ATPG-PRO

Abstract

The advent of VLSI circuits has resulted in a dramatic increase in the number of components on a single chip. As a consequence of the large circuit density, the probability of fault occuring in the circuit also increases. But complete functional testing of circuits is impractical and the functional patterns generated by designers provided only less coverage. Thus their arised the need to generate test vectors for testing the faulty circuits.

Early test generation algorithms such as boolean difference and literal proposition were not practical to implement on a computer. And also testing very large-scale Integrated circuits with a high fault coverage is a difficult task because of complexity. Therefore many Automatic Test Pattern Generation (ATPG) methods have been developed to address combinational circuits.

As design trends move towards nanometer technologies , new manufacturing testing problems are emerging. During design validation, engineers can no longer ignore the effects of crosstalk and power supply noise on reliability and performance. Current fault modelling and vector generation techniques are giving way to new models and techniques that consider timing information during tst generation that are scalable to larger designs, and that can capture extreme design conditions. Thus for nanometer technologies, many current design validation and ATPG techniques will be needed.

The main objective of the project is to generate test cases for single stuck-at-faults for combinational circuits. Thus it is an electronic design automation technology to find an input sequence that enable testers to distinguish between correct behaviour and faulty circuit behaviour caused by defects.

Introduction

Digital system denotes a complex digital circuit. The complexity of circuit is related to the level of abstraction required to describe its operation in a meaningful way. The level of abstraction is characterized by the type of information processed by the circuit. The level of abstraction includes the Logic level, the Register level, the Instruction set level, the Processor level and the System level.The stimuli and the response defining a testing experiment correspond to the type of information processed by the system under test. Thus testing covers widely different activities and environments such as one or more subsystems testing another by sending and receiving messages, a processor testing itself by executing a diagnostic program, automatic test equipment(ATE) checking a circuit by applying and observing binary patterns.An incorrect operation of the system being tested is referred to as an (observed) error. The causes of observed errors may be design errors, fabrication errors, fabrication defects and physical failures. These are collectively referred to as physical faults. Physical faults can be classified as permanent, intermittent and transient.

Another approach to classification is to classify a fault as logical or parametric. A logical fault changes the Boolean function realized by the digital circuit, while a parametric fault alters the magnitude of a circuit parameter. Stuck-at, bridging and crosspoint faults are three important types of logical faults.A stuck-at fault is said to have occurred if a signal line appears to have its value fixed at either a logical 1 or a logical 0, irrespective of the input signals applied to the circuit. When the signal line is always at logical 1(0), the fault is known as stuck-at-one or SA1 (stuck-at-zero or SA0) fault. Stuck-at faults are one of the simplest faults to analyze. They are proved to be very effective in modelling the fault behaviour of actual devices since they represent the most commonly occuring circuit faults. Hence most of the proposed testing techniques cater to this type of fault. A stuck-at fault model assumes that the faults affect only the interconnections, especially inputs and outputs of the logic gates.

An important task during the manufacturing process is to determine whether the circuit contains a fault, and if so, to locate it so that the faulty components can be replaced. The task of determining whether a fault is present or not is called fault detection, and the task of isolating the fault is fault location. The combined task of fault detection and location is referred to as fault diagnosis. The technique adopted to diagnose faults is testing.

Generally, testing of logical circuits consists of applying a set of input combinations to the primary inputs of the circuit. An input combination which in the presence of a fault produces an output different from the fault-free output is known as a test vector(TV). The set of test vectors used for testing the circuit is called test set.

Test Generation(TG) is the process of finding the set of test vectors that can detect faults in a circuit. The generation of test vectors forms one of the most important and difficult tasks in the testing process. The complexity arises mainly due to the reduction in the ratio of the number of primary inputs and outputs to the number of internal inaccessible points in the circuit. Generating the testset for checking such an enormous number of states is infeasible. However it has been found that the assumption of a single stuck-at fault model provides a good practical basis for testing most circuits. For the circuit with n lines, there are atmost 2n possible single stuck-at faults. However, even for the single stuck-at fault model, there is a need for good techniques for efficient test generation. The objective of test generation is to obtain a minimal complete test set, a complete test set that contains the minimum number of test vectors.

ATPG is an electronic design automation method/technology used to find an input(test) sequence that when applied to a digital circuit, enables tester to distinguish between correct circuit behaviour and the faulty circuit behaviour caused by defects. The generated patterns are used to to test semiconducter devices after manufacture, and in some cases to assist with determining the cause of failure. The effectiveness of ATPG is measured by the amount of modelled defects, or fault models, that are detected and number of generated patterns.

Bibliography

[1] Digital Systems Testing And Testable Design : Miron Abramovici, Melvin A. Breuer and Arthur D. Friedman

[2] Programming Python : Mark Lutz

[3] Logic Design Theory : Nripendra N. Biswas

[4] Logic Design Verification via Test Generation : M. S. Abadir, J. Ferguson and T. E. Kirkland

[5] A Machine for Design Verification and Testing Problems : M. Abramovici and P. R. Menon

[6] Testability Measures- What Do They Tell Us? : V. D. Agrawal and M. R. Mercer

[7] SMART and FAST- Test Generation for VLSI Scan-Design Circuits : M. Abramovici, J. J. Kulikowski, P. R. Menon and D. T. Miller

This Mini Project is done by

IMMANUAL ANDREWS

LUBIN N

NIMISHA V

RAJASREE P R

SACHIN SOMAN