RTL Based Scan BIST

Subrata Roy
Lucent Technologies, Bell Laboratories
P.O. Box 900, Princeton, NJ
email: subrataroy@lucent.com

Abstract

Synthesis of ASICs from Register Transfer Level(RTL) source is often a bottom up iterative process where synthesis process is carefully controlled to produce a gate-level design which meets the desired constraints. Test logic, such as Built-in Self Test(BIST), is typically inserted at the gate level. This may cause new violations of timing/area goals thus causing an expensive cycle of re-optimization on the complete chip at the end of the design process. Performing BIST logic insertion at the RT level source allows synthesis technology to consider test logic while optimizing to meet area/timing goals thus avoiding an expensive re-optimization. It also provides opportunity for sharing of functional and test logic. Scan Based BIST utilizes scan chains to apply random vectors and observe signal values within the random logic. This paper will describe techniques for inserting scan chains at the RTL-VHDL source in the context of Scan BIST and report its impact on design optimization and fault coverage.

1: Introduction

Testing complex ASICs, to ensure high quality of shipped parts, may require large number of test vectors. These test vectors may be very expensive to generate and may not achieve desired stuck-at fault coverage. Design For Testability techniques[1] are used to provide high quality test at a substantially reduced test development cost. Scan techniques[2], which connect some or all flip-flops in shift registers allow automatic test generation tools to generate high quality test in a reasonable amount of time. BIST can eliminate the need for test generation and allow at-speed testing by embedding test generation and signature computation hardware within the design.

Scan based BIST[10] utilizes scan chains to apply random vectors and observe signal values within the random logic. In addition, BIST requires a controller surrounding the random logic and test points to address random pattern resistant faults. The controller for BIST is available as RTL VHDL source, which can be instantiated within the VHDL design. However, the scan chains and test points within the random logic are inserted at the gate level. An internal scan chain causes an extra multiplexor delay and an extra loading on the flip-flop output. In a RTL synthesis based ASIC design methodology this can cause an expensive cycle of re-optimization. Inserting scan chains at the RTL source allows the designer to utilize the full power of synthesis to optimize the test and the functional logic. This is particularly useful in BIST, since the test logic needs to operate at the same speed as the functional logic. Since test logic and functional logic are optimized together, the final design is better optimized than the method of gate-level scan insertion followed by a re-optimization phase. The quality of the design (in terms of area/timing goals) is further enhanced if the scan memory elements are ordered to take advantage of the functional relationships identifiable in the RTL source.

This paper will describe a technique of inserting scan chains in the RTL-VHDL[1] source. Test points needed to handle random pattern resistant faults, may still need to be inserted at the gate level and is not considered in this paper. The goals of such a technique are:

1. Integration with bottom up iterative synthesis process,
2. Simple and identifiable modification to the VHDL source,
3. No loss of fault coverage as evaluated at the gate level, and
4. Better at meeting design constraints relative to testability insertion at the gate level followed by re-optimization.

Various other approaches have been proposed for adding BIST at RTL[7][8][9]. In contrast to these approaches scan based BIST allows the scan chains to be used for applying diagnostic or supplementary deterministic vectors and is applicable to a large variety of designs.

2: Scan Insertion at RTL

The technique consists of two basic steps:

Proceedings of the 1997 VHDL International User's Forum (VIUF '97)
0-8186-8180-2/97 $10.00 © 1997 IEEE

1. Scan Analysis: In this phase, VHDL variables and signals to be converted to flip-flops are identified and partitioned into a set of lists representing the scan chains.
2. Scan Insertion: The ordered list of variable/signals are used to insert scan assignments in the appropriate sequential process in VHDL.

Selection of scan chains is best done on the complete design. In BIST environment, as opposed to scan test methodology, the number of scan chains are not limited by input/output pins. Hence it is desirable to have larger number of scan chains of shorter length. Balancing of such scan chains is best done, if scan chain selection is based on the complete design. Although synthesis is done separately on each sub-circuit, the complete RTL VHDL is available for RTL simulation before any significant synthesis work is done. We use this complete RTL-VHDL for scan analysis in a process shown in Figure 1. The result of scan analysis is a set of scan chains associated with each synthesizable module. Scan insertion is done separately for each module based on the results of the scan analysis. The following

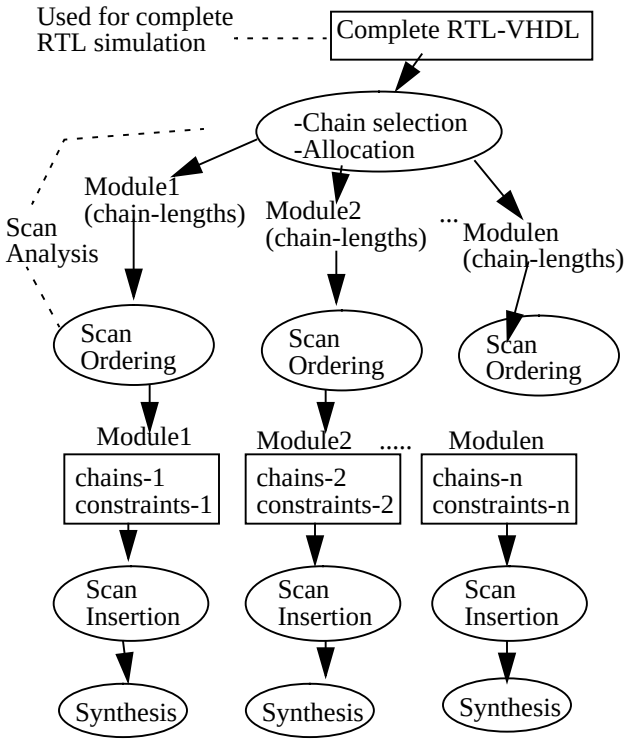

Figure 1. Process for Scan insertion at RTL

sections describe the two basic steps in the context of the process flow in Figure 1.

2.1: Scan Analysis

A complete RTL-VHDL description of a circuit consists of the hierarchy of VHDL entities. A VHDL architecture, describing the behavior of an entity, consists of several concurrent statements. We will assume that an architecture consists of two kinds of concurrent statements, process statements and component instantiation statements. A process statement is the most general concurrent statement describing the time behavior of signals using RTL operators and synchronization statements. A component instantiation statement is used to call another entity. Any other concurrent statement can be modeled in terms of a process statement. The lowest level entities will only contain process statements or equivalents.

Each process statement needs to be analyzed to identify the flip-flops. A process statement contains two types of data carriers called variables and signals. Variables are local to a process and are typically used for intermediate computation. Signals are visible within an entity and are used for communicating across processes. A set of general rules identifying the data carriers that infer flip-flops in an RTL-VHDL description have been developed in ([3] pp-85). These rules can be further simplified depending on the synthesis tool. The process statement is analyzed to construct ff_list(P1), which is the list of data carriers that infer flip-flops within the process P1. The following algorithm is used to construct this list for process P1.

Analyze_process(P1):

1. locate the clocking statement within the process.
2. For any signal assignment statement following the clock event, add the target signal to ff_list(P1)
3. For any variable assignment following the clock event, such that the target variable is read before being the target of an assignment, add the target variable to ff_list(P1).

Some examples of clocking statements are shown in Figure 2. Depending on the synthesis tool limited number of

```
if(clk ='1'and clk'event)
then
  X <= X - Y;
end if;

wait until clk = '1';
X <= X - Y;
....
```

Clocking Statement

Figure 2. Examples of clocking statement

variation of clocking statements are allowed. For combinational processes, there are no clocking statements

and hence ff_list(P1) is empty. The following describes the individual steps in the scan analysis phase.

2.1.1: Chain Selection

This step selects the scan chains for the complete circuit. The signal/variable name *svname* in the list ff_list(P1) can be uniquely identified relative to the complete circuit by the triplet (<entity-path>, P1, *svname*), where <entity-path> is the hierarchical path to the instance described by the entity containing the process P1. By analyzing all the processes in the entity hierarchy, we can construct the complete list of potential flip-flops. Based on user specified parameter of maximum chain length and/or no. of chains. we partition this list into several chains of equal size.

2.1.2: Allocation

From the chains for the complete circuit, we allocate the part of the chain to its respective modules that are synthesized together. Since the chains will get re-ordered in the next step, each module keeps a list of chain sizes {l1, l2, ... ln}.

2.1.3: Scan ordering

New chains are constructed based on the list of chain sizes {l1, l2 . ln} for the specific module. The data carriers in a scan chain are ordered based on functional relationship. This allows the synthesis to share functional logic with test logic during design optimization. [4] demonstrates how a functional data transfer of n-bit value between 2 n-bit registers can be used in scan mode by modifying the control logic. [5] identifies “beneficial” logical relationship among scan variables which allow implementation of scan with less than multiplexor overhead. Both techniques are relevant, since in practice, RTL VHDL uses both RTL operators and logical expressions. The functional relationship among data carriers are analyzed in terms of a signal-variable graph (SV-graph) which consists of edges of the form (n1, n2) where n1, n2 are signals or variables in a process. For each type of statement within a process there is a rule for constructing an edge in the SV-graph as shown in Figure 3. Edges corresponding to a direct assignment are

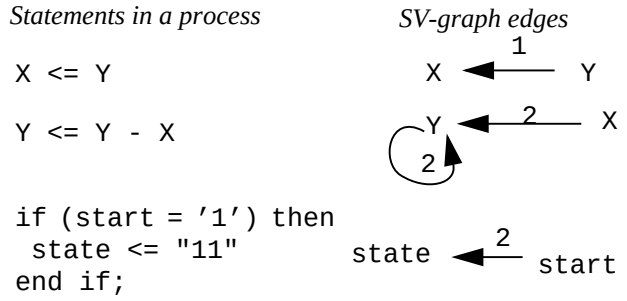

Figure 3. Constructing SV-graph

labeled priority 1. All other edges are labeled priority 2. SV-graphs are constructed for each process and then merged to form an SV-graph for an hierarchy of entities as shown in Figure 4. Any node (IN) that is not a potential flip-flop or

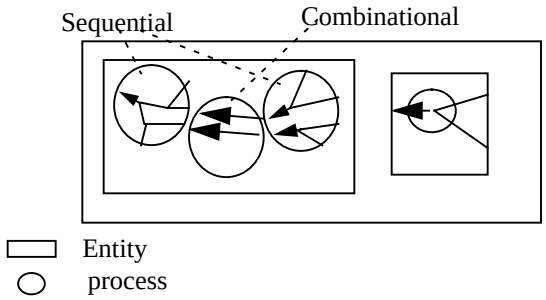

Figure 4. Merging of SV-graphs

an input/output port of the top level entity in the module is removed by replacing edges of the form (x, IN) and (IN, y) with an edge (x, y).

The SV-graph for the whole entity is used to order flip-flops to construct chains to meet sizes specified in the list {l1, l2, .. ln} for this entity. We choose edges on the SV-graph in order of their priority and construct chains in a step by step fashion so that we have the chains of the right sizes.

2.2: Scan Insertion Phase:

Scan insertion is done by adding VHDL assignment statements within each process. The objective is to insert scan with minimal modifications to the VHDL source. Let d1, d2 be the adjacent data carriers in a scan chain. The scan direction is from d1 to d2. We have to consider the following two cases.

2.2.1: d1 is a Signal

An assignment of the form “d2 <= d1;” or “d2 := d1;” is added to the process P1 depending on whether d2 is a Signal or a variable. For Signal d2, the process P1 is chosen such that d2 belongs to ff_list(P1). The choice of the process ensures that a signal is not driven by two different processes. For variable d2, P1 must be the process in which the variable is defined. These assignments are conditional on the scan mode signal and added as a last statement in the process. The scan assignments ‘‘v1 := s1’’ and ‘‘s3 <= s2’’ illustrate this in Figure 5. Note that the functional statements are not modified in any way. The semantics of a signal/variable assignment ensures that the last assignment is effective. Also d1 being a signal, the value of d1 used in the scan assignment is the value that existed prior to entering this process.

```
P1: process(clk)
  variable v1, tmp_v1 : std_logic;
begin
  tmp_v1 := v1;          <- previous value of v1
  if(clk = '1' and clk'event)
  then
   s1 <= v1;
   s2 <= x + y;          <- functional statements
   v1 := s2 - 1;
   .....
   if(scan_mode = '1')
   then
    v1 := s1
    s2 <= tmp_v1;        <- scan assignments (s1, v1, s2, s3)
    s3 <= s2;
    ...
   end if;
  end if;
end process;
```

Figure 5. Inserting Scan assignments

2.2.2: d1 is a Variable

The previous solution doesn't quite work when d1 is a variable since intermediate assignments to d1 in the functional or scan statements will affect the value of d1 used in the scan assignment "d2 := d1;", i.e d2 will not take the value of d1 in the previous clock cycle. To solve this we have to declare a temporary variable, say tmp_d1 and add an assignment "tmp_d1 := d1" at the beginning of the process and use "d2 := tmp_d1" (or ''d2 <= tmp_d1'')as the scan assignment at the end of the process construct. The scan assignment ''s2 <= tmp_v1'' illustrates this in Figure 5.

2.2.3: Scan across Entity Hierarchy

The input and output ports of an entity and component instantiation needs to be changed to connect scan data across the entity hierarchy. Care needs to be taken to make use of existing ports of Entities as much as possible.

3: BIST Controller

A random logic BIST controller in RTL VHDL is instantiated adjacent to scan inserted core logic as shown in Figure 6. The controller is synthesized along with RTL

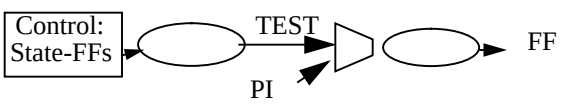

Figure 7. New BIST Path

random logic. Care must be exercised in setting up the

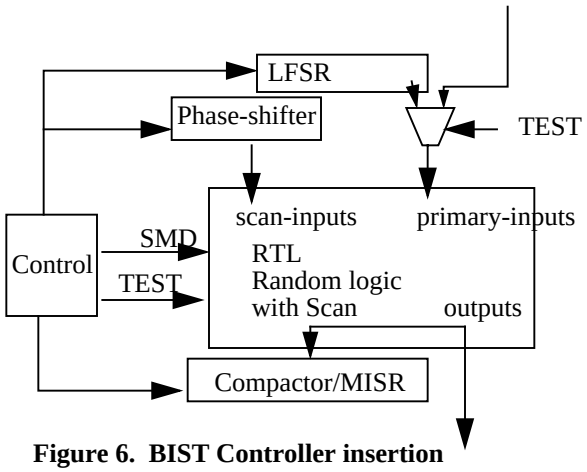

Figure 6. BIST Controller insertion

constraints for the primary inputs, since now these inputs get random signals through a multiplexor under the test mode. This new path shown in Figure 7, may impose tighter timing constraints on the primary input PI. The impact of this may not be significant if the primary inputs are registered before any significant combinational processing.

4: Implementation

We are using a VHDL front end software LVS, which provides a VHDL parser and a procedural interface to VHDL syntax tree[6]. This system is used to identify clocks, signals and variables that infer flip-flops and the SV-graph for each process.

5: Experimental Results

The RTL source was modified for full scan and BIST and the resultant circuit, after synthesis, was evaluated in terms of fault coverage and area overhead. This was compared with the results of gate level full scan and BIST circuit.

The following four gate-level circuits were obtained from each RTL design.

1. **RTL SCAN:** This circuit was obtained by synthesizing after inserting scan assignments at the RTL source.
2. **RTL BIST**: BIST controller and scan assignments were added to the RTL source and then synthesized together to produce this circuit.
3. **GATE SCAN**: Gate level full scan tool was run on gate level circuit synthesized from the RTL code. The output circuit from the full scan tool, was re-optimized to meet timing.
4. **GATE BIST**: Gate level BIST tool[10] was run on the circuit synthesized from the RTL code for GCD. The output circuit was re-optimized to meet timing.

These circuits are compared in terms of fault coverage and area in Table 1 and Table 2 for two example RTL designs. They were all synthesized for the Lucent/5cmos technology. Fault Coverage (F.C.) is the percentage of detected faults. Test Efficiency is the percentage of detected and untestable faults as determined by the ATPG tool.

Table 1:

gcd: 50 FFs area=3653 clock=22ns	RTL SCAN	RTL BIST	GATE SCAN	GATE BIST
area over-head	-7.6%	17.2%	14.3%	71%
F.C/T.E %	98.5/99.8	82.7	99.05	80.2

The area overhead for circuits with BIST excludes the

Table 2:

diffeq: 78 FFs area=7616 clock=30ns	RTL SCAN	RTL BIST	GATE SCAN	GATE BIST
area overhead	2.6%	20.4%	20.8%	36.4%
F.C/T.E %	98.4/99.9	93.4	96.7/99.9	95.8

controller since that part is fairly independent of circuit size. The results demonstrate that scan/bist insertion at RTL results in lower overhead with no loss of testability. The extra overhead includes additional logic required to meet timing. The overhead impact may have been enhanced due to choice of tight timing for the original design. The negative overhead for RTL_SCAN on gcd is the result of different technology mapping under which more of the functionality was mapped to a single standard cell. The significantly high overhead of GATE BIST on gcd is due to the new BIST path shown in Figure 7, which is optimized a lot better when starting from an RTL design than from a gate level design. The low fault coverage for BIST on gcd is due to random pattern resistance. With gate level test points, BIST on gcd produced about 96% fault coverage.

6: Conclusions

We have described a technique of inserting scan chains at the RTL source with minimal modification to the source.The advantages of inserting scan based testability at RTL are:

1. Allows full power of the synthesis process to be utilized for optimizing test logic as well as the functional logic.
2. Area overhead for the same timing constraints are substantially less if scan is inserted at the RTL
3. No loss of fault coverage based on fault simulation at gate level.

More experimentation with larger circuits are required to study the impact of:

1. initial timing margin on the overhead benefit shown in Table 1.
2. different scan chain ordering algorithm on the overhead benefit.

References

[1] M. Abramovici, M.A. Breuer, and A.D. Friedman, “Digital System Testing and Testable Design”, IEEE press, New York 1990

[2] K.T.Cheng and V.D.Agarwal, “A Partial Scan Method for Sequential with Feedbacks”, *IEEE Transactions on Computers*, Vol 39, No 4, April 1990, pp 544-548

[3] Ronald Airiau, Jean-Michel Berge and Vincent Olive, “Circuit Synthesis with VHDL”, Kluwer Academic Publishers, 1994

[4] S. Bhattacharya and S. Dey, H-SCAN: “A High-Level Alternative to Full-Scan Testing with Reduced Area and test Application Overhead”, *14th IEEE VLSI Test Symposium*, April 1996, Princeton, NJ

[5] R.B. Norwood and E.J. McCluskey, “Synthesis-for-scan and Scan Chain Ordering”, *14th IEEE VLSI Test Symposium*, April 1996, Princeton, NJ

[6] LEDA VHDL System, User’s Manual Version 4.0.0, LEDA S.A., Meylan, France

[7] K.D. Wagner and S. Dey, “High-Level Synthesis for Testability: A Survey and Perspective”, in *Proceedings of Design Automation Conference*, pp 131-136, 1996

[8] C. Papachristou and J. Carletta, “Test Synthesis in the Behavioral Domain”, in *Proceedings of the International Test Conference*, 1995

[9] L. Goodby and A. Orailoglu, “Towards 100% testable FIR digital filters”, in *Proceedings of the International Test Conference*, 1995

[10] C.J.Lin, Y.Zorian and S.Bhawmik, “Integration of Partial Scan and Built-In Self-Test”, *Journal of Electronic Testing*, Vol-7, No. 1/2, August/October 1995