Annales UMCS Informatica AI 2 (2004) 341-349

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

RT-level fast fault simulator

Stanisław Deniziak, Krzysztof Sapiecha*

Department of Computer Engineering, Cracow University of Technology, Warszawska 24, 31-155 Kraków, Poland

Abstract

In this paper a new fast fault simulation technique is presented for calculation of fault propagation through HLPs (High Level Primitives). ROTDDs (Reduced Ordered Ternary Decision Diagrams) are used to describe HLP modules. The technique is implemented in the HTDD RT-level fault simulator. The simulator is evaluated with some ITC99 benchmarks. A hypothesis is proved that a test set coverage of physical failures can be anticipated with high accuracy when RTL fault model takes into account optimization strategies that are used in CAE system applied.

1. Introduction

Recent developments in the area of deep-submicron technology have challenged integrated circuit (IC) test methods as never before [1]. The increasing complexity of systems being designed makes test development more time-consuming. Moreover, nanometer technology has introduced new problems such as new types of defects or higher data rates. To reduce manufacturing cost and time-to-market, efficient fault detection and location should be used. One of the most essential tasks in fault diagnosis is fault simulation [2].

Current Computer-Aided Engineering (CAE) tools must address the needs for new generation of ICs e.g., systems-on-a-chip (SOC). Recent works in this area have increased emphasis on new design techniques such as high-level synthesis, behavioral synthesis, design reuse and IP-based design. For this reason, new ATPG tools that reflect new design flows should be developed, especially tools working at higher level of abstraction than gate-level.

Several approaches for High-Level Automatic Test Pattern Generation (HLATPG) have already been proposed. In Artist [3,4] a quality of generated test sequence is measured in terms of the number of blocks of statements in source description of a system activated during its true-value RT-level simulation. Artist accepts synthesizable functional register transfer level (RTL)

* Corresponding author: *e-mail address*: pesapiec@cyf-kr.edu.pl

descriptions in VHDL. A genetic algorithm is used for computing test sequences. Fault coverage (FC) of test sequences generated by Artist is generally comparable with that obtained with the help of gate-level ATPG. But these sequences are significantly longer. HTest [5] uses an ADD (Assignment Decision Diagram) representation of functional RTL descriptions. Test generation is based on symbolic computations based on RTL algebra. HTest generates test sequences comparable or better than Arists or even gate-level ATPG methods. Moreover, it is significantly faster (up to 4 orders of magnitude). However, in comparison with other algorithms, test sequences are longer. This is due to the fact that HTest does not use fault simulation to find faults already detected and no fault dropping can be performed to reduce test sequence length. In BEHATE [6] behavioral fault-free and faulty VHDL descriptions are translated into BDD (Binary Decision Diagram) representation. Test generation is based on comparison of the fault-free and faulty BDDs. In this algorithm also no high-level fault simulation is used.

Generally, recently developed HLATPG algorithms usually generate test sequences giving high stuck-at fault coverage, when simulated at the gate-level [3,5]. However, in some cases low quality test sequences are obtained despite high value of the high-level test metric. Such divergence indicates the lack of direct relationship between the high-level metrics and the gate-level stuck-at fault model.

Most commonly used high-level test metrics are based on observability basic block coverage [3] or observability statement coverage [7]. These metrics indicate the percentage of statements (or RTL blocks) that are activated by a given test pattern. It has been experimentally proved that fault models based only on statement coverage are insufficient [8] and more sophisticated fault model including bit coverage and condition coverage [9] should be used. The main problem, which makes computing high-level fault coverage (HLFC) impossible for more advanced fault models is the lack of efficient high-level fault simulator [3]. Hence, developing an efficient high level fault simulation method is a key problem which should be solved to find efficient HLATPG [10].

In this paper a new deductive bit-parallel fault simulation technique is presented for calculation of fault propagation through High Level Primitives (HLPs). ROTDDs (Reduced Ordered Ternary Decision Diagrams) are used to describe HLP modules. The technique is implemented in the HTDD RT-level fault simulator [11]. The simulator is evaluated with the help of some ITC99 benchmarks [4]. Besides high efficiency (in comparison with existing high–level fault simulators), it shows high accuracy (in terms of fault coverage) when RTL fault model takes into account optimization strategies that are used in CAE system applied.

2. High-level fault simulation

In the past, research concerning fast fault simulation methods was primarily concerned with sequential algorithms based on the architectural descriptions on the gate-level and adopting a single stuck-at fault model. Main directions in developing fast fault simulators have been the following:

1. acceleration of classic algorithms e.g. bit-parallel one: this can be achieved with reducing the number of simulated faults [12] or by using static and dynamic fault grouping [13],
2. parallel processing: this can be done thanks to fault partitioning or circuit partitioning [14]. Efficient synchronization and communication between parallel processes is the main problem in those techniques,
3. hierarchical fault simulation: in this technique some parts of the system are simulated on the gate-level while others on RT-level [15]. Gate-level fault simulation is performed only for the modules with internal faults (gate-level faults) for which there are no equivalent faults on module pins.

High-level fault simulators should accept descriptions of a system architecture consisting of HLP, like multiplexers, adders, registers, ALUs, etc., or, in the case of system level fault simulators, more complex modules like processors, controllers, memories and dedicated processing elements. The function of HLPs should be well defined while their gate-level structure may be not known. RT-level fault simulation can be applied for RT-level test generation, for optimizing DFT (Design For Testability) with full or partial scan path, and for testability analysis during behavioral synthesis (e.g., when testability is taken into consideration as a quality measure).

RT-level fault simulation seems to be the most appropriate for HLATPG conforming to new trends in CAE. Moreover, for very large systems gate-level fault simulation can not be performed due to long runtime or large memory requirements. For such systems RT-level fault simulation may be the only alternative for estimation of quality of tests generated using HLATPG.

Architectural fault simulators usually consider only stuck-at faults of all module inputs and outputs (the so called module-level faults). One of the faults is equivalent to many internal faults. A technique of fault effects propagation can be used here [15]. However, for functional descriptions usually stuck-at fault model is not sufficient. In such cases some additional functional or behavioral faults are defined. Examples of functional faults are as follows [16]:

- *stuck-at-then* (*stuck-at-else*): this is a fault where the set of statements under the *then* (*else*) clause of an *if* statement is always executed regardless of the value of its condition expression,
- *dead-clause*: this is a fault where a clause of a *case* statement is selected, but the set of statements under the clause is not executed,
- *assignment control fault*: this is a fault where the value of the right-hand expression is not transferred to the left-hand signal,

– *micro-operation fault*: this is a fault where an operator in an expression is faulted to another one,
– *behavioral stuck-at*: this is a stuck-at fault taking into consideration also so called virtual signals, i.e. unnamed signals formed by all possible sub-expressions.

Sometimes fault models for RTL modules are based on the functional analysis of their gate-level implementations [17]. In this way *Clause-CORRUPT OR* and *Clause-CORRUPT AND* faults for *if* statement (multiplexer) were defined. These faults cause that the multiplexer output is ored or anded with one of its inputs. Similar fault models were defined for adders and ALUs.

Existing RT-level fault simulation methods are based on the commercial HDL simulators [18,19]. Faults are injected into the source description of a system and simulation is repeated for each fault from a fault list. Such an approach is very time consuming. Fault simulation, even for small circuits consisting of some hundreds of gates (about 100 RTL-VHDL lines), lasts hundreds or thousands seconds for 500 random tests [18].

3. TDD-based bit-parallel deductive fault simulation for RT modules

One of the most efficient algorithms of fault simulation for the gate-level is the bit-parallel one. It is fast and can be easily implemented with small memory requirements. But it is difficult to apply this algorithm for higher levels of abstraction e.g. for RT-level. For this purpose the deductive fault simulation is much more convenient.

In the HTDD fault simulator ROTDDs are used to describe HLP modules (Figure 1). A VHDL functional description is translated into corresponding RT-level architecture (e.g., each *if* statement is replaced with a multiplexer). A new deductive bit-parallel fault simulation technique is applied for calculation fault propagation through HLPs. In Table 1, the method of calculation of fault lists propagated from inputs to the output *O* of a HLP is given.

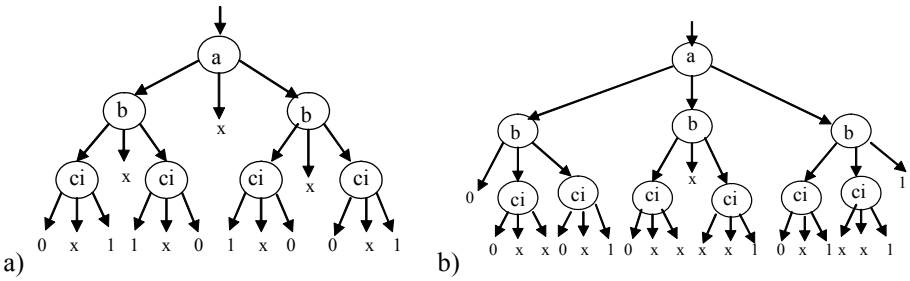


Fig. 1. ROTDD for outputs *y* (a) and *co* (b) of a full-adder

Table 1. Procedure of fault propagation for HLP

```
Propagate(O)
{
        L^0 (O)= [0..0];
        L^x (O)= [0..0];
        L^1 (O)= [0..0];
        EvaluateLists(I_0,[1...1]);
}
```

where:
I_0 is a first node in TDD (a root),
$L^0(l)$– fault list for which state of the line l is equal 0
$L^x(l)$– fault list for which state of the line l is equal x
$L^1(l)$– fault list for which state of the line l is equal 1.

Function *EvaluateLists* is the following:

```
EvaluateLists(I, L)
 {
  for(i in {0,x,1} ) {
    if(Val(I)==i){
     if(i==0) Mask = !(L^x | L^1) ;
      else if(i==x) Mask = !(L^0 | L^1);
          else  Mask = !(L^x | L^0);
     L' = L & Mask;
    }
   else L'=L & L^i(I);
   if(L'!=[0..0])
     if(Node(e^i(I ))==leaf  and  Val(Node(e^i(I ))!=Val(O))  L^Val(Node(ei(I ))) (O)+=L';
       else  EvaluateLists(Node(e^i(I )), L');
  }
}
```

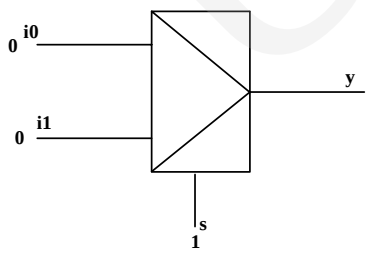
where: *Val(I)* – state of the line *I* in the fault free circuit,
Node(e^i(I)) - returns node ending edge corresponding to the value *i* and starting from the
node *I* (*leaf* means that edge ends with leaf).

The following fault model was adopted in the HTDD fault simulator:
- stuck-at-0 and stuck-at-1 faults for all signals and variables in a source description,
- stuck-at-then and stuck-at-else faults for all *if* statements,
- selection faults in *case* statements.

The last two groups of faults correspond to stuck-at faults for selection inputs of corresponding multiplexers. Static and dynamic fault reduction (Figure 2) is used whenever it is possible. More information can be found in literature [11].

4. Experimental results

The HTDD RT-level fault simulator [11] was used for verifying the hypothesis that the HLFC evaluated using RT-level fault simulation can be used for estimation of FC for gate-level [4]. The experimental results obtained for RT-level were compared to those obtained for the gate-level for the same test sets. For this purpose a standard set of ITC99 RT-level benchmarks was used [4]. Test sequences were generated using RAGE77 [20], the RT-level ATPG system based on genetic algorithm. Gate-level FC was evaluated using Hope fault simulator [12] with the fault model built in this simulator (a collapsed set of all single stuck-at faults). All computations were done using 450 MHz PC. Memory requirements and CPU time (less than 0.05s) were negligible.



(a)

Tests			y	Faults which propagate to the output y	Equivalent and implied faults
i0	i1	s			
0	0	0	0	y/1,i0/1	i0/0=>y/0 i1/0=>y/0 i0/1=>y/1 i1/1=>y/1
0	0	1	0	y/1,i1/1	
0	1	0	0	y/1,i0/1,s/1	
0	1	1	1	y/0,i1/0,s/0	
1	0	0	1	y/0,i0/0,s/1	
1	0	1	0	y/1,i1/1,s/0	
1	1	0	1	y/0,i0/0	
1	1	1	1	y/0,i1/0	

(b)

Fig. 2. Static and dynamic fault reduction for a multiplexer (a). Static fault reduction (fault collapsing) deletes all equivalent and implied faults from simulation (faults y/0 and y/1 for the multiplexer). Dynamic fault reduction drops faults that do not propagate through the module for a given test (faults s/0 and i0/1 in this example)

In the first group of experiments none of fault reducing techniques on RT-level was applied. Figure 3 shows the relationships between HLFC calculated for RT-level and the gate-level of the benchmark descriptions. HLFC obtained on RT-level was comparable with that obtained with the help of gate-level fault

simulation (e.g., for B02, B04, B06, B09), but for some circuits HLFC is significantly smaller (e.g., B01, B03). This might be caused by several factors: an inadequate fault model adopted at the RT-level, no fault collapsing applied, benchmark characteristics (no resource sharing on RT, for example), or by a test pattern generation method (RAGE77 is based on statement activation in the source description during simulation and may activate faults of selected classes only).

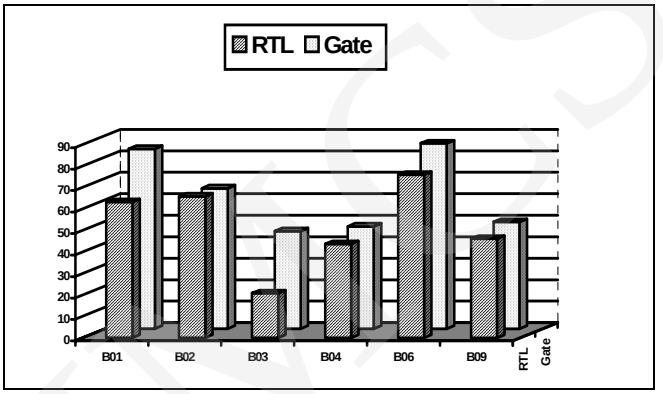


Fig. 3. RTL-level and gate-level FCs for benchmark circuits

Table 2. FC for B01 benchmark

Test no.	Test length	Gate-level FC (131 collapsed faults)	RT-level FC (198 faults)	RT-level FC (118 collapsed faults)	RT-level FC, with resource sharing (85 collapsed faults)
1	16	85.50	66.67	67.80	92.94
2	10	47.33	27.78	31.36	40.00
3	18	87.78	62.63	62.71	89.41
4	18	92.37	74.75	72.03	94.12
5	18	90.08	74.24	67.80	91.77
6	18	83.21	65.15	68.64	85.88
7	18	93.13	71.72	69.50	94.12
8	38	84.73	73.74	77.97	91.77
9	18	83.21	54.04	55.09	83.53
10	18	84.73	68.69	66.95	88.24
11	18	89.31	68.18	65.25	91.77
12	18	95.42	71.21	70.34	91.77
13	18	72.52	53.03	52.54	76.47

The next group of experiments revealed the reason for this phenomenon. A representative example is shown in Table 2. Fault collapsing made the calculations shorter but did not change the relationships between FC on RT and

gate levels. On the contrary, a better fault model due to resource sharing appropriate to the strategy of CAE system used (Design Compiler by Synopsys in this case) did well.

5. Conclusions

In our experiments the HTDD fault simulator was used. In this simulator, functions of all modules have to be described using TDD. It is possible to generate such descriptions directly from any standard VHDL or Verilog specification. It is also possible to define different functional fault models. This can be done simply by modifying TDD.

A new TDD-based fault simulation technique was developed. It combines advantages of both bit-parallel and deductive fault simulation. It is fast and easy for implementation. It can be used for evaluation of high-level fault models [21] and in simulation based test generation methods [22]. It can also be built into existing HLATPG algorithms to enable fault dropping.

RT-level fault simulation can not evaluate accurate value of FC, yet. It gives us only the estimation of FC. Precise value of FC can only be computed using gate-level or hierarchical fault simulators. Sometimes high-level fault simulators can not be useful. The lack of information about the structure of some modules may cause that some physical defects can not be modeled.

However, high-level fault simulation, particularly RT-level one, has many advantages making this approach very attractive and useful. First, test set coverage of physical failures can be anticipated with high accuracy when RTL fault model takes into account optimization strategies that are used in a CAE system applied. The possibility of simulating modules for which corresponding gate-level structure is not known, e.g., embedded cores, is guaranteed. HTDD representation enables evaluation of different high-level fault models and metrics. Each TDD corresponds to one RTL block or VHDL statement, so block coverage or statement coverage metrics can be computed simply by tracing TDD activities. Moreover, all metrics commonly used in software engineering (e.g. branch coverage, condition coverage, path coverage) [9] can be calculated in the similar way. Hence, our HTDD fault simulator can also be used for software test generation. Lower memory requirements and greater speed (simulated systems have fewer components and hence fewer faults collapsed) can be achieved through static and dynamic fault reduction. This does not seem to be as efficient as in the case of gate-level. Information about test quality and about expected testability of the designed system can be obtained before logic synthesis. If untestable components of the system can be located in the corresponding functional description, then the system can be easily redesigned for increased testability.

References

[1] Zorian Y., *Testing the Monster Chip*, IEEE Spectrum, 36(7) (1999) 54.
[2] Levendel Y., Menon P.R., *Fault simulation* in *Fault Tolerant Computing: Theory and Techniques*, ed. D.K.Pradhan, Prentice-Hall, Englewood Cliffs, 1 (1986) 184.
[3] Corno F., Reorda M.S., Squillero G., *High-Level Observability for Effective High-Level ATPG*, VLSI Test Symposium, (2000).
[4] Corno F., Reorda M.S., Squillero G., *RT-level ITC'99 Benchmarks and First ATPG Results*, IEEE Design & Test of Computers, July-Sept., (2000) 44.
[5] Zhang L., Ghosh I., Hsiao M., *Efficient Sequential ATPG for Functional RTL Circuits*, Proc. of the International Test Conference, (2003) 290.
[6] Ferrandi F., Fummi F., Sciuto D., *Implicit Test Generation For Behavioral VHDL Models,* Proc. of the International Test Conference, (1988) 587.
[7] Fallah F., Devadas S., Keutzer K, *OCCOM – Efficient Computation of Observability-Based Code Coverage Metrics for Functional Validation*, IEEE Transactions On Computer-Aided Design of Integrated Circuits And Systems, 20(8) (2001) 1003.
[8] Goloubeva O., Sonza Reorda M., Violante M., *Experimental Analysis of Fault Models For Behavioral-Level Test Generation*, IEEE Design & Diagnostic of Electronics Circuits and Systems, (2002) 416.
[9] Ferrandi F., Ferrara G., Sciuto G., Fin A., Fummi F., *Functional Test Generation for Behaviorally Sequential Models,* Proc. of the Design Automation and Test in Europe, (2001) 403.
[10] Deniziak S., Sapiecha K., *Developing a High-Level Fault Simulation Standard*", IEEE Computer, May (2001) 89.
]11] Sapiecha K., Sapiecha J., Deniziak S., *HTDD Based Parallel Fault Simulator*, Proc. of the 5th IEEE Int. Conference on Electronics, Circuits and Systems, Lisbon, 2 (1998) 217.
[12] Lee H. K., Ha D. S., *HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 15(9) (1996) 1048.
[13] Graham C.R., Rudnick E.M., Patel J.H., *Dynamic Fault Grouping for PROOFS: A Win for Large Sequential Circuits*, Int. Conference on VLSI Design, January (1997) 542.
[14] Pakers S., Banerjee P., Patel J.H, *A parallel algorithm for fault simulation based on PROOFS*, Proc. IEEE Int. Conf. Computer Design, October (1995) 616.
[15] Samb D.G., Mueller-Thuns R.B., Blaauw D., Rahmeh J.T., Abraham J.A., *Hierarhical Multi-Level Fault Simulation of Large Systems*, Journal of Electronic Testing Theory and Applications, 1(2) (1990) 139.
[16] Cho C.H., Armstrong J.R., *B-algorithm: A Behavioral Test Generation Algorithm*, International Test Conference, (1994) 968.
[17] Hayne R.J., Johnson B.W., *Behavioral Fault Modeling in a VHDL Synthesis Environment*, VLSI Test Symposium, (1999) 333.
[18] Corno F., Cumani G., Sonza Reorda M., Squuillero G., *RT-level Fault Simulation Techniques based on Simulation Command Scripts,* Proc. of Design of Circuits and Integrated Sytems, (2000) 825.
[19] Fin A., Fummi F., *A VHDL Error Simulator for Functional Test Generation*, Proc. of the Design Automation and Test in Europe, (2000) 390.
[20] Corno F., Prinetto P., Reorda M.S., *Testability analysis and ATPG on behavioral RT-level VHDL*, International Test Conference, (1997) 753.
[21] Corno F., Reorda M.S., Squuillero G., *An Interpretation Framework for Evaluating High-Level Fault Models and ATPG Capabilities,* Proc. of Design of Circuits and Integrated Systems, (2001) 273.
[22] Guo R., Pomeranz I., Reddy S.M., *A Fault Simulation Based Test Pattern Generator for Synchronous Sequential Circuits*, VLSI Test Symposium, (1999) 260.