Università degli Studi di Verona, Facoltà di Scienze MM. FF. NN.
Dipartimento di Informatica
EDALab: Embedded System Design Center

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

Verona, 15/03/2007

Design for Testability

Nicola Bombieri

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

1 Required Background

Students interested in learning design for testability are required to know the fundamentals of logic nets and basic concepts related to modeling and synthesizing digital systems. Moreover, it is advisable that students are familiar with at least one among the following traditional hardware description languages (HDL): VHDL, SystemC, Verilog.

2 Goal

The goal of this lecture consists of describing the basic concepts related to the use of the Automatic Test Pattern Generator (ATPG) called Flextest to dinamically verify the correctness of a digital system description with respect to the initial specification. Students will learn to:

- apply dinamic verification to designs under test.
- exploit ATPG results to evaluate design correctness and verification quality.

3 Digital circuit testing: testability

Testing difficulty is always increasing due to different reasons:

- Increasing size of circuits and systems.
- Incresing density of integration (more functionaliy to be tested).
- Inaccessibility of interior of the circuits.
- Decreasing ratio gates/pins.
- Increasing speed..

Several tecniques rely on dynamic verification (i.e., *simulation*) to help designers in verifying digital circuit correctness.

First, we give the following definitions in order to distinguish the meaning of fault from error:

- *Fault*: a physical failure mechanism due to some defects of the circuit.
- *Error*: the condition (or state) of a system containing a fault (deviation from correct state).

Thus, in the context of physical analisys, physical defects are modeled by using faults. In particular, logic-level models, such as *stuck-at,* are adopted to measure circuit testability. Functional verification based on simulation aims at detecting *faults* of circuits. Figure 1 shows the basic fault detection process.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

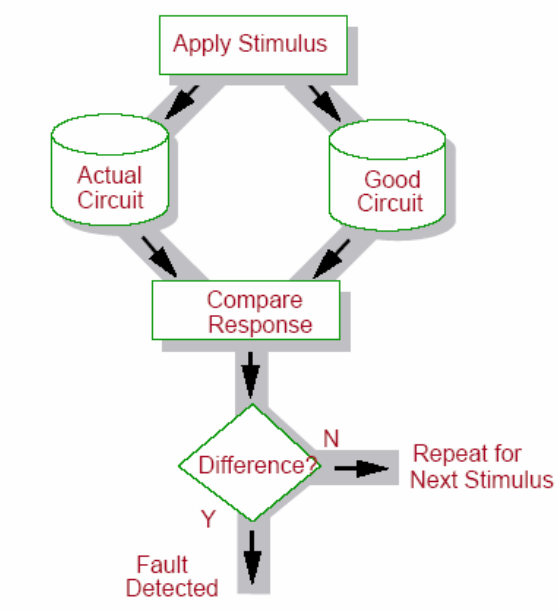


Figure 1

Every possible fault is simulated into the circuit under test. A simulated fault is said to be *injected* into the circuit. Then, faults detection works by comparing the response of a know-good versione of the circuit to that of the actual circuit, for a given stimulus set. A fault exists if there is any difference in the response. The process is repeated for each stimulus set.

3.1 Functional Testing and the Stuck-At Fault Model

Functional testing uses the *single stuck-at model*, the most common fault model used in fault simulation, because of its effectiveness in finding many common defect types. The stuck-at fault models the behavior that occurs if the terminals of a gate are stuck at either a high (stuckat-1) or low (stuck-at-0) voltage. The fault sites for this fault model include the pins of primitive instances. Figure 2 shows the possible stuck-at faults that could occur on a single AND gate.

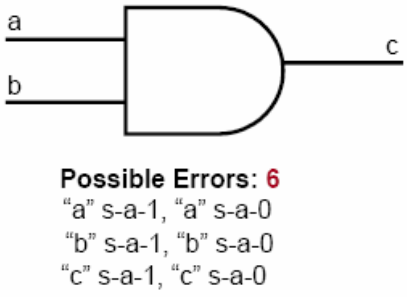


Figure 2

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

3.2 Observability and Controllability

The actual fault detection methods vary. One common approach is *path sensitization*. The path sensitization method, which is used by FlexTest to detect stuck-at faults, starts at the fault site and tries to construct a vector to propagate the fault effect to a primary output.
When successful, the tools create a stimulus set (a test pattern) to detect the fault. They attempt to do this for each fault in the circuit's fault universe. Figure 3 shows an example circuit for which path sensitization is appropriate.

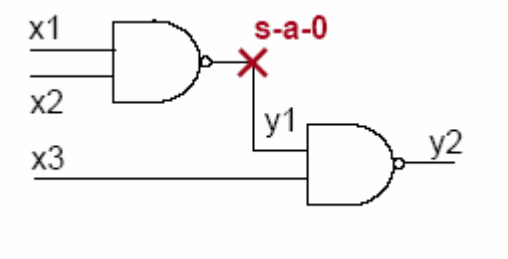


Figure 3

The circuit in Figure 3.2 has a *stuck-at-0* on line y1 as the target fault. The x1, x2, and x3 signals are the primary inputs, and y2 is the primary output. The path sensitization procedure for this example follows:

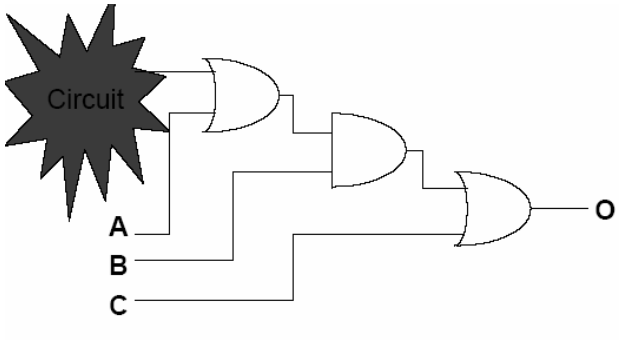
1. Find an input value that sets the fault site to the opposite of the desired value. In this case, the process needs to determine the input values necessary at x1 and/or x2 that set y1 to a 1, since the target fault is s-a-0. Setting x1 (or x2) to a 0 properly sets y1 to a 1.
2. Select a path to propagate the response of the fault site to a primary output. In this case, the fault response propagates to primary output y2.
3. Specify the input values (in addition to those specified in step 1) to enable detection at the primary output. In this case, in order to detect the fault at y1, the x3 input must be set to a 1.

Circuit testability is defined by two main measures:

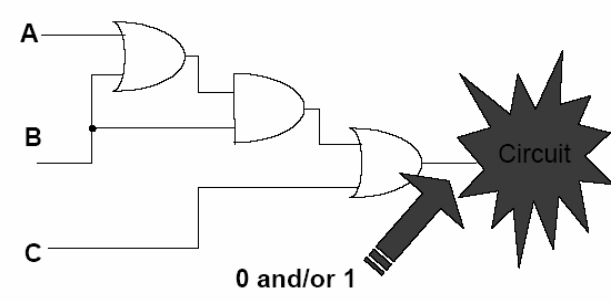
Observability: how easily internal signals can be seen at the outputs.

Controllability: how easily specific internal signal values can be produced by applying signals to inputs.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068



For example, if a fault exists into the circuit given above, the observability is guaranteed by applying values A=0, B=1 and C=0 to carry out the fault to the output O. On the other hand, controllability of a fault into the circuit given below, e.g., stuck-at 0, is guaranteed by applying values A=B=C=0. For a stuck-at-1, insted, one of the possible array of values is A=B=C=1.



4 Flextest: an Automatic Test Pattern Generator

ATPG stands for Automatic Test Pattern Generation. *Test patterns*, sometimes called *test vectors*, are sets of 1s and 0s placed on primary input pins during the manufacturing test process to determine if the chip is functioning properly. When the test pattern is applied, the Automatic Test Equipment (ATE) determines if the circuit is free from manufacturing defects by comparing the fault-free output—which is also contained in the test pattern with the actual output measured by the ATE.

4.1 The ATPG process

The goal of ATPG is to create a set of patterns that achieves a given *test coverage*, where test coverage is the total percentage of testable faults the pattern set actually detects. ATPG consists of two main steps:

1. generating patterns and
2. performing fault simulation to determine which faults the patterns detect.

Flextest automates these two steps into a single operation or ATPG process. This ATPG process results in patterns you can then save with added tester-specific formatting that enables a tester to load the pattern data into a chip's scan cells and otherwise apply the patterns correctly.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

The two most typical methods for pattern generation are random and deterministic. Additionally, the ATPG tools can fault simulate patterns from an external set and place those patterns detecting faults in a test set. Flextest is used to verify the correctness of RTL or gate-level models of HW components as reported in Figure 4.

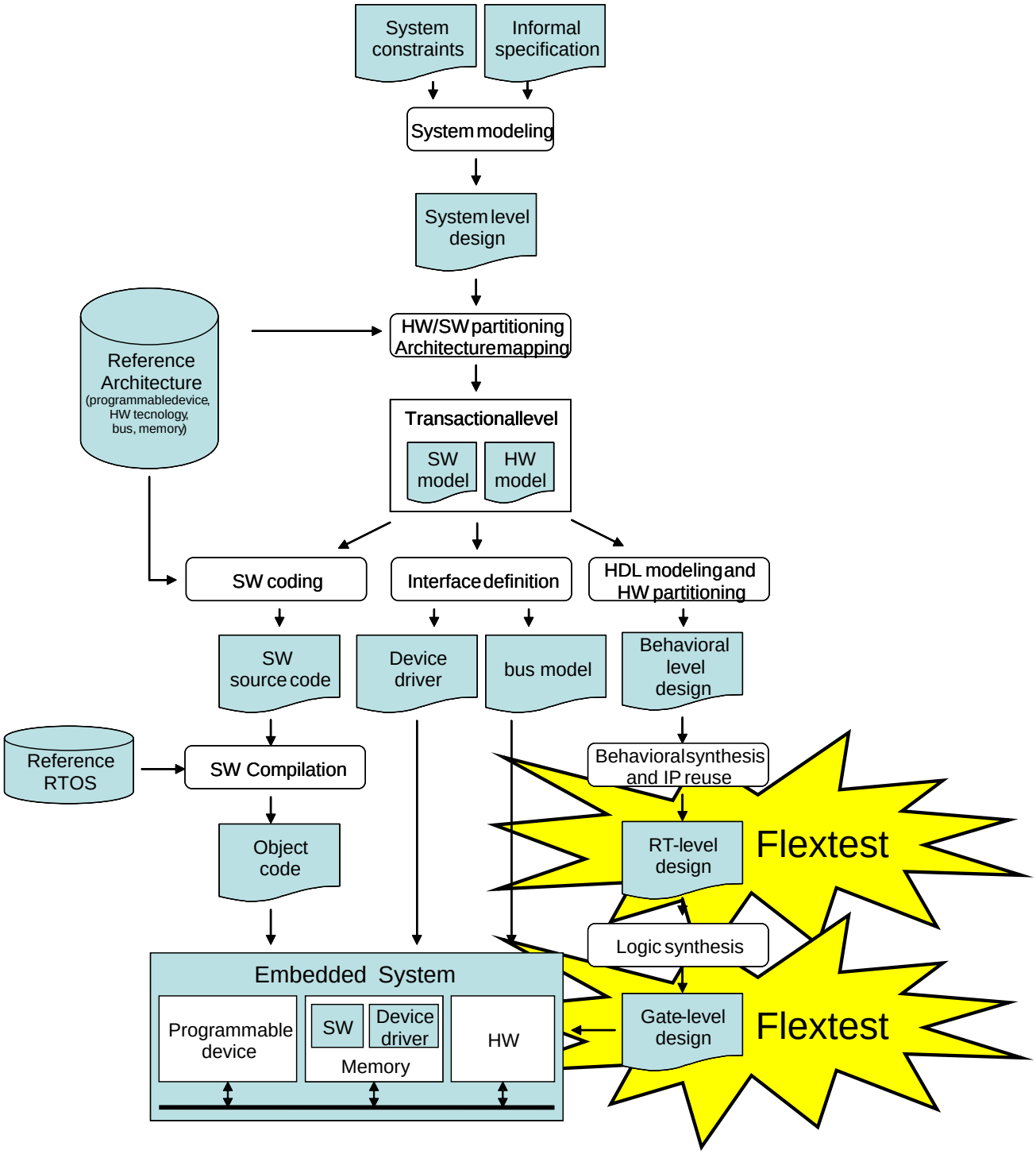


Figure 4: Embedded system design flow.

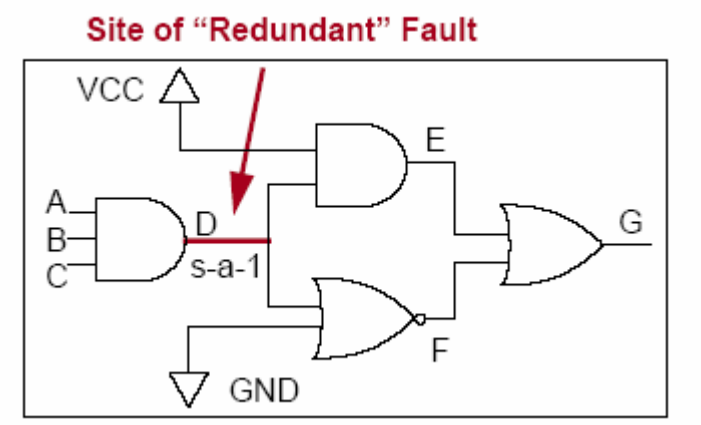
Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

4.2 Flextest configuration and use

Run flextest from a shell and select the vhdl file representing the design under test. Specify the entity name in the top module box, and select library minc.lib as ATPG library. Run the configuration file by selecting *flextest_script_all_faults.do.*

The computation result produces the following information:

- **Fault Coverage:** it consists of the percentage of faults detected from among all faults that the test pattern set tests.
- **ATPG Effectiveness:** it measures the ATPG tool ability to either create a test for a fault, or prove that a test cannot be created for the fault under the restrictions placed on the tool.
- **Untestable (UT) faults:** they are faults for which no pattern can exist to either detect or possible detect them. Untestable faults cannot cause functional failures, so the tools exclude them when calculating test coverage.
- **Redundant (RE) faults:** this fault class includes faults the test generator considers undetectable. After the test pattern generator exhausts all patterns, it performs a special analysis to verify that the fault is undetectable under any conditions. Figure 5 shows the site of a redundant fault. In this circuit, signal G always has the value of 1, no matter what the values of A, B, and C. If D is stuck at 1, this fault is undetectable because the value of G can never change, regardless of the value at D.



- **Testable (TE) faults:** they are all those faults that cannot be proven untestable.