A brief introduction to design for testability dft computer science

DFT is a technique that adds certain testability features to the design which makes an IC more testable. DFT technique improves the controllability and observability of internal nodes, so that embedded functions can be tested easily. Two basic properties determine the testability of a node: 1) controllability, which is a measure of the difficulty of setting internal circuit nodes to 0 or 1 by assigning values to primary inputs (PIs), and 2) observability, which is a measure of the difficulty of propagating a node’s value to a primary output (POs). A node is said to be testable if it is easily controlled and observed. DFT is also a measure of how easy it is to generate test sets having high fault coverage and can be defined as the ability to generate, evaluate, and apply tests to improve quality and minimize test time and test cost. DFT has a cost benefits in both product validation and manufacturing process.

1. 2 Present day scenario of the DFT in the Industries

DFT is a leading method used by all the Chip Implementation Industries to locate and diagnose the faults before the fabrication of any kind of chips. The main advantage of this DFT is all the faults are detected and diagnosed before the fabrication of the chip which makes the chip implementation easier according to the required specification and reduces life cycle cost, wastage of resources etc.

1. 3 Motivation to do this project

In past, testing of complex design before or after the hardware fabrication with only primary ports has become a difficult task. This poses many new design challenges in the field of testing. Controlling the internal nodes of the

design using only primary IO’s has become complicated. To overcome this, DFT technique is introduced before the fabrication of chip which provides controllability and observability to all the nodes in the design through scan chains with input and output ports at the gate level. Faults that are present anywhere in the design can be detected and diagnosed. Further Timing analysis is done at IP and top level to validate all possible timing paths with timing constraints for any timing violations.

1. 4 Objective of the project work

The objective of this project is to implement the DFT technique for DUT to increase the test coverage without increasing the test cost during testing of chip at gate level. This DFT technique includes scan insertion and compression for DUT. Further Static Timing Analysis (STA) is done for DFT inserted DUT to fix all timing violations. Formal verification is done between two versions of design (RTL-RTL, RTL-Netlist, Netlist-Netlist) for logical equivalence check.

1. 5 Project Work Schedule

Ramp up aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦..

Synthesis aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦..

Scan Insertion for DUT aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦.

Scan Compression aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦.

Timing Analysis aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦

TetraMAX aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦

Formal verification aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦aˆ¦

1 month

1 month

2 months

2 months

2 months

1 month

1 month

1. 6 Organization of the project report

This report is a guide for the work done in this project. The First chapter is a brief introduction which includes mainly objective and Time plan of the project. The second chapter discusses about the Literature review and Background theory of the project. The third chapter contains methodology adopted to implement the project. The fourth chapter elaborates on the results obtained and inference of the results. The Fifth chapter entails to the conclusions and future scope of the project

CHAPTER 2

BACKGROUND THEORY

This chapter includes the following topics

Introduction to DFT

Background theory

Types of Faults

2. 1 Introduction to DFT

DFT refers to the design techniques that make the task of subsequent testing easier. There is definitely no single methodology that solves all embedded system-testing problems. There also is no single DFT technique, which is effective for all kinds of circuits. DFT techniques can classified into two categories

Ad-hoc method

structured method

Ad-hoc Method

In this method large designs are partioned to small design to reduce the test cost and test points are added manually to the designs to increase testability and observability. The controllable points (cp) are active points and observable points (op) are passive ones.

Fig 2. 1. Ad-hoc method – Test point insertion

Disadvantages

Experts are needed and test generation is often manual

No guarantee of result (poor fault coverage)

Increase design iterations, hence not suitable for large designs

Structured Method

Structured method will provide good controllability and observability of internal state variables for testing by serial shifting of data. This method includes

Scan technique

Scan design is implemented to provide good controllability and observability of internal state variables for testing a circuit. It is also effective for circuit partitioning. A scan design with full controllability and observability turns the sequential test problem into combinational one.

Scan design requirements

One (more) test control ports are required at PI and PO, which are called as scan-in and scan-out port respectively

Test structure (hardware) is added for DUT

Normal Flip flops are converted to Scan Flip flops (Scan cells) and are connected so that they behave as a shift register in test mode. Scan Flip Flop is a flop with extra logic

Combinational ATPG is used to obtain tests for all testable faults in the combinational logic

Shift registers tests are applied and ATPG are converted into scan sequences for use in manufacturing test.

Different types of Scan cells used for scan design

Mux based scan cells

In this approach MUX is inserted at the input side of flip flop. MUX’s select line is connected to scan enable, which decides the operating mode of the design. This MUX is added to increase controllability at input side of each flip flop. Two types of inputs, Data and Scan input are connected to D0 and D1 pins of MUX respectively. Based on scan enable signal, corresponding input is fed to flop.

Fig 2. 2. MUX based Scan cell

Clocked Scan cells

In clocked scan cells, input selection is conducted using two Independent clocks. In normal/capture mode, the data clock is used to capture the contents present on the data input into the clocked scan cell. In shift mode, the shift clock is used to shift the new data from scan input to clocked scan cell, while the content of clocked scan cell is being shifted out.

Fig 2. 3. Clocked scan cell

LSSD

It is a latch based design which guarantees race-free and hazard-free system operation as well as testing. It is insensitive to component timing variations such as rise time, fall time and delay. It uses two latches (one for normal operation and another for scan) and three clocks. LSSD requires that the circuit to be level sensitive.

Normal mode : A-clk = B-clk = 0, sys-clk = 0 —- 1

Test (scan) mode: sys-clk = 0, A-clk, B-clk = 10 —- 01 to shift scan data through Latch Fig 2. 4. Level sensitive scan design

Advantages

FSM is reduced to combinational logic as for as testing is concerned, Hazards and Races are eliminated, which simplifies test generation and fault simulation

Disadvantages

Complex design rules are imposed on the designers

Asynchronous designs are not allowed in this approach

Sequential routing of latches can introduce irregular structure

Test application becomes very slow process and not good for memory intensive designs

Boundary Scan

JTAG or Boundary scan is primarily used for testing board connections, without unplugging the chip from the board. Boundary scan is accessed through 5 pins, TCK, TMS, TRST, TDI and TDO. It builds capability of observing and controlling pins into each chip to make board test easier. Chips with internal scan chains can access the chains through boundary scan for unified test strategy.

Fig 2. 5. Boundary Scan

Background theory

2. 2. 1 Traditional ASIC Design Flow

Fig 2. 6 ASIC design flow

Chip design commences from the concept of idea dictated by the market. These ideas are then translated into architectural and electrical specification. The Architectural specification will define the functionality and partioning of chip. The electrical specification will define the inter connection of these blocks in terms of timing information. Next step is to implement these design specifications. In past, early days, these specifications are implemented manually by drawing schematics and layouts, which is very time consuming and impractical for design reuse. To overcome these problems Hardware description language (HDL) were developed. Most commonly used are Verilog and VHDL.

RTL code is developed according to design specification by Verilog or VHDL. To check the functionality of the design RTL code is simulated using test bench. If the code meets all the design specifications, then synthesis is

carried out. Synthesis is done by a Synopsis tool called Design Compiler (DC). DC translates the RTL design into a gate level optimized mapped netlist. These RTL and netlist are formally verified for logical equivalence by a formality tool.

Static Timing Analysis (STA) will allow the user to analyze all the critical path in the design. Prime Time (PT) is used for timing analysis. In pre-layout STA PT uses wire load model specified in the library to estimate the delays. The timing constraints are fed to PT by providing the relation between primary IOs and clocks. If the timing for all the critical path is satisfactory, a constraint file is developed for forward annotation to layout tools. This standard constraint file is Standard Delay Format (SDF).

If timing in pre-layout STA is satisfied, Then Placement and routing is done. It consists of five steps; Initial floor planning, Cell placement, Clock tree (CT) insertion, Global and then detailed routing. CT insertion is done to check the quality of cell placements. The netlist generated during synthesis lacks from CT information. Hence CT is re-routed to netlist and formally verified for CT inserted netlist and Original netlist. Global routing will estimates the actual delays. If timings are met, Detailed routing is done where real delays are estimated. In next step, Post layout STA the extracted delays are back-annoted until timing requirements are satisfied. Before Tape out of the design, if there is any hardware bug encounters in the design, this bug can be removed by Engineering Change Order (ECO) instead of redesigning.

2. 3 Types of Faults

Fault is basically the physical defect in the circuit. It is classified mainly into four different classes;

1) Permanent Faults

These kinds of faults will be present permanently in the design. It has four types

Stuck-at fault : Fault in the logic gate results in one of its input or output node is fixed at either logic 0 or logic 1. In general for m number of inputs, there will be 2(m+1) number of stuck-at faults will be present.

Delay fault : It includes transition and path delay faults. Transition delay fault model includes single node slow-to-rise and slow-to-fall faults. Path delay faults tests and categorizes critical timing paths in the design.

IDDQ fault : IDDQ fault model assumes that the circuit defect will cause excessive current drain due to internal short circuit from node to ground or to power supply

Bridging fault : Two or more signal lines are connected accidentally in the logic circuit

2) Temporary Faults

It includes Intermittent and Transient faults. Intermittent faults are recurring faults which will reappear on regular basics. Transient faults are non recurring and non repairable faults, because there is no physical damage to the hardware.

3) Equivalent Faults

Two faults are said to be equivalent if every test for one fault also detects the other.

4) Redundant Faults

Faults are said to be redundant/latent its effect does not results on the output logic.

CHAPTER 3

METHODOLOGY

This chapter contains the methodology used to implement this project. It includes the following steps,

Synthesis – to translate RTL designs to gate level netlist

DFT (scan) Insertion

DFT compression (codec insertion)

Timing analysis

TetraMAX – for ATPG generation

Formal Verification

3. 1 Synthesis

Synthesis is a process to translate the RTL designs into a gate-level, optimized, mapped netlist. The basic synthesis flow is carried out as shown below,

Fig 3. 1. Basic synthesis flow

3. 1. 1 To generate Netlist for given RTL using DC

Develop HDL files

Usually the input files for tool are written in Verilog or VHDL. When writing the HDL files, designers had to follow design partioning and coding guidelines to achieve the best synthesis result possible. This is given by the RTL designers. This step is not included in synthesis.

Specify libraries

Initially, all the libraries required for synthesis are stored in . synopsys_dc. setup file, which includes Link library, Target library, GTECH library and Design ware library. This setup file will automatically links during synthesis in the design environment.

Ex: Set link_library

/sw/unicad/CLOCK65LPHVT/5. 1/libs/ CORE65LPHVT_wc_1. 10V_125C_10y. db

/sw/unicad/CORE65LPLVT/5. 1/libs/ CORE65LPHVT_wc_1. 10V_125C_10y. db

/sw/unicad/CORE65LPHVT/5. 1/libs/ CORE65LPHVT_wc_1. 10V_125C_10y. db

Set target_library

/sw/unicad/CORE65LPSVT/5. 1/libs/CORE65LPSVT_wc_1. 10V_125C_10y. db

Reading design

Analyze and elaborate commands are used to read Verilog/VHDL design into dc-shell Environment. Analyze command will stores the current design in an intermediate format in the specified library. Elaborate will builds the design from this intermediate format.

Ex: analyze -format verilog/vhdl < . v/. vhd file >

elaborate < top_design_name >

Current Design

The current design is set by using set current_design command in design compiler.

Ex: set current_design < top_design_name >

Link

Link command is required to locate all the designs and library components and connects (links) them to current design.

Ex: link

Uniquification

Removes multiply-instantiated hierarchy in the current design by creating a unique design for each cell instance

Ex: uniquify

Set design Constraints

The design constraints like clock latency, clock transition, clock uncertainty, IO delays etc... are sourced.

Ex: create_clock -name < clk_name > -period 10 -waveform {0 5} [get_ports < port_name>]

set_clock_latency -max 1 [get_clocks {clk_name}]

set_clock_uncertainty -setup 0. 47 [get_clocks clk_name]

set_clock_uncertainty -hold 0. 25 [get_clocks clk_name]

set_input_delay 5. 0 min/max -clock < clk_name> [get_ports < port_name>]

Compile

There are two methods of compiling the design. Top down and Bottom up. In compile stage, optimization and mapping is done. compile -scan will do scan replacement in addition to optimization and mapping.

Ex: compile -scan

Resolve Design problems

In this stage, the design problems are fixed by changing the constraints until the design requirements are met.

Write stage

The translated RTL design into the netlist is saved in the form of . v/. ddc file by using write command

Ex: write -format verilog -hierarchy -output < outputfile. v >

write -format ddc -hierarchy -output < outputfile. ddc >

3. 2 DFT Insertion

Once the netlist is available, Scan insertion is done to increase the controllability and observability of the internal nodes in the design.

DFT involves extra design effort invested in making an IC more testable. This extra design effort will increase the controllability and observability of the internal nodes in the design. DFT includes inserting or modifying logic

There are mainly two DFT techniques

Structured DFT – Highly Automated

Adhoc DFT – Adding testability logic at designers discretion

3. 2. 1 Modes of Operation

Scan operates in shift and capture cycles. Data is injected into the device through primary inputs and is shifted out of the device through the “ SD” input port of the flops. Assume scan_en port is active high for shift operation. Once the chain has been flushed out and compared, the scan_en signal is toggled (driven low). Now a single clock pulse is applied to capture the data

into the flops through the “ D” inputs, before the scan_en is toggled again (driven high) and the data shifted out for comparison. It is as shown below

Fig 3. 2. Scan Shift

Fig 3. 3. Scan capture

3. 2. 2 Scan Insertion (DFT insertion) flow

Fig 3. 4. DFT insertion flow chart

Read design : Design is read in the form of ddc/verilog

Create test protocol : It will describes how the signal will operates in scan mode

Pre DFT-DRC : Performs design rule checking to scan design and will list out design rule violations

Specify scan architecture : It will define type of scan style, number of scan chains, chain length, handling multiple clocks, lockup elements, registers to be omitted from scan chains in the design

Preview DFT: Checks scan architecture before implementing to actual design. This allows for quick iteration cycles when changes need to be made in scan architecture

Insert scan : Scan architecture is inserted to the design

Post DFT-DRC : validates that scan chain trace properly

3. 2. 3 Three types of DRC checks performed by DFTC on the design

RTL DFT-DRC : Run during RTL development phase to identify DFT issues early in design flow

Pre DFT-DRC : Run prior to scan insertion to determine which scanable elements can be put on scan chains

Post DFT-DRC : Run after scan insertion to validate that the implemented scan chains can be traced

3. 2. 4 Some issues generated in DFT analysis and its solution

a) No clock pulse at the scan flop due to gating of clock

Fig 3. 5 . Clock gated scan flop

Solution:

DRC automatically infers clock signals by tracing back from flip-flop clock pins to PIs

To fix this violation, AND gate is made transparent

If A is not a PI, add logic to inject 1through PI test mode port

b) No clock pulse at scan flop due to dividing down of the clock

Solution: Adding a bypass Mux that selects an external test clock or functional clock by holding TM port high.

Fig 3. 6 Clock divider solution with test clock

Fig 3. 7 Clock divider solution with functional clock

c) Unexpected asynchronous reset asserted at flop

Solution: Asynchronous set/reset is controlled by combinational block

Fig 3. 8 Asynchronous set/reset solution

d) Hold time problem due to short net or clock skew

Solution: Clock tree insertion to reduce total delays and clock skew

e) Tri-state DFT issue – fault on enable pin of tristate buffer is not detectable.

Solution: Adding pull-up resistors, bus keepers, replacing tristate buses by ordinary multiplexed buses

f) Bus contention – tristate buses are subjected to problems.

Solution: Enabling a single tristate driver per bus, adding decoders

g) Multiple clock domains mixing, same clock edges mixing.

Solution: lockup latch insertion

Lockup latch will holds the data for further half cycle during negative cycle of launch flop. This makes capture flop to capture data without violations. It also helps in concatenation of flops of different domains at top level

http://img. cmpnet. com/planetanalog/features/Mentor_Fig2. gif

Fig 3. 9 Lockup latch

3. 2. 5 Four methods to fix DRC violations

By editing RTL code and resynthesize design with DFT logic

By using Auto fix to insert bypass or injection logic. This Auto fix feature can automatically fix DFT rule violations

By using UDTP and editing netlist with respective commands

3. 2. 6 Balancing scan cells in chain

Scan chain with balanced scan cells and proper grouping will help in

Preventing test application time on tester and pattern memory on tester

Cost reduction

Fig 3. 10 Balancing scan cells

3. 2. 7 Limitations of DRC

They cannot take gate/net delays into account

They are unaware of clock tree delays/skews

Test tools do not perform Static Timing Analysis (STA)

3. 2. 8 Commands used for Scan Insertion

read_design (. v/. ddc) format

set current_design (top)

link

source constraints

set_scan_configuration -style (multiplexed flip flop)

compile -scan

set_dft_signal -view (exist/spec) – type (scan_clk/reset/) -port (port_name)

set_dft_signal -type (scan_data_in/scan_data_out/scan_enable) -port (port_name)

create_test_protocol

dft_drc -pre

preview_dft

insert_dft

dft_drc -post

write_scan_def -outupt

write_test_protocol -output

write -format (verilog/vhdl) -hierarchy -output

3. 3 DFT compression (Codec insertion)

DFT Max is the next generation of DFT synthesis. In this Adoptive scan technique is used for data volume compression with no impact of test coverage. The main advantages are

Tester cycle reduction

Test application time reduction

Uses minimum number of ports

In this technique de-compressor logic is added at the input side of flop to achieve controllability and compressor logic is added after flop for observability

Due to the addition of compressor and decompressor logic, Adaptive scan compression is also known as Codec insertion.

Fig 3. 11 Compressor and decompressor logic added to scan chains

There are two modes of operation in DFT Max,

Internal scan mode (regular scan) : In this mode, de-compressor and compressor logic are bypassed from accessing the scan chain

Scan compression mode : DFT compiler allows de-compressor and compressor logic to access scan chain

3. 3. 1 Scan Compression working

Available long scan chains are split into shorter chains. Shorter chains required less time to load and less data to be loaded on tester. It is as shown in figure

Fig 3. 12 Adaptive scan compression

Test application time = patterns * length of scan chain

With higher level of compression come higher area overhead, increased risk of routing congestion and only a small incremental improvement in Test Application Time Reduction (TATR) and Test Data Volume Reduction (TDVR).

3. 3. 2 Hierarchical Adaptive Scan Synthesis (HASS)

In HASS Codec insertion is done in bottom-up flow. All the adaptive scan logic is placed at core level and codec is inserted block by block. At top level these codec inserted blocks are combined. This mainly reduces the routing congestion at top level and issues can be debugged easily.

An adaptive scan core contains scan chains that are configured in two modes of operations compression mode (scan compression mode) and reconfigurable scan mode (internal scan mode). A pure scan core contains scan chains configured in a single mode of operation (internal scan mode). At the chip level, these cores are integrated to provide two modes of operations:

aˆ? Compression mode – activates all adaptive scan chains as well as all pure scan core chains

aˆ? Reconfigurable scan mode – activates the reconfigured scan chains of each adaptive scan core as well as all pure scan core chains.

Fig 3. 13 Hierarchical Adaptive Scan Synthesis Architecture

3. 3. 3 Tester Time Reduction Graph

Fig 3. 14 Tester time reduction graph

3. 3. 4 Commands used for Codec insertion

– read_verilog/ddc (compiled netlist)

– set current_design top

– link_design

– source constraints

– set_scan_configuration (DFT signal type, chain count, Max length)

– set_scan_state -test_ready

-set_dft_configuration -scan_compression enable

– set_scan_compression_configuration -test_mode

– create_test_protocol

– dft_drc (pre DRC check)

– preview_dft

– insert_dft

– dft_drc – coverage_estimate (post DRC check)

3. 4 Timing Analysis

Timing analysis is performed by a synopsys tool called primetime (PT). PrimeTime is a full-chip, gate-level static timing analysis tool that is an essential part of the design and analysis flow for today’s large chip designs. PrimeTime exhaustively validates the timing performance of a design by checking all possible paths for timing violations, without using logic simulation or test vectors.

3. 4. 1 Types of Checking performed

PrimeTime performs the following types of design checking:

aˆ? Setup, hold, recovery, and removal constraints

aˆ? User-specified data-to-data timing constraints

aˆ? Clock-gating setup and hold constraints

aˆ? Minimum period and minimum pulse width for clocks

aˆ? Design rules (minimum/maximum transition time, capacitance, and fan-out)

Static timing analysis is a method of validating the timing performance of a design by

checking all possible paths for timing violations. PrimeTime checks for violations in the same

way that we would do it manually, but with much greater speed and accuracy. To check a design for violations, PrimeTime breaks the design down into a set of timing paths, calculates the signal propagation delay along each path, and checks for violations of timing constraints inside the design and at the input/output interface.

3. 4. 2 Timing Paths

The first step performed by PrimeTime for timing analysis is to break the design down into a set of timing paths. Each path has a start point and an endpoint. The start point is a place in the design where data is launched by a clock edge. The data is propagated through combinational logic in the path and then captured at the endpoint by another clock edge. The start point of a path is a clock pin of a sequential element, or possibly an input port of the design (because the input data can be launched from some external source). The endpoint of a path is a data input pin of a sequential element, or possibly an output port of the design (Because the output data can be captured by some external sink). The different timing paths are as shown below in the figure.

Fig 3. 15 Timing paths

3. 4. 3 Delay Calculation

After breaking down a design into a set of timing paths, PrimeTime

calculates the delay along each path. The total delay of a path is the sum of

all cell and net delays in the path. Cell delay is the amount of delay from input to output of a logic gate in a path. PrimeTime calculates the cell delay from delay tables provided in the technology library for the cell. Net delay is the amount of delay from the output of a cell to the input of the next cell in a timing path. This delay is caused by the parasitic capacitance of the interconnection between the two cells, combined with net resistance and the limited drive strength of the cell

driving the net

Total delay = Cell delay + Net delay

3. 4. 4 Constraints Checking

After PrimeTime determines the timing paths and calculates the path delays, it can check for violations of timing constraints, such as setup and hold constraints. A setup constraint specifies how much time is necessary for data to be available at the input of a sequential device before the clock edge that captures the data in the device. This constraint enforces a maximum delay on the data path relative to the clock path. A hold constraint specifies how much time is necessary for data to be stable at the input of a sequential device after the clock edge that captures the data in the device. This constraint enforces a minimum delay on the data path relative to the clock path.

In addition to setup and hold constraints, PrimeTime can also check recovery/removal constraints, data-to-data constraints, clock-gating setup/hold constraints, and minimum pulse width for clock signals. The

amount of time by which a violation is avoided is called the slack. For example, for a setup constraint, if a signal must reach a cell input at no later than 8 ns and is determined to arrive at 5 ns, the slack is 3 ns. A slack of 0 means that the constraint is just barely satisfied. A negative slack indicates a timing violation. The figure shows setup and hold checks

Fig 3. 16 Setup and Hold check

3. 4. 5 Timing Exceptions

When certain paths are not intended to operate according to the default setup/hold behavior assumed by PrimeTime, we should specify those paths (false paths, multi cycle paths etc…) as timing exceptions. Otherwise, PrimeTime might incorrectly report those paths as having timing violations.

3. 4. 6 Analysis Flow in PT

Specify library

Read gate-level design into pt-shell environment

Add constraints to design

Update & Check designs

Perform full analysis and examine results by Report timing, clocks, constraints and analysis coverage

Generate reports

3. 4. 7 Commands used in Timing Analysis (TA)

read_verilog (netlist)

current_design

link

update_timing

check_timing

report_timing

report_constraints -all_violators

report_clocks

report_analysis_coverage

3. 5 TetraMAX

TetraMAX is a high-speed, high-capacity automatic test pattern generation (ATPG) tool. It can generate test patterns that maximize test coverage while using a minimum number of test vectors for a wide variety of design types and design flows. It is well suited for designs of all sizes up to millions of gates. It can be used either in GUI mode or in normal mode.

3. 5. 1 TetraMAX ATPG Features

Functional testing and stuck-at testing are the traditional circuit testing methods. With functional testing, the tester applies a sequence of input data and detects the resulting sequence of output data. The output sequence is

compared against the expected behavior of the device. Functional testing exercises the device as it would actually be used in the target application. However, this type of testing has only a limited ability to test the integrity of the device’s internal nodes.

With scan testing, the sequential elements of the device are connected into chains and used as primary inputs and primary outputs for testing purposes. Using ATPG techniques, you can test a much larger number of internal faults than with functional testing alone. The goal of ATPG is to set all nodes of the circuit to both 0 and 1, and to propagate any defects to nodes where they can be detected by the test equipment.

3. 5. 2 ATPG Modes

TetraMAX offers three different ATPG modes:

Basic-Scan ATPG

In Basic-Scan mode, TetraMAX operates as a full-scan, combinational-only ATPG tool. To get high test coverage, the sequential elements need to be scan elements. Combinational ROMs can be used to gain coverage of circuitry in their shadows in this mode.

Fast-Sequential ATPG

Fast-Sequential ATPG provides limited support for partial-scan designs. In this mode, multiple capture procedures are allowed between scan load and scan unload, allowing data to be propagated through non-scan sequential elements in the design such as functional latches, non-scan flops, RAMs and

ROMs. However, all clock and reset signals to these non-scan elements must still be directly controllable at the primary inputs of the device.

Full-Sequential ATPG

Full-Sequential ATPG is similar to Fast-Sequential ATPG which supports multiple capture cycles between scan load and unload and thus increasing test coverage in partial-scan designs. Clock and reset signals to the non-scan elements do not need to be controllable at the primary inputs; and there is no specific limit on the number of capture cycles used between scan load and unload.

3. 5. 3 Design flow using DFT compiler and TetraMAX

Fig 3. 17 Design flow using DFT compiler and TetraMAX

TetraMAX is compatible with a wide range of design-for-test tools such as DFT Compiler. The design flow using DFT Compiler and TetraMAX ATPG is recommended for maximum ease of use and good quality of results. Figure 3. 17 shows how TetraMAX ATPG fits into the DFT Compiler design-for-test flow for a module or a medium-sized design of less than 750K gates. Starting with an HDL netlist at the register transfer level (RTL), within DFT Compiler first Test-Ready compilation is done, which integrates logic optimization and scan replacement. The compile -scan command maps all sequential cells directly to their scan equivalents. At this point, we still don’t know whether the sequential cells meet the test design rules. Next, test design rule checking is performed; the check_scan command reports, if any sequential cells that violates test design rules. After resolving the DRC violations,

preview_scan command is used to examine the scan architecture. This procedure is repeated until the scan architecture is satisfied, then insert_scan command is used, which implements the scan architecture.

Finally, the check_scan command is used once again to identify any remaining DRC violations and to infer a test protocol. When the netlist is free of DRC violations, it is then ready for ATPG. For medium-sized and smaller designs, DFT Compiler provides the write_test_protocol command, which allows writing out a STIL protocol file. TetraMAX then reads the STIL protocol file and design netlist.

STIL file basically contains

Top level signals

Scan structures

Timing

Procedures

Macro defs

3. 5. 4 ATPG Design flow

Figure 3. 17 illustrates the basic ATPG design flow. This flow consists of the following steps:

1. If necessary, netlist is preprocessed to meet the requirements of TetraMAX.

2. Read the netlist.

3. Read the library models.

4. Build the ATPG design model.

5. Read the STIL test protocol file, generated by DFT Compiler or another source. Perform test DRC and make any necessary corrections.

6. To prepare the design for ATPG, set up the fault list, analyze buses for contention, and set the ATPG options.

7. Run automatic test pattern generation.

8. Review the test coverage and rerun ATPG if necessary.

9. Rerun ATPG.

10. Run pattern compression

11. save the test patterns and fault list.

Fig 3. 18 ATPG Design flow

3. 5. 5 Commands used in TetraMAX

read netlist

run build_model

run drc

report_violations -all

set drc -trace (for scan chain tracing)

run atpg

report summaries

report fault -summaries

3. 6 Formal Verification

Formal verification is a technique that performs the validation of the design using Mathematical methods. Formal verification is an alternate to verification through simulation. The main advantage is, verification is done by mathematical methods without using test vectors. Two versions of design (RTL-RTL, RTL-Netlist, Netlist-Netlist) are logically checked for logical equivalence. To perform this equivalence checking, formality tool (synopsys EDA tool) is used.

Equivalence checkers prove or disprove that one design representation is logically equivalent to another. That is to say, they are used to prove that two circuits will exhibit the same exact behavior under all conditions despite different representations

Fig 3. 19 Formal verification

The Reference (golden) design and implementation design is loaded to fm-shell and setup is performed. Formality tool generates compare points for matching. The compare points are:

Primary outputs (Pos)

Internal registers (sequential elements)

Input pins of black boxes and

Nets driven by multiple drivers.

The tool performs matching in five steps;

Exact – name matching

Name filtering

Topological Equivalence

Signature Analysis

Compare point matching based on net names

If all the compare points of Design A and Design B are matched, Equivalence check is done successfully.

3. 6. 1 Commands used to do Formal Verification in fm_shell

read_db -r -tech -lib LIBS [list of . db libraries for reference design]

read_verilog -netlist -r

set_top

read_db -i -tech -lib LIBS [list of . db libraries for implementation design]

read_verilog -netlist -i

set_top

match

verify

report_compare_points

report_dont_verify_points

report_passing_points

CHAPTER 4

RESULT ANALYSIS

In this chapter includes the results of,

DFT coverage of an IP

Timing Analysis result of an IP

TetraMAX results

Formal Verification

4. 1 DFT Coverage of an IP

Inference: This report is generated after scan and codec insertion to the design under test netlist. It will give the test (DFT) coverage of current design.

4. 2 Timing Analysis result of an IP (Top level)

Inference: This is an analysis coverage report generated during timing analysis. Initially timing analysis is done at the IP level by fixing all setup violations. Setup violations can be fixed by reducing the clock speed or by reducing the combinational logic delay. At the top level all other remaining violations are fixed. Hold violation can be fixed by increasing combinational logic delay by adding buffers or resizing (upsizing or downsizing) the cells. Recovery is similar to setup check for an asynchronous signals. Removal is similar to hold check for asynchronous signals.

4. 3 TetraMAX results

4. 3. 1 Test Coverage before ATPG run

Inference: This report is generated before running ATPG. The main reason for lower test coverage is due to AU and AN fault. AU faults are ATPG untestable faults and AN faults are ATPG undetectable.

4. 3. 2 Test coverage after ATPG run

Inference: This report is generated after ATPG run. TetraMAX automatically generates ATPG which increases the test coverage.

4. 3. 3 Scan Chain Tracing

Internal Scan Mode

Inference: This is an Internal scan mode report, which infers that all the scan chains are properly built and there is no any blockage or breaking of scan chain. It contains 8 scan chains with equally distributed scan cells in each

scan chain. Since all the scan chains are very long, test application time increases. To over come this Scan-compression mode is used.

Scan – Compression Mode

Inference: This is Scan-compression mode report. In this there are 247 scan chains with 250 scan cells in each scan chain. Here maximum scan cells in each scan chain is fixed to 250. Once after reaching 250 scan cells in a chain, tool will automatically break the current scan chain and will start with new one. This reduces the test application time.

4. 4 Formal Verification

Example of RTL – Netlist verification

Inference: Initially RTL is developed for Bit counter and loaded as a reference design in formality setup. Netlist generated after the synthesis is loaded as an implementation design in formality setup. Then the formality tool generates compare points automatically. Once after generating compare points in both the design, matching is done. This report infers that all the compare points in the reference design are matched with implementation design successfully. After that verification is done.

Verify Report

Inference: If all the compare points matched, verification is done. This report infers that all the compare points are verified successfully and formal verification is succeeded. Similarly RTL-RTL, Netlist-Netlist verification can be done.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE OF WORK

The advancement in SoC technology results in embedding multiple cores on a single chip. This increases the difficulty in testing the design at the chip level and also reduces the access to the internal nodes which makes more difficult in detecting and diagnosing the faults in DUT. To overcome these problems and to enhance the test coverage DFT method is implemented which includes mainly scan and codec insertion.

In this project DFT analysis is done to identify the faults in the complex designs and studied to perform synthesis for a given RTL, scan insertion is done to increase the controllability and observability of internal nodes, scan compression is done to reduce number of tester cycles and test application time without increasing the test cost, Timing analysis is performed for codec inserted design to fix the timing violation and TetraMAX is run to maximize the test coverage and to validate the scan chains.

5. 1 Conclusion

With the high level of integration of today’s IC’s, the microelectronics industry now needs testability implementation more than ever. DFT techniques implemented during design can save hundreds of thousands of dollars by reducing the need for redesign because of system failures. However, with most designs still being implemented in silicon without DFT consideration, designers are being forced to add DFT on second and third generations of chip designs as an “ after thought.” The DFT techniques that

are the easiest to insert during the final stages of design and take the least amount of silicon area have gained the popularity.

In this project DFT insertion is done for a design to locate and diagnose faults at gate level. Test coverage of the design is increased from 67% to 95%.

5. 2 Future Scope of work

BIST insertion can be done along with DFT insertion to test macros in DUT which further improves the test coverage and provide best results.

As the technology is shrinking day by day, chip size reduces and more systems get embedded on a single chip resulting in routing congestion. At that stage, Testing can only be done by using DFT method without increasing the test cost.

After developing scan chains for DUT, Users may destroy or modify the existing scan chains. To provide security for the scan chains cryptography technique can be embed along with scan insertion.

DFT method can be extended for IDDQ testing in Low Power Analysis and for testing any kind of chips available in any fields like medical, automobile etc. for accurate test results.