Int. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), pg 247-257. September 2008.

Generating Worst-Case Stimuli for Accurate Power Grid Analysis

P. Marques Morgado[1], Paulo F. Flores[1],
José C. Monteiro[1], and L. Miguel Silveira[2]

[1] INESC-ID/IST, TU Lisbon,*
morgado@algos.inesc-id.pt, jcm@inesc-id.pt, pff@inesc-id.pt
[2] INESC-ID / IST, TU Lisbon / Cadence Research Labs, lms@inesc-id.pt

Abstract. Power distribution systems provide the voltages and currents that devices in a circuit need to operate properly and silicon success requires its careful design and verification. However, problems like voltage drop, ground bounce and electromigration, which may cause chip failures, are worsening, as more devices, operating at higher frequencies, are placed closer together. Verification of this type of systems is usually done by simulation, a costly endeavor given the size of current grids, making the determination of the worst-case input setting a crucial task. Current methodologies are based on supposedly safe settings targeting either unrealistic simultaneous switching on all signals or heuristic accounts of the joint switching probability of nearby signals. In this paper we propose a methodology for computation of the worst-case stimuli for power grid analysis. This is accomplished by determining the input vector that maximizes the number of gates, in close proximity to each other, that can switch in a given time window. The addition of these temporal and spatial restrictions makes the solution of the underlying optimization problem feasible. Comparisons with existing alternatives show that only a fraction of the gates change in any given timing window, leading to a more robust and efficient verification methodology.

1 Introduction

Power distribution system design is of paramount importance for silicon success. According to available data [1], more than 50% of tapeouts using 0.13-micron technology would have failed, if the power distribution system were not validated beforehand. Continued system and technology trends for increased miniaturization make this an area of increasing concern in the semiconductor industry. Lower operating voltages, increased device integration density and leakage currents, higher operating frequencies and the use of low power design techniques, all tend to stress the power grid as technology evolves.

Power grid verification is usually accomplished by simulation [2, 3] implying that only settings corresponding to the chosen stimuli are simulated and thus verified. Therefore, stimuli must be chosen appropriately and should be worst-case representatives of the relevant scenarios. Since the power grid encompasses the whole die area, its description is rather large and the simulation process is slow, costly, and highly complex. This results from the necessity to take into

* This research was supported in part by the Portuguese FCT under program POSI, ref. EEA-ESE/61528/2004.

account a huge number of power grid parameters (RLC non-idealities) and all the devices that take current from it.

Simulating the power grid with all the devices might be impossible for VLSI circuits, as it would consume too many resources. Furthermore, simulating for all possible device settings is also impossible, as it would take too long. In addition, given the size and complexity of current designs, it is also impossible to assume that designer intervention, albeit useful, will be sufficient to generate appropriate sets of stimuli for the grid verification. Therefore, an automatic way of generating realistic sets of stimuli given the knowledge of the actual circuit implementation is necessary. A naive and conservative approach to stimuli generation is to assume that in the worst case all gates conspire to request current at the same time. Obviously this situation is unrealistic and may lead to gross over-design. Furthermore, from a power grid standpoint, the worst-case scenario is not directly tied only to the number of gates switching. The most detrimental situation in terms of voltage drop or ground bounce occurs whenever there is a significant number of active devices in a short period of time drawing current from close regions of the power grid.

In this paper we propose to determine the worst-case settings that causes such a current surge from the power grid. This is accomplished by applying pseudo-boolean optimization (PBO) over a boolean network that symbolically represents the conditions for the gates in the original circuit making a rising transition (similar analysis can be conducted for falling transitions). The solution of such a problem is the input vector pair that maximizes the number of gates making a rising transition. Similar approaches have been proposed, specially in the context of peak power determination [4] but also for determining power grid stimuli [5]. In [6], the authors directly tackle this problem considering zero-delay and unit-delay models. An extension to arbitrary delay models is mentioned, but in this case the number of potential time instants in which gates can transition grows very quickly, and the problem cannot be realistically tackled in the manner outlined. Equivalently, for realistic delay descriptions, the networks resulting from symbolic simulation grow too quickly for practical usage as the different delay values lead to too many combinations of signal arrival times. However, in the particular setting that we are concerned with, in which only switching within a given timing and spatial window is relevant, several optimizations are possible and the resulting network can be pruned to a workable size. By repeatedly applying this procedure to a set of sliding timing and spatial windows, we can determine, for each of these windows, the worst case input vector.

The outline of this paper is as follows. In Section 2, we present some background on power grid analysis and related stimuli generation. Then in Section 3 we present the details of the proposed technique when applied to combinational circuits. We describe the techniques involved, discuss the effects of different delay models and propose an extension for handling sequential circuits. We also show how to generate the pseudo-boolean problem whose solution is the worst-case sought after. The results obtained using the proposed method are presented in Section 4, and conclusions are drawn in Section 5.

2 Background

Two of the most common problems with power grids are voltage drop and ground bounce. Voltage drop, also called IR drop, is the voltage reduction that occurs on power supply networks as a result of current flow through the power (and ground) wiring. As the underlying circuit and logic gates switch, current is requested from the supply network which travels on the non-ideal wiring causing resistive drop. This effect can be static or dynamic and in essence causes fluctuations in the power rails. Similar effects may be found in ground wiring, usually referred to as ground bounce. Both effects contribute to lower operating voltages within devices (i.e. logic cells/gates in digital circuits), which in general increase the overall time response of a device and might cause a failure in its operation.

Simulation is the most commonly used method to validate the power grid. It enables one to verify if the power grid is suited for a given design, that is, if it is robust enough to deal with problems such as voltage drop and ground bounce. Typically, after power grid design, a simulation (at electrical level) of the grid must be performed which requires that a model of the power grid must be generated, via extraction. To proceed with the simulation, a set of grid stimuli must also be generated and applied to the grid. This mainly consists of a simplified model of the circuit cells and their corresponding current waveforms. Given the number of cells attached to the power grid and the intricate correlations that exist between them, picking the right set of cells to use as stimuli is far from trivial. Some of the simulation tools consider all the circuit devices as independent stimuli to the power grid assuming this is a worst-case scenario. Others allow users to define which stimuli should be applied, i.e., which circuit cells are going to be active during the simulation. Most of the times this definition is based on user experience and knowledge. However, both options may deteriorate the quality and resulting accuracy of power grid simulation. A critical region may be neglected if the user misses the combination of grid stimuli that will cause the worst voltage drop or ground bounce (a false negative). Results from a simulation obtained on the assumption that all cells need to be accounted for, may also identify invalid critical regions of the power grid that are supposedly affected by voltage drop or ground bounce (a false positive). This occurs because in normal working conditions all cells in the circuit can not draw current from the power grid at the same time. Moreover, this type of simulation may also increase total run-time and memory requirements from simulators. After this simulation procedure, the designer will try to solve IR-drop problems, usually by placing decoupling capacitance inside those critical regions or widening the metal lines for higher current availability. If those regions are non-critical, from a voltage drop and ground bounce point of view, the insertion of decoupling capacitance will only increase the overall static power consumption and it will be a waste of silicon area. This circuit changes can itself cause voltage drop and ground bounce to appear in other circuit regions. It might be argued that the worst-case setting corresponds to an unlikely scenario, therefore of limited applicability. However, if such scenario may lead to chip malfunction, it needs to be addressed and a solution provided. Furthermore, it first needs to be identified.

Algorithm 1 High level pseudo-code for the proposed methodology.

```
Worst_Case_Input_Vector_Pairs(net)
  for each spatial window sw do
    for each temporal window tw do
      symbNet = Symbolic_Simulation(net, sw, tw);
      (cts, of) = Translate_Net_ILP(symbNet);
      ivp = PBO_Solver(cts, of);
```

From a design standpoint, determination of such a worst-case can also guide designers to budget area for decap capacitors in the appropriate areas, instead of wasting precious space in locations where such capacitors are really not needed. From a power grid standpoint, the worst case stimuli is related to conditions leading to a surge of power current demand, whereby a large number of gates closely located, all switch in a narrow window of time, placing demands on the power grid that cannot be met. Determining such a worst-case requires the analysis of the possible conditions that may lead logic or other circuitry to evidence bursts of activity in a narrow spatial and time window. This problem can be cast as an optimization problem whereby one determines the maximum number of gates that may switch in a given time+spatial window. Note that without the restrictions provided by the spatial and timing windows, the problem is most likely computationally unfeasible.

3 Determination of the Worst Case Setting

In this section we present the proposed approach to determining the worst-case input vector pair that causes a maximum number of rising transitions, within a given time period, over a set of gates in close proximity. The pseudo-code presented in Algorithm 1 describes our method in generic terms. For each combination of gates in the spatial window sw and of events within the temporal window tw, we compute a boolean circuit $symbNet$ that has as primary inputs two copies, I_{-1} and I_0, of the primary inputs of the original network, I, representing the input transition $I_{-1} \rightarrow I_0$, and as primary outputs signals identifying a rising transition for each gate g_i in the original circuit at every time instant t where such a transition is possible (assuming that g_i and t fall respectively, in the spatial and temporal windows sw and tw). One output evaluating to 1 indicates that the corresponding gate g_i in the original circuit makes a rising transition at the corresponding instant t when input vector I_0 follows the input vector I_{-1}. We call this the symbolic network ($symbNet$) of the original circuit [7].

We then create a Pseudo-Boolean Optimization (PBO) problem by translating the symbolic network into a set of 0-1 Integer Linear Programming (ILP) constraints (cts). The objective function (of) to maximize is simply the appropriately weighted sum of the primary outputs of the symbolic network. We are currently using Bsolo [8] to compute the worst-case input vector pair (ivp), but any generic PBO solver can be used instead, e.g. MiniSat+ [9]. An important observation to make regarding the method presented in Algorithm 1 is that

the analysis of each timing interval in each spatial window can be performed independently. Therefore, simultaneous usage of multiple machines or parallel engines can be made, as the parallelization is trivial, leading to very efficient analysis. In the following we discuss relevant issues regarding each step of the algorithm described by the pseudo-code of Algorithm 1.

3.1 Spatial and Temporal Windows

For the problem at hand, we are interested in analysing the behavior of the power grid in the worst-case scenario where a significant number of devices in close proximity are active in a short period of time. First of all, we define spatial windows by partitioning the set of power rails into spatial windows of a given size. The idea here is to capture locality in terms of current consumption from the power grid, therefore in a realistic scenario these windows should be defined on the 3D structure that represents the multiple metal layers encompassing the power grid. We then proceed by partitioning the gates into sets that fall inside each spatial windows. We then restrict our analysis to maximizing the number of rising transitions in each spatial window. The window size does not have to be fixed a-priori and can change depending on the density and type of devices in each region. Furthermore, the size of this window can be parametrized as a function of certain design parameters, such as the pitch of the power grid, the type of packaging used (i.e. the proximity to a bias source), and the type of cells used in the design (and their corresponding current signatures, which can be characterized offline). For simplicity however, we will assume in this paper, a fixed size grid. Ideally, the number of spatial windows would be determined by incrementally sliding this window to include all possible combinations of gates entering and leaving the window. In this situation we could compute the exact worst case for a give spatial window size. However, even though as we discussed all windows can be analyzed concurrently, this would lead to a prohibitively large number of windows to analyze. Hence, in practice we are considering windows that have some relevant overlap between them (in the results section we used 50% overlap, but higher overlap, potentially at additional computational cost, will likely lead to a better approximation).

Compounded with this restriction of gates to analyze, we will only consider events that occur within a given temporal window. If the events occur sufficiently apart, the power grid will have time to recover, and thus the multiple switching does not constitute a troublesome situation. The length of the temporal window should be configured based on technology data, the power rails width and the grid's ability to provide current at a given rate. For efficiency, one should limit the symbolic simulation such that the resulting network only includes events within the time interval defined by the window. Note that, as we slide the temporal window, the symbolic network can be computed incrementally, by removing all the events (and transitive fanin cone that drive exclusively those gates) that leave the temporal window and adding the new events that enter the window. Such incrementality leads to additional efficiency improvements. Again, all temporal windows can be concurrently analyzed, as previously mentioned.

The resulting symbolic network for a given spatial and temporal window is therefore much simpler than if generated for the full-blown original circuit (where all events at all time instants are considered). This allows us to use realistic delays instead of being bound to simplified, unrealistic delay models, such as zero or unit delay. It also translates into much easier PBO problems, putting them within reach of fast state-of-the-art solvers. In fact, our approach reduces the overall complexity by splitting a large problem of exponential complexity into a polynomial number of smaller problems of exponential complexity.

3.2 Symbolic Simulation

The symbolic simulation of a logic circuit generates a new logic circuit which has the Boolean conditions for all values that each gate in the original network may assume at different time instants, given an input vector pair [7]. If a zero delay model is used, each gate in the circuit can only assume two different values, one corresponding to each input vector. For this simple case, the symbolic network corresponds to two copies of the original network, one copy evaluated with the first input vector and the other copy with the second, and an additional gate to detect the nodes transitions. In the case of unit or general delay models, the gate output nodes of a multilevel network can have multiple transitions in response to a two-vector input sequence. In this procedure, the simulator processes one gate at a time, moving from the primary inputs to the primary outputs of the circuit. For each gate g_i, an ordered list of the possible transition times of its inputs is first obtained. Then, possible transitions at the output of the gate are derived, taking into account transport delays from each input to the gate output [7]. The processing done is similar to the "time-wheel" in a timing simulator.

Crucial to the accuracy of our method is the usage of realistic gate delays during symbolic simulation. In our case, these delays are obtained from the technology library. Using a general delay model increases significantly the complexity of the symbolic network as the number of events at different time instants at the inputs of a given gate scales exponentially with the logic level of the gate (as opposed to linearly in the case of unit delay, a situation which is however grossly unrealistic). However, in spite of this explosion in the growth of the network, our approach is viable because the symbolic network is computed only for the gates inside the spatial window, and their transitive fanin cone, and then only for the timing window under study.

We are interested in determining rising transitions at the gates within the spatial window. To this end, we add an AND gate between pairs of gates in the symbolic network corresponding to consecutive time instants and relating to the same node in the original network, negating the input with the earliest instant. When one of these AND gates in the symbolic network evaluates to 1, it means that the two-input vector sequence at the input of the symbolic network causes a rising transition at the gate and time instant to which the AND corresponds to. For example, consider two consecutive possible values, at instants t_1 and t_2, a gate g in the original network may assume. Then the symbolic simulation will include logic signals representing these two values, g_{t_1} and g_{t_2}. We add the AND

gate $g_{t_1}\text{-}g_{t_2}$, defined as $g_{t_1}\text{-}g_{t_2} = \overline{g_{t_1}} \cdot g_{t_2}$. If $g_{t_1}\text{-}g_{t_2} = 1$ for some input vector pair, clearly g will make a rising transition at instant t_2 under this sequence of input vectors. Naturally, we can as easily compute the falling transitions by negating the AND input with the latest instant, instead of the earliest. We just need to be careful not to use EXOR gates, as in the case of determining transitions for power estimation [7], since we are concerned with maximizing transitions all in the same direction, not any transition.

Modeling Inertial Delay. Logic gates require energy to switch state. The energy in a gate input signal is a function of its amplitude and duration. If its duration is too small, the signal will not force the gate to switch. The minimum duration for which an input change must persist in order for the gate to switch states is called the *inertial delay* of an element (cf. [10, p. 187]).

In determining the input vector pair that maximizes the number of gates making a rising transition we should eliminate transitions that in reality do not occur due to the inertial delays of the gates. In [11] a method as been proposed that eliminates these transitions directly in the construction of the symbolic network. Basically, if we have three consecutive time points for a gate, g_{t_1}, g_{t_2} and g_{t_3}, within the inertial delay Δ_{in} from t_1 ($t_3 < t_1 + \Delta_{in}$), we assure there are no transitions in g_{t_2} by making $g_{t_2} = g_{t_1}$ when $g_{t_1} = g_{t_3}$. In this situation we create a new output g'_{t_2} as

$$g'_{t_2} = \begin{cases} g_{t_1} & \text{when } g_{t_1} = g_{t_3} \quad \text{(eliminates the spike on } g_{t_2}\text{)} \\ g_{t_2} & \text{otherwise} \qquad\qquad\quad \text{(propagates the } g_{t_2} \text{ transition)} \end{cases} \tag{1}$$

which leads to the following Boolean function

$$g'_{t_2} = g_{t_2}(g_{t_1} + g_{t_3}) + g_{t_1} g_{t_3} \tag{2}$$

for every three time points within the inertial delay of gate g. The g'_t functions are used as the inputs to the AND gates added to compute the rising transitions that could occur on the output of g in time point t. Also, we use the g'_t functions for the next logic level, thus any transitions eliminated at the output of a gate are not propagated to its transitive fanout.

Sequential Circuits. The symbolic simulation can be directly adapted to handle sequential circuits. For this type of circuits, the inputs to the combinational logic block can be partitioned into the primary inputs of the circuit, I (external inputs as in the combinational case) and present state lines, S (coming from the internal state registers). The problem is still to find the worst-case input transition for active elements in the spatial/temporal window, but now in terms of $(I_{-1}, S_{-1}) \rightarrow (I_0, S_0)$. The additional constraint is that the next state lines, S_0, are a function of the primary inputs and the present state lines (I_{-1}, S_{-1}). This constraint is easily solved by introducing in the symbolic network the next state logic block that generates S_0 from (I_{-1}, S_{-1}). Hence, the resulting symbolic network has (I_{-1}, S_{-1}, I_0) as primary inputs and the worst-case input transition will be a function of the present state. With our approach, we can avoid invalid values for the present state, e.g. by limiting their values to the set of reachable

states with additional constraints when generating the PBO problem. Alternatively, we can force a fixed present state using additional constraints, which might be useful if analysis is sought for a preset number of states. This would for instance be the case if realistic traces of system execution are available, and verification is sought for those conditions. Care should be taken however, that those conditions reflect the worst case behavior of the network.

3.3 Conversion to a PBO Problem

The mapping of the Boolean network into a 0-1 ILP optimization model is obtained by representing each gate in the symbolic network in Conjunctive Normal Form (CNF) format [12]. For example, a 2-input AND gate, $c = a \wedge b$, is translated to CNF as $(a+\overline{c})(b+\overline{c})(\overline{a}+\overline{b}+c)$. Each clause is then converted into a 0-1 ILP constraint using the straightforward mapping presented in [13]. The above AND gate would be described by the following set of restrictions:

$$\begin{aligned} a - c &\geq 0 \\ b - c &\geq 0 \\ -a - b + c &\geq -1 \\ a, b, c &\in \{0, 1\} \end{aligned}$$

We define the optimization variables as the set of outputs of the AND gates that identify rising transitions in the gates within the spatial/temporal window. The cost function is then defined simply as a linear function of the optimization variables. We set the cost value of each optimization variable to 1, but we can as easily use a cost value that models the driving capability of each gate, using for instance information from the pre-characterized current signatures of each type gate and taking into account the load for each instance. This will model more accurately the current derived from the power grid by a transition on the corresponding gate. The model thus obtained can serve as input to generic solvers of PBO problems which will find the optimal solution for the given constraints.

4 Results

Our approach was run on a Pentium IV at 3Ghz, with 1GB of RAM memory over the combinational set of benchmark circuits from ISCAS89. Each of these circuits was mapped into a technology library and placed in a die with a square aspect ratio. We defined 64 overlapping spatial windows for each circuit resulting from 8 windows in both axis which have 50% of overlap between windows. For each spatial window we computed the exact sequence of two input vectors that maximizes the number of rising transitions within that spatial window. Since the circuits we considered were fairly small in depth, for simplicity, but without loss of generality, in the following, we did not consider any time windows (i.e considered the whole interval as a single window). We point out that while these circuits are very small with respect to the overall die size and capacity, this is actually a realistic setting to consider. Likely, the available die space will be

Table 1. Average size and CPU time for the generation of the symbolic networks.

	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c7552
CPU(s)	14.7	0.1	0.1	0.1	2.9	0.2	52.2	2.2	2.8
Nodes	17796	1489	1858	1275	9020	2450	24415	5570	9681
POs	8861	700	906	613	4466	1183	12116	2713	4726

Table 2. Results for a particular spatial window of the c3540 circuit under different delay models.

	0-delay	1-delay	gen-delay
CPU	0.1	1.4	415.0
Nodes	1851	10249	119963
POs	617	4816	59758
Max Trans	210	538	1131

filled with several smaller sub-circuits that are either functionally disconnected or sometimes connected through latches or register banks. In either case, they can be analyzed separately (and concurrently) for determining the worst-case input switching pattern. Table 1 presents statistics for generating the symbolic networks under a general delay model. We give the CPU time in seconds, the number of nodes in the symbolic network and the number of primary outputs, i.e., the total number of possible events in gates within this window. The results presented are an average over all the 64 spatial windows.

To underline the importance of the general delay model, we compare the symbolic networks for one particular case in Table 2. We can observe that indeed the number of possible events with general delay is much larger than unit-delay. Consequently it does take significantly more time to generate and the resulting circuits are much larger. This is the reason why solving for the entire circuit at all time instants is intractable for medium to large size problems. In Fig. 1 we compare, for a few circuits, the results of our method with approximate results obtained by logic simulation using the SIS system. These were generated with 100,000 random input vectors applied to the circuit generated after symbolic simulation. The conclusion is that simulation is in general off by a factor of two. These results are typical of what we saw with most examples.

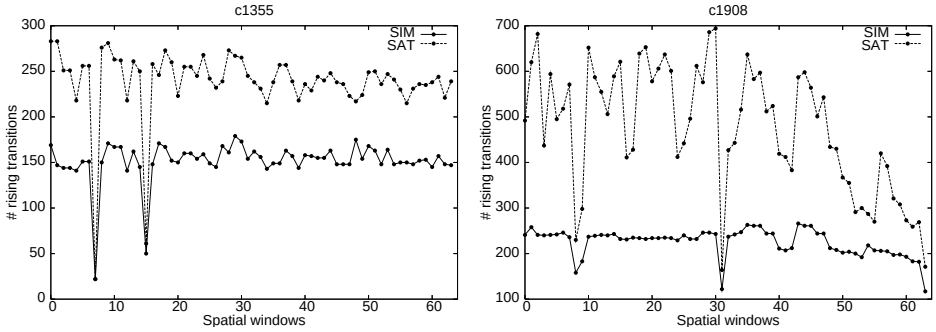

Fig. 1. Comparison of results obtained for circuits c1355 and c1908 using random simulation (SIM) and PBO (SAT).

5 Conclusions

We have proposed a methodology for the computation of worst-case stimuli for power grid analysis. The method uses placement and netlist information to generate to partition the network into a set of spatial and temporal windows where simultaneous switching could lead to lowered bias voltage and circuit malfunction. A sequence of pseudo-boolean optimization problems is then solved to determine the worst-case solution corresponding to the highest space and time concentrated activity, from which the corresponding input stimuli is derived.

A prototype implementing the proposed method was developed and the result obtained over several benchmark circuits were presented. The results showed that only a fraction of the gates are active at any time and enabled the identification of the time interval and region of the power grid where simultaneously switching of cells may lead to considerable impact over the grid. These results should help the circuit designer in optimizing the power grid for a more robust behaviour.

References

1. : Power grid verification. Whitepaper, Cadence Design Systems, Inc. (2001)
2. Nassif, S.R., Kozhaya, J.N.: Fast power grid simulation. In: Proc. of ACM/IEEE Design Automation Conf. (DAC), ACM/IEEE (June 2000) 156–161
3. Zhong, Y., Wong, M.D.F.: Fast algorithms for ir drop analysis in large power grid. In: ICCAD'05: Proceedings of the 2005 IEEE/ACM Int. conference on Computer-aided design, IEEE Computer Society (2005) 351–357
4. Chai, D., Kuehlmann, A.: Circuit-based preprocessing of ilp and its applications in leakage minimization and power estimation. In: IEEE Int. Conference on Computer Design: VLSI in Computers and Processors, IEEE (October 2004) 387–392
5. Kriplani, H., Najm, F.N., Hajj, I.N.: Pattern independent maximum current estimation in power and ground buses of cmos vlsi circuits: algorithms, signal correlations, and their resolution. IEEE Transactions on Computer-Aided Design of Integrated Circuits (TCAD) **14**(8) (August 1995) 998–1012
6. Mangassarian, H., Veneris, A., Safarpour, S., Najm, F.N., Abadir, M.S.: Maximum circuit activity estimation using pseudo-boolean satisfiability. In: Proc. of Design, Automation and Test in Europe conference (DATE). (April 2007) 1538–1543
7. Ghosh, A., Devadas, S., Keutzer, K., White, J.: Estimation of Average Switching Activity in Combinational and Sequential Circuits. In: Proceedings of the 29^{th} Design Automation Conference. (June 1992) 253–259
8. Manquinho, V., Marques-Silva, J.: Effective Lower Bounding Techniques for Pseudo-Boolean Optimization. In: Proc. of Design, Automation and Test in Europe conference (DATE). (March 2005)
9. Een, N., Sorensson, N.: An Extensible SAT-solver. Theory and Applications of Satisfiability Testing **2919** (2004) 502–518
10. Breuer, M., Friedman, A.: 4. In: Diagnosis and Reliable Design of Digital Systems. Computer Science Press (1976)
11. Monteiro, J., Rinderknecht, J., Devadas, S., Ghosh, A.: Optimization of Combinational and Sequential Logic Circuits for Low Power Using Precomputation. In: Proceedings of the 1995 Chapel Hill Conference on Advanced Research on VLSI. (March 1995) 430–444
12. Flores, P., Neto, H., Marques-Silva, J.: An Exact Solution to the Minimum Size Test Pattern Problem. ACM Transactions on Design Automation of Electronic Systems **6**(4) (October 2001) 629–644
13. Barth, P.: A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization. Technical report, Max-Planck-Institut Fűr Informatik (January 1995)