An Efficient Wake-Up Strategy Considering Spurious Glitches Phenomenon for Power-Gating Designs
Da-Cheng Juan, Yu-Ting Chen, Ming-Chao Lee, and Shih-Chieh Chang, Member, IEEE

Abstract — During the power mode transition, simultaneously turning on sleep transistors provides a sufficiently large surge current, which may cause a large IR drop in the power networks. The IR drop in turn causes errors in the retention sequential elements of the sleep modules or errors of the non-sleep modules. One efficient way to control the surge current is to schedule the turn-on sequences of sleep transistors. In this paper, we introduce several important properties of the surge current during the power mode transition for the Distributed Sleep Transistor Network (DSTN) design, which is a popular power-gating design style. Based on these properties, we propose an accurate estimation of the surge current and provide efficient schedules on the DSTN structure. Our methods achieved significantly better results than previous works—on average, 261 times wake-up time reduction and 30% less energy loss during the power mode transition.

Index Terms — low power, leakage, multi-threshold CMOS (MTCMOS), power gating, power mode transition, schedule.

I. INTRODUCTION

As CMOS process advances to nano-meter scale, leakage increases exponentially and becomes a significant drain on total power consumption [6][18][19]. Among the leakage reduction techniques, the power-gating technique has become one of the most effective methods. Figure 1 shows a power gating structure where a high $V_{th}$ sleep transistor is placed in series to the low $V_{th}$ devices [16]. Recently, many industrial power-gating designs have adopted the Distributed Sleep Transistor Network (DSTN) structure [14][22]. Figure 2 shows a DSTN structure, in which sleep transistors are connected by the virtual ground line ($VGND$).

Under the power gating structure, a circuit operates in two different modes. In the active mode, the sleep transistors are turned on and can be treated as the functional redundant resistances. In the sleep mode, the sleep transistors are turned off to reduce the leakage power. During the sleep period, all internal devices and the virtual ground are gradually charged because of the leakage in the low $V_{th}$ devices. With enough sleep time, we assume that all internal capacitances of logic devices will be charged and accumulate to the level close to $V_{DD}$. Therefore, when sleep transistors are turned on, a sudden discharge of the accumulated charges of internal devices leads to a large current, called a surge current, flowing through the sleep transistors to ground. The excessive surge current causes the Ldi/dt, IR drop and electromigration, which greatly affect the reliability and performance of a circuit [13][21]. In the worst case, the large IR drop may cause short term $V_{DD}$ collapse, resulting in the states saved in retention sequential elements or memories to be corrupted [11]. As a result, constraining the surge current during the power mode transition is vital for power gating designs. Since the current flowing through sleep transistors is proportional to the total size of the turned-on sleep transistors, the number and the timing to turn on sleep transistors should be restricted to avoid the excessive surge current. Hence, the turn-on sequence of the sleep transistors, called the wake-up schedule, has become a major challenge to constrain the surge current in a power gating design [7].

1 In certain power-gating structures, such as [3], the virtual ground voltages are controlled to a specific value during the sleep mode to solve the trade-off between the wake-up penalty and leakage savings. Hence, it is not always the case that the internal capacitances are charged to the level close to $V_{DD}$.
Several previous works have been developed to provide a wake-up schedule on the DSTN structure. In [12], researchers proposed two methods to reduce the maximum surge current. The first method is to turn on all sleep transistors stepwise using a weak wake-up signal as shown in Figure 3(a). The second method is to turn on one sleep transistor per clock period from the smallest size to the largest size as shown in Figure 3(b). In [1], researchers proposed an algorithm to partition logic devices to form a cluster-based power-gating design [4] for eliminating short-circuit problems. They also provided a wake-up schedule for the cluster-based structure [1][4]. In [17], their work is based on the fine-grained power-gating structure, i.e. each gate is connected its own sleep transistor. They formulated the wake-up scheduling problem as a Mixed Integer Linear Problem (MILP) and proposed an algorithm successively relaxing the MILP to a computationally efficient LP. In [20], the researchers propose a two-pass turn-on mechanism to handle surge currents and IR drops in a power-gating SoC design.

We now describe two important observations during the power mode transition. All experiments are implemented in TSMC 90nm CMOS technology. We illustrate the first observation in Figure 4. Let us consider the waveforms of the surge currents under two different wake-up schedules. Before the 4000th nanosecond, the circuit is in the sleep mode. The dotted line stands for the surge current under an All-On schedule that turns on all transistors simultaneously at the 4000th nanosecond. The bold line presents the surge current under a One-by-One schedule that turns on sleep transistors one by one in a one-nanosecond time interval. Here, a dilemma scenario arises. On one hand, the maximum surge current of the All-On schedule is 53.2 times larger than that of the One-by-One schedule. On the other hand, the wake-up time of the All-On schedule is 57.1% shorter than that of the One-by-One schedule. Therefore, to develop an efficient wake-up schedule, surge current should be as large as possible to discharge the internal energy in a short duration such as the All-On schedule. Nevertheless, the large surge current seriously affects the reliability of a circuit. As a result, the surge current should be limited for a reliable wake-up schedule at the cost of a long wake-up time such as the One-by-One schedule. Conventionally, this trade-off between the wake-up time and the surge current has been modeled as a scheduling problem under a designer-specified surge-current constraint. If the upper bound of a surge current can be obtained, designers can decide how many sleep transistors to be turned on simultaneously while the surge current still meets the constraint. Thus, a practical methodology to estimate the upper bound of a surge current is required for an effective schedule.

The other important observation is that spurious glitches are the major factors in total energy loss during the power mode transition. Figure 5 shows the voltage waveforms of an industrial design. The bold line stands for the voltage of an internal node A ($V_A$) whereas the dotted line presents the voltage of VGND ($V_{VGND}$). Initially, node A and VGND are charged close to $V_{DD}$ in the sleep mode. During the power mode transition, VGND needs sufficient time to stabilize to ground. Meanwhile, several spurious glitches occur before node A reaches its final value. In this case, it takes 250ns for VGND to become stable. Many spurious glitches can occur during the power mode transition. According to our experiments, on average, 75.71% of the total discharging energy comes from spurious glitches that greatly increase the wake-up time.

In this paper, we analyze the behaviors of the surge current in DSTN designs. Based on the analyses, we devise
a wake-up schedule that significantly reduces the wake-up time. To deal with spurious glitches, we also propose an improved schedule that reduces the total energy loss and further improves the wake-up time. In comparison to [12], on average, our method achieves 261 times reduction in the wake-up time and reduces energy loss from spurious glitches by 30%.

The remainder of this paper is organized as follows. Section II presents the preliminaries of the DSTN structure and of the surge current. In Section III, we perform a surge current analysis and introduce our problem formulation. In Section IV, we propose two effective wake-up schedules for the DSTN designs. Section V and Section VI present our implementation flow and experimental results respectively. Finally, Section VII concludes the paper.

II. PRELIMINARIES

In this section, we introduce the features of the DSTN structure and the calculation of the surge current. We assume a design contains several modules as shown in Figure 6(a) and one or several modules may go into sleep and wake up at the same time. Instead of the whole design, we focus on one DSTN module which is in the power mode transition as in Figure 6(b). Each cluster is connected to the corresponding sleep transistor $ST_i$ and to other sleep transistors by the virtual ground ($VGND$). Traditionally, logic clusters have been modeled as current sources whereas both sleep transistors and each segment of $VGND$ have been modeled as resistors in the active mode. In Figure 6(b), $V(ST_i)$ stands for the voltage of the node connected to the corresponding $ST_i$ on $VGND$. We would like to point out that $V(ST_i)$ may vary from one to another, especially in a large module, due to the resistance and capacitance of $VGND$. To simplify the model, in the DC analysis, a power-gating module can be transformed into a resistance network. By solving the resistance network, we can estimate the current flowing through turned-on $ST_i$ in the active mode but not during the mode transition [8][15].

In a system point of view, the magnitude of a surge current is an essential reference to examine the severity of how the ground rail ($GND$) is influenced, when one or more of the modules are during the power mode transition. In a DSTN module, a surge current that occurs during the power mode transition is treated as the summation of the total current flowing through each turned-on sleep transistor to $GND$. For simplicity without losing generality, surge current, which originally should be a function of time, is expressed as Eq(1) with parameter $i$ hidden:

$$surge\_current = \sum I_{\text{turnon}}(ST_i) \quad \forall i$$

where $I_{\text{turnon}}(ST_i)$ stands for the current flowing through the turned-on $ST_i$ during the power mode transition. Normally, once a sleep transistor is turned on, it will not be turned off again during the power mode transition.

III. SURGE CURRENT ANALYSIS

A. Surge Current versus Wake-up Time

To begin with, the wake-up time is the time required when (1) all sleep transistors are turned-on, and (2) all $V(ST_i)$s are stable within $\pm 5\%$ of nominal. Note that the wake-up time begins to be counted after any sleep transistor is turned on, called $T_{\text{begin}}$. According to [1][12], the wake-up time is formally defined as:

$$\text{wake-up time} = \max(T_{\text{stable}}(ST_i), T_{\text{turnon}}(ST_i)) - T_{\text{begin}} \quad \forall i$$

where $T_{\text{stable}}(ST_i)$ is the time when $V(ST_i)$ is stable within $\pm 5\%$ of nominal; $T_{\text{turnon}}(ST_i)$ stands for the time when $ST_i$ is
turned on.

Now we discuss how a surge current influences the wake-up time. The magnitude of a surge current determines the speed of discharging the accumulated energy. Hence, a short wake-up time can be achieved with a large surge current. Nevertheless, since an excessive surge current is the major source of noise on the power distribution network, the surge current should be under a designer-specified constraint to maintain the reliability of a circuit. Take Figure 7 as an example. The bold line presents the surge current waveform under schedule A while the dotted line stands for schedule B. We assume that the surge-current constraint is 79mA marked as the horizontal dotted line. Both schedule A and schedule B satisfy the constraint. However, compared to schedule A, schedule B is relatively pessimistic. The wake-up time of schedule B is 10.8 times longer than that of schedule A. As a result, the upper bound of a surge current is needed to develop an efficient wake-up schedule that operates under a given constraint.

B. Problem Formulation

Inputs: (1) $W(ST_i)$ for all $i$ (2) $SC\_CONSTRAINT$ (3) A sleep vector

Initial Condition: $ST_i$ is turned off for all $i$

Decision variable: $T_{\text{turnon}}(ST_i)$ for all $i$

Objective function:

Minimize $\max(T_{\text{stable}}(ST_i), T_{\text{normal}}(ST_i))$, for all $i$

Subject to: $SC\_CONSTRAINT \geq$ surge current, where surge current $=$ $\sum I_{\text{normal}}(ST_i)$ for all $i$

Figure 8. Wake-up schedule problem formulation.

Our problem formulation is shown in Figure 8. First, the following three inputs are given: (1) a set of practical sizes for $ST_i$, $W(ST_i)$ [8][9][14], (2) the surge-current constraint ($SC\_CONSTRAINT$), and (3) a sleep vector [2][21]. Note that the sleep vector is an input vector applied to the primary inputs of a circuit before entering the sleep mode. The values of sleep vector are known and will remain unchanged during both the sleep period and the wake-up process [1].

Next, the initial condition is that all $ST$s are turned off. The decision variable is $T_{\text{normal}}(ST_i)$. Again, $T_{\text{normal}}(ST_i)$ stands for the time when $ST_i$ is turned on. The objective function is to minimize the wake-up time defined in Eq(2). Finally, the surge current at all times must satisfy $SC\_CONSTRAINT$.

C. Surge Current versus Virtual Ground

In this section, we present the key factors affecting the estimation of a surge current. According to Eq(1), the surge current is the summation of $I_{\text{normal}}(ST_i)$ at a given time. Hence, let us focus on the calculation of $I_{\text{normal}}(ST_i)$ during the saturation region and the linear region. The equations are shown as Eq(3) and Eq(4) respectively.

$$I_{\text{normal}}(ST_i) = k_e \frac{W(ST_i)}{L} (V_{GS} - V_{TH}) (1 + \lambda V(ST_i))$$  \hspace{1cm} \text{Eq(3)}$$

$$I_{\text{normal}}(ST_i) = k_e \frac{W(ST_i)}{L} (V_{GS} - V_{TH}) \frac{\partial}{\partial V(ST_i)} - \frac{1}{2} V(ST_i)$$  \hspace{1cm} \text{Eq(4)}$$

where $W(ST_i)$ is the width of $ST_i$, $k_e$ is the process transconductance, $L$ is the channel length, $V_{GS}$ is the gate-source voltage, $V_{TH}$ is the threshold voltage, and $\lambda$ is the channel-length modulation parameter. Under a given set of $W(ST_i)$, the magnitude of $I_{\text{normal}}(ST_i)$ depends on $V(ST_i)$ which equals to the potential difference across $ST_i$.

Figure 9. Two segments of the virtual ground of an industrial design.

We now describe two key characteristics of $V(ST_i)$. First, $V(ST_i)$ is time-varying and thus difficult to calculate analytically. The main difficulty is that each time spurious glitches occur, additional charges flow into parts of the internal nodes. Meanwhile, the accumulated energy is still discharged through sleep transistors. The second characteristic is that, empirically, $V(ST_i)$ is strictly decreasing in DSTN designs. Figure 9 shows this characteristic by two waveforms: $V(ST_1)$ and $V(ST_2)$. No bounce happened on either $V(ST_1)$ or $V(ST_2)$ during the power mode transition. Moreover, we have simulated a large quantity of benchmarks under different wake-up schedules, finding that the empirical property holds in all DSTN designs. The reason for this consistency, according to our experiment, is that the discharge rate through sleep transistors is always larger than the charge rate from spurious glitches. However, we would like to mention that due to RLC parasitic from power networks and from the package, the monotonic decreasing of virtual grounds might not be true. In other words, when the RLC effects of the package are taken into consideration, large surge current may lead to longer wake-up time. Still, in the case when there are only one or few small modules in the power mode transition and the surge current is limited to some value, we
think the inductance effect can be very marginal. In Section VI.C, we have a further discussion about the package RLC parasitics.

IV. WAKE-UP SCHEDULES FOR WAKE-UP TIME MINIMIZATION CONSIDERING SPURIOUS GLITCHES

A. An Efficient Wake-up Schedule for Wake-Up Time Minimization

Figure 10: Wake-up schedule example.

Without losing generality, we illustrate our method using an example in Figure 10. To begin with, since \( V(ST) \) varies over time, we need to expand \( V(ST) \) into \( V_{ST}(t=t_0) \), which stands for the value of \( V(ST) \) at \( t = t_0 \). Similarly, we expand \( I_{\text{turnon}}(ST) \) into \( I_{\text{turnon}}(ST, t=t_0) \). We assume that the surge-current constraint, \( SC\_CONSTRAINT \), is set to 100mA and \( W(ST) \)s are given. Initially, before \( t = 30 \), both \( ST_1 \) and \( ST_2 \) are turned off. Again, we assume that \( V_{ST}(t=30) \) and \( V_{ST}(t=30) \) have been charged to \( V_{DD} \). Let the wake-up process begin at \( t = 30 \), i.e. \( T_{\text{begin}} = 30 \). According to Eq(3)(4), we can calculate both \( I_{\text{turnon}}(ST_1, t=30) \) and \( I_{\text{turnon}}(ST_2, t=30) \). Then we can determine which \( ST \)s can be turned on while \( SC\_CONSTRAINT \) is still satisfied. In Figure 10(a), we have \( I_{\text{turnon}}(ST_1, t=30) = 90 \)mA and \( I_{\text{turnon}}(ST_2, t=30) = 60 \)mA. To satisfy \( SC\_CONSTRAINT \), we can turn on either \( ST_1 \) or \( ST_2 \) but not both, at \( t = 30 \). Let us choose to turn on \( ST_1 \) because \( I_{\text{turnon}}(ST_1, t=30) \) is larger than \( I_{\text{turnon}}(ST_2, t=30) \) and may lead to a shorter wake-up time. Therefore, we have \( T_{\text{turnon}}(ST_1) = 30 \). Moreover, since \( V(ST) \) is strictly decreasing, \( V_{ST}(t=30) \) is the largest value of \( V_{ST}(t=30) \). We emphasize that the strictly decreasing characteristic of \( V(ST) \) on the DSTN structure is important because it can be applied to estimate the upper bound of a surge current. As a result, \( I_{\text{turnon}}(ST_1, t=30) \) is also the largest value of \( I_{\text{turnon}}(ST_1, t=30) \) according to Eq(3)(4). Hence, \( SC\_CONSTRAINT \) remains satisfied before we turn on the next \( ST \)s. In our technique, if any \( ST \) still remains turned-off, we will iteratively check and might turn on several \( ST \)s after a certain time interval. The time interval is empirically decided. In our experiment, we set the time interval to 30ps. Hence, the next time interval is between \( t = 30 \) and \( t = 60 \). Since \( ST_2 \) remains turned off, we will simulate the design with \( ST_1 \) turned on between \( t = 30 \) and \( t = 60 \). All \( V(ST) \)s are obtained through simulations because, as mentioned in Section III.C, \( V(ST) \)s are difficult to calculate analytically. Figure 10(b) shows the second iteration of our technique. After the SPICE-like simulator updates the waveforms of \( V(ST) \), we identify that \( I_{\text{turnon}}(ST_1, t=60) + I_{\text{turnon}}(ST_2, t=60) = 90 \)mA, which does not exceed \( SC\_CONSTRAINT \). Therefore, we can turn on \( ST_2 \) at \( t=60 \) and have \( T_{\text{turnon}}(ST_2) = 60 \). However, \( V_{ST}(t=60) \) and \( V_{ST}(t=60) \) are not yet stable in this time interval. At the end of the second iteration, we continue the simulation with both \( ST_1 \) and \( ST_2 \) turned on. Figure 10(c) shows that \( V_{ST}(t=90) \) and \( V_{ST}(t=90) \) are stable within ±5% of nominal. Therefore, \( T_{\text{stable}}(ST_1) \) and \( T_{\text{stable}}(ST_2) \), defined in Section III.A, are set to 90. As a result, when \( T_{\text{begin}} = 30 \), the wake-up time is 90ps – 30ps = 60ps according to Eq(2).

Figure 11 presents the details of our schedule algorithm for Wake-up Time Minimization (WTM). Initially, \( ST \)s are

![Algorithm](image-url)
set to OFF, \textit{surge\_current} set to 0, and \( t \) is set to \( T_{\text{begin}} \). In the beginning of the iteration, we update \( V_{ST}(t) \) according to the simulation results. With Eq(3)(4), \( I_{\text{turnon}}(ST_1,t) \) can be calculated under a given set of \( W(ST_1) \). For those \( ST_i \)s which remain turned off, we aim to turn on several additional \( ST_i \)s to have \textit{surge\_current} as large as possible without exceeding \textit{SC\_CONSTRAINT}. Note that this problem can be transformed into a well-known knapsack problem, which can be solved efficiently by the dynamic programming technique [10]. Hence, we apply the dynamic programming technique to decide which \( ST_i \)s should be turned on. According to the obtained result, we turn on \( ST_i \) and have \( T_{\text{turnon}}(ST_i) \) set to \( t \). Meanwhile, we also need to update \textit{surge\_current}. At the end of this iteration, we perform the simulation with all turned-on \( ST_i \)s and update \( t \) with \( t + \textit{time\_interval} \). Again, \textit{time\_interval} is empirically determined. After all \( ST_i \)s are turned on, our algorithm returns the wake-up schedule, \( T_{\text{turnon}}(ST_i) \).

B. An Improved Wake-up Schedule Considering Physical Implementation Issues and Spurious Glitches

i. Physical Implementation Issues

Physically, sleep transistors are normally placed in order [15][22] and, in most cases, are deployed at the ends of rows. Therefore, if there are five rows, there will be five sleep transistors aligned in order as the example in Figure 12. Typically, a sleep/wake-up signal is provided from a power management unit. Turning on sleep transistors without conforming to their order of physical placement may lead to a large routing area due to the complicated power management units. The penalty may make the wake-up schedule impractical. For example, the turn-on sequence \{\( ST_1, ST_2, ST_3, ST_4, ST_5 \}\) does not conform to the physical order of \{\( ST_5, ST_3, ST_1, ST_4, ST_2 \}\).

As a result, we assume sleep/wake-up signals are arranged as a daisy-chain-like implementation [11]. In our wake-up schedule, we restrict the turn-on sequence of sleep transistors as follows. First, in our formulation, sleep transistors are numbered from \( ST_1 \) to \( ST_5 \). Several sleep transistors can be turned on at the same time. We assume that a wake-up sequence first turns on sleep transistors between \( ST_a \) and \( ST_b \), where \( a \leq b \leq k \). After that, only consecutive \( ST_i \)s whose positions are next to \( ST_a \) or \( ST_b \) can be turned on. Figure 12 shows a turn-on sequence \{(\( ST_a \), \( ST_d \))\}, \((ST_3, ST_2),(ST_5, ST_4)\) following the above formulation. In the example, \( ST_d \) is turned on first followed by \( ST_3 \) and \( ST_2 \) together. Finally, \( ST_2 \) and \( ST_3 \) are turned on at the same time.

ii. Spurious Glitches Phenomenon

Spurious glitches waste energy during the mode transition. Another objective of this paper is to minimize spurious glitches by controlling the wake-up schedule. In the following paragraphs, two useful properties are described to reduce spurious glitches.

We now illustrate how different schedules affect the seriousness of spurious glitches. Figure 13(a) shows three cascaded inverters located in three different logic clusters in a DSTN design. For simplicity of explanation, the environment is set up as follows. The supply voltage is 1V,
the delay of an inverter is 1 time unit, and the input of the first inverter is logic 1 during the wake-up process. We assume that schedule A is \( T_{\text{turnon}}(ST_i) = 2, T_{\text{turnon}}(ST_j) = 0 \), \( T_{\text{turnon}}(ST_k) = 4 \) and schedule B is \( T_{\text{turnon}}(ST_i) = 0, T_{\text{turnon}}(ST_j) = 4, T_{\text{turnon}}(ST_k) = 2 \). Figure 13(b)(c)(d) show the voltage waveforms of \( V(ST_i) \)s and of three inverter’s outputs, \( V_{NI} \), under schedule A. Figure 13(e)(f)(g) show the waveforms under schedule B. From these figures, schedule B is superior to schedule A in reducing spurious glitches. The results are as follows. First, schedule A has a spurious glitch on \( V_{NI} \) in Figure 13(d) while schedule B does not in Figure 13(g). Moreover, in Figure 13(c)(f), schedule A has a larger glitch on \( V_{NJ} \) than schedule B. Let us now explain why schedule B outperforms schedule A on spurious glitch reduction. The output of a logic gate remains unstable if its inputs are unstable. As a result, the fan-in cones of a gate should be stabilized earlier than the gate itself. From the above statement, if a gate is topologically close to the primary inputs, it should be stabilized earlier to avoid the propagation of spurious glitches. In Figure 13(a), \( N_j \) is closer to the primary inputs than \( N_i \) and \( N_k \). The spurious glitches may be reduced if the voltage of \( N_j \), \( V_{NJ} \), is stabilized earlier than \( V_{N1} \) and \( V_{N2} \). Moreover, since each logic gate is connected to the corresponding \( ST_i \), the output value of the gate, if equal to logic 0, is bounded by \( V(ST_i) \). For example, since \( N_j \) is connected to \( ST_j \), \( V_{NJ} \) will be stabilized after \( V(ST_j) \) becomes stable as shown in Figure 13(b)(c). Furthermore, \( V(ST_i) \) can be stabilized faster by turning on its corresponding sleep transistor, \( ST_i \), due to the current discharging balance on the DSTN structure [8][14]. In this example, since schedule B turns on \( ST_j \) at \( t=0 \) whereas schedule A turns on \( ST_j \) at \( t=2 \), \( V(ST_j) \) is stabilized earlier in schedule B than in schedule A. Therefore, \( V_{NJ} \) is stabilized earlier in schedule B. Since \( V_{NJ} \) is stable, \( V_{N2} \), the output of \( V_{NJ} \), is also stabilized earlier in schedule B than in schedule A. From the above example, the first property to reduce spurious glitches is that the logic gates should be stabilized in their corresponding topological order.

The second property is that \( ST_i \), which is connected to more gates whose final output values are logic 0, should have higher priority to be turned on. We illustrate this concept using the same example in Figure 13. Let us first focus on \( V_{N2} \) whose final value is 1. Under the assumption that \( V_{N2} \) settles down to logic 0, the stabilization of \( V_{N2} \) is independent from \( V(ST_j) \) since \( V_{N2} \) is charged by \( V_{DN2} \), not discharged by \( V(ST_j) \).

### iii. An Improved Wake-up Schedule Considering Spurious Glitches and Physical Implementation Issues

An improved schedule for wake-up time minimization has to consider both spurious glitches and physical implementation issues. From Section IV.B.ii, we observe that the sleep transistor with the following two properties should have higher turn-on priority – (1) connected to more logic gates close to primary inputs, and (2) connected to more gates whose outputs are logic 0. Nevertheless, the gates with the same topological order may not be placed in the same cluster physically in practice. In this section, we propose an Improved schedule for Wake-up Time Minimization (IWTM).

Let us explain the differences between IWTM and WTM mentioned in Section IV.A. IWTM calculates \( \text{composit_weight}(ST_i) \) as the turn-on priority of the sleep transistors. \( \text{composit_weight}(ST_i) \) is calculated as:

\[
\text{composit_weight}(ST_i) = \text{topological_weight}(ST_i) + \text{position_weight}(ST_i) + \text{width_weight}(ST_i)
\]

where \( \text{topological_weight}(ST_i), \text{position_weight}(ST_i) \) and \( \text{width_weight}(ST_i) \) represent three major parts which impact the wake-up time and the energy loss of a schedule. First, let the level of a logic gate represents its topological order from primary inputs to the gate similar to the definition in [5]. After that, level of each gate is calculated and then \( \text{topological_weight}(ST_i) \) can be expressed as:

\[
\text{topological_weight}(ST_i) = \Sigma (1 / \text{level of gate } j)
\]

where \( j \) represents all gates whose final outputs are logic 0 for each \( i \). \( \text{topological_weight}(ST_i) \) represents the turn-on sequence which potentially makes the majority of gates stabilized in the topological order.

Second, \( \text{position_weight}(ST_i) \) stands for the impact factor arising from the physical position of a sleep transistor. Under the physical limitation mentioned in Section VI.B.i, the starting point to turn on sleep transistor is important because it limits the choices of sleep transistors to be turned on afterward. Hence, the middle position is preferred since the set of turn-on choices may be larger and the \( V(ST_i) \)s may decrease averagely because of the current discharging balance [14]. We assume that the positions of sleep transistors are numbered from 1 to \( k \). \( \text{position_weight}(ST_i) \) is calculated as:

\[
\text{position_weight}(ST_i) = \text{middle_pos} - |\text{pos}(ST_i) - \text{middle_pos}|
\]

where \( \text{middle_pos} = (1 + k) / 2 \) and \( \text{pos}(ST_i) \) is the position number of \( ST_i \).

Finally, \( \text{width_weight}(ST_i) \) is the third index to decide the turn-on priority. Since the dynamic programming technique
is no longer suitable to be applied because of the limited turn-on choices and of the considerations to reduce spurious glitches, width_weight(ST) aims to turn on a sleep transistor with a larger width first, which leads to a larger \( I_{\text{turnon}}(ST) \). width_weight(ST) is expressed as:

\[
\text{width\_weight}(ST) = W(ST) / \text{width\_avg}
\]

where \( W(ST) \) is the width of \( ST \), and \( \text{width\_avg} \) is the average width of all sleep transistors. According to composit_weight(ST), we prioritize \( ST \)'s in a decreasing order. In the first iteration, under the surge-current constraint, we manage to turn on the highest priority sleep order. In the first iteration, under the surge-current constraint, we aim to turn on several high priority \( ST \)'s if the summation of \( I_{\text{turnon}}(ST, t) \) and surge_current is smaller than or equal to \( SC_{\text{CONSTRAINT}} \). Then, our iterative technique begins to accomplish the turn-on schedule. Note that the prioritized \( ST \)'s can be turned on only when they have turned-on neighbors and satisfy \( SC_{\text{CONSTRAINT}} \) at the same time.

V. IMPLEMENTATION FLOW

![Implementation flow diagram](image)

Here, we present the implementation flow of our sleep transistor wake-up schedule in Figure 14. First, an RTL netlist is synthesized by Synopsys DesignVision™ to generate a gate-level netlist. After that, the gate-level netlist is placed and routed by Cadence SOC Encounter™. After placement and routing, we can extract the parasitic capacitances by SOC Encounter™. We also obtain the DEF file containing the physical location of each gate. Note that the gates in the same row are grouped into a cluster. Each sleep transistor is placed at the end of corresponding cluster shown in Figure 12. For simplicity of discussion, we use one-footer power-gating structure to demonstrate our idea. In practice, normally, two footers are adopted along both ends of a row. We implement [21] as the sleep vector file and [9] as the sleep transistor size file. Note that the TSMC 90nm CMOS technology process is applied throughout all experiments. According to [23], the current flowing through a sleep transistor cannot be accurately calculated with Eq(3)(4) under 90nm process or below. Hence, to replace Eq(3)(4), we use “footer sleep transistor behavior file” for the current estimation. Note that “footer sleep transistor behavior file” is generated via HSPICE™ simulation under different \( W(ST) \)'s and \( V(ST) \)'s. Next, to construct “virtual ground behavior file” which contains all \( V(ST) \)'s at a specified time point, the design is simulated through Nanosim™ with the accuracy set to 6. With updated \( V(ST) \)'s and the given \( W(ST) \)'s of all sleep transistors, we can obtain \( I_{\text{turnon}}(ST) \)'s using the “footer sleep transistor behavior file”. Finally, we iteratively simulate the design via NanoSim™ and then feed “virtual ground behavior file” back to our IWTM Engine until all sleep transistors are turned on.

VI. EXPERIMENTAL RESULTS

A. The Comparisons among All-On, Kim’s Method, WTM and IWTM

Here, we re-implemented the second method of [12] and compared that with our WTM and IWTM on the DSTN structure. Again, WTM presented in Section IV.A does not consider spurious glitches and physical implementation issues while IWTM presented in Section IV.B.iii takes both into consideration. In this section, the package parasitics are not added for both our methods and [12].

Next, the parameters of our experiments are set as follows. First, the time interval is set to 30ps in WTM and IWTM for simulation. In our experimental experience, the time interval of 30ps is a good balance between program runtime and wake-up time improvement. Secondly, since a well-designed circuit normally can tolerate the maximum instantaneous current (MIC) arising from gate switching in the active mode, the MIC is used as the surge-current constraint to guarantee the reliability of the circuit. We use PrimePower™ to generate the MIC as the surge-current constraint. Note that practically designers need to guardband the surge-current constraint to prevent the worst-case surge current from exceeding the constraint, which may damage the reliability of a circuit. Finally, to re-implement the method of [12], the clock period of each
The circuit benchmark is set to the corresponding longest path delay.

Table I shows our experimental results. Take circuit C1355 as an example. The wake-up time is 0.03ns from All-On, 47.50ns from [12], 0.36ns from WTM, and 0.27ns from IWTM. The energy loss from spurious glitches during the power mode transition is 3.39pico-coulomb from All-On, 5.94pico-coulomb from [12], 4.41pico-coulomb from WTM, and 4.13pico-coulomb from IWTM. The surge current is 203.97mA from All-On, which violates the surge-current constraint, whereas the other methods satisfy the constraint successfully.

On average, the wake-up time of our IWTM is 261 times faster than that of [12]. Also, IWTM has 30% less energy loss than [12]. Our results are significant on large benchmarks such as C6288 and an industrial Advanced Encryption System (AES). In AES, the wake-up time of our IWTM algorithm is 643 times faster than that of [12] and has 75.8% less energy loss than [12]. It is worth mentioning that the runtime of our flow can finish within one hour on all ISCAS benchmarks and within eight hours on AES. In addition, IWTM outperforms WTM in the aspects of wake-up time, energy loss, and maximum surge current on average cases. The reason is that spurious glitches are taken into account to reduce additional energy loss, which contributes the wake-up time minimization. Furthermore, IWTM inherits the main idea of WTM to have the surge current as large as possible without exceeding the constraint. Therefore, even if the physical implementation issue constrains the solution space of IWTM, IWTM still averagely performs better than WTM. The results clearly demonstrate that our IWTM always achieves impressive wake-up time and energy loss reduction in both benchmarks and the industrial design.

2 The averages in the bottom line of Table I are computed as follows. First, for a benchmark, the results from each method are normalized to those from IWTM, e.g. Column 4 to 7 are normalized to Column 7. Then, for each column, the normalized values are averaged.
We re-implement the method in [12] to turn on sleep transistors one-by-one also in a 30ps time interval to observe how the results change. The results are demonstrated in Table II. Here, let us focus on the results of [12]. Take C6288 for an example, the wake-up time is 1.41ns and the energy loss is 32.54pico-coulomb. However, the maximum surge current is 64.3mA, which violates the surge-current constraint, 36.72mA.

Based on the experimental results, 60.0% of the benchmarks fail to satisfy the surge-current constraints. Therefore, the method in [12] may run the risk that the maximum surge current exceeds the constraint especially when a short time period is applied. We would like to point out this potential danger arises from the inability to estimate the surge current.

C. Package RLC Parasitics

To further consider the RLC effects, we applied DIP-40 package model in IWTM. For simplicity of discussion, we select C7552 as the representative to elaborate the experimental results. Under IWTM scheduling, the wake-up time is 10.54ns, the energy loss is 20.51pico-coulomb, and the maximum surge current is 64.3mA, which satisfies the surge-current constraint. Compared to the results without packages, the wake-up time of C7552 increases from 0.33ns (in Table I of Section VI.A) to 10.54ns, due to the inductively induced voltage fluctuation of VGNDs.

Next, with the same package model, we re-implement [12] and set the time interval to one clock period. The wake-up time is 185.32ns, the energy loss is 27.91pico-coulomb, and the maximum surge current is 1.57mA. Compared to [12], IWTM achieves 17.58 times wake-up time reduction when the package is applied. The results demonstrate that, even with package, IWTM still performs well on the wake-up time minimization.

Here, we would like to point out that simulating a design with the package RLC parasitics is extremely time-consuming. With the accuracy of Nanosim™ set to 6, the simulation of the industrial design AES lasts seven days and cannot conclude. As a result, it may not be practical to simulate an entire SoC design with packages, especially under the high accuracy setting. In the early stage of the design process, the surge-current constraint is usually applied as a popular alternative to ensure the reliability of a design during the power mode transitions [1][17]. In this paper, we also assume that a properly-selected surge-current constraint is an efficient and effective method to maintain the design reliability.

VII. CONCLUSIONS

We have presented an effective and practical wake-up schedule, IWTM, on the DSTN structure for wake-up time minimization. The main idea of our schedule is to apply the strictly decreasing property of the virtual ground on the surge current estimation. We also minimize the energy loss from spurious glitches by IWTM considering physical implementation issues. The results show that our IWTM schedule, compared with [12], can achieve 261 times greater wake-up time reduction and reduce energy loss by 30% more.

ACKNOWLEDGEMENT

The authors would like to thank the National Science Council of Taiwan (NSC-97-2220-E-007-013) for supporting this project.
REFERENCES


Da-Cheng Juan received the B.S. degree and M.S. degree from National Tsing Hua University, HsinChu, Taiwan, in 2005 and 2007 respectively, both in computer science. His research interests include low-power related topics, electronic design automation (EDA) techniques, and algorithm design. Mr. Juan was the recipient of the Academic Achievement Award from the National Tsing Hua University in 2003, Masterpiece prize of National College Programming Contest in 2004, Two-Strait College Programming Contest Award (third place) from National Tsing Hua University in 2005. He is currently with the Department of Electronic Engineering and Computer Science, National Tsing Hua University.

Yu-Ting Chen received the B.S. degrees in both computer science and economics and the M.S. degree in computer science from National Tsing Hua University, HsinTsu, Taiwan, R.O.C., in 2005 and 2007 respectively. His research interests include power-gating design issues, low power circuit design techniques, and related topics of electronic design automation (EDA).

In 2005, he was with Taiwan Semiconductor Manufacturing Company Ltd. (TSMC) as a summer intern. He is currently with the Department of Electronic Engineering and Computer Science, National Tsing Hua University.

Ming-Chao Lee received the B.S. degree in the Department of Information and Computer Engineering from the Chung Yuan Christian University, ChungLi, Taiwan, R.O.C., in 2002, the M.S. degree in Computer Science and Information Engineering from the National Dong Hwa University, Hualien, Taiwan, R.O.C., in 2004. He is currently pursuing the Ph.D. degree in the Department of Computer Science from the National Tsing Hua University, HsinTsu, Taiwan, R.O.C. His current research interests include VLSI electronic design automation, low-power circuit design issues, and power-gating design techniques.

Shih-Chieh Chang (S’92–M’95) received the B.S. degree in electrical engineering from the National Taiwan University, Taiwan, R.O.C., in 1987, and the Ph.D. degree in electrical engineering from the University of California, Santa Barbara, in 1994. He is currently a Professor with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R.O.C. From 1995 to 1996, he worked with Synopsys, Inc., Mountain View, CA. From 1996 to 2001, he joined the faculty with the Department of Computer Science and Information Engineering, National Chung Cheng University, ChiaYi, Taiwan, R.O.C. His current research interests include logic synthesis, functional verification for SoC, and noise analysis. Dr. Chang was a recipient of a Best Paper Award at the 1994 Design Automation Conference.