Even Distribution Evaluation in Random Stimulus Generation

Zhiqiu Kong, Shujun Deng, Jinian Bian, Yanni Zhao

Department of Computer Science and Technology, Tsinghua University, Beijing, China
kzq07@mails.tsinghua.edu.cn, dengsj04@mails.tsinghua.edu.cn,
bianjn@tsinghua.edu.cn, zhaoyn05@mails.tsinghua.edu.cn

Abstract

This paper has two contributions: First is to analyze the entropy evaluation for random stimulus generation in one paper of DATE 2008 [1]; second is to present better methods to evaluate the solutions' even distribution for random stimulus generation. An evaluation strategy called min-distance-sum takes the solution space as a ring and calculates the adjacent solutions' distances. Experimental results for SAT and circuit benchmarks showed that our method can evaluate the even degree better than entropy, and also prove that the XOR-based method can improve the distribution of solutions for random stimulus generation.

Keywords: Random stimulus generation, design verification, even distribution, SAT

1. Introduction

Functional verification has become the main bottleneck in the design process and design teams spend 50% to 70% of their time verifying designs. For large scale designs, formal verification is difficult, so simulation is still the main stream in industry. However, constrained random stimulus generation which is between formal verification and simulation is more and more important, especially during the move to high level design and verification [2, 3, 4, 5]. Most of constrained random stimulus generation methods are using BDD model and BDD random walks [3] where space explosion easily occurs. One substitute is to use SAT-based methods due to the rapid efficiency improvements of SAT solvers in the last decade [6, 7, 8]. However, SAT-based constrained randomization can not directly get even distribution [9]. The same problem occurs in random stimulus generation [10]. Fortunately, there have been some works using XOR constraints [1, 10, 11] to achieve even distribution.

In [1], authors presented a tool named Toggle to find out inadequately sensitized areas during simulation and automatically simulate those areas with uniform distributed solutions. In order to estimate the activity and biasing of solutions, the authors adopted entropy.

However, we found entropy is meaningless when evaluating the even degree of solutions. In this paper, we present better evaluation strategies based on min-distance-sum to guide the random stimulus generation. These strategies treat the solution space as a ring and calculate distances of adjacent solutions to ensure a most even distribution.

The remainder of this paper is organized as follows: Section 2 describes some backgrounds of achieving even distribution in random stimulus generation. Shortage of entropy in [1] is analyzed in Section 3. Our new evaluation strategy is presented in

Section 4. In Section 5, the experiments and comparisons are presented. Section 6 concludes the paper.

2. Background

The framework of random stimulus generation is shown in Figure 1. Vectors generated by random generator are sent to a design under test (DUT), then a monitor checks the results according to expected results and calculate the coverage.

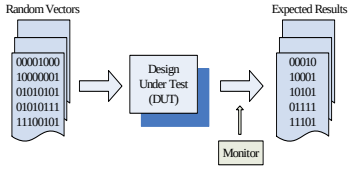


Figure 1: Test framework of random stimulus generation

Usually, the constraints in random stimulus generation can be classified into two types: environment constraints and feature ones. The former includes the basic constraints of gates while the latter defines the desired features [3]. So the solution space for random stimulus generation can be illustrated in Figure 2. The big circle is the total solution space (2^n) for a design with n variables. The small items (squares and circles) are the solution space under environment constraints, and the squares stand for the solution space under both environment and feature constraints.

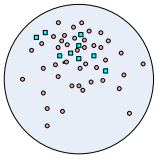


Figure 2: Solution spaces with different constraints

Usually, the solution space for random stimulus generation (with environment and feature constraints) is not evenly distributed. BDD based method can easily get even distributed solutions by enumerating all the solutions and get branching probabilities of leading to solutions on every node. However, SAT solvers are usually used to find out one solution even if there are ALL-SAT techniques [12]. Essentially, ALL-SAT techniques get all the solutions by enumerating all satisfying assignments using a SAT solver and it is not practical for large circuit design problems. In order to achieve an even distribution using SAT solvers without the understanding of the total solution space, some works [1, 10, 11] applied XOR constraints.

Plaza et al. [1] emphasized an aspect of this theory: Adding a random XOR constraint into a SAT problem can reduce the solution space into halves with high probability. In [1], the constraint $v*w_1 = 0$ in base-2 arithmetic was expressed by an XOR constraint as following:

$$x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_j} \oplus 1 \quad (1)$$

According to the translation method introduced in [13], the constraint above can be translated into CNF:

$$(y_1 \Leftrightarrow x_{i_1} \oplus x_{i_2}) \wedge (y_2 \Leftrightarrow y_1 \oplus x_{i_3}) \wedge \ldots \wedge (y_{j-1} \Leftrightarrow y_{j-2} \oplus x_{i_j}) \wedge (y_{j-1} \oplus 1) \quad (2)$$

or

$$(x_{i_1} \oplus y_1) \wedge (y_1 \Leftrightarrow x_{i_2} \oplus y_2) \wedge \ldots \wedge (y_{j-2} \Leftrightarrow x_{i_{j-1}} \oplus y_{j-1}) \wedge (y_{j-1} \Leftrightarrow x_{i_j} \oplus 1) \quad (3)$$

Then the XOR constraints can be translated into CNF as following:

$$\begin{aligned} & z = x \oplus y \\ & \Leftrightarrow (z \rightarrow x \oplus y) \cdot (x \oplus y \rightarrow z) \\ & \Leftrightarrow (\bar{z} + x \cdot \bar{y} + \bar{x} \cdot y) \cdot \left[\overline{(x \cdot \bar{y} + \bar{x} \cdot y)} + z \right] \\ & \Leftrightarrow (\bar{x} + y + z) \cdot (x + \bar{y} + z) \cdot (x + y + \bar{z}) \cdot (\bar{x} + \bar{y} + \bar{z}) \end{aligned} \quad (4)$$

And

$$x \oplus y \Leftrightarrow (x \cdot \bar{y} + \bar{x} \cdot y) \Leftrightarrow (\bar{x} + y) \cdot (x + \bar{y}) \quad (5)$$

3. Analysis of Entropy Evaluation

In Section 3.2 of [1], in order to estimate cuts activities and biasing, authors applied the following formula to compute

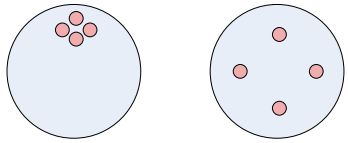
the entropy (we normalize it by dividing a constant $\log_2 K$) of a set of signals F.

$$E_F^K = \frac{-\sum_{vec:occ(vec)\neq 0} \frac{occ(vec)}{K} \log_2 (\frac{occ(vec)}{K})}{\log_2 K} \quad (6)$$

Where K is the number of the stimuli, and occ(vec) is the number of occurrences of the signal vector vec.

This formula is quite good to evaluate the reduplication degree of solutions used by Toggle to analyze the activities of each portion. Assume there are K simulation vectors, and the number of bits associated with the cut of one portions is n. Entropy formula (6) gets its maximal value when every vector is unique.

However, entropy cannot distinguish the even degree of solutions. For example, The Entropies of the two distributions in Figure 3 are both maximum 1.



(a) Bad distribution (b) Even distribution

Figure 3: Different solutions

Obviously the second one is more even since the solutions in the first one assemble. But entropy can not identify this difference.

Furthermore, In the algorithm of Figure 4 in [8], the authors used "add_vector(solution)" to prevent getting the same solution, so the algorithm can ensure that the entropy is 1. Under this circumstance, entropy is inadequate to evaluate the solutions' distribution.

4. Even Distribution Evaluation

4.1. Min-Distance-Sum Formula

A better evaluation formula named min-distance-sum for even degree is as following:

$$D = \frac{\sum_{i=0}^{k-1} \left| \frac{2^n}{k} - \Delta_i \right|}{\frac{k-1}{k} 2^{n+1}} \quad (7)$$

$$\Delta_i = \begin{cases} S_i - S_{i-1}, & i = 1, 2, ..., k-1 \\ S_0 + 2^n - S_{k-1}, & i = 0 \end{cases}$$

Where n is the bit-width of the cut in design portions, and k is the number of random generated stimuli. Δ_i is the distance between the adjacent solutions.

In fact, the total solution space of a SAT problem with n variables is 2^n, from 0 to 2^n-1. Let these possible solutions be distributed on a ring like Figure 4. So $2^n/k$ is the average distance between solutions. In order to evaluate the distribution of solutions, we use the absolute difference from average distance. This formula increases with the absolute distance sum, so we call it min-distance-sum.

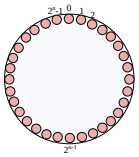


Figure 4: Solutions on a ring

In [9], the authors presented the best case of distributing *k* elements evenly among *N-solutions* space when the remainder of *N/k* is *r*. At the bottom level, we distribute *k* solutions into *N* spaces, the best case of distances is *k-r m* and *r m*+1 where m is $\lfloor N/k \rfloor$; then at the second bottom level, we distribute *r* elements evenly among *k* space, i.e. *r* is the new *k*, the *k* is the new *N*. This process continues till the top level where the remainder is 0.

The best case and the worst case of formula (7) are presented in theorem 1, in which the worst case is a little different from [9].

Theorem 1. The maximum value of D in formula (7) is 1 iff all the solutions are the same. If 2^n can be divided by k, the

minimal value of D is 0 iff the solutions are distributed evenly with the average distance $2^n/k$.

The proof is omitted owing to space constraints.

4.2. Improved Method

Formula (7) is adequate to estimate the even distributing differences between solutions. But for the randomization of stimulus generation, the differences may be quite trivial. For example, both the values of formula (7) for solutions (n=1000, k=4)

$$\underbrace{\begin{cases}1111.....1111101000\\1111.....1111101001\\1111.....1111111010\\1111.....1111111011\end{cases}}_{1000}$$

and

$$\underbrace{\begin{cases}1111.....1111101000\\1111.....1111101100\\1111.....1111111000\\1111.....1111111100\end{cases}}_{1000}$$

are

$$D=\frac{\sum_{i=0}^{k-1}\left|\frac{2^n}{k}-\Delta_i\right|}{\frac{k-1}{k}2^{n+1}}=\frac{\sum_{i=0}^{3}\left|\frac{2^{1000}}{4}-\Delta_i\right|}{\frac{3}{4}2^{1001}}\approx\frac{2}{3}\ (\Delta_i<<\frac{2^n}{k})$$

We proposed an extended form for formula (7) which ignores the same bits and compresses *s* bits combinations. The solutions in the above case can be divided into 3 parts:

$$\begin{cases}1111.....1111101000\\1111.....1111101100\\1111.....1111111000\\\underbrace{1111.....1111}_{x}\underbrace{11}_{s}\underbrace{00}_{y}\end{cases}$$

We ignore the left x bits because they are the same. The following *s* bits are only 01 and 11, and if both 00 and 10 are UNSAT, we can compress 01, 11 to 0 and 1. So the above solutions can be compressed into:

$$\begin{cases}0000\\0001\\1010\\1011\end{cases}\ \text{and}\ \begin{cases}0000\\0100\\1000\\1100\end{cases}$$

The values of formula (7) are 0.5 and 0. Obviously, he second group is better.

5. Experimental Results

We use the method similar to the one mentioned in [1] to generate random stimuli. The SAT solver is Minisat 2 [8]. All the experiments were completed on an Intel Xeon 3GHz CPU with 6G RAM. The benchmarks are from SATLIB [15] and ISCAS89 [16]. Each benchmark was run 20 times independently. The test-cases we chose are those whose solution space $|S_c| > k$, where k is 8 and 32. Consequently, all the entropy values calculated using the formula in [1] are 1.

The results for the SAT benchmarks from SATLIB are listed in Table 1and 2. The first column presents the benchmark names. The second and the third columns illustrate the number of variables and clauses. The fourth and fifth columns show the processing time and the evaluation value using the formula (7) for direct random and XOR-based simulation. According to formula (7), the less the evaluation value, the better the even distribution.

Table 1: Test results for SATLIB (k = 8)

Test-case	#Var	#Cls	Dir. Rand		XOR-based	
			T(s)	E	T(s)	E
2bitcomp_5	125	310	0.016	1.00	0.017	**0.72**
2bitmax_6	252	766	0.022	1.00	0.022	**0.80**
CBS_k3_n100_m403_b10_500	100	403	0.029	1.00	0.039	**0.65**
CBS_k3_n100_m403_b10_999	100	403	0.064	0.85	0.063	**0.60**
flat30-50	90	300	0.018	0.86	0.023	**0.71**
flat30-100	90	300	0.019	0.86	0.021	**0.71**
RTI_k3_n100_m429_100	100	429	0.063	1.00	0.097	**0.71**
RTI_k3_n100_m429_300	100	429	0.051	0.57	0.047	**0.48**

Table 2: Test results for SATLIB (*k* = 32)

Test-case	#Var	#Cls	Dir. Rand		XOR-based	
			T(s)	E	T(s)	E
2bitcomp_5	125	310	0.078	1.00	0.073	**0.70**
2bitmax_6	252	766	0.105	1.00	0.108	**0.73**
CBS_k3_n100_m403_b10_500	100	403	0.168	1.00	0.274	**0.71**
CBS_k3_n100_m403_b10_999	100	403	0.267	0.96	0.408	**0.69**
flat30-50	90	300	0.089	0.87	0.151	**0.79**
flat30-100	90	300	0.092	0.96	0.130	**0.79**
RTI_k3_n100_m429_100	100	429	0.254	0.94	0.727	**0.81**
RTI_k3_n100_m429_300	100	429	0.222	0.71	0.192	**0.56**

In Table 1 and 2, no benchmarks can be seen getting less evaluation via direct random method than XOR-based method. We can also see that the evaluation values are irrelative to k. This experiment showed that the XOR-based can get better even distribution than direct random for SAT benchmarks. On average, the run time of the XOR-based method is a bit more than the direct random one since there're more constraints to handle.

Similarly, the test results for ISACS89 circuit benchmarks with expanded time-frames are presented in Table 3 (k=8) and Table 4 (k=32). The constraints in our benchmarks include the basic constraints of gates and some user specified constraints which were added randomly. The first column is the test-case name while the second column shows the expanded time-frames. A point to be noted here, only the primary inputs need to be considered when adding XOR constraints as the solution space of a circuit is totally controlled by them.

Table 3: Test results for ISCAS89 (*k* = 8)

Circuit	#Exp_T	Dir. Rand		XOR-based	
		T(s)	E	T(s)	E
S27	70	0.037	**0.64**	0.036	0.89
S298	8	0.054	**0.57**	0.050	0.57
S344	5	0.038	1.00	0.036	**0.78**
S349	5	0.043	1.00	0.038	**0.57**
S382	6	0.046	0.88	0.043	**0.66**
S386	6	0.049	0.86	0.046	**0.58**
S1196	1	0.069	1.00	0.063	**0.63**
S1238	1	0.030	1.00	0.029	**0.65**

Table 4: Test results for ISCAS89 (*k* = 32)

Circuit	#Exp_T	Dir. Rand		XOR-based	
		T(s)	E	T(s)	E
S27	70	0.247	**0.76**	0.204	0.89
S298	8	0.341	0.84	0.263	**0.63**
S344	5	0.244	1.00	0.198	**0.76**
S349	5	0.266	0.99	0.202	**0.61**
S382	6	0.304	0.89	0.232	**0.71**
S386	6	0.327	0.77	0.239	**0.59**
S1196	1	0.173	1.00	0.151	**0.71**
S1238	1	0.165	1.00	0.145	**0.74**

For most of the cases in Table 3 and 4, the average CPU time and evaluation of XOR-based method are better than direct random one, especially for the large circuit test-cases. The direct method is better than XOR-based for the S27 test-case. That is because the S27 circuit had been expanded for 70 time-frames, too much reduplication led to a quite tortuous solution space. Consequently, XOR-based cannot find evenly distributed solutions easily. On average, the run time of the XOR method is less. We think the reason is that the XOR constraints added with primary inputs can improve the constraint propagation in SAT solving.

This experiment confirms that XOR-based method can get evenly distributed solutions in stimulus generation and the *min-distance-sum* formula can distinguish these differences adequately.

6. Conclusions

We have analyzed the random stimulus generation method using the entropy and XOR constraints in [1] and presented a better evaluation method named *min-distance-sum* to estimate the solutions' even distribution. We have proved this method in theory and by experiments and got the conclusion that our method is better to estimate the evenness of distributions and can considerably improve the stimulus generation guide in practice.

The future works are to apply this method to the practical circuit stimulus generation. Firstly, we need to find out the inactive part in test using our method. Secondly, we can guide the test stimulus generation with the *min-distance-sum* strategy.

7. Acknowledgments

This work was funded in part by the National Grand Fundamental Research 973 Program of China under Grant No.2005CB321605 and the National Natural Science Foundation of China under Grant No.90607001.

8. References

[1] Stephen M. Plaza, Igor L. Markov, and Valeria Bertacco, "Random Stimulus Generation using Entropy and XOR Constraints", in Proc. of the conference on Design, Automation and Test in Europe, 2008. pp. 664-669.

[2] "Constrained-random test generation and functional coverage with Vera", Technical report, Synopsys, Inc, Feb, 2003.

[3] Jun Yuan, Carl Pixley, Adnan Aziz. Constraint-Based Verification. Springer US, Jan. 2006.

[4] Leena Singh, Leonard Drucker. Advanced Verification Techniques: A SystemC Based Approach for Successful Tapeout. Springer US, 2004.

[5] SystemC Verification Working Group. "Systemc verification standard specification". OSCI website: http://www.systemc.org, May. 2003.

[6] J.P. Marques-Silva and K.A. Sakallah, "GRASP—A New Search Algorithm for Satisfiability," in Proc. of ICCAD'96. 1997, 220-227.

[7] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an Efficient SAT Solver," in Proc. of DAC'01. 2001, 530-535.

[8] N. Eén and N. Sörensson, "An extensible SAT solver," in Proc. of International Conference on Theory and Applications of Satisfiability Testing. 2003, 502-518.

[9] A. J. Compton, "An Algorithm for the Even Distribution of Entities in One Dimension". The Computer Journal, 1985 28(5). pp. 530-537.

[10] Carla P. Gomes, Ashish Sabharwal, Bart Selman, "Near-Uniform Sampling of Combinatorial Spaces Using XOR Constraints" in Proc. of the 20th Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, Dec 2006. pp. 481-488.

[11] Carla P. Gomes, Willem-Jan van Hoeve, Ashish Sabharwal, Bart Selman, "Counting CSP Solutions Using Generalized XOR Constraints" in Proc. of the 22nd Conference on Artificial Intelligence, Vancouver, BC, Canada, July 2007. pp.204-209.

[12] B. Li, M. S. Hsiao, and S. Sheng, "A novel SAT all-solutions for efficient preimage computation" in Proc. of the Conference on Design, Automation and Test in Europe, March 2004. pp.380–384.

[13] T.Larrabee, "Test pattern generation using Boolean satisfiability" IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Volume 11, Jan. 1992, pp. 4-15.

[14] http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

[15] http://www.ece.vt.edu/mhsiao/iscas89.html