# Maximum Power-Up Current Estimation in Combinational CMOS Circuits

Assim Sagahyroon and Fadi Aloul

Department of Computer Engineering
American University of Sharjah, UAE
*{asagahyroon, faloul}@aus.edu*

## Abstract

The continuing decrease in feature size and increase in chip density is causing leakage current to be a major contributor to power dissipation in integrated circuits. A viable approach to the reduction of leakage current is to use power cut-off or gating techniques. In power gating, a PMOS sleep transistor is used to turn-on or turn-off the $V_{dd}$ source to the circuit block. In combinational circuits, the maximum power up current depends only on the input vector that wakes up the circuit from its sleep mode. In this work, we formulate the problem of estimating the maximum power-up current as an integer linear programming (ILP) problem and use advanced Boolean satisfiability (SAT) and generic ILP solvers. Results indicate that generic ILP solvers are very useful in estimating the maximum power-up current.

## 1. Introduction

The recent surge in the deployment and utilization of portable electronic devices has brought power dissipation to the forefront as a major design concern. Leakage power is becoming a growing problem as technology scales for battery-operated devices and will grow exponentially as power supplies and threshold voltages scale down in future processes.

One of the techniques used for reducing leakage power is to partition the design into different blocks, each operating on a block-specific voltage. This technique provides designers with the ability to switch off the block supply when the functional block does not have any logic activity to perform. Switching the supply off will minimize leakage current and the power dissipation caused by it. This power cut-off or power gating technique is implemented using a sleep transistor. For example, in CMOS circuits, a PMOS sleep transistor with a high threshold voltage can be used to switch-on or switch-off the $V_{dd}$ supply of a block. Similarly, an NMOS transistor can be used to connect or disconnect the block path to ground. Figure 1 illustrates the presented technique.

When the circuit blocks are woken-up from their sleep mode, a significant power-on charging current comparable to that of a normal switching current is induced [12]. This current, if excessive, produces surges that may cause Ldi/dt and IR voltage drops and electromigration. This has a negative impact on circuit reliability and performance. Therefore an estimate of the maximum current during power-up is essential in designing reliable and high performance CMOS combinational circuits. Unlike the maximum switching current which depends on two input vectors [20], maximum wake-up current depends only on one input vector. In this paper, advanced Boolean satisfiability and generic ILP solvers are used to identify the one input vector that can lead to the maximum wake-up current.

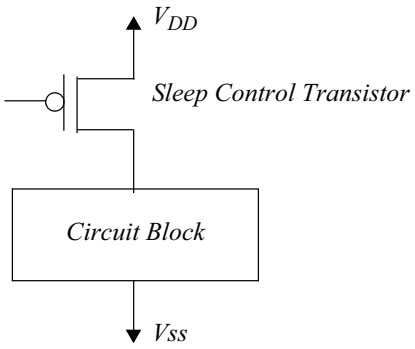



**Figure 1. Circuit with power gating**

This paper is organized as follows: in Section 2 we discuss further the power estimation problem and derive an approximation for it. We describe Boolean satisfiability solvers and show examples of its application in solving electronic design automation problems in Section 3. Section 4 summarizes the formulation of the power estimation problem within the context of SAT. Sections 5 and 6 include discussions of the experimental results and the conclusion of the paper, respectively.

## 2. Activity Estimation

If we assume that we are using a PMOS sleep transistor then all internal nodes are fully discharged during sleep mode. Therefore, power-up current will be proportional to the total charge that needs to be recovered after wake-up [13]. This power up current is given by:

$$P = \sum_{allGates} (Val(g) \cdot C(g) \cdot V_{dd}) \quad (1)$$

In (1), $C(g)$ represents the load capacitance of the gate $g$, $Val(g)$ is the logic value of the gate output and $V_{dd}$ is the supply voltage. Assuming that all gates have the same input capacitance and ignoring wire capacitance and assuming $C(g)$ is proportional to the fanout of the gate, the above equation can be simplified to [13]:

$$P = \sum_{allGates} (Val(g) \cdot Fanout(g)) \quad (2)$$

$P$ in (2) can be used as a measure of the power up current. Hence, the estimation problem reduces to finding the input vector that maximizes $P$. A gate with a logical out of 1 implies that there is a charge of $(C(g) \cdot V_{dd})$ stored in the load capacitance. A possible solution is to exhaustively simulate *all* possible input vectors, however, this is impractical for circuits with large number of inputs. In [12, 13], algorithms based on Automatic Test Pattern Generation (ATPG) techniques were used to find a vector that maximizes $P$. The results reported were encouraging but further investigation of this problem is still warranted. In this work, we show how to formulate the estimation problem as an ILP problem. We experiment with powerful SAT-based and generic ILP solvers in finding an estimate for the power $P$. The presented approach is *complete* and identifies the input vector that guarantees the maximum possible value for $P$.

## 3. Boolean Satisfiability

Recent years have seen a remarkable growth in the use of Boolean Satisfiability (SAT) models and algorithms for solving various problems in Electronic Design Automation (EDA). This is mainly due to the fact that SAT algorithms have seen tremendous improvements in the last few years, allowing larger problem instances to be solved in different applications domains [3, 8, 10, 14, 17, 25]. Such applications include formal verification [4], FPGA routing [19], global routing [1], logic synthesis [16], and sequential equivalence checking [5]. SAT has also been extended to a variety of applications in Artificial Intelligence including other well-known NP-complete problems such as graph colorability, vertex cover, and Hamiltonian path [7].

In SAT, given a formula $f$, the objective is to identify an assignment to a set of Boolean variables that will satisfy a set of constraints. If an assignment is found, it is known as a *satisfying* assignment, and the formula is called *satisfiable*. Otherwise if an assignment doesn't exist, the formula is called *unsatisfiable*. The constraints are typically expressed in conjunctive normal form (CNF). In CNF, the formula consists of the conjunction (AND) of $m$ clauses $\omega_1, \ldots, \omega_m$ each of which consists of the disjunction (OR) of $k$ literals. A literal $l$ is an occurrence of a Boolean variable or its complement. Hence, in order to satisfy a formula, each of its clauses must have at least one literal evaluated to true.

As an example, the CNF instance:

$$f(a, b, c) = (a + \bar{b}) \cdot (a + b + c) \quad (3)$$

consists of 3 variables, 2 clauses, and 5 literals. The assignment $\{a = 0, b = 1, c = 0\}$ leads to a conflict, whereas the assignment $\{a = 0, b = 0, c = 1\}$ satisfies $f$.

Despite the problem being NP-Complete, there have been dramatic improvements in SAT solver technology over the past decade. This has lead to the development of several powerful SAT solvers that are capable of solving problems consisting of thousands of variables and millions of constraints in a few seconds [3, 10, 14, 17, 25].

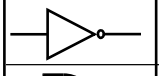
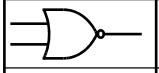
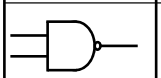
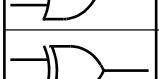
| Gate Type | Gate Equation | CNF Expression |
|---|---|---|
|  | $z = NOT(x)$ | $(x + z) \cdot (\bar{x} + \bar{z})$ |
|  | $z = NOR(x, y)$ | $(\bar{x} + \bar{z}) \cdot (\bar{y} + \bar{z}) \cdot (x + y + z)$ |
|  | $z = NAND(x, y)$ | $(x + z) \cdot (y + z) \cdot (\bar{x} + \bar{y} + \bar{z})$ |
|  | $z = AND(x, y)$ | $(x + \bar{z}) \cdot (y + \bar{z}) \cdot (\bar{x} + \bar{y} + z)$ |
|  | $z = OR(x, y)$ | $(\bar{x} + z) \cdot (\bar{y} + z) \cdot (x + y + \bar{z})$ |
|  | $z = XOR(x, y)$ | $(\bar{x} + y + z) \cdot (x + \bar{y} + z) \cdot (\bar{x} + \bar{y} + \bar{z}) \cdot (x + y + \bar{z})$ |

**Table 1. CNF formulas representing simple gates.**

Recently, SAT solvers [1, 6, 9, 23, 24] have been extended to handle pseudo-Boolean (PB) constraints which are linear inequalities with integer coefficients that can be expressed in the normalized form [1] of:

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \geq b \quad (4)$$

where $a_i, b \in Z$ and $x_i$ are Boolean variables. PB constraints can, in some cases, replace an exponential number of CNF constraints. They have been found to be very efficient in expressing "counting constraints" [1]. Furthermore, PB extends SAT solvers to handle *optimization* problems as opposed to only *decision* problems. Subject to a given set of CNF and PB constraints, one can request the minimization (or maximization) of an objective function which consists of a linear combination of the problem's variables. Note that each CNF constraint can be viewed as a PB constraint. For example the CNF constraint $(a + \bar{b})$ can be viewed as the PB constraint $a + \bar{b} \geq 1$. PB constraints represent ILP inequalities. Hence, generic ILP solvers, such as CPLEX [11], can also be used to solve SAT-encoded instances. Recent studies showed that both SAT-based ILP solvers and generic ILP solvers are competitive and compete on different benchmarks [1, 6].

In this paper, we are interested in using SAT solvers to measure the maximum power dissipation in combinational circuits. Circuits are easily represented as a CNF formula by conjuncting the CNF formulas for each gate output. A gate output can be expressed using a set of clauses which specify the valid input-output combinations for the given gate. Hence, a CNF formula $\varphi$ for a circuit is defined as the union of set of clauses $\varphi_x$ for each gate with output $x$:

$$\varphi = \bigcup_{x \in Q} \varphi_x \quad (5)$$

where $Q$ denotes all gate outputs and primary inputs in the circuit. Table 1 shows generalized CNF formulas for various gates. For example, a NOR gate with inputs $x$ and $y$ and output $z$ is represented using the following set of clauses $(\bar{x} + \bar{z}) \cdot (\bar{y} + \bar{z}) \cdot (x + y + z)$. If $x$ is assigned the value 1, the first clause will imply $z = 0$, since this is the only possible assignment that will satisfy the first clause. Similarly if $x$ and $y$ are assigned the value 0, $z$ will be implied to 1 since this is the only assignment that will satisfy the third clause.

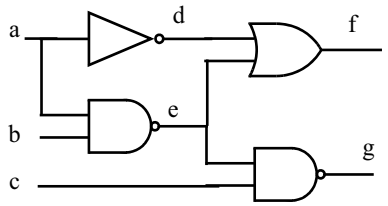
## 4. Problem Formulation

To estimate the maximum instantaneous power, one needs to search for an input vector V1 that maximizes the weighted wake-up activity of $P$. The weight of each gate is determined by its fanout as indicated in equation (2). The power estimation problem is formulated as a SAT (i.e. 0-1 ILP) problem as follows:

- A set of clauses, i.e. CNF constraints, representing the logical behavior of the circuit after the application of the input vector V1.
- An objective function, expressed as a PB constraint, representing the power-up for each gate.

The CNF constraints are represented as explained in Section 3. The PB constraint representing the objective function, consists of the sum of all gate's outputs. In other words, this can be viewed as a constraint representing the predicate, "*there exist a one input vector that can cause a weighted summation of gate transitions > k*" where $k$ is an integer value. In formulating the problem, integer coefficients are used to represent the fanout (capacitance) of each gate.

An example is shown in Figure 2 that clearly illustrates the various steps of the proposed approach. The circuit shown in the example has four gates and three primary inputs. A total of seven variables are needed in the problem. CNF constraints representing the circuit's logical behavior are generated. The objective function consists of the sum of the output of all four gates. Each output is associated with an integer coefficient that is equal to the fanout of the gate. In the given example $d$, $f$, and $g$ have a fanout of 1 and $e$ has a fanout of 2. The optimization instance is passed to advanced SAT and ILP solvers which return the assignment: $\{a, b, c\} = \{0, 1, 0\}$. The assignment yields the input vector that generates a maximum power-up of 5.




Circuit Logic (CNF)

$(a + d) \cdot (\bar{a} + \bar{d})$

$(a + e) \cdot (b + e) \cdot (\bar{a} + \bar{b} + \bar{e})$

$(\bar{d} + f) \cdot (\bar{e} + f) \cdot (d + e + f)$

$(c + g) \cdot (e + g) \cdot (\bar{c} + \bar{e} + \bar{g})$

Objective Function (PB)

$Max(d + 2e + f + g \geq 0)$

Solution:
Max power-up = 5
$\{a, b, c\} = \{0, 1, 0\}$

**Figure 2. Illustrative example showing how to determine the maximum power-up in a circuit.**

## 5. Experimental Results

Table 2 summarizes the results obtained using the SAT-based ILP solver PBS [1] and the commercial ILP solver CPLEX [11]. The PBS experiments were conducted on a Pentium-IV 2.8 Ghz workstation running Linux with 500 MB of RAM. The CPLEX experiments were conducted on a SunBlade 1000 workstation with 2MB cache running SunOS 5.9. We used the default settings for PBS and CPLEX. We used the MCNC [18] benchmark circuits. Each benchmark was sensitized using "sis" [21] into a circuit consisting of 2-input NAND, NOR and inverter gates. The runtime was set to a limit of 1000 seconds.

In order to speed up the ILP solvers by eliminating a large number of input vectors and thus reducing the search space, an initial objective goal was identified by generating 10K random primary input vectors and identifying the maximum power-up current estimate achieved using these randomly generated vectors.

Columns two and three list the number of primary inputs and the number of gates in each circuit. Column three of the table (*MaxPos*) is the theoretical upper maximum for $P$ (it assumes that all the gates in the circuit will contribute to the summation in equation (2)). The *Random* column represents the best estimate found using random vector generation. The *Time* column indicates the runtime (in seconds) for each solver. The *Value* column represents the maximum weighted activity value obtained using each solver. The *%-Max* column gives the percentage of the activity reported by the solver (*Value*) relative to the maximum upper bound (*MaxPos*). While PBS timed out on a few instances, CPLEX was successful in solving all presented instances. Nevertheless, both search engines succeeded in improving the results obtained using the random approach by returning higher estimates. In cases where the solver completes the search (timed-out), the estimate provides an upper (lower) bound of the maximum wake-up switching activity possible in the circuit.

## 6. Conclusion

In this paper, we formulated the maximum current estimation problem as an ILP problem. Advanced complete SAT-based and generic ILP solvers were used to find an estimate for the maximum activity in the circuit upon switching from sleep to a wake-up mode. The proposed method was implemented and tested on a sizable set of circuits. Results indicate that ILP solvers are very successful in estimating the maximum power activity.

| Circuit Name | #PI | #Gates | MaxPos | Random | PBS | | | CPLEX | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Time | Value | %-Max | Time | Value | %-Max |
| **sct** | 19 | 143 | 198 | 126 | 0.05 | 129 | 65.2 | 0.03 | 129 | 65.2 |
| **frg1** | 28 | 143 | 167 | 105 | 14.45 | 111 | 66.5 | 0.02 | 111 | 66.5 |
| **b9** | 41 | 147 | 183 | 118 | 0.7 | 123 | 67.2 | 0.02 | 123 | 67.2 |
| **c8** | 28 | 211 | 289 | 188 | 2.63 | 192 | 66.4 | 0.03 | 192 | 66.4 |
| **9symml** | 9 | 252 | 356 | 229 | 0.09 | 229 | 64.3 | 0.04 | 229 | 64.3 |
| **C432** | 36 | 282 | 436 | 263 | 36.9 | 274 | 62.8 | 0.05 | 274 | 62.8 |
| **ttt2** | 24 | 303 | 425 | 262 | 2.68 | 267 | 62.8 | 0.06 | 267 | 62.8 |
| **C880** | 60 | 442 | 616 | 350 | 1000 | 355 | 57.6 | 0.35 | 367 | 59.6 |
| **alu2** | 10 | 462 | 738 | 425 | 0.24 | 425 | 57.6 | 0.48 | 425 | 57.6 |
| **term1** | 34 | 525 | 739 | 464 | 133 | 485 | 65.6 | 0.08 | 485 | 65.6 |
| **C1355** | 41 | 552 | 894 | 578 | 1000 | 581 | 65.0 | 0.76 | 588 | 65.8 |
| **C499** | 41 | 567 | 891 | 530 | 1000 | 532 | 59.7 | 0.89 | 544 | 61.1 |
| **C1908** | 33 | 771 | 1219 | 734 | 1000 | 745 | 61.1 | 1.19 | 754 | 61.9 |
| **alu4** | 14 | 878 | 1419 | 799 | 6.71 | 801 | 56.4 | 11.86 | 801 | 56.4 |
| **C2670** | 155 | 1087 | 1628 | 957 | 1000 | 989 | 60.7 | 5.92 | 1014 | 62.3 |
| **vda** | 17 | 1417 | 2419 | 1571 | 2.26 | 1572 | 65.0 | 0.32 | 1572 | 65.0 |
| **C6288** | 32 | 2400 | 4271 | 2827 | 1000 | 2832 | 66.3 | 54.54 | 2832 | 66.3 |
| **i10** | 257 | 3366 | 5454 | 2946 | 1000 | 3002 | 55.0 | 32.76 | 3120 | 57.2 |
| **C7552** | 206 | 3381 | 5547 | 3159 | 1000 | 3229 | 58.2 | 47.35 | 3306 | 59.6 |
| **i8** | 133 | 3764 | 6504 | 3823 | 1000 | 3823 | 58.8 | 38.67 | 3882 | 59.7 |
| **t481** | 16 | 4767 | 7176 | 4952 | 867 | 4952 | 69.0 | 1.29 | 4952 | 69.0 |

**Table 2. Experimental results using the SAT-based 0-1 ILP solvers PBS and the generic ILP solver CPLEX.**

## References


[1] F. Aloul, A. Ramani, I. Markov, and K. Sakallah, "Generic ILP versus Specialized 0-1 ILP: An Update," in *Proc. of the Int'l Conf. on Computer-Aided Design* (ICCAD), 450-457, 2002.

[2] P. Barth, "A Davis-Putnam based Enumeration Algorithm for Linear Pseudo-Boolean Optimization," *Technical Report MPI-I-95-2-003, Max-Planck-Institut Für Informatik*, 1995.

[3] R. J. Bayardo Jr. and R. C. Schrag, "Using CSP Look-Back Techniques to Solve Real World SAT Instances," in *National Conference on Artificial Intelligence* (AAAI), 203-208, 1997.

[4] A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu, "Symbolic Model Checking using SAT procedures instead of BDDs", in *Proc. of the Design Automation Conf.* (DAC), 317-320, 1999.

[5] P. Bjesse and K. Claessen, "SAT-based Verification Without State Space Traversal", in *Proc. of Formal Methods in CAD*, 372-389, 2000.

[6] D. Chai and A. Kuehlmann, "A fast pseudo-boolean constraint solver," in *Proc. of the Design Automation Conference* (DAC), 830-835, 2003.

[7] N. Creignou, S. Kanna, and M. Sudan, "Complexity Classifications of Boolean Constraint Satisfaction Problems", in *SIAM Press*, 2001.

[8] M. Davis, G. Logemann, and D. Loveland, "A Machine Program for Theorem Proving," in *Comm. of the ACM*, 5(7), 394-397, 1962.

[9] H. Dixon and M. Ginsberg, "Inference Methods for a Pseudo-Boolean Satisfiability Solver," in *National Conference on Artificial Intelligence* (AAAI), 635-640, 2002.

[10] E. Goldberg and Y. Novikov, "BerkMin: A fast and robust SAT-solver," in *Proc. of the Design Automation and Test Conference in Europe* (DATE), 142-149, 2002.

[11] ILOG CPLEX, *http://www.ilog.com/products/cplex*

[12] F. Lei and L. He, "Maximum current estimation considering power gating," in *Proc. of Intl. Symp. on Physical Design* (ISPD), 222-227, 2001.

[13] F. Lei, L. Hei, and K. Saluja, "Estimation of maximum power-up current," in *ASPDAC*, 51-58, 2002.

[14] J. Marques-Silva and K. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability", in *IEEE Trans. on Computers*, (48)5, 506-521, 1999.

[15] MCNC Benchmarks, *http://www.cbl.ncsu.edu/CBL_Docs/Bench.htm*

[16] S. Memik and F. Fallah, "Accelerated Boolean Satisfiability-Based Scheduling of Control/Data Flow Graphs for High-Level Synthesis", in *Proc. of the Int'l Conf. on Computer Design*, 2002.

[17] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an Efficient SAT Solver," in *Proc. of the Design Automation Conference* (DAC), 530-535, 2001.

[18] MCNC benchmarks, *http://www.cbl.ncsu.edu/CBL_Docs/Bench.html*

[19] G. Nam, F. Aloul, K. Sakallah, and R. Rutenbar, "A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints," in *Proc. of Intl. Symp. on Physical Design* (ISPD), 222-227, 2001.

[20] M. Pedram, "Power minimization in IC design: principles and applications," ACM Trans. on Design Automation of Electronic Systems, 1(1), 3-56, 1996.

[21] E. Sentovich et. al, "SIS: A System for Sequential Circuit Synthesis," *Univ of California-Berkeley, UCB/ERL M92/41*, 1992.

[22] J. Silva and K. Sakallah, "Boolean satisfiability in electronic design automation," in *Proc. of the Design Automation Conference* (DAC), 675-680, 2000.

[23] H. Sheini and K. Sakallah, "Pueblo: A Modern Pseudo-Boolean SAT Solver," in *Proc. of the Design Automation and Test Conference in Europe* (DATE), vol. 2, 684-685, 2005.

[24] J. Whittemore, J. Kim, and K. Sakallah, "SATIRE: A New Incremental Satisfiability Engine," in *Proc. of the Design Automation Conference* (DAC), 542-545, 2001.

[25] H. Zhang, "SATO: An Efficient Propositional Prover", in *Proc. of the Int'l Conference on Automated Deduction*, 272-275, 1997.