By

Stephen Y.H. Su
Project Director
Dept. of Computer Science
State University of New York
Binghamton, New York 13901
(607)-798-2296 (office
(607)-798-4802 (secretary)

A. Contractor's name and address:

The Research Foundation of
State University of New York
P.O. Box 9, Albany, New York 12201

B. Contract No. DAAB07-82-K J05-6

C. Date of Report - August 3, 1983

D. Title: The Fourth Quarterly Status Report for the project "Functional Testing of LSI/VLSI Based Systems with Measure of Fault Coverage"

E. Period Covered: April 28 to July 27, 1983

F. Description of Progress: See Attached

G. Spending: Actual Quarter Cost: $ 15,523.00

Cumulative Cost to date:$ 90,800.00

Microprocessor is a type of complex sequential machine. The current approach is to test microprocessor by instruction execution. Generally, before executing an instruction-under-test, we have to write certain data into some registers, and after executing the instruction, we have to read the contents of the registers. Therefore, if the write or read instruction is faulty, we may not be able to test the instruction-under-test. To solve this problem, Thatte and Abraham [1] have to label instructions and define test order in detail before testing. In addition, they do not consider the partial execution of an instruction. So one of the instruction decoding faults $I_{(j)} / I_{(i)}+I_{(k)}$ is assumed that, instead of executing $I_{(j)}$, both instructions $I_{(j)}$ and $I_{(k)}$ are executed to completion. This raises the problem of practicality.

Abraham and Packer's method [2] is simple, but their "register read test" procedure does not guarantee the correctness of write and read register functions for any data.

In our work, we consider the write and read register function as a kernal of a microprocessor; similar to a sequential machine, and we use the checking experiment to verify the kernal based on the fault models. Then we use the kernal for testing other instructions.

Our approach consists of three steps:

Step 1: Establish the fault model for representing functional faults

Step 2: Determine the requirement for test generation

Step 3: Based on the requirements, develop testing procedures.

We shall briefly describe the fault model and the requirements. First of all, we need to establish a fault model for a microprocessor.

The semiconductor manufacturers consider the circuit diagrams of microprocessor chips proprietary information and therefore, these diagrams are not available to users. The only information we can get is from manufacturer's data books and application notes, including the architecture, the instruction set of a microprocessor, and so forth, so that microprocessor testing here is a functional testing.

The functions of a microprocessor are mainly performed by instruction execution. The functions of an instruction can be described by RTL (Register Transfer Language) [3] . We consider here that an instruction consists of a series of RTL statements. The typical statement is defined as

$$(\text{conditions}), \quad D \leftarrow f(S_1, S_2, \ldots, S_i \ldots)$$

where D - destination

S_i - source

$f(S_1, S_2, \ldots, S_i, \ldots)$ - operation

A microprocessor usually can be divided into two sections: the data processing part, and the control part. Here we only discuss testing instructions of a microprocessor. The sequencing of RTL statements reflects the control part of a microprocessor. The RTL statement itself is corresponding to the data processing part as well as the control part. We consider two types of faults.

A. Data Processing Faults

1. Data storage fault (R)/(R)' i.e. due to a fault, the content of register R is changed to a different value. It means register element fault and includes stuck-at and pattern sensitive faults.

2. Data transfer fault $\leftarrow/\leftarrow'$

   The fault appears at a source or destination transfer path and includes stuck-at and bridging fault.

3. Data manipulation fault $(f)/(f)'$

   This means operation execution fault, with this type of fault, the operation f is executed, but the result of execution is wrong.

B. Control Faults

This kind of fault invloves register decoding faults, instruction decoding faults and other control faults. Register decoding faults means the missing and changing of the selected register, and selection of extra register, i.e. R/ϕ, R/R' and $R/R+R'$ respectively. For instruction decoding faults, we consider that an instruction can be executed partially. It means missing and changing of selected operation and selecting of extra operation in RTL. In that case, the instruction decoding fault may be I_j/ϕ, $I_j/\Delta I_j$, $I_j/\Delta I_k$, I_j/I_k, $I_j/\Delta I_j + \Delta I_k$, $I_j/I_j + \Delta I_k$, $I_j/I_j + I_k$, and so forth, where ΔI means part of instruction I. In fact, the register decoding faults can be considered as instruction control faults, i.e. $R/\phi \in I_j/\phi$, $R/R' \in I_j/I_k$, $R/R+R' \in I_j/I_j + I_k$, where I_j, I_k are transfer instructions among registers. In addition, The control faults also include instruction execution sequence faults, condition faults and so on.

We will define the above control faults at the RTL level. Let f denote $D \leftarrow f(S_1, S_2, \ldots)$, which is an operation on the instruction-under-test, f' denote $D' \leftarrow f'(S_1', S_2', \ldots)$, which is unexpected (faulty) operation, where source and destination may be registers or

external busses. In the following, without special explanation, the register means register itself or external bus.

We now define three types of control faults.

1) f/ϕ - No operation is executed

2) f/f' - Instead of performing the operation of f, another operation f' is executed. It contains two subtypes of faults.

   2-1) Df/f' : Here D means that Registers D and D' are different and the fault is f/f'.

   2-2) Sf/f' : S denotes that Registers D and D' are the same

3) $f/f+f'$ - In addition to operation f, another operation f' is also executed. It can be subdivided as follows: Firstly, Register D and D' are different.

   3-1) $Df/f+f'$: The results of f and f' do not affect each other

   3-2) $Df/f'f$: The result f' affects the result of f and the execution order is

   1. $D' \leftarrow f'(S_1', S_2', \ldots)$
   2. $D \leftarrow f(S_1, S_2, \ldots, D'^*)$

   where register without * denotes its content before executing the operation; register with * denotes its content after executing the operation.

   3-3) Df/ff' : The result of f affects the result of f' and the execution order is

   1. $D \leftarrow f(S_1, S_2, \ldots)$
   2. $D'^{\leftarrow} f'(S_1', S_2', \ldots, D^*)$

   Secondly, Register D and D' are the same, i.e. D'=D. When the results of f and f' do not affect each other, if the execution order is f', f, the fault does not execution. If the execution affect an instruction ~~execution~~. If the

order is f,f', it is the same as the case with the fault Sf/f'.

3-4) Sf/f'f: The result of f' affects the result of f and the execution order is

1. $D \leftarrow f'\ (S_1', S_2', \ldots)$
2. $D \leftarrow f\ (S_1, S_2, \ldots, D^*)$

3-5) Sf/ff': The result of f affects the result of f' and the execution order is

1. $D \leftarrow f(S_1, S_2, \ldots)$
2. $D \leftarrow f'\ (S_1', S_2', \ldots, D^*)$

3-6) Sf/fLf': Both f and f' are executed at the same time

$D \leftarrow f(S_1, S_2, \ldots)$

$D \leftarrow f'(S_1', S_2', \ldots)$

In this case, the final content of the destination D is the composite value of both results of f and f', i.e. $f(S_1 S_2, \ldots)\ L\ f'\ (S_1', S_2', \ldots)$, where symbol L denotes logical AND or logical OR depending on circuit implementation.

Note that the above control faults can be at any place in an instruction execution sequence.

From the above classification of control faults, we can obtain several requirements for testing these faults.

Let V_i denote the data value of a register i, any register can be the source or the destination for a RTL operation. So register values can also be named VS_i (operand in the source register S_i), VD (operand in the destination register D), and VD* (result of operation f in the destination register D). For an unexpected operation, we have VS_i', VD', and VD'*.

First of all, for testing any operation f, we require the following:

Q1. $f(VS_1, VS_2, \ldots) \neq VD$.

This requirement, Q1, means that a data which is different from the result of operation f is stored in the destination register D. If after operation f, the content of D is not changed to the expected value, then the operation f is incorrect. This test requirement can cover faults f/ϕ and Df/f'. For other faults, we need additional requirement as follows.

For fault Sf/f', the results of f and f' should be different.

Q2. $f(VS_1, VS_2, \ldots) \neq f'(VS_1', VS_2', \ldots)$

For faults Df/f+f' and Df/f'f, we need only to detect the extra operation f'. The requirement is given below.

Q3. $f'(VS_1', VS_2', \ldots) \neq VD'$

For fault Df/ff', the extra operation f' is detected by the following requirement.

Q4. $f'(VS_1', VS_2', \ldots, f) \neq VD'$, where $f = f(VS_1, VS_2, \ldots)$

For fault Sf/f'f, the incorrect result of f due to the fault should be different from the correct one.

Q5. $f(VS_1, VS_2, \ldots, f') \neq f(VS_1, VS_2, \ldots, VD)$, where $f' = f'(VS_1', VS_2', \ldots)$

For fault Sf/ff', similarly, we require the following.

Q6. $f'(VS_1', VS_2', \ldots, f) \neq f(VS_1, VS_2, \ldots)$, where $f = f(VS_1, VS_2, \ldots)$

For fault $Sf \not\perp f'$, the composit of both results of f and f' should be different from the correct result of f.

Q7. $f(VS_1, VS_2, \ldots) \perp f'(VS_1', VS_2', \ldots) \neq f(VS_1, VS_2, \ldots)$

Based on the above seven requirement, we shall develop procedur for testing control faults on microprocessors.

[1] S.M. Thatte and J.A. Abraham, "Test Generation for Microprocessors", IEEE Trans. on Computers, C-29, N.6, June 1980, pp. 429-441.

[2] J.A. Abraham and K.P. Parker, "Practical Microprocessor Testing: Open and Closed Loop Approaches", IEEE Compcon, Spring 1981, pp. 308-311.

[3] S.Y.H. Su and U.I. Hsieh, "Testing Functional Faults in Digital Systems Described by Register Transfer Language", Journal of Digital Systems, Vol. 6 No. 2/3, summer/fall 1982, pp. 161-184. Also in proceedings of 1981 International Test Conference, pp. 447-457.

END

FILMED

1-86

DTIC