Verification of Distributive Arithmetic based Discrete Wavelet Transform IP Core

Lakshmi Kiran A¹, Cyril Prasanna Raj P²
1. Lecturer – VLSI System Design, 2. Asst. Professor MSRSAS
Centre for VLSI System Design
M.S.Ramaiah School of Advanced Studies, Bangalore

Abstract
Establishing the correctness of integrated circuit designs continues to be a daunting task as the designs continue to increase in size. The demand for shorter design cycles only makes this problem more severe, which necessitates uncovering any errors as early in the development process as possible. Therefore, detecting and removing errors from a high level implementation, referred to as design verification, has received much attention.

The variety of methodologies that have been suggested to verify designs may be classified under two broad categories, simulation and formal methods. Functional verification of the design using validation tests is the primary technique used within the industry to verify large designs due to its simplicity and scalability. Exhaustive simulation is often impractical. So, we will go for Formal Verification (by Formal Compiler), Design for Testability (by DFT Compiler) and finally we will go for TetraMax for generating ATPG.

In this dissertation we described an approach to create a test structure that includes the logic to select and configure the design for test i.e logic blocks and memory blocks are separated initially. Logic cells are tested by scan insertion and memory blocks are tested by generating BIST. The test structure allows the selection of DFT steps such as scan, bist, delta iddd, boundary scan and analog test mode. The test structure is inserted in the netlist and equivalence check is used to ensure the connectivity of test structure. Overall process of insertion of test logic is automated and simplified by using simple Tcl script.

1. INTRODUCTION
The modern integrated circuit is a complex device manufactured by a complicated fabrication process. At any stage in the process physical defects can occur in the silicon, which can render the resultant device useless. Transistor construction can be flawed, metal tracks can be shorted or open circuit, die can be damaged during wafer sawing or packaging and human error or mechanical failure can result in a processing step being incorrectly implemented. For this reason ICs are tested twice during manufacture.
The first test occurs at the completion of fabrication. Each die is subjected to a Wafer Test during which a series of test patterns are applied to the input bond pads by wafer probes and the resulting signals on the output bond pads captured for analysis.

A second test occurs when the devices have been packaged, often using the same set of test patterns. Customers may also test the devices prior to and after assembly into a product. Failure to detect even a small percentage of faults can be costly. The designer must therefore supply a comprehensive set of test patterns capable of identifying every possible fault that can occur in the device.

The paper is organized as follows; Section 2 describes the brief concept of the simulation based functional verification. Section 3 & 4 explains DFT and TetraMax followed by problem statement and methodologies applied and their corresponding results and conclusions.

2. FUNCTIONAL VERIFICATION
In Simulation based Functional Verification methodology we will perform system level verification i.e. selective module-level verification. We will perform those by writing test programs in Assembly, C and HDL languages. There will be detailed verification plan for each module. Gate level simulations with all directed tests complete at required operating conditions. However, exhaustive simulation is often impractical, and the likelihood that these tests will uncover subtle errors is diminishing.

Formal verification is the process of checking whether a design satisfies some check (properties). Formal verification means proving that a property holds of a model of a design. Formal verification techniques have gained large attention for ensuring 100% functional correctness. Verification provides mathematical proof, in contrast to conventional simulation and test, which tells only that nothing went wrong on the specific cases we tried. At present we are using two types of formal verification [10]. They are

- BDD based formal verification
- SAT based formal verification

2.1 BDD Based Formal Verification

BDD stands for the “Binary Decision Diagrams”. Binary Decision Diagrams are commonly used to implicitly represent large solution spaces in
combinational and sequential problems that arise in synthesis and verification. A BDD is a directed acyclic graph constructed in such a way that its directed paths represent objects of interest.

A BDD is just a data structure for representing a Boolean function. Bryant introduced the BDD in its current form, although the general ideas have been around for quite some time. Conceptually, we can construct the BDD for a Boolean function as follows. First, build a decision tree for the desired function, obeying the restrictions that along any path from root to leaf, no variable appears more than once, and that along every path from root to leaf; the variables always appear in the same order. Next, apply the following two reduction rules as much as possible: (1) merge any duplicate (same label and same children) nodes, and (2) if both child pointers of a node point to the same child, delete the node because it is redundant. The resulting directed, acyclic graph is the BDD for the function.

![Fig 1 Creating the BDD for (X Exor Y Exor Z)](image)

There are four types of BDD algorithms for formal hardware verification using BDDs.
- Combinational equivalence algorithm
- Symbolic simulation algorithm
- Sequential equivalence algorithm
- Model checking algorithm

2.2 SAT Based Formal Verification

Symbolic model checking based on BDDs have come a long way since their introduction more than a decade ago. However, they are still incapable of handling the largest problems encountered in current industrial practice. Reduction in feature size coupled with the recent move towards IP-based design has led to dramatic increases in the size and complexity of systems that are being designed, thereby posing new challenges for functional verification methods. Hence there is a growing need to investigate and develop more robust and scalable verification methods based on novel and alternative technologies.

SAT stands for “Satisfiability”. Verification methods based on SAT solvers have recently emerged as a promising solution. Dramatic improvements in SAT solver technology over the past decade have led to the development of several powerful SAT solvers. Verification methods based on these solvers have been shown to push the envelope of functional verification in terms of both capacity and efficiency, as reported in several academic and industrial case studies. This has fueled further interest and intense research activity in the area of SAT based verification. The Boolean Satisfiability problem is a well known constraint satisfaction problem with wide applications in VLSI field. Given a propositional problem formula, the SAT problem posed on \( \phi \) is to determine whether there exists a variable assignment under which \( \phi \) evaluates to true. Such an assignment, if it exists, is called a satisfying assignment for \( \phi \) and \( \phi \) is called Satisfiability. Otherwise \( \phi \) is said to be unsatisfiable.

SAT also can be used in Model checking. There are three types of SAT solvers, which are currently used in present day formal verification techniques.
- CNF based SAT solvers
- Circuit based SAT solver
- Hybrid SAT solvers

2.3 Achievements in SAT based verification

There has been significant progress in the area of SAT-based verification over the last decade. However, much remains to be done to make this technology more pervasive in industrial design verification flows. The single most important achievement of SAT-based verification has been its emergence as an orthogonal technology to BDD-based model checking techniques (bounded and unbounded). This means that there are several instances where one technology significantly outperforms the other and vice versa. Further, SAT based techniques have been found to be less sensitive to the problem size and typically require much less user tuning of parameters. Hence, such methods are capable of verifying much larger systems than those typically handled by BDDs, and of enhancing productivity by obviating the need for user ingenuity and tuning effort. Bounded Model checking (BMC) based on SAT methods has been found to be particularly effective at generating counterexamples for hard to find bugs at short to medium depths (up to depth 50-60) of sequential behavior.

3. DESIGN FOR TESTABILITY

As more applications are added on a single chip with shrinking technologies, the design size and complexity continues to grow. However the time to design, manufacture and test these products is reducing due to market pressure. A significant time of design effort is spent to ensure the design is testable and acceptable fault coverage can be achieved. Design for Test (DFT) is a technique to check the structural faults.

The scan DFT methodology [5] is a standard DFT practice followed in industry for testing digital logic circuits using automatic test pattern generation.
The test procedure involves several steps such as scan test patterns for stuck-at and delay tests, logic and memory BIST patterns, differential IDDQ test patterns, burn-in tests, and several other tests to achieve acceptable fault coverage. With several DFT steps and tools from different vendors, test implementation becomes a challenge particularly if the DFT is to be implemented at gate level netlist with IO constraints. In order to deal with these issues a DFT methodology is proposed. In this methodology a test module is created in RTL which includes the test logic for selecting and activating appropriate DFT steps such as ATPG, boundary scan, BIST or selecting the testing of analog circuit in loop-back & other modes. The test module is synthesized and inserted in the design automatically with Tcl scripts. Equivalence check flow guarantees the equivalence check before and after the insertion of test module in the design to ensure the connectivity of the test logic.

Fig 2: DFT Flow [12]

3.1 DFT Implementation

3.1 Scan Insertion
Scan is a DFT technique that increases the overall testability of the design by inserting controllable and observable flip-flops in the design. During scan test mode the scan chain can serially shift in the controlled values and shift out the observed values.

3.2 Scan Signals
PAD_SE: The Scan enable signal PAD_SE is asserted during the internal scan chain shift operation and is deasserted during the capture operation of scan test vector. The scan enable should be directly controllable from a dedicated test pin.
PAD_TEST: The test mode signal is set to ‘1’ during the scan test mode. The test mode should be directly controllable from a dedicated test pin.

3.3 Common Scan Violations
*Clock controllability: All the system and test clocks should be controllable for test purposes. The test engineer should add test logic to control all internally generated or gated clocks in the design.

*Asynchronous Feedback loop: Circuits with asynchronous feedback loops should be avoided. Any asynchronous feedback loop in the design can cause generation of incorrect test vectors due to simulation inaccuracies. Designers should avoid feedback loops and ensure all logic is synchronous. Asynchronous pins can have protocol violations. This is typical with asynchronous clear and reset pins. An "OR" gate should be used to logically OR the asynchronous signal with the test mode signal. The test mode signal defines a static value breaking the feedback loop.

*Cross coupled gates: Standard library cells implemented as level sensitive latch should be used instead of cross coupled gates; otherwise it will reduce the fault coverage.

*Latches in design: Latches should be avoided. For designs with latches; if a test compiler cannot find a latch equivalent and treats it as a block box, latch should be made transparent. In this mode the output of the latch depends only on the input. However this may cause some reduction in fault coverage.

*Mixed Clocks and Multiple Clock edges: Individual clock domains should group the scan chains, to avoid any clock skew problems during scan shift operation. In case of mixed clock edges, the negative edge flip flop group should be first in the scan chain, followed by positive edge scan chain group. The STIL (Standard Test Interface Language) protocol file generated from Synopsys test compiler can be used in TetraMax tool from Synopsys to generate scan patterns. The STIL file defines the scan chain structure, clocks, and constrained ports, test setup.

3.4 Boundary Scan

Boundary scan helps in testing task of PCBs. The primary purpose of Boundary scan is to test for manufacturing defects. Boundary scan cells can capture data into the core logic in parallel mode or shift the data around the core logic in serial mode.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Default Value</th>
<th>FO</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTAG_JMS</td>
<td>-1</td>
<td>J dovock</td>
<td></td>
</tr>
<tr>
<td>JTAG_TCK</td>
<td>1</td>
<td>Testdata</td>
<td></td>
</tr>
<tr>
<td>JTAG_TEST</td>
<td>1</td>
<td>Testdata active low</td>
<td></td>
</tr>
<tr>
<td>JTAG_TDI</td>
<td>1</td>
<td>Testdata</td>
<td></td>
</tr>
<tr>
<td>JTAG_TDI ?</td>
<td>1</td>
<td>Testup</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Boundary Scan Signals

The boundary scan tap controller cells must have pull-ups on all input pins and tristates on the output data pins. Test engineer should give special consideration to the type of PAD cell used. The JTAG primary input / output pins should be instantiated in the design with their outputs unconnected. All test
mode (TM) and the test enable (TEN) should be wired together to global test mode and global test enable signals, which can be controlled from the top level dedicated input/output pins.

**BIST**

The BIST is used to test the embedded memory in the design. In BIST test, a series of test patterns are generated for each memory. The BIST block compares the output from each memory with the expected response to determine any test failure.

**IDDQ test**

IDDQ testing is the measurement of power supply current in the quiescent logic state of CMOS device. The IDDQ test observes the power supply current to verify whether the device under test is affected by manufacturing defects. Detecting these faults is done by test program stimulating the fault and tester electronics, measuring the increased current. With scaling technology it becomes difficult to measure the quiescent power supply so differential IDDQ or delta IQQD technique is used for current technologies. In this section we describe the IDDQ test procedure with Synopsys tool TetraMax and Synopsys is used to generate the test bench with test patterns for IDDQ testing. The “set faults -model iddq” command is used to change the default stuck-at model to IDDQ type fault model. In the IDDQ fault model, TetraMax attempts to sensitize each node in the design, but does not propagate faults to the device outputs. Besides specifying the iddq fault model, the “-bidi_fill” option should be used for set simulation command. This will enable TetraMax to drive ‘1’ or ‘0’ on bidirectional IOs in input mode. Verilog simulation should be run with the PLI routine call for IDDQ to generate the IDDQ database. This PLI routine task allows seeding fault into the design and selecting good qualified strobos. Power Fault tool verifies and selects a minimum set of vectors for maximum fault coverage from this IDDQ database.

The Power Fault ipro run generates the “frpt” and “sfrpt” reports. The frpt report provides details on the status of faults and the sfrpt report, indicates the IDDQ coverage achieved. Then we should analyze the leaky report file for floating inputs and drive contention. Certain leaky states can be ignored by using the “allow” command. The “disallow” command can be used to specify leaky state on certain IO cells with pull-ups. The STIL protocol file should be modified appropriately before running the TetraMax to generate test patterns. The primary pins can be initialized to define the initial value in the “test_setup” procedure. The initialization can include defining the clock in the off state, enabling any test pins, and specifying the initial value of IOs.

4. AUTOMATIC TEST PATTERN GENERATION

Upon completion of scan insertion in the design, the test patterns may be generated for the entire design using TetraMax. This is an independent ATPG tool, used solely for creating test patterns and to provide seamless interface to DC. During dynamic simulation the test patterns are used as input stimuli to the design to exercise all the scan paths. TetraMax will do DRC check on the scan inserted design using .spf file and finally writes test pattern and also does the simulation. This step will be performed at the full chip level and preferably after layout. The design flow is as shown in the figure ATPG flow in TetraMax starts with the reading the netlist and library models in the TetraMax followed by following stages.

**Build:** Building the ATPG design models takes place those parts of the design that are to be part of ATPG process, removes the hierarchy, and puts them into an immemory image that TetraMax can use. One can build the ATPG model for his design using the run build model command.

**DRC:** Test DRC involves analysis of many aspects of the design. Review the DRC report for error and violation. Correct all DRC violations that are errors ignoring or overlooking those might result in ATPG patterns that can not be simulated.

**Preparing for ATPG:** In this section we will initialize the fault list, select the pattern source, choose settings for bus contention checking and specify the pattern generation effort.

**Run ATPG:** Using the run ATPG dialog box, or by using Run command in the command line can perform this. By default, TetraMax performs BASIC-Scan ATPG first followed by fast sequential ATPG. To get good balance between pattern compaction and execution speed, we have to adjust the abort limit and merge effort settings of the set ATPG command.

Fig. 3 ATPG Flow In TeraMax
Review test coverage and return ATPG: one can view the results of the test coverage and the number of patterns generated using the report summary dialog box or can enter the report summaries command at the command line. Test coverage can be improved by changing the value for abort_limitand_merge using set ATPG command.

**Pattern Compression:** To achieve further pattern compression beyond the pattern merging done during ATPG, one can use the run pattern compression dialog box or can enter the run pattern compression command at the command line.

**Save pattern:** one can format and save the test pattern using the Write patterns dialog box or can enter the write patterns command at the command line. TetraMaX supports the following file formats Binary, PTDL, STIL, TDL91, Verilog, VHDL etc [2].

4.1 Brief of ATPG Process

ATPG process consist of two modes, they are:
- Shift mode
- Capture mode

In scan insertion done using multiplexed_flipflop

scan style this mode is controlled by scan enable input port. During the shift mode, input comes from the scan input port or output of the previous scan cell whereas during capture mode input to each scan element comes from the combinational logic block. So it is very important to control the value of scan enable during the ATPG process.

**Spf and its Importance:**

SPF stands for STIL Procedure file. STIL is an emerging standard for simplifying the number of test vector formats that ATE vendors and CAE tool vendors must support. SPF is an output from test Compiler or it can be manually created and carries the information needed by TetraMaX to do DRC check. This check includes
- Whether scan chain inputs and outputs are logically connected
- Whether all the clocks and asynchronous set/reset pins connected to scan chain flip flops are controlled only by primary input ports.
- Whether the clock/set/reset are off when you switch from normal mode to scan shift mode and again when you switch back to normal mode.
- Whether any internal multiple-driver nets can be in contention.

5. PROJECT OBJECTIVE

The specific objectives of the research were
- Understand why DFT is important.
- Understand the major principles behind scan based testing and the generation of test vectors.
- To sensitize and propagate faults to scanned outputs.
- To achieve high fault coverage (% of potential faults that are detectable)
- Minimize test application time.
- Maximize fault resolution (isolating fault to smallest replaceable component).
- To provide controllability & observability.

5.2 Methodology:

The Methodology adopted was
- Functional verification was done for 64x64 image input using Modelsim simulator.
- Formality was done to verify that optimization or logic synthesis was done correctly by comparing the circuit before and after i.e. Equivalence checking.
- In DFT, Scan based testing is done. In this technique, the flip-flops are connected serially to read in test vectors and results were examined.
- The most commonly used fault models are stuck-at-0, stuck-at-1, transition delay and path delay fault models
- Boundary scan to test the interconnect between chips.
- As manual test pattern generation is a tedious process, ATPG is done using TetraMAX.

**Tools used:** Various tools are used in different stages for verifying the DA based DWT IP Core
- RTL Coding & Simulation using Model Sim$SE from Mentor Graphics.
- Logic synthesis using Design compiler from Synopsys.
- Formal Verification using Formality from Synopsys.
- DFT compiler from Synopsys is used for scan insertion, testing etc.
- ATPG are generated using TetraMax.

6. VALIDATION AND DISCUSSION OF RESULTS

Functional verification has been done by writing test bench in VHDL. In this we will give specific input sequences to test certain functionality. The outputs are compared with the expected outputs.
6.1 Matching Results

18 Compare points matched by name
   • 0 Compare points matched by signature analysis
   • 0 Compare points matched by topology
   • 11 Matched primary inputs, black-box outputs
   • 0(0) Unmatched reference compare points
   • 0(0) Unmatched reference primary inputs, black-box outputs

6.2 Verification Results

Verification SUCCEEDED

Reference design: r:/WORK/top

Implementation design: r:/WORK/top

18 Passing compare points
Matched Compare Points
BBPin Loop BBNet Cut Port DFF LAT TOTAL

---

Passing (equivalent) 0 0 0 0 1 8 0 0 18
Failing (not equivalent) 0 0 0 0 0 0 0 0
Not Compared
Constant reg 924 0 924

6.3 DFT of DA based DWT IP core:

In the next stage of verification DFT has been undertaken for the design under test. Overall process of insertion of test logic is automated and simplified by using simple tcl script. This process of creation and insertion of test structure can be reused. The inputs and outputs of this task are as follows...

Inputs:
1. Synthesized netlist from the Design Compiler.
2. Library files such as database files .db symbol library .sdb

Outputs:
1. Spf file of STIL format will be created which comprises of info about signals, scan structures used and timing

Reports: Various reports such as Fault summary, power, area and timing has been evaluated.

Uncollapsed Stuck Fault Summary Report:

Fault class code #faults

| Detected DT 34536 | 1 |
| Possibly detected PT 0 | 0 |
| Undetectable UD 34 | 0 |
| ATPG untestable AU 71 | 0 |
| Not detected ND 7 | 0 |

| Total faults 34648 |
| Test coverage 99.77% |

Area Report:

| Number of ports: 34 |
| Number of nets: 160 |
| Number of cells: 23 |
| Number of references: 15 |
| Combinational area: 5639.750000 |
| Noncombinational area: 6330.000000 |
| Net Interconnect area: 1710.957275 |
| Total cell area: 11969.750000 |
| Total area: 13680.707031 |

Report: power-analysis effort low:

Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW(derived from V,C,T units)
Leakage Power Units = 1 pW
Cell Internal Power = 2.6127 mW (94%)
Net Switching Power = 180.2094 uW (6%)
Total Dynamic Power = 2.7929 mW (100%)
Cell Leakage Power = 69.9701 uW

Timing Report:
data required time 55.02
data arrival time -22.60

Slack (MET) 32.42

Comparing the results obtained by synthesizing using DC with that of Design for Testability we observed the following:
1. Cell leakage power get reduces.
2. Total cell area also get reduced

6.4 ATPG using TetraMax:

As manual generation of test pattern is a tedious process we will go for ATPG by using TetraMax.

Inputs:
1. .Spf files from test compiler.
2. Library files such as class.db, class.sdb are used.

Outputs:
The following reports have been generated:

Transition delay fault model fault coverage result
Uncollapsed Transition Fault Summary Report

// fault class code #faults
// -------------------------------
// Detected DT 6031
// Possibly detected PT 26
// Undetectable UD 3531
// ATPG untestable AU 4396
// Not detected ND 8
// --------------------------------
// total faults 13992
// test coverage 57.78%
// -------------------------------

/// Pattern Summary Report
//------------------------------
// #internal patterns 77
// #basic_scan patterns 77
//------------------------------

Stuck-at fault model fault coverage result
Uncollapsed Stuck Fault Summary Report

// fault class code #faults
// ---------------
// Detected DT 10610
// Possibly detected PT 80
// Undetectable UD 3026
// ATPG untestable AU 270
// Not detected ND 6
// --------------------------------
// total faults 13992
// test coverage 97.12%
// -------------------------------

/// Pattern Summary Report
//-----------------------------
// #internal patterns 168
// #basic_scan patterns 168
//-----------------------------

7. CONCLUSIONS

Based on the methods and analyses presented in the previous sections, the following conclusions are drawn:

1. With the functional testing, we applied a sequence of input data and resulting output is compared against expected behaviour of the device. And observed that this type of testing has limited ability to test the integrity of the device’s internal nodes.

2. Among the various existing scan design techniques internal scan design is the most popular DFT technique and has the greatest potential for high fault coverage.

3. This internal scan design technique divided the complex sequential designs into fully isolated combinational blocks or semi isolated combinational blocks.

4. Internal scan design modifies existing sequential elements in the design to support a serial shift capability in addition to their normal functions.

5. By adding scan circuitry to the design increased design size and power, as the scan cells are larger than the non-scan cells they replace, and the nets used for the scan signals occupied additional area.

6. Because of scan insertion design performance decreased marginally as the electrical characteristics of scan cells are changed.

7. By using ATPG techniques large number of internal faults are tested by setting all nodes
of the circuit to both 0 and 1 and then propagating the defects to nodes.
8. TetraMAX reads the design netlists in Verilog, Vhdl and EDIF formats and test protocol information in STIL format.
9. Test patterns are written in STIL format.

REFERENCES

[1]. "Formal Hardware verification with BDDs: An introduction" by Alan J.Ho, Dept of computer science, University of British Columbia.
[2]. "A survey of Recent advances in SAT based formal verification" by Mukul R.prasad – Fujitsu labs of America; USA, Armin Biere – JohannesKepler University; Austria, and Aarthi guptha; Nclabs,USA.
[3]. Essentials of Electronic testing for digital memory and mixed signal VLSI circuits, Vishwani D. Agarwal, Michel L.Bushnell
[4].“Art of ATPG generation from TetraMax” by Resty Roy, VLSI design engineer, Wipro Technologies, Bangalore, India.
[7]. Corporate Test Logic(CTT), DFT Seminar, Infineon Technologies
[8]. Database available with in Solvnet.
[9]. "Finding Manufacturing Processing Defects With ATPG", by Taher Abbasi &Ricky Bedi

Websites:
www.eetimes.com
www.lateralsands.com