A Robust Power Gating Structure and Power Mode Transition Strategy for MTCMOS Design

Afshin Abdollahi
University of Southern California
afshin@usc.edu

Farzan Fallah
Fujitsu Labs. of America
farzan@us.fujitsu.com

Massoud Pedram
University of Southern California
pedram@usc.edu

Abstract - *The large magnitude of supply/ground bounces, which arise from power mode transitions in power gating structures, may cause spurious transitions in a circuit. This can result in wrong values being latched in the circuit registers. We propose a design methodology for limiting the maximum value of the supply/ground currents to a user-specified threshold level while minimizing the wake up (sleep to active mode transition) time. In addition to controlling the sudden discharge of the accumulated charge in the intermediate nodes of the circuit through the sleep transistors during the wake up transition, we can eliminate short circuit current and spurious switching activity during this time. This is in turn achieved by reducing the amount of charge that must be removed from the intermediate nodes of the circuit and by turning on different parts of the circuit in a way that causes a uniform distribution of current over the wake up time. Simulation results show that, compared to existing wakeup scheduling methods, the proposed techniques result in a one to two orders of magnitude improvement in the product of the maximum ground current and the wake up time.*

I. Introduction

The most obvious way of reducing the leakage power dissipation of a VLSI circuit in the STANDBY state is to remove its supply voltage. Multi-threshold CMOS (MTCMOS) technology provides low leakage and high performance operation by utilizing high speed, low V_t transistors for logic cells and low leakage, high V_t devices as sleep transistors. Sleep transistors disconnect logic cells from the power supply and/or ground to reduce the leakage in sleep mode. More precisely, this can be done by using one PMOS transistor and one NMOS transistor in series with the transistors of each logic block to create a virtual ground and a virtual power supply as depicted

in Figure 1. In practice only one transistor is necessary. Because of the lower on-resistance, NMOS transistors are usually used.

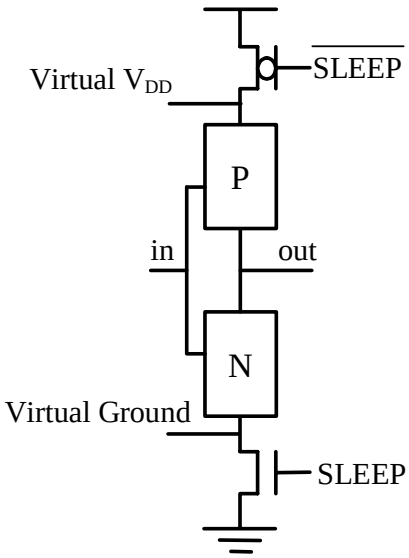

Figure 1: Power gating circuit.

In the ACTIVE state, the sleep transistor is on. Therefore, the circuit functions as usual. In the STANDBY state, the transistor is turned off, which disconnects the gate from the ground. To lower the leakage, the threshold voltage of the sleep transistor must be large. Otherwise, the sleep transistor will have a high leakage current, which will make the power gating less effective. Additional savings may be achieved if the width of the sleep transistor is smaller than the combined width of the transistors in the pull-down network. In practice, Dual V_T CMOS or Multi-Threshold CMOS (MTCMOS) is used for power gating [1][2]. In these technologies there are several types of transistors with different V_T values. Transistors with a low V_T are used to implement the logic, while high-V_T devices are used as sleep transistors.

To guarantee the proper functionality of the circuit, the sleep transistor has to be carefully sized to decrease the voltage drop across it when the sleep transistor is turned on. The voltage drop decreases the effective value of the supply voltage that the logic gate receives. In addition, it increases the threshold voltage of the pull-down transistors due to the body effect. This

phenomenon in turn increases the high-to-low transition delay of the circuit. The problem can be solved by using a large sleep transistor. On the other hand, using a large sleep transistor increases the area overhead and the dynamic power consumed for turning the sleep transistor on and off. Note that because of this dynamic power consumption, it is not possible to save power for very short idle periods. There is a minimum duration of the idle time below which power saving is impossible. Increasing the size of the sleep transistors increases this minimum duration.

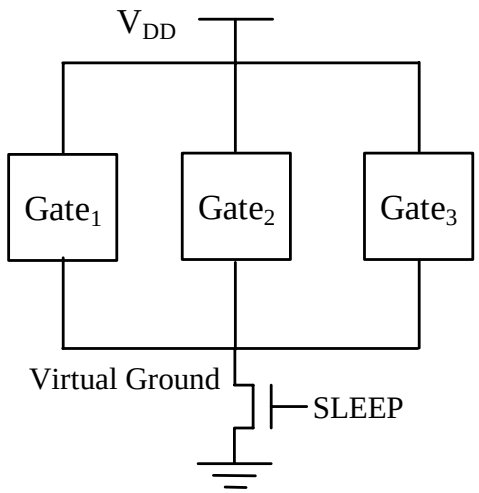

Figure 2: Using one sleep transistor for several gates.

Since using one transistor for each logic gate results in a large area and power overhead, one transistor may be used for each group of gates as depicted in Figure 2. Notice that the size of the sleep transistor in this figure ought to be larger than the one used in Figure 1. To find the optimum size of the sleep transistor, it is necessary to find the vector that causes the worst case delay in the circuit. This requires simulating the circuit under all possible input values, a task that is not possible for large circuits.

In this technology, also called *power gating*, wake up latency and power plane integrity are key concerns. Assuming a sleep/wake up signal provided from a power management unit, an important issue is to minimize the time required to turn on the circuit upon receiving the wake up signal since the length of wake up time can affect the overall performance of the VLSI circuit. Furthermore, the large current flowing to ground when sleep transistors are turned on can become a major source of

noise on the power distribution network, which can in turn adversely impact the performance and/or functionality of the other parts of the circuit. There is trade off between the amount of current flowing to ground and the transition time from the sleep mode to the active mode.

In this paper we introduce an approach for reducing the transition time from sleep mode to active mode for a circuit part while assuring power integrity for the rest of the system by restricting the current that flows to ground during the transition. The problem is to minimize the wakeup time while constraining the current flowing to ground during the sleep to active mode transition. During the process we will also consider another important objective which is limiting the number of sleep transistors. This paper is the extended version of the conference paper publication [3].

Section 2 describes the previous work. In Section 3 we present the key observations that our technique is based on. Section 4 presents problem statement and our method for graph modeling of the problem. A two step solution is offered in section 5 while an improved method is introduced in section 6. Alternative approaches for reducing the ground bounce are proposed in Section 7. Simulation results are presented in Section 8. Section 9 concludes the paper by briefly summarizing our results.

II. Previous Work

Optimal sizing of the sleep transistors for an arbitrary circuit to meet a performance constraint is an important design problem. Sleep transistors cause logic cells to slow down because of the voltage drop across the functionally-redundant sleep transistors and due to the increase in the threshold voltages of logic cell transistors as a result of the body effect. The performance penalty of a sleep transistor depends on its size and the amount of current that goes through it. In [4], sleep transistors are modeled as resistors and subsequently sized according to the following approximation for propagation delay: $T_{pd} \propto C_L V_{dd} / (V_{dd} - V_x - V_t)^{\alpha}$ where C_L is the total load capacitance, V_{dd} is the supply voltage, V_x is the voltage drop across the sleep transistor, V_t is the threshold voltage and α is a constant modeling the short channel effects. This delay model is used to bound the performance penalty for the worst case input vector. In [5], the authors propose a different method for sizing the sleep transistors. They first size the sleep transistor of each cell to limit the performance degradation to a specified level. Next, they merge sleep transistors whose discharge current patterns

are mutually exclusive based on a unit delay model. In [6], the authors use a more precise delay model to do the same steps. In [7], the authors propose a power gating structure to support an intermediate power-saving mode and a traditional power cut-off mode. The idea is to add a PMOS transistor in parallel with each NMOS sleep transistor whereby applying zero voltage to the gate of the PMOS transistor the circuit can be put in the intermediate mode. In the intermediate mode leakage reduction and data retention are realized. Furthermore, the magnitude of power supply voltage fluctuations during power-mode transitions is reduced by transitioning through this intermediate mode while changing between sleep and active modes. In the cut-off mode the gate of the additional PMOS transistor is connected to V_{dd}.

None of these works attempt to minimize the wake up time and the noise generated by the power gating structure and until recently only few researchers have addressed this problem. In [8] the authors introduce two power mode transition techniques to reduce the ground bounce while turning on the circuit. Instead of quickly turning on a large sleep transistor to suddenly reduce the resistance between the virtual ground and the (actual) ground, they propose to gradually reduce the resistance of the sleep transistor in order to limit the peak current flowing to the ground. This can be accomplished by employing one of the following two methods:

Parallel Sleep Transistors (Parallel-ST): Use of parallel-connected sleep transistors with gradually increasing widths (cf. Figure 3.) The sleep transistors are turned on in several time steps, starting from the smallest one. Since the voltage of the virtual ground is initially at its maximum value, a relatively high resistance value is used to discharge it; this limits the peak current. In the subsequent time steps, the resistance of the path between virtual and actual grounds is reduced by turning on wider sleep transistors.

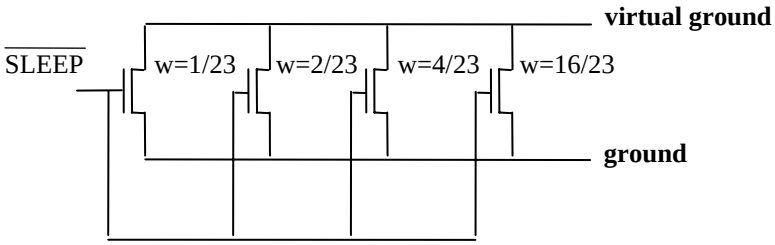

Figure 3. Power gating structure consisting parallel sleep transistors.

Staircase Sleep Signal (Staircase-SS): Use of a single sleep transistor, but turning it on gradually. Initially a voltage less than V_{dd} is used to weakly turn on the sleep transistor and thus, somewhat reduce the voltage of the virtual ground. In subsequent steps, the sleep transistor is turned on more strongly to further reduce the resistance between the virtual and actual grounds.

The two methods of [8] are restricted to using one sleep signal for the entire circuit block and provide only a temporal solution to the peak current flow problem. In contrast, in this paper, we provide an efficient *spatio-temporal solution* with its supporting power gating structure (i.e., with the ability to turn on different logic cells in the circuit block at different times.) This solution enables us to minimize the wake up time subject to an upper bound constraint on the total maximum current through the sleep transistors.

III. Key Observations

It is a well known fact that there is no need to have both NMOS and PMOS sleep transistors to encapsulate a logic cell. In particular, NMOS sleep transistors can be used to separate the (actual) ground from the virtual ground of the logic cell. Upon entering the sleep mode, a circuit block is disconnected from the ground. This causes the voltage levels of some intermediate nodes in the circuit block to rise toward V_{dd}. When the circuit block is woken up, the nodes will transition to zero. This transition in turn causes the logic cells in the immediate fanout of the node to carry a potentially large amount of short-circuit current as explained next. Consider the inverter chain shown in Figure 4, which is connected to the ground through an NMOS sleep transistor.

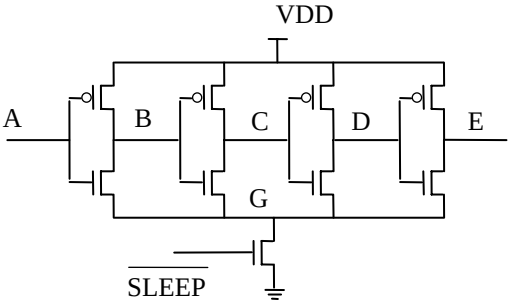

Figure 4. A chain of four inverters with an NMOS sleep transistor.

If the input of the circuit is low, then, in the active mode (i.e., SLEEP=0), $V_A=V_C=V_E=V_G=0$ and $V_B=V_D=V_{DD}$. When entering the sleep mode, the voltages of B and D do not change, but the voltages of C, E, and G gradually increase and will be equal to V_{DD} if the sleep period is long enough (note the driver of signal A is not controlled by the SLEEP signal). This happens because

the leakage through the PMOS transistors will charge up all the floating capacitances. Figure 5 shows the voltage waveforms of nodes C, E, and G generated by HSPICE simulation. While turning on the sleep transistor, nodes G, C and E discharge as depicted in Figure 6.

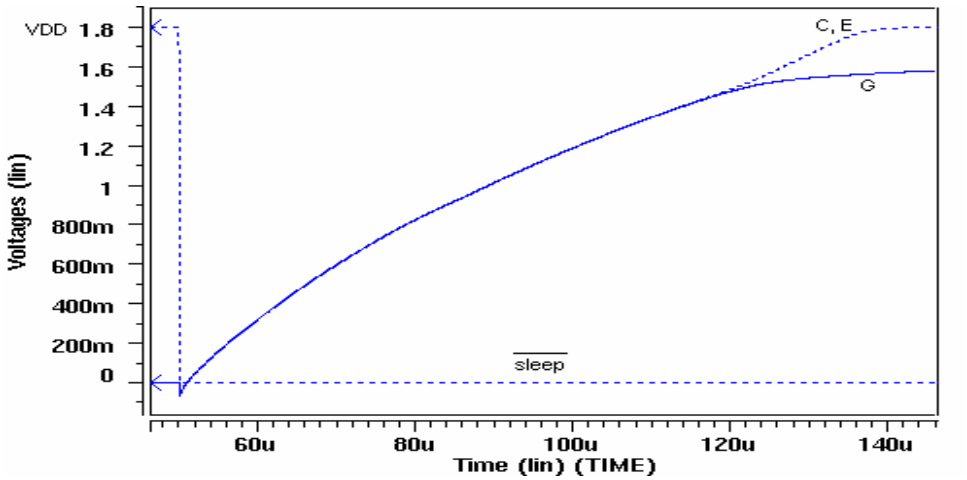

Figure 5. Voltage waveforms for nodes C, E and G of the circuit in Figure 2 when the circuit is in the sleep mode.

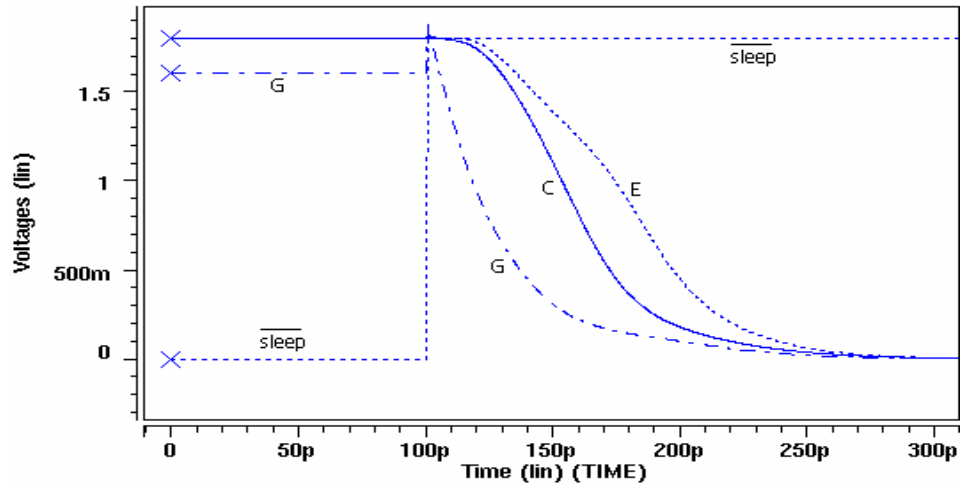

Figure 6. Voltage waveforms for nodes C, E and G of the circuit in Figure 2 when the circuit is transitioning from the sleep to active mode.

As one can see when the voltage of G quickly reaches its final value, the voltages of C and E are still between zero and V_{DD}. This results in a significant amount of short circuit current in the logic cells driven by nodes C and E since these nodes turn on both transistors of the inverters present in their fanout.

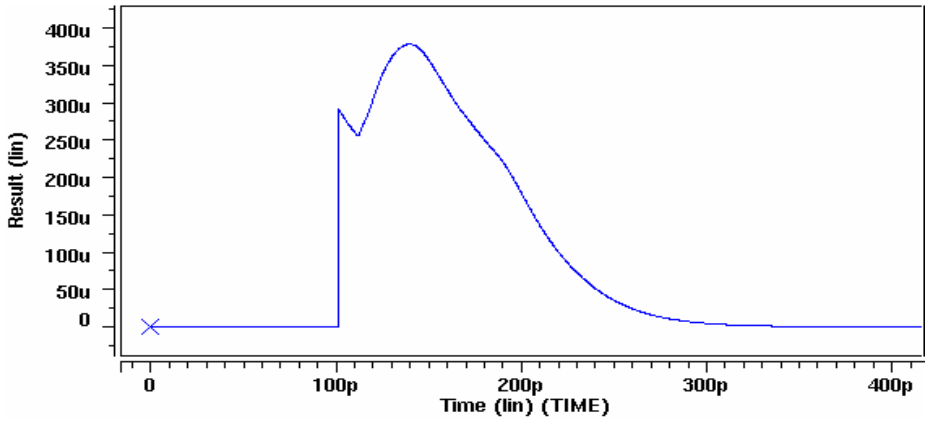

Figure 7. Total current flowing to ground while turning on the circuit.

The current shown in Figure 7 flowing through the sleep transistor is the result of not only discharging the accumulated charge in some intermediate nodes (i.e., C, E, and G in the inverter chain example), but also the short circuit current flowing through some logic cells of the circuit (e.g., the third inverter in the chain which is driven by signal C). The smaller the number of nodes that are discharged, the smaller the amount of current that flows to ground.

The objective is to design a power gating structure and a wake up strategy to minimize the wakeup time while constraining the current flowing to ground during the sleep to active mode transition. Our approach is driven by the desire to avoid short circuit currents and spurious transitions by turning them on at proper times. *The basic idea is to turn on each cell only if the voltage levels of the logic cells in its fanin have already reached their final values.*

Consider an inverter chain with one sleep transistor per cell as depicted in Figure 8.

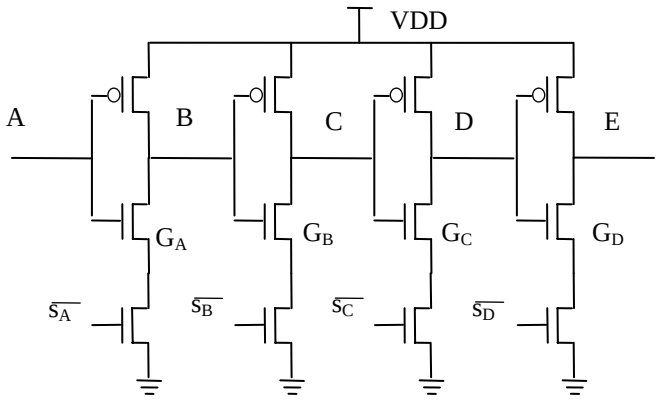

Figure 8. A chain of inverters with separate sleep transistors.

If we turn on the sleep transistors one at a time starting from the first inverter on the left, the short circuit current will be zero. The reason is that when each cell is turned on, its fanout cell continues

to stay in the sleep mode. Therefore, the possible transition of the output node of the logic cell does not result in any short circuit current in its fanout cell. Furthermore, there will be no spurious transition in the circuit since the inputs of the logic cells that have been turned on will not change at a later time. Figure 9 shows the total current flowing to ground while turning on the circuit of Figure 8 by employing this wake up strategy. As we can see, compared to the data of Figure 7, the maximum current in Figure 9 is reduced from 375µA to 280µA.

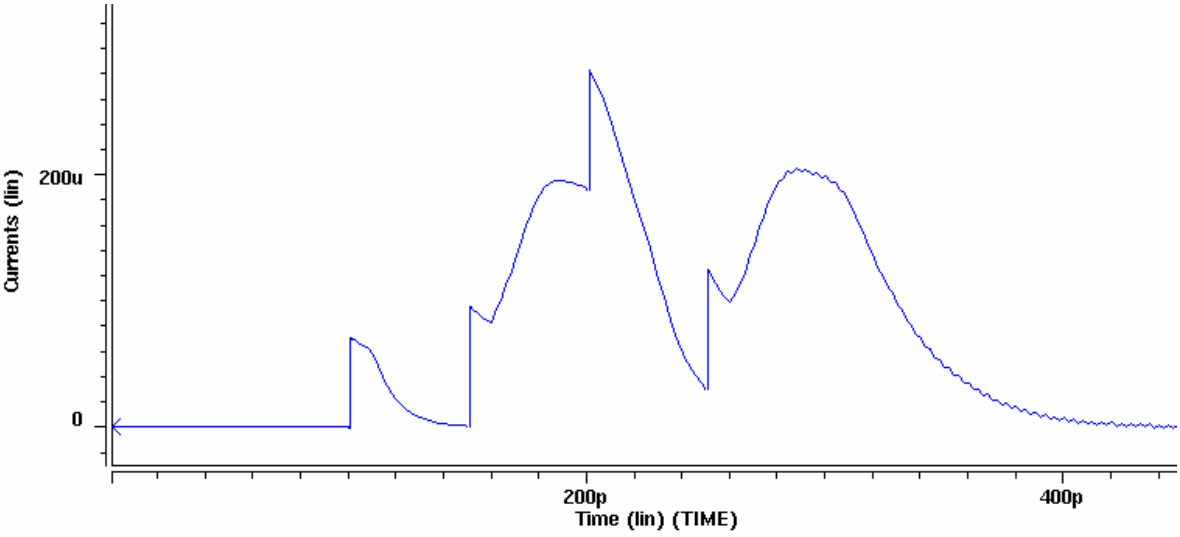

Figure 9. Total current flowing to ground while turning on the circuit.

There are several peaks in the current waveform of Figure 9. This is due to the fact that the sleep transistors are turned on in four steps. This obviously increases the wake up time. Note that in Figure 6, we can simultaneously turn on the first and third inverters before turning on the second and forth inverters without producing any short circuit current. This will reduce the wake up delay of the circuit. In the next section, we use a constraint graph to capture the zero-short-circuit current requirement for the power gating structure.

IV. Constraint Graph and Problem Formulation

It is assumed that the circuit has been in the sleep mode for a sufficiently long period of time (about 100µs as can be seen in Figures 3) so that the output voltages of all its logic cells have risen to their final steady state level. In addition the vector that is applied to the circuit primary inputs when entering the sleep mode and during that time period is known. This vector, which we call the *sleep vector*, remains unchanged during the wake up time.

The Constraint Graph, $G(V,E)$, is a weighted directed acyclic graph. Each vertex, v_i, in the graph corresponds to a logic cell in the circuit. There is a directed edge $e(v_i, v_j)$ from v_i to v_j exactly if v_j is in the immediate fanout of v_i and the output of v_i transitions from 1 to 0 during the circuit turn-on time under the specified sleep vector. There is a positive weight, $w(i, j) = T_{SETTLE}(v_i)$, associated with the edge $e(v_i, v_j)$, where $T_{SETTLE}(v_i)$ denotes the time required for the output of cell v_i to settle to its final value when its associated sleep transistor is turned on. Notice that $T_{SETTLE}(v_i)$ values could be incorporated in the graph as weights of nodes rather than edges. However, as will be seen later, these weights will be combined with another set of weights that should of necessity be defined on the edges. It is for this reason that we have defined $w(i, j)$ as edge weights.

Its value is calculated by circuit simulation as follows. Since the sleep vector is known and each logic cell is turned on only after all its fanin cells have settled to their final values, the input values of the logic cell are known at the time that the sleep transistor is turned on. Therefore, we can simulate the cell under the specific sleep vector value to find $T_{SETTLE}(v_i)$ and the current profile of the cell (i.e., $I_{TURNON}(v_i, t)$) after its sleep transistor is turned on at t=0. Notice that $I_{TURNON}(v_i, t) = 0$ for $t<0$ or $t>T_{SETTLE}(v_i)$. Furthermore, let's denote by $T_{TURNON}(v_i)$ the time at which the wakeup signal for turning on the sleep transistor associated with cell v_i arrives and refer to it as the *turn-on time* of cell v_i. To guarantee that there is no short circuit current during the wakeup time, the following constraint must be enforced:

$$T_{TURNON}(v_j) > T_{TURNON}(v_i) + T_{SETTLE}(v_i) = T_{TURNON}(v_i) + w(i,j) .$$

The contribution of v_i to the total discharge current at time t is $I_{TURNON}(v_i, t–T_{TURNON}(v_i))$ and the total turn-on current is $I_{TURNON}(t)= \Sigma I_{TURNON}(v_i, t–T_{TURNON}(v_i))$, where the summation is taken over all cells v_i. The total wakeup time, T_{WAKEUP}, is the time that the output last cell has settled to its final value i.e., $T_{TURNON} = max(T_{TURNON}(v_i)+ T_{SETTLE}(v_i))$ where the maximum is taken over all cells v_i.

The objective is to minimize T_{TURNON} while limiting the total turn-on current by a given threshold i.e., $I_{TURNON}(t)<I_{MAX}$. Logic cells with the same turn-on time can share a single sleep transistor. It is beneficial to limit the number of sleep transistors since it reduces the routing complexity. In our approach, we minimize the number of sleep transistors by matching the turn-on times of as many logic cells as possible without violating the aforementioned constraints. Consequently, all cells in the circuit are grouped into a minimum number of clusters. A single sleep transistor is allocated to each logic cell cluster, and a sleep/wake up signal is assigned to each such sleep transistor. The size of sleep transistors can be determined by using well known methods e.g., those in [5] and [6].

Clustering is done in such a way that the total turn-on current of each cluster does not exceed the given threshold I_{MAX}. Let $C_1, C_2, \ldots, C_M$ denote the clusters. Then the turn on current of cluster C_k is $I_{TURNON}(C_k, t) \sum_{v_i \in C_k} I_{TURNON}(v_i, t) < I_{MAX}$.

A necessary condition to prevent the flow of short circuit current during the wakeup time may be stated as follows. For cells v_i and v_j to belong to the same cluster, $T_{TURNON}(v_j) = T_{TURNON}(v_i)$ i.e., no edge can exist between v_i and v_j in the constraint graph. Unfortunately, this condition alone does not guarantee zero short circuit current. The sufficiency condition shall be described in the context of the *Cluster Constraint Graph* defined next.

We define a new directed graph, G_C, called the *Cluster Constraint Graph*. Vertices of this graph correspond to clusters $C_1, C_2, \ldots, C_M$. There is an edge from C_i to C_j in G_C exactly if there is at least one edge from some node of C_i to some node of C_j in the original constraint graph G. There is a positive weight associated with each edge in G_C. The edge weight is calculated as follows: $w(C_K,C_L) = max\{w(v_i,v_j)|v_i \in C_K, v_j \in C_L\}$. Clearly, if there is an edge $e(v_i, v_j)$ where both v_i and v_j are in the same cluster, their corresponding logic cells will be turned on at the same time, and the output of node v_i will be making a falling transition. Hence, a significant amount of short circuit current can flow through cell v_j (cf. discussion following Figure 6), which is undesirable. To avoid this scenario, we shall show below that there ought not to exist any directed path between any two vertices in the same cluster (this path evidently goes through some vertices outside that cluster.)

Although G is acyclic (assuming combinational logic circuits), there is no guarantee that a clustering solution will result in an acyclic G_C. An example is provided in Figure 10 where there exists a cycle between clusters C_K and C_L. Clearly, there is no way to schedule C_K and C_L to avoid short circuit current. If C_L is turned on after C_K, there will be a cell $v_d \in C_L$ driving another cell $v_b \in C_K$ which is already on. Therefore, cell v_b will consume short circuit current. A similar problem arises if C_L is turned on first. Hence, to avoid the flow of short circuit current during the circuit wakeup process, G_C must be acyclic. To achieve this requirement, the following constraint is enforced on the clusters. *If nodes v_i and v_j are in the same cluster C_k, then there ought not exist any directed path from v_i to v_j or vice versa in the constraint graph, G* .

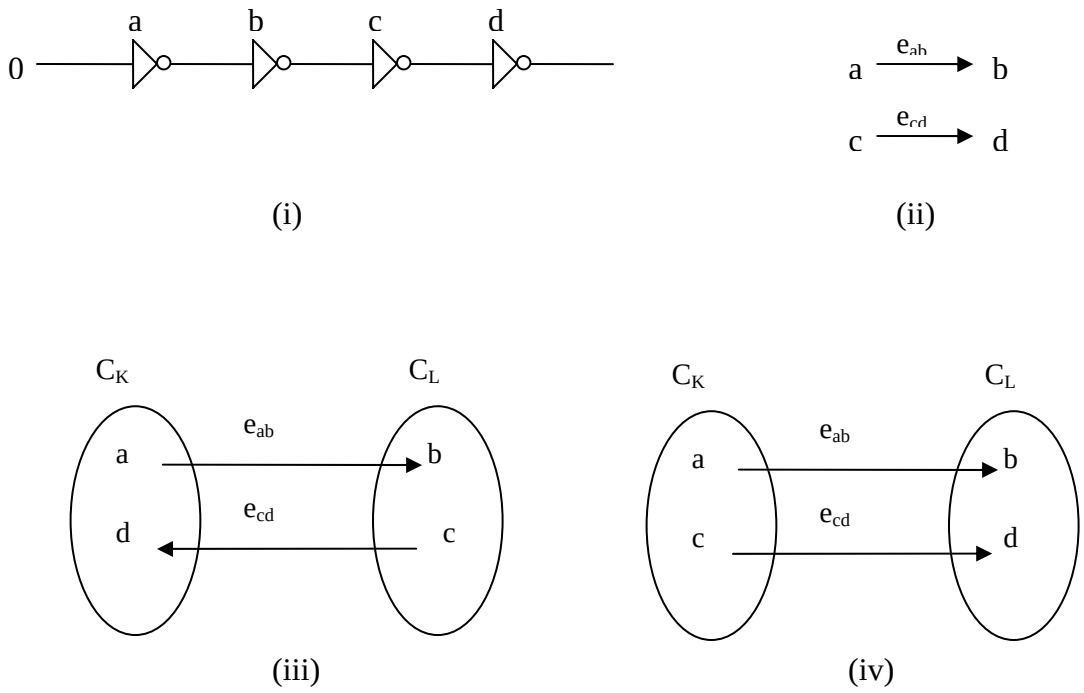

Figure 10. (i) An example of a circuit, (ii) The constraint graph G, (iii) A cyclic cluster constraint graph and (iv) A cycle free cluster constraint graph.

To better explain the clusters in the context of the constraint graph, we may construct $G'(V, E')$, the complement of graph $G(V, E)$, as follows. G' is an undirected graph with vertex set V. Furthermore, there exist an undirected edge $e'(v_i, v_j)$ between v_i and v_j in G' exactly if there exists no directed path between v_i and v_j in graph G. The vertices that belong to the same cluster will create a clique in graph $G'(V, E')$. A clustering of the cells in $G(V, E)$ corresponds to a partitioning of the vertices of $G'(V, E')$ into a number of cliques. For each clique in $G'(V, E')$, the summation of turn-on currents of cells that belong to that clique should not exceed I_{MAX} .

Consider a clustering of the cells where a sleep transistor is assigned to each cluster. The problem is to determine the turn-on times of the clusters so as to minimize the overall circuit turn-on time without causing short circuit current or violating the I_{MAX} limit.

The constraint imposed on the sleep (or wake up) signal scheduling by the presence of an edge $e(C_K,C_L)$ in G_C is:

$$T_{ON}(C_K)+w(C_K,C_L) \leq T_{ON}(C_L)$$

where $T_{ON}(C_K)$ and $T_{ON}(C_L)$ are the turn-on times of clusters C_K and C_L, respectively.

For a given ordering of clusters, $T_{ON}(C_1)<\ldots<T_{ON}(C_K)<T_{ON}(C_{K+1})< \ldots <T_{ON}(C_M)$, it may be possible to shift the current waveforms of two clusters $I_{TURNON}(C_K, t)$ and $I_{TURNON}(C_{K+1}, t)$ to overlap one another without violating the constraint $I_{TURNON}(t)<I_{MAX}$. The question is how close $T_{ON}(C_K)$ and $T_{ON}(C_{K+1})$ can be scheduled without violating the I_{MAX} constraint. To address this problem, we augment G_C with a new set of weighted directed edges $d(C_K,C_{K+1})$ as follows:

$$d(C_K,C_{K+1}) = min\{\Delta T\} \quad s.t. \; I_{TURNON}(C_K , t)+I_{TURNON}(C_{K+1}, t-\Delta T)< I_{MAX}$$

where $I_{TURNON}(C_{K+1},t-\Delta T)$ is the waveform $I_{TURNON}(C_{K+1}, t)$ shifted right on the time axis by an amount ΔT (cf. Figure 11).

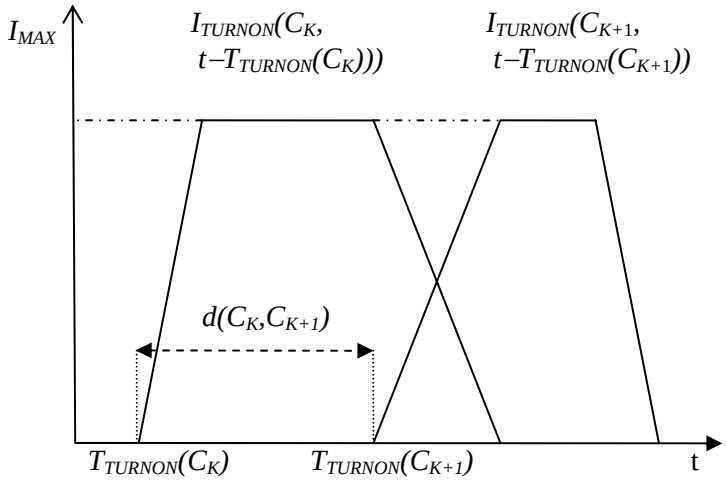

Figure 11. Shifting clusters turn on current waveforms.

Next, we combine edge weights $w(C_K,C_L)$ and $d(C_K,C_L)$ to construct new edge weights, which capture both constraints:

$$f(C_K,C_L) = max\{w(C_K,C_L), d(C_K,C_L)\}.$$

For the given ordering of M clusters, the minimum turn-on time can be described as $\sum_{K=1}^{M-1} f(C_K, C_{K+1}) + T_{SETTLE}(C_M)$, which is the weight of a path in graph G_C with edges $f(C_K,C_L)$ going through each vertex exactly once plus $T_{SETTLE}(C_M)= max\{T_{SETTLE}(v_i)|v_i \in C_M\}$.

The next section presents an algorithm to minimize the turn on time for an arbitrary combinational circuit.

V. Two Step Approach for Clustering and Scheduling

With these definitions and observations the precise problem statement is as follows:

Wakeup Signal Scheduling (WSS) Problem: *Cluster the logic cells into a minimum number of clusters and find the optimum turn-on times for logic clusters in the circuit so as to minimize the overall turn-on time* T_{TURNON} *of the circuit while eliminating the short circuit current and satisfying* $I_{TURNON}(t) < I_{MAX}$ *for all t.*

We propose an algorithm, called *Wakeup Scheduler* (WS), to solve the problem. The WS comprises of two steps:

a) **Logic Cell Cluster Generator:** We partition logic cells in the target circuit into a number of disjoint clusters C_1, C_2, …, C_M and assign *exactly one sleep transistor* with one sleep/wake up signal to all the cells in each cluster. The goal of clustering is to minimize the number of clusters, *M*, such that the total turn-on current flowing through the sleep transistors associated with each cluster, $I_{TURNON}(C_i)$, does not exceed I_{MAX}.

b) **Inter-Cluster Sleep Signal Scheduler:** Consider a single sleep signal that drives the sleep transistor of a cluster. The goal of wake up signal scheduling is to provide the ordering and relative timing of the activation signals for the *M* sleep signals in the circuit to minimize the overall wake up time while limiting the total current flowing to ground to I_{MAX}.

In this section, we solve the WSS problem by solving each of the clustering and scheduling problems separately and sequentially. Based on the discussion in the previous section, since short circuit currents can be avoided by an appropriate turn-on strategy, which in turn reduces the total I_{TURNON}, we do clustering and scheduling in a way that short circuit currents are eliminated.

Logic Cell Clustering (LCC) Problem: *Partition logic cells* $v_1, v_2, \ldots, v_N$ *into a minimum number, M, of clusters* $C_1, C_2, \ldots, C_M$ *such that there is no cycle in* G_C *and* $MAX(I_{TURNON}(C_K)) \leq I_{MAX}$*, for all k where,* $I_{TURNON}(C_k, t) = \sum_{v_i \in C_k} I_{TURNON}(v_i, t)$.

where summation is point-wise and MAX is taken over time. Note $I_{TURNON}(C_K,t)$ and $I_{TURNON}(v_i,t)$ represent the turn-on current waveforms, and not scalar current values.

LCC Algorithm

```
1 For all cells v_i in the circuit do {
2     For all clusters C_K created so far do {
3         If adding v_i to cluster C_K creates a cycle in G_C or
              violates the I_MAX threshold for C_K
4         Then goto step 2; // continue with the next cluster
5         Else {add v_i to cluster C_k; update G_C;
              goto step 1; // continue with the next cell }
          }
6     Create a new cluster and add v_i to it; Update G_C;
  }
```

Notice that we aim to minimize the number of clusters in order to reduce the number of sleep signals that are required in our proposed power gating structure. This will in turn simplify the power management circuitry. While using one sleep signal per cluster may seem costly, it is notable that in [10], a *sleep signal tree* (which is merely an inverter tree) similar to a clock tree has been proposed to drive large sleep transistors used in power gating structures. It is, therefore, possible to generate different timing for sleep signals going to different clusters by simply inserting delay elements (buffers) in the sleep signal tree.

***Sleep** Signal **Scheduling (SSS) Problem:** Determine $T_{ON}(C_k)$ values to minimize the total turn-on time subject to $I_{TURNON}(C_k, t) \leq I_{MAX}$ and $T_{ON}(C_k)+e(C_k,C_l) \leq T_{ON}(C_l)$ constraints.*

As described in the previous section, the minimum turn-on time can be described as $\sum_{k=1}^{M-1} f(C_K, C_{k+1}) + T_{SETTLE}(C_M)$, which is the weight of a path in graph G_C that goes through each vertex exactly once plus $T_{SETTLE}(C_M)$. To consider the settling time of the last cluster, we add a dummy vertex C_D to the graph with no outgoing edges and the following incoming edges, $f(C_K,C_D) = T_{SETTLE}(C_K)$ for all K.

A scheduling of the clusters corresponds to a Hamiltonian path in the cluster constraint graph. More precisely, the WSS problem may be stated as: "*Find the minimum weighted directed Hamiltonian path on graph G_C with edges $f(C_K,C_L)$.*" Recall that a Hamiltonian path between two vertices of a graph is one that visits each vertex of the graph exactly once [12]. Clearly, a Hamiltonian path of G_C must end up on dummy node, C_D, which has no outgoing edges. The start vertex can be any other node of G_C. There are many heuristics for solving the minimum Hamiltonian path problem, which is an NP-complete problem [12]. However, because the number of clusters is usually small

even for a large circuit, use of an exhaustive search for solving the minimum Hamiltonian path is also feasible.

The scheduling step results in the optimal turn-on times, $T_{TURNON}(C_K)$ for a given ordering of clusters. If the number of clusters is small, it is possible to exhaustively try all possible orderings, and thereby, find the best ordering. Otherwise, an ordering of clusters can be arbitrarily or heuristically selected. One heuristic could be as follows: Select an arbitrary cluster C_1 as the first cluster to be scheduled to wake up. Next find the next cluster C_2 that minimizes $d(C_1,C_i)$ (i.e., $d(C_1,C_2) \leq d(C_1,C_i)$ *for every i*) and continue in the same way (i.e., at step *k:* $d(C_k,C_{k+1}) \leq d(C_k,C_i)$.)

In practice since the shape of current profile of clusters is very similar, the initial ordering used in our algorithm is not important. Note that in our method by changing the value of the maximum current bound, the wake up time can be reduced.

VI. Simultaneous Clustering and Scheduling

In the previous section, we solved the WSS problem by solving each of the clustering and scheduling problems separately and sequentially. However, these two steps are not independent. The heuristic used for clustering returns only one solution out of many possible solutions. The result of scheduling depends on the solution provided by the clustering algorithm. The optimality of the overall result for the WSS problem depends on our choice of clustering solutions among many. However it is not practical to examine all clustering solutions to find the minimum turn on time. This is the main reason that in the previous we adopted the two step approach in which the clustering is done without any regard to its effect on the scheduling part.

In this section we propose a new technique where clustering and scheduling are done simultaneously and the overall objective (minimum turn-on) is targeted continuously throughout the algorithm. This technique is also a heuristic but tends to produce better results compared to the two-step approach (cf. our experimental results.)

The WSS problem is analogous to the task scheduling problem where the constraint graph corresponds to a data flow graph and logic cells correspond to tasks. This analogy is useful in developing our algorithm. The outline of the Simultaneous Clustering and Scheduling (SCS) is as follows. First we determine the set of all cells (vertices), S_1, that can be scheduled to turn on at t=0. S_1 includes those vertices that have no incoming edges in the constraint graph. We are to select a subset of S_1 to form the first cluster C_1 which is scheduled to turn on at t=0. If the total turn-on

current of cells in S_1 is less than I_{MAX}, then the first cluster C_1 will include all of the cells in S_1. Otherwise, we must select $C_1 \subset S_1$ to minimize $T_{SETTLE}(C_1)$. Next, we move on to cluster C_2 which will be scheduled to turn on at $T_{ON}(C_2) = f(C_1,C_2)$ where $f(C_1,C_2)$ was defined in section 4. Again, we identify the set, S_2, of all vertices that are not in C_1, but can be included in C_2. In particular, each vertex $v_j \subset S_2$ should either have no incoming edges or if there is an incoming edge $e(v_i,v_j)$, then v_i must have already been included in C_1. We select $C_2 \subset S_2$ to minimize $f(C_1,C_2)+T_{SETTLE}(C_2)$ (in general $f(C_k, C_{k+1})+ T_{SETTLE}(C_{k+1})$) subject to $I_{TURNON}(C_2, t) \leq I_{MAX}$. We continue in this manner to construct C_3, C_4, C_5, … until all cells have been placed in some cluster.

At k where C_1, C_2, …, C_{k-1} have already been computed, S_k is easily obtained as explained next. Recall that S_k includes all remaining vertices $v_j \notin C_1 \cup C_2 \cup \ldots \cup C_{k-1}$ that have no incoming edge or if there exist an edge $e(v_i,v_j)$, then v_i has already been included in previous clusters i.e., $v_i \in C_1 \cup C_2 \cup \ldots \cup C_{k-1}$. Now, we describe how to determine $C_k \subset S_k$ with the objective of minimizing $f(C_k, C_{k+1})+ T_{SETTLE}(C_{k+1})$ while not violating the constraint $I_{TURNON}(C_k, t) \leq I_{MAX}$. For each cell v_i in S_k if adding v_i to C_k does not violate the I_{MAX} constraint, we will include v_i in S_k. Otherwise (i.e., if adding v_i to C_k violates the I_{MAX} constraint,) we will consider replacing it with a cell $v_j \in C_k$ if such a replacement does not violate the I_{MAX} constraint and reduces $f(C_k,C_{k+1})+T_{SETTLE}(C_{k+1})$. In fact, we select $v_j \in C_k$ to be a cell in C_k that results in the maximum reduction in $f(C_k,C_{k+1})+T_{SETTLE}(C_{k+1})$ while not violating the I_{MAX} constraint. The pseudo code of the SCS algorithm is provided below.

SCS Algorithm

Initialize $V = \{v_1, v_2, \ldots, v_n\}$; $k = 1$;
While $V \neq \{\}$ **do** {
  $C_k = \{\}$; Create S_k;
  For all v_i in S_k **do** {
    If $max[I_{TURNON}(C_k \cup \{v_i\}, t)] < I_{MAX}$ {
      $C_k = C_k \cup \{v_i\}$;
    }
    Else {
      $v_m = arg\_min[f(C_{k-1}, C_k - \{v_j\} \cup \{v_j\}) + T_{SETTLE}(C_k - \{v_j\} \cup \{v_j\})]$ where *min* is taken over all v_j in $C_k \cup \{v_i\}$ that $max[I_{TURNON}(C_k - \{v_j\} \cup \{v_j\}, t)] < I_{MAX}$;
      $C_k = C_k - \{v_m\} \cup \{v_i\}$;
    }
  }
  $V = V - C_k$;
  $k = k+1$;
}

VII. Input-driven Sleep Transistor Typing

In the previous sections, we saw how the short circuit current can be avoided when turning on a circuit. Another approach to eliminating the short circuit current during the wakeup is to judiciously use an NMOS or a PMOS sleep transistor for each logic cell in the circuit (we call this technique *Input-driven Sleep Transistor Typing*, or ISTT for short.) The basic idea is that, for the given sleep vector, if the output of a logic cell in the circuit is logic 1, then an NMOS sleep transistor will be used to disconnect that cell from the ground; otherwise, a PMOS sleep transistor will be used to disconnect the output from V_{dd} as is shown in Figure 12.

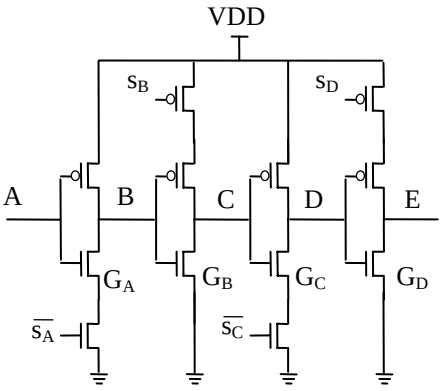

Figure 12. Using NMOS or PMOS sleep transistors.

With this simple ISTT algorithm, we ensure that every logic cell uses the type of the sleep transistor that minimizes the leakage current through the off-path of the logic cell through the well-known sleep transistor induced stack effect [11]. As a result, the output of every logic cell under the given sleep vector is driven to a *hard* zero or one logic level. Therefore, no logic cell will have a floating output node (which would have resulted in intermediate signal values changing during the sleep mode thereby causing a potentially large short-circuit current during transition to the wakeup mode.) Furthermore, in this case, the only floating nodes in the circuit are some of the internal nodes of logic cells (e.g., the shared diffusion area between source of the NMOS driver transistor and drain of the NMOS sleep transistor in the first stage of the inverter chain of Figure 12). These internal floating nodes can change during the sleep mode, and therefore, there will be some current dissipation on wakeup time to recover their correct values. However, this current is significantly less than the current that will flow thru the circuit when only NMOS sleep transistors are used. The reason, is that in the latter case, not only some of the internal nodes of logic gates are floating, but also, on average, half of the output nodes of the logic cells (which typically drive larger

capacitances), will be floating. Therefore, the peak current on circuit wakeup tends to be much larger than the case with ISTT.

The shortcoming of using NMOS sleep transistors for some gates and PMOS for others is that the delay overhead in the active mode is potentially twice that of the case with only NMOS sleep transistors. The reason is that in ISTT method, the delays of all logic cells on the critical path of a circuit are degraded, whereas the delays of only half of the logic cells are degraded when all sleep transistors are NMOS type. Notice that it is possible to combine the ISTT technique with the WS technique to achieve even better results by scattering in time the current that must flow to the ground, thereby, reducing the peak current; (ISTT+WS.)

VIII. Simulation Results

We used HSPICE to find the delay and current profile of each logic cell in a 0.09 μm standard cell library for a given sleep vector for all possible input combinations to the logic cell. Next we applied our algorithm to a number of circuits from the ISCAS test benchmark suite. Table 1 provides the amount of charge that is flowing to the ground during the wakeup time, and the amount of charge that is sourced from the power supply during the same time. We report results for the following techniques: a single (properly sized) NMOS device is used as the sleep transistor for the entire circuit (named Single-N), the two techniques of reference [8] (named Staircase-SS and Parallel-ST), the proposed WS (Wakeup Scheduler) with NMOS sleep transistors only (WS-N), the ISTT (Input-driven Sleep Transistor Typing), and finally the ISTT+WS technique. For each technique two columns of data are reported; one includes the amount of charge flowing to ground and the other one is the amount of charge coming from supply voltage. The last two columns in the table provide the number of clusters for the WS and ISTT+WS techniques. Note that for the latter, there are two types of clusters, one for the NMOS sleep transistors and one for the PMOS sleep transistors. Both data values are reported in the last column of Table 1 with a '+' separator. All data in the tables are generated with a 1.8V V_{dd}. The clock cycle time for each circuit was set to the worst cast delay.

Circuit	Charge Flowing to Gnd during Wakeup					Charge Coming from Vdd during Wakeup						# Clusters	
	Single ST		Staircase SS		Parallel ST	WS		ISTT		ISTT+ WS		WS	ISTT+ WS
9sym	8.8	1.8	8.0	1.1	8.1	1.1	7.9	1.1	1.7	3.5	1.4	3.0	9	3+6
C432	5.5	0.7	5.4	0.6	8.3	2.0	3.8	0.7	1.5	3.2	1.4	3.5	4	2+2
C1355	10.7	1	10.3	0.9	13.9	1.3	9.2	1.6	3.7	5.2	3.7	5.6	16	5+5
C1908	24.9	7.1	21.2	3.9	18.7	2.2	19.9	3.0	4.2	8.5	4.2	8.2	15	4+8
C2670	36.0	10.4	30.0	4.5	30.0	4.5	28.0	4.0	6.2	12.2	6.1	12.4	15	4+8
C3540	55.0	22.0	48.0	11.0	39.0	4.3	41.0	6.0	7.8	14.6	8.0	17.0	21	6+11
C5315	49.9	5.97	48.5	5.2	63.2	7.0	34.5	4.5	14.3	29.4	13.4	28.7	18	5+9
C6288	83.5	23.8	68.4	12.0	61.3	10.4	48.5	6.7	18.7	38.1	16.6	36.3	14	5+6
C7552	116	41.6	94.8	21.5	127	13.9	61.5	8.9	16.2	57.6	15.3	33.1	22	6+10

Table 1. Charge sinked to Gnd or sourced from Vdd (pico Coulombs.)

We observe that the ground current dominates the supply current for techniques that use NMOS sleep transistor only. Situation is reversed when both NMOS and PMOS sleep transistors are in use. This is because in the former case a lot more nodes (that were incidentally charged up during the sleep time) will have to be discharged to Gnd to assume their correct values at the onset of the circuit wakeup. Compared to previous techniques (Single ST, Staircase SS, and Parallel ST), the WS technique reduces the amount of charge flowing to the ground during wakeup. Further reduction can be achieved by using a mixture of NMOS and PMOS transistors in the circuit to reduce the number of internal nodes that need to be discharged during the wake up time. This is seen by the significant reduction in the amount of charge that is flowing to the ground for ISTT and ISTT+WS methods.

Table 2 shows the maximum current of the ground and supply lines for all of the above techniques. For Parallel-ST, we have used the worst case delay of the circuit as the period of a clock signal, which is then used to turn on the sleep transistors in multiple cycles. To make the comparisons meaningful, we set the I_{MAX} constraint for the WS algorithm to the best that is achieved by Parallel-ST and Staircase-SS. As one can see, our proposed techniques reduce the maximum current of ground more than any other technique. The I_{max} values for ISTT are higher than that for WS or ISTT+WS because we have not restricted the ground current for ISTT.

Circuit	$I_{Ground-max}$				$I_{Supply-max}$							
	Single ST		Staircase SS		Parallel ST		WS		ISTT		ISTT +WS	
9sym	132	9.0	22	1.0	48	1.3	22	3.3	53	87	19	196
C432	108	5.6	15	0.4	41	2.1	14	2.5	48	76	12	17
C1355	226	8.7	32	0.6	89	1.6	30	2.8	128	124	33	45
C1908	329	27.0	51	2.7	119	2.0	48	6.6	138	211	46	45
C2670	468	36.0	72	3.4	168	4.6	45	5.9	203	295	44	46
C3540	679	53.0	105	5.4	246	4.0	62	8.0	284	452	59	67
C5315	1025	47.6	144	3.4	398	6.2	125	13.7	463	718	119	116
C6288	1036	100.5	160	10.8	452	23.8	146	43.1	513	877	139	127
C7552	1391	117.2	214	11.4	703	6.5	197	20.9	811	1337	198	204

Table 2. Maximum ground and supply currents (in mA).

Table 3 shows the wakeup time and the product of the maximum ground current and the wake up delay for all techniques.

Circuit	T_{TURNON}				$I_{Ground-max} \times T_{TURNON}$							
	Single ST		Staircase SS		Parallel ST		WS		ISTT		ISTT +WS	
9sym	494	65	4000	88	4000	192	624	2.4	45	13.7	252	4.8
C432	240	26	7800	117	7900	323	854	12.0	46	2.2	350	4.2
C1355	132	30	6000	192	6500	579	861	5.4	42	25.9	212	7.1
C1908	267	88	8000	408	8000	952	797	6.1	44	38.3	324	14.9
C2670	578	270	9300	670	9400	1580	1070	9.3	46	48.2	422	18.6
C3540	1500	1019	12000	1260	12100	2952	1096	68.0	47	13.4	473	27.9
C5315	1320	1353	11000	1584	11200	4457	916	115	46	21.3	446	53.1
C6288	2100	2176	18000	2880	18500	8362	1430	209	45	23.1	457	63.6
C7552	2310	3213	20000	4280	21000	14763	1680	331	83	67.3	787	156

Table 3. Wake up time (in pico Seconds), and product of the maximum ground current and wake up time (in pico Coulombs).

In terms of the product of maximum ground current and wake up time, again our proposed techniques (WS and/or ISTT) are superior to the previous ones by between one to two orders of magnitude. Note that this means for a given maximum current threshold, the wake up delay of our technique is much smaller than the other methods. From the table it is also clear that using both NMOS and PMOS sleep transistors increases the maximum supply current, however, the amount of charge that is flowing to the ground is significantly reduced compared to Staircase-SS and Parallel-ST. Note that for all circuits, the wakeup time calculated by our proposed techniques was always less than one clock period.

We also used SCS algorithm for turning on the benchmark circuits. Table 4 compares the wakeup times and maximum ground currents of SCS and WS algorithms. The table shows that the SCS algorithm improved the wakeup delay by 10%-15% over the WS algorithm while maintaining approximately the same I_{MAX}. This result is expected since the SCS algorithm performs the clustering and scheduling simultaneously.

Circuit	WS		SCS	
	T_{TURNON}	$I_{Gnd\text{-}Max}$	T_{TURNON}	$I_{Gnd\text{-}Max}$
9sym	624	22	560	23
C432	854	14	768	15
C1355	861	30	740	30
C1908	797	48	710	50
C2670	1070	45	960	46
C3540	1096	62	956	62
C5315	916	125	885	124
C6288	1430	146	1240	144
C7552	1680	197	1450	192

Table 4. Wake up time (in pico Seconds), and maximum ground current (in mA.)

In another experiment for Parallel-ST and Staircase-SS methods, we uniformly distributed the sleep signal arrival times within a single clock cycle (in data reported above we used multiple clock cycles per ref. [8].) Next, we measured the maximum ground current and report the product of this current and the single-cycle wakeup time. The results are reported in Table 5.

Circuit	$T_{Wake\text{-}up}$		$I_{Ground\text{-}max}$		$I_{Ground\text{-}max} \times T_{Wake\text{-}up}$	
	Staircase-SS (single cycle)			Parallel-ST (single cycle)		
9sym	1300	80	104	1450	180	261
C432	2500	56	139	2600	189	491
C1355	1900	122	232	2150	338	726
C1908	2300	197	452	2500	580	1451
C2670	3000	237	710	3150	639	2012
C3540	4000	355	1420	4150	993	4119
C5315	3900	438	1710	4000	1422	5690
C6288	6000	603	3618	6150	1802	11080
C7552	6500	906	5890	6700	2803	18780

Table 5. Wake up time (pico Seconds), maximum ground current (mA) and their product (pico Coulombs).

Comparing the products of maximum ground current and wake up time of our method in Table 3 and those in Table 5, we conclude that our techniques maintain the advantage (between one and two

orders of magnitude) over Staircase-SS and Parallel-ST techniques even when they are implemented in a single cycle by between one and two orders of magnitude. Note that the wakeup times reported in Table 5 were calculated as the summation of the time required to apply the wakeup signals and the time required for all nodes in the circuit to settle.

IX. Conclusions

We introduced a new method for reducing the wake up time and maximum current flowing to ground for power gating structures. One of the proposed techniques is based on effectively clustering logic cells and scheduling wakeup signals for the clusters to achieve the mentioned objectives. The algorithms provided in this paper have low computational complexity and yet very effective. Experimental results for our methods showed between one and two orders of magnitude improvement in the amount of maximum current going to ground multiplied by the wake up time compared to the previous methods

Acknowledgment

The authors would like to thank Thomas Sidle, the VP of Advanced CAD Technologies group at Fujitsu Labs. of America for his support of this project.

References

[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 CMOS," *IEEE Journal Solid-State Circuits* 30, No. 8, August 1995, pp. 847–854.

[2] J. Kao, A. P. Chandrakasan, ``Dual-threshold voltage techniques for low-power digital circuits," *IEEE Journal of Solid-State Circuits*, Vol. 35, July 2000, pp. 1009-1018.

[3] A. Abdollahi, F. Fallah, and M. Pedram, "An Effective Power Mode Transition Technique in MTCMOS Circuits," *Design Automation Conference*, pp. 37-42, 2005.

[4] J. Kao, A. Chandrakasan and D. Antoniadis, "Transistor Sizing Issues and Tool for Multi-Threshold CMOS Technology," *Design Automation Conf.*, pp. 409-414, 1997.

[5] J. Kao, S. Narenda and A. Chandrakasan, "MTCMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Patterns," *Design Automation Conf.*, pp. 495 - 500, 1998.

[6] M. Anis, S. Areibi, M. Mahmoud and M. Elmasry, "Dynamic and Leakage Power Reduction in MTCMOS Circuits Using an Automated Efficient Gate Clustering Technique," *Design Automation Conf.*, pp. 480-485, 2002.

[7] S. Kim, S.V. Kosonocky, D. R. Knebel, and K. Stawiasz, "Experimental measurement of a novel power gating structure with intermediate power saving mode," *Intl. Symp. on Low Power Electronics and Design*, pp. 20-25, 2004.

[8] S. Kim, S. V. Kosonocky, Stephen, and D. R. Knebel, “Understanding and minimizing ground bounce during mode transition of power gating structures”, *Intl. Symp. on Low Power Electronics and Design*, pp. 22-25, 2003.

[9] Hyo-Sig Won, et al., “An MTCMOS Design Methodology and Its Application to Mobile Computing,” *Intl. Symp. on Low Power Electronics and Design*, pp. 110-115, 2003.

[10] Usami, et al., ”Automated Selective Multi-Threshold Design for Ultra-Low Standby Applications,” *Intl. Symp. on Low Power Electronics and Design*, pp. 202-206, 2002.

[11] M. Johnson, D. Somasekhar, and K. Roy, “Leakage Control with Efficient use of Transistor Stacks in Single Threshold CMOS,” *Design Automation Conf.*, pp. 442-445, 1999.

[12] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.