Minimizing stand-by leakage power in static CMOS circuits

Srinath R.Naidu E.T.A.F.Jacobs
Design Automation Section, Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands

Abstract

In this paper we concern ourselves with the problem of minimizing leakage power in CMOS circuits consisting of AOI (and-or-invert) gates as they operate in stand-by mode or an idle mode waiting for other circuits to complete their operation. It is known that leakage power due to sub-threshold leakage current in transistors in the OFF state is dependent on the input vector applied. Therefore, we try to compute an input vector that can be applied to the circuit in stand-by mode so that the power loss due to sub-threshold leakage current is the minimum possible. We employ a integer linear programming (ILP) approach to solve the problem of minimizing leakage by first obtaining a good lower bound (estimate) on the minimum leakage power and then rounding the solution to actually obtain an input vector that causes low leakage. The chief advantage of this technique as opposed to others in the literature is that it invariably provides us with a good idea about the quality of the input vector found.

1. Introduction

In future technologies, the problem of sub-threshold leakage power in CMOS circuits will grow in significance. When transistors are switched off, a certain amount of leakage current flows and this results in leakage power of magnitude (I_{leak})(Vdd). The leakage current is exponentially dependent on the value of the threshold voltage such that if the threshold voltage is reduced (as it will be in future technologies), the leakage current registers an exponential increase. We can control the leakage current by technological measures where dual-threshold transistors are used [9], or by supplying a particular input vector that minimizes leakage power since the leakage power of a gate is dependent upon the state in which the inputs of the gate are maintained. [5], [6], and [10] have previously studied the problem of finding a minimum leakage vector that can be applied to a circuit in stand-by mode. The approaches in these papers focus on finding the input vector that minimizes leakage power heuristically. It is known [5] that the ratio between the maximum and minimum leakage power could be quite large and for some circuits could be as large as 5. We present here an integer linear programming model to compute an input vector that can be applied to the primary inputs of the circuit in stand-by mode in order to minimize leakage power. Since the integer linear program consists of a large number of variables, we do not solve it exactly. Instead we solve an appropriate linear relaxation to provide a lower bound on the integer optimum and then employ a technique called randomized rounding to round the solution to the linear relaxation, which is typically fractional, to an integer solution. Experimental results are presented to show the practicality of the method for moderate-sized circuits.

2. Problem Formulation

Case (a): *Library consisting of only two-input NAND gates and INVERTERS.*

This basic case has certain features that allow it to be modeled in a way that is different from the general case. Specifically we show that the objective vector of the ILP can be linearized in a very natural way using only variables inherent to the problem.

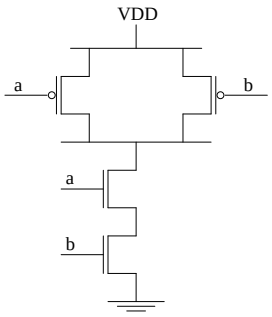

Fig 1: Schematic of a 2-input NAND gate.

Consider the NAND gate in Fig. 1. We analyze the leakage power of the gate for all four input vectors, ab = 11, 10, 01, 00. Let P_0 be the leakage power when the input vector ab = 00, P_1 the power for ab = 10, P_2 the power for

ab = 01 and P_3 the power for ab = 11. To get the values of P_0, P_1, P_2 and P_3 we need to examine the on-off states of the transistors in the NAND gate of Fig 1. When ab = 11, the two NMOS transistors in series are on, while the two PMOS transistors in parallel are off. Thus in this case we need to consider the sum of sub-threshold leakage currents in the two off PMOS transistors. When ab = 10, the leakage current pertains to one of the NMOS transistors being in the off state and similarly for ab=01. When ab =00, the leakage current pertaining to two NMOS transistors in series. P_0, P_1, P_2 and P_3 may be regarded as constants dependent on the CMOS process technology.

For an integer linear programming model (ILP) we need a linear objective function over a set of constraints. The objective function for the NAND gate is MIN: P_0 (1-a)(1-b) + P_1 a(1-b) + P_2 (1-a)b + P_3 ab. This is not linear but we can rewrite it as $(P_0 - P_1 - P_2 + P_3)$ ab + $(P_1 - P_0)$a + $(P_2 - P_0)$b + P_0. Now for a NAND gate we know that the output c = 1 – ab. So we can substitute (1-c) for ab. The objective function for the NAND gate is linear in the Boolean variables representing the inputs and output of the NAND gate. Next we build the constraint set pertaining to the NAND gate. To do this we use a clausal description of the function of the NAND gate, i.e. we come up with a set of clauses (linear constraints) as in [7]. The four input-output combinations can be described by the following three conditions:

$$\begin{aligned} &\sim a \Rightarrow c \text{ or } a + c \geq 1, \\ &\sim b \Rightarrow c \text{ or } b + c \geq 1, \\ &ab \Rightarrow \sim c \text{ or } -a - b - c \geq -2 \\ &a,b,c \in [0,1] \end{aligned} \quad (1)$$

For a one-input NOT gate with a as input and b as output, we have a + b = 1. We construct a set of clauses for each gate in the circuit and also an objective function for each gate in the circuit. The integer program for the whole circuit is merely:

Minimize: sum of objective functions, subject to
(Union of the Constraint sets for all the gates).

For an n-input, m gate circuit, it appears that the integer program has m+n 0-1 variables. However, in practice we can relax the integrality constraints on the internal variables of the circuit, as these are forced to become integral when the n primary inputs are integral.

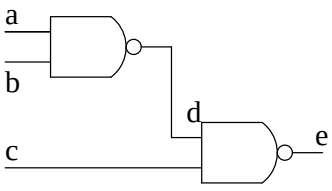

Fig 2: A circuit with NAND gates.

Let us consider the circuit in Fig 2 and the corresponding integer program. The objective function for this circuit reads as:

MIN : $(P_0 - P_1 - P_2 + P_3)(1 - d) + (P_1 - P_0)a + (P_2 - P_0)b + P_0 + (P_0 - P_1 - P_2 + P_3)(1 - e) + (P_1 - P_0)d + (P_2 - P_0)c + P_0$

The constraint set is simply the union of the constraint sets for the two NAND-gates in the circuit:

$$\begin{aligned} &a + d \geq 1 \\ &b + d \geq 1 \\ &-a - b - d \geq -2 \\ &d + e \geq 1 \\ &c + e \geq 1 \\ &-d - c - e \geq -2 \\ &a,b,c,d,e \in [0,1] \end{aligned} \quad (2)$$

For a NOT-gate with a as input and b as output, we have:

MIN: Q_0 a + Q_1 b subject to

$$\begin{aligned} &a + b = 1 \\ &a,b \in [0,1] \end{aligned} \quad (3)$$

The precise values we use for P_0, P_1, P_2 and P_3 are inspired by the observations of [3] as also our own Spice simulations. We note that, in their paper, P_3 is about 3.5 times the value of P_0, P_2 is about 2 times the value of P_0 and P_1 is about 1.5 times the value of P_0 .In our experiment, we use values of P_0, P_1, P_2 and P_3 in consonance with these facts. To calculate Q_0 we note that the leakage power Q_0 represents corresponds to the leakage power of a PMOS transistor in the off state, while P_3 corresponds to the leakage power of two PMOS transistors in parallel that are in the off state. Therefore we take Q_0 = 0.5 P_3. In [3], the leakage powers for 10 and 01 are different although both correspond to one NMOS transistor in the off state. This is because for P_1 the leakage power corresponds to a source-drain voltage drop of Vdd – V_T at the NMOS transistor in the off state, but for P_2 the leakage power corresponds to a source-drain voltage drop of Vdd at the NMOS transistor in the off state. We choose to take $Q_1 = P_1$.

Case (b): *Library consisting of arbitrary AOI gates.*

The model described above can be extended to treat more complex AOI gates as well. An example of an AOI gate is the following f : ~(a.b + c). Clearly the basic two-input NAND gate is also an AOI gate with f : ~(a.b). We work with standard series-parallel implementations of AOI gates. This means that the function f : ~(x.y + z) is implemented as shown in Fig 2. The problem in case of AOI gates arises from the fact that the objective function could include

many non-linear terms. To linearize the objective function we replace all the non-linear terms with new variables and then add additional constraints to establish the logical equivalence between the new variables and the terms they represent. This procedure will in general result in the addition of a large number of constraints but the number would remain manageable as long as each complex gate has only a few inputs. There are two ways of linearizing the objective function. We will show that while the linear relaxation of the first ILP always computes a trivial lower bound, that of the second ILP generally computes a better lower bound. We demonstrate the procedures for the function f : ~(x.y + z).

Formulation 1:

First we introduce a new 0-1 variable cube_1 representing the first cube xy. We introduce linear constraints expressing the logical equivalence, cube_1 $\Leftrightarrow$ xy. Thus,

$$\begin{aligned} &\text{cube\_1} \Rightarrow x \\ &\text{cube\_1} \Rightarrow y \\ &(x) \wedge (y) \Rightarrow \text{cube\_1} \end{aligned} \qquad (4)$$

In terms of linear constraints we have,

$$\begin{aligned} &-\text{cube\_1} + x \geq 0 \\ &-\text{cube\_1} + y \geq 0 \\ &-x - y + \text{cube\_1} \geq -1 \end{aligned} \qquad (5)$$

where x, y are required to be 0-1 variables. Having established the relationship between cube_1 and xy, we have f : ~(cube_1 + z). We must now introduce constraints to establish the relationship between f and the variables cube_1 and z. Clearly the following constraints perform this role:

$$\begin{aligned} &\text{cube\_1} \Rightarrow \sim f \\ &z \Rightarrow \sim f \\ &(\sim \text{cube\_1}) \wedge (\sim z) \Rightarrow f \end{aligned} \qquad (6)$$

In terms of linear constraints we have

$$\begin{aligned} &-\text{cube\_1} - f \geq -1 \\ &-z - f \geq -1 \\ &\text{cube\_1} + z + f \geq 1 \end{aligned} \qquad (7)$$

We have now introduced all the constraints needed to explain the logical behavior of the gate. We now turn our attention to the objective function. For each input vector, we have a specific leakage power. Let us denote all the eight leakage coefficients for the gate in question by P_0, … P_7. Next we introduce 0-1 variables for each input configuration such that when the input vector occurs then the variable is true. The first of these variables, aoi_0 represents the input vector xyz = 000. We express this in the form of the constraint aoi_0 $\Leftrightarrow$ (~x)(~y)(~z). In terms of linear constraints this can be expressed as:

$$\begin{aligned} &-x - \text{aoi\_0} \geq -1 \\ &-y - \text{aoi\_0} \geq -1 \\ &-z - \text{aoi\_0} \geq -1 \\ &x + y + z + \text{aoi\_0} \geq 1 \end{aligned} \qquad (8)$$

We introduce similar constraints for aoi_1 which represents xyz = 001, aoi_2 and so on upto aoi_7. We add a constraint for each gate to indicate that the leakage power for the gate must exceed the minimum leakage power for that gate.

$$P_0\ \text{aoi\_0} + P_1\,\text{aoi\_1} + \ldots + P_7\,\text{aoi\_7} \geq P_0 \qquad (9)$$

The objective function can then be written as:

$$\text{min: } P_0\ \text{aoi\_0} + P_1\,\text{aoi\_1} + \ldots + P_7\,\text{aoi\_7}.$$

Clearly for every assignment to the input variables x,y,z, one and only one variable in the objective function above is set to one picking the leakage power for that input assignment.

The above formulation works for all AOI gates. However, in case of two-input gates it is not as compact as the special formulation for two-input NAND gates. For an arbitrary AOI gate consisting of k inputs and r cubes, the formulation described in this section introduces $2^k + r$ new variables, $O(2^k + r)$ constraints and 2^k additional terms to the objective function.

For every input combination, exactly one aoi variable is set to 1 and the objective function evaluates to the leakage power associated with that variable. This ILP certainly computes the vector causing minimum leakage when it is solved exactly. But the linear relaxation of this formulation always computes a trivial lower bound on the leakage power dissipated by the circuit – namely, the sum of the minimum leakage powers of the gates in the circuit provided all gates in the circuit have two or more inputs.

Theorem 1: The optimum solution of the linear relaxation of the above ILP is always equal to the sum of the minimum leakage powers of the gates in the circuit.

Proof: Consider any constraint in the ILP formulation containing only non-aoi variables. It is of the form :

$$\Sigma C_i - \Sigma D_i \geq 1 - N \qquad (10)$$

which represents the logical constraint $C_1 \vee C_2 \vee \ldots C_M \vee \sim D_1 \vee \sim D_2 \vee \ldots \sim D_N$. Here M is the number of positive variables in the logical constraint and N is the number of

negated variables. Clearly setting all the variables C_i as well as all the variables D_i to 0.5 satisfies the above inequality. Also consider any inequality containing an aoi variable, say aoi_p_q (corresponding to the qth input combination of the pth gate). If the gate in question has n inputs then any inequality containing aoi_p_q either has two variables or n+1 variables. A two-variable inequality containing aoi_p_q is of the form $-v - aoi\_p\_q \geq -1$ or of the form $v - aoi\_p\_q \geq 0$, where v is an input of the pth gate. Since v is set to 0.5, we have in either case, $aoi\_p\_q \leq 0.5$. The inequality of n+1 variables in which aoi_p_q appears reads as $\Sigma C_i - \Sigma D_i + aoi\_p\_q \geq 1 - N$ which represents the logical constraint $C_1 \vee C_2 \vee \ldots C_M \vee \sim D_1 \vee \sim D_2 \vee \ldots \sim D_N \vee aoi\_p\_q$. Setting all the C_i and D_i to 0.5 this inequality becomes

$$aoi\_p\_q \geq 1 - 0.5*(M + N) \qquad (11)$$

where M + N = n, the number of inputs to the gate. This means that as long as we have only gates with two inputs or more in the circuit, this inequality is the same as $aoi\_p\_q \geq 0$.

Thus setting all circuit variables to 0.5 satisfies all the Boolean constraints and imposes constraints of the form $0 \leq aoi\_p\_q \leq 0.5$ on the aoi variables. Assuming that P_0 is the minimum leakage power for the gate, we can set $aoi\_p\_0 = 0.5$, $aoi\_p\_1 = (0.5*P_0)/(P_1)$ and the other variables aoi_p_2 … aoi_p_n to 0. Then the objective vector evaluates to P_0, for the gate p. The same argument can be extended to every gate in the circuit. Thus the optimum of the LP is the sum of the minimum leakage powers of all the gates in the circuit. END

Formulation 2:

We need to come up with an alternative formulation, which does not always compute as its minimum the trivial lower bound as in Formulation 1. It turns out that the formulation below works well in practice.

As in formulation 1 we create a variable for each cube and introduce constraints relating the variable to the elements of the cube. Thus for the gate f : ~(x.y + z) we have the following constraints:

$$\begin{aligned} &cube\_1 \Rightarrow x \\ &cube\_1 \Rightarrow y \\ &(x) \wedge (y) \Rightarrow cube\_1 \end{aligned} \qquad (12)$$

In terms of linear constraints we have,

$$\begin{aligned} &-cube\_1 + x \geq 0 \\ &-cube\_1 + y \geq 0 \\ &-x - y + cube\_1 \geq -1 \end{aligned} \qquad (13)$$

where x, y are required to be 0-1 variables. Having established the relationship between cube_1 and xy, we have f : ~(cube_1 + z). We must now introduce constraints to establish the relationship between f and the variables cube_1 and z. Clearly the following constraints perform this role:

$$\begin{aligned} &cube\_1 \Rightarrow \sim f \\ &z \Rightarrow \sim f \\ &(\sim cube\_1) \wedge (\sim z) \Rightarrow f \end{aligned} \qquad (14)$$

In terms of linear constraints we have

$$\begin{aligned} &-cube\_1 - f \geq -1 \\ &-z - f \geq -1 \\ &cube\_1 + z + f \geq 1 \end{aligned} \qquad (15)$$

The difference between formulations 1 and 2 arise in the way we create aoi-variables and the objective vector. For a 3-input gate the objective vector reads as:

min: P_0 x'y'z' + P_1 x'y'z + P_2 x'yz' + P_3 x'yz + P_4 xy'z' + P_5 xy'z + P_6 xyz' + P_7 xyz

Since x' = (1 – x), y' = (1 – y) and z' = (1 – z) we can expand the above to get:

min: $(-P_0 + P_4)$ x + $(-P_0 + P_2)$ y + $(-P_0 + P_1)$ z + $(P_0 - P_2 - P_4 + P_6)$ xy + $(P_0 - P_2 + P_3 - P_1)$ yz + $(P_0 - P_1 - P_4 + P_5)$ xz + $(-P_0 + P_1 + P_2 - P_3 + P_4 - P_5 - P_6 + P_7)$ xyz + P_0

We introduce variables for each of the terms in the above expansion, although we could skip the first three linear terms, namely x, y and z. Thus we have

$$\begin{aligned} &aoi\_1 = x \\ &aoi\_2 = y \\ &aoi\_3 = xy \\ &aoi\_4 = z \\ &aoi\_5 = xz \\ &aoi\_6 = yz \\ &aoi\_7 = xyz \end{aligned} \qquad (16)$$

We also introduce constraints to describe the relationship between the aoi variables and the input variables x, y and z in the same manner as in formulation 1. Unlike in formulation 1, in this case the minimum computed by the linear relaxation is not necessarily trivial. Computational experience shows that it always performs better than formulation 1. In fact we can show an interesting property of this relaxation for circuits consisting only of 2-input NAND gates and INVERTERS.

Theorem 2: If for any circuit consisting of 2-input NAND gates and INVERTERS, the LP relaxation returns a

fractional solution then the leakage power corresponding to that solution is definitely greater than the sum of the minimum leakage powers of all the gates.

Proof: The objective vector for a 2-input NAND gate reads as $P_0 + (-P_0 + P_1)a + (-P_0 + P_2)b + (P_0 + P_3 – P_1 – P_2)ab$. In formulation 2 we replace ab by the variable aoi_3 which is related to the variables a and b as

$$\begin{aligned} &a – aoi\_3 \geq 0 \\ &b – aoi\_3 \geq 0 \\ &-a – b + aoi\_3 \geq -1 \end{aligned} \qquad (17)$$

The first two inequalities above merely state that aoi_3 ≤ min(a,b). Without loss of generality assume that a > b. Then aoi_3 ≤ b. For a 2-input NAND gate P_0 is the minimum leakage value and P_3 is the maximum leakage value. Depending on the technology the coefficient of ab in the objective vector could be positive or negative. In case it is positive it is easy to see that the objective vector for any fractional solution is greater than P_0. In case it is negative then we have $E = P_0 + (-P_0 + P_1)a + (-P_0 + P_2)b + (P_0 + P_3 – P_1 – P_2)$ aoi_3 > $P_0 + (-P_0 + P_1)b + (-P_0 + P_2)b + (P_0 + P_3 – P_1 – P_2)b$.

$$E > P_0 + (P_3 – P_0)b \qquad (18)$$

If b > 0 then the objective vector is greater than P_0. For an inverter the objective vector $E > P_0 + (P_1 – P_0)a$, where a is the input of the inverter. Thus if the optimum solution of the linear relaxation is fractional then we have a non-trivial lower bound. END

The significance of the above theorem lies in the observation that solving the linear relaxation always benefits us. If the solution to the LP is fractional then we have a non-trivial lower bound that tells us something about the minimum leakage power dissipated by the circuit, and if the solution is non-fractional (or integral) then we have actually obtained a valid leakage vector of optimal cost.

3. AOI gates vs 2-input gates

It is not difficult to argue that the relative ease of the ILP formulation for 2-input gates (in terms of number of variables needed) comes at a certain price. In general, if minimizing leakage power is the criterion then 2-input gates are a poor choice for synthesis. This is because for a 2-input NAND gate (or an inverter) the ratio of maximum leakage power to minimum leakage power is usually no greater than 3. The maximum leakage power case occurs when both inputs to a NAND gate are high corresponding to both PMOS transistors in the pull-up network being in the OFF state. The minimum leakage power occurs when both inputs to the NAND gate are low corresponding to the two NMOS transistors in series in the pull-down network being in the OFF state. In case of an arbitrary AOI gate consisting of more than 3 NMOS transistors in series and more than 3 PMOS transistors in parallel, the ratio of maximum to minimum leakage grows larger. Thus the easier formulation of the ILP for 2-input gates translates into a less interesting search space while the harder ILP formulation for arbitrary AOI gates translates into a more interesting search space as the gains of leakage power minimization are greater in this case.

4. Computational Details

We use Formulation 2 for our experiments with circuits consisting of AOI-gates. Ideally one would solve the ILP formulation using a branch and bound procedure. While the special formulation for 2-input gates can be solved exactly using a branch and bound procedure for most of the standard benchmark circuits, branch and bound quickly becomes infeasible for the ILPs modeling arbitrary AOI gates. Hence we resort to an alternate approach that shows considerable promise.

It is known that for any 0-1 integer linear program, relaxing the restriction on the binary variables and letting them take any value between 0 and 1 produces a linear program, which is much easier to solve. Significantly for us, the optimum to the linear program is a lower bound on the sought after optimum to the ILP. However it could be a lower bound of poor quality in some cases and of very good quality in other cases. Generally speaking, to get good tight lower bounds, we must closely approximate the convex hull of feasible solutions. This is done by adding as many constraints as possible to the original ILP so that the resulting LP continues to contain all feasible solutions of the ILP and thus provides a lower bound to the ILP. In practice, however, one would not need to add too many constraints.

We now focus on a technique to obtain an actual assignment to the primary input variables so as to cause a leakage power dissipation that compares well with the optimum to the linear relaxation. This is done by rounding the value of each primary input variable computed by the LP to 0 or 1. If the value of a given primary input variable is p, $0 \leq p \leq 1$, we round the variable to 1 with probability p, and round it to 0 with probability (1 – p). Randomized rounding [8], as this technique is called, works reasonably well in other settings such as computing the maximum number of satisfiable clauses in a Boolean formula, computing the vertex cover of a graph of least weight and so on. In our case it works reasonably well. On most benchmark circuits, it provides us a vector that is within 5% of the true optimum. It needs to be emphasized here that the optimum computed by the linear relaxation is a leakage power value. It provides us with a good idea of the minimum leakage power dissipated by the circuit in question. Typically solving the linear relaxation is much

easier than solving the corresponding ILP. Therefore, for many circuits, we do not need to expend a lot of computational effort to ascertain the minimum leakage power dissipation.

5. Experimental Results

In Table 1 we give experimental results for benchmark circuits consisting of AOI gates. We use lp_solve, a mixed integer linear programming solver written by Michel Berkelaar [2] on the linear relaxations of the integer programming formulations for each circuit (using formulation 2). The optimum solution to the linear relaxation provides a lower bound on the optimum of the ILP. Column 3 in Table 1 gives us the time taken to generate the best leakage vector (i.e. of minimum leakage) in 100 iterations. As explained above each leakage vector is obtained by performing a randomized rounding operation on the solution vector for the linear relaxation. In column 5 we compute the error between the leakage power of the best vector that is computed after randomized rounding and the lower bound provided by the linear relaxation using the formula:

Error % = ((estimated minimum leakage after randomized rounding) – (LP_optimum))/(LP_optimum) * 100

It should be clear that the error figures in column 5 are an upper bound on the true error, since

$$\text{LP\_optimum} \leq \text{true minimum leakage power.}$$

This makes the small size of the error figures in column 5 of the table all the more remarkable. In most cases we are able to approximate the true minimum leakage power within the order of a few percent. For some circuits like i4, i6, i7 and C499, we are actually able to get an integer solution to the linear relaxation. So for these circuits, no time is expended in computing the best vector. Also the fact that integer solutions are obtained by solving the linear relaxation is a good statement on the tightness of the relaxation. For examples larger than those reported in Table 1, lp_solve reports numerical problems in some cases and gives exorbitant run-times in others. This might also be an indication of numerical difficulties in solving the linear program, which can just be due to lp_solve. Therefore a commercial solver might be able to deal with larger examples than lp_solve. This is a subject for further investigation.

6. Conclusions

We addressed the problem of finding an input vector to apply to a circuit consisting of and-or-invert (AOI) gates operating in stand-by mode so that the leakage power of the circuit is minimized. We formulated the problem as an integer linear program (ILP) in two different ways. Our solution scheme consists of relaxing the ILP formulation to obtain a lower bound on the minimum leakage power that is dissipated by the circuit. The linear relaxation of the first ILP formulation is shown theoretically to be of poor quality, while the relaxation to the second ILP formulation is shown to work much better in practice. To obtain a good input vector to apply to the circuit, we use a technique known as randomized rounding to round the solution of the linear relaxation [8]. This technique is seen to work quite well in providing an input vector of good quality. Further the solution technique developed here might also be of use in other ILP formulations of problems relating to digital circuits, for example the Boolean Constrained Optimization Problem (BCOP) of Chen and Keutzer [4].

Table 1: Estimates of leakage power values for benchmark circuits and their quality.

Circuit	# of inputs	Run Time For Best Vector	Run Time For LP (s)	% error
i2	201	77	12	0.23
i4	192	0	13	0
i5	133	55	29	3.31
i6	138	0	600	0
i7	199	0	357	0
i10	257	32400	25200	7.12
C499	41	0	25	0
C880	60	45.36	1852	4.22
C1355	41	105	65	4.16
C1908	33	410	869	7.61
C2670	233	802	480	6.88

References

[1] M.R.C.M. Berkelaar and J.A.G. Jess, “Technology Mapping for Standard-Cell Generators”, Proceedings, IEEE International Conference on Computer-Aided Design, 1988, pp 470-473.

[2] M.R.C.M. Berkelaar, a public domain mixed integer linear programming solver available at ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.

[3] S. Bobba and I. Hajj, “Maximum leakage power estimation for CMOS circuits”, Proceedings, IEEE Alessandro Volta Memorial Workshop on Low-Power Design, 1999, pp 116-124.

[4] P. Chen and K. Keutzer, “Towards True Crosstalk Analysis”, Proceedings, IEEE International Conference on Computer-Aided Design 1999, pp 132-137.

[5] Z. Chen, L. Wei, M.Johnson, and K. Roy, “Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks, Proceedings, International Symposium on Low-Power Electronics and Design, 1998, pp 239-244.

[6] J.P. Halter and F. Najm, “A gate-level leakage power reduction method for ultra-low-power CMOS circuits”, Proceedings, IEEE Customs Integrated Circuits Conference, 1997, pp 475-478.

[7] T.J. Larrabee, “Test Pattern generation using Boolean Satisfiability”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 11, No. 1, January 1992, pp 4-15.

[8] P.Raghavan and C.D.Thompson, “Randomized rounding: a technique for provably good algorithms and algorithmic proofs”, Combinatorica ,7:4, 1987, pp 365-374.

[9] H. Veendrick, Deep Submicron CMOS ICs, Kluwer Academic, 1998.

[10] Q. Wang and S.B.K. Vrudhula, “ Static Power optimization of deep submicron CMOS circuits for Dual V_T Technology, Proceedings, IEEE/ACM International Conference on Computer-Aided Design, 1998, pp 490-496.