Random Stimulus Generation using Entropy and XOR Constraints

Stephen M. Plaza, Igor L. Markov, Valeria Bertacco
EECS Department, University of Michigan, Ann Arbor, MI 48109-2121
{splaza, imarkov, valeria}@umich.edu

Abstract

Despite the growing research effort in formal verification, constraint-based random simulation remains an integral part of design validation, especially for large design components where formal techniques do not scale. However, stimulating important aspects of a design to uncover bugs often requires the construction of complex constraints to guide stimulus generation. We propose Toggle, a stimulus generation engine, which features (1) an entropy-based coverage analysis to efficiently find portions of the design inadequately sensitized by simulation and (2) a novel strategy to automatically stimulate these portions through a specialized SAT algorithm that uses small randomized XOR constraints. As our experimental results demonstrate, Toggle requires minimal input from the verification engineer, and significantly improves the coverage qualities of the generated stimuli when compared to plain random simulation.

1 Introduction

Today, verification costs are increasing rapidly due to the shrinking time-to-market and growing design complexity, which also limits usage of inherently unscalable formal verification tools. However, random simulation enables partial validation of large designs even in situations when the error conditions are sequentially deep. Despite this, determining when the validation effort has provided adequate verification coverage is an ongoing challenge. Industrial verification often relies on a constraint-based random simulation methodology [1], where a set of constraints limits and controls the input combinations that are sent to the design. Note, however, that the use of constraints can create important challenges. On the one hand, it requires the verification team to have a thorough knowledge of the design. On the other hand, it assumes powerful algorithms to identify high-quality input stimuli that satisfy given constraints. The latter concern is partially addressed in [14], with constraints modeled as BDDs and simulation vectors obtained through a random walk of the BDD. However, that approach still requires complex constraint specifications and is limited in the constraint complexity it could handle due to its dependency on the BDD size.

In this work, we introduce *Toggle*, a novel solution for quantifying and improving simulation coverage, while at the same time, decreasing the engineering effort. Toggle provides an algorithmic solution that can evenly stimulate each internal region of a design, hence boosting the gate-level toggle coverage. Our contribution is based on an efficient and novel entropy-based coverage analysis, which highlights the regions of the design that experience low coverage. We then target those regions through a SAT-based algorithm and create stimuli thoroughly exercising them.

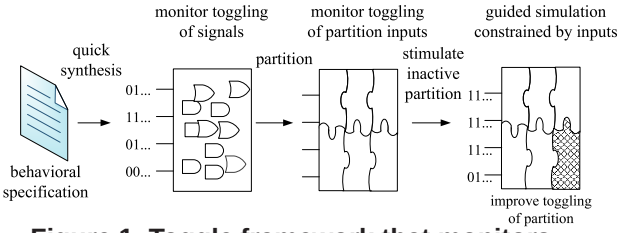

Figure 1. Toggle framework that monitors coverage and evenly stimulates the design.

The high-level flow of Toggle is illustrated in Figure 1. First, we perform low-effort synthesis on the behavioral specification, so as to leverage efficient gate-level tools available. We introduce an entropy-based coverage metric to analyze the toggle activity of each signal to guide netlist partitioning. This partitioning allows us to efficiently capture signal correlations within groups of internal wires. Our analysis determines which partitions should be stimulated a greater amount to achieve an even distribution of signal activity throughout the design and expose corner-case behavior. To stimulate a design uniformly, we generate small random XOR constraints that involve inadequately stimulated signals and derive stimuli with a SAT solver. Because the XOR constraints necessary for evenly sensitizing a design are small with respect to the size of the design, our SAT engine experiences little performance degradation, compared to techniques where large XOR constraints are added to a SAT instance. Thus, our technique is flexible in that it can evenly sensitize parts of the design while satisfying additional user-specified constraints.

We apply our analysis to commonly-used benchmark designs and demonstrate that many of them experience very low toggle coverage under random simulation. However, our technique achieves higher simulation coverage, since we can evenly sensitize a design without requiring user-specified input constraints. Toggle is also orders of magnitude faster compared to a directed simulation approach.

In Section 2, we review previous work in simulation and

refinement and constrained random simulation. In Section 3, we propose our solution for monitoring activity in a design using entropy. In Section 4, we propose our strategy to re-simulate areas of a circuit to increase its toggling activity. Finally, experimental results comparing Toggle to constrained-random simulation are shown in Section 5.

2 Previous Work

Successful verification methodologies [1, 2] often require detailed knowledge of a design and significant design-specific effort. To reduce the demands on the engineering team, automatic stimuli refinement can be used, often leading to improvement in the verification coverage. For example, at the instruction-level, Markov models can be used [13] to produce instruction sequences that effectively stimulate certain parts of the design. However, explicit monitors are necessary to guide this refinement, which still requires detailed knowledge of the design. At the gate-level, simulation can also be refined [11] to help distinguish nodes, but this is primarily useful for equivalence checking. In Toggle, we automatically identify parts of a circuit inadequately stimulated and add constraints to guide stimulus generation to expose corner-case behavior.

Deriving an even distribution of simulation vectors that satisfy complex constraints is challenging because state-of-the-art SAT solvers do not provide any guarantees on the distribution of solutions generated. Using BDDs [14] may require prohibitive amounts of memory. We observe that these challenges can be addressed using techniques developed in the AI community where a SAT solver evenly samples the solution space [9, 3]. However, the SAT engines they use are inefficient on EDA instances, and DPLL-based solvers are incompatible with their techniques. This limitation is partially addressed in [7], which uses randomly generated XOR constraints to modify the SAT instance so any SAT solver can sample its solution space more evenly. At first sight, these techniques are not directly applicable to IC verification since we desire to derive simulation vectors that expose corner-case behavior in a circuit, but our work provides several missing links to make this connection.

3 Finding Inactive Parts of a Circuit

In this section, we adapt the notion of Shannon's entropy to estimate verification coverage within a gate-level circuit and propose its use to find inadequately stimulated regions.

3.1 Toggle Activity of a Signal

The toggle coverage for a particular signal s in a circuit C evaluates the distribution of 0s and 1s seen under input stimuli. Capturing this distribution with two frequency values, we recall that Shannon's entropy E_s estimates the uncertainty of the signal:

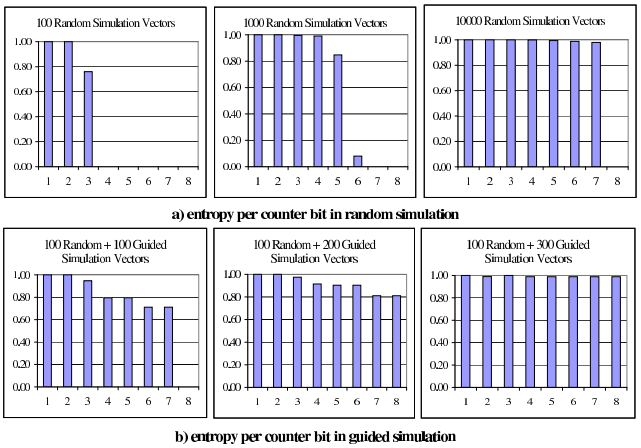

Figure 2. The entropy of each bit for an 8-bit bidirectional counter after 100 random simulation vectors are initially applied. After only 300 guided simulation vectors, the entropy is almost evenly distributed.

$$E_s = -\frac{nOnes}{K}\log_2(\frac{nOnes}{K}) - \frac{nZeroes}{K}\log_2(\frac{nZeroes}{K}) \quad (1)$$

where *nOnes* is the number of times that $s = 1$ and K is the number of simulation vectors examined. The formula gives values that range from 0 to 1 where higher entropy indicates greater activity, *i.e.*, an even distribution of ones and zeroes. We observe that the signal entropy can be trivially increased by asserting the signal to a desired value and deriving satisfying input stimuli with a SAT solver.

As a practical example of guiding simulation with signal entropy, consider the impact of random simulation on an 8-bit bidirectional counter, as shown in Figure 2a. The results indicate that after many simulation vectors, random stimuli do not adequately toggle the most significant bit. We target the bit with the smallest entropy by deriving a sequence of counter operations that toggles this bit. Figure 2b shows that the techniques described in this paper achieve an even distribution of entropy after only 300 simulation vectors. However, analysis of single wires does not account for correlations between multiple bits.

3.2 Toggle Activity of Multiple Bits

We extend signal entropy to a set of signals that experience low activity when correlated to each other. We identify these sets of signals as small cuts in the circuit (through automatic netlist partitioning) with respect to entropy-based net weights. We then define a coverage metric to assess the activity along the partition inputs that accounts for signal correlation. This allows us to explore activity at a finer granularity than just at the primary inputs or latches.

Automatic circuit partitioning: Circuit partitioning has been explored in physical placement applications where

net-cut minimization generally leads to smaller wirelength. The Fiduccia-Mattheyses (FM) min-cut partitioning algorithm [5] is commonly used and runs in linear time per pass. Furthermore, multi-level extensions of this algorithm scale near-linearly to very large designs. In our simulation flow, we only partition the design once and hence the runtime is amortized by verification costs, making it negligible.

We perform recursive bisection, *i.e.*, make multiple cuts until the circuit is partitioned to a desired granularity, either specified by the user or dynamically derived. The goal of this procedure is to minimize the total number of connections between partitions while ensuring that partitions have similar sizes. This objective leads to the generation of large partitions with few input signals, which allows us to examine only a few signals that control a large section of logic.

To identify input cuts that experience low activity, we use the signal entropy defined in Equation 1 to guide the partitioning objective function. We note that the entropy E_F of a set of signals F is upper-bounded by: $\sum_{s\in F} E_s$. After assigning E_s as a net weight to net s, we can employ netlist partitioning to find cuts with small entropy. This creates partitions with inputs of smaller accumulated entropy and exposes parts of the circuit that are inadequately sensitized.

Estimating cut activity and biasing through entropy: The cuts can be analyzed for activity to assess the amount of coverage in each partition. Consider the following metric for cut activity:

$$A_F^c = num\_diff\_vecs(< F_1, \cdots, F_m >) \quad (2)$$

where `num_diff_vecs` is the number of different simulation patterns on partition F's m-input cut. We observe that this formula does not consider the frequency of certain simulation vectors — it only provides the number of different simulation vectors. For example, when considering two different partitions together, we prefer to have an even distribution of simulation vectors for each partition while maximizing the number of different vectors and minimizing correlations in vectors seen at each partition.

To improve this coverage metric, we can also measure the amount of information (entropy) associated with the signals along a partition's inputs under K simulation vectors. This measure accounts for multiple observed simulation vectors along the cut and captures simulation bias. We compute the entropy of F as:

$$E_F^K = -\sum_{vec:occ(vec)\neq 0} \frac{occ(vec)}{K} \log_2\left(\frac{occ(vec)}{K}\right) \quad (3)$$

where *occ* is the number of occurrences of a particular vector *vec* along the input cut, which is represented by an integer value. In this formulation, the entropy is high when there is an even distribution across different simulation vectors. This measure is effective for most of the partition cuts that we examine because the number of possible vectors along the cut is much larger than the number of simulation vectors applied.

4 Targeted Re-simulation

Toggle uses the entropy measure previously described to find parts of the design with low activity. We now introduce a SAT-based strategy that uses random XOR constraints to produce an even distribution of simulation vectors along a partition cut with low activity. The motivation for producing an even distribution is to find corner-case behavior, which couldn't be exposed previously without detailed knowledge of the design and the generation of complex constraints.

4.1 Random Simulation with SAT

To evenly simulate a design, we first introduce the theoretical underpinnings of our approach, and then we propose an efficient approach that applies these concepts on a circuit while satisfying any additional specified constraints.

Theoretical background: Consider a SAT instance with $N > 1$ solutions. According to [12], it can be reduced to an instance that admits only one of those N solutions. The algorithm is randomized and adds a limited number of XOR constraints. It succeeds with probability $\geq 1/4$. Below, we discuss an aspect of this result relevant to our work, which states that adding a random XOR constraint reduces the solution space roughly in half with high probability.

Assume a SAT instance f with variables $x_1, x_2, ..., x_n$ that has solutions $v \in \{0,1\}^n$. To reduce the solution space, we randomly pick an assignment $w_1 \in \{0,1\}^n$ and add the following constraint to f: $v \bullet w_1 = 0$ in base-2 arithmetic. This can be expressed as follows:

$$f \wedge (x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_j} \oplus 1) \quad (4)$$

where i_j represents the indices of x_i where w_1 is 1. This results in an XOR constraint whereby an even polarity of x_{i_j} determined by w_1 need to be assigned to 1. Alternatively, a CNF representation can be given as:

$$f \wedge (y_1 \Leftrightarrow x_{i_1} \oplus x_{i_2}) \wedge (y_2 \Leftrightarrow y_1 \oplus x_{i_3}) \wedge \cdots \wedge (y_{j-1} \Leftrightarrow y_{j-2} \oplus x_{i_j}) \wedge (y_{j-1} \oplus 1) \quad (5)$$

Example 1 *Consider the SAT CNF* $(a+b+c')(b'+d)(a'+d')$. *This solution space* $\{0001, 0101, 0111, 1000, 1010\}$ *can be reduced by generating an XOR clause for the randomly generated* $w_1 : a = 0, b = 1, c = 1, d = 0$. *The resulting CNF would be* $(a+b+c')(b+d')(a'+d)(y \Leftrightarrow b \oplus c)(y \oplus 1)$ *where only* 3 *solutions remain* $\{0001, 0111, 1000\}$.

If S_F is the set of all solutions of F, then the addition of constraints from k random w_k vectors probabilistically reduces the solution space to $\sim 2^{-k}|S_F|$.

Random simulation with SAT: Consequently, through XOR-based reductions to U-SAT, any particular solution can be generated, ignoring the all-0s solution, which is the basis for our approach for deriving an even distribution of simulation vectors. Based on the results in [12], we can estimate that adding n XOR constraints for a circuit C with n inputs will produce a randomized U-SAT instance. We can

add multiple sets of n different XOR constraints to derive an even distribution of U-SAT instances. Since the solution space of the circuit is controlled by the primary inputs of the circuit, adding larger XOR constraints involving internal signals is unnecessary, and any SAT solver can be used on the modified SAT instance in principle.

While our approach does not always produce instances with a unique solution, according to [12], we expect the generation of U-SAT instances. Using a SAT solver, we can derive an even distribution of simulation vectors as shown in Section 5. However, if, for example, only 64 evenly distributed input vectors are desired for circuit C where $2^n > 64$, a more efficient procedure can be used that requires the addition of fewer constraints and minimizes the number of unsatisfiable instances produced. In this case, 6 XOR constraints can be added to approximately reduce the solution space to $\frac{1}{64}$ of the original size. By adding different random sets of 6 XOR constraints 64 times, we can still achieve an even distribution of solutions for the number of solution vectors desired, with faster simulation runtimes as shown in Section 5. In general, if we seek K simulation vectors, we solve K SAT instances each with different sets of $log_2(K)$ XOR constraints.

The addition of engineer-specified constraints does not affect the XOR formulation previously described. Therefore, an even distribution of simulation vectors can be derived that satisfies them. Consider a circuit C with $|S_C|$ solutions and a constrained circuit C^* with $|S_{C^*}|$ solutions. When $|S_{C^*}| << |S_C|$, random solutions S_{C_i} may rarely exist in S_{C^*} as illustrated in Figure 3a. By adding $log_2(K)$ XOR constraints, we can derive K vectors $S_{C_i^*}$ that are evenly distributed. If numerous UNSAT instances occur, implying that $K > |S_{C^*}|$, then one can alternatively exhaustively enumerate all the solutions.

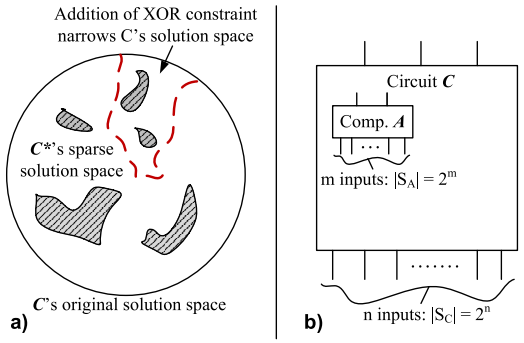

Figure 3. a) A sparse solution space from adding constraints to C. b) Stimulating component A within C.

4.2 Partition-Targeted Simulation

We now propose an approach to automatically stimulate internal partitions while satisfying input constraints.

Stimulating a component within a design: Evenly stimulating an inadequately sensitized component from the primary inputs of the design is not straightforward, because the relationship between the distribution of stimuli on the primary inputs and on the component is often complex. In Figure 3b, we show a component A with m input signals that we desire to stimulate that is many logic levels from the primary inputs of circuit C. We denote the solution space of A with respect to the input constraints as S_{A^C}, and as S_A not considering the input constraints. By applying random vectors to the m signals and checking whether the input constraints are satisfied, we can evenly stimulate A. However, limited controllability could mean that the input constraints are rarely satisfied leading to prohibitive runtimes as shown in Section 5. To this end, we propose a new SAT-based methodology that expands upon our circuit simulation strategy in Section 4.1. For circuit C and its subcircuit A, we observe the following relation between CNF formulae:

$$CNF(C) = CNF(C/A) \wedge CNF(A) \text{ where } S_C \rightarrow S_A \quad (6)$$

Therefore a solution to C implies a solution to A. Since the m signals uniquely determine every legal simulation vector to A, we can probabilistically reduce the solution space of S_A and subsequently S_{A^C} by adding XOR constraints involving the variables m:

$$CNF(C/A) \wedge CNF(A) \wedge (m_{i_1} \oplus m_{i_2} \oplus \cdots \oplus m_{i_j} \oplus 1) \quad (7)$$

This formulation probabilistically reduces S_A in half, and since the input constraints are accounted for by the constraint $CNF(C/A)$, subsequently reduces S_{A^C} in half. Although many S_{C_i} may map to one S_{A_i}, we are concerned with the reduction of the S_{A^C} solution space to achieve an even distribution of simulation along the partition.

Algorithm: Function `partition_sim`, shown in Figure 4, generates an even distribution of simulation vectors by adding multiple random XOR constraints according to Equation 7. The number of random XOR constraints added is determined by the number of simulation vectors (`num_sims`) desired. After constructing the CNF, engineer-specified constraints can be added (`add_additional_constrs`). Then, we add different sets of XOR constraints for each pass of the while loop by function `add_xor_constrs`. When large XOR constraints are added, the high performance of the SAT solver can be maintained by (easily) adding specialized data structures and decision procedures. However, our experiments indicate that small XOR constraints added in our context do not slow down the SAT solver appreciably and do not justify solver extensions as in [6].

If the instance is satisfiable, we add the new simulation vector and decrement `num_sims`. One could also add **blocking clauses** to prevent re-deriving a simulation vector. However, since the solution space that we consider is typically much larger than the number of simulation vectors that we desire, re-derivation should occur infrequently. If many unsatisfiable instances are produced, we can reduce

```
void partition_sim(Partition part, Circuit C, int num_sims){
  num_xor = log2(num_sims);
  CNF = construct_cnf(C);
  add_additional_constrs(CNF);
  while(num_sims){
    add_xor_constrs(num_xor, part, CNF);
    if(Solve(CNF, solution)) {
      add_vector(solution);
      num_sims–;
    }
    remove_xor_constraints(part, CNF);
} }
```

Figure 4. Partition simulation algorithm.

the number of XOR constraints added. We again expect this to occur infrequently as there are generally many more possible solutions than `num_sims`.

Controllability Estimation with XORs: To maximize the effectiveness of our SAT-based simulation, we seek to target poorly-sensitized regions where the number of possible vectors is also sufficiently greater than the number seen so far. Otherwise our SAT-based simulation will fail to generate many new vectors. However, ensuring this requires estimating the number of solutions for partition A with respect to the input constraints, $|S_{A^C}|$.

Using XOR constraints, we can estimate that $|S_{A^C}| > (1+\Delta) * num\_diff\_vecs$, where we consider $\Delta = 1$ in this work, so that less than 50% of the possible vectors have been seen. We use the result from [8] to estimate the number of SAT solutions with random XOR constraints. On average, if the addition of x different XOR constraints does not produce an UNSAT result, we can estimate that the solution space is of size $\geq 2^x$. By examining multiple sets of different XOR constraints, we can obtain bounds with high-accuracy as in [8]. Since we desire a lower-bound computation and need XOR constraints that only involve the partition inputs, we can improve the efficiency of [8] for our specific circuit environment. Thus, our strategy will typically generate an even distribution of *new* simulation vectors.

5 Empirical Validation

We show that adding XOR constraints can evenly stimulate a design and that Toggle can improve activity for poorly stimulated partitions, while being considerably more efficient than directed random simulation. We use MiniSAT [4] to derive simulation vectors and hMetis [10] to perform circuit partitioning. We examine circuits from the IWLS 2005 suite [15] and consider only their combinational portions.

Simulation with SAT: In Table 1, we show the entropy and number of different simulation vectors (`diff sim`) generated along the primary inputs using SAT-based simulation for circuit *alu*4 with the runtime given in seconds (`s`). For the results under `SAT-based`, we add 14 random XOR constraints many times to probabilistically generate U-SAT instances and derive `#sims` simulation vectors. We compare this approach with random simulation and achieve competitively high entropy. Since many of the reductions using 14 XOR constraints produce UNSAT instances, this formulation is computationally expensive. Therefore, we show, under `aprox SAT-based` in Table 1, that by adding fewer XOR constraints determined by $log_2(\#sims)$, we can significantly improve the runtime of the previous SAT-based formulation with nominal degradation to the entropy. Although random simulation is sufficient for this simple example, we now show that even distributions of simulation can be efficiently generated for internal signals while satisfying input constraints.

#sim	Dir.Rand			SAT-based			approx SAT-based			
	diff sim	entr	(s)	diff sim	entr	(s)	#xor	diff sim	entr	(s)
64	64	1.00	<1	63	0.99	2	6	58	0.97	<1
128	128	1.00	<1	128	1.00	4	7	119	0.98	1
256	253	1.00	<1	256	1.00	6	8	240	0.98	1
512	499	0.99	<1	499	0.99	13	9	485	0.99	2
1024	991	0.99	<1	989	0.99	26	10	968	0.99	5

Table 1. Generating even stimuli through random XOR constraints for *alu*4 with 14 inputs. We normalize the entropy by $log_2(\#sim)$, so that 1.0 is the highest entropy possible.

circuit	#gates	avg entr	wrst entr	(Guide+32) new comb	(Rand+32) new comb
spi	3010	9.5	6.4	+26.2	+1.6
systemcdes	3196	9.1	5.6	+15.0	+14.0
tv80	6847	8.9	1.6	+18.6	+0.8
systemcaes	7453	9.7	5.2	+26.6	+12.4
ac97_ctrl	10284	10.0	9.5	+24.8	+21.8
usb_funct	11889	9.9	7.4	+26.4	+12.2
aes_core	20277	7.5	4.1	+17.0	+4.6
wb_conmax	28409	8.8	6.2	+25.2	+3.6
ethernet	37634	9.9	1.6	+26.2	+1.4
des_perf	94002	9.1	5.0	+13.4	+5.0

Table 2. Entropy analysis on partitioned circuits and the number of new input combinations found after adding 32 guided input vectors versus 32 random ones.

Improving Activity with Toggle: In Table 2, we show circuits ordered by their number of `#gates` that are partitioned using the signal-entropy weighting objective so that each partition is $\sim$ 100 gates in size. We find this size to effectively tradeoff our desire for examining the coverage of large parts of the circuit while minimizing the number of the signals considered for entropy analysis and re-simulation. Our results are averaged over many runs.

We show the average (`avg entr`) and worst entropy (`wrst entr`) for the set of partitions derived under 1024 random input vectors, where 10.0 is the maximum entropy. The results indicate that, while the average entropy for each circuit is close to 10.0, there is usually at least one partition that is considerably lower as in `tv80`. We can then perform simulation mainly over these few poorly covered partitions.

In the next part of Table 2, we assess the improvement of our SAT-based targeted re-simulation on a partition with low entropy and a sufficiently large solution space by deriving 32 additional simulation vectors. Our guided simulation is compared with generating 32 more random vec-

circuit	our guided+32 sim time(s)	rand guided+32 sim time(s)	entropy time(s)
spi	1	210	<1
systemcdes	1	110	<1
tv80	1	time-out	<1
systemcaes	1	110	1
ac97_ctrl	1	2	<1
usb_funct	1	18	<1
aes_core	2	time-out	1
wb_conmax	4	232	1
ethernet	10	107	2
des_perf	20	23	2

Table 3. Comparing SAT-based re-simulation with random re-simulation over a partition for generating 32 vectors. The time-out is 10000 seconds. The runtime is also shown for the entropy analysis performed.

tors. We report the number of new combinations seen at the partition inputs averaged over many runs by `new comb`. Our approach outperforms random simulation on almost every circuit. Random simulation performs poorly, e.g., for `ethernet` and `tv80`, indicating strong bias under random simulation. Our approach can still re-derive some previously seen vectors, but we minimize these occurrences by our estimation of the partition's solution space size, which prevents re-simulation on partitions with limited controllability. Even for `ac97`, which is evenly sensitized under random simulation, we see some improvements because the worst-case entropy for the partition targeted for re-simulation is not the maximum value of 10.0.

In Table 3, we show that evenly simulating a partition by randomly assigning values to its inputs and checking whether the primary input constraints are satisfied, is often much slower than using SAT-guided simulation. The results indicate that the SAT-based simulation scales well for larger circuits, in part, because the size of the XOR constraints required are typically small compared to the size of the circuit. Also, our SAT-based simulation often achieves orders of magnitude runtime improvement over random simulation, such as for `wb_conmax` and `ethernet`. Some benchmarks time-out at 10000 seconds, such as for `tv80` and `aes_core`. These results indicate that the solution space of the partition stimulated is sparse with respect to the input constraints. We expect that our technique will perform even better when additional engineer-specified constraints are added, which would further reduce the size of the solution space. For completeness, the last column shows the runtime of the entropy calculation in Equation 3. Clearly, this calculation is fast and scales to large designs.

6 Conclusion

Traditional simulation-driven verification has been rather *ad hoc* and has avoided rigorous theory for efficiency reasons. However, our work shows that certain theoretical results, not used in verification previously, hold the potential to significantly improve simulation coverage through careful feedback on coverage and biasing of input vectors to better stimulate poorly-sensitized parts of the circuit. To achieve these goals, we have introduced 1) an entropy metric to characterize the verification coverage of internal signals and 2) a novel simulation framework using XOR constraints to generate even distributions of stimuli while satisfying complex constraints. Our coverage metric reveals circuit regions that are inadequately stimulated under random simulation. We also show that adding only a few XOR constraints is often sufficient to evenly sensitize a design. Finally, our results indicate that guided simulation is useful for bypassing coverage biases, and outperforms purely random simulation in quality and runtime.

References

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

[2] Specman elite — testbench automation, 2004. (http://www.verisity.com/products/specman.html)

[3] R. Dechter, K. Kask, E. Bin, R. Emek, "Generating random solutions for constraint satisfaction problems", *AAAI*, pp. 15-21, 2002.

[4] N. Een and N. Sorensson, "An extensible SAT-solver", *SAT '03*. (http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/).

[5] C. Fiduccia and R. Mattheyses, "A linear-time heuristic for improving network partitions", *DAC*, pp. 175-181, 1982.

[6] C. Gomes, W. Hoeve, A. Sabharwal, B. Selman, "Counting CSP solutions using generalized xor constraints", *AAAI*, pp. 204-209, 2007.

[7] C. Gomes, A. Sabharwal, B. Selman, "Near-uniform sampling of combinatorial spaces using xor constraints", *NIPS*, pp. 481-488, 2006.

[8] C. Gomes, A. Sabharwal, B. Selman, "Model counting: a new strategy for obtaining good bounds", *AAAI*, pp. 54-61, 2006.

[9] W. Jordan, "Towards efficient sampling: exploiting random walk strategies", *AAAI*, pp. 670-676, 2004.

[10] G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar, "Multilevel hypergraph partitioning: applications in VLSI domain", *IEEE TVLSI*, pp. 69-79, 1999.

[11] A. Mishchenko, S. Chatterjee, R. Jiang, R. Brayton, "FRAIGs: A unifying representation for logic synthesis and verification", *ERL Technical Report*, Berkeley. `http://www.eecs.berkeley.edu/~alanmi/publications/`.

[12] L. Valiant and V. Vazirani. "NP is as easy as detecting unique solutions", *Theor. Comput. Sci.*, pp. 85-93, 1986.

[13] I. Wagner, V. Bertacco, T. Austin, "StressTest: an automatic approach to test generation via activity monitors", *DAC*, pp. 783-788, 2005.

[14] J. Yuan, K. Albin, A. Aziz, Carl Pixley, "Simplifying Boolean constraint solving for random simulation-vector generation", *IEEE TCAD*, pp. 412-420, 2004.

[15] `http://www.opencores.com/`