Journal of
Low Power Electronics
Vol. 3, 1–8, 2007

Wakeup Scheduling in MTCMOS Circuits Using Successive Relaxation to Minimize Ground Bounce

Anand Ramalingam[1], Anirudh Devgan[2], and David Z. Pan[1,*]

[1] *Department of Electrical and Computer Engineering, The University of Texas, Austin, TX 78712, USA*
[2] *Magma Design Automation, Austin, TX 78759, USA*

(Received: xx Xxxx xxxx; Accepted: xx Xxxx xxxx)

Power gating is a very effective technique to reduce the subthreshold leakage by using sleep transistors to turn off the functional blocks or cells when they are not used. When the sleep transistors are turned on, the power grid may experience a huge current surge which may violate the integrity of the power grid. This paper addresses this problem by formulating the wakeup scheduling of sleep transistors as an exact mixed integer linear program (MILP). Since the resulting MILP is $\mathcal{NP}$-hard, we propose a very efficient yet near optimal algorithm by successively relaxing the MILP to a sequence of linear program (LP) problems. The results obtained on the ISCAS benchmarks indicate that our proposed algorithm obtains a near optimal solution with a speedup of 15× on average compared to the MILP. The proposed algorithm has a runtime complexity which is linear in practice.

Keywords: MTCMOS, Ground Bounce, MILP, Leakage, Sleep Transistors, Wakeup Scheduling.

1. INTRODUCTION

As technology scales, the supply voltage (V_{DD}) needs to be scaled down since it has a quadratic relationship with the dynamic power. But scaling down V_{DD} alone results in loss of performance. One way to maintain performance, is scaling down both V_{DD} and V_T.[1] However, scaling down V_T exponentially increases the subthreshold leakage current. One of the techniques to reduce subthreshold leakage is power gating. Power gating is a circuit technique in which a sleep transistor is inserted between the functional block and ground. The feasibility of the sleep transistors has been demonstrated in Refs. [1–4] through the reduction of leakage in fabricated chips.

In the *active* mode, the sleep transistor is turned on to retain the functionality of the circuit. In Ref. [5], it was shown that the sleep transistor can be approximated by a linear resistor that creates a finite voltage drop $V_{sleep} \approx R_{sleep} I(t)$, where $I(t)$ is the switching current through the sleep transistor. Sleep transistor sizing is one of the key issues in power gated circuits. If the sleep transistor size is overestimated, the silicon area is wasted and it also increases the switching energy. If the size is underestimated, the required performance might not be achieved due to the increased resistance to the ground.[5]

In the literature, a lot of work has been done to size the sleep transistor when the circuit is in the *active* mode. In Ref. [6], *module* based design was proposed where a single sleep transistor is used for the entire circuit. In Ref. [7], the circuit is partitioned into *clusters* to minimize the maximum simultaneous switching current. Each cluster has an individual sleep transistor. In Ref. [8], all the individual sleep transistors of Ref. [7] are wired together and the resulting mesh is called the *distributed* sleep transistor network (DSTN). The discharging current is shared by the sleep transistor network which reduces the size of the sleep transistor. The sizing in all the above methodologies are based on the *maximum* worst case switching current I_{peak}.[8] In Ref. [9], the sizing was improved due to an efficient vectorless estimate of the worst case switching current and timing criticality information obtained from a static timing analyzer.

In the *sleep* mode, the sleep transistor is turned off, and the source nodes of the gates in the functional block float, thus cutting off the leakage path to the ground. The only path that is left for the leakage current to flow is the parasitic capacitances of the transistor shown in Figure 1. Thus the leakage current starts charging up the parasitic capacitances of the transistors as shown in Figure 2.

When the circuit is turned on, these capacitances need to be discharged to get the circuit back to the normal mode of operation. This discharge of current causes ground bounce in the circuit shown in Figure 3. This ground bounce might

*Author to whom correspondence should be addressed.
Email: dpan@cerc.utexas.edu

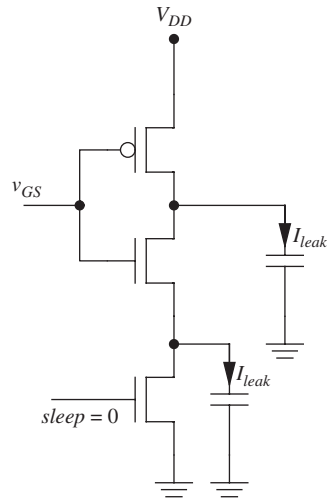


Fig. 1. Parasitic capacitances in a power gated inverter being charged up by the leakage current when the sleep transistor is off.

cause performance degradation or even worse, functional failures in the adjacent units.[10] The discharge of current also causes a huge surge in the power grid which might violate the reliability of the circuit.[11]

In contrast to the work in the active mode, there is relatively less work in the area of minimizing the ground bounce. In Ref. [10], to reduce the ground bounce, power gating structures were proposed in which sleep transistors are turned on in a non-uniform stepwise manner. The staggered switching of the sleep transistors reduces the ground bounce, is illustrated in Figure 3. A key observation from the figure is the tradeoff between the peak current and the time needed to switch on the circuit. In Ref. [11], the circuit is partitioned under the constraint of maximum current drawn from the power grid when the sleep transistor is switched on. The partitioning is done using a polynomial time heuristic.

In this paper, we assume that the gate can be turned on anytime subject to the fanin and current constraints. This assumption gives more freedom to schedule gates

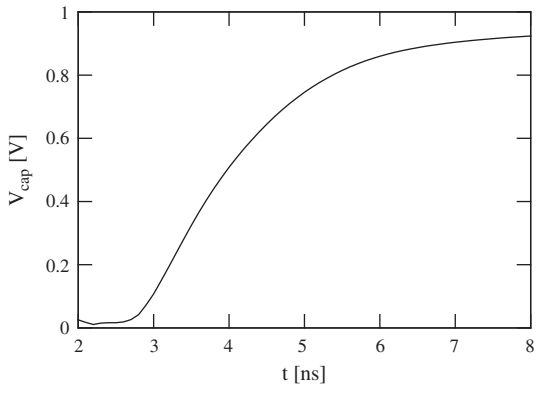


Fig. 2. A parasitic capacitor charged up to almost V_{DD} in *sleep* state. Since the leakage path to ground is cutoff, the parasitic capacitances get charged up. This simulation data is from a parasitic node in c499 under BPTM[18, 19] 45 nm technology with $V_{DD} = 1$ V.

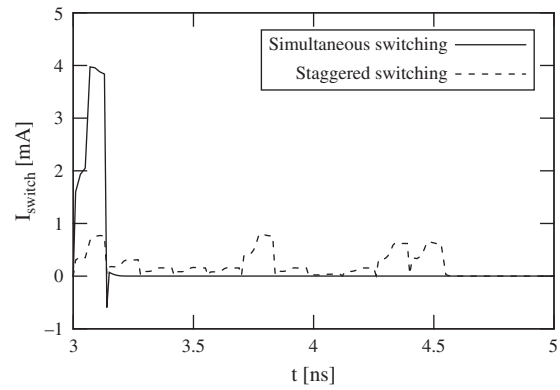


Fig. 3. Switching on sleep transistor(s) for c499. This simulation is done using BPTM 100 nm technology with $V_{DD} = 1$ V. In the *simultaneous switching*, there is a single sleep transistor for the entire circuit. We see a ground bounce of 4 mA around 3 ns, the time it switched on. In the *staggered switching* of c499, the gates in c499 are switched on at different times starting at 3 ns so that the constraint of drawing less than 1 mA is met.

than restricting a gate to a partition as in Ref. [11] which entails switching on a partition at the beginning of every clock cycle. Thus the tradeoff is between the time taken to wakeup and the granularity of the scheduling.

In this paper, we make the following contributions.

- We formulate the wakeup scheduling of sleep transistors as an exact Mixed Integer Linear Program (MILP).
- Since the resulting MILP is $\mathcal{NP}$-hard, we propose an algorithm which is computationally tractable. This is achieved by successively relaxing the MILP to a computationally efficient LP.
- The results obtained on the ISCAS benchmarks indicate that our proposed algorithm obtains a solution which are near optimal with a speedup of 15× on average compared to the MILP. The proposed algorithm has a runtime complexity which is linear in practice.

The rest of the paper is organized as follows. Section 2 states and formulates the problem as a MILP in a mathematically rigorous fashion. The successive relaxation algorithm to solve the MILP efficiently is presented next in the Section 3. The experimental results are presented in Section 4, followed by conclusion in Section 5.

2. OPTIMAL FORMULATION OF WAKEUP SCHEDULING

We assume that there is a sleep transistor for each gate in the circuit.[12] Each sleep transistor is controlled by an unique wakeup signal. An illustration for the inverter chain with each inverter having an individual sleep transistor is shown in Figure 4. The objective is to minimize the time to switch on the gates under a given current constraint of the power grid. We also assume that the gate can be turned on anytime subject to the fanin and current constraints.

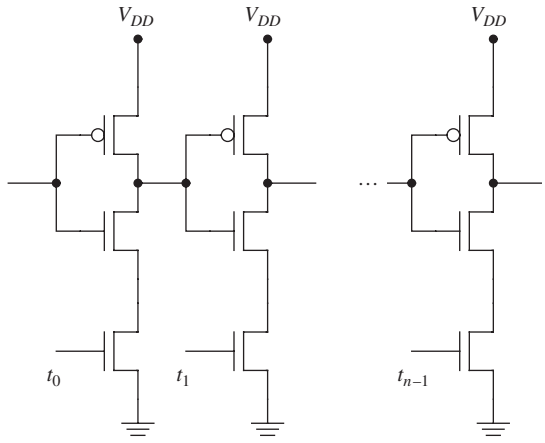


Fig. 4. Each inverter has an individual sleep transistor. Also note that each sleep transistor is controlled by an unique wakeup signal.

Problem Statement. Given a **combinational** circuit and a sleep transistor for each gate, minimize the time to switch on the sleep transistors under a given current constraint of the power grid.

Now we can mathematically formulate the problem. First, we define the data sets used in the model. The set G contains all the gates in the circuit. The set N is the discrete set of time points where a gate can switch. The reason why the time is discretized is described later in this section.

- $g \in G$ gates: instances of gates of many driving strengths
- $n \in N$ time: time n indexes a *discrete* set N

The gate switching is captured by a binary variable,

$$x_{gn} = \begin{cases} 1, & \text{if gate } g \text{ is switched at time } n \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Since a gate can be turned on exactly once,

$$\sum_{n \in N} x_{gn} = 1 \quad \forall g \tag{2}$$

Before turning on a gate g, we must turn on all the *fanin* gates of g. If we turn on a gate g before its fanin gates have settled, it might lead to spurious activity leading to dissipation of more power. Also, due to the fanin constraints when we turn on a gate driving a primary output we will have valid output.

$$\sum_{n \in N} n x_{g_j n} \geq \sum_{n \in N} n x_{g_i n} + d_{g_i} \quad \forall (g_i, g_j) \in G \tag{3}$$

where d_{g_i} is the delay of the gate g_i. In Ref. [11], the above constraint is captured as, a gate g should lie in a partition which is turned on later than its fanins partition. Since we are doing a fine grained scheduling, we need to a schedule a gate only after its fanins have settled down to a steady state value which explains the need for d_{g_i}, the delay of the gate g_i in Eq. (3). The current dissipated at time n,

$$i_n = \sum_{g \in G} \sum_{k=0}^{n} x_{gk} c_{g(n-k)} \quad \forall n \tag{4}$$

where c_{gn} is the current dissipated by the gate g at time n. The current is a data set and hence it is oblivious to the current characterization technique used. We used *FO4* load to characterize the current seen by a gate when it is switched on. The current at any given time $n \in N$ should be less than the given maximum $c_{\max}$,

$$i_n \leq c_{\max} \quad \forall n \tag{5}$$

Now we explain why the time is discrete. Suppose the time is a continuous quantity, the Eq. (3) can be written as,

$$t_{g_j} \geq t_{g_i} + d_{g_i} \quad \forall (g_i, g_j) \in G \tag{6}$$

where t_{g_j} is the time at which g_j can switch. But this leads to a need for boolean variable in modeling the index of the current in Eq. (4). Thus we need to discretize the time points. Now that we have justified discretizing time, we will finish our formulation by modeling our objective of minimizing the number of switching time points. Let us introduce another variable z to model this,

$$z \geq n x_{gn} \quad \forall g, n \tag{7}$$

which makes our objective,

$$\min z$$

Putting everything together,

$$\begin{aligned} &\min z \\ \text{s.t.} &\sum_{n \in N} x_{gn} = 1 \quad \forall g \\ &\sum_{n \in N} n x_{g_j n} \geq \sum_{n \in N} n x_{g_i n} + d_{g_i} \quad \forall (g_i, g_j) \in G \\ &i_n = \sum_{g \in G} \sum_{k=0}^{n} x_{gk} c_{g(n-k)} \quad \forall n \\ &i_n \leq c_{\max} \quad \forall n \\ &z \geq n x_{gn} \quad \forall g, n \\ &x_{gn} \in \{0, 1\} \quad \forall g, n \end{aligned}$$

This MILP formulation results in an optimal solution for the minimum wakeup time to switch on the gates under the given fanin and current constraints. The MILP turns out to be $\mathcal{NP}$-hard and it can be reduced from the Max-Weighted-Clique,[13] a known $\mathcal{NP}$-hard problem. We propose a near optimal successive relaxation algorithm in the next section.

3. SUCCESSIVE RELAXATION ALGORITHM

In this section, we describe the proposed successive relaxation algorithm. The optimal MILP formulation becomes

an LP if we relax the binary constraint $x_{gn} \in \{0, 1\}$ to a continuous constraint $0 \leq x_{gn} \leq 1$. The key idea behind the algorithm is using this LP as a subroutine. Since we are solving an LP it is faster to solve and scales much better. During every solve of the LP, the variables x_{gn} are guided to 0 or 1 depending upon if they are below or above the threshold respectively. The algorithm is presented in Algorithm 1.

Algorithm 1. Minimize Wakeup Time

Input: G gates, N switching time points, maximum current $c_{\max}$
Output: G gates *scheduled* time for wakeup
1: build and solve the LP got by relaxing the MILP
2: *scheduled* $\leftarrow \phi$
3: *ready* $\leftarrow$ gates driven by primary inputs
4: **while** all the gates are **not** *scheduled* **do**
5: **for** every gate g in *ready* **do**
6: **if** one of x_{gn} is above threshold **then**
7: Add a new constraint to the LP by setting the x_{gn} above threshold to 1
8: **if** the new LP is feasible **then**
9: transfer the gate g from *ready* to *scheduled*
10: **for** every successor gate g_s of gate g **do**
11: **if** all the fanins of g_s in *scheduled* **then**
12: add gate g_s to *ready*
13: **else**
14: Remove the constraint in line 7
15: Increase the number of switching time points N
16: Incrementally solve the new LP
17: **else**
18: guide it to threshold by narrowing the time window in which the gate can switch

To give a concise summary of the Algorithm 1, first we build and solve the LP got from relaxing the MILP model. On solving the LP model we get x_{gn} to be a continuous value between [0, 1]. The rest of code attempts to constrain it to a binary value.

The data structure *scheduled* contains the gates g and time index n such that $x_{gn} = 1$. The data structure *ready* contains the gates whose fanins are in *scheduled* and hence ready to be scheduled now. The core of the algorithm involves bringing gates from *ready* to *scheduled* after solving the LP. The core of the algorithm can be further broken down into two cases,

For a gate $g \in$ *ready*,
(1) If there exists a k, such that x_{gk} is greater than the *threshold*. This case constitutes the lines 6 to 16 Algorithm 1. This is further explained in Section 3.1.
(2) If there exists *no* k, such that x_{gk} is greater than the *threshold*. This case constitutes the line 18 in Algorithm 1. This is further explained in Section 3.2.

3.1. Above Threshold

For a gate $g \in$ *ready*, if there exists k such that x_{gk} is greater than the *threshold*, we are almost close to scheduling the gate g. The value of the *threshold* is set to be 0.5. The value for threshold was tuned after running several experiments and 0.5 gave the best tradeoff between runtime and the optimal solution.

We force $x_{gk} = 1$ and solve the constrained LP. If the LP is *feasible*, we have scheduled the gate g at time k. Once a gate is scheduled, we check if its successors can be added to *ready*. If the constrained LP is *infeasible* then we need to remove the constraint and increase the number of switching time points. We increase the switching time points to give the LP a chance to spread the successor gates away from this time k and give the gate g a greater chance of being scheduled here. But this case of infeasibility occurs rarely and if we have a good guess of the number of switching time points its occurrence is almost avoided.

A good starting value for the number of switching time points is obtained by multiplying the logic depth with the average delay of the gates in the library and scale it by a constant. If that value turns out infeasible, we can use the standard technique of bisection to efficiently find the lowest value which makes the LP feasible.

3.2. Below Threshold

For the gates $g \in$ *ready* and having all their x_{gn} below the *threshold*, we need to narrow the time over which they switch so that we will get atleast one k such that x_{gk} is greater than the *threshold*. We illustrate this idea using an example.

To illustrate the algorithm we will use a 1-bit CLA shown in Figure 5 as our input circuit. Initially, *ready* contains $\{g_0, g_1\}$, since they are the only gates having primary inputs as fanin. Suppose after solving the LP, we find that g_0, g_1 can be *scheduled* at time $n = 0$. Now the variables $x_{g_0 n|_{n=0}}, x_{g_1 n|_{n=0}}$ are equal to 1. All the other variables $x_{g_0 n}, x_{g_1 n}, n \neq 0$ equal 0. Observe that the variables $x_{g_0 n}, x_{g_1 n}$ are now reduced to constants thus reducing the problem size of the LP. Now the gates g_2, g_3 can be put in *ready* since all their fanins have been *scheduled*.

We now illustrate the concept of time narrowing which guides the LP to get atleast one k such that x_{gk} is greater than the *threshold*. Let none of the $x_{g_2 n}$ be above the threshold. Since the only fanin of the gate g_2 is g_0, the Eq. (3) must be satisfied. Let the delay of g_0, $d_{g_0} = 10$ *units*,

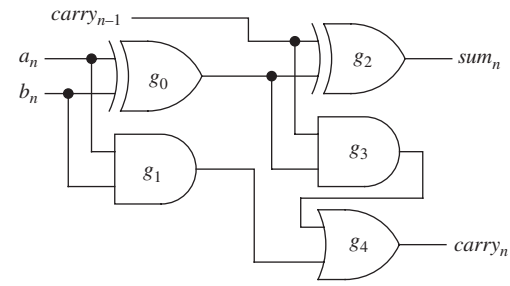
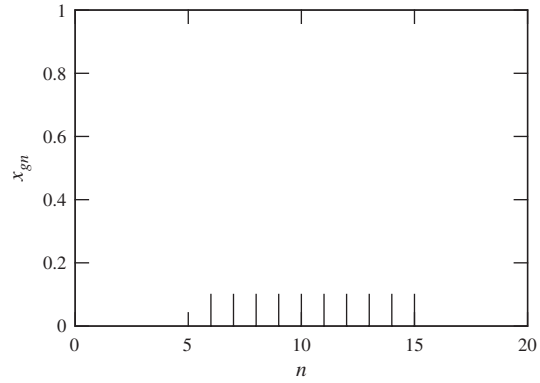
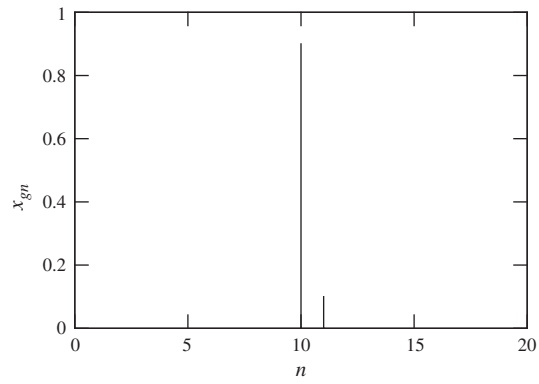


Fig. 5. 1-bit CLA for illustrating the algorithm.

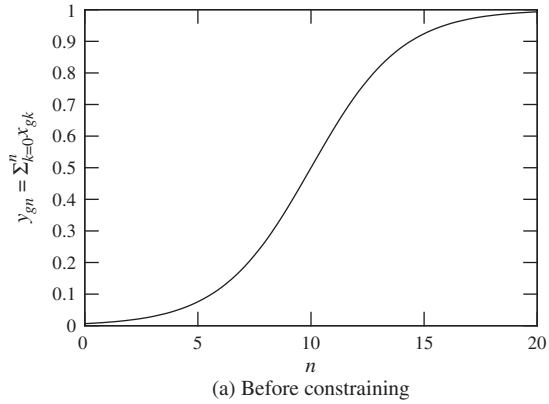


(a) Before constraining

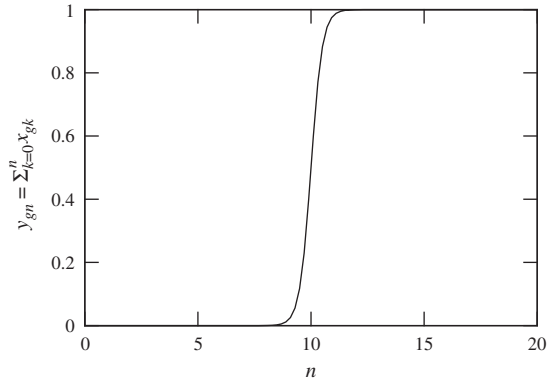


(b) After constraining

Fig. 6. Illustrates the concept of narrowing time window using gate g_2 in Figure 5. Let *threshold* = 0.5. Figure 6(a) shows that none of the $x_{g_2 n}$ are above the *threshold*. After we constrain that g_2 to switch between [10, 12], we see that $x_{g_2 n|_{n=10}}$ is above the *threshold* in Figure 6(b). Another way to look at the same concept is shown in Figure 7.



(a) Before constraining



(b) After constraining

Fig. 7. This figure shows a another way to look at the concept of narrowing time window shown in Figure 6. If x_{gn} can be thought of as a probability density function (*pdf*), then y_{gn} can be thought of as the cumulative density function (*cdf*). Figures 7(a), (b) can be thought of as the *cdf* of the *pdf* in the Figures 6(a), (b) respectively. The idea of narrowing the time window can viewed as making the *cdf* as close to a step function as possible.

where a *unit* is 10 ps. Since g_0 is scheduled at 0 and $d_{g_0} = 10$, the Eq. (3) reduces to,

$$\sum_{n \in N} n x_{g_2 n} \geq \sum_{n \in N} n x_{g_0 n} + d_{g_0} \geq 0 + 10 \tag{8}$$

As shown in Figure 6, gate g_2 switching spans time units [6, 15] and $x_{g_2 n} = 0.1, n \in [6, 15]$. The Eq. (8) now reduces to,

$$\sum_{n=6}^{15} n \times 0.1 \geq 10$$

$$10.5 \geq 10$$

We have now satisfied the Eq. (3) while having all x_{gn} below the *threshold*. Another way to look at this is shown in Figure 7. If x_{gn} can be thought of as a probability density function (*pdf*), then $y_{gn} = \sum_{k=0}^{n} x_{gk}$ can be thought of as the cumulative density function (*cdf*). The idea in this algorithm is to make the *cdf* as close to a step function as possible. After the first solve, the *cdf* looks to be a slow ramp function. To make it closer to a step function, we need to restrict the time window over which g_2 can switch. Since the delay of the fanin gate g_0 is 10 *units*, gate g_2 can not switch before 10. So we can set the $x_{g_2 n} = 0$, $n \in [0, 9]$, thus increasing the lower bound. We need to decrease the upper bound, and let us be aggressive in setting the upper bound to 12. After an incremental LP solve, we find that the *cdf* is sharper and it is brought out clearly in Figure 6(b) where $x_{g_2 n|_{n=10}} = 0.9$ and is above the *threshold*. Another way to look at this is shown in Figure 7(b) where the *cdf* has a sharper transition compared to the transition in Figure 7(a). Now $x_{g_2 n|_{n=10}}$ can be forced to 1 and the LP resolved. This process is repeated till all the gates are *scheduled*. It is important to notice that we intelligently guided the LP to get a x_{gn} above the *threshold* by increasing the lower bound and decreasing the upper bound.

Suppose this narrowing turned the problem to be infeasible, we increase the timing window till the LP is feasible again. If we again end with all the x_{gn} to be less than the *threshold* in the new LP, we scale the number of switching time points and repeat the algorithm. The argument for increasing the switching time points is same as the one presented in Section 3.1.

The next section presents the results after the proposed algorithm was run on the ISCAS benchmarks 14.

4. EXPERIMENTAL RESULTS

The proposed relaxation algorithm has been implemented in C++ and its results are presented for various benchmark circuits. We used the CPLEX library[15] to solve our LP model. The results were obtained after running on a 32-bit SPARC dual 1 GHz processor with 2 GB RAM. The current was characterized using the BPTM 100 nm technology with $V_{DD} = 1$ V. In Table I, we present the benchmarks on which our proposed algorithm was run. The table contains both the logic depth and the number of gates for a given circuit after it was mapped to our library. In Table II, we present the time to wakeup the benchmark circuit along with the runtime in seconds. The *FO4* delay in 100 nm is around 45 ps. If we assume the clock period to be $15 \times FO4 = 675$ ps,[16] then all the circuits except c6288 can be turned on within 6 clock cycles. In Table III, we compare the results obtained from the proposed algorithm and the optimal MILP algorithm and also their runtime. For c432, c499, c880, and c1355 the proposed algorithm is optimal. For c1908 it is within 1.6% of the optimal solution. Due to prohibitive memory and runtime we could not run MILP on the bigger benchmarks. In terms of runtime, the proposed algorithm is superior in almost all the cases by atleast 15×. The results were verified by doing random HSPICE simulations on the benchmarks. In Figure 8, we plot the runtime got by running our benchmarks against the gates ($|G|$)× switching time points ($|N|$). We observe that the runtime increases linearly with $|G||N|$, thus enabling the algorithm to scale well.

Table I. Logic depth and gate count of the ISCAS benchmarks after mapping to our library.

Circuit	Logic depth	Gates
c432	20	168
c499	11	202
c880	24	383
c1355	23	546
c1908	41	892
c2670	32	1193
c3540	47	1701
c5315	49	2311
c6288	124	2416

Table II. The time needed to switch on circuits under the maximum current constraint of 10 mA. We used 100 nm technology to generate the results. The runtime is also provided.

Circuit	t_{wakeup} [ns]	Runtime [s]
c432	1.85	2.72
c499	1	3.71
c880	2.05	22.92
c1355	2.3	60.77
c1908	3.2	256.13
c2670	2.7	492.32
c3540	3.7	1652.1
c5315	4.05	2128.1
c6288	12.2	5206.44

Table III. Comparison of the minimum wakeup time obtained by the proposed algorithm and the optimal MILP formulation.

	Proposed		Optimal MILP	
Circuit	t_{wakeup} [ns]	Runtime [s]	t_{wakeup} [ns]	Runtime [s]
c432	1.85	2.72	1.85	51.68
c499	1	3.71	1	44.48
c880	2.05	22.92	2.05	215.10
c1355	2.3	60.77	2.3	630.76
c1908	3.2	256.13	3.15	6066.79

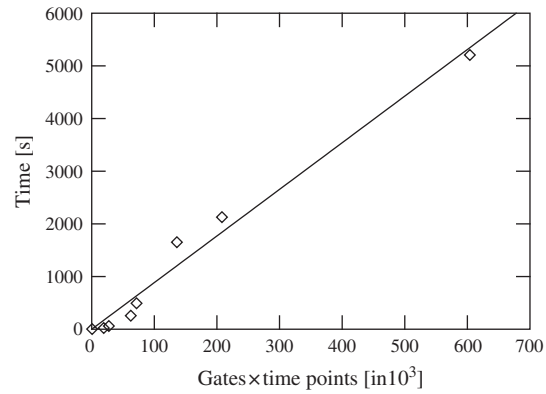


Fig. 8. Runtime complexity is $\approx O(|G||N|)$.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a near optimal algorithm for the wakeup scheduling of the sleep transistors. The proposed algorithm is based on the successive relaxation of the optimal MILP formulation of the wakeup scheduling. We demonstrate that across the ISCAS benchmarks our algorithm achieves a near optimal solution with a speedup of 15× on average compared to the MILP. The runtime is linear in gates ($|G|$)× switching time points ($|N|$) thus enabling scaling of the algorithm.

In the future, we plan to explore the formulation when the sleep transistors will be shared by many gates. One way to reuse the above algorithms is to partition the circuit into a set of supergates,[17] where supergate consists of many gates. Once we obtain this partition, we can apply the proposed Algorithm (1) to the circuit. The result obtained would be a sleep transistor for each supergate or in other words a sleep transistor which shared among several gates. The algorithmic complexity has a new component which accounts for the time to partition the circuit and generate supergates. Also, the proposed algorithm's complexity goes down since the number of supergates is always less than the number of gates.

Acknowledgments: Anand Ramalingam thanks Sreekumar V. Kodakara and Balasubramanian Srinivasan for their help in implementing the CPLEX part of the code. This work is partially sponsored by IBM Faculty Award. We used computers donated by Intel Corporation.

References and Notes

1. S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS, *IEEE J. Solid State Circuits* **(1995)**, Vol. 30, pp. 847–854.
2. S. V. Kosonocky, M. Immediato, P. Cottrell, T. Hook, and R. Mann, and J. Brown, Enhanced multi-threshold MTCMOS circuits using variable well bias, *Proceedings of the International Symposium on Low Power Electronics and Design* **(2001)**, pp. 165–169.
3. J. W. Tschanz, S. G. Narendra, Y. Ye, B. A. Bloechel, S. Borkar, and V. De, Dynamic sleep transistor and body bias for active leakage power control of microprocessors, *IEEE J. Solid State Circuits* **(2003)**, Vol. 38, pp. 1838–1845.
4. S. Henzler, T. Nirschl, S. Skiathitis, J. Berthold, J. Fischer, P. Teichmann, F. Bauer, G. Georgakos, and D. Schmitt-Landsiedel, Sleep transistor circuits for fine-grained power switch-off with short power-down times, *Proceedings of International Solid State Circuits Conference* **(2005)**, pp. 302–303.
5. J. Kao, A. Chandrakasan, and D. Antoniadis, Transistor sizing issues and tool for multi-threshold CMOS technology, *Proceedings of Design Automation Conference* **(1997)**, pp. 409–414.
6. J. Kao, S. Narendra, and A. Chandrakasan, MTCMOS hierarchical sizing based on mutual exclusive discharge patterns, *Proceedings of Design Automation Conference* **(1998)**, pp. 495–500.
7. M. Anis, S. Areibi, and M. Elmasry, Design and optimization of multithreshold CMOS (MTCMOS) circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **(2003)**, Vol. 22, pp. 1324–1342.
8. C. Long and L. He, Distributed sleep transistor network for power reduction, *Proceedings of Design Automation Conference* **(2003)**, pp. 181–186.
9. A. Ramalingam, B. Zhang, A. Devgan, and D. Z. Pan, Sleep transistor sizing using timing criticality and temporal currents, *Proceedings of Asia South Pacific Design Automation Conference* **(2005)**, pp. 1094–1097.
10. S. Kim, S. V. Kosonocky, and D. R. Knebel, Understanding and minimizing ground bounce during mode transition of power gating structures, *Proceedings of the International Symposium on Low Power Electronics and Design* **(2003)**, pp. 22–25.
11. A. Davoodi and A. Srivastava, Wake-up protocols for controlling current surges in MTCMOS-based technology, *Proceedings of Asia South Pacific Design Automation Conference* **(2005)**, pp. 868–871.
12. V. Khandelwal and A. Srivastava, Leakage control through fine-grained placement and sizing of sleep transistors, *Proceedings of the International Conference on Computer-Aided Design* **(2004)**, pp. 533–536.
13. M. R. Garey and D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA **(1979)**.
14. F. Brglez, P. Pownall, and R. Hum, Accelerated ATPG and fault grading via testability analysis, *Proceedings of the International Symposium on Circuits and System* **(1985)**, pp. 695–698.
15. *ILOG CPLEX Reference Manual*, ILOG Inc., **(2002)**.
16. D. G. Chinnery and K. Keutzer, Closing the gap between ASIC and custom: An ASIC perspective, *Proceedings of Design Automation Conference* **(2000)**, pp. 637–642.
17. J.-J. Liou, K.-T. Cheng, S. Kundu, and A. Krstic, Fast statistical timing analysis by probabilistic event propagation, *DAC '01: Proceedings of the 38th Conference on Design Automation* **(2001)**, pp. 661–666.
18. BPTM webpage, http://www-device.eecs.berkeley.edu/∼ptm.
19. Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation, *Proceedings of Custom Integrated Circuits Conference* **(2001)**, pp. 201–204.

Anand Ramalingam

Anand Ramalingam *received his B.E. degree from PSG College of Technology, Coimbatore, India and his M.S. degree from Stanford University in 2000 and 2003, respectively. He is currently pursuing the Ph.D. degree in Computer Engineering at The University of Texas at Austin. His research interests include timing analysis, circuit simulation, and leakage analysis.*

Anirudh Devgan

Anirudh Devgan *joined Magma in 2005 and manages product development for Magma's Custom Design Business Unit. He spent 12 years at IBM in various management and technical positions at IBM Thomas J. Watson Research Center, IBM Server Division, IBM Microelectronics Division and IBM Austin Research Lab. Devgan, named an IEEE Fellow in 2006, was the recipient of DAC's Best Paper Award in 2005, the IEEE William J. McCalla Best Paper Award in 2003, IBM Microelectronics Division's Excellence Award in 2001, the 2000 IBM Corporate Award, and the IBM Outstanding Innovation Award and IBM Outstanding Research Accomplishment Award, both in 1999. He has served on program committees for various international conferences including DAC, ICCAD, ASP-DAC, VLSI Design, and ISQED. Devgan has published more than 70 research papers and has 22 U.S. patents either issued or pending on various aspects of electronic design automation and circuit design. Devgan received a B. Tech. degree in Electrical Engineering from Indian Institute of Technology, Delhi, and M.S. and Ph.D. degrees in Electrical and Computer Engineering from Carnegie Mellon University.*

David Z. Pan

David Z. Pan *received his Ph.D. degree in Computer Science from University of California at Los Angeles (UCLA) in 2000. Prior to joining the UT faculty, he spent three years as a Research Staff Member at IBM T. J. Watson Research Center's VLSI Design Automation Department, where he contributed to various key aspects of Placement Driven Synthesis (PDS)—IBM's flagship design closure tool, which won IBM Research 2002 Outstanding Accomplishment. His research interests include: nanometer physical CAD, design for manufacturability (DFM), variation-tolerant designs, VLSI interconnects, novel circuitry and CAD for low power, vertical integration of architecture, circuit and technology. Dr. Pan is an Associate Editor for IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), and IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II), from January 2006. He is a member in the Design Technology Working Group of International Technology Roadmap for Semiconductor (ITRS). He has served in the Technical Program Committees of many key VLSI/CAD conferences, such as ICCAD, ISPD, DATE, ASPDAC, ISQED, ISCAS, SLIP, GLSVLSI, and ICICDT. He is the Program Committee Chair for ISPD 2007, CAD track Co-Chair for ISCAS*

2006 and 2007, Design of Reliable Circuits and Systems (DFR) Track Chair for ISQED 2007, Publicity Co-Chair for SLIP 2003, and Local Arrangement Chair for GLSVLSI 2002. He is a member of the ACM/SIGDA Technical Committee on Physical Design. He is a Technical Advisory Board member of Pyxis Technology, Inc. Dr. Pan has received the Best Paper in Session Award from SRC Techcon 1998, IBM Research Fellowship in 1999, Dimitris Chorafas Foundation Award in 2000, SRC Inventor Recognition Award in 2000, Outstanding Ph.D. Award from UCLA in 2001, IBM Research Bravo Award in 2003, IBM Faculty Award in 2004–2006, the ACM/SIGDA Outstanding New Faculty Award in 2005, and Best Paper Award Nominations at DAC 2006 and ASPDAC 2006. Dr. Pan is a Senior Member of IEEE, and a member of ACM/SIGDA, SPIE, and ASEE.