WHY PARTIAL DESIGN VERIFICATION WORKS BETTER THAN IT SHOULD

Jacob Savir
International Business Machines Corporation
Data Systems Division
P. O. Box 950
Poughkeepsie, NY 12602 USA
(914) 433-5719

ABSTRACT

The problem of verifying the correctness of a combinatorial design is known to be NP complete. Nevertheless, most products reaching the consumer are functionally correct. This paper attempts to explain this phenomenon by sizing the effort of going through less than perfect design verification process and explaining why many design errors are relatively easily caught.

INTRODUCTION

The problem of verifying the correctness of a fully specified design is known to be equivalent to proving that a combinational function is a tautology [1]. Verifying correctness of incompletely specified designs is even harder.

Many algorithms have been proposed [2-5] that attempt to prove the correctness of a design. The experience with these algorithms is that in many cases they exemplify exponential complexity. Some workers tried to get around the complexity problem by using a family of algorithms, realizing that no single algorithm is likely to work efficiently on all designs.

No proposed algorithm will run efficiently on all designs because there are no "characteristic" design errors that people make. Having no model of characteristic design errors to work against, an exhaustive search to expose the design error is as good as any.

Notice that use of a family of algorithms to prove correctness of a design is nothing but a "semi-random" search, because there is no prior knowledge as to which algorithm from the family, if any, will work well on a given implementation.

Practical experience with design verification has been reported in [6]. The evolution of CAD tools in VLSI design has been reported in [7].

DESIGN VERIFICATION ANALYSIS

The problem can be formulated as follows. Let C_1 and C_2 be two designs of a combinational network. Are the two designs producing the same outputs for all inputs in the domain of the functions? In practice C_1 may be a model specifying the desired function and C_2 its actual implementation, or C_1 a computer generated design and C_2 a manually generated design. In the sequel we implicitly consider both interpretations of the problem, and rely on the reader to identify the appropriate case.

Let the functions C_1 and C_2 have n inputs and m outputs. In many cases there are many "unused input combinations" in digital systems design. For example, not all combinations in an operand code field of a computer are used to represent an instruction. Thus, the functions to be designed are very often ***incompletely specified***. The unused input combinations are referred to as ***don't cares***. The designers may assign whatever value they wish as a response to an unspecified input combination. Usually designers will assign such values to the unspecified inputs that will minimize their implementation cost. Since different designers may specify different response values to the unspecified inputs, the two designs C_1 and C_2 may both meet design specification and still not be identical. Thus two designs are considered to be a correct implementation of a desired function if and only if for every input in the domain of the functions the corresponding outputs are identical whenever specified. We refer to these designs as being ***functionally compatible***.

Let I be the set of input vectors for which at least one output is specified. Let N be the cardinality of this set, namely $N = |I| \leq 2^n$. Only elements of I can serve as useful verification patterns. The two designs C_1 and C_2 may respond differently to elements not in I and still be functionally compatible.

No assumption is made as to what class of design errors are possible. As a result, there is no algorithm short of exhaustive application of all members in I that can guarantee either the correctness of the design, or will otherwise expose its design error.

It is commonly believed that patterns designed for

manufacturing testing can serve as good design verification patterns. A simple example will dismiss this belief. Consider the following design specification:

Design a two input circuit F(A,B) such that F(A,B)=1 if and only if A=B=1.

Assume that the designer "thought" that the specification was:

Design a two input circuit F(A,B) such that F(A,B)=1 if and only if A=B.

As a result of the designer's wrong perception of the problem, the function $G=AB+\bar{A}\,\bar{B}$ was implemented instead of F=AB. The single stuck-fault test set for F is {11,01,10}. Unfortunately, the only pattern that can expose the design error is A=B=0 which is not a single stuck-fault test vector.

Some circuits have the property that none of the outputs depend on all input lines. In this special case it is possible to devise a test where all output functions are exhausted in parallel while the input space is not exhausted. This ***pseudoexhaustive test*** [8] can drastically reduce the design verification test time when applicable. Note that no such time saving is possible if at least one output function depends on all input lines.

N is quite large in most designs. The task of fully guaranteeing the correctness of a design is computationally prohibitive. The question asked is to what extent can one "guarantee" the correctness of a design without actually exhausting I. In order to answer this question let D be the ***discrepancy set*** defined as the set of all input vectors for which the response of design C_2 differs from the response of design C_1 whenever specified. Obviously D is a subset of I. Let the cardinality of D be $s=|D|$. Note that both C_1 and C_2 are functionally compatible if s=0.

The verification process used here is defined as randomly applying members of I and checking the corresponding outputs of C_1 and C_2 whenever specified. A discrepancy between at least two specified outputs constitutes a proof that the two designs are not functionally compatible, and the verification test can stop right there. If all specified outputs show identical values, a new vector can be drawn from I and applied to both C_1 and C_2. Random drawing from I is done without replacement. If "enough" vectors have been applied and no evidence is found that the two designs differ, then the verification process will declare C_1 and C_2 to be functionally compatible with a certain degree of confidence. The question is how long should that test be?

The hypergeometric waiting time distribution [9] may be used to answer the question posed above. Let p(N,s,t) be the probability that the t-th pattern drawn from I will prove that C_1 differs from C_2. Then,

$$(1)\qquad p(N,s,t) = \frac{\binom{N-t}{s-1}}{\binom{N}{s}}$$

Let P(N,s,T) be the distribution function of the random variable t, namely let

$$(2)\qquad P(N,s,T) = \sum_{t=1}^{T} p(N,s,t)$$

The meaning of P(N,s,T) is the probability that C_1 will be proven to be functionally incompatible with C_2 in at most T patterns. Using the relation [10]

$$(3)\qquad \sum_{k=0}^{n} \binom{k}{m} = \binom{n+1}{m+1},\ \text{integer } m,n \geq 0,$$

we get

$$(4)\qquad P(N,s,T) = 1 - \frac{\binom{N-T}{s}}{\binom{N}{s}}$$

The ***escape probability*** [11] is defined as the probability that the verification procedure will fail to expose the design error in T patterns. The escape probability $\phi(N,s,T)$ is therefore

$$(5)\qquad \phi(N,s,T) = \frac{\binom{N-T}{s}}{\binom{N}{s}}$$

Assume now that the designer can only make design errors with a limited discrepancy set sizes. Thus s is bounded, $1 \leq s \leq S$. The parameter S is, therefore, the maximum size error the designer is expected to make. Let $\bar{\phi}$ be the average escape probability the verification process is expected to undergo due to design errors with limited discrepancy set sizes. Assuming that all design errors with discrepancy set sizes $s \leq S$ are equally likely, the probability that the design being questioned has a discrepancy set size s is given by $\binom{N}{s}/\sum_{s=1}^{S}\binom{N}{s}$. The average escape probability of the verification process is therefore given by

$$(6)\quad \bar{\phi}(N,S,T) = \frac{\sum_{s=1}^{S} \binom{N-T}{s}}{\sum_{s=1}^{S} \binom{N}{s}}$$

The case of the inept designer

Consider a sloppy designer who is likely to make unlimited design errors. We refer to him as an inept designer because he is likely to make as many mistakes as possible if he is asked to design the circuit to specification. The average escape probability for this case can be calculated by substituting S=N in (6):

$$(7)\quad \bar{\phi}(N,N,T) = \frac{\sum_{s=1}^{N} \binom{N-T}{s}}{\sum_{s=1}^{N} \binom{N}{s}} = \frac{2^{N-T}-1}{2^{N}-1}, \quad T \le N$$

Since N is very large we have

$$(8)\quad \bar{\phi}(N,N,T) \approx 2^{-T}$$

The verification test length necessary to prove that the inept design is incorrect with an escape probability no larger than ε is approximately

$$(9)\quad T \approx -\log_2 \varepsilon$$

The results in (8) and (9) are not that surprising due to the fact that an inept designer has a fifty-fifty chance of making a design error in each and every member of I.

The case of the careful designer

Recalling that N>>T and N>>S, eq. (6) can be approximated to

$$(10)\quad \bar{\phi}(N,S,T) \approx \frac{\binom{N-T}{S}}{\binom{N}{S}} \approx \left(1 - \frac{T}{N}\right)^{S}$$

The verification test length necessary to achieve an average escape probability no larger than ε is therefore approximately

$$(11)\quad T \approx N(1 - \varepsilon^{1/S})$$

We define the ***verification coefficient*** τ as the ratio T/N for ε=.1, namely

$$(12)\quad \tau \approx 1 - (0.1)^{1/S}$$

The verification coefficient is a measure of how difficult it is to catch errors made by a careful designer (one who is not perfect but nearly so). Obviously, the larger the discrepancy set size, the easier it is to catch the error. The verification coefficient uses a standard escape probability of .1 for purpose of comparison (this is equivalent to verification test quality of 90 percent). Note that $0 \le \tau \le 1$, and that it determines how close to a full exhaustion of I the verification process has to get in order to achieve a confidence of 90 percent that the design is correct. Table 1 displays some values of τ as a function of S.

Unfortunately there are no available statistics on discrepancy set sizes and, therefore, it is impossible to determine what the verification test length should be. It is interesting, however, that most products reach the consumer market error-free. The previous analysis can still explain why most design errors are caught before the design is released to production. The reason is that most errors made by designers, regardless of how careful they are, are mapped into large discrepancy set sizes. Consider the following design requirement:

TABLE 1: Verification coefficients of various design errors

Discrepancy set size S	Verification coefficient τ
1	.9
2	.68
3	.53
4	.44
5	.37
10	.20
100	.023
1000	2.3×10^{-3}
10000	2.3×10^{-4}

Design an n-input function F that should be activated with an ENABLE signal E=1.

The designer incorrectly assumes that the requirement is:

Design an n-input function F that should be activated with an ENABLE signal E=0.

Thus, the designer implemented the function $\bar{E}F$ rather than EF. The discrepancy set size due to this design error is the number of minterms in the function F, which may be very large. An average n-input combinational function has 2^{n-1} minterms. This is just one example of a very simple error that any designer might make, and that will end up having a very large discrepancy set size.

It is important to mention, though, that there have been products that reached the consumer with design errors that were only detected after they were used in the field for quite sometime. Obviously, these are cases where the discrepancy set size was relatively small.

CONCLUSIONS

The parameter influencing the success of catching design errors in combinational circuits is the discrepancy set size. Even careful designers might make errors with large discrepancy set sizes. This is a fortunate situation, since it allows an early detection of the design error, and avoid releasing an erroneous design to production. On the other hand, erroneous designs with small discrepancy set sizes may end up being in the field for a long time before their fault is being exposed.

There is no algorithm short of exhausting the "usable input vectors" that will guarantee that a given design meets specification. Less than guaranteed verification of a good design is possible with much less patterns.

REFERENCES

[1] Ruey-sing Wey, and Alberto L. Sangiovanni-Vincentelli, "PPROTEUS: A logic verification system for combinational circuits," Proc. 1986 Int. Test Conf., pp. 350-359, Sept. 1986.

[2] Roth, J.P., Computer logic, testing, and verification, Computer Science Press, 1980.

[3] Roth, J.P., "VERIFY: An algorithm to verify a computer design," IBM Tech. Disc. Bulletin 15, 2646-2648(1973).

[4] Roth, J.P., "Hardware verification," IEEE Trans. Comput., Vol. C-26, pp. 1292-1294, Dec. 1977.

[5] Smith, G.L., Bahnsen, R.J., and H. Halliwell, "Boolean comparison of hardware and flowcharts," IBM J. Res. Develop., Vol. 26, pp.106-116, Jan. 1982.

[6] Jacobs, H., "Verification of a second-generation 32-bit microprocessor," Computer, pp.64-70, April 1986.

[7] Newton, A.R., and Alberto L. Sangiovanni-Vincentelli, "Computer-aided design for VLSI circuits," Computer, pp. 38-60, April, 1986.

[8] McCluskey, E.J., "Verification testing - a pseudoexhaustive test technique," IEEE Trans. Comput., Vol. C-33, pp.541-546, June 1984.

[9] Wilks, S.S., Mathematical statistics, John Wiley & Sons, 1963.

[10] Knuth, D.E., The art of computer programming, Vol. 1, Second edition, Addison-Wesley, 1975.

[11] Savir J., and P.H. Bardell, "On random pattern test length," IEEE Trans. Comput., Vol. C-33, pp. 467-474, June, 1984.