# Which Set of Bridging Faults Should Test Compilers Target ? [1]

Sreejit Chakravarty    Paul J. Thadikaran
Department of Computer Science
State University of New York
Buffalo, NY 14260

February 28, 1996
( SUBMITTED TO IEEE INTERNATIONAL TEST CONFERENCE, 1996 )

**ABSTRACT**

Existing test compilers target stuck-at faults. Recent experimental evidence suggest targetting bridging faults (BF) because they model between 40-50% of physical defects. For BFs, one suggestion has been to extract a small set of BFs using "layout analysis". We argue that this is not a feasible option for test compilers thereby making a case for "layout independent analysis" of BFs.

Opting for "layout independent analysis" implies that computer-aided test tools target "all possible BFs". Targetting such a large number of faults is generally considered unreasonable. We present new experimental data and analyze published data. The analysis reveal that for many problems very good techniques have already been developed to target such a large number of faults.

# 1 Introduction

There has been an increasing trend to integrate "test related tasks" early in the "design cycle". This is loosely termed "test compilation". A variety of commercial test compilers are already available. There is one similarity among these compilers: all of them base their analysis/computation on the single stuck-at fault model. Experimental results have shown the inadequacy of the stuck-at fault model to model physical defects. In addition, it has been shown that bridging faults(BFs) model between 40-50% of physical defects[23, 36, 42]. Consequently, the next generation of test compilers must target BFs in addition to other faults.

In this discussion we ask a very fundamental question, answer to which is required, prior to developing the next generation of test compilers. The question is: Which set of BFs must test compilers target ?

[1] Research partially supported by NSF Grant No MIP9102509

Conventional wisdom suggests that in order to contain the computational cost test tools for BFs should target a "small subset" of BFs. With this in mind many test tools restrict the target set of BFs to a "realistic set of BFs". Such a set of BFs is computed by analysing the layout of the circuit[29]. Techniques developed for stuck-at faults, that "explicitly enumerate" ( one by one ) this set of BFs, can then be used.

Is this an adequate answer to the question we raised ? The above approach can be used if the layout of the circuit is available to the test tools. We argue in Section 2 that test compilers often do not have any layout information. Thus, test compilers are **forced to use layout independent analysis of BFs**. This contradicts the assumption of much of the research on BFs where the availability of layout information is assumed.

Given that test compilers are forced to perform *layout independent analysis* of BFs the immediate question that needs an answer is: which set of BFs do test tools target ? Two possible answers are: (i) target all possible BFs; and (ii) randomly select a small set of BFs. Conventional wisdom rejects the first option because the number of such faults is too large. The second option leaves us with the difficult task of understanding the quality of the test sets. Infact, we cannot determine the test quality !! Based on the above reasoning we believe that for "layout independent analysis", which test compilers are forced to perform, they must target all possible BFs.

Targetting such a large number of faults, especially for larger circuits, is a "very scarry thought". Most researchers would, based on the number of target faults alone, argue that it is unreasonable to target all possible BFs. We argue that there is increasing evidence that contradicts this popular myth.

The popular scepticism stems from the misconception that performance of test tools grow "almost linearly" with the "number of target faults". This misconception is based on the assumption that test tools must explicitly enumerate faults. This is true of all techniques used by test tools targetting stuck-at faults. We argue that this need not be the case when analysing BFs. There are a growing number of techniques that fall into this category. We will not discuss these techniques. However, we analyse some new as well as published experimental data. This analysis reveals that, for many of the fundamental problems, it is not at all unreasonable to target all possible BFs.

We hasten to add though that there is still a need to develop techniques that target all possible BFs for a number of other problems. The intent of this discussion is twofold: to highlight the fact that it is not necessarliy unreasonable to target all BFs and in some situations there is no choice; and to urge researchers to direct their energy to solve the unsolved problems and to improve solutions that have been presented.

The rest of the paper is organized as follows. In section 2 we discuss why layout independent analysis of BFs is required. In section 3 we present an analysis of some new and published experimental data. We conclude in Section 4.

# 2 Why Layout Independent Analysis ?

**Layout independent analysis** means that test sets to detect BFs have to be evaluated and computed in the absence of any layout information. In the next few subsection, a variety of situations in which *layout analysis*, to extract a *realistic set of BFs*, cannot be used are presented. This supports the claim that for several *test compilation tasks* the test tools must perform *layout independent analysis.*

To understand this, we first have a look at how the design flow has changed with the advent of "test compilation". Next, we look at a variety of design for testability tasks, usually performed by test compilers, where we cannot have knowledge about the layout. We then look at the problem of synthesizing board level tests and point out the difficulty of using layout dependent analysis. Finally, we argue why, for some diagnosis problems also, we cannot assume knowledge of the layout.

## 2.1 Changes in the Design Flow

For our purpose we include both "fabrication" and "test set preparation" activities in "design flow". Figure 1 shows two different design flows.

Design Flow I is the one assumed by **inductive fault analysis**[29]. It is used by many chip designers who generate the layout, extract a set of BFs and then compute the test set. All the layout information is available to the test tool. A *realistic set of faults* can be extracted by analysing the layout. These set of faults can then be targetted by the test tools.

The scenario changes when *test compilation* is included in the design flow and Design Flow I cannot be used. The design flow will depend greatly on the kind of *test related* *tasks* that are performed by the *test compiler*. Rather than list all possibilities we provide a simplified design flow depicted as Design Flow II in Figure 1. It captures the essence of the point we are trying to make.

The important point to note about the change in the design flow is that the design house/group, after finalizing the *initial gate level design* (without any test structure) gives the test compiler a gate level description of the circuit. Such a service can be provided either in-house by a different group or by another organization. The test compiler then, using testability analysis, modifies the gate level description to improve the testability of the circuit. The modified gate level description follows the path labelled BDEF and at the end of it we get the fabricated chips. The modified gate level description is also used by the test compiler to generate a good quality test set.

If we only consider stuck-at faults, no layout information is used and the impact of the change in design flow on test tools has not been of much concern. When test tools start targetting BFs this change in the design flow raises some very important questions about the set of BFs the next generation of test tools must target. Here we argue that the change in the design flow implies that no layout information is available to the test tools, making up the test compiler, for many tasks it must perform.

## 2.2 Design for Testability Tasks

We consider some of the design for testability tasks that are usually performed by test compilers. For none of these tasks can we assume any information about the layout. Our assumption is that the test tools target BFs.

- Consider the task of determining which LFSR to be used as a test generator for a portion of the circuit. Fault simulation of BFs, to determine fault coverage and test length of the psuedo random test set, is performed. Since the gate level description is not yet finalized one cannot have any layout information prior to this fault simulation task.

- Weighted random patterns are used to reduce test length in a BIST environment[1, 31, 38, 44, 53, 54, 55]. In addition to determining an appropriate LFSR, the weight sets have to be computed. These weight sets determine the design of the on-chip test generator. The weight sets are computed using fault simulation. Clearly, for the required fault simulation, no layout information is available.

- In one approach to designing on-chip test generators a test set is used as a guide[26, 27, 43, 52]. This approach improves fault coverage and reduces test application time. Since the on-chip test generator has not yet been finalized, the test set used has to be computed without any layout information.

- In a full-scan environment reduction of test application time is an important objective. Solutions to this problem use a combination of two techniques: finding a good ordering of flip/flops in the scan-chain; and finding a good order in which to apply the test vectors. The heuristics use extensive fault simulation and requires a test set[20, 25, 35, 41, 45].

  The circuit design is not complete prior to this task. Thus, the test set that is used as a guide has to be computed without any knowledge of the layout. A similar observation can be made about the fault simulation tools that are required.

- In a partial scan environment, the primary objective is to select a subset of the flip flops and properly order the selected set of flip/flops. Flip-Flop selection algorithms require extensive fault simulation and testability analysis[**?**]. Any simulation, or other kind of testability analysis tool used cannot assume knowledge of the layout because the design has not yet been finalized.

For all these examples one is tempted to make the following argument. Adding the testability structure modifies about 10 or 15, maybe 20% of the circuit. The two circuits, before and after adding the test structures, are "almost the same". So why not generate a <u>preliminary layout</u> without the test structures ? Use the *preliminary layout* to extract BFs using <u>proximity relationship between nets</u>. After the design is completed and the test structure is inserted generate the <u>final layout</u>. The circuit tested is the one fabricated using the *final layout*.

The above argument is flawed for the following reason. The *preliminary layout* and *final layout* will be generated using two different runs of the same layout generator. All layout generators use non-deterministic algorithms like simulated annealing, genetic-algorithm, simulted evolution etc. for placement and routing. Examples include variants of TimberWolf[47]. Therefore, even if we have exactly the same circuit description and the same layout generator we would get vastly different layouts from two different runs !! The *proximity relationship between nets*, even for identical circuits, leave alone circuits that are *almost the same*, will not be preserved across two layouts generated by two different runs of the same layout generator. *Proximity relationship between nets* is the basis on which faults are extracted[29]. **The consequence is that, the "realistic set of faults" extracted using the *preliminary layout* will not be a "realistic set of faults" for the *final circuit* !!**

## 2.3 Synthesizing Board/MCM Level Tests

We have so far considered the design flow for a single VLSI chip. Such chips are often integrated into larger systems. Two examples are board level and MCM level integration. As shown in Figure 2 the VLSI chips, or their unpackaged counter part for MCMs, are mounted on a board/substrate.

For testing, the boundary scan test structure is built into the board/MCM. The assembled system is tested by first testing the boundary scan interconnect and then applying the test sets for each of A, B, C and D either in turn or in groups. In either case, test sets for each of A, B, C, D are required.

When such assemblies are manufactured in large volume the instance of A, B, C, D etc. usually come from a number of different IC vendors. In this case, if different test sets depending on the layout used by the supplier is used, there will have to be a large number of distinct test sets for the assembly. During production testing, for each possible combination of suppliers for A, B, C and D the proper test set will have to be selected. For example, assume that there are just two suppliers for each of A, B, C and D. This implies that a total of 16 different tests for this assembly will have to be maintained. The number of such distinct tests grow exponentially with the number of chips. This is clearly not a feasible option.

One way to get around this is to use a standard test set (independent of the supplier layout) for each of these chips. If BFs are to be targetted then a standard test set obtained using layout independent analysis of BFs have to be computed.

## 2.4 Diagnosis Problems

The above arguments justify performing layout independent analysis of BFs by test tools that compute test sets, compute fault coverage etc. These arguments do not justify performing layout independent analysis of BFs for diagnosis. If we consider *failure analysis* ( or *defect diagnosis* ) where we are interested in identifying defect locations that caused the chip to fail the layout of the chip is known. In this case, a *realistic set of BFs* can be extracted and a suitable diagnosis algorithm used to locate defects as is done in [12].

On the other hand, diagnosis is also used to identify design, layout and fabrication errors prior to mass scale manufacturing[24]. This is often referred to as *design validation.* $I_{DDQ}$ testing has been used very effectively for this purpose. As stated in [24]: "$I_{DDQ}$ testing is routinely used to detect layout errors by companies using gate-arrays". Such errors could include missing contact of pins from other pins in the net, shorting of two distinct nets etc.

The later kind of errors can be modelled as BFs. Such errors donot occur as a result of any kind of manufacturing defect. They cannot be modelled by point defects. For such errors we cannot assume anything about the closeness of the nets that are shorted, as is assumed in layout analysis where point defects are assumed. An error could result in two nets that are very far apart in the layout to be bridged. Hence these errors cannot be modelled by *realistic set of BFs.* Thus, to locate such errors, diagnosis tools that consider all possible BFs, while locating faults, are required.

# 3 Analysis of Experimental Data

If we consider all *fault detection tasks* and *diagnosis tasks* that a test compiler needs to perform there are a large number of computational problems that has to be addressed. We can start to address the rest of them once we have a better understanding of the fundamental problems both in detecting and diagnosing BFs.

Considerable progress has been made to solve the fundamental problems in diagnosing all possible BFs. They include work on diagnosis of BFs[8, 9, 10, 11, 13], $I_{DDQ}$ diagnostic test generation for BFs[21, 22], and diagnostic simulation of BFs[6]. Here we concentrate on solutions for the fundamental fault detection problems for BFs. They include: fault simulation, test generation and test set compression. BFs can be detected using either $I_{DDQ}$

measurement or Voltage/Logic measurement. Test tools must be developed for both these detection techniques.

Our strategy in analysing the available experimental data is to compare the performance of two kinds of test tools. Tools that target a **subset of BFs** and tools that target **all possible BFs**. The first category includes faults that have been derived using inductive fault analysis or is the set of leakage faults. Subsequently, we refer to such a set of faults as **Extracted-BFs**.

There are two different sets of BFs in the second category. Primary inputs and gate outputs are the **lines** of the circuit. The set of all possible BFs between *lines* of the circuit are known as **Metal-BFs**. In addition to *lines* of the circuit there are other nodes internal to each gate. Such nodes are referred to as **internal nodes**. The set of all possible BFs between any pair of line and/or internal nodes is referred to as **All-BFs**.

*All-BFs* is the most comprehensive set of BFs. However, in many situations when layout independent analysis is required it may be reasonable to target only *Metal-BFs*. Consider the case of gate-array based designs. The only layer used for customizing a design is the metal layer. This layer interconnects *lines* in the same net. The array is tested prior to customization. In this case a reasonable layout independent set of BFs is the the set of BFs that occur between *lines* in the circuits which is the set of *Metal-BFs*.

**In our analysis we compare the performance of test tools for the following two cases: *Extracted-BFs, Metal-BFs*; and *Extracted-BFs, All-BFs*.**

Another issue that we have to contend with is the fact that the performance results of the available test tools that we compare are on different workstations. The three workstations in question are: SUN 4/670, SUN 5/70 and HP9000/735. The available performance ratings are shown in Table 1. Based on these ratings we use the average relative performance ratings shown in Table 2. We also assume that this is also the speedup factor. Thus, we assume that the same code runs 2.6 times faster when run on a SUN5/70 instead of a SUN4/670. In reality the actual speed up will be a little different from this number because it is also a function of the compiler used, the optimization level used etc. However, we believe that it will not differ much from the speedup numbers assumed. In addition, this difference will not change the broader conclusions that we are seeking in this analysis.

## 3.1 $I_{DDQ}$ Test Tools

Considerable progress has been made to develop techniques for $I_{DDQ}$ measurement based tools that target *All-BFs*, *Metal-BFs* and *Extracted-BFs*. In the next two subsections we analyze published data on fault simulation and test generation. In the third subsection we analyze new data as well as published ones on test set compression which is similar to the problem of computing $I_{DDQ}$ measurement points. We do not discuss the details of the techniques used but merely point out the following characteristics of the solution. All existing techniques for developing computer-aided testing tools for stuck-at faults enumerate the faults they target explicitly i.e. one by one. Such a fault enumeration scheme, when a large number of BFs is targetted, is a *recipe for disaster*. So, *implicit enumeration* techniques for representing BFs were developed. This was integrated with efficient fault dropping techniques to derive efficient fault simulation, test generation and test compression algorithms.

The total number of *Metal-BFs*, *All-BFs* were computed assuming that the *fanout stem* along with all its *fanout branches* correspond to only one node. The numbers reported are the number of BFs prior to *fault collapsing* by identifying equivalent nodes [50].

### 3.1.1 Fault Simulation

Fault Simulation of $I_{DDQ}$ Tests for *All-BFs*[15, 51] and *Metal-BFs*[14, 16, 17] have been studied for both combinational as well as sequential circuits.

Among the published results the performance of [51] is the best for both *All-BF* and *Metal-BF*. There is almost no published results on fault simulation of *Extracted-BFs*. We have some results of our own[51] which we use.

Table 3 compares the performance of fault simulation for *Extracted-BFs* and *Metal-BFs* for Sequential circuits. The test set used is that derived using GENTEST[7]. Only results for ISCAS89[3] circuits of more than 1000 gates for which GENTEST test set was available are presented. The times are CPU seconds on a SUN 4/670. The number of faults in the two sets is given under column *Faults* under the headings *Extracted-BFs* and *Metal-BFs*. The column *Faults* under heading *Ratio* gives the ratio of the size of *Metal-BFs* and *Extracted-BFs*. The columns labelled *Time* are similarly defined. A closer examination of columns *Faults* and *Time* under the heading *Ratio* suggests that fault simulation of *Metal-BFs*,**inspite of having a considerably larger number of faults, actually requires less time than**

**the simulation of *Extracted-BFs*!!**

This very counter intuitive is not really surprising and can be explained as follows. The simulation algorithm for *Extracted-BFs* reduces to explicitly enumerating the faults and simulating them as is done for stuck-at faults. Note that this is what most commercially available $I_{DDQ}$ simulation tools do for *Extracted-BFs*. On the other hand, implicit enumeration techniques and efficient fault dropping rules can be exploited for *Metal-BFs*. It would be fair to argue that these simulation tools are prototypes and they could be speeded up using parallel pattern, concurrent and other fault simulation techniques. For the sake of argument, let us assume that the simulation algorithm for *Metal-BFs* may not be speeded up by as much using techniques like parallel fault simulation etc. We assume that the net effect of these improvements is that the running time of *Extracted-BF* will be improved by a factor of 4 or 5 over the running time of *Metal-BFs*. We therefore use a multiplicative factor of 4 to get an *Adj Time Ratio* which is the projected Time Ratio. Even with such adjustments we note that there is a marginal difference in the running time inspite of the vast difference in the number of faults.

Table 4 contains similar data for *Extracted-BFs* and *All-BFs*. The *Adj Time Ratio* and the *Fault Ratio* suggests that for all circuits a similar conclusion, as was made for *Metal-BFs*, can be drawn.

Tables 5, 6 present a comparison of the performance of fault simulation for *Extracted-BFs* with respectively *Metal-BFs, All-BFs* in combinational circuits. The test set used was generated using [32]. The circuits whose names are prefixed with "c" are from[2] while the circuits whose names are prefixed with "s" are scan versions of sequential circuits from [3].

From these tables we conclude the following. **If proper techniques are developed, the running time of fault simulation algorithm can be made to grow very slowly compared with the growth in the size of the fault list !!** Infact, considerable progress has already been made in this direction.

### 3.1.2 Test Generation

Techniques for computing $I_{DDQ}$ tests for *All-BFs*, *Metal-BFs* have been addressed both for combinational[4, 5, 14, 37, 18, 28, 46, 49] and sequential circuits[34]. Some of them target *Extracted-BFs*[4, 37, 18, 28] while others target either *Metal-BFs*[5, 14, 46, 49] or

*All-BFs*[49, 34]. For *Extracted-BFs* the best performance is reported in [37]. For *Metal-BFs, All-BFs* in combinational circuits the best performance is reported in [49]. For *All-BFs* in sequential circuits the only result is reported in [34].

In Table 7 we compare the performance of the test generator for *Metal-BFs*[49] and the test generator for *Extracted-BFs*[37]. In [49] SUN4/670 was used whereas in [37] an HP9000/735 was used. Accordingly, as explained above, $Adj\ Time = \frac{Time(Metal-BFs)}{Time(Extracted-BFs)}/4.47$. The factor 4.47 is from Table 2. **Note that the ratio of the test generation time is significantly smaller than the ratio of the number of faults targetted!!**

In Table 8 we compare the performance of the test generator for *All-BFs*[49] and the test generator for *Extracted-BFs*[37]. In [49] SUN4/670 was used whereas in [37] an HP9000/735 was used. Accordingly, as explained above $Adj\ Time = \frac{Time(All-BFs)}{Time(Extracted-BFs)}/4.47$. The factor 4.47 is from Table 2. In this case the results are not as good as in the other comparisons that we have made so far. However, we should not rush to conclude that targetting *All-BFs* is unreasonable. We have to keep a number of points in mind.

Firstly, in [49] the stuck-at test generator ATALANTA[33] was used. A modified circuit along with a list of stuck-at faults was generated and stored in a file. ATALANTA then read the file from secondary storage and generated some additional vectors. Clearly, this is not a good strategy to develop an $I_{DDQ}$ test generator. A system that integrates the stuck-at ATPG into the $I_{DDQ}$ test generator would perform considerably better.

Secondly, during test generation of *Metal-BFs* or *All-BFs* a large number of the simulated stuck-at faults are redundant. A stuck-at test generator that identifies redundancy efficiently would result in better performance. CONTEST, SOCRATES etc. [30, 37, 40, 48] used in [37] are considerably better at this than ATALANTA. We conjecture that the results in [49] would be considerably better if these stuck-at ATPG instead of ATALANTA is used.

There is very little data on computing $I_{DDQ}$ tests for either *Metal-BFs* or *All-BFs* in sequential circuits. The work reported in [34] is, we believe, very preliminary. The experiments were run on a SUN 5/70. We have compared the performance of the sequential test generator from [34] and the test generator for *Extracted-BFs* in combinational circuits[37]. The results are tabulated in Table 9. Ignore the results for s9234, s13207, s15850 because the coverage of the sequential test set is very low. For others, the sequential test generator actually outperforms the combinational test generator in the following sense. The adjusted time ratio grows very slowly compared to the fault ratio. However, we caution our readers not to jump to any conclusion. The sequential test generator cannot identify redundant faults and saves time required for redundancy identification.

From all these data we cannot make strong conclusions about test generation for *All-BFs*. However, the techniques are already good for some circuits which gives us reasons to be very hopeful about its feasibility.

### 3.1.3 Test Compression

Test set compression usually refers to the reduction of the size of a test set without sacrificing coverage. In the context of $I_{DDQ}$ testing, since $I_{DDQ}$ testing is a slow process, the same technique can also be used to identify $I_{DDQ}$ measurement points[39]. These problems have been studied for leakage faults[19], *All-BFs* and *Metal-BFs* in both sequential and combinational circuits[15, 50].

Only [37] gives comprehensive data for this problem for *Extracted-BFs* in combinational circuits. For *Metal-BFs, All-BFs* [50] provides the best results for combinational and sequential circuits. In Table 10 ( Table 11 ) we compare results for *Extracted-BFs* and *Metal-BFs* ( *Extracted-BFs* and *All-BFs* ). In [50] several algorithms are discussed. We use the performance results for the algorithm that results in most compression. This algorithm also requires the most amount of time. The time & fault ratios were computed as was done in the previous cases.

The size of the initial test set has a significant impact on the performance of the two algorithms. Infact, the complexity of both the heurisitcs is at least quadratic in the size of the initial test set size. In comparing the running time of the two algorithms we used the same sized initial test set size. Both for *All-BFs* and *Metal-BFs* the test set used was the first $N$ vectors from an uncompressed test set for *All-BFs* from [49]. Here, $N$ is the size of the initial test set used in [37].

From Tables 10, 11 we conclude that: **for both *All-BFs, Metal-BFs*, if proper techniques are developed then the growth of the ratio of the time to compute $I_{DDQ}$ measurement points is considerably smaller than the fault ratio**. Infact, considerable progress has been made in this direction.

### 3.2 Observation

It would be misleading to claim that these problems have been completely solved. Only performance results of initial prototypes are available. The results for *Metal-BFs* even with the prototypes are very good. These results certainly demonstrates the feasibility of using *Metal-BFs* for $I_{DDQ}$ testing.

The performance of these systems for *All-BFs* need more improvement. However, the results are very encouraging. For reasons discussed in the introduction and section 2, and keeping these observations in mind, commercial CAD tool vendors should direct their energy to develop such tools.

### 3.3 Logic Testing

The story is very bleak if we assume that logic testing is used to detect BFs. The main stumbling block has been to model the behaviour of circuits with BFs in them. On the one hand we have the wired, voting and biased voting model. These are approximate gate level models. On the other hand very detailed switch and circuit level models have been used. Apart from the wired model, which is applicable to a restricted class of CMOS circuits, these models are too cumbersome. Good techniques to use these models that target all possible BFs need to be developed. Existing techniques make it possible to solve such problems only for some small circuits. The models need to be refined. Better techniques to develop CAD tools that target all BFs need to be developed.

## 4 Concluding Remarks

The purpose of this paper was to discuss some very important issues that will have to be addressed when developing the next generation of test compilers. A decade of investigation should have convinced test engineers that defects manifesting as BFs should be targetted by the tools they use. There is a popular belief that layout analysis can be performed and a small set of BFs extracted for test tools to target. We hope that we have provided sufficient arguments to convince our readers that in many situations layout information is not available. Hence, layout independent analysis is required.

Given that layout independent analysis is required, an issue of importace to developers of computer-aided test tools is: which set of BFs should these tools target ? We believe that we should target *All-BFs*. In some cases we can get away with targetting *Metal-BFs*. There are no doubt sceptics among our readers who would discard such an idea citing the difficulty of targetting such a large number of BFs. We believe that this is not the case. We also believe that good techniques can be developed to target such a large number of faults. In fact, for some problems very powerful techniques have already been developed. One certainly will not argue with the experimental evidence available for *Metal-BFs* and $I_{DDQ}$ testing. We also believe that the methods for *All-BFs* and $I_{DDQ}$ testing are pretty good at this point and can be improved further. We still need to explore techniques to solve many of the fundamental problems to detect and diagnose BFs when logic testing instead of $I_{DDQ}$ testing is used.

We hope that we have made a strong enough case to encourage researchers to concentrate on these problems. The problems are going to be difficult. But researchers ought to take up the challenge if future test compilers are going to be effective.

# References


[1] M. Bershteyn, "Calculation of Multiple Sets of Weights for Weighted Random Testing," *IEEE Int'l Test Conference*, 1993, pp. 1031 - 1040.

[2] F. Brglez and H. Fujiwara, "A Neural Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN", Special Session on ATPG and Fault Simulation, *IEEE Int'l Symposium on Circuits and Systems* 1985.

[3] F. Brglez , D. Bryant, K. Kozminski, "Combinational profiles of sequential benchmark circuits", Proceedings of *IEEE Int'l. Symposium on Circuits and Systems*, 1989, pp. 1929-1934.

[4] S. W. Bollinger, S. F. Midkiff, "Test Generation for $I_{DDQ}$ Testing of Bridging Faults in CMOS Circuits", *IEEE Trans. on Comput.-Aided Design*, pp. 1413-1418, 1994.

[5] S. Chakravarty, "Computing Tests for Bridging and Leakage Faults: Complexity Results and Universal Test Sets," *5th IEEE/ACM International Conference on VLSI Design*, 1992, pp. 49-54.

[6] S. Chakravarty, "A Sampling Technique for Diagnostic Fault Simulation", *IEEE VLSI Test Symposium, 1996.*

[7] W. T. Cheng, T. J. Chakraborty, "GENTEST: An automatic Test Generation System for Sequential Circuits", *Computer*, April 1989.

[8] S. Chakravarty and Y. Gong, "An Algorithm for Diagnosis of Bridging Faults", *30th ACM/IEEE Design Automation Conference*, 1993.

[9] S. Chakravarty and Y. Gong, "Voting Model Based Diagnosis of Bridging Faults in Combinational Circuits," *IEEE International Conference on VLSI Design*, 1995, pp. 338 - 342.

[10] S. Chakravarty and M. Liu, "Iddq Measurement Based Diagnosis of Bridging Faults in Combinational Circuits," Journal of Electronic Testing: Theory and Application, Vol. 3, pp. 377 - 385, 1992. Kluwer Academic Publishers.

[11] S. Chakravarty and M. Liu, "Algorithms for Current Monitor Based Diagnosis of Bridging and Leakage Faults". *29th ACM/IEEE Design Automation Conference*, pp. 353-356, 1992.

[12] B. Chess, D. B. Lavo, F. J. Ferguson, T, Larabee, "Diagnosis of Realistic Bridging Faults with Single Stuck-at Information", *IEEE/ACM Conference on Computer-Aided Design*, 1995, pp. 185-192.

[13] S. Chakravarty and S. Sunder, "Iddq Measurement Based Diagnosis of Bridging Faults in Full-Scan Circuits". *IEEE International Conference on VLSI Design*, 1994, pp. 179 - 182.

[14] S. Chakravarty, P. J. Thadikaran, "Simulation and Generation of $I_{DDQ}$ Tests for Bridging Faults in Combinational Circuits", *1993 IEEE VLSI Test Symposium*, pp. 25-32.

[15] S. Chakravarty, P. J. Thadikaran, "A study of $I_{DDQ}$ Subset Selection Algorithms for Bridging Faults",*1994 International Test Conference*, pp. 403-412.

[16] S. Chakravarty, P. J. Thadikaran, "Simulation of $I_{DDQ}$ Tests for Bridging Faults in Combinational Circuits",*To appear in IEEE Trans. on Comput.*

[17] W. Debany, "Coverage of Node Shorts Using Internal Access and Equivalence Classes", *VLSI Design*, Vol. 1, No. 1, pp. 71-86, 1993.

[18] F. J. Ferguson, T. Larabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS ICs", *Intl. Test Conference*, pp. 492-499, 1991.

[19] R. R. Fritzemeier, J. M. Soden, R. K. Treece, C. F. Hawkins, "Increase CMOS IUC stuck-at fault coverage with reduced $I_{DDQ}$ test sets," *IEEE Int'l Test Conference*, pp. 427-435, 1990.

[20] R. Gupta, M. Breuer, "Ordering Storage Elements in a Single Scan Chain," *IEEE Int'l Conf. on Comput.-Aided Design*, 1991, pp. 408-411.

[21] Y. Gong, "Exploting Dynamic Computation to Diagnose Bridging Faults", *PhD Dessertation*, Dept. of Computer Science, State University of New York, Buffalo, NY.

[22] Y. Gong, S. Chakravarty, "Adaptive Diagnostic Test Generation," *IEEE/ACM Int'l Conf. on Comput.-Aided Design*, 1995.

[23] J. Galiay, Y. Crouzet, and M. Vergniault, "Physical Versus Logical Fault Models for MOS LSI Circuits: Impact on their testability," *IEEE Trans. on Comput.*, 29(6):, June 1980.

[24] J. M. Soden, C. F. Hawkins, R. K. Gulati and W. W. Mao, "$I_{DDQ}$ Testing: A Review", *Journal of Electronic Testing: Theory and Applications*, Vol. 3, No. 4, pp. 291-303, 1992. Also appears as Chapter 1 in $I_{DDQ}$ Testing of VLSI Circuits, Kluwer Academic Publishers.

[25] Y. Higami, S. Kajihara, K. Kinoshita, "Reduced ScanShift: A New Testing Methodology for Sequential Circuits", *IEEE Int'l Test Conference*, 1994, pp. 624-630.

[26] S. Hellebrand, B. Reeb, S. Tarnick, H. Wunderlich, "Pattern Generation for a Deterministic BIST Scheme", *International Conference on Computer-Aided Design*, pp. 88-94, 1995.

[27] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", *IEEE International Test Conference*, pp. 120-129, 1992.

[28] E. Isern, J. Figueras, "Test generation with covewrages for quiescent current test for bridging faults in combinational circuits", *Intl. Test Conference*, pp. 73-82, 1993.

[29] A. Jee, F. J. Ferguson, "CARAFE: An inductive fault analysis tool for CMOS VLSI Circuits", *IEEE VLSI Test Symposium*, pp. 92-98, 1993.

[30] W. Kunz, D. K. Pradhan, "Recursive learning: An attractive alternative to the decision tree for test generation in digital circuits", *IEEE Int'l Test Conf.*, pp. 816-825, 1992.

[31] R. Kapur, S. Patil, T. J. Snethen, T. W. Williams, "Design of an Efficient Weighted Random Pattern Generation System," *IEEE Int'l Test Conference*, 1994, pp. 491-500.

[32] T. Larabee, "Test pattern Generation using Boolean Satisfiability", *IEEE Trans. on Comput.-Aided Design*, Vol. 11, No. 1, pp. 4-15, January 1992.

[33] H. K. Lee, D. S. Ha, "On The Generation of Test Patterns for Combinational Circuits", Tech. Report No 12-93, Dept. of Elect. Eng., Virginia Polytechnique Institute and Sate University, Va.

[34] T. Lee, I. N. Hajj, E. M. Rudnick, J. H. Patel, "Genetic-Algorithm-Based Test Generation for Current Testing of Bridging Faults in CMOS VLSI Circuits", To appear in *IEEE VLSI Test Symposium*, 1996.

[35] S. Y. Lee and K. K. Saluja, "An Algorithm to Reduce Test Application Time in Full Scan Design," *IEEE Int'l Conference on Computer-Aided Design*, 1992, pp. 17-20.

[36] W. Maly, "Realistic Fault modelling for VLSI Testing", *IEEE/ACM Design Automation Conference*, pp. 173 - 180, 1987.

[37] U. Mahlstedt, J. Alt, M. Heinitz, "CURRENT: A Test Generation Systems for $I_{DDQ}$ Testing", *IEEE VLSI Test Symposium*, pp. 317-323, 1995.

[38] F. Muradali, V. K. Agrawal, B. Nadeau-Dostie, "A New Procedure for Weighted Random Built-In Self-Test," *IEEE Int'l Test Conference*, pp. 660-669, 1990.

[39] W. Mao, R. K. Gulati, D. K. Goel, M. D. Ciletti, "QUIETEST: A Quiescent Current Testing Methodology for Detecting Leakage Faults," *IEEE Int'l Conferenc on Computer-Aided Design*, pp. 280-283, 1990.

[40] U. Mahlstedt, T. Gruning, C. Ozcan, W. Daehn, "CONTEST: A Fast ATPG Tool for Very Large Combinational Circuits", *IEEE/ACM Int'l Conf. on Comput.-Aided Design*, pp. 222-225, 1990.

[41] S. P. Morley and R. A. Marlett, "Selectable Length Partial Scan: A Method to Reduce Vector Length," *IEEE Int'l Test Conference*, 1991, pp. 385-392.

[42] W. Maly, J. P. Shen and F. J. Ferguson, "Systematic Characterization of Physical Defects for Fault Analysis of MOS IC Cells," *IEEE Int'l Test Conference*, pp. 390-399, 1984.

[43] S. Pateras, J. Rajski, "Cube Contained Random Patterns and Their Application to the Testing of Synthesized Multi-Level Circuits," *IEEE Int'l Test Conference*, 1991, pp. 473 - 482.

[44] I. Pomerantz, S. M. Reddy, "3-Weight Pseudo Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Trans. on Comput.-Aided Design*, 12(7):1050-1058, July, 1993.

[45] D. K. Pradhan and J. Saxena, "A Design for Testability Scheme to Reduce Test Application Time," *IEEE VLSI Test Symposium*, pp. 55 - 60, April 1992.

[46] R. S. Reddy, I. Pomerantz, S. M. Reddy, S. Kajihara, "Compact Test Generation for Bridging Faults under $I_{DDQ}$ Testing", *IEEE VLSI Test Symposium*, pp. 310 - 316, 1995.

[47] C. Sechen, *VLSI Placement and Global Routing Using Simulated Annealing.* Kluwer Academic Publishers, 1988.

[48] M. H. Schulz, E. Trischler, T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System", *IEEE Trans. on Comput.-Aided Design*, Vol. 7, No. 1, pp. 126-137, January 1988.

| | MIPS | SPECint92 | SPECfp92 | SPECrateint92 | SPECratefp92 |
|---|---|---|---|---|---|
| SUN 4/670 | 28.50 | 23.90 | 21.80 | 567.00 | 517.00 |
| SUN 5/70 | 105.40 | 59.60 | 46.80 | 1414.00 | 1110.00 |
| HP9000/735 | | | | 1832.00 | 2950.00 |

Table 1: Performance Rating

| | MIPS | SPEC int92 | SPEC fp92 | SPEC rateint92 | SPEC ratefp92 | AVG Speedup |
|---|---|---|---|---|---|---|
| SUN5/SUN4 | 3.70 | 2.49 | 2.15 | 2.49 | 2.15 | 2.60 |
| HP9000/SUN4 | | | | 3.23 | 5.71 | 4.47 |
| HP9000/SUN5 | | | | 1.30 | 2.66 | 1.98 |

Table 2: Relative Speedups


[49] P. J. Thadikaran and S. Chakravarty, "Fast Algorithms for Computing $I_{DDQ}$ tests for Combinational Circuits," *IEEE International Symposium on VLSI Design*, 1996, pp. 103-106.

[50] P. Thadikaran, S. Chakravarty, "Improved algorithms for selecting $I_{DDQ}$ measurement points", in preperation.

[51] P. Thadikaran, S. Chakravarty, J. Patel, "Techniques for Simulating $I_{DDQ}$ Test for Bridging Faults in Sequential Circuits", *IEEE/ACM Fault-Tolerant Computing Symposium*, 1995.

[52] S. Venkataraman, J. Rajski, S. Hellebrand, S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers", *International Conference on Computer Aided Design*, pp. 572-577, 1993.

[53] H-J Wunderlich, "Self-test Using Unequiprobable Random Patterns," *IEEE/ACM Int'l Fault-Tolerant Computing Symposium*, 1987, pp. 258-263.

[54] H-J Wunderlich, "Multiple Distribution for Biased Random Test Patterns," *IEEE Int'l Test Conference*, 1988, pp. 236-244.

[55] J. Waicukauski, E. Lindbloom, "Fault Detection Effectiveness of Weighted Random Patterns," *IEEE Int'l Test Conference*, pp. 245-255, 1988.

| | | Extracted-BFs | | Metal-BFs | | Ratio | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Adj |
| | Vectors | Faults | Time | Faults | Time | Faults | Time | Time |
| S1238 | 349 | 18467 | 18.17 | 145530 | 5.52 | 7.88 | 0.30 | 1.22 |
| S1423 | 36 | 22246 | 4.45 | 279378 | 2.13 | 12.56 | 0.48 | 1.91 |
| S1494 | 469 | 27510 | 31.35 | 218130 | 10.83 | 7.93 | 0.35 | 1.38 |
| S1488 | 590 | 27314 | 43.68 | 222111 | 13.63 | 8.13 | 0.31 | 1.25 |
| S5378 | 408 | 65310 | 82.20 | 4477528 | 63.93 | 68.56 | 0.78 | 3.11 |
| S35932 | 86 | 493025 | 162.23 | 158909878 | 53.42 | 322.32 | 0.33 | 1.32 |

Table 3: Fault Simulation, Sequential Circuits, Metal-BF

| | | Extracted-BFs | | All-BFs | | Ratio | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Adj |
| | Vectors | Faults | Time | Faults | Time | Faults | Time | Time |
| S1238 | 349 | 18467 | 18.17 | 869221 | 49.75 | 47.07 | 2.74 | 10.95 |
| S1423 | 36 | 22246 | 4.45 | 1261666 | 8.67 | 56.71 | 19.48 | 7.79 |
| S1494 | 469 | 27510 | 31.35 | 1929630 | 103.95 | 70.14 | 3.32 | 13.26 |
| S1488 | 590 | 27314 | 43.68 | 1902225 | 121.83 | 69.64 | 2.79 | 11.16 |
| S5378 | 408 | 65310 | 82.20 | 10878780 | 587.85 | 166.57 | 7.15 | 28.61 |
| S35932 | 86 | 493025 | 162.23 | 620065720 | 5885.53 | 1257.68 | 36.28 | 145.12 |

Table 4: Fault Simulation, Sequential Circuits, All-BF

| | | Extracted BFs | | Metal BFs | | Ratio | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Adj |
| | Vectors | Faults | Time | Faults | Time | Faults | Time | Time |
| C1355 | 99 | 16282 | 6.96 | 171991 | 2.03 | 10.56 | 0.29 | 1.17 |
| C1908 | 146 | 22316 | 11.38 | 416328 | 5.20 | 18.66 | 0.46 | 1.83 |
| C2670 | 310 | 39130 | 36.33 | 1127251 | 19.26 | 28.81 | 0.53 | 2.12 |
| C3540 | 227 | 50106 | 37.20 | 1476621 | 14.96 | 29.47 | 0.40 | 1.61 |
| C5315 | 370 | 76944 | 90.75 | 3086370 | 34.68 | 40.11 | 0.38 | 1.53 |
| C6288 | 83 | 71232 | 23.57 | 2995128 | 7.18 | 42.05 | 0.30 | 1.22 |
| C7552 | 278 | 103195 | 88.87 | 6913621 | 40.83 | 67.00 | 0.46 | 1.84 |

Table 5: Fault Simulation, Combinational Circuits, Metal-BFs

| | | Extracted-BFs | | All-BFs | | Ratio | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Adj |
| | Vectors | Faults | Time | Faults | Time | Faults | Time | Time |
| C1355 | 99 | 16282 | 6.96 | 675703 | 26.62 | 41.50 | 3.82 | 15.30 |
| C1908 | 146 | 22316 | 11.38 | 1269621 | 22.38 | 56.89 | 1.97 | 7.87 |
| C2670 | 310 | 39130 | 36.33 | 3904615 | 136.48 | 99.79 | 3.76 | 15.03 |
| C3540 | 227 | 50106 | 37.20 | 6402831 | 98.60 | 127.79 | 2.65 | 10.60 |
| C5315 | 370 | 76944 | 90.75 | 15100260 | 863.03 | 196.25 | 9.51 | 38.04 |
| C6288 | 83 | 71232 | 23.57 | 12941328 | 112.50 | 181.68 | 4.77 | 19.09 |
| C7552 | 278 | 103195 | 88.87 | 27162135 | 435.22 | 263.21 | 4.90 | 19.59 |

Table 6: Fault Simulation, Combinational Circuits, All-BFs

| | METAL-BFs | | | | Extracted-BFs | | | | Ratio | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | Adj |
| Circuit | Faults | Tests | FC | Time | Faults | Tests | FC | Time | Faults | Time |
| C5315 | 3086370 | 36 | 99.95 | 14 | 101478 | 49 | 99.89 | 1.0 | 30.4 | 3.1 |
| C6288 | 2995128 | 28 | 99.99 | 13 | 110176 | 25 | 99.88 | 1.0 | 27.2 | 2.9 |
| C7552 | 6913621 | 51 | 99.95 | 25 | 138068 | 70 | 99.87 | 2.0 | 50.0 | 2.8 |
| S5378 | 4649725 | 96 | 99.93 | 28 | 91546 | 113 | 99.81 | 1.5 | 50.8 | 4.18 |
| S9234 | 17073246 | 162 | 99.85 | 145 | 163749 | 127 | 99.83 | 3.5 | 104.2 | 9.2 |
| S13207 | 37658181 | 243 | 99.94 | 501 | 232269 | 206 | 99.92 | 6.0 | 162.1 | 3.2 |
| S15850 | 53929305 | 194 | 99.93 | 276 | 279102 | 172 | 99.90 | 6.5 | 193.2 | 9.4 |
| S38417 | 286095160 | 257 | 99.98 | 1128 | 667346 | 250 | 99.96 | 19.5 | 428.7 | 13 |
| S38584 | 217747146 | 315 | 99.54 | 1674 | 716616 | 344 | 99.52 | 39 | 303.8 | 9.6 |

Table 7: Test Generation, Combinational, Metal-BFs. All times in CPU Mins.

| | All-BFs | | | | CURRENT | | | | Ratio | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | Adj |
| Circuit | Faults | Tests | FC | Time | Faults | Tests | FC | Time | Faults | Time |
| c5315 | 15100260 | 93 | 99.92 | 128 | 101478 | 49 | 99.89 | 1.0 | 148 | 29 |
| c6288 | 12941328 | 56 | 99.97 | 43 | 110176 | 25 | 99.88 | 1.0 | 117 | 10 |
| c7552 | 27162135 | 226 | 99.89 | 2022 | 138068 | 70 | 99.87 | 2.0 | 196 | 226 |
| s5378 | 11146281 | 372 | 99.95 | 258 | 91546 | 113 | 99.81 | 1.5 | 121 | 39 |
| s9234 | 46113606 | 548 | 99.84 | 860 | 163749 | 127 | 99.83 | 3.5 | 281 | 55 |
| s13207 | 91374921 | 548 | 99.95 | 5491 | 232269 | 206 | 99.92 | 6.0 | 393 | 204 |
| s15850 | 137572578 | 1004 | 99.95 | 3949 | 279102 | 172 | 99.90 | 6.5 | 492 | 136 |
| s38417 | 727692175 | 1038 | 99.99 | 6296 | 667346 | 250 | 99.96 | 19.5 | 1090 | 73 |

Table 8: Test Generation, Combinational, All-BFs. All times in CPU Mins.

| | Sequential(All) | | | | Combinational(Ext) | | | | Ratio | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | Adj |
| Circuit | Faults | Tests | FC | Time | Faults | Tests | FC | Time | Faults | Time |
| s5378 | 5380840 | 831 | 92.28 | 587 | 91546 | 113 | 99.81 | 1.5 | 58 | 20 |
| s9234 | 19099290 | 229 | 5.56 | 53 | 163749 | 127 | 99.83 | 3.5 | 116 | 8 |
| s13207 | 33215325 | 351 | 6.7 | 231 | 232269 | 206 | 99.92 | 6.0 | 143 | 19.25 |
| s15850 | 55540530 | 739 | 23.6 | 1700 | 279102 | 172 | 99.90 | 6.5 | 198 | 130 |
| s35932 | 509044278 | 172 | 99.52 | 1263 | 639758 | 23 | 99.72 | 9.0 | 796 | 90 |

Table 9: Test Generation, Combinational(Extracted), All-BFs(Sequential). All times in CPU Mins

| | Tests | METAL-BFs | | Extracted-BFs | | Ratio | |
|---|---|---|---|---|---|---|---|
| | | | | | | | Adj |
| Circuit | Tests | Faults | Time | Faults | Time | Faults | Time |
| C5315 | 87 | 3086370 | 5 | 101478 | 0.5 | 30.4 | 2.2 |
| C6288 | 50 | 2995128 | 2 | 110176 | 0.3 | 27.2 | 1.5 |
| C7552 | 131 | 6913621 | 11.5 | 138068 | 1.0 | 50.0 | 2.57 |
| S13207 | 383 | 37658181 | 297 | 232269 | 4.5 | 162.1 | 14.7 |
| S15850 | 317 | 53929305 | 246 | 279102 | 4.5 | 193.2 | 12.2 |
| S38417 | 480 | 286095160 | 1188 | 667346 | 15.0 | 428.7 | 17.7 |
| S5378 | 206 | 4649725 | 25 | 91546 | 1.0 | 50.8 | 5.6 |
| S9234 | 280 | 42628761 | 101 | 163749 | 2.1 | 104.2 | 10.7 |

Table 10: Test Compression, Combinational, METAL-BFs. All times in CPU Min.

| | Tests | All-BFs | | Extracted-BFs | | Ratio | |
|---|---|---|---|---|---|---|---|
| | | | | | | | Adj |
| Circuit | Tests | Faults | Time | Faults | Time | Faults | Time |
| C5315 | 87 | 15100260 | 43 | 101478 | 0.5 | 148 | 19 |
| c6288 | 50 | 12941328 | 7 | 110176 | 0.3 | 117 | 5.2 |
| c7552 | 131 | 27162135 | 145 | 138068 | 1.0 | 196 | 32.4 |
| s13207 | 383 | 91374921 | 1412 | 232269 | 4.5 | 393 | 70.2 |
| s15850 | 317 | 137572578 | 1457 | 279102 | 4.5 | 492 | 493 |
| s38417 | 480 | 727692175 | | 667346 | 15.0 | 1090 | |
| s5378 | 206 | 11146281 | 173.8 | 91546 | 1.0 | 121 | 38.8 |
| s9234 | 280 | 17073246 | 501.6 | 163749 | 2.1 | 281 | 53.4 |

Table 11: Test Compression, Combinational, All-BFs. All times in CPU Min.

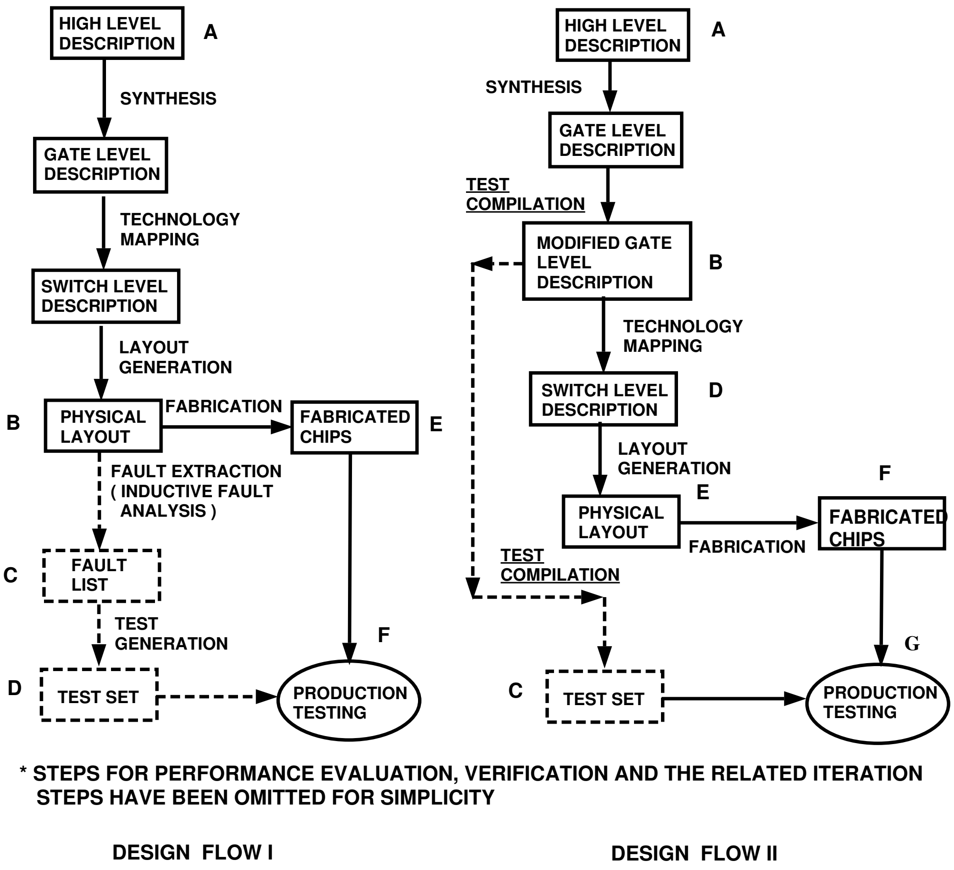


Figure 1: Design Flows

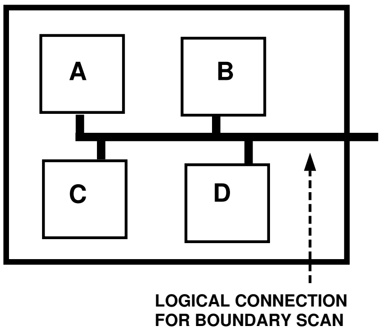


Figure 2: Board/MCM Assembly