Wakeup Synthesis and Its Buffered Tree Construction for Power Gating Circuit Designs

Seungwhun Paik
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

Sangmin Kim
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

Youngsoo Shin
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

ABSTRACT

A power gating circuit suffers from large amount of rush current when it wakes up, especially when all switch cells are turned on at the same time. If each switch cell is turned on in different instant of time, the rush current can be reduced. It is shown in this paper that the rush current can be reduced even more if signal transition time (or signal slew) to each switch cell is adjusted. The wakeup synthesis that we define is to determine the turn-on time and signal slew of each switch cell; the goal is to minimize wakeup delay while rush current is kept below the maximum value that is allowed. The corresponding synthesis algorithm is proposed. The determined turn-on time and signal slew are implemented using a buffered tree, where a source is a wakeup signal and sinks are multiple switch cells; the synthesis algorithm to generate the tree is proposed. The wakeup synthesis and buffered tree construction are integrated into a design flow that receives a netlist of power gating circuit as an input and produces a layout of netlist with wakeup network embedded. Experiments in an industrial 1.1 V, 45-nm technology demonstrate that the wakeup delay is reduced by 43% on average of example circuits compared with 2-pass turn-on, which is widely used.

Categories and Subject Descriptors: B.7.1 Integrated Circuits: Types and Design Styles [VLSI]

General Terms: Algorithms, Design

Keywords: Power gating, leakage, wakeup synthesis

1. INTRODUCTION

Power gating has become one of the most widely used circuit techniques to reduce leakage current. It has been employed in standard-cell library designs, supported by a few commercial CAD tools [1, 2] and foundry design services [3], and adopted as industry standard [4]. Power gating reduces leakage when a circuit is in standby mode by cutting the circuit off from its V_{ss} by means of a current switch called a footer (or off from its V_{dd} by means of a header).

The footer can be implemented in two different ways: embedded in each standard cell or shared by all standard cells in a circuit. The former approach allows the conventional design tools to be used, i.e. each cell is pre-characterized and is used during logic synthesis, timing analysis, and physical synthesis. Due to large increase of area, however, this method is typically used in a selective number of cells [5]. The latter approach is used in most industrial applications [6], thus is the focus of discussion throughout this paper.

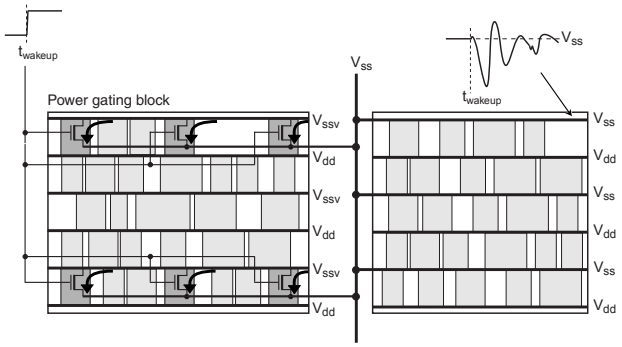


Figure 1: Large rush current during wakeup causes excessive IR drop, which affects adjacent blocks that are actively switching.

When the footer is shared, it is physically implemented as multiple footer cells, which are dispersed over placement region in a regular fashion, as shown in Figure 1. When a circuit makes a transition from active- to standby-mode, these footer cells can be turned off at the same time. During the opposite transition (wakeup), however, turning on all footer cells simultaneously causes a large amount of current to rush through the cells, which in turn causes IR drop on V_{ss} rails to exceed its allowance. This may result in malfunctions of adjacent blocks that are actively switching as illustrated in Figure 1. To reduce this IR drop, each footer cell may be turned on one by one with the interval of turning on arbitrarily long, but that increases wakeup delay. The problem therefore is to determine the time when each footer cell is turned on such that the wakeup delay is minimized while the constraint on maximum rush current is honored.

Let I_{max} be the maximum rush current that is allowed. The minimum latency can be achieved if footer cells are turned on in a way that rush current is always made equal to I_{max}. This is impossible to achieve in practice because of two problems: there are only limited number of footer cells that can be controlled, where the number is typically determined by active-mode circuit delay; the turn-on time of each cell, once determined, may not be exactly realized since footer cells are controlled by a single wakeup signal that is routed to all the cells through wires and buffers, similar to a clock network. In this paper, we solve the first problem by deciding the signal slew

ISLPED'10, August 18–20, 2010, Austin, Texas, USA.

as well as the time when turn-on signal arrives (turn-on time) at each footer cell, so that we attain a better capability of fine-tuning cell current than deciding turn-on time alone. Regarding the second problem, we propose a method to synthesize a buffered tree for wakeup network, which accurately implements the determined slew and turn-on time.

Our main contributions are:

- In-depth analysis of wakeup process in power gating circuits through experiments in 45-nm industrial technology (Section 2.1).
- Wakeup synthesis that determines the signal slew and turn-on time of each footer cell, such that wakeup delay is minimized while I_{max} is honored (Section 3).
- Buffered tree synthesis for construction of wakeup network (Section 4).
- Integration of wakeup synthesis and buffered tree construction into complete design flow that receives a netlist of power gating circuit and produces a layout where wakeup network is embedded; comprehensive experiments to assess wakeup delay in 45-nm technology (Section 5).

2. PRELIMINARIES

2.1 Wakeup Process

Figure 2(a) shows a part of circuit c5315, which is one of ISCAS benchmark circuits. Node a takes logic 1 and nodes b and c take logic 0 if footer is turned on. When footer is turned off, V_{ssv} increases toward V_{dd} until it reaches a steady-state potential, which, in our experiment with 1.1 V 45-nm technology, is about 0.9 V as shown in its voltage waveform. Note that the potential of node b (and c) is equal to V_{ssv}. Once footer is turned on, V_{ssv} starts to discharge. Since the potential of node c is 0.9 V when wakeup just starts and V_{ssv} is faster to discharge than node c does, nMOS of inverter turns on and discharges a; once c discharges enough, however, pMOS of inverter then turns on and charges a again. Node b simply discharges since the momentary discharge at a is not enough to change the logic value of b. Due to this simultaneous discharging of all nodes a, b, and c as well as V_{ssv}, the sum of their current (see Figure 2(a)) flowing through the footer, which constitute rush current, can be very large. This can be compared with switching current in normal circuit shown in Figure 2(b); there is less overlap of discharge current this time.

In Figure 3(a), we report average and maximum discharge current of several normal circuits. Each circuit was submitted to fast SPICE simulation [7] to obtain discharge current, which was repeated 200 times using random vectors; taking average and picking maximum, respectively, yield average and maximum discharge current. These currents are compared with rush current, which was also obtained by fast SPICE simulation after converting each circuit into a power gating counterpart. Maximum current is significantly larger (10.6× on average) than average current in all circuits, which is typically observed [8]. Power networks are usually designed such that maximum current can be sustained. Figure 3(a) signifies the importance of rush current (when footer cells are turned on simultaneously), which is even larger than maximum current; it may restrict the integrity of power distribution network.

Since large rush current is caused by turning on all footer cells at the same time, a natural solution is to turn them on one by one, which can be physically implemented by connecting the cells in a daisy chain fashion [9]. Figure 3(b) shows a simple experiment, where we take c6288; 9 footer cells were assumed to be individually turned on with uniform interval of Δt. Rush current decreases

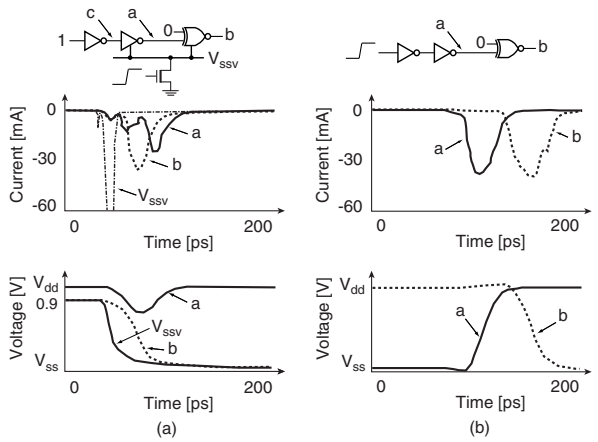


Figure 2: (a) Transient waveform of V_{ssv} and internal nodes during wakeup of power gating circuit, and (b) the same circuit in active mode.

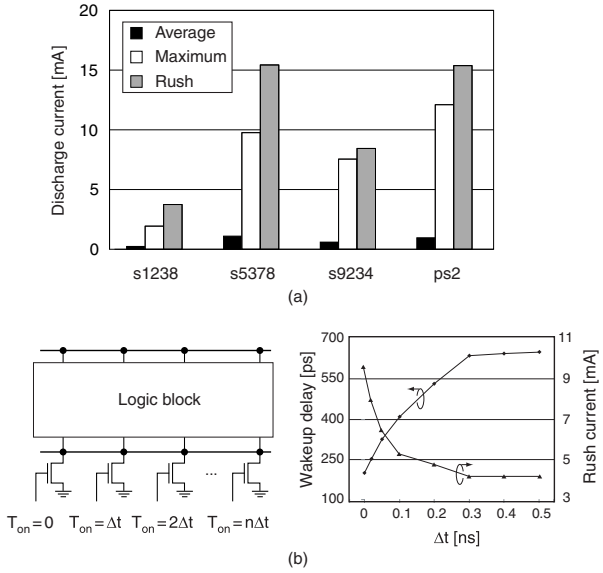


Figure 3: (a) Comparison of average and maximum discharge current of normal circuits together with rush current of power gating circuits (when footer cells are turned on at the same time), and (b) wakeup delay and rush current when footers are turned on one by one.

as we increase Δt; wakeup delay, however, increases. Note that, we ignore the effect of di/dt in this simple experiment. The wakeup delay when $\Delta t = 0$, thus, can be larger than the number reported, meaning that the wakeup delay may follow a convex curve rather than a monotonic line.

2.2 Related Work

There have been two directions of research to reduce rush current and wakeup delay: use a circuit that is conceptually different from basic power gating or adjust the turn on strategy of basic power gating.

The virtual power/ground rails clamp (VRC) [10, 11] uses a forward biased diode placed in parallel with footer. The potential of V_{ssv} rises once footer is turned off but is clamped by the built-in potential of diode once it is turned on. VRC benefits rush cur-

rent and wakeup delay due to less increase of V_{ssv}, but suffers from large standby leakage. In zigzag power gating (ZPG) [12–14], logic states of internal nets are maintained by applying a predetermined input vector and by using footers and headers in zigzag fashion. The amount of rush current, therefore, is reduced, which also benefits wakeup delay. Its design, especially in standard-cell designs, is very complicated [14]. Note that there is no guarantee in both VRC and ZPG that rush current is kept smaller than I_{max}, though they effectively reduce the amount.

If a footer is weakly turned off [15] by applying a voltage larger than 0 but smaller than its threshold voltage, the increase of V_{ssv} during standby mode can be suppressed. It, however, suffers from a large standby leakage though both rush current and wakeup delay are reduced as in VRC. In two-pass turn-on [6, 9], a header (or footer) consists of a group of weak cells and another group of strong cells. The first pass turns on the weak cells one by one such that V_{ddv} is fully restored (to voltage close to V_{dd}) when the last cell is turned on. The limited current flow of the weak pMOS cell constrains the rush current. The weaker it is, the lower current peak becomes but wakeup delay increases. In the second pass, main strong cells are turned on and then, together with weak ones, take a responsibility of supplying the current needed for active mode of operation. Gradually turning on footer cells, as in two-pass turn-on, lends itself to a problem of wakeup network synthesis we address in this paper. Using turn-on time alone [16] (without adjusting signal slew) has a limited capability to reduce wakeup delay; the method to physically implement the determined turn-on time has not been addressed.

3. WAKEUP SYNTHESIS

3.1 Problem Formulation

We are given a power gating circuit, which consists of a logic block and a list of distributed footer cells $1,2,\ldots,n$. The number of cells n is typically determined by assuming a negligible voltage drop (say 1% of V_{dd}) across footer when the logic block is switching, so that employing power gating does not introduce any practical increase of delay. The assumed voltage drop together with average discharge current from the logic block yields the value of n [17].

The wakeup synthesis is to determine, for each cell i, the time when turn-on signal arrives (t_i) and its signal slew (τ_i), where slew is bounded, i.e. $\tau_{min} \le \tau_i \le \tau_{max}$. The objective is to minimize wakeup delay while the sum of rush current through footer cells does not exceed rush current constraint all the time, i.e.

$$\sum_{i=1}^{n} I_i(t) \le I_{max}, \tag{1}$$

where I_i is rush current of footer cell i.

The wakeup delay is defined (provided that the first footer cell is turned on at time 0) as the time when V_{ssv} becomes close to V_{ss} (e.g. $V_{ssv} \le 0.02V_{dd}$) and all n footer cells are turned on so that the logic block can resume its computation. There is no preference as to the footer cells during synthesis, i.e. they are considered in order of $1,2,\ldots,n$. If we are given a power gating circuit that is already placed, we may perform wakeup synthesis in a way that adjacent footer cells are turned on in adjacent instances of time, which we address in Section 4 in more detail.

In our approach to wakeup synthesis, we regard time t as a discrete variable, which allows us to consider synthesis at time $0,\Delta t,2\Delta t,\ldots$, where Δt is a small time increment. The key component of synthesis is to compute the sum of rush current at time $n\Delta t$, $\sum_i I_i(n\Delta t) = \sum_i I_i[n]$, while synthesis is performed so that (1) is honored.

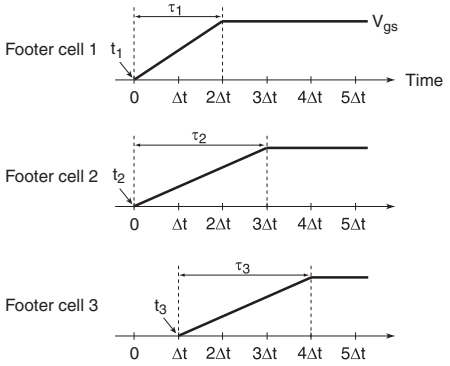


Figure 4: Estimation of rush current.

3.2 Estimation of Rush Current

Consider Figure 4, where three footer cells are being considered. Assume that we want to compute the total rush current at time $3\Delta t$, $I_1[3]+I_2[3]+I_3[3]$. The rush current, which is drain-to-source current of nMOS footer cell, can be characterized as a function of V_{gs} and V_{ds}. The V_{gs} is easily obtained once turn-on time t_i and signal slew τ_i are determined; for instance, V_{gs} of cell 3 at time $3\Delta t$ is equal to $V_{dd}/\tau_3\,(3\Delta t - t_3)$ (see Figure 4). The V_{ds}, which is equal to V_{ssv}, however, is not readily obtained. This is because V_{ssv} at particular instance of time during wakeup process is determined by total amount of charge that has been drained by rush current, while rush current itself is determined by V_{ssv}.

To break the cyclic dependency between rush current and V_{ssv}, we rely on iterative approximation. At time 0 when wakeup is initiated, V_{ssv} is at its steady-state potential, say $V_{ssv}[0] = V_{dd}$, which is the potential during standby mode. To obtain $I_1[1]$ and $I_2[1]$ at time Δt in Figure 4, we use $V_{ssv}[0]$ as an approximate value of $V_{ds}[1]$, which, together with V_{gs} values that are obtained from Figure 4, yields $I_1[1]$ and $I_2[1]$. We then update V_{ssv}, i.e. we derive $V_{ssv}[1]$ by setting $k = 1$ in the equation:

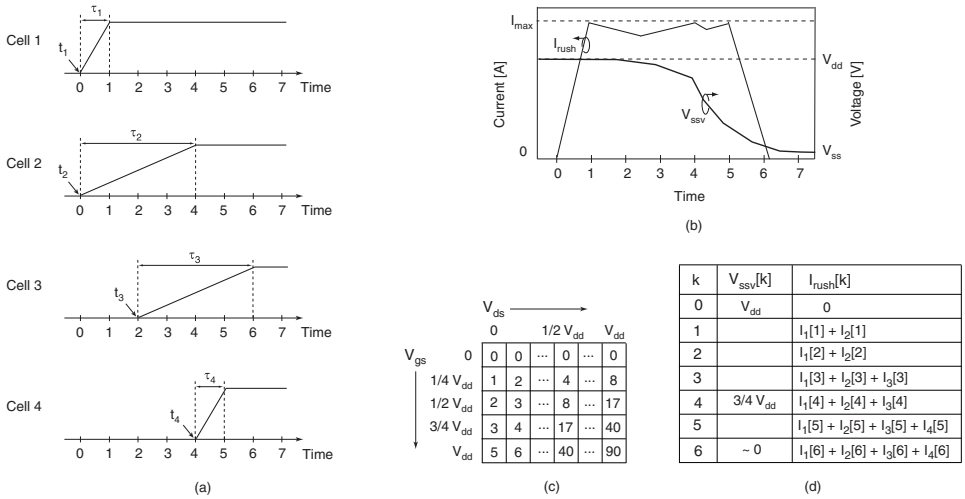
$$V_{ssv}[k] = V_{ssv}[k-1] - \frac{\sum_i I_i[k]\Delta t}{Q_{ssv}} V_{ssv}[0], \tag{2}$$

where Q_{ssv} denotes a total charge that has to be drained during wakeup process. In (2), we assume that V_{ssv} decreases linearly in proportion to the amount of charge that has been drained by footer cells during time interval Δt, $\sum_i I_i[k]\Delta t$, which was experimentally verified to be close to SPICE simulation. The above process repeats until we reach $3\Delta t$: $V_{ssv}[1]$ is used as an approximate value of $V_{ds}[2]$ to yield $I_1[2]$, $I_2[2]$, and $I_3[2]$, which are then used in (2) to derive $V_{ssv}[2]$, and so on.

The rush current of a footer cell can be characterized either analytically or by using a lookup table. A simple analytic model [16] has been used, which turned out to be very inaccurate in our experiments. We thus use a lookup table (LUT), which stores I_{ds} of a footer cell obtained by SPICE simulation for each combination of V_{gs} and V_{ds}. Each voltage is sampled at 11 evenly spaced discrete values in our implementation, which results in 121 entries in LUT. An interpolation is performed for the current that is not in the table; cubic spline method is selected for interpolation due to its balance of accuracy and computation time.

3.3 Synthesis Algorithm

We use an example shown in Figure 5 to explain the wakeup synthesis algorithm. The basic strategy is to turn on each footer cell as early as possible and make the wakeup signal as steep as possible so that the amount of total rush current (to drain the charges from V_{ssv})



k	$V_{ssv}[k]$	$I_{rush}[k]$
0	V_{dd}	0
1		$I_1[1] + I_2[1]$
2		$I_1[2] + I_2[2]$
3		$I_1[3] + I_2[3] + I_3[3]$
4	3/4 V_{dd}	$I_1[4] + I_2[4] + I_3[4]$
5		$I_1[5] + I_2[5] + I_3[5] + I_4[5]$
6	~ 0	$I_1[6] + I_2[6] + I_3[6] + I_4[6]$

Figure 5: Example of wakeup synthesis: (a) determine (t_i, τ_i) of four footer cells, (b) waveform of V_{ssv} and total rush current I_{rush}, (c) rush current LUT of a single footer cell, and (d) estimation of V_{ssv} and I_{rush} at each iteration.

is maximized while it does not exceed I_{max} thereby minimizing the wakeup delay.

Assume that $I_{max} = 100$ and signal slew is bounded by $1 \le \tau_i \le 4$. We first schedule cell 1 at time 0 with τ_{min} as signal slew as shown in Figure 5(a). Its rush current is 90, which is smaller than I_{max}, at $k = 1$ provided that V_{ssv} is still close to V_{dd} (see LUT in Figure 5(c) indexed by $V_{gs} = V_{dd}$ and $V_{ds} = V_{dd}$). When we schedule cell 2, we want to make t_2 as small as possible while $t_1 \le t_2$ (cells are assumed to be turned on in order of $1, 2, \ldots$) and τ_2 as small as possible while $\tau_1 \le \tau_2$. If $t_2 = 0$ and $\tau_2 = \tau_{min}$, the total rush current exceeds I_{max} at $k = 1$: $I_1[1] + I_2[1] = 90 + 90 = 180$. We therefore have to increase either t_2 or τ_2. In our approach, we increase signal slew first until rush current becomes smaller than I_{max}; if rush current still exceeds at τ_{max}, we then increase turn-on time. In the example, $\tau_2 = 4$ satisfies I_{max} constraint. At time 1, $I_1[1] = 90$ and $I_2[1] = 8$ (V_{gs} of cell 2 is $1/4V_{dd}$ and $V_{ds} = V_{dd}$, which can be used to look up its rush current from LUT). At time 4, $I_1[1] = 40$ and $I_2[1] = 40$ (V_{ssv} that determines V_{ds} is about 75% at $k = 4$ as shown in Figure 5(d)). When we schedule cell 3, τ_2 is already maximum signal slew; we therefore inherit the same signal slew for cell 3, i.e. $\tau_3 = \tau_{max}$; we then increase t_3 until total rush current from all three cells is kept below I_{max}, which can be shown to be $t_3 = 2$.

When the potential of V_{ssv} approaches V_{ss} (i.e V_{ds} of footer cells is very small), $I_i[k]$ becomes less sensitive to τ_i as can be identified from Figure 5(c). In this situation, large signal slew only increases the wakeup delay, i.e. cell may be fully turned on after V_{ssv} is already close to 0. In Figure 5(b), V_{ssv} approaches 0 around $k = 6$. It can be shown that $t_4 > 3$ to satisfy I_{max} constraint. Therefore, $\tau_4 = \tau_{max}$ would make cell 4 fully turned on after time 6; we therefore use $\tau_4 = \tau_{min}$ this time and determine the earliest time t_4 can take, which is 4 as shown in Figure 5(a).

Figure 6 outlines the wakeup synthesis algorithm, which the example of Figure 5 is based on.

Algorithm *Wakeup_Synthesis*
L1 $t \leftarrow 0$; $\tau \leftarrow \tau_{min}$;
L2 **while** cell i exists, which is not scheduled **do**
Step 1: determine signal slew with $t = 0$
L3 **if** $t = 0$ and $\tau < \tau_{max}$ **do**
L4 Determine the smallest value of τ such that $I_{rush} < I_{max}$
L5 **if** such τ does not exist **then** $\tau \leftarrow \tau_{max}$; goto L7
L6 **else** $(t_i, \tau_i) \leftarrow (0, \tau)$
Step 2: determine turn-on time with $\tau = \tau_{max}$
L7 **else if** $\tau = \tau_{max}$ **do**
L8 Determine the smallest value of t such that $I_{rush} < I_{max}$
L9 **if** $V_{ssv} \sim 0$ before $t + \tau_{max}$ **then** $\tau \leftarrow \tau_{min}$; goto L11
L10 **else** $(t_i, \tau_i) \leftarrow (t, \tau_{max})$
Step 3: determine turn-on time with $\tau = \tau_{min}$
L11 **else if** $\tau = \tau_{min}$ **do**
L12 Determine the smallest value of t such that $I_{rush} < I_{max}$
L13 $(t_i, \tau_i) \leftarrow (t, \tau_{min})$

Figure 6: Wakeup synthesis algorithm.

Algorithm *Buffered_Tree_Synthesis*
L1 **for** each cell i in increasing order of i **do**
L2 **if** (t_i, τ_i) is identical to (t_{i-1}, τ_{i-1}) **then**
L3 Place i to location nearest to $i - 1$
L4 **else**
L5 Place i to closest available location from wakeup signal pin
L6 Cluster footer cells with the same signal slew and turn-on time
L7 **for** each cluster j **do**
L8 $t_j \leftarrow t_i$, $\tau_j \leftarrow \tau_i$, $i \in j$
L9 Estimate load capacitance $C_j \leftarrow C_{load} + C_{wire}$
L10 Find a driving buffer B_j s.t. its signal slew is close to τ_j
L11 Set B_j as a new sink of wakeup signal and assign turn-on time
L12 Synthesize wakeup network to meet t_j of driving buffers

Figure 7: Algorithm of buffered tree synthesis for wakeup network.

4. BUFFERED TREE SYNTHESIS FOR WAKEUP NETWORK

The output of *Wakeup_Synthesis*, turn-on time and signal slew of each footer cell, is submitted to *Buffered_Tree_Synthesis* shown in Figure 7, to generate a buffered tree. A layout of power gating circuit, where potential locations of footer cells are specified, constitutes another input of the algorithm.

We do not consider the location of footer cells when we schedule them in *Wakeup_Synthesis*. Therefore, to implement wakeup network, we first need to determine the location of each scheduled

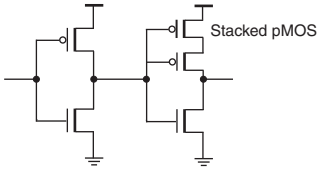


Figure 8: A normal inverter followed by a pMOS-stacked inverter.

footer cell out of the potential locations. To this end, we place each footer cell, in increasing order of i, to the closest available location from the wakeup signal pin (L5); this is to place a footer cell with smaller t_i to a location that is closer to the wakeup signal pin. However, if signal slew and turn-on time of i-th footer cell is identical to that of $(i-1)$-th footer cell, we place i-th cell to a location that is nearest to $(i-1)$-th footer cell (L2–L3); this is because we will eventually assign a single buffer to drive cells with the same signal slew and turn-on time and thus it is better for them to stay close to each other.

After the locations of all footer cells are determined, the cells with the same signal slew and turn-on time are grouped to form a cluster (L6). For each cluster j, we first realize its signal slew τ_j, which is determined by the load capacitance C_j and a driving buffer B_j, which drives all the cells in j. Note that C_j is a constant and can be estimated by summing up the gate capacitance of the footer cells in j and the capacitance of wires that connect them (L9). To obtain the wire capacitance, we estimate the length of the wire by using a minimum rectilinear Steiner tree [18] and multiply the unit capacitance of wire to it. Then B_j is determined by iteratively selecting a buffer from library* and performing SPICE simulation to check if its signal slew is close to τ_j (L10). Once the buffer is determined, it becomes the sink of wakeup signal and the turn-on time, which is obtained by subtracting the delay of buffer from t_j, is assigned to the buffer (L11). Then the wakeup network is synthesized by using industrial tool [1] to realize delay from wakeup signal pin to buffer B_j to satisfy its turn-on time; this is similar to clock tree synthesis with non-zero clock skew.

5. EXPERIMENTAL RESULTS

We carried out experiments on a set of sequential circuits taken from the ISCAS benchmark circuits, as well as on some circuits extracted from open cores [19]. The second and third columns of Table 1 show the numbers of combinational gates and flip-flops of each circuit, which was synthesized [20] in an industrial 1.1 V, 45-nm CMOS technology.

The voltage drop across footer, which affects active-mode circuit delay, was assumed to be 1% of V_{dd}; average discharge current was reported [21] by assuming 0.5 as a signal probability at all primary inputs; the voltage drop and discharge current were then used [17] to count the number of footer cells that are required for each design, which is reported in the fourth column. Each footer cell consists of 10.14 μm of gate width (W). The rush current constraint I_{max} shown in the fifth column was set to three times of average discharge current, which is one of empirical approaches used in practice to set maximum instantaneous current [8].

5.1 Wakeup Delay

In the remaining columns of Table 1, we compare wakeup delay from three different approaches. In 2-pass turn-on [9], which we briefly reviewed in Section 2.2, a group of footer cells are turned on at the same time; the number of cells in that group is determined by I_{max}, i.e. sum of their rush current is less than I_{max}. The remaining footer cells, which constitute the second group, are turned on after V_{ssv} becomes stable ($V_{ssv} \sim 0$). In turn-on scheduling [16] that determines the turn-on time alone, footer cells are scheduled as early as possible yet their overall rush current can be maintained less than I_{max}. This approach relies on analytic model to estimate rush current, which turned out to be very inaccurate. We thus manually delayed turn-on time whenever I_{max} constraint is violated. The result from our approach *Wakeup_Synthesis* is shown in the last column. We used τ_{min}=30 ps and τ_{max}=650 ps as a bound of signal slew; short-circuit current of buffered tree turned out to be very small in this range of signal slew.

*We used pMOS-stacked buffers shown in Figure 8, to realize a large slew even when C_j is small, as well as various sizes of standard buffers.

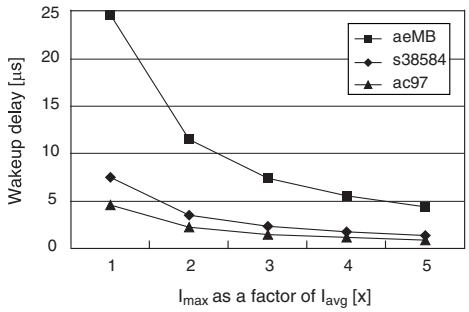


Figure 9: Wakeup delay under various rush current constraint I_{max}.

To obtain the delay number reported in Table 1, each circuit was placed and routed [1] followed by C_{ssv} being extracted [22]. A list of footer cells as well as their scheduling result, I_{max}, and C_{ssv} constitute the input of the routine to estimate rush current in Section 3.2, which then returns delay number. To assess the accuracy of this method, we compared the delay number to that reported by an industrial tool [22]; the difference was only 0.9% on average in all circuits we tested.

The wakeup delay obtained by *Wakeup_Synthesis* was always smaller than that obtained by the other two methods. Since 2-pass turn-on and turn-on scheduling do not adjust signal slew, we tried three different values of signal slew for fair comparison as shown in Table 1. The wakeup delay from *Wakeup_Synthesis* is smaller than that of 2-pass turn on by 43%∼44% on average, and is smaller than that of turn-on scheduling by 13%∼23%.

In Figure 9, we plot the wakeup delay obtained by *Wakeup_Synthesis* while we vary I_{max} constraint, which is expressed as a factor of average current. The wakeup delay decreases as I_{max} increases, since more footer cells can be turned on in earlier time of wakeup process. This reduction, however, becomes marginal when I_{max} constraint is large enough, which can be used to guide the trade-off between wakeup delay and power network design to sustain such I_{max}.

5.2 Design Flow

Figure 10 illustrates a design flow, where the proposed wakeup synthesis and buffered tree synthesis (shaded boxes) are integrated. Initial placement and routing is performed from a netlist of power gating circuit, which is followed by the extraction of C_{ssv}. The LUT of footer cell current and I_{max} constraint, in addition to C_{ssv}, are the input to wakeup synthesis, which outputs (t_i, τ_i) of each footer cell. The scheduling result and layout information form the input of buffered tree synthesis. The wakeup network is annotated to the original circuit, and incremental placement and routing is

Table 1: Comparison of wakeup delay from three different approaches. Delay is in ns. Average is a normalized value

Name	# Gates	# FFs	# Footer cells	I_{max} (mA)	2-pass turn-on			Turn-on scheduling			*Wakeup_Synthesis*
					$\tau = 30$ ps	50 ps	100 ps	$\tau = 30$ ps	50 ps	100 ps	
s38584	10897	1275	72	29	4.10	4.17	4.41	2.67	2.74	3.05	2.37
s38417	10073	1564	75	37	2.76	2.83	3.07	1.65	1.76	1.98	1.46
s35932	6107	1728	144	54	1.86	1.93	2.16	1.23	1.30	1.54	1.07
aquarius	25427	1211	22	21	16.32	16.39	16.63	9.48	9.55	9.76	8.99
or1200	27987	817	27	14	25.02	25.99	25.33	17.31	17.46	17.73	15.34
ucore	18391	1196	44	19	10.38	10.46	10.69	7.12	7.2	7.45	6.13
ac97	13107	2199	84	49	2.64	2.71	2.95	1.69	1.70	1.97	1.50
aeMB	25593	3330	96	29	12.91	12.98	13.22	8.65	8.79	8.92	7.84
aes_core	25322	530	120	40	5.44	5.51	5.74	3.49	3.56	3.85	3.03
des3	87566	8808	1056	223	2.65	2.73	2.96	1.81	1.91	2.23	1.55
Average					1.74	1.77	1.78	1.15	1.18	1.30	1.00

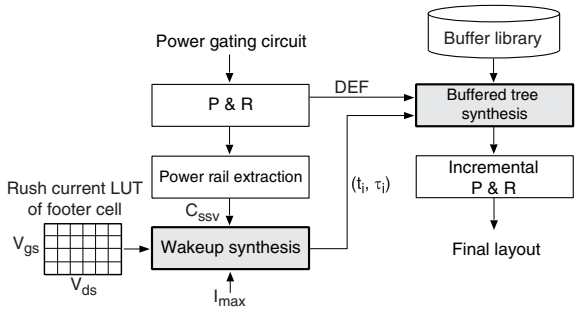


Figure 10: Wakeup synthesis and buffered tree synthesis integrated into a design flow.

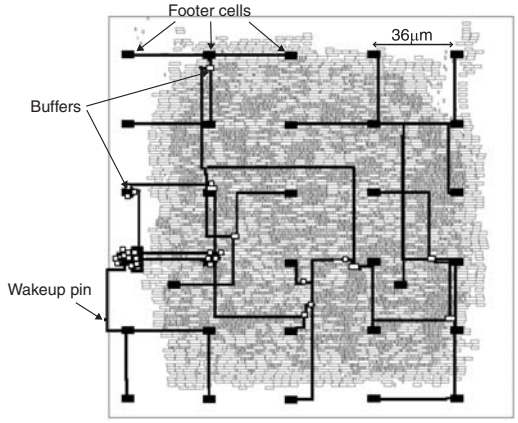


Figure 11: Layout of s38584 with wakeup network highlighted.

performed to complete the layout. Figure 11 shows a final layout of an example circuit s38584, which was obtained following the suggested design flow.

6. CONCLUSION

We have presented a wakeup synthesis method, which determines the turn-on time and signal slew of each footer cell; the objective is to minimize wakeup delay while rush current constraint is honored. The method to synthesize a buffered tree to implement a wakeup network has been addressed. The synthesis methods have been integrated into a design flow, and have been assessed using 1.1 V, 45-nm technology.

Rush current can cause ground bounce (or V_{dd} fluctuation when header is used) due to inductance of power network and chip package. The wakeup delay increases as a result. Wakeup synthesis while we take account of ground bounce in addition to rush current constraint is worth of further investigation.

Acknowledgement

This work was supported by Samsung Electronics.

7. REFERENCES

[1] Synopsys, "IC Compiler User Guide," Dec. 2008.
[2] Redhawk-ALP. Apache. [Online]. Available: http://www.apache-da.com/
[3] TSMC, "Reference Flow," 2009, http://www.tsmc.com/.
[4] M. Keating *et al.*, *Low Power Methodology Manual For System-on-Chip Design*. Springer, 2007.
[5] K. Usami *et al.*, "Automated selective multi-threshold design for ultra-low standby applications," in *Proc. Int. Symp. on Low Power Electronics and Design*, Aug. 2002, pp. 202–206.
[6] K. Shi and D. Howard, "Challenges in sleep transistor design and implementation in low-power designs," in *Proc. Design Automation Conf.*, July 2006, pp. 113–116.
[7] Synopsys, "NanoSim User Guide," Sept. 2008.
[8] A. Dharchoudhury *et al.*, "Design and analysis of power distribution networks in PowerPC microprocessors," pp. 738–743, June 1998.
[9] P. Royannez *et al.*, "90nm low leakage SoC design techniques for wireless applications," in *Proc. ISSCC*, Feb. 2006, pp. 138–139.
[10] K. Kumagai *et al.*, "A novel powering-down scheme for low Vt cmos circuits," in *Proc. Symp. on VLSI Circuits*, June 1998, pp. 44–45.
[11] A. Tada, H. Notani, and M. Numa, "A novel power gating scheme with charge recycling," *IEICE Electronics Express*, vol. 3, no. 12, pp. 281–286, June 2006.
[12] M. Horiguchi, T. Sakata, and K. Itoh, "Switched-source-impedance CMOS circuit for low standby subthreshold current giga-scale LSI's," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 11, pp. 1131–1135, Nov. 1983.
[13] K.-S. Min, H. Kawaguchi, and T. Sakurai, "Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: an alternative to clock-gating scheme in leakage dominant era," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2003, pp. 400–401.
[14] Y. Shin, S. Paik, and H. Kim, "Semicustom design of zigzag power-gated circuits in standard cell elements," *IEEE Trans. on Computer-Aided Design*, vol. 28, no. 3, pp. 327–339, Mar. 2009.
[15] K. Agarwal *et al.*, "Power gating with multiple sleep modes," in *Proc. Int. Symp. on Quality Electronic Design*, Mar. 2006, pp. 633–637.
[16] Y. Chen *et al.*, "An efficient wake-up schedule during power mode transition considering spurious glitches phenomenon," in *Proc. Int. Conf. on Computer Aided Design*, Nov. 2007, pp. 779–782.
[17] S. Mutoh *et al.*, "Design method of MTCMOS power switch for low-voltage high-speed LSIs," in *Proc. Asia South Pacific Design Automation Conf.*, Jan. 1999, pp. 113–116.
[18] GeoSteiner 3.1. University of Copenhagen. [Online]. Available: http://www.diku.dk/geosteiner/
[19] OpenCores. [Online]. Available: http://www.opencores.org/
[20] Synopsys, "Design Compiler User Guide," Sept. 2008.
[21] ——, "PrimeTime User Guide," June 2008.
[22] ——, "PrimeRail User Guide," Dec. 2008.