Fault Simulation of Digital Circuits at Register Transfer Level

Suma M.S
R.V.College of Engineering
Department of E & C
Bangalore-59

K.S.Gurumurthy
U.V.College of Engineering
Department of E & C
Bangalore-01

ABSTRACT
As the complexity of Very Large Scale Integration (VLSI) is growing, testing becomes tedious and tougher. As of now fault models are used to test digital circuits at the gate level or below that level. By using fault models at the lower levels, testing becomes cumbersome and will lead to delays in the design cycle. In addition, developments in deep submicron technology provide an opening to new defects. We must develop efficient fault detection and location methods in order to reduce manufacturing costs and time to market. Thus there is a need to look for a new approach of testing the circuits at higher levels to speed up the design cycle. This paper proposes on Register Transfer Level (RTL) modeling for digital circuits and computing the fault coverage. The result obtained through this work establishes that the fault coverage with the RTL fault model is comparable to the gate level fault coverage.

General Terms
Fault Model, Test Pattern Generation.

Keywords
Automatic test pattern generation (ATPG), fault coverage, fault simulation, stuck-at fault, RTL.

1. INTRODUCTION
VLSI industry is growing as per Moore's law and integrated circuit designs are accordingly becoming more and more complex. As a result of this, VLSI testing has become expensive in terms of cost. Existing gate level fault simulation techniques exhibit poor performance standards when applied to such designs and are unsuitable for early testability analysis or fault simulations. Current computer-aided design tools must address the needs for a new generation of integrated circuits such as systems on chip. Test sequences consist of many thousands of test patterns, which make gate-level fault simulation inappropriate due to its lengthy computational time. Also test generation and fault simulation efforts in the post synthesis phase do not contribute to the improvement in the design. Therefore, we need ATPG tools that reflect new design flows, especially tools that work at a higher level of abstraction than gate-level. Many high-level fault models and fault simulation techniques have been proposed. No single fault model is universally acceptable since no fault model has been developed so far that comprehensively covers all classes of circuits. The RTL description is at a higher level of abstraction and may not cover all the gate level faults [2].To be widely accepted an RT-level fault simulator must accept input formats in standard hardware description languages such as VHDL or Verilog and should show high efficiency. Fault simulation plays a key role in ATPG systems; including high-level ATPG. Gate-level fault simulation is not appropriate for large systems because of long runtimes or large memory requirements. RT-level fault simulation may be the only alternative for estimating the quality of tests generated using high-level ATPG [3]. A high-level fault model should guarantee fault coverage comparable to the gate-level fault coverage obtained for the same test sequence.

The fault model proposed by F.Corno, G.Cumani, M.Sonza Reorda and G.Squillero [2] adopts a particular instantiation of the observability enhanced statement coverage metric in addition to the single stuck-at bit faults on all assignments targets of the executed statements. The model implies observability enhanced statement coverage by modeling one of the possible fault classes on executed statements. This is an incomplete modeling of the various faults associated with the RTL description of the circuit.

The fault model by Barry W. Johnson is developed via abstraction of industry standard single-stuck-line (SSL) faults into the behavioral domain. A functional analysis technique was used to evaluate the effects of the SSL faults on gate-level implementation. Since the gate-level netlist changes drastically during logic synthesis, the authors in [4] concluded that modeling all possible gate-level faults at the RTL is highly inefficient.

The RTL fault model and simulation approach proposed by Mao and Gulati [5] uses the single stuck-at fault for each bit of all variables in the RTL model. The model employs both the RTL description and functional verification patterns. But their approach required one to run fault simulation twice, first in an optimistic mode and then in the pessimistic mode and to use the average of the results to reduce the difference between the RTL and the gate-level fault coverage. The experimental data shows as much as 10 % error between the actual gate-level fault coverage and the RTL fault coverage.

Another fault model proposed by Devadas and Ghosh [6] is the Observability Enhanced Statement Coverage Metric. This model requires that all statements in the RTL description are executed at least once and that their effects are propagated to at least one primary output. As this approach can be fruitfully exploited for the test pattern for fault simulation, more accurate results are needed.

The fault model proposed by Karunaratne et al. [7] does not consider stuck-at faults in the signal bit values and also not

account for these faults. Also the process of locating the RTL faults and mapping them to the corresponding Gate-Level faults is to be done. It is therefore desirable to develop the fault model at a higher level of abstraction than the gate level. Fault simulation and testing at the higher levels of abstraction have a better chance of being integrated well into the overall design process.

Jose M.Fernandes et al. [8] has proposed a new probabilistic method for controllability evaluation based on a traitorously selection of registers to form groups. This work needs further optimization by computing the probabilistic impact of the simultaneous correction of different testability problems.

Digital circuits are commonly designed at multiple levels of abstraction, including the layout, transistor, gate, register-transfer (RTL) and behavioral levels. Designers describe circuits in a hierarchical, top-down fashion, typically using computer-aided design (CAD) tools. To simplify the design process, designers try to model circuits at a fairly abstract level. Conventional gate-level implementation is hard to understand, i.e., poor readability. By modeling circuits at a higher level, the number of primitive elements in a circuit is reduced, thus making the problem size more tractable. This allows larger circuits to be handled in less time. The authors [9] conclude that over 1000 times reduction in test-generation time is achievable by performing automatic test pattern generation (ATPG) at the RTL without any compromise in fault coverage.

The Unit II of the paper deals with the methodology, Unit III deals with the fault model and simulation, Unit IV with results and finally Unit V with conclusion.

2. METHODOLOGY

In this work Verilog Hardware Description Language is used for writing the RTL models. Although extensive work has been done on Verilog based simulation and synthesis, test generation and other test related issues are still to explore the capabilities of Verilog. The basic assumption is that the components are fault free and only their interconnections are affected. These map to the operators and variables in the RTL descriptions respectively. Gate level primitives can be instantiated in a model using gate instantiation as these are supported for synthesis. These primitive gates describe the hardware. Therefore synthesizing a gate primitive generates logic based on the gate behavior which eventually gets mapped to the target technology [1].Based on this the single stuck-at fault is modeled. The assumption is also that at most one fault occurs at a time in the circuit.

The proposed fault model is an improvement over the model given by Karunaratne et al. [6].Stuck-at faults in the signal bit values was not considered and accounted. Also the process of locating the RTL faults and mapping them to the corresponding Gate-Level faults was not implemented.

The analysis flow for the modeling approach is of two ways as shown in Figure 2. One way targets on the gate-level fault coverage while the other is on the RTL fault coverage. In the RTL path, the RTL design description is obtained based on the specification. Since the fault model is at the RTL, the fault is induced at the input and at the output. This is done by using a buffer for each bit in all of the variables in the RTL code. These buffers are inserted in the fault free circuit and should not disturb the functionality of the circuit. As a result, a modified faulty RTL circuit is obtained. To enable fault simulation the process of generating faulty circuits by inducing faults into the fault-free circuit is done. For each of the faults a new circuit is created.

Testbench is developed and the simulation is first run on a good circuit and then on each of the faulty circuits using the commercial simulator. The outputs obtained in each case of the faulty circuits are compared with the output of the good circuit to determine which faults are detected. That is the new faulty circuit and the fault free circuit is simulated and the outputs so obtained are compared. The fault list is tabulated. The ratio of the numbers of RTL faults detected to the total number of RTL faults gives the RTL fault coverage. At the gate-level, for each RTL description, gate level netlists are obtained for 65 nanometer target technology using logic synthesis tool and fault coverage obtained by Tetramax tool. The fault list of both the RTL as well as Gate-level faults is compared. The effectiveness of our fault model is determined by comparing RTL fault coverage with the fault coverage obtained at the gate level.

3. FAULT MODEL AND SIMULATION

Test generation plays an important role in the area of digital design. Test generation is a process of finding input test patterns for detecting possible faults in the circuit. It is difficult to generate test for real defects due to the diversity of VLSI defects. For generating and evaluating a set of test patterns, fault models are needed. Widely a good fault model should almost give a true nature of the behavior of defects and it should also computationally work well in terms of fault simulation and test pattern generation. It is necessary to propose a fault model, that is a fault model for how faults occur and their impact on circuits and to do with the business of good and bad parts, many fault models have been proposed [4], but unfortunately, no single fault model accurately reflects the behaviour of all possible defects that can occur. As a result, a combination of different fault models at many instances are used in the generation and evaluation of test vectors and testing approaches developed for VLSI devices [2]. Developing a test for faults at higher level of abstraction and then determining the percentage of faults at the lower levels being covered is a good strategy. Fault models at higher levels result in significant savings in test cost and test time required for deriving tests.

The most common model used for logical fault is the single stuck-at fault (SSF). In this a fault in a logic gate gives a favorable outcome in one of its inputs or the output being fixed to either a logic 0(stuck-at-0) or a logic 1(stuck-at-1).

For our approach divider is taken as an example.

```
module divider #(parameter n = 7)(
output reg [n:0] remainder,
output reg [0:n] quotient,
input      [n:0] dividend, divisor);
reg [2*n+1:0] accumulator;
integer i;

always @ (dividend or divisor)
begin
      accumulator =
{{(n+1){1'b0}},dividend};
             if(divisor == {(n+1){1'b0}})
             begin
```

```
                    quotient  =
{n+1{1'bX}};
                    remainder =
{n+1{1'bX}};
             end
             else  if(dividend ==
{n+1{1'b0}})
             begin
                    quotient  =
{n+1{1'b0}};
                    remainder = divisor;
             end
             else
             begin
                for (i=0; i<(n+1) ; i = i
+1)
              begin
              if (accumulator[2*n+1:n] <
{1'b0,divisor}) quotient[i] = 1'b0;
              else
              begin
                   quotient[i] = 1'b1;
                   accumulator[2*n+1:n] =
accumulator[2*n+1:n] - {1'b0,divisor};
             end
                accumulator[2*n+1:0] =
{accumulator[2*n:0],1'b0};
            end
                    remainder =
accumulator[2*n+1:n+1];
      end
end
endmodule
```

The above RTL design description is a fault freemodule. Faulty module is created such that the functionality will remain same as the fault free module. This is done by inserting the buffer for each of the ports. The faulty module appears as shown below.

```
module divider #(parameter n = 7)(
output  [n:0] remainder,
output  [0:n] quotient,
input      [n:0] dividend, divisor);
reg [2*n+1:0] accumulator;
reg[0:n] quotient_fault;
reg[n:0] remainder_fault;
integer i;
wire [n:0] dividend_fault,divisor_fault;
buf D7(divisor_fault[7],divisor[7]);
buf D6(divisor_fault[6],divisor[6]);
buf D5(divisor_fault[5],divisor[5]);
buf D4(divisor_fault[4],divisor[4]);
buf D3(divisor_fault[3],divisor[3]);
buf D2(divisor_fault[2],divisor[2]);
buf D1(divisor_fault[1],divisor[1]);
buf D0(divisor_fault[0],divisor[0]);
buf DR7(dividend_fault[7],dividend[7]);
buf DR6(dividend_fault[6],dividend[6]);
buf DR5(dividend_fault[5],dividend[5]);
buf DR4(dividend_fault[4],dividend[4]);
buf DR3(dividend_fault[3],dividend[3]);
buf DR2(dividend_fault[2],dividend[2]);
buf DR1(dividend_fault[1],dividend[1]);
buf DR0(dividend_fault[0],dividend[0]);
buf Q0(quotient[0],quotient_fault[0]);
buf Q1(quotient[1],quotient_fault[1]);
buf Q2(quotient[2],quotient_fault[2]);
buf Q3(quotient[3],quotient_fault[3]);
buf Q4(quotient[4],quotient_fault[4]);
buf Q5(quotient[5],quotient_fault[5]);
buf Q6(quotient[6],quotient_fault[6]);
buf Q7(quotient[7],quotient_fault[7]);
buf R0(remainder[0],remainder_fault[0]);
buf R1(remainder[1],remainder_fault[1]);
buf R2(remainder[2],remainder_fault[2]);
buf R3(remainder[3],remainder_fault[3]);
buf R4(remainder[4],remainder_fault[4]);
buf R5(remainder[5],remainder_fault[5]);
buf R6(remainder[6],remainder_fault[6]);

always @ (dividend_fault or divisor_fault)
begin
      accumulator =
{{(n+1){1'b0}},dividend_fault};
             if(divisor_fault ==
{(n+1){1'b0}})
             begin
                    quotient_fault  =
{n+1{1'bX}};
                    remainder_fault  =
{n+1{1'bX}};
             end
             else  if(dividend_fault ==
{n+1{1'b0}})
             begin
                    quotient_fault   =
{n+1{1'b0}};
                    remainder_fault   =
divisor_fault;
             end
             else
             begin
                for (i=0; i<(n+1) ; i = i
+1)
             begin
            if (accumulator [2*n+1:n] <
{1'b0,divisor_fault}) quotient_fault[i] =
1'b0;
           else
           begin
                  quotient_fault [i] =
1'b1;
                  accumulator[2*n+1:n] =
accumulator[2*n+1:n] -
{1'b0,divisor_fault};
           end
               accumulator [2*n+1:0] =
{accumulator[2*n:0],1'b0};
           end

                    remainder_fault  =
accumulator[2*n+1:n+1];
      end
end
endmodule
```

To these faulty and fault free modules fault simulation is performed with the reduced number of test patterns for each of the faults. The outputs obtained in each case of the faulty circuits are compared with the output of the good circuit to determine which faults are detected, finally to obtain the fault coverage. A single pattern can detect many faults or a single fault. Many patterns can detect many faults or a single fault. The challenge in testing is to obtain a minimal number of test patterns which guarantees high fault coverage of a circuit with known set of faults. The simulated waveform for the output signal remainder stuck at '1' is as shown in Figure. 1.

4. RESULTS

At the writing of this paper, we have tested our approach on combinational logic circuits and sequential circuits. The results obtained by applying our approaches to the RTL design descriptions and their corresponding Gate-level descriptions have been tabulated in table 1. At the gate-level, the gate-level netlist is created for each of the circuit used. Fault coverage is obtained for the scan inserted gate-level netlists. From the results it can be observed that the RTL Fault Coverage obtained by the proposed fault modeling methodology has a close match to the Gate-Level Fault Coverage for the tested digital circuits.

5. CONCLUSION

With the progress of semiconductor technology testing of VLSI circuits becomes more and more difficult and at the same time cost is also increasing. Therefore it is important to achieve high fault efficiency with low cost. With this approach RTL designer can have an estimation of the achieved fault coverage before doing synthesis and also it is possible for the designer to locate faults at a higher level of abstraction. At present our approach is applied to combinational logic circuits and few sequential logic circuits. Further we would like to extend the approach to complex sequential circuits such that there is a close match to the gate level fault coverage and hence reducing the impact on time to market.

Table 1. RTL versus Gate-Level Fault Coverage

Name of the circuit	RTL Fault Coverage	Gate-Level Fault Coverage
JK flip-flop	100%	100%
D flip-flop	100%	100%
Updown counter	100%	100%
Johnson counter	100%	100%
Multiplier repeated addition	100%	100%
Multiplier Booths	100%	100%
Universal Shifter	100%	100%
PISO	100%	100%
Divider using shift left algorithm	100%	100%
Division repeated subtraction	100%	100%

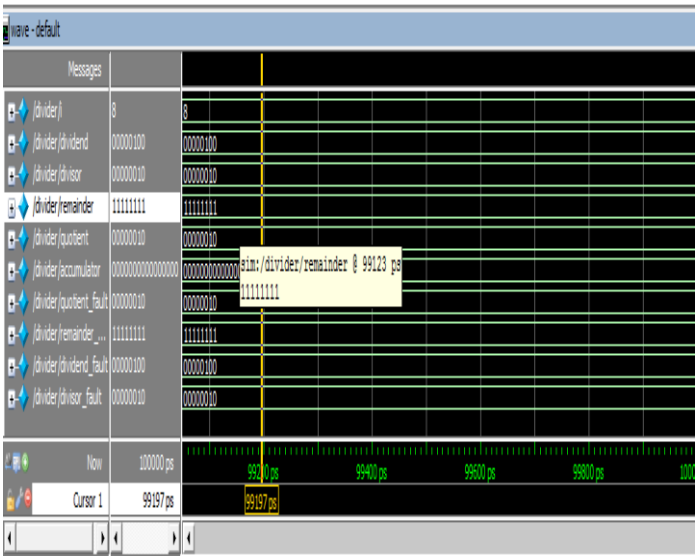

Figure 1: Simulated waveform for the output signal remainder stuck at '1'

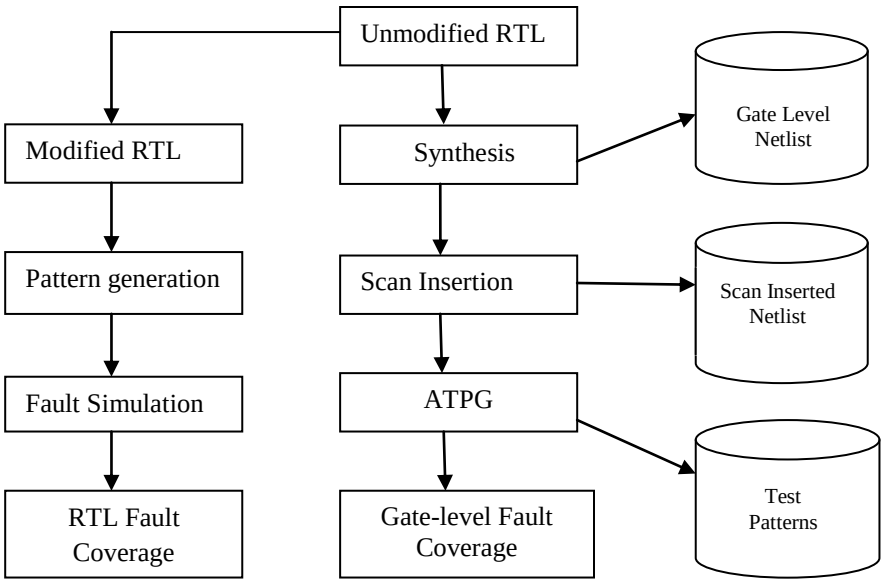

Figure 2: Design flow with the proposed method

6. REFERENCES

[1] J.Bhaskar., 2004, "Verilog HDL Synthesis, A Practical Primer", BSPublications.

[2] F.Corno, G.Cumani,M.Souxa Reorda,G.Squillero,2000,"An RT-level Fault Model with High Gate Level Correlation "Proceedings of the IEEE International High_Level Design Validation Test Workshop.

[3] Deniziak S,Sapiecha K," Developing a High-Level Fault Simulation Standard ", Computer ,May 2001,pp 89-90.

[4] Ronald J Hayne and Barry W.Johnson, 1999"Behavioral Fault Modeling in a VHDL Synthesis Environment", IEEE VLSI Test Symposium.

[5] Weiwei Mao, Ravi K Gulati, 1996,"Improving Gate Level Fault Coverage by RTL Fault Grading", IEEE Proceedings of the International Test Conference.

[6] Devadas, A.Ghosh, K.Keuter, 1996,"An Observability-Based Code Coverage Metric for Functional Simulation"Proceedings of IEEE/ACM International Conference on Computer Aided Design.

[7] Karunaratne, Sagahyroon, Prodhuturi, 2005,"RTL Fault Modeling"IEEE Circuits and Systems August.

[8]Jose M.Fernandes,Marcelino B.Santos,Arlindo L.Oliveira,Joao C.Teixeira,2006,"IEEE International High Level Design and Test Workshop.

[9] Chen C.H,Noh T.H,1998,"VHDL behavioral ATPG and fault simulation of digital systems",IEEE transactions Aerospace and Electronic Systems,April,pp 430-447.

[10] P.Goel, 1980,"Test Generation Cost Analysis and Projections," Design Automation Conference.

[11] Himanshu Bhatnagar, 2000" Advanced ASIC Chip Synthesis "Kluwer Academic Publishers.

[12] P.K.Lala, 1997,"Digital Circuit Testing and Testability, "Academic Press.