

PowerRanger: Assessing Circuit Vulnerability to Power Attacks Using SAT-Based Static Analysis

Jeff Hao, Valeria Bertacco
Department of Computer Science and Engineering
University of Michigan
Ann Arbor, United States of America
{jeffhao, valeria}@umich.edu

Abstract—Cryptographic cores, though algorithmically secure, can leak information about their operation during execution. By monitoring the power dissipation of a core, an attacker can extract secret keys used for encryption. To guard against this, designers must minimize the variation of power dissipation of their circuits over time. Unfortunately, power dissipation is a complex function of several different factors, and an exhaustive search for its maximum range is computationally infeasible. In this paper, we propose PowerRanger, a technique based on Boolean satisfiability to produce tight upper and lower bounds on both maximum and minimum power dissipation. In addition, we incorporate min-cut partitioning in our solution to improve its scalability for large designs. We evaluated the quality and performance of PowerRanger on a number of ISCAS benchmarks, as well as two cryptographic cores, showing that our technique significantly outperforms previously known solutions.

Keywords—Security, Power Analysis; Power Estimation; Boolean Satisfiability; Partitioning;

I. INTRODUCTION

Modern cryptographic algorithms such as DES, AES, and RSA are generally considered mathematically secure, in that knowing the encrypted text reveals no information about the original data. However, the physical implementation of these algorithms as digital circuits can introduce weaknesses which can be exploited by attackers. Kocher *et al.* [1] introduced two methods of analyzing a circuit’s power consumption to break into a cryptosystem: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). In SPA, a single trace of a circuit’s power consumption during execution is exploited to yield information about a device’s operation. In DPA, multiple power traces must be gathered to build a correlation function that can reveal individual bits of the encryption key. Both techniques have been shown to allow attackers to break into a variety of cryptographic cores.

The most effective countermeasure to SPA and DPA attacks is to minimize the variation of power consumed by a circuit over time. By reducing the potential range of power dissipation of the circuit, small variations in dynamic power become difficult to isolate. When done well enough, these small power fluctuations during execution become obscured by noise, making SPA attacks almost as time-

consuming as brute-force key searching attacks. DPA attacks are sophisticated and much more difficult to counter, since they have the potential to discern even small variations in power consumption. However, a smaller power dissipation range reduces the amount of information gained by each trace. More traces become necessary, and the computational time needed to analyze them quickly grows to be impractical [2]. A conscientious designer aware of power dissipation during the design process can build cores that are highly resistant to SPA and DPA. However, power dissipation in CMOS circuits is difficult to estimate as it depends on clock frequency, gate delays, process parameters, circuit topology, and input patterns [3]. Within a given design, most of these factors are constant, so that input patterns determining switching activity play the main role in power consumption.

Determining the complete range of possible switching activity would require searching an exponential number of possible input combinations. However, this task can be formulated as a Boolean satisfiability (SAT) problem. SAT solvers today are powerful enough to solve complex problems very quickly. Unfortunately, for very large instances, even state-of-the-art SAT solvers have difficulty producing solutions. For this reason, we develop a novel solution which integrates min-cut partitioning with SAT solving, so that partitioning can be used to break down the problem into smaller blocks and improve performance and scalability.

A. Contributions

In this paper we propose PowerRanger, a novel technique that relies on a min/max ones SAT formulation and min-cut partitioning to generate tight minimum and maximum bounds on the switching activity of a digital circuit within a single clock cycle. To achieve a more accurate correspondence between switching activity and power consumption, PowerRanger takes into account gate fanout of the internal nodes of the design. In our experimental setup over a broad range of ISCAS designs and two cryptographic cores, we have found that PowerRanger can narrow these bounds down to a very small range in every case. Also, in many cases it can find exact solutions for either the minimum or maximum power consumed, or both.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 presents our methodology while Section 4 describes the algorithms that we developed for PowerRanger and provides implementation insights. Section 5 reports experimental results, and Section 6 concludes the paper.

II. PREVIOUS WORK

Several approaches have been proposed to generate bounds on the maximum power dissipation of a circuit, and they can be grouped into one of two categories. The first category includes techniques that search for a sequential pair of input vectors causing maximum switching activity, and provide a lower bound on the maximum if no exact solution can be found. Examples of this include Wang and Roy [4], who greedily assign transitions to gates with large fanout, and then rely on Automatic Test Pattern Generation (ATPG) techniques to propagate those values. Hsiao [5] describes a genetic "spot-optimization" algorithm that attempts to maximize switching activity on one group of nodes at a time. Finally, Qiu *et al.* [6] propose a method based on extreme order statistics to generate maximum power estimates while simulating only a few input vectors. The second category provides an upper bound on the maximum, guaranteeing that every execution of the circuit will dissipate no more power than the estimate. Kriplani *et al.* [7] present a technique to reason on possible node values throughout the circuit, and generate a maximum current envelope, which provides an upper bound on power. This technique was further improved by Hsieh *et al.* in [8] by considering signal correlation to achieve tighter bounds. Our technique is able to compute both lower and upper bounds on the maximum switching activity, allowing us to close in on the maximum from both directions. In addition, PowerRanger provides both bounds on minimum power consumption, which is critical in protecting against SPA and DPA attacks.

We believe that Devadas *et al.* [9] were the first to propose a solution utilizing SAT-solvers to estimate power consumption in CMOS combinational circuits. Their approach was later extended by Mangassarian *et al.* [10] to sequential circuits through the use of pseudo-Boolean satisfiability solvers. Our approach shares the use of a SAT-based framework with these two previous works. However, it differs in our integration of min-cut partitioning, which greatly enhances the scalability of our solution.

III. METHODOLOGY

PowerRanger's goal is to compute lower and upper bounds of minimum and maximum power dissipation for a digital circuit. We make the assumption that, for a given circuit, switching activity is the most influential factor in determining power consumption. Hence, PowerRanger analyzes the variation in switching activity over all possible input stimuli.

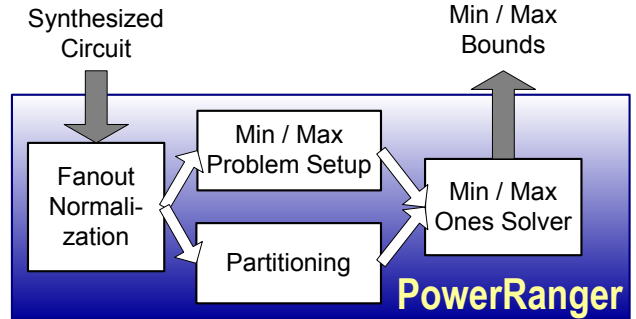


Figure 1. PowerRanger execution flow. A gate-level circuit description is normalized for fanout, then partitioned and converted into a SAT instance. These two components are used by the min/Max ones solver to generate min and Max power bounds.

Figure 1 illustrates the various phases of the PowerRanger execution flow. PowerRanger starts its analysis by taking a gate-level description of a circuit and normalizing its fanout. Then the circuit undergoes partitioning, concurrently with its transformation to a min/Max ones SAT problem. Finally, a min/Max ones solver evaluates the SAT instance and the partitioning, and generates bounds for minimum and maximum switching activity.

A. Circuit Model

PowerRanger makes a few assumptions on the working conditions of the circuit to be analyzed. First, we assume that the entire circuit operates at the same voltage and on the same clock. If that is not the case, the user can partition the circuit based on the various voltage and clock regions and apply PowerRanger separately to each. Based on our assumptions, dynamic power dissipation becomes proportional to the sum of the capacitive load of each switching node. This relationship can be expressed in the formula:

$$P \propto \sum_{i=1}^n C_i T_i \quad (1)$$

where C_i denotes the capacitive load of node i and T_i represents whether node i transitions. PowerRanger estimates power dissipation by normalizing the design to the same C_i for all nodes and then computing the minimum and maximum values for T_i . We also make the assumption that all gates have the same size, and we use the fanout normalization phase to normalize the fanout capacitive load. Finally, we assume that the circuit is glitch-free to avoid asynchronous dissipation.

B. Fanout Normalization

To avoid the use of more heavy weight pseudo-Boolean SAT solvers, it is necessary to make the power "weight" of each node in the graph a single fixed value. Thus, we change the power "weight" of a node with n gates of fanout to n nodes with unit fanout. Consequently, when the node switches, n nodes with unit capacitance will switch together, effectively multiplying the number of transitioning

nodes by the number of fanout gates. These additional nodes have floating outputs that are kept to more accurately model the switching capacitance of each node. Although this transformation makes the fanout worse from an electrical standpoint, the corresponding SAT problem only detects the additional switching nodes, effectively applying a weight factor which corresponds to the original node's fanout.

C. Min/Max Problem Setup

To determine switching activity in the circuit, we unroll the circuit once, and compare 2 consecutive cycles of execution, as shown in Figure 2. Each internal node in the first cycle is XORed with its corresponding copy in the next. The output of each XOR represents a transition on that node, and we can use these outputs to count the total number of transitioning nodes in the circuit.

After unrolling the circuit, the initial state for the first copy and the primary inputs for both cycles are left unconstrained. By choosing different vectors for initial state and the 2 cycles of inputs, we can generate different circuit configurations and vary the number of switching nodes. Unfortunately, to find the specific state and input combinations that would maximize or minimize power dissipation, we would have to iterate over all possible patterns, a task that grows exponentially with the number of state elements and primary inputs. Instead, we rely on a Boolean satisfiability solver, which can perform this task in approximately linear time in the average case, due to advanced search and pruning techniques.

Given a Boolean expression in conjunctive normal form (CNF), Boolean satisfiability is the problem of finding an assignment for each variable in the expression such that the expression evaluates to true. While the Boolean satisfiability problem is NP-complete for any instance including disjunctions of three or more variables, researchers have recently discovered several clever techniques to address this family problems and have made available software tools, called SAT solvers, capable of solving in practice extremely complex problems in almost linear time. Consequently, we formulate our switching activity problem as a SAT instance,

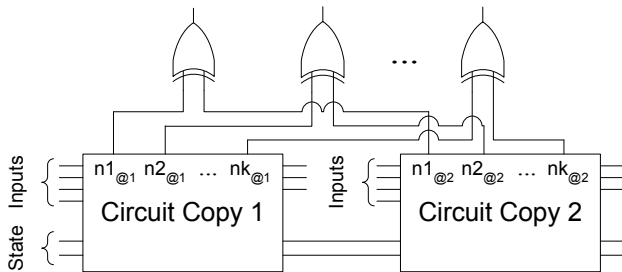


Figure 2. During min/Max problem setup, the circuit is unrolled once. Sequential outputs from the first copy are connected to sequential inputs of the second. Then XOR gates are added, connecting corresponding signals between the two copies. Each XOR output indicates if a transition has occurred at the corresponding node, and we call the corresponding SAT variable a *transition variable*.

and convert our unrolled circuit into CNF format. This can be done easily by representing each internal node in the circuit as a Boolean variable and converting each gate into a set of equivalent Boolean clauses [11]. The complete transformation can be done in linear time.

D. Partitioning

Even though SAT solvers typically perform well, they can struggle to produce solutions for very large instances. To improve the scalability of our approach, we use a partitioning algorithm to divide the circuit into smaller blocks. This information can be used in two ways, depending on which type of bound we are trying to explore. In one case, we use the partitioned design to create a set of simpler goals to be accomplished simultaneously by the SAT solver. These goals act as "lighthouses" in directing the search toward the maximization or minimization goal. In the other case, we use the partitioning to decompose the problem into a series of sub-problems to be solved one at a time.

The technique we use is a min-cut partitioning algorithm that considers the circuit as a hypergraph, where wires are vertices and gates are edges, and partitions the circuit's wires (corresponding to SAT variables) into a user-selected number of sets. The algorithm is a min-cut in that it minimizes the number of edges crossing any set. From our circuit's standpoint, this means that the partitioning minimizes the dependency between SAT variables belonging to distinct sets, thus making it easier for a SAT solver to satisfy distinct constraints over the different sets. Finally, note that the min-cut partitioning tool that we use, hMETIS [12], attempts to roughly balance the size of each partition.

E. Min/Max Ones Solver

Once the circuit is partitioned and converted into a SAT instance, it is ready for input into a min/Max ones SAT solver. Normally, the output of a SAT solver is an assignment of all Boolean variables such that the SAT instance evaluates to true. However, we need to find a satisfiable solution where the number of asserted Boolean variables corresponding to switching nodes is minimized or maximized. This variant of the SAT problem is called a *min/Max ones* problem. Note that, rather than considering all variables in the CNF instance, we must minimize or maximize only those variables that correspond to the outputs of the XOR gates added during setup, which we call the *transition variables*. As these variables capture whether a node in the original circuit transitions, they allow us to obtain the min and max switching activity of the circuit. The problem of minimizing or maximizing the asserted variables within a subset of those in the SAT instance is formally known as *min/Max distinguished ones*, though we refer to it as *min/Max ones* throughout the rest of this paper.

IV. BOUNDS SEARCH ALGORITHM

In order to find bounds for minimum and maximum switching activity we developed two algorithms: Inner Bound Search and Outer Bound Search. The first is used to determine an upper bound on the minimum switching activity and the lower bound on the maximum, while the second is used to compute the other two bounds, a lower minimum bound and upper maximum bound. Although they are different, at their core both these algorithms use a min/Max ones solver.

A. At Least/At Most SAT Constraints

While several CNF SAT solvers are publicly available, PowerRanger requires a specialized min/Max ones solver. To this end, we enhanced an existing DPLL-style SAT solver [13] with two additional functionalities, `atMost` and `atLeast`, which force the solver to find a satisfiable solution with the additional constraint of maximizing and minimizing the number of asserted transition variables, respectively. We implemented the two functions into MiniSAT [14], since it is both a powerful solver and has a flexible code structure. The `atMost` function takes two arguments, the list of transition variables and a bound `bound`, and forces the solver to return a satisfiable solution where at most `bound` transition variables are asserted. If no solution is possible under this constraint, the min/Max solver will return UNSAT. The `atLeast` function operates similarly by forcing at least `bound` transition variables to be asserted.

B. Inner Bound Search

The Inner Bound Search algorithm begins by searching for a satisfiable solution to the problem without posing any constraints on the number of asserted transition variables. It then repeats the search iteratively, adding new constraints until the solver returns UNSAT. In the maximum lower bound computation, Inner Bound Search creates `atLeast` constraints in each iteration, forcing the new solution to include strictly more asserted transition variables. In addition,

```
Inner Bound Search
Sol = SAT_solve(instance);
while (Sol) {
  sum = 0;
  foreach (partition p) {
    bound[p] = asserted_trans_vars (p, Sol);
    add atMost/atLeast(trans_vars(p), bound[p]);
    sum += bound[p]; }
  add atMost/atLeast(trans_vars, sum-1/sum+1);
  Sol = SAT_solve(instance); }
return sum;
```

Figure 3. Pseudocode for Inner Bound Search algorithm. Once an initial solution is found, it is parsed to determine the number of asserted transition variables in each partition. New bounds are created for each partition to force successive solutions to have similar transitions. One final bound is placed over the whole instance to force either strictly increasing or decreasing numbers of transitions throughout the design, and then the instance is rerun through the SAT solver. This process is iterated until the instance becomes UNSAT.

we also constrain the number of transition variables asserted in each partition set to be non-decreasing. The additional constraints on the partition sets greatly simplify the task of the solver, enabling it to reach a solution faster and with a smaller memory footprint. When searching for the upper bound to the minimum switching activity, we proceed similarly to the previous case, with the exception of using `atMost` constraints instead of `atLeast`, and with the inequalities inverted. The pseudocode for this algorithm is shown in Figure 3.

In the Inner Bound Search algorithm, the SAT solver is guided by the partitioning toward solutions similar to the previous one. Since the solver's search space is reduced, solutions can be found faster. However, the partition constraints may lead the solver to a sub-optimal solution. In the worst case, Inner Bound Search will return a worse bound than theoretically possible without partitioning constraints. However, the complexity of the non-partitioned search often prevents it from ever advancing the search to the point achieved through partitioning.

C. Outer Bound Search

The Outer Bound Search algorithm has several similarities with Inner Bound Search. It is used to compute the upper bound on the maximum switching activity and the lower bound on the minimum. We search for a bound in each circuit partition independently, and then add the individual bounds together, which leads to a conservative estimate. For instance, when searching for the maximum's upper bound, we choose a partition in the circuit and attempt to solve the instance by forcing all of the transition variables in that partition to switch. If we cannot find any solution, we relax our constraint, asking for all but one (or all but a few) variables to switch. We continue to relax this constraint until the solver can find a satisfiable solution. Then we apply this process to each partition. Finally, we add the number of asserted transition variables we could find in each partition and obtain our upper bound. The lower bound on

```
Outer Bound Search
sum = 0;
foreach (partition p) {
  Sol = UNSAT;
  bound = 0/trans_vars(p);
  while (Sol is UNSAT) {
    add atMost/atLeast(trans_vars(p), bound);
    Sol = SAT_solve(instance);
    bound++/bound--; }
  bound--/bound++; //obtain last valid bound
  sum += bound; }
return sum;
```

Figure 4. Pseudocode for Outer Bound Search algorithm. When searching for the upper max, we create a bound forcing all variables within a single partition to transition. The instance is checked to see if it can be satisfied with this constraint, and if not, the bound is relaxed and the instance rerun until a solution is found. Iterated over all partitions, the sum of bounds found form a bound for the upper max. When searching for the lower min, the initial bounds force none of the variables to transition instead.

the minimum is calculated with a dual approach. Figure 4 outlines the process described. To be runtime-aware in the search, instead of optimizing one partition at a time, we work in a round robin fashion on all the partitions.

Note that, while computationally more scalable, this approach is approximate. When we attempted to solve the same problem without partitioning, we could reach a tighter upper bound of 521. With fewer partitions, the solutions found will tend to be closer to the optimum, but the algorithm’s runtime increases. The ideal number of partitions is the point where partition size becomes almost too large for the solver to handle, which varies greatly from one executing host to the next. If a single partition is used, bounds found through Outer Bound Search can be optimal.

V. EXPERIMENTAL RESULTS

We evaluated our solutions on a broad range of ISCAS benchmarks, both combinational and sequential, and two cryptographic cores: AES and DES [15]. All experiments were performed on a 3.2 GHz Intel Pentium IV processor with 1GB of RAM.

In order to evaluate the effects of partitioning over the quality of the results, we ran PowerRanger several times on the same testbench while varying the block size used for partitioning during each run. In Figure 5 we show the results for *c7552*. The graph plots the best lower maximum values found over time during a PowerRanger run. The curves shows the trends for a number of partitions ranging from 1 to 64 in powers of two. The chart clearly shows that a greater number of partitions can help achieve tighter bounds more quickly, as the estimated lower bound found at 10,000 seconds improves from 1 through 32 partitions. However, the gains end when we use 64 partitions, where PowerRanger falls into a local optimum and terminates after only 88 seconds of runtime. The best partition of 32 sub-blocks corresponds to 185 nodes per block. Using results from this and other testbenches as guidelines, we determined the best partitioning is at approximately 150 nodes per block.

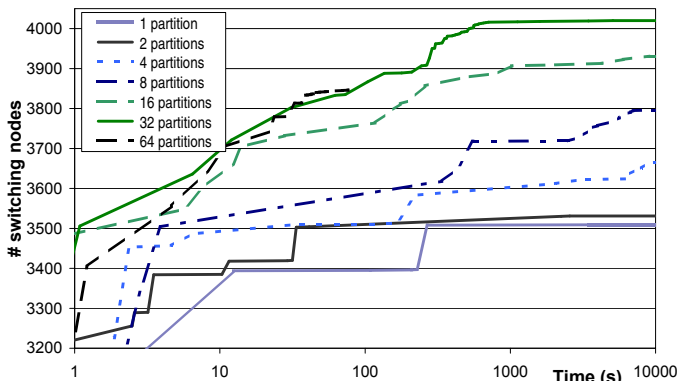


Figure 5. PowerRanger computing the lower maximum bound for the *c7552* benchmark. The plot shows how different size partitioning affects the speed at which PowerRanger converges.

We then compared our technique, PowerRanger, against two other approaches, a typical SAT-based approach, and random simulation. We configured PowerRanger to use a single partition for small circuit instances, and partitioned the larger circuits such that each partition contained approximately 150 nodes. For the SAT-based approach, we used the same flow as PowerRanger, but without min-cut partitioning. For random simulation, all primary inputs were randomly stimulated at each iteration and simulated with Synopsys’ VCS simulator. For each testbench and technique, we allocated 10,000 seconds of runtime.

Table I summarizes our results. For each testbench, the table lists the number of internal nodes it contains after fanout normalization, and then compares side by side the results obtained for all four bounds with each of the three techniques: simulation (Sim), SAT-based (SAT) and PowerRanger (PR). The values reported in each column indicate the best bound found for the number of internal switching nodes with the corresponding technique. The goal is to obtain values which are proportional to the power dissipation of the circuit. Since simulation cannot be used to generate lower min and upper max bounds, we must use 0 as the lower min bound and the number of gates as the upper max bound in this approach. Bold entries indicate that an optimal value was found (the solver terminated when running with a single partition). The last six columns indicate the range of the the minimum and maximum bounds found for all three approaches as a percentage of the total number of nodes.

The results show that a simulation-based approach can perform moderately well at estimating the minimum switching activity for a circuit, reducing the range to within 5% for many of the benchmarks. However, the simulation-based maximum estimates are quite poor, with a range of over 80% for the AES core. This is mostly due to the lack of a bound for the upper maximum. Without it, it is impossible to tell how close simulation was to the actual maximum. This same problem does not exist for the minimum because the actual minimums happen to be close to zero. For the SAT-based solution, the results show that it can perform very well on benchmarks of a few thousand gates. The minimum and maximum for many designs can be pinpointed precisely. Unfortunately, for larger circuits, the performance of the SAT-based solution deteriorates rapidly. The estimates for the upper minimum are worse than simulation for many benchmarks, though the search for the lower minimum is able to terminate optimally for all benchmarks except AES. Estimates for the lower maximum are generally better than simulation, and the range of the maximum found by the SAT-based solution is often reduced by 20% or more when compared with simulation.

PowerRanger is able to do as well as a SAT-based solution for small instances, but the incorporation of min-cut partitioning enables it to perform well even on complex problem instances. PowerRanger’s minimum switching ac-

Design Nodes	lower min		upper min			lower Max			upper Max		min range			Max range			
	SAT	PR	Sim	SAT	PR	Sim	SAT	PR	SAT	PR	Sim	SAT	PR	Sim	SAT	PR	
c432	265	0	0	2	0	0	176	206	206	206	206	1%	0%	0%	34%	0%	0%
c499	328	0	0	16	0	0	217	200	221	245	235	5%	0%	0%	34%	14%	4%
c880	533	0	0	25	0	0	352	444	447	469	450	5%	0%	0%	34%	5%	1%
c1355	888	0	0	90	177	0	455	482	514	653	562	10%	20%	0%	49%	19%	5%
c1908	1445	0	0	117	580	0	907	952	1026	1187	1095	8%	40%	0%	37%	16%	5%
c2670	1990	0	0	293	0	0	1235	1316	1499	1815	1597	15%	0%	0%	38%	25%	5%
c3540	2655	0	0	107	0	0	1433	1421	1592	2019	1777	4%	0%	0%	46%	23%	7%
c5315	4003	0	0	868	1594	0	2457	2545	2946	3861	3233	22%	40%	0%	39%	33%	7%
c6288	4320	0	0	857	815	0	2264	2535	2911	4139	2978	20%	19%	0%	48%	37%	2%
c7552	5944	0	0	1354	2204	0	3441	3508	4019	5701	4368	23%	37%	0%	42%	37%	6%
s298	261	0	0	0	0	0	186	221	221	221	221	0%	0%	0%	29%	0%	0%
s344	251	0	0	0	0	0	178	196	196	196	196	0%	0%	0%	29%	0%	0%
s382	328	0	0	0	0	0	143	270	270	270	270	0%	0%	0%	56%	0%	0%
s386	324	0	0	0	0	0	213	242	242	242	242	0%	0%	0%	34%	0%	0%
s444	376	0	0	0	0	0	163	282	282	282	282	0%	0%	0%	57%	0%	0%
s526	469	0	0	0	0	0	212	370	370	370	370	0%	0%	0%	55%	0%	0%
s820	675	0	0	0	0	0	466	540	540	540	540	0%	0%	0%	31%	0%	0%
s832	686	0	0	0	0	0	477	546	546	546	546	0%	0%	0%	30%	0%	0%
s953	718	0	0	0	0	0	281	303	303	303	303	0%	0%	0%	61%	0%	0%
s713	598	0	0	0	0	0	377	491	491	491	491	0%	0%	0%	37%	0%	0%
s1238	881	0	0	0	0	0	455	502	502	521	527	0%	0%	0%	48%	2%	3%
s1423	1226	0	0	7	0	0	677	902	964	1041	977	1%	0%	0%	45%	11%	1%
s1488	1356	0	0	0	0	0	832	909	909	909	909	0%	0%	0%	39%	0%	0%
s5378	4405	0	0	252	1884	0	1676	2618	3220	3951	3548	6%	43%	0%	62%	30%	7%
s9234	8155	0	0	225	3180	0	3301	4509	5621	7288	6280	3%	39%	0%	60%	34%	8%
s13207	11879	58	58	691	3862	65	2810	5882	7554	10235	8775	6%	32%	0%	76%	37%	10%
s15850	14214	5	5	471	4725	5	4420	7199	9162	13190	10794	3%	33%	0%	69%	42%	11%
s35932	27841	0	0	0	10390	0	17931	17633	18607	27759	23155	0%	37%	0%	36%	36%	16%
s38417	33742	19	19	718	11989	20	9199	15972	21106	33414	26536	2%	35%	0%	73%	52%	16%
s38584	34448	0	0	2398	16793	0	14778	18476	22857	33976	27149	7%	49%	0%	57%	45%	12%
DES	4403	118	118	710	1165	132	1669	1608	1900	2848	2297	16%	24%	0%	62%	28%	9%
AES	29706	8	158	3577	5066	339	5493	6063	7563	28751	16696	12%	17%	1%	82%	76%	31%

Table I

TABLE OF RESULTS COMPARING A SIMULATION-BASED SOLUTION, A SAT-BASED ONE [10], AND POWERRANGER. ON THE LEFT, THE NUMBERS SHOW THE BEST BOUND FOUND WITHIN 10000 SECONDS OF RUNTIME, IN NUMBER OF TRANSITIONING NODES, WITH OPTIMAL VALUES IN BOLD. ON THE RIGHT, THE POSSIBLE RANGES FOR MINIMUM AND MAXIMUM ARE SUMMARIZED AS A PERCENTAGE OF THE TOTAL DESIGN.

tivity estimates are optimal for all but 4 benchmarks, and the minimum range is no more than 1%. PowerRanger's maximum estimates are equally good, many of them with a maximum range of less than 10%. Even in the toughest benchmark, the AES core, PowerRanger is still able to achieve a maximum range of 31%, compared with 76% from SAT and 82% from simulation. Overall, PowerRanger is able to achieve tighter bounds than both SAT and simulation.

VI. CONCLUSIONS

In this paper, we presented PowerRanger, a technique to achieve very tight bounds on both maximum and minimum power dissipation using an integrated SAT and min-cut partitioning solution. PowerRanger is effective in estimating the range of power dissipation of a digital circuit, and can be used to protect against power attacks to cryptographic cores. We are able to produce optimal results for a number of circuits, and demonstrate how our technique can outperform both random simulation and SAT alone.

REFERENCES

- [1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. CRYPTO*, 1999.
- [2] C. Clavier, J.-S. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," in *Proc. CHES*, 2000.
- [3] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI*, 1994.
- [4] C.-Y. Wang and K. Roy, "Maximum power estimation for CMOS circuits using deterministic and statistic approaches," in *Proc. VLSID*, 1996.
- [5] M. Hsiao, "Peak power estimation using genetic spot optimization for large VLSI circuits," in *Proc. DATE*, 1999.
- [6] Q. Qiu, Q. Wu, and M. Pedram, "Maximum power estimation using the limiting distributions of extreme order statistics," in *Proc. DAC*, 1998.
- [7] H. Kriplani, F. Najm, and I. Hajj, "Maximum current estimation in CMOS circuits," in *Proc. DAC*, 1992.
- [8] C.-T. Hsieh, J.-C. Lin, and S.-C. Chang, "A vectorless estimation of maximum instantaneous current for sequential circuits," in *Proc. ICCAD*, 2004.
- [9] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in cmos combinational circuits using boolean function manipulation," *IEEE Trans. on CAD*, Mar. 1992.
- [10] H. Mangassarian, A. Veneris, S. Safarpour, F. N. Najm, and M. S. Abadir, "Maximum circuit activity estimation using pseudo-boolean satisfiability," in *Proc. DATE*, 2007.
- [11] J. Marques-Silva and K. Sakallah, "Boolean satisfiability in electronic design automation," in *Proc. DAC*, 2000.
- [12] G. Karypis and V. Kumar, "hMETIS: A hypergraph partitioning package," 1998.
- [13] M. Davis and H. Putnam, "A computing procedure for quantification theory," *Journal of the ACM*, 1960.
- [14] N. Een and N. Srensson, "MiniSat - a SAT solver with conflict-clause minimization," in *Proc. SAT*, 2005.
- [15] "Opencores.org, <http://www.opencores.org>," 2007. [Online]. Available: www.opencores.org