Active Leakage Power Optimization for FPGAs

Jason H. Anderson, *Student Member, IEEE*, and Farid N. Najm, *Fellow, IEEE*

***Abstract*—Active leakage power dissipation is considered in field-programmable gate arrays (FPGAs) and two "no cost" approaches for active leakage reduction are presented. It is well known that the leakage power consumed by a digital CMOS circuit depends strongly on the state of its inputs. The authors' first leakage reduction technique leverages a fundamental property of basic FPGA logic elements [look-up tables (LUTs)] that allows a logic signal in an FPGA design to be interchanged with its complemented form without any area or delay penalty. This property is applied to select polarities for logic signals so that FPGA hardware structures spend the majority of time in low-leakage states. In an experimental study, active leakage power is optimized in circuits mapped into a state-of-the-art 90-nm commercial FPGA. Results show that the proposed approach reduces active leakage by 25%, on average. The authors' second approach to leakage optimization consists of altering the routing step of the FPGA computer-aided design (CAD) flow to encourage more frequent use of routing resources that have low leakage power consumptions. Such "leakage-aware routing" allows active leakage to be further reduced, without compromising design performance. Combined, the two approaches offer a total active leakage power reduction of 30%, on average.**

***Index Terms*—Computer-aided design, field-programmable gate arrays (FPGAs), leakage, optimization, power.**

I. Introduction

TRENDS in technology scaling make leakage power an increasingly dominant component of total power dissipation. Leakage power has two main forms in modern integrated circuit (IC) processes: 1) subthreshold leakage; and 2) gate leakage. Subthreshold leakage power is due to a nonzero current between the source and drain terminals of an OFF metal–oxide semiconductor (MOS) transistor. With each process generation, supply voltages are reduced and transistor threshold voltages (V_{TH}) must also be reduced to mitigate performance degradations. Reducing V_{TH} leads to an exponential increase in subthreshold leakage. Gate leakage, on the other hand, is due to tunneling current through the gate oxide of an MOS transistor. In modern IC processes, gate oxides are thinned to improve transistor drive capability, which has led to a considerable increase in gate leakage. Leakage power is a growing concern in complementary metal oxide semiconductor (CMOS) design, and a recent work suggests that it may constitute over 40% of total power at the 70-nm technology node [1].

Field-programmable gate arrays (FPGAs) are a popular choice for digital circuit implementation because of their growing density and speed, short design cycle, and steadily decreasing cost. Several recent works have studied FPGA power consumption [2]–[4] and have shown that the power consumed by the largest FPGA devices is increasing, with such devices now consuming watts of power [3]. These prior works have been mainly concerned with dynamic power consumption (due to logic transitions on the signals of a circuit) and suggest leakage power to be a small component of total power. However, these analyses have been based on IC technologies having feature sizes of 0.15 μm or larger, making them somewhat out of step with today's state-of-the-art FPGAs, which are fabricated in sub-100-nm technology [5]. The programmability of FPGAs implies that more transistors are needed to implement a given logic circuit, in comparison with custom application-specific integrated circuit (ASIC) technologies. Leakage power is proportional to total transistor count, and consequently, leakage optimization will likely be a key design objective in future FPGA technologies. Reducing the power consumption of FPGAs is beneficial as it lowers packaging/cooling costs, improves reliability, and enables FPGA usage in low-power applications, such as mobile electronics.

Unlike ASICs, an FPGA circuit implementation uses only a fraction of the FPGA's resources. Leakage power is dissipated in both the used and the unused part of the FPGA. Prior work on leakage optimization differentiates between active mode and sleep (standby) mode leakage power. Standby leakage power is that consumed in circuit blocks that are temporarily inactive and that have been put into a special "sleep" state, in which leakage is minimized. The sleep concept is commonly used for leakage power reduction in the ASIC domain; however, support for a sleep mode has yet to appear in commercial FPGAs. Active leakage power, on the other hand, is that consumed in circuit blocks that are "awake" (blocks that are in use). The absence of sleep mode support in current FPGAs implies that at present, all leakage power dissipated in the used part of an FPGA can be considered active leakage.

In this paper, the focus is on optimizing active leakage power dissipation in FPGAs. How the leakage power of typical FPGA hardware structures depends strongly on the state of their inputs is illustrated. A novel leakage reduction approach that leverages a property of basic FPGA logic elements that allows either polarity of a logic signal to be used without any area or delay penalty, and without any modifications to the underlying FPGA hardware, is then presented. Polarities are intelligently chosen for signals in a way that places hardware structures into their low-leakage states. Following this, a second

Manuscript received July 12, 2004; revised December 9, 2004. This work was supported in part by a Natural Sciences and Engineering Research Council of Canada Postgraduate Scholarship and an Ontario Graduate Scholarship. This paper was recommended by Editor-in-Chief E. Macii.

J. H. Anderson is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada, and also with Xilinx, Inc., Toronto, ON M4V 3A1, Canada (e-mail: janders@eecg.toronto.edu).

F. N. Najm is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: f.najm@utoronto.ca).

Digital Object Identifier 10.1109/TCAD.2005.853692

leakage optimization technique in which the leakage power consumptions of FPGA routing resources are taken into account during the routing step of the FPGA computer-aided design (CAD) flow is presented. The objective of such "leakage-aware routing" is to produce routing solutions in which a design's signals are routed using low-leakage routing resources.

The remainder of the paper is organized as follows: Section II discusses related work on leakage optimization in ASICs and microprocessors. Section III describes typical FPGA hardware structures, studies their leakage power characteristics, and reviews recent published work on leakage optimization in FPGAs. Section IV describes the first approach to leakage reduction, based on intelligent polarity selection. Section V presents the second leakage optimization technique: leakage-aware routing. Both of the proposed leakage reduction approaches are validated experimentally by applying them to optimize leakage in a 90-nm Xilinx commercial FPGA. Section VI presents the conclusions. A preliminary version of a portion of this work has appeared in [6].

II. Leakage Power Optimization

In this section, a few of the important leakage reduction techniques used in ASICs and microprocessors are summarized. A more detailed overview can be found in [7].

Several recent works have considered standby leakage power optimization. In [8] and [9], the authors introduce high-threshold sleep transistors into the N-network (or P-network) of CMOS gates. Sleep transistors are ON when a circuit is active and are turned OFF when the circuit is in standby mode, effectively limiting the leakage current from supply to ground. A different approach to leakage reduction (and one that is related to the first of the proposed techniques) is based on the fact that a circuit's leakage depends on its input state. In [10] and [11], a specific input vector that minimizes leakage power in a circuit is identified; the vector is then applied to circuit inputs when the circuit is placed in standby mode. This idea requires only minor circuit modifications and has been shown to reduce leakage by up to 70% in some circuits [11].

Active leakage reduction has also been addressed in the literature. One approach performs dynamic V_{TH} adjustment based on system workload [12], [13]. The body effect is used to raise transistor V_{TH} when high system throughput is not required and the circuit can be slowed down. Such body bias methods can also be used for standby leakage power reduction [14]. Other circuit-level techniques include the use of multi- or dual-threshold CMOS [15], [16], in which transistors having different threshold voltages are available. In this approach, low-V_{TH} transistors are used in delay critical paths, and high-V_{TH} transistors are used in noncritical paths. Considerable leakage power reductions are possible, as there are usually few delay critical paths. Another popular technique is to replace individual transistors in gates with "stacks" of transistors in series [1], [17], [18]; transistor stacks leak less than individual transistors when in the OFF state. A related approach is to use transistors with longer channel lengths, which are known to have better leakage characteristics [7]. Note that the leakage improvements offered by the techniques mentioned here do not come for free—each has an associated cost, impacting circuit area, delay, or fabrication cost.

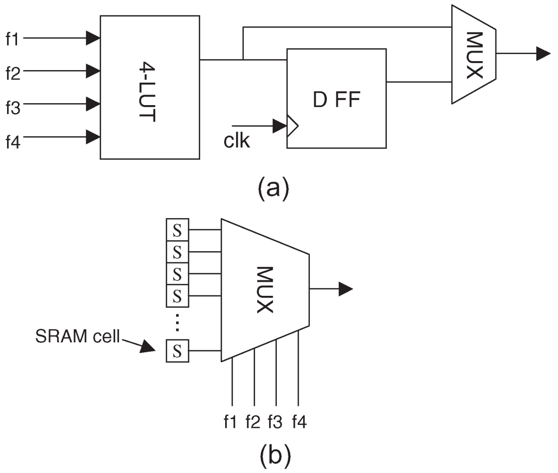


Fig. 1. FPGA logic block. (a) Logic block. (b) 4-LUT.

III. FPGA Hardware Structures and Leakage Characteristics

Before describing the proposed leakage reduction methods, the circuit structures that are common to current FPGAs are reviewed and their leakage characteristics are studied.

FPGAs consist of an array of programmable logic blocks that are connected through a programmable interconnection network. Most commercial FPGAs use four-input look-up tables (4-LUTs) as the combinational logic element in their logic blocks. 4-LUTs are small memories that can implement any logic function having no more than four inputs. An abstract view of an FPGA logic block is shown in Fig. 1(a), comprising a 4-LUT along with a flip-flop (flip-flop can be bypassed). Fig. 1(b) shows the internal details of a 4-LUT. Sixteen static random access memory (SRAM) cells hold the truth table for the logic function implemented by the LUT. The LUT inputs (labeled f1–f4) select a particular SRAM cell whose content is passed to the LUT output. Note that logic blocks in commercial FPGAs contain clusters of LUTs and flip-flops. For example, a logic block in the Xilinx Virtex-II PRO FPGA contains eight LUTs and eight flip-flops [19].

Connections between logic blocks in an FPGA are formed using a programmable interconnection network, composed of variable length wire segments and programmable routing switches. A typical FPGA routing switch is shown in Fig. 2 [20], [21]. It consists of a multiplexer, a buffer, and SRAM configuration bits. The multiplexer inputs (labeled $i1 - in$) connect to other routing conductors or to logic block outputs. The buffer's output connects to a routing conductor or to a logic block input. The programmability of an FPGA's interconnection fabric is realized through SRAM cells in the configuration block (labeled "config" in Fig. 2). The SRAM cell contents control which input signal is selected to be driven through the buffer.

The multiplexers in FPGA interconnect and LUTs are typically implemented using NMOS transistor trees [20], such as

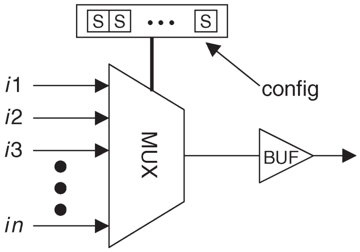


Fig. 2. Routing switch.

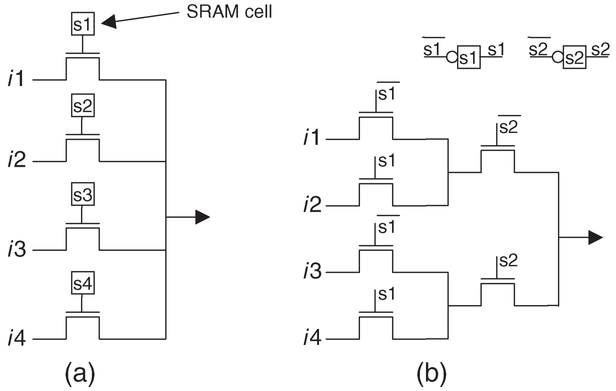


Fig. 3. Multiplexer implementations. (a) Decoded multiplexer. (b) Encoded multiplexer.

those shown in Fig. 3. Note that full CMOS transmission gates are generally not used to implement multiplexers in FPGAs because of their larger area and capacitance [22]. Fig. 3 shows two possible implementations of a four-to-one multiplexer. Fig. 3(a) shows a "decoded" multiplexer, which requires four configuration SRAM cells if used in an FPGA routing switch. Input-to-output paths through this decoded multiplexer consist of a single NMOS transistor. Fig. 3(b) shows an "encoded" multiplexer that requires only two configuration SRAM cells, though has larger delay as its input-to-output paths consist of two transistors in series. In larger multiplexers, a combination of the designs shown in Fig. 3 is also possible, allowing one to tradeoff area for delay or *vice versa*.

When a logic 1 is passed through an NMOS-based multiplexer, a "weak 1" appears on the multiplexer's output ($\approx V_{\mathrm{DD}} - V_{\mathrm{TH}}$). The weak 1 has the potential to cause excessive leakage in the buffer attached to the multiplexer's output in Fig. 2. To deal with this, the buffer is normally implemented as a "level-restoring" buffer [23], [24]. In a level-restoring buffer, the buffer's input is pulled up to rail V_{DD} when logic 1 is passed through the multiplexer. It is important to recognize, however, that the multiplexers in modern FPGAs are large, and are deeper than one level of NMOS transistor. Only the output of the multiplexer is pulled up to rail V_{DD} by the level-restoring buffer; weak 1s will appear on internal multiplexer nodes.

SPICE simulations (at 110 °C) were performed to measure the leakage power of the multiplexers in Fig. 3. The proposed simulations were conducted using BSIM4 SPICE models for a 1.2-V 90-nm commercial CMOS process. Values were assigned to the select signals of the multiplexers so that input $i1$ was passed to the multiplexer output and then all 16 possible input vectors were simulated. Fig. 4 shows the multiplexer leakage power results. A vertical bar illustrates the leakage for each input vector. From Fig. 4, it is observed that leakage power in the multiplexers is highly dependent on input state. For the decoded multiplexer, the highest leakage occurs when logic 0 appears on input $i1$ (the input whose signal is passed to the output) and logic 1 appears on all other inputs; the lowest leakage occurs when all inputs are logic 1. For the decoded multiplexer, there is a $13.7\times$ difference in leakage power between the highest and lowest leakage states; for the encoded multiplexer, the leakage power difference is $14.2\times$. In addition to the leakage for each input vector, Fig. 4 shows the average leakage power consumed when the output of the multiplexer is a logic 1 (solid horizontal line) and when the output of the multiplexer is a logic 0 (dashed horizontal line).

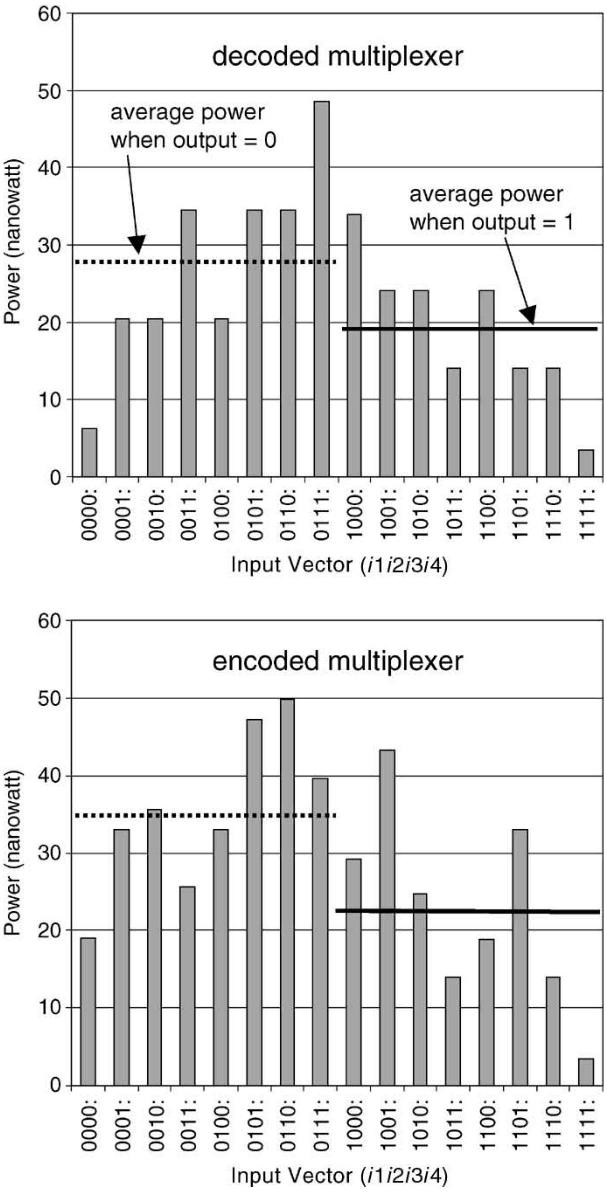


Fig. 4. Leakage power for multiplexers.

Observe that for both multiplexers in Fig. 3, the average leakage for passing a logic 1 to the multiplexer output is substantially smaller than the average leakage for passing logic 0. There are several reasons for this: First, as mentioned above,

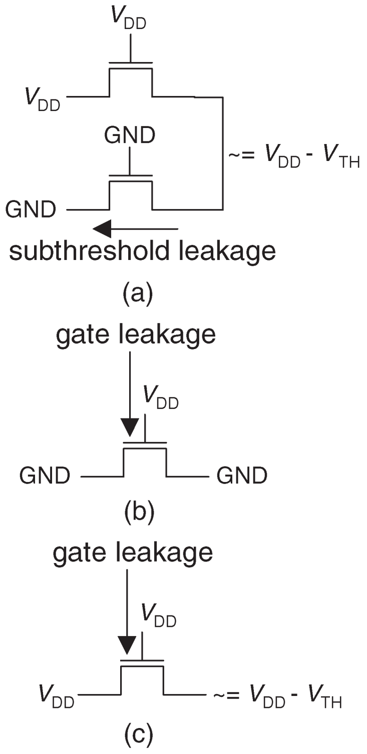
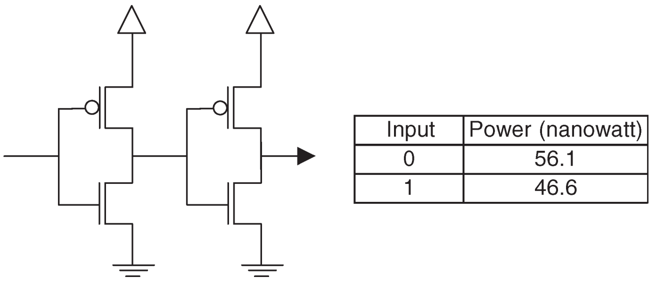


Fig. 5. Examples of transistor leakage states. (a) Reduced subthreshold leakage. (b) High gate leakage. (c) Low gate leakage.

when logic 1 (V_{DD}) is applied to the drain terminal of an ON NMOS device, a weak 1 ($\approx V_{\mathrm{DD}} - V_{\mathrm{TH}}$) appears at the source terminal. The weak 1 leads to reduced subthreshold leakage power in other multiplexer transistors that are OFF, versus when the potential difference across an OFF transistor is V_{DD} [see Fig. 5(a)]. This is due to the effect of drain-induced barrier lowering (DIBL) in short-channel transistors, which causes threshold voltage to decrease (subthreshold current to increase) when drain bias is increased [7].

In addition to affecting subthreshold leakage, another significant source of leakage power variation is due to a reduction in gate oxide leakage when the multiplexer is passing logic 1 to its output. Gate leakage is a considerable fraction of total leakage in the 90-nm technology. Gate leakage in an ON NMOS transistor depends significantly on the applied bias [25]. When an NMOS transistor is passing logic 0, the voltage difference between the gate and source is V_{DD} (that is, $V_{\mathrm{GS}} = V_{\mathrm{DD}}$) and the transistor is in the strong inversion state [see Fig. 5(b)]. Conversely, when the transistor is passing logic 1, the transistor is in the threshold state ($V_{\mathrm{GS}} \approx V_{\mathrm{TH}}$) [see Fig. 5(c)]. Gate oxide leakage in the threshold state is typically several orders of magnitude smaller than in the strong inversion state [25]. This property makes it preferable to pass logic 1 (versus logic 0) from the gate leakage perspective.

Another important circuit element in FPGAs is a buffer since they are present throughout the routing fabric and also within logic blocks. The two stage buffer shown in Fig. 6 was simulated and leakage power was measured in both input states. The buffer's transistors were sized to achieve equal rise and fall times, and the second stage was chosen to be three times larger than the first stage. Leakage power results for the buffer are shown on the right side of Fig. 6. Although the difference in power between the two input states is not as pronounced as the differences observed for the multiplexers, it is seen that about 20% more power is consumed when the buffer's input is a 0 versus when its input is a 1. The dependence of the buffer's leakage on input state is a result of NMOS and PMOS devices having different leakage characteristics (both gate oxide leakage and subthreshold characteristics) and the dependence of leakage on transistor size. For example, gate oxide leakage is considerably higher in NMOS versus PMOS transistors [26] and is also directly proportional to transistor size. Therefore, overall gate leakage is minimized when the large NMOS transistor in the buffer's second inverter stage is OFF, which occurs when the buffer's output state is logic 1.



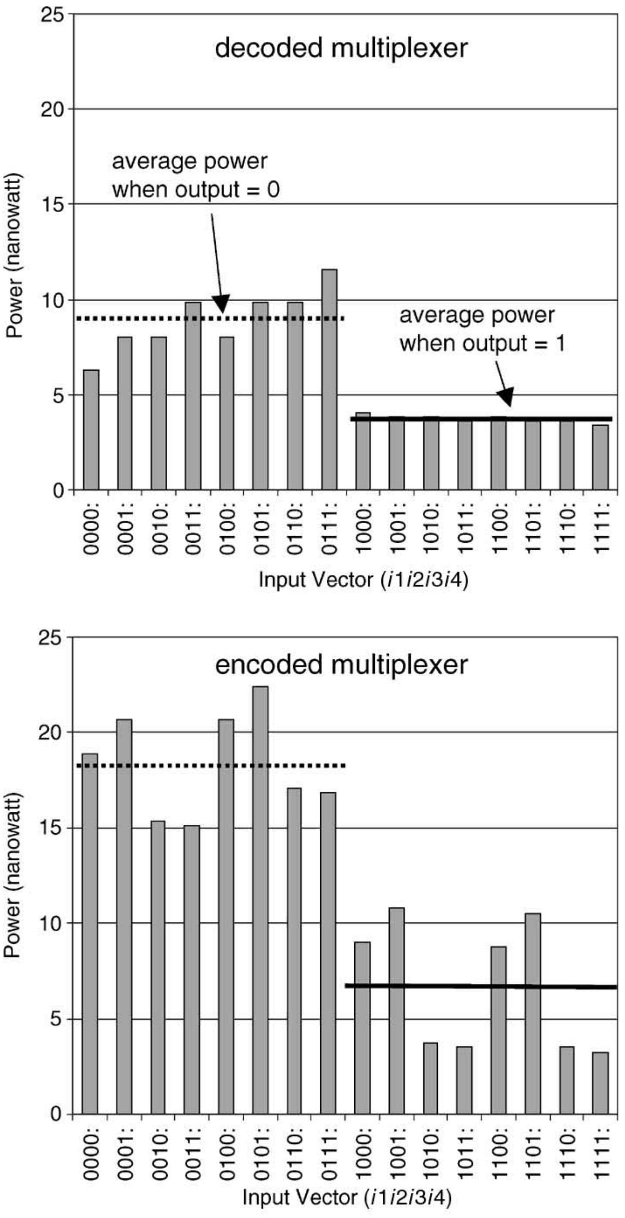
Input	Power (nanowatt)
0	56.1
1	46.6

Fig. 6. Buffer implementation and leakage power.

Subthreshold leakage increases exponentially with temperature, and consequently, leakage is primarily a problem at high temperature. This work concerns active leakage power in the operating (hot) part of the FPGA, and therefore, in this paper, the proposed leakage reduction techniques are evaluated at high temperature (110°). Unlike subthreshold leakage, gate oxide leakage is almost insensitive to temperature [27]. At low temperature, gate oxide leakage comprises a significantly larger fraction of total leakage. For completeness, the leakage characteristics of the basic FPGA hardware structures at low temperature (40°) are also examined. The results are shown in Fig. 7. Observe that similar leakage bias trends are apparent at low temperature; namely, less leakage is consumed when logic 1 is passed through the multiplexers and buffer versus when logic 0 is passed through these structures. In fact, in the multiplexers, the bias is more pronounced at low temperature. For example, in the decoded multiplexer, the average leakage power when the output is logic 0 is 140% higher than when the output is logic 1. At high temperature, the leakage difference between the two states is only 44%. In the buffer, the same bias is present at low temperature, but it is less pronounced; buffer leakage in the logic 0 state is about 7% higher than in the logic 1 state (versus 20% higher at high temperature).

A. Leakage Power Optimization in FPGAs

Several recent studies have considered techniques for leakage power reduction in FPGAs. Optimization of sleep mode leakage in FPGA logic blocks was addressed in [28], which proposed the creation of fine-grained "sleep regions," making it possible for a logic block's LUTs and flip-flops to be put to sleep independently. In [29], the authors propose a more



(a)

Input	Power (nanowatt)
0	19.3
1	18

(b)

Fig. 7. Low temperature leakage power results for multiplexers and buffer (40 °C). (a) Leakage power for multiplexers. (b) Leakage power for buffer.

coarse-grained sleep strategy in which entire regions of unused logic blocks may be placed into a low-leakage sleep state.

Active leakage in FPGAs has been addressed through the use of dual-$V_{\rm DD}$ techniques. Li *et al.* [30] proposed dual-$V_{\rm DD}$ FPGAs in which some logic blocks are fixed to operate at high $V_{\rm DD}$ (high speed), and some are fixed to operate at low $V_{\rm DD}$ (low power but slower). In [31], Li *et al.* extended their dual-$V_{\rm DD}$ FPGA work to allow blocks to operate at either high or low $V_{\rm DD}$. Gayasen *et al.* [32] apply configurable dual-$V_{\rm DD}$ concepts to both logic blocks and interconnect. The costs involved with deploying dual-$V_{\rm DD}$ techniques include the distribution of multiple power grids and the need to supply multiple voltages at the chip level.

Another approach to reducing leakage in FPGA interconnect is to borrow and apply well-known leakage reduction techniques from the ASIC domain [23]. In particular, Rahman and Polavarapuv [23] propose: 1) using a mix of low-$V_{\rm TH}$ and high-$V_{\rm TH}$ transistors in the multiplexers; 2) using body-bias techniques to raise the $V_{\rm TH}$ of multiplexer transistors that are OFF; 3) negatively biasing the gate terminals of OFF multiplexer transistors; and 4) introducing extra SRAM cells to allow for multiple OFF transistors on "unselected" multiplexer paths. A more recent paper by Ciccarelli *et al.* applies dual-$V_{\rm TH}$ techniques to the routing switch buffers in addition to the multiplexers [33]. Unlike the techniques noted here, the proposed leakage reduction methods impose no advanced process or biasing requirements and do not degrade area efficiency or performance.

IV. Active Leakage Power Optimization via Intelligent Polarity Selection

In Section III, it was observed that in a modern commercial CMOS process, the leakage power dissipated by elementary FPGA hardware structures, namely buffers and multiplexers, is typically smaller when the output and input of these structures is logic 1 versus logic 0. The first approach to active leakage power optimization approach works by choosing a polarity for each signal in an FPGA design, in a manner that enables signals to spend the majority of their time in the logic 1 state (the logic state associated with low leakage power). A fundamental property of a digital signal is its static probability, which is the fraction of time a signal spends in the logic 1 state. A signal with static probability greater than 0.5 spends more than 50% of its time at logic 1. This approach alters signal polarity to achieve high static probability for most signals. Unlike in ASICs, signal polarity inversion in FPGAs can be achieved without any area or delay penalty, by leveraging a unique property of the basic FPGA logic element.

Fig. 8 illustrates how a signal's polarity can be reversed in an FPGA. Fig. 8(a) shows a logic circuit having two AND gates and an exclusive-OR gate. Fig. 8(b) of the figure shows the circuit mapped into two-input LUTs. The memory contents are shown for each LUT and represent the truth table of the logic function implemented by the LUT's corresponding gate. In this example, the aim is to invert the signal int, so that its complemented rather than its true form is produced by a LUT and routed through the FPGA interconnection network. There are two steps to inverting a signal. First, the programming of the LUT producing the signal must be changed. Specifically, to invert the signal, all of the 0s in its driving LUT must be changed to 1s and the 1s must be changed to 0s. Second, the programming of LUTs that are fanouts of the inverted signal must be altered to "expect" the inverted form. This is achieved by permuting the bits in the SRAM cells of such "downstream" LUTs. Fig. 8(c) shows the circuit after the signal int is inverted. The permutation of bits in the inverted signal's fanout LUT is shown through shading: the contents of the top two SRAM cells in the downstream LUT are

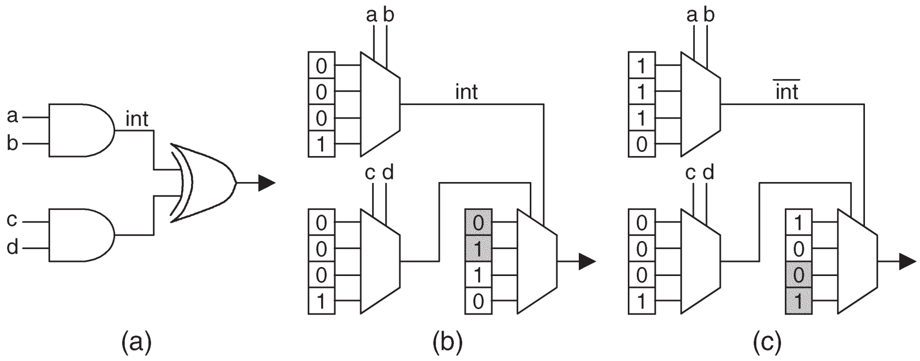


Fig. 8. LUT circuit implementation; illustration of signal inversion. (a) Original circuit. (b) 2-LUT implementation. (c) After signal inversion.

interchanged with the contents of the bottom two SRAM cells in the LUT. Through this method, signal inversion in FPGAs can be achieved by simply reprogramming LUTs.

The first approach to leakage power optimization is shown in Fig. 9. The input to the algorithm is an FPGA circuit as well as static probability values for each signal in the circuit. Iteration through the signals is carried out and those signals having static probability less than 0.5 are selected. Such signals spend most of their time in the logic 0 state and, thus, they are candidates for inversion. For each candidate signal, first it must be checked if it can be inverted (discussed below). If a candidate signal is invertible, it is inverted by reprogramming the FPGA configuration memory accordingly. After processing all signals, the output of the proposed algorithm is a modified design, having signals that spend the majority of their time in the logic state favorable to low leakage power.

A majority of the signals in FPGA designs are produced by LUTs and drive LUTs, and all such signals can be inverted using the approach shown in Fig. 8. In a commercial FPGA, however, in addition to LUTs, other types of hardware structures are usually present. Some signals driven by or driving non-LUT structures may also be invertible, since FPGA vendors frequently include extra circuitry for programmable inversion. However, some signals may not be invertible, such as those driving special control circuitry, entering the FPGA device from off-chip, or driving certain pins on non-LUT structures. As a concrete example, consider that the Xilinx Virtex-II PRO FPGA contains 18×18 block multipliers [19]. The inputs to the multipliers do not have programmable inversion. Therefore, any signal feeding a multiplier input should not be inverted by the proposed polarity selection approach, as doing so would be functionally incorrect (it would change the multiplication results). Similarly, Virtex-II contains large blocks of SRAM memory. Inverting a signal that drives a block RAM address input is not straightforward, as it implies a shuffling of memory contents, and block RAM memory contents is frequently preloaded during an FPGA's initial configuration phase. A two-pass approach would be needed to invert block RAM address signals: First, the polarity selection optimization would be executed, permitting block RAM address signal inversion. Then, the polarity selection results would be used to determine the appropriate rearrangement of block RAM memory contents. The memory contents would be shuffled appropriately, prior to FPGA configuration.

```
function OptimizeLeakage(design, signal static probabilities)
  for each signal $n$ in the design do
    if static_probability($n$) < 0.5 then
      if signal $n$ can be inverted then
        invert($n$)
        // FPGA is reprogrammed; $n$ replaced with $\overline{n}$
  return new design
```

Fig. 9. Leakage optimization algorithm.

Altering the polarity of a signal n with static probability $P(n)$, changes the signal's probability to $1 - P(n)$. Therefore, for signals having static probability close to 0.5, the benefits of inversion on leakage optimization are minimal, since the static probability of such signals remains close to 0.5 after inversion. Low leakage power can be achieved when signals have static probability close to 0 or 1. The question that arises then is whether the signals in real circuits exhibit this property. It is shown below that it is unlikely that the majority of signals in circuits will have probabilities close to 0.5, which bodes well for the proposed leakage optimization approach.

The average rate of logic transitions on a (nonclock) signal n, $F(n)$, can be expressed as a function of the signal's static probability [34], [37]

$$F(n) = 2P(n)[1 - P(n)] \tag{1}$$

where $F(n)$ is commonly referred to as signal n's normalized switching activity. $F(n)$ ranges from 0 to 0.5 and can be interpreted as the fraction of clock cycles in which signal n toggles. Note that (1) is a frequently used approximation that becomes exact in the absence of temporal correlations in signal n's switching activity (n's values in two consecutive clock cycles are independent). Solving (1) for $P(n)$ yields

$$P(n) = \frac{1 \pm \sqrt{1 - 2F(n)}}{2} \tag{2}$$

which is plotted in Fig. 10. Observe that $P(n)$ is 0.5 only when $F(n)$ is 0.5 and that for a fixed decrease in $F(n)$, there is a change in $P(n)$ towards either 0 or 1. From Fig. 10, it is inferred that if the switching activities of the majority of

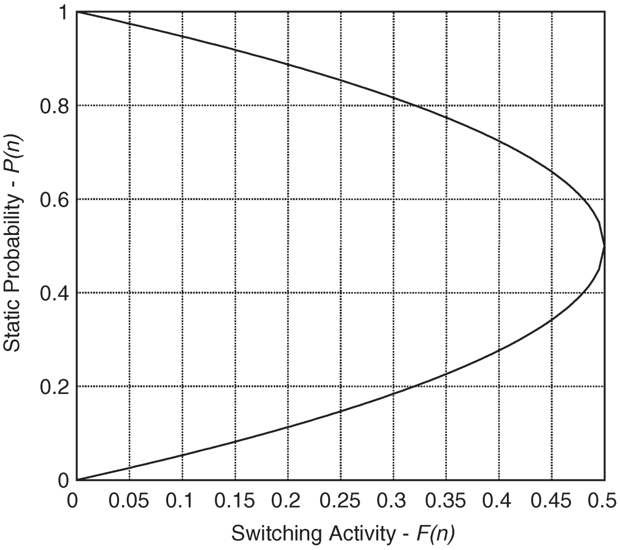


Fig. 10. Static probability versus switching activity.

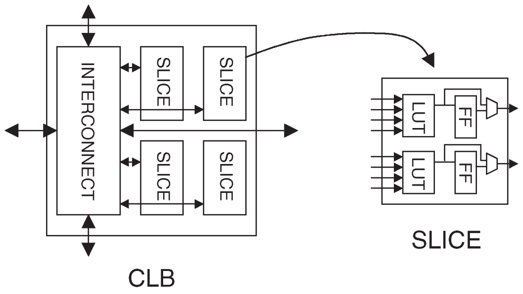


Fig. 11. CLB tile.

TABLE I
MAJOR CIRCUIT BLOCKS IN TARGET FPGA

Circuit Block	Details
IMUX	30-to-1 multiplexer, buffer
OMUX	24-to-1 multiplexer, buffer
DOUBLE	16-to-1 multiplexer, buffer
HEX	12-to-1 multiplexer, buffer
LONG	n-to-1 multiplexer, buffer (n device/orientation dependent)
LUT	16-to-1 multiplexer, in/out buffers
FLIP-FLOP	programmable set/reset

signals in circuits are not clustered close to 0.5, then the static probabilities of signals will also not be clustered close to 0.5. Switching activity in combinational circuits is well studied. Prior work by Nemani and Najm found that switching activities are generally not clustered around a single value and that, on average, activity decreases quadratically with combinational depth in circuits [35]. It can therefore be expected that there is a range of different static probabilities among the signals of a circuit and that "deeper" signals in circuits will have static probabilities approaching either 0 or 1. This analysis suggests that for many signals, changing polarity will have a significant impact on leakage power.

A. Experimental Study and Results

The effectiveness of the proposed leakage power reduction approach is evaluated by applying it to optimize active leakage in a state-of-the-art 1.2-V 90-nm Xilinx commercial FPGA. An analysis of the leakage in this FPGA has appeared recently in [36]. First, the proposed methodology is described and, subsequently, results are provided.

1) Methodology: The target FPGA is composed of an array of configurable logic block (CLB) tiles, I/Os, and other special-purpose blocks such as multipliers and block RAMs. Smaller versions of the FPGA contain only the CLB array and I/Os. An embedded version of the FPGA, containing the CLB array only, is also available for incorporation into custom ASICs. In this paper, the focus is on leakage optimization within the FPGA's CLB array, which represents the bulk of the FPGA's silicon area, especially in smaller devices and the embedded version. The non-CLB blocks (e.g., block RAMs) are not unique to FPGAs; leakage optimization in these blocks has been studied in other contexts.

A CLB tile contains both logic and routing resources. A simplified view of a CLB is shown in Fig. 11. The logic resources in a CLB consist of four logic subblocks, called SLICEs. Each SLICE contains two LUTs, two flip-flops, as well as arithmetic and other circuitries. The interconnect consists of variable length wire segments that connect to one another through programmable buffered switches similar to that shown in Fig. 2. Table I provides further detail on the major circuit blocks in a CLB tile. The input multiplexer (IMUX) selects and routes a signal to a SLICE input pin. The output multiplexer (OMUX) selects and routes a signal from a SLICE output pin to a neighboring logic block. Other interconnect blocks are named corresponding to their length: DOUBLE blocks drive wire segments that span two CLB tiles, HEX blocks drive wires that span six CLB tiles, and LONG resources span the entire width or height of the FPGA. Note that a single CLB tile contains multiple instances of each of the blocks listed in Table I.

Fig. 12 shows the proposed leakage optimization and analysis flow. As mentioned above, the input to the proposed algorithm is an FPGA circuit as well as the static probability value for each of the circuit's signals. In the experiments, ten large combinational MCNC benchmark circuits and six industrial circuits collected from Xilinx customers are used; the circuits are listed in Table II. The MCNC circuits are first synthesized from VHDL using Synplicity's Synplify Pro tool (ver. 7.0). Then, the circuits are technology mapped, placed, and routed in the target FPGA using the Xilinx software tools (ver. M6.2i). The industrial circuits are already available in technology-mapped form so only the placement and routing steps are required for these circuits. Column 4 of Table II lists the combinational depth of each benchmark circuit, as reported by the Xilinx static timing analysis tool. For most circuits, the longest path contains only LUTs; however, for three of the industrial circuits (marked with $*$ in Table II), the longest path contains both LUTs and carry logic.

In [6], preliminary results are presented for circuits that were not optimized for performance (speed). In practice, however, most FPGA users seek a high-performance design implementation. Consequently, the Xilinx place and route tools were

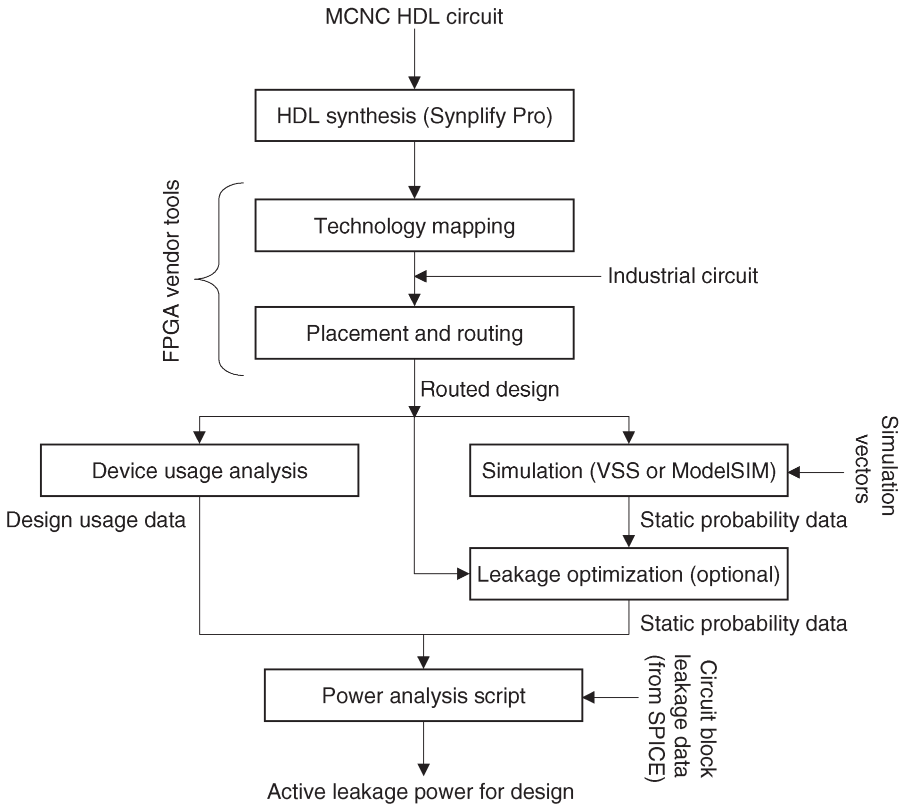


Fig. 12. Leakage analysis flow.

TABLE II
CHARACTERISTICS OF BENCHMARK CIRCUITS

Circuit	LUTs	FFs	Number of Logic Levels
alu4	500	0	8
apex4	1078	0	16
cps	524	0	11
dalu	323	0	8
ex1010	1112	0	32
ex5p	557	0	11
misex3	257	0	10
pdc	609	0	14
seq	1193	0	13
spla	229	0	7
industry1	1511	2128	16*
industry2	1654	1278	16*
industry3	2818	368	20
industry4	2942	1262	4
industry5	8676	5507	8
industry6	4895	318	24*

* Longest path has both LUTs and carry logic.

used to generate a performance-optimized layout for each benchmark design as follows: First, each design was placed and routed with an easy-to-meet timing (critical path delay) constraint. Then, based on the performance achieved, a more aggressive constraint was generated, and the place and route tools were reexecuted using the new constraint. The entire process was repeated until a constraint that could not be met by the layout tools was encountered. The proposed leakage reduction technique is evaluated for the layout solution corresponding to the most aggressive (but achievable) constraint observed throughout the entire iterative process.

To gather static probability data, the routed benchmark circuits were simulated using either the Synopsys VHDL System Simulator (VSS) or Mentor Graphics' ModelSIM. The simulators have built-in capabilities for capturing the fraction of time a signal spends at logic 1 (i.e., static probability). Since there is no access to simulation vectors for the circuits, the circuits were simulated using 10 000 randomly chosen input vectors.[1] In the vector set for each design, the probability of each primary input toggling between successive vectors was 50%. Note that, given the static probabilities of a circuit's primary input signals, the static probabilities of the circuit's internal signals can be computed using well-known probabilistic techniques [37]. Thus, simulation is not a requirement for the use of the proposed optimization approach, and it is expected that the approach could be incorporated into EDA tools that automatically perform the proposed leakage optimization.

SPICE simulations were performed for each type of circuit block in the FPGA's CLB tile and the leakage power consumed by each block for each of its possible input vectors was captured. Circuit regularity permitted the blocks with many inputs to be partitioned into subblocks, which were then simulated independently. To illustrate, consider a 16-to-1 multiplexer, constructed using four 4-to-1 multiplexer in the "first stage," and a fifth 4-to-1 multiplexer in the "second stage." One need not simulate all 2^{16} input combinations of the 16-to-1 multiplexer to gather accurate leakage data for each

[1] Clock and control inputs on circuits were presented with appropriate (nonrandom) signals.

of these input combinations. One can simulate the individual 4-to-1 multiplexers and combine their leakage results to produce leakage data for the large 16-to-1 multiplexer. This was the approach taken to gather leakage data for the commercial blocks with many inputs. Notably, the leakage characteristics of the commercial FPGA's circuit blocks were observed to be similar to those of the generic structures studied in Section III.

In the present experiments, the total active leakage power L_{active} was computed twice for each benchmark circuit, both with and without the proposed active leakage optimization. L_{active} is defined as the sum of the leakage power in each used circuit block. By analyzing the FPGA (routed) implementation solution for a benchmark, its circuit block usage can be determined, including the signals on the inputs and outputs of each used circuit block.

Computing the leakage for a used instance of a circuit block in a benchmark involves combining the power data extracted from the block's SPICE simulation with usage data from the benchmark circuit's FPGA implementation and static probability data from the benchmark's HDL simulation. It is worth reinforcing that the power data presented in Section III are not used; power data extracted from SPICE simulations of the commercial FPGA's circuit blocks are used instead.

Consider a used instance B of a circuit block in a benchmark and let $\vec{v}$ represent an input vector that may be presented to block B. Each bit b_i in vector $\vec{v}$ corresponds to an input i on block B. Let $S_{B,i}$ represent the signal on input i of block B in the benchmark's FPGA implementation. The static probability of signal $S_{B,i}$, $P(S_{B,i})$, is a known quantity, extracted from the benchmark's HDL simulation. If bit b_i is logic 1 in vector $\vec{v}$, then the static probability of bit b_i, $P_B(b_i)$, is defined to be equal to $P(S_{B,i})$. On the other hand, if b_i is logic 0 in $\vec{v}$, then $P_B(b_i)$ is defined to be $1 - P(S_{B,i})$. The probability of vector $\vec{v}$ appearing on the inputs of block B, $P_B(\vec{v})$, can be computed as the product of its constituent bit probabilities

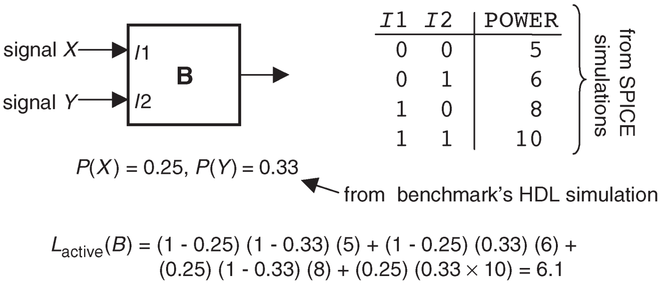
$$P_B(\vec{v}) = \prod_{b_i \epsilon \vec{v}} P_B(b_i). \tag{3}$$

The average active leakage power for a used circuit block B, $L_{\text{active}}(B)$, is computed as a weighted sum of the leakage power consumed by B for each of its input vectors

$$L_{\text{active}}(B) = \sum_{\vec{v} \epsilon V_B} P_B(\vec{v}) L_{\text{active}}(B_{\vec{v}}) \tag{4}$$

where V_B represents the set of all possible input vectors for circuit block B and $L_{\text{active}}(B_{\vec{v}})$ represents the leakage power consumed by block B when its input state is vector $\vec{v}$, obtained from SPICE simulations.

An example of the leakage power computation approach for a block with two inputs is shown in Fig. 13. In the example, the signal X on block input $I1$ has a static probability of 0.25 and the signal Y on input $I2$ has a static probability of 0.33. A table gives the power consumed by the block for each possible input vector. Consider, for example, the vector in which $I1 = 1$ and $I2 = 0$. The leakage power consumed by the block for this vector is 8. The probability of the vector appearing on the inputs of the block is: $P(X)[1 - P(Y)] = 0.25(1 - 0.33) = 0.1675$. Thus, the contribution of this vector to the block's active leakage is $0.1675(8) = 1.34$, which is the third term in the equation shown in Fig. 13.



I1	I2	POWER
0	0	5
0	1	6
1	0	8
1	1	10

$$L_{\text{active}}(B) = (1 - 0.25)\,(1 - 0.33)\,(5) + (1 - 0.25)\,(0.33)\,(6) + (0.25)\,(1 - 0.33)\,(8) + (0.25)\,(0.33 \times 10) = 6.1$$

Fig. 13. Example active leakage power computation.

Note that it is entirely possible that some inputs to a used circuit block may have no signal on them. For example, some inputs to a routing switch (see Fig. 2) may attach to conductors that are not used in the FPGA implementation of a benchmark circuit. In a commercial FPGA, unused routing conductors are not allowed to "float" to an indeterminate voltage state. In the target Xilinx FPGA, unused routing conductors are pulled up to logic 1. Pulling unused routing conductors into the low-leakage logic 1 state benefits overall leakage in the FPGA, since an FPGA implementation of a benchmark circuit requires only a fraction of the FPGA's routing resources. To demonstrate this, a detailed analysis of a portion of the routing in the industry4 benchmark was performed. In industry4's routing, it was found that there were 15 235 used DOUBLE resources and 5918 used HEX resources. On average, 10.4 (of 16) inputs on each DOUBLE resource in industry4's routing attached to unused routing conductors. The remaining 5.6 inputs (on average) attached to routing conductors with an active logic signal on them; that is, 5.6 inputs attached to routing conductors that were used in the routing of industry4. Likewise, the HEX resources in industry4 had 7.6 (of 12) inputs attached to unused routing conductors, on average, with the remaining 4.4 inputs attached to used routing conductors. In other words, considering all HEX and DOUBLE resources used in industry4, nearly 2/3 of the inputs to these resources attach to unused routing conductors and are therefore pulled to logic 1. This amplifies the need for the "prefer logic 1" approach taken in the polarity selection optimization.

Leakage power was not a primary design consideration in the target commercial FPGA. It is envisioned that the proposed active leakage reduction approach will be used in conjunction with a future, leakage-optimized FPGA architecture. Consequently, in the experiments, only the active leakage power is considered and the leakage in the unused part of the FPGA is ignored.[2] Unused leakage is viewed as a separate optimization problem that can be addressed by either powering down the unused circuit blocks or by applying the standby leakage optimization techniques mentioned in Section II. Further, the leakage in the FPGA's SRAM configuration cells is not in-

[2] The leakage power results for a given benchmark circuit include all leakage in the FPGA circuit blocks that are used in the benchmark's FPGA implementation, whether or not such used circuit blocks are idle.

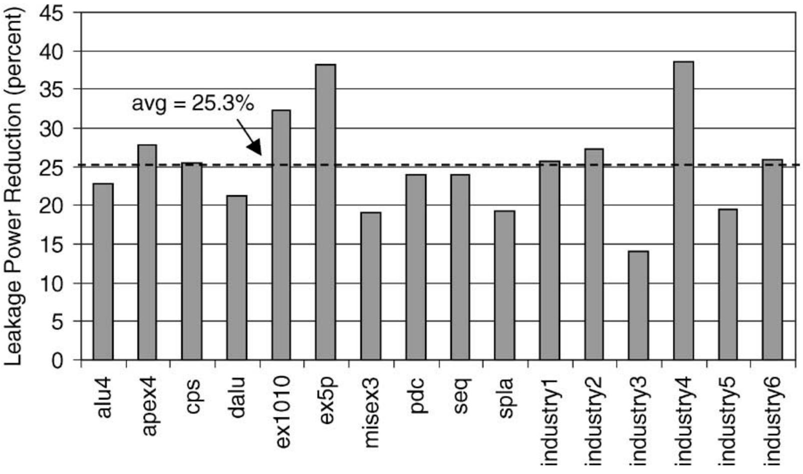


Fig. 14. Leakage power reduction results.

TABLE III
DETAILED ACTIVE LEAKAGE POWER RESULTS

	Unoptimized			Optimized		
Circuit	Interconnect (microwatt)	Other (microwatt)	Total (microwatt)	Interconnect (microwatt) (percent)	Other (microwatt) (percent)	Total (microwatt) (percent)
alu4	690	193	883	494 (28.4)	187 (3.1)	681 (22.8)
apex4	1625	415	2040	1060 (34.8)	410 (1.2)	1470 (27.9)
cps	698	183	881	476 (31.8)	180 (1.6)	656 (25.6)
dalu	465	126	591	341 (26.7)	125 (0.8)	466 (21.1)
ex1010	1747	427	2174	1045 (40.2)	424 (0.7)	1469 (32.4)
ex5p	829	210	1039	432 (47.8)	210 (0.0)	642 (38.2)
misex3	295	99	394	222 (24.8)	97 (2.0)	319 (19.1)
pdc	854	235	1089	598 (30.0)	230 (2.1)	828 (24.0)
seq	1895	453	2348	1335 (29.6)	451 (0.4)	1786 (23.9)
spla	315	89	404	239 (24.1)	87 (2.2)	326 (19.3)
industry1	3415	1557	4972	2164 (36.7)	1530 (1.7)	3693 (25.7)
industry2	2392	1340	3732	1408 (41.1)	1306 (2.6)	2713 (27.3)
industry3	4987	1573	6560	4086 (18.1)	1558 (0.9)	5644 (14.0)
industry4	6856	1927	8782	3531 (48.5)	1856 (3.7)	5386 (38.7)
industry5	14 696	6486	21 183	10 657 (27.5)	6412 (1.1)	17 069 (19.4)
industry6	6429	3524	9953	3919 (39.0)	3464 (1.7)	7382 (25.8)
Average				33.1%	1.6%	25.3%

cluded. Since the contents of such cells changes only during the initial FPGA configuration phase,[3] their speed performance is not critical. Thus, the SRAM configuration cells can be slowed down and their leakage reduced or eliminated using previously published low-leakage memory techniques (e.g., [38]) or by implementing memory cells with high-V_{TH} or long-channel transistors.

2) Results: The active leakage power consumed in the optimized circuits was compared with that consumed in the unoptimized circuits. Fig. 14 shows the percentage reduction in active leakage power for each circuit. The improvement ranges from 15% to 38%, with the average being 25%. The power benefits observed are quite substantial, considering that the proposed optimization has no impact on circuit area or delay, and requires no hardware changes.

Table III gives the detailed power results for each circuit. Columns 2–4 give power data for the unoptimized circuits. Columns 2 and 3 present the power dissipated in the interconnect and noninterconnect (labeled “other”) circuit blocks, respectively. Column 4 presents the total active leakage power for each circuit. Columns 5–7 present analogous data for the optimized circuits. In these columns, percentage improvement values (versus the unoptimized circuits) are shown in parentheses. From Table III, it is seen that the proposed optimization is more effective at reducing leakage in the interconnect versus the noninterconnect circuit blocks. The noninterconnect blocks include LUTs, flip-flops, and other circuitry. It was observed that flip-flop leakage power was only slightly dependent on whether the flip-flop was storing a logic 0 or a logic 1. Consequently, flip-flop leakage is not affected substantially by the proposed method. Similarly, it was found that the LUTs in the target FPGA contain additional input buffers and other circuitry that make their leakage less sensitive to their input state. In the unoptimized circuits, 24% of active leakage power is dissipated in the noninterconnect circuit blocks (on average) and 76% in the interconnect blocks. In the optimized circuits, 32% of leakage is attributable to noninterconnect blocks.

[3] FPGA device configuration is typically done only once: at power-up.

The results in Table III show that there is a wide variation in improvement across the circuits. This can be partially explained by considering the distribution of static probabilities among a circuit's signals. The proposed technique offers the greatest benefit in circuits having many signals with low static probability, and the least benefit in circuits having many signals with static probability ≥ 0.5 (these signals are already in the low-leakage state). Note that the static probability of a signal in a circuit is a function of both the simulation vector set as well as the circuit's logic functionality. From Table III, it is seen that the best results were achieved for the circuit industry4, with leakage reduced by 38%. Fig. 15(a) shows a histogram of static probabilities in this circuit, extracted from the ModelSIM simulation. The horizontal axis represents static probability; the vertical axis represents the fraction of the circuit's signals having static probability in a specific range. Observe that for this circuit, the majority of signals have low static probability, with more than 60% of signals having probability less than 0.1. It was verified that the skewed distribution was not a result of the simulation vector set failing to adequately exercise the circuit. In fact, more than 90% of the signals in circuit industry4 experienced toggling during its simulation. Fig. 15(b) shows the histogram for the circuit industry3, for which the worst results were observed. Here, many signals having static probability close to 0.5 are seen. For such signals, the static probability remains close to 0.5 after inversion, limiting the benefit of the leakage reduction approach. Further characterization and control of static probability in FPGA circuits is a direction for future work.

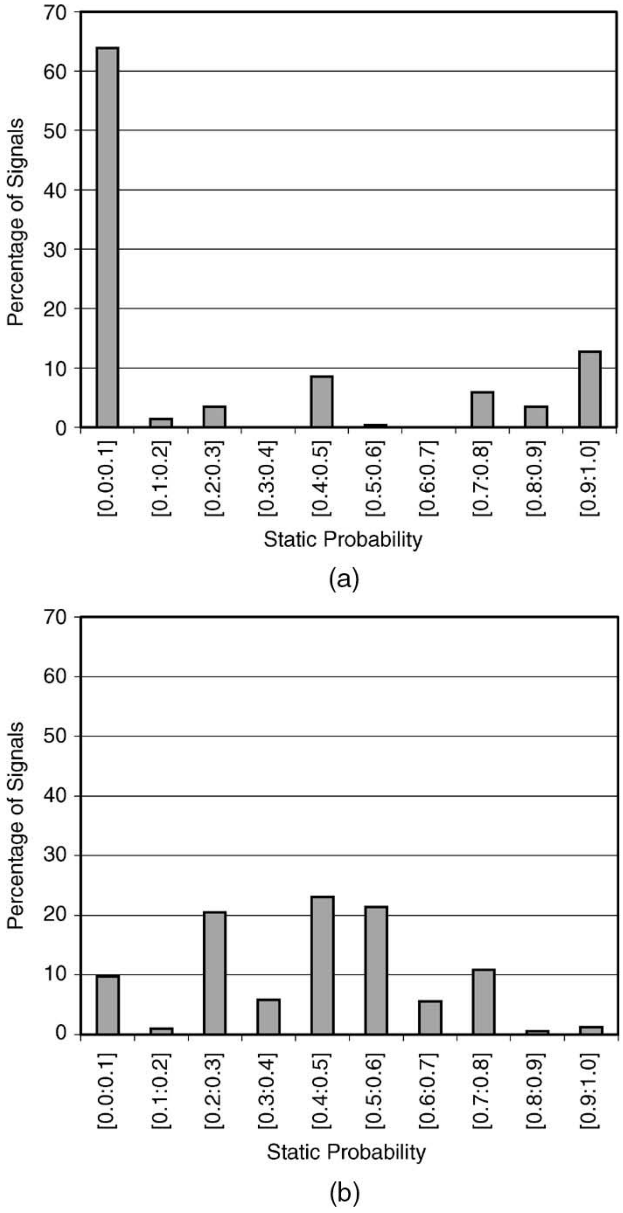


Fig. 15. Histograms of static probability. (a) Circuit industry4. (b) Circuit industry3.

V. Active Leakage Power Optimization via Leakage-Aware Routing

The second approach to active leakage optimization, which is referred to as leakage-aware FPGA routing, is now introduced. The idea is based on two observations.

1) Different routing switch types in an FPGA have different leakage power consumptions. For example, as illustrated in Table I, some switch types have wider input multiplexers or larger buffers than other switch types, leading to higher average leakage.
2) Between any two logic block pins in an FPGA, there exist a variety of different routing paths, comprised of different routing switch types. The routing step of the CAD flow is tasked with selecting a path between the driver and load pin on each of a design's signals.

FPGA routers employ a cost function and aim to find low-cost paths through the routing fabric from each signal's source pin to its load pin(s) [39], [40]. The cost of a complete routing path is defined as the sum of the costs of the path's constituent routing resources (switches). A cost function associates a particular cost value with each routing resource in the FPGA. Cost values can be chosen based on any number of criteria, for example, delay, scarcity, capacitance, or congestion. The idea behind leakage-aware routing is to select the cost for each routing resource in proportion to the resource's leakage power consumption, and then to use such costs during routing. The intent is to associate higher costs with more "leaky" switch types, making them less likely to be selected during routing, ultimately producing routing solutions having lower active leakage power consumptions.

The router in the Xilinx CAD flow classifies a design's driver/load connections as either critical or noncritical, based on their timing slack relative to the design's performance constraints. Critical and noncritical connections are then routed in timing-driven or cost-driven mode, respectively [41]. In timing-driven mode, detailed *RC* delay calculations are used during routing to minimize driver/load connection delay. In cost-driven mode, each routing resource is given a specific cost (as mentioned above) and the router attempts to minimize the total path cost for a given driver/load connection. The specific resource cost assignment used within the Xilinx router is proprietary; however, it reflects a combination of delay, wirelength, and scarcity. The original unmodified Xilinx router is referred to as the baseline router.

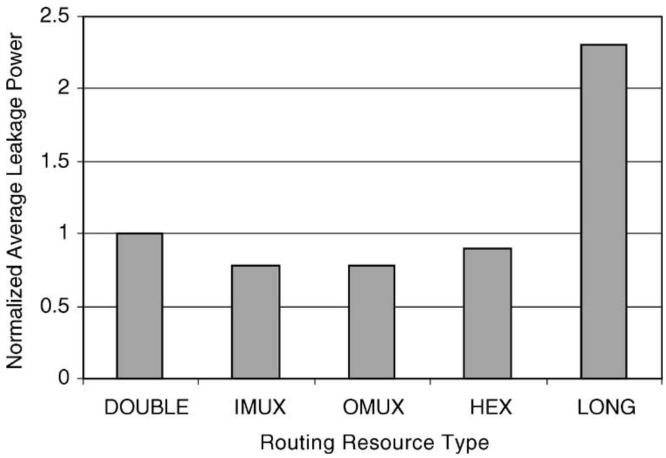


Fig. 16. Average leakage of routing resource types.

The proposed leakage-aware routing approach can be applied in tandem with the polarity selection optimization described in Section IV. Consequently, the optimized circuits (optimized through polarity selection) were used to derive a set of new leakage-aware routing resource costs. The leakage of each used routing resource in the optimized circuits was analyzed, and from this, the average leakage of each routing resource type was computed. The results are shown in Fig. 16, normalized to the leakage consumed by a DOUBLE resource. Observe that the average leakage of a HEX resource (which spans six CLB tiles) is slightly lower than that of a DOUBLE resource (which spans two CLB tiles), implying that on a leakage basis, using a HEX should be "cheaper" than using a DOUBLE.[4] This relative costing is counter to other traditional costing criteria, such as wirelength, in which the cost of a HEX would be set considerably higher than the cost of a DOUBLE.

The Xilinx router was modified and the costs that are used in cost-driven mode were altered, setting the cost of each routing resource in proportion to the average leakage of its routing resource type. Since the aim of this paper is to reduce leakage without compromising performance, the authors continue to allow the router to route timing-critical connections in timing-driven mode. Only noncritical connections are routed using the new leakage-derived costs. The modified router is referred to as the leakage-aware router.

A. Experimental Study and Results

Using the leakage-aware router, the 90-nm commercial FPGA described in Section IV-A is targeted with the same set of 16 benchmark circuit designs. The procedure described in Section IV-A1 that computes an aggressive but feasible timing constraint for each design was repeated. These constraints were compared with those produced using the baseline router. The results are shown in Table IV. Columns 2 and 3 show the critical path delay constraint for each circuit routed using the baseline and leakage-aware routers, respectively. Note that the same placer was used in both cases. The parentheses in column 3 show the percentage degradation in performance when the leakage-aware router is used versus the baseline router. Ten of the 16 circuits experienced a slight performance degradation, though no degradation was larger than 5%. The performance of the remaining six circuits actually improved slightly (negative values in the table). Changes to the router's cost function lead to variability in the routing solutions produced, resulting in performance improvements in some cases. On average, the degradation across all circuits was 0.3%, which is considered to be noise. It is concluded that any reductions in leakage power offered by leakage-aware routing do not come at the expense of speed performance. As with the polarity selection optimization presented in Section IV, leakage-aware routing is a "no cost" leakage reduction technique.

TABLE IV
EFFECT OF LEAKAGE-AWARE ROUTING ON CRITICAL PATH DELAY

Circuit	Baseline Routing Performance (nanosecond)	Leakage-Aware Routing Performance (nanosecond) (percent degradation)
alu4	11.05	11.12 (0.6)
apex4	14.31	14.79 (3.4)
cps	11.59	11.90 (2.7)
dalu	11.43	11.32 (-0.9)
ex1010	21.84	22.08 (1.1)
ex5p	12.92	12.32 (-4.6)
misex3	11.89	11.53 (-2.9)
pdc	13.93	13.63 (-2.0)
seq	13.30	13.55 (1.9)
spla	10.76	10.91 (1.4)
industry1	4.63	4.51 (-2.6)
industry2	10.83	10.78 (-0.5)
industry3	16.79	17.19 (2.3)
industry4	4.83	4.94 (2.1)
industry5	5.13	5.23 (2.0)
industry6	21.86	22.07 (1.0)
Average Degradation		0.3%

The polarity selection optimization was applied in conjunction with leakage-aware routing and the leakage in the resultant circuits was computed. Leakage was computed using the same approach described in Section IV-A1. Fig. 17 summarizes the results observed and illustrates the reduction in leakage in the optimized versus unoptimized circuits. Each bar in the figure represents the percentage reduction in leakage for a given circuit; the bars are partitioned to show the portion of the total reduction due to the polarity selection and leakage-aware routing optimizations, respectively. The average reduction across all circuits is 30.2%. Though the bulk of the power reduction is due to the polarity selection optimization, the benefits of leakage-aware routing are nonetheless substantial, especially in the industrial benchmark circuits.

Detailed leakage power results for each circuit are shown in Table V. Columns 2 and 3 give data for the interconnect and noninterconnect (labeled "other") circuit blocks, respectively. Column 4 gives the total active leakage power. The numbers in parentheses are percentage improvement values that show the

[4] A routing resource that drives a long wire segment may consume less leakage than some other resource that drives a short wire segment. This is possible since switch leakage does not depend on the metal segment length. Rather, leakage depends on the switch multiplexer size and structure, and transistor sizings in the multiplexer and buffer.

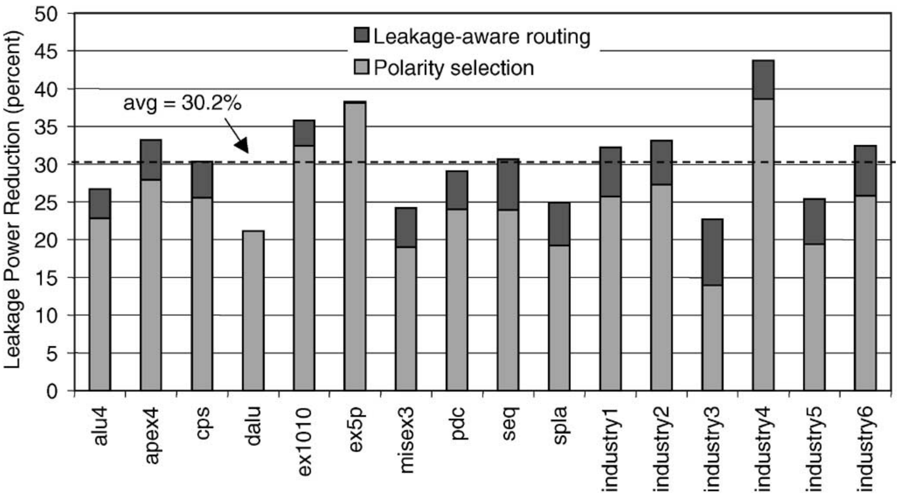


Fig. 17. Leakage power reduction results for combined polarity selection and leakage-aware routing.

TABLE V
DETAILED ACTIVE LEAKAGE POWER RESULTS FOR LEAKAGE-AWARE ROUTING COMBINED WITH POLARITY SELECTION

Circuit	Interconnect (microwatt) (percent)	Other (microwatt) (percent)	Total (microwatt) (percent)
alu4	460 (33.3)	187 (3.1)	647 (26.7)
apex4	953 (41.4)	410 (1.2)	1363 (33.2)
cps	434 (37.9)	180 (1.6)	614 (30.3)
dalu	341 (26.7)	125 (0.8)	466 (21.2)
ex1010	972 (44.4)	424 (0.7)	1396 (35.8)
ex5p	431 (48.0)	210 (0.0)	641 (38.3)
misex3	201 (31.6)	97 (2.0)	298 (24.2)
pdc	542 (36.5)	230 (2.1)	772 (29.1)
seq	1177 (37.9)	451 (0.4)	1628 (30.7)
spla	217 (31.3)	87 (2.2)	304 (24.9)
industry1	1840 (46.1)	1530 (1.7)	3369 (32.2)
industry2	1191 (50.2)	1306 (2.6)	2497 (33.1)
industry3	3515 (29.5)	1558 (0.9)	5073 (22.7)
industry4	3085 (55.0)	1856 (3.6)	4941 (43.7)
industry5	9404 (36.0)	6412 (1.1)	15816 (25.3)
industry6	3268 (49.2)	3464 (1.7)	6731 (32.4)
Average (MCNC)	36.9%	1.4%	29.4%
Average (industrial)	44.3%	2.0%	31.6%

reduction in leakage power relative to the unoptimized circuits (they compare the data in Table V with the data in columns 2–4 of Table III). Notice that, as expected, only leakage in the interconnect circuit blocks is affected by leakage-aware routing; leakage in the "other" circuit blocks is unchanged versus using the polarity selection optimization alone (see column 6 of Table III). For the MCNC circuits, the average reduction in total active leakage was 29.4%. In the industrial circuits, larger leakage reductions were observed, with the average reduction being 31.6%, due primarily to larger reductions in interconnect leakage for these circuits. The circuit industry4 experienced the largest leakage reduction of nearly 44%. In summary, the results show that the additional leakage power reductions offered by leakage-aware routing are considerable, especially given that the approach involves software changes only, and imposes no hardware, fabrication, or performance cost.

As mentioned previously, the cost of a HEX resource in the leakage-aware router is similar to that of a DOUBLE resource. Whereas, in the baseline router, the cost of HEX is higher than that of a DOUBLE. Certainly, leakage-aware routing leads to higher HEX utilization, and since the capacitance of a HEX is larger than that of a DOUBLE, it is conceivable that leakage-aware routing may increase dynamic power consumption. A future research direction is to investigate this possibility, and, if deemed a problem, to enhance leakage-aware routing to account for it, perhaps by taking signal switching activity into account when deciding how a signal should be routed. That being said, it is anticipated that the proposed techniques will be applied in a future low-leakage FPGA, perhaps implemented in 65- or 45-nm process technology. At such technology nodes, it is expected that leakage power, not dynamic power, will be the overriding power consideration.

VI. CONCLUSION

Trends in technology and voltage scaling have made leakage power a first class consideration in digital complementary metal oxide semiconductor (CMOS) design. In this paper, two "no cost" approaches to active leakage power reduction in field-programmable gate arrays (FPGAs) were presented. First, the leakage power characteristics of common FPGA hardware structures were studied. It was observed that the leakage comsumed by FGPA interconnect and logic circuitry depends strongly on the applied input state. A novel approach for leakage power reduction in which polarities are selected for logic signals to place hardware structures into low-leakage states as much as possible was proposed. The proposed technique is based on a unique property of FPGA logic elements [look-up tables (LUTs)] that permits either the true or complemented form of a signal to be generated, without any area or delay penalty. Experimental results for a 90-nm state-of-the-art commercial FPGA show that the proposed approach reduces active leakage by 25%, on average. Subsequently, the idea of leakage-aware routing was introduced, in which the cost function used during the routing step of the FPGA computer-aided design (CAD) flow is altered to consider the leakage power consumptions of routing resources. Leakage-aware routing incurs no significant performance penalty and offers additional leakage reductions. Combining the two techniques produces a total active leakage reduction of up to 44%, with the average reduction being 30%.

ACKNOWLEDGMENT

The authors would like to thank T. Tuan of Xilinx Research Laboratories for his helpful suggestions and his assistance with the HSPICE simulations. The authors also thank Xilinx for the infrastructure support. The comments provided by the anonymous reviewers are gratefully acknowledged.

REFERENCES

[1] J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold leakage modeling and reduction techniques," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, 2002, pp. 141–148.

[2] K. Poon, A. Yan, and S. J. E. Wilton, "A flexible power model for FPGAs," in *Proc. Int. Conf. Field Programmable Logic and Applications*, Montpellier, France, 2002, pp. 312–321.

[3] L. Shang, A. Kaviani, and K. Bathala, "Dynamic power consumption in the Virtex-II FPGA family," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 2002, pp. 157–164.

[4] V. George and J. Rabaey, *Low-Energy FPGAs: Architecture and Design*. Boston, MA: Kluwer, 2001.

[5] *Spartan-3 FPGA Data Sheet*. San Jose, CA: Xilinx, Inc, 2003.

[6] J. Anderson, F. Najm, and T. Tuan, "Active leakage power optimization for FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 2004, pp. 33–41.

[7] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proc. IEEE*, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[8] M. Anis, S. Areibi, M. Mahmoud, and M. Elmasry, "Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique," in *Proc. ACM/IEEE Design Automation Conf.*, New Orleans, LA, 2002, pp. 480–485.

[9] T. Sakurai, "Minimizing power across multiple technology and design levels," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, 2002, pp. 24–27.

[10] J. Halter and F. Najm, "A gate level leakage power reduction method for ultra-low-power CMOS circuits," in *Proc. IEEE Custom Integrated Circuits Conf.*, Santa Clara, CA, 1997, pp. 475–478.

[11] A. Abdollahi, F. Fallah, and M. Pedram, "Runtime mechanisms for leakage current reduction in CMOS VLSI circuits," in *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design*, Monterey, CA, 2002, pp. 213–218.

[12] C. Kim and K. Roy, "Dynamic Vth scaling scheme for active leakage power reduction," in *Proc. IEEE Design, Automation and Test in Europe Conf.*, Paris, France, 2002, pp. 163–167.

[13] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, 2002, pp. 721–725.

[14] A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar, and V. De, "Effectiveness of reverse body bias for leakage control in scaled dual Vt CMOS ICs," in *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design*, Huntington Beach, CA, 2001, pp. 207–211.

[15] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda, and D. Blaauw, "Duet: An accurate leakage estimation and optimization tool for dual-Vt circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 2, pp. 79–90, Apr. 2002.

[16] K. Usami, N. Kawabe, M. Koizumi, K. Seta, and T. Furusawa, "Automated selective multi-threshold design for ultra-low standby applications," in *Proc. ACM/IEEE Int. Conf. Low Power Electronics and Design*, Monterey, CA, 2002, pp. 202–206.

[17] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan, "Scaling of stack effect and its application for leakage reduction," in *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design*, Huntington Beach, CA, 2001, pp. 195–200.

[18] M. Johnson, D. Somasekhar, L.-Y. Choiu, and K. Roy, "Leakage control with efficient use of transistor stacks in single threshold CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 1, pp. 1–5, Feb. 2002.

[19] *Virtex II PRO FPGA Data Sheet*. San Jose, CA: Xilinx, Inc., 2003.

[20] G. Lemieux and D. Lewis, "Circuit design of routing switches," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 2002, pp. 19–28.

[21] D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cliff, and J. Rose, "The Stratix routing and logic architecture," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 2003, pp. 12–20.

[22] G. Lemieux, "Design of interconnection networks for programmable logic devices," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 2003.

[23] A. Rahman and V. Polavarapuv, "Evaluation of low-leakage design techniques for field-programmable gate arrays," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 2004, pp. 23–30.

[24] J. Anderson and F. Najm, "A novel low-power FPGA routing switch," in *Proc. IEEE Custom Integrated Circuits Conf.*, Orlando, FL, 2004, pp. 719–722.

[25] R. Guindi and F. Najm, "Design techniques for gate-leakage reduction in CMOS circuits," in *Proc. IEEE Int. Symp. Quality Electronic Design*, San Jose, CA, 2003, pp. 61–65.

[26] B. Yu, H. Wang, C. Riccobene, Q. Xiang, and M.-R. Lin, "Limits of gate-oxide scaling in nano-transistors," in *Proc. IEEE Symp. VLSI Technology*, Honolulu, HI, 2000, pp. 90–91.

[27] A. Agarwal, C. Kim, S. Mukhopadhyay, and K. Roy, "Leakage in nano-scale technologies: Mechanisms, impact and design considerations," in *Proc. ACM/IEEE Design Automation Conf.*, San Diego, CA, 2004, pp. 6–11.

[28] B. Calhoun, F. Honore, and A. Chandrakasan, "Design methodology for fine-grained leakage control in MTCMOS," in *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design*, Seoul, South Korea, 2003, pp. 104–109.

[29] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan, "Reducing leakage energy in FPGAs using region-constrained placement," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 2004, pp. 51–58.

[30] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-Vdd/dual-Vt fabrics," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 2004, pp. 42–50.

[31] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-Vdd," in *Proc. ACM/IEEE Design Automation Conf.*, San Diego, CA, 2004, pp. 735–740.

[32] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan, "A dual-Vdd low power FPGA architecture," in *Proc. Int. Conf. Field Programmable Logic and Applications*, Antwerp, Belgium, 2004, pp. 145–157.

[33] L. Ciccarelli, A. Lodi, and R. Canegallo, "Low leakage circuit design for FPGAs," in *Proc. IEEE Custom Integrated Circuits Conf.*, Orlando, FL, 2004, pp. 715–718.

[34] M. Cirit, "Estimating dynamic power consumption of CMOS circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Santa Clara, CA, 1987, pp. 534–537.

[35] M. Nemani and F. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 6, pp. 697–713, Jun. 1999.

[36] T. Tuan and B. Lai, "Leakage power analysis of a 90 nm FPGA," in *Proc. IEEE Custom Integrated Circuits Conf.*, San Jose, CA, 2003, pp. 57–60.

[37] G. Yeap, *Practical Low Power Digital VLSI Design*. Boston, MA: Kluwer, 1998.

[38] C. Kim, J.-J. Kim, S. Mukhopadhyay, and K. Roy, "A forward body-biased low-leakage SRAM cache: Device and architecture considerations," in *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design*, Seoul, South Korea, 2003, pp. 6–9.

[39] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 1995, pp. 111–117.

[40] J. Swartz, V. Betz, and J. Rose, "A fast routability-driven router for FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, 1998, pp. 140–149.

[41] J. Anderson, S. Nag, K. Chaudhary, S. Kalman, C. Madabhushi, and P. Cheng, "Run-time-conscious automatic timing-driven FPGA layout synthesis," in *Proc. Int. Conf. Field Programmable Logic and Applications*, Antwerp, Belgium, 2004, pp. 168–178.

Jason H. Anderson (S'97) received the B.Sc. degree in computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 1995, and the M.A.Sc. and Ph.D. degrees in electrical and computer engineering (ECE) from the University of Toronto, Toronto, ON, Canada, in 1997 and 2005, respectively.

In 1997, he joined Xilinx, Inc. in San Jose, CA, as a member of the implementation tools group, where he developed placement and routing tools for Xilinx field-programmable gate arrays (FPGAs). Presently, he is a Senior Staff Engineer at the Xilinx Toronto Development Centre and an Adjunct Professor of ECE at the University of Toronto. He is an inventor of more than a dozen issued and pending U.S. patents. His research interests include all aspects of computer-aided design (CAD) and architecture for FPGAs.

Dr. Anderson received the Ross Freeman Award for Technical Innovation, the highest innovation award given by Xilinx, for his contributions to the Xilinx placer technology in 2000. He was also awarded the Natural Sciences and Engineering Research Council (NSERC) of Canada Postgraduate Scholarship in 2001, and the Ontario Graduate Scholarship in 2003 and 2004.

Farid N. Najm (S'85–M'89–SM'96–F'03) received the B.E. degree in electrical engineering from the American University of Beirut (AUB), Beirut, Lebanon, in 1983, and the M.S. and Ph.D. degrees in electrical and computer engineering (ECE) from the University of Illinois at Urbana-Champaign (UIUC) in 1986 and 1989, respectively.

He worked with Texas Instruments in Dallas, TX, 1989–1992, then joined the ECE Department at UIUC as an Assistant Professor, becoming Associate Professor in 1997. In 1999, he joined the ECE Department at the University of Toronto, Toronto, ON, Canada, where he is now Professor and Vice-Chair of ECE. He has coauthored the text Failure Mechanisms in Semiconductor Devices, 2nd ed. [New York: Wiley, 1997]. His research is on computer-aided design (CAD) for integrated circuits, with an emphasis on circuit level issues related to power dissipation, timing, and reliability.

Dr. Najm received the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS Best Paper Award in 1992, the NSF Research Initiation Award in 1993, and the NSF CAREER Award in 1996. He was an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS from 1997 to 2002. He served as General Chairman for the 1999 International Symposium on Low-Power Electronics and Design (ISLPED-99), and as Technical Program Co-Chairman for ISLPED-98. He has also served on the technical committees of ICCAD, DAC, CICC, ISQED, and ISLPED. Currently, he is an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.