Sleepy Stack Leakage Reduction

Jun Cheol Park and Vincent J. Mooney III, *Senior Member, IEEE*

***Abstract*—Leakage power consumption of current CMOS technology is already a great challenge. International Technology Roadmap for Semiconductors projects that leakage power consumption may come to dominate total chip power consumption as the technology feature size shrinks. Leakage is a serious problem particularly for CMOS circuits in nanoscale technology. We propose a novel ultra-low leakage CMOS circuit structure which we call "sleepy stack." Unlike many other previous approaches, sleepy stack can retain logic state during sleep mode while achieving ultra-low leakage power consumption. We apply the sleepy stack to generic logic circuits. Although the sleepy stack incurs some delay and area overhead, the sleepy stack technique achieves the lowest leakage power consumption among known state-saving leakage reduction techniques, thus, providing circuit designers with new choices to handle the leakage power problem.**

***Index Terms*—Dual-$V_{\rm th}$, low-leakage power dissipation, transistor stacking.**

I. Introduction

POWER consumption is one of the top concerns of VLSI circuit design, for which CMOS is the primary technology. Today's focus on low power is not only because of the recent growing demands of mobile applications. Even before the mobile era, power consumption has been a fundamental problem. To solve the power dissipation problem, many researchers have proposed different ideas from the device level to the architectural level and above. However, there is no universal way to avoid tradeoffs between power, delay, and area, and thus, designers are required to choose appropriate techniques that satisfy application and product needs.

Power consumption of CMOS consists of dynamic and static components. Dynamic power is consumed when transistors are switching and static power is consumed regardless of transistor switching. Dynamic power consumption was previously (at 0.18-μm technology and above) the single largest concern for low-power chip designers since dynamic power accounted for 90% or more of the total chip power. Therefore, many previously proposed techniques, such as voltage and frequency scaling, focused on dynamic power reduction. However, as the feature size shrinks, e.g., to 0.09 and 0.065 μm, static power has become a great challenge for current and future technologies. Based on the International Technology Roadmap for Semiconductors (ITRS) [1], Kim *et al.* report that subthreshold leakage power dissipation of a chip may exceed dynamic power dissipation at the 65-nm feature size [2].

Manuscript received August 5, 2005; revised July 7, 2006.

J. C. Park is with the Mobility Group, Intel Corporation, Folsom, CA 95630 USA (e-mail: juncheol.park@intel.com).

V. J. Mooney III is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: mooney@ece.gatech.edu).

Digital Object Identifier 10.1109/TVLSI.2006.886398

One of the main reasons causing the leakage power increase is the increase of subthreshold leakage power. When technology feature size scales down, supply voltage and threshold voltage also scale down. Subthreshold leakage power increases exponentially as threshold voltage decreases. Furthermore, the structure of the short channel device decreases the threshold voltage even lower. In addition to subthreshold leakage, another contributor to leakage power is gate-oxide leakage power due to the tunneling current through the gate-oxide insulator. Since gate-oxide thickness may reduce as the channel length decreases, in sub 0.1-μm technology, gate-oxide leakage power may be comparable to subthreshold leakage power if not handled properly. However, we assume other techniques will address gate-oxide leakage; for example, high-k dielectric gate insulators may provide a solution to reduce gate-leakage [2]. Therefore, this paper focuses on reducing subthreshold leakage power consumption.

In this paper, we provide a new circuit structure named "sleepy stack" as a remedy for static power consumption. The sleepy stack has a novel structure that uniquely combines the advantages of two major prior approaches, the sleep transistor technique and the forced stack technique. However, unlike the sleep transistor technique, the sleepy stack technique retains the original state; furthermore, unlike the forced stack technique, the sleepy stack technique can utilize high-$V_{\rm th}$ to achieve up to two orders of magnitude leakage power reduction compared to the forced stack. Unfortunately, the sleepy stack technique comes with delay and area overheads. Therefore, the sleepy stack technique provides new Pareto points [3] to designers who require ultra-low leakage power consumption and are willing to pay some area and delay cost.

The main contributions of this paper are as follows: 1) introduction of a sleepy stack structure that can save leakage power up to two orders of magnitude for circuits that require extremely low leakage power consumption and 2) analysis of example sleepy stack logic circuits in terms of various ways (transistor scaling, threshold voltage, and transistor width) circuit design engineers can employ to adopt the sleepy stack technique as necessary.

This paper is organized as follows. In Section II, prior work about low-leakage logic design is discussed. In Section III, the sleepy stack structure is explained and an analytical delay model is discussed. In Section IV, an empirical methodology applying the sleepy stack to generic logic is explained. In Section V, the experimental results of the sleepy stack for generic logic is presented. In Section VI, conclusions are given.

II. Previous Work

In this section, we discuss previous low-power techniques that primarily target reducing leakage power consumption of CMOS circuits. Techniques for leakage power reduction can

be grouped into the following two categories: 1) state-saving techniques where circuit state (present value) is retained and 2) state-destructive techniques where the current Boolean output value of the circuit might be lost [2]. A state-saving technique has an advantage over a state-destructive technique in that with a state-saving technique the circuitry can immediately resume operation at a point much later in time without having to somehow regenerate state. We characterize each low-leakage technique according to this criterion.

State-destructive techniques cut off transistor (pull-up or pull-down or both) networks from supply voltage or ground using sleep transistors [4]. These types of techniques are also called gated-$V_{\rm dd}$ and gated-Gnd (note that a gated clock is generally used for dynamic power reduction). Motoh *et al.* propose a technique they call multithreshold-voltage CMOS (MTCMOS) [4], which adds high-$V_{\rm th}$ sleep transistors between pull-up networks and $V_{\rm dd}$ and between pull-down networks and ground while logic circuits use low-$V_{\rm th}$ transistors in order to maintain fast logic switching speeds. The sleep transistors are turned off when the logic circuits are not in use. By isolating the logic networks using sleep transistors, the sleep transistor technique dramatically reduces leakage power during sleep mode. However, the additional sleep transistors increase area and delay. Furthermore, during sleep mode, the pull-up and pull-down networks will have floating values and, thus, will lose state. These floating values significantly impact the wake-up time and energy of the sleep technique due to the requirement to recharge transistors which lost state during sleep (this issue is nontrivial, especially for registers and flip-flops).

To reduce the wake-up cost of the sleep transistor technique, the zigzag technique is introduced [5]. The zigzag technique reduces the wake-up overhead by choosing a particular circuit state (e.g., corresponding to a "reset") and then, for the exact circuit state chosen, turning off the pull-down network for each gate whose output is high while conversely turning off the pull-up network for each gate whose output is low.

By applying, prior to going to sleep, the particular input pattern chosen prior to chip fabrication, the zigzag technique can prevent floating. Although the zigzag technique retains the particular state chosen prior to chip fabrication, any other arbitrary state during regular operation is lost in power-down mode.

Another technique to reduce leakage power is transistor stacking. Transistor stacking exploits the stack effect; the stack effect results in substantial subthreshold leakage current reduction when two or more stacked transistors are turned off together. Narendra *et al.* study the effectiveness of the stack effect including effects from increasing the channel length [6]. Since forced stacking of what previously was a single transistor increases delay, Johnson *et al.* propose an algorithm that finds circuit input vectors that maximize stacked transistors of existing complex logic [7]. As a variation of the stacking transistors, Hanchate and Ranganathan introduce self-controlled stacked transistors which are inserted between pull-up and pull-down networks and reduce leakage power by increasing internal resistance [8].

Our sleepy stack structure can achieve more power savings

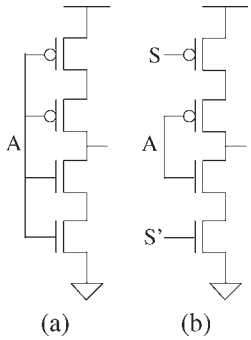

Fig. 1. (a) Forced stack technique applied to an inverter. (b) Sleep transistor technique applied to an inverter.

stack transistor or the self-controlled stacked transistors). Furthermore, the sleepy stack can save exact logic state unlike gated-$V_{\rm dd}$ and gated-Gnd techniques (conventional sleep transistor technique) and the zigzag technique.

In Section III, we will discuss the sleepy stack structure and sleepy stack operation.

III. Sleepy Stack Structure

We introduce our new leakage power reduction technique we name "sleepy stack." The sleepy stack technique has a combined structure of the forced stack technique and the sleep transistor technique. However, unlike the sleep transistor technique, the sleepy stack technique retains exact logic state when in sleep mode; furthermore, unlike the forced stack technique, the sleepy stack technique can utilize high-$V_{\rm th}$ transistors without 5$\times$ (or greater) delay penalties. Therefore, far better than any prior approach known to the authors of this paper, the sleepy stack technique can achieve ultra-low leakage power consumption while saving state.

We, first, explain the structure of the sleepy stack technique using an inverter. Then, we describe the details of sleepy stack operation in active mode and sleep mode. The advantages of the sleepy stack technique over the forced stack technique and the sleep transistor technique are explored. Finally, we derive a first-order delay model that compares the sleepy stack technique to the forced stack technique analytically.

A. Sleepy Stack Approach

In this section, we explain our sleepy stack structure comparing to the forced stack technique and the sleep transistor technique. The details of the sleepy stack inverter are described as an example. Two operation modes, active mode and sleep mode, of the sleepy stack technique are explored.

1) Sleepy Stack Structure: The sleepy stack structure has a combined structure of the forced stack and the sleep transistor techniques. Although we mentioned these two techniques in Section II, we focus on explaining forced stack and sleep transistor inverters here for the purposes of comparison with a sleepy stack inverter. Fig. 1(a) depicts a forced stack inverter and Fig. 1(b) depicts a sleep transistor inverter. The forced stack inverter breaks existing transistors into two transistors and forces a

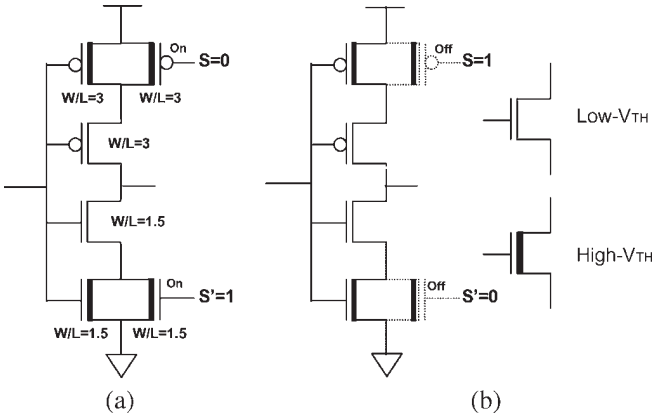

Fig. 2. (a) Sleepy stack inverter with W/L of each transistor and active mode S, S' assertion. (b) Sleep mode S, S' assertion.

Fig. 1(b) isolates existing logic networks using sleep transistors. The stack structure in Fig. 1(b) saves leakage power consumption during sleep mode. This sleep transistor technique frequently uses high-$V_{\rm th}$ sleep transistors (the transistors controlled by S and S') to achieve larger leakage power reduction.

The sleepy stack technique has a structure merging the forced stack technique and the sleep transistor technique. Fig. 2 shows a sleepy stack inverter. The sleepy stack technique divides existing transistors into two transistors each typically with the same width W_1 half the size of the original single transistor's width W_0 (i.e., $W_1 = W_0/2$), thus, maintaining equivalent input capacitance. The sleepy stack inverter in Fig. 2(a) uses $W/L = 3$ for the pull-up transistors and $W/L = 1.5$ for the pull-down transistors, while a conventional inverter with the same input capacitance would use $W/L = 6$ for the pull-up transistor and $W/L = 3$ for the pull-down transistor (assuming $\mu_n = 2\mu_p$). Then sleep transistors are added in parallel to one of the transistors in each set of two stacked transistors. We use a transistor sized as half the width of the original transistor (i.e., we use $W_0/2$) for the sleep transistor width of the sleepy stack. Although we exclusively use $W_0/2$ for the width of the sleep transistor, changing the sleep transistor width in various ways may provide additional tradeoffs between delay, power, and area. However, in this paper, we mainly focus on applying the sleepy stack structure with $W_0/2$ sleep transistor widths to generic logic circuits while varying technology feature size, threshold voltage, and temperature. Please note that halving transistor width is not possible for a circuit that uses minimum size transistors. However, many circuits use nonminimum size to gain driving strength. In any case, if we cannot halve transistor width, then we simply use minimum width.

2) Sleepy Stack Operation: Now we explain how the sleepy stack works during active mode and during sleep mode. Also, we explain leakage power savings using the sleepy stack structure.

The sleep transistors of the sleepy stack operate similar to the sleep transistors used in the sleep transistor technique in which sleep transistors are turned on during active mode and turned off during sleep mode. Fig. 2 depicts the sleepy stack operation

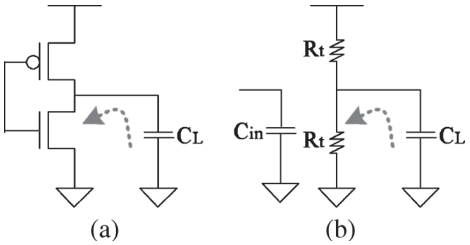

Fig. 3. (a) Inverter circuit schematic. (b) RC equivalent circuit.

are turned on. This sleepy stack structure can potentially reduce circuit delay in two ways. First, since the sleep transistors are always on during active mode, the sleepy stack structure achieves faster switching time than the forced stack structure; specifically, in Fig. 2(a), at each sleep transistor drain, the voltage value connected to the sleep transistor source is always ready and available at the sleep transistor drain, and thus, current flow is immediately available to the low-$V_{\rm th}$ transistors connected to the gate output regardless of the status of each transistor in parallel to the sleep transistors. Furthermore, we can use high-$V_{\rm th}$ transistors (which are slow but 1000× or so less leaky) for the sleep transistors and the transistors parallel to the sleep transistors (see Fig. 2) without incurring large (e.g., 2× or more) delay increase.

During sleep mode [Fig. 2(b)], $S = 1$ and $S' = 0$ are asserted, and so both of the sleep transistors are turned off. Although the sleep transistors are turned off, the sleepy stack structure maintains exact logic state. The leakage reduction of the sleepy stack structure occurs in two ways. First, leakage power is suppressed by high-$V_{\rm th}$ transistors, which are applied to the sleep transistors and the transistors parallel to the sleep transistors. Second, stacked and turned off transistors induce the stack effect [11], which also suppresses leakage power consumption. By combining these two effects, the sleepy stack structure achieves ultra-low leakage power consumption during sleep mode while retaining exact logic state. The price for this, however, is increased area.

We will derive an analytical delay model of the sleepy stack inverter and compare the sleepy stack technique to the forced stack inverter in the next section. This analytical comparison of the next section, Section III-B, can be skipped if desired. The detailed experimental methodology and the results will be presented in Section IV.

B. Analytical Comparison of Sleepy Stack Inverter Versus Forced Stack Inverter

In this section, an analytical delay model of a sleepy stack inverter is explained and compared to a forced stack inverter, the best prior state-saving leakage reduction technique we could find.

Generally, the transistor delay of a conventional inverter shown in Fig. 3 driving a load of C_L can be expressed using the following equation:

$$T_{d0} = C_L R_t \tag{1}$$

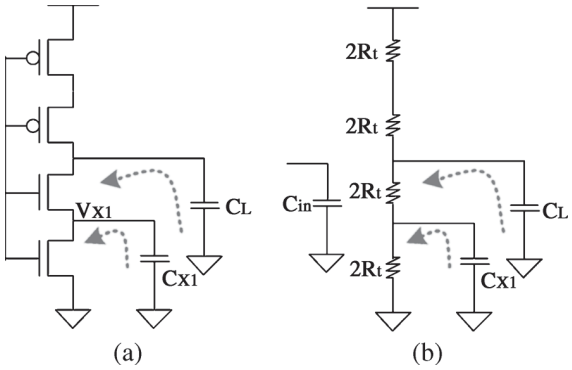

Fig. 4. (a) Forced stack technique inverter circuit schematic. (b) *RC* equivalent circuit.

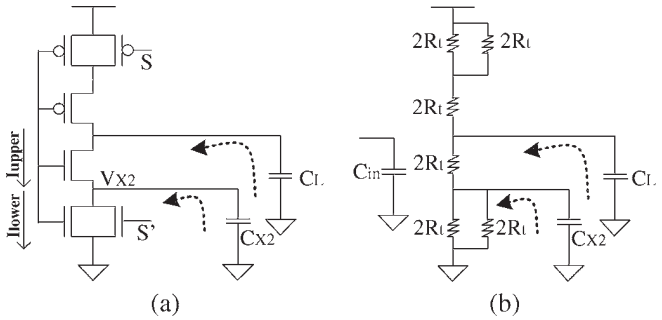

Fig. 5. (a) Sleepy stack technique inverter schematic. (b) *RC* equivalent circuit.

nonsaturation mode equation is complicated, we can predict the adequate first-order gate delay from (1) [14].

Now we derive the delay of the inverter with the forced stack technique shown in Fig. 4. Since we assume that we break each existing transistor into two half sized transistors (see Section III-A1), the resistance of each transistor of the forced stack technique is doubled, i.e., $2R_t$, compared to the standard inverter; furthermore, in this way, we can maintain input capacitance equal to Fig. 3(b). In Fig. 4, C_{x1} is internal node capacitance between the two pull-down transistors. Using the Elmore equation [10], we can express the delay of the forced stack inverter as follows:

$$T_{d1} = (2R_t + 2R_t)C_L + 2R_tC_{x1} \tag{2}$$
$$= 4R_tC_L + 2R_tC_{x1}. \tag{3}$$

Similarly, we can depict the sleepy stack inverter and its resistance-capacitance (*RC*) equivalent circuit as shown in Fig. 5. Two extra sleep transistors are added and each sleep transistor has a resistance of $2R_t$ (as discussed in Section III-A1, please note that increasing sleep transistor width reduces the sleep transistor resistance further—however, let us continue with the approach of Section III-A). The internal node capacitance is C_{x2}.

Using the Elmore equation, we can derive the transistor delay of the sleepy stack inverter as follows:

$$T_{d2} = (2R_t + R_t)C_L + R_tC_{x2} \tag{4}$$

We assume that the internal node capacitance C_{x2} is 50% larger than C_{x1} because C_{x2} is the capacitance from three transistors connected, while C_{x1} is the capacitance from two transistors connected. Then

$$R_tC_{x2} = 1.5R_tC_{x1} \tag{6}$$
$$T_{d2} = 3R_tC_L + 1.5R_tC_{x1} = \frac{3}{4}T_{d1}. \tag{7}$$

Therefore, T_{d2} is 25% faster than T_{d1} if we use the same $V_{\rm dd}$ and $V_{\rm th}$ for the forced stack inverter and the sleepy stack inverter. Alternatively, we may increase $V_{\rm th}$ of the sleepy stack inverter and make the delay of the sleepy stack inverter and the delay of the forced stack inverter the same.

Let us take an example. The gate delay of a CMOS circuit can be expressed as shown in the following approximated equation:

$$T_d \propto \frac{V_{\rm dd}}{(V_{\rm dd} - V_{\rm th})^{\alpha}} \tag{8}$$

where T_d, $V_{\rm th}$, and α denote the gate delay in a CMOS circuit, the threshold voltage, and velocity saturation index of a transistor, respectively. Using (8), the delay of the forced stack (T_{d1}) and the delay of the sleepy stack (T_{d2}) can be expressed as follows:

$$T_{d1} = K_1\frac{V_{\rm dd}}{(V_{\rm dd} - V_{\rm th1})^{\alpha}} \tag{9}$$
$$T_{d2} = K_2\frac{V_{\rm dd}}{(V_{\rm dd} - V_{\rm th2})^{\alpha}} \tag{10}$$

where K_1 and K_2 are delay coefficients of the forced stack inverter and the sleepy stack inverter, respectively. When the threshold voltage of the forced stack $V_{\rm th1}$ is the same as the threshold voltage of the sleepy stack $V_{\rm th2}$, we calculate $K_2 = 0.75K_1$ from (7). If we assume that $\alpha = 1.3$, $V_{\rm dd} = 1$ V, and $V_{\rm th} = 0.25$ V, we can make T_{d1} equal to T_{d2} by applying $V_{\rm th2} = 0.423$, which is 69% higher than the $V_{\rm th1}$ of the forced stack inverter. This higher $V_{\rm th}$ can potentially result in large leakage power reduction (e.g., 10×).

In this section, we introduced the sleepy stack technique for leakage power reduction. By combining the forced stack technique and the sleep transistor technique, the sleepy stack can achieve smaller transistor delay than the forced stack technique while retaining state unlike the sleep transistor technique. The main advantage of the sleepy stack approach is the ability to use high-$V_{\rm th}$ for both the sleep transistors and the transistors in parallel with the sleep transistors. The increased threshold voltage transistors of the sleepy stack technique potentially brings much larger (>10×) leakage power reduction than the forced stack technique while achieving the same transistor delay. From the analytical model of the sleepy stack inverter, we observe that the sleepy stack inverter can reduce delay by 25%, which alternatively can be used to increase $V_{\rm th}$ by 69%. Using this increased threshold voltage, the sleepy stack inverter can potentially achieve a large (e.g., 10×) leakage power reduction compared to the forced stack inverter.

In this section, we explained the sleepy stack structure and sleepy stack operation. We also described a first-order delay

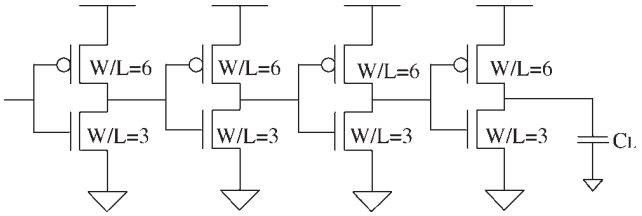

Fig. 6. Chain of four inverters with W/L of each transistor.

HSPICE—see Section IV-C). In the next sections, we apply the sleepy stack structure to generic logic circuits, explaining in detail our methodology.

IV. Applying Sleepy Stack to Logic Circuits

In this section, we first explain target benchmark circuits we use focusing on generic logic to evaluate our sleepy stack technique [11]. Then we explain low-leakage techniques we consider for purposes of comparison; although the basic ideas of the compared techniques have been covered in Section II, this section will give detailed structure with transistor sizing for each prior technique to be compared to our sleepy stack approach. Finally, we explain experimental methodology that we use to compare our technique to the previous techniques we consider.

A. Benchmark Circuits

To show that the sleepy stack technique is applicable to general logic design, we choose three benchmark circuits, which are as follows: 1) a chain of 4 inverters; 2) a 4:1 multiplexer; and 3) a 4-bit adder.

1) Chain of Four Inverters: A chain of four inverters shown in Fig. 6 is chosen because an inverter is one of the most basic CMOS circuits and is typically used to study circuit characteristics. We size each transistor of the inverter to have equal rise and fall times in each stage. Instead of using the minimum possible size of the transistor in a given technology, we use $W/L = 6$ for pMOS and $W/L = 3$ for nMOS transistors. Please refer to [12] for a layout of the chain of four inverters in TSMC 0.18-μm technology using the widths shown in Fig. 6; note that in Fig. 6, for 0.18-μm technology, all pMOS transistors have $W = 1.08$ μm and $L = 0.18$ μm while all nMOS transistors have $W = 0.54$ μm and $L = 0.18$ μm.

2) 4:1 Multiplexer: A possible implementation of a 4:1 multiplexer is shown in Fig. 7, in which $I_0 - I_3$ are input signals, S_0 and S_1 are selection signals, and E is an enable signal. The multiplexer consists of an inverter, two-input NAND gates, and two-input NOR gates. All gates are sized to have rise and fall times equal to an inverter with pMOS $W/L = 6$ and nMOS $W/L = 3$. Although the 4:1 multiplexer shown in Fig. 7 is not the most efficient way to implement a 4:1 multiplexer, we use the design of Fig. 7 to show that the sleepy stack can be applicable to a combination of (a logic network of) typical CMOS gates. Please refer to [12] for NAND and NOR layouts used in this 4:1 multiplexer.

3) 4-Bit Adder: By use of the 1-bit full adder shown in Fig. 8, we implement a 4-bit adder. A full adder is an example of a

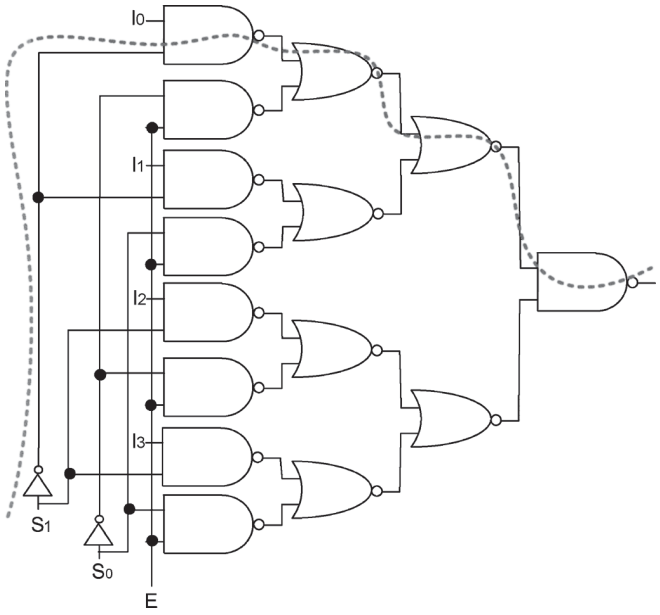

Fig. 7. 4:1 multiplexer with delay critical path along the dashed line.

sizing of the full adder is noted in Fig. 8. Please refer to [12] for the full adder layout we use.

These three benchmark circuits (chain of 4 inverters, 4:1 multiplexer, and 4-bit adder) designed in a conventional CMOS structure are used as our base case. In the next section, we explain the low-leakage techniques to which we compare to our sleepy stack technique. These three benchmark circuits are also implemented using the low-leakage techniques explained in the next section, Section IV-B.

B. Prior Low-Leakage Techniques Considered for Comparison Purposes

The sleepy stack technique is compared to a conventional CMOS approach, which is our base case, and three other well-known previous approaches, i.e., the forced stack, sleep, and zigzag techniques explained in Section II. We also explore the impact of $V_{\rm th}$ and transistor width on the sleepy stack technique.

1) Base Case: In this paper, we use the phrase "base case" to refer to the conventional CMOS technique shown in Fig. 9 and described in a classic textbook by Weste and Eshraghian [13]. Fig. 9 shows a pull-up network and a pull-down network using as few transistors as possible to implement the Boolean logic function desired. The base case of a chain of four inverters is sized as explained in Section IV-A1. The base case of a 4:1 multiplexer is sized as explained in Section IV-A2. The base case of a 4-bit adder is sized as explained in Section IV-A3.

2) Sleepy Stack Technique: Fig. 10 shows the sleepy stack technique applied to a conventional CMOS design. When we apply the sleepy stack technique, we replace each existing transistor with two half sized transistors and add one extra sleep transistor as shown in Fig. 10. If dual-$V_{\rm th}$ values are available, high-$V_{\rm th}$ transistors are used for sleep transistors and transistors that are parallel to the sleep transistors.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.