# Active Leakage Power Optimization for FPGAs

Jason H. Anderson[†,‡], Farid N. Najm[†], and Tim Tuan[⋆]

[†]ECE Department, University of Toronto, Toronto, ON, Canada
[‡]Xilinx Toronto Development Centre, Toronto, ON, Canada
[⋆]Xilinx Research Labs, San Jose, CA, USA
janders@eecg.toronto.edu, f.najm@utoronto.ca, tuan@xilinx.com

## ABSTRACT

We consider active leakage power dissipation in FPGAs and present a "no cost" approach for active leakage reduction. It is well-known that the leakage power consumed by a digital CMOS circuit depends strongly on the state of its inputs. Our leakage reduction technique leverages a fundamental property of basic FPGA logic elements (look-up-tables) that allows a logic signal in an FPGA design to be interchanged with its complemented form without any area or delay penalty. We apply this property to select polarities for logic signals so that FPGA hardware structures spend the majority of time in low leakage states. In an experimental study, we optimize active leakage power in circuits mapped into a state-of-the-art 90$n$m commercial FPGA. Results show that the proposed approach reduces active leakage by 25%, on average.

## Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Design Aids

## General Terms

Design, Algorithms

## Keywords

Field-programmable gate arrays, FPGAs, power, leakage, optimization, low-power design

## 1. INTRODUCTION

Trends in technology scaling make leakage power an increasingly dominant component of total power dissipation. Leakage power has two main forms in modern IC processes: *subthreshold* leakage and *gate* leakage. Subthreshold leakage power is due to a non-zero current between the source and drain terminals of an OFF 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 CMOS design and a recent work suggests that it may constitute over 40% of total power at the 70$n$m technology node [8].

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 [18, 21, 4] and have shown that the power consumed by the largest FPGA devices is increasing, with such devices now consuming Watts of power [21]. 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$\mu$m or larger, making them somewhat out of step with today's state-of-the-art FPGAs, which are fabricated in sub-100$n$m technology [25]. The programmability of FPGAs implies that more transistors are needed to implement a given logic circuit, in comparison with custom 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.


*FPGA'04*, February 22-24, 2004, Monterey, California, USA.

In this paper, we focus on optimizing active leakage power dissipation in FPGAs. We illustrate how the leakage power of typical FPGA hardware structures depends strongly on the state of their inputs. We then present 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 requires no modifications to the underlying FPGA hardware. We intelligently choose polarities for signals in a way that places hardware structures in their low leakage states. To our knowledge, no prior work has considered active leakage power optimization in FPGA technology. The paper is organized as follows: In Section 2, we discuss related work on leakage optimization. Section 3 describes typical FPGA hardware structures and studies their leakage power characteristics. Our approach to leakage reduction is described in Section 4. In Section 5, we validate our approach experimentally by applying it to optimize leakage in a 90$n$m commercial FPGA. Conclusions are offered in Section 6.

## 2. LEAKAGE POWER OPTIMIZATION

In this section, we summarize a few of the important leakage reduction techniques used in ASICs and microprocessors. A more detailed overview can be found in [19].

Several recent works have considered standby leakage power optimization. In [2, 20], 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 that is related to our own is based on the fact that a circuit's leakage depends on its input state. In [6, 1], a specific input vector is identified that minimizes leakage power in a circuit; 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 [1].

Active leakage reduction has also been addressed in the literature. One approach performs dynamic $V_{TH}$ adjustment based on system workload [11, 15]. 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 [9]. Other circuit-level techniques include the use of multi-threshold (MT) CMOS [22, 24], in which low-$V_{TH}$ transistors are used in delay critical paths and high-$V_{TH}$ transistors are used in non-critical 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 [16, 7, 8]; 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 [19]. 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.

## 3. FPGA HARDWARE STRUCTURES

Before describing our leakage reduction method, we review the circuit structures that are common to current FPGAs and study their leakage characteristics.

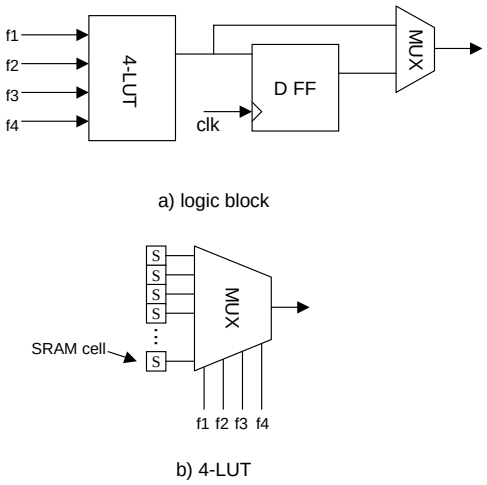



**Figure 1: FPGA logic block.**

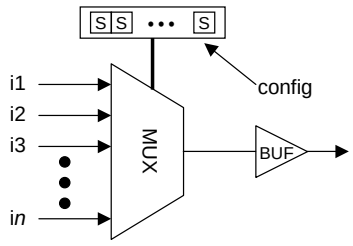



**Figure 2: Routing switch.**

FPGAs consist of an array of programmable logic blocks that are connected through a programmable interconnection network. Most commercial FPGAs use 4-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 4 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. 16 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$^{TM}$-II PRO FPGA contains 8 LUTs and 8 flip-flops [26].

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 [13, 14]. It consists of a multiplexer, a buffer and SRAM configuration bits. The multiplexer inputs (labeled i1-i$n$) 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

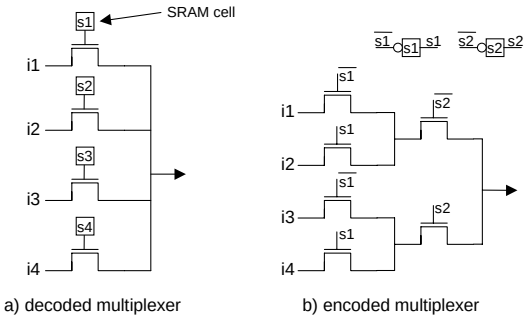



**Figure 3: Multiplexer implementations.**

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 [13], such as those shown in Fig. 3[1]. The figure shows two possible implementations of a 4-to-1 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 trade-off area for delay or vice-versa.

We performed SPICE simulations (at 110°C) to measure the leakage power of the multiplexers in Fig. 3. Our simulations were conducted using BSIM4 SPICE models for a 1.2V 90$n$m commercial CMOS process. We assigned values to the select signals of the multiplexers so that input $i1$ was passed to the multiplexer output and then simulated all 16 possible input vectors.

Fig. 4 shows the multiplexer leakage power results. A vertical bar illustrates the leakage for each input vector. From Fig. 4, we observe that leakage power in the multiplexers is highly dependent on input state. For both multiplexers, 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 13.7X difference in leakage power between the highest and lowest leakage states; for the encoded multiplexer, the leakage power difference is 14.2X. 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).

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, when logic 1 ($V_{DD}$) is applied to the drain terminal of an ON NMOS device, a "weak 1" ($\approx V_{DD} - V_{TH}$) 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 partly 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 [19].

[1]Note that full CMOS transmission gates are generally not used to implement multiplexers in FPGAs because of their larger area and capacitance [12].

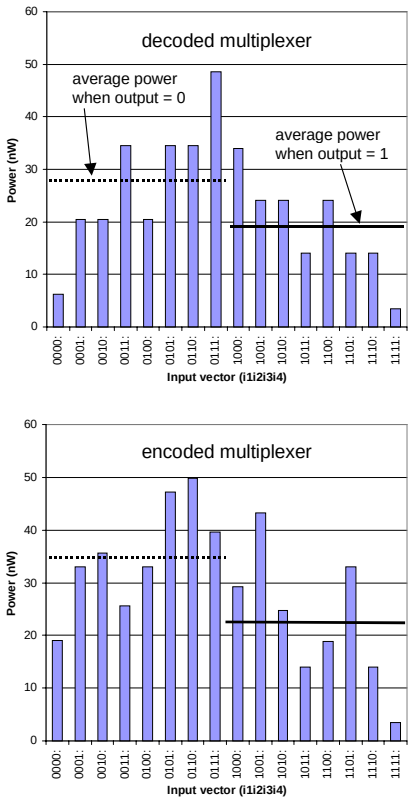



**Figure 4: Leakage power for multiplexers.**

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 90$n$m technology. Gate leakage in an ON NMOS transistor depends significantly on the applied bias [5]. When an NMOS transistor is passing logic 0, the voltage difference between the gate and source is $V_{DD}$ (that is, $V_{GS} = V_{DD}$) 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 ($V_{GS} \approx V_{TH}$) (see Fig. 5(c)). Gate oxide leakage in the threshold state is typically several orders of magnitude smaller than in the strong inversion state [5]. 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. We simulated the two stage buffer shown in Fig. 6 and measured its leakage power 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 3 times larger than the first stage. Leakage power results

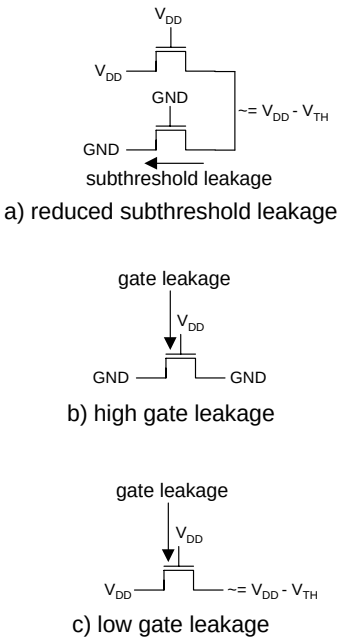



**Figure 5: Examples of transistor leakage states.**

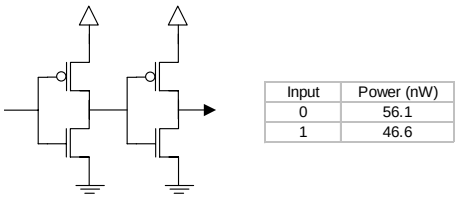



**Figure 6: Buffer implementation and leakage power.**

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, we see 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 considerably different leakage characteristics (both gate oxide leakage and subthreshold characteristics), and the dependence of leakage on transistor size.

# 4. ACTIVE LEAKAGE POWER OPTIMIZATION

In Section 3, we 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. Our 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. Our 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. 7 illustrates how a signal's polarity can be reversed in an FPGA. Part (a) of the figure shows a logic circuit having two AND gates and an exclusive-OR gate. Part (b) of the figure shows the circuit mapped into 2-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. Part (c) of Fig. 7 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 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 re-programming LUTs.

Our approach to leakage power optimization is shown in Fig. 8. The input to the algorithm is an FPGA circuit as well as static probability values for each signal in the circuit. We iterate through the signals and select those signals having static probability less than 0.5. Such signals spend most of their time in the logic 0 state and thus, they are candidates for inversion. For each candidate signal, we first must check if it can be inverted. The majority of signals in FPGA designs are produced by LUTs and drive LUTs and all such signals can be inverted using the approach shown in Fig. 7. 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 or entering the FPGA device from off-chip. If we find that a candidate signal is invertible, we invert it by reprogramming the FPGA configuration memory accordingly. After processing all signals, the output of our algorithm is a modified design, having signals that spend the majority of their time in the logic state favourable to low leakage power.

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. Below, we show that it is unlikely that the majority of signals in circuits will have probabilities close to 0.5, which bodes well for our leakage optimization approach.

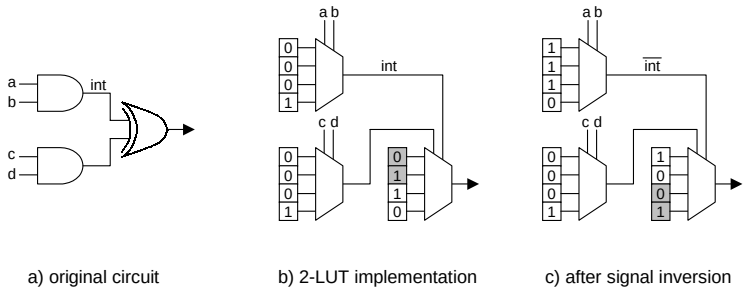



**Figure 7: LUT circuit implementation; illustration of signal inversion.**

```
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 re-programmed; n replaced with n̄
  return new design
```

**Figure 8: Leakage optimization algorithm.**

The average frequency of logic transitions on a (non-clock) signal $n$, $F(n)$, can be expressed as a function of the signal's static probability [3]:

$$F(n) = 2 \cdot P(n) \cdot [1 - P(n)] \quad (1)$$

where $F(n)$ is commonly referred to as signal $n$'s *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:

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

which is plotted in Fig. 9. 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. 9, we infer that if the switching activities of the majority of 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 [17]. We can therefore expect there to be a range of different static probabilities amongst 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.

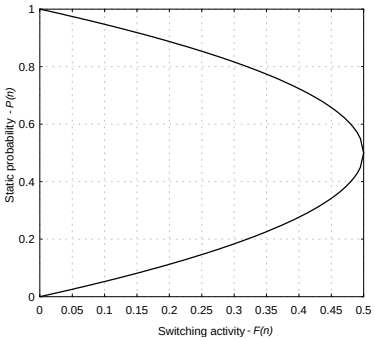



**Figure 9: Static probability versus switching activity.**

# 5. EXPERIMENTAL STUDY AND RESULTS

We evaluate the effectiveness of the proposed leakage power reduction approach by applying it to optimize active leakage in a state-of-the-art 1.2V 90$n$m commercial FPGA. An analysis of the leakage in this FPGA appeared recently in [23]. We first describe our methodology and subsequently we provide results.

## 5.1 Methodology

The target FPGA is composed of an array of configurable logic block (CLBs) 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, we focus 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. 10. The logic resources in a CLB consist of four logic sub-blocks, called SLICEs. Each SLICE contains two LUTs, two flip-flops as well as arithmetic and other circuitry. The interconnect consists of variable length wire segments that connect to one another through programmable, buffered switches similar

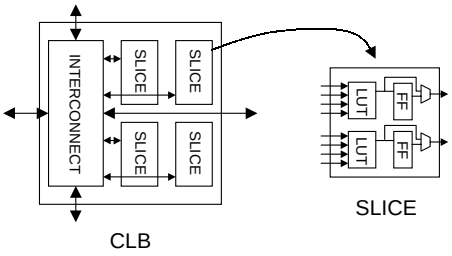



**Figure 10: Configurable logic block (CLB) tile.**

**Table 1: 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 | |

to that shown in Fig. 2. Table 1 provides further detail on the major circuit blocks in a CLB tile. The IMUX (input multiplexer) selects and routes a signal to a SLICE input pin. The OMUX (output multiplexer) 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 2 CLB tiles, HEX blocks drive wires that span 6 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 1.

Fig. 11 shows our leakage optimization and analysis flow. As mentioned in Section 4, the input to our algorithm is an FPGA circuit as well as the static probability value for each of the circuit's signals. In our experiments, we use 10 large combinational MCNC benchmark circuits and 6 industrial circuits collected from Xilinx customers; the circuits are listed in Table 2. 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 commercial vendor tools. The industrial circuits are already available in technology mapped form so only the placement and routing steps are required for these circuits.

To gather static probability data, the routed 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 we do not have access to simulation vectors for the circuits, the circuits are simulated using 10,000 randomly chosen input vectors[2]. In the vector set for each design, the probability of each each primary input toggling between successive vectors is 50%. Note that, given the static probabilities of a

[2]Clock and control inputs on circuits were presented with appropriate (non-random) signals.

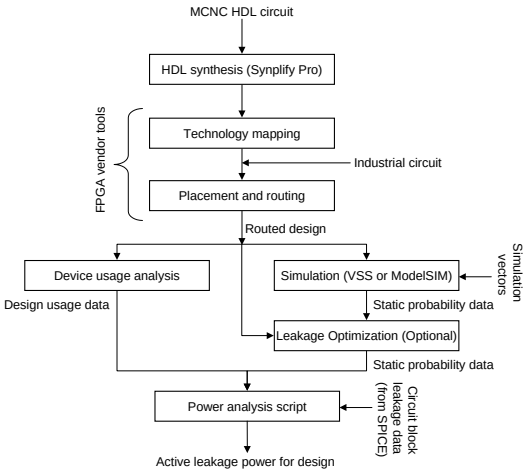



**Figure 11: Leakage analysis flow.**

**Table 2: Characteristics of benchmark circuits.**

| Circuit | LUTs | FFs |
|---|---|---|
| alu4 | 500 | 0 |
| apex4 | 1,078 | 0 |
| cps | 524 | 0 |
| dalu | 323 | 0 |
| ex1010 | 1,112 | 0 |
| ex5p | 557 | 0 |
| misex3 | 257 | 0 |
| pdc | 609 | 0 |
| seq | 1,193 | 0 |
| spla | 229 | 0 |
| industry1 | 1,511 | 2,128 |
| industry2 | 1,654 | 1,278 |
| industry3 | 2,818 | 368 |
| industry4 | 2,942 | 1,262 |
| industry5 | 8,676 | 5,507 |
| industry6 | 4,895 | 318 |

circuit's primary input signals, the static probabilities of the circuit's internal signals can be computed using well-known probabilistic techniques [27]. Thus, simulation is not a requirement for the use of our optimization approach and we expect that the approach could be incorporated into EDA tools that automatically perform the proposed leakage optimization.

In our experiments, the total active leakage power, $L_{active}$, is 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, we can determine its circuit block usage, including the signals on the inputs and outputs of each used circuit block.

We performed SPICE simulations for each type of circuit block in the FPGA's CLB tile and captured the leakage power consumed by each block for each of its possible input vectors[3]. Circuit regularity permitted the blocks with many inputs to be partitioned into sub-blocks which were then

[3]SPICE simulations conducted at 110°C using BSIM4 device models. High temperature simulations were used since this work concerns *active* leakage power in the operating (hot) part of the FPGA.

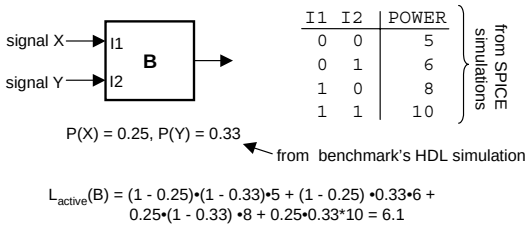



**Figure 12: Example active leakage power computation.**

simulated independently. We observed the leakage characteristics of the commercial FPGA's circuit blocks to be similar to those of the generic structures studied in Section 3. 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 we do not use the power data presented in Section 3 in our experimental study; rather, we use power data extracted from SPICE simulations of the commercial FPGA's circuit blocks.

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 we define the static probability of bit $b_i$, $P_B(b_i)$, 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 $1 - P(S_{B,i})$. We can compute the probability of vector $\vec{v}$ appearing on the inputs of block $B$, $P_B(\vec{v})$, as the product of its constituent bit probabilities:

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

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. Such unused inputs are pulled up to logic 1 in the target FPGA device.

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

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

where $V_B$ represents the set of all possible input vectors for circuit block $B$ and $L_{active}(B_{\vec{v}})$ 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 2 inputs is shown in Fig. 12.

Leakage power was not a primary design consideration in the target commercial FPGA. We envision that our active leakage reduction approach will be used in conjunction with a future, leakage-optimized FPGA architecture. Consequently, in our experiments, we consider only the active leakage power and ignore leakage in the unused part of the FPGA. We view unused leakage 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 2. Further, we do not include the leakage in the FPGA's SRAM configuration cells. Since the contents of such cells changes only during the initial FPGA configuration phase[4], 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., [10]) or by implementing memory cells with high-$V_{TH}$ or long channel transistors.

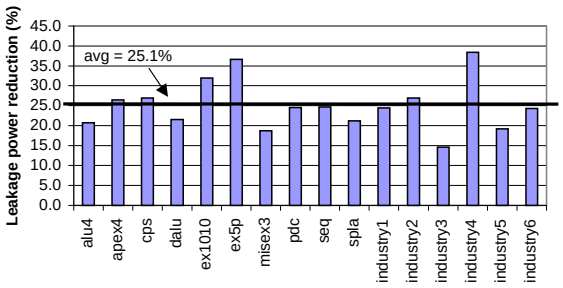



**Figure 13: Leakage power reduction results.**

## 5.2 Results

We begin by comparing the active leakage power consumed in the unoptimized circuits versus that consumed in the optimized circuits. Fig. 13 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 3 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 non-interconnect (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 3, we see that the proposed optimization is more effective at reducing leakage in the interconnect versus the non-interconnect circuit blocks. The non-interconnect blocks include LUTs, flip-flops and other circuitry. We observed that flip-flop leakage power was only slightly dependent on whether the flip-flop was storing a 0 or a 1. Consequently, flip-flop leakage is not affected substantially by the proposed method. Similarly, we 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, 30% of active leakage power is dissipated in the non-interconnect circuit blocks (on average) and 70% in

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

**Table 3: Detailed active leakage power results.**

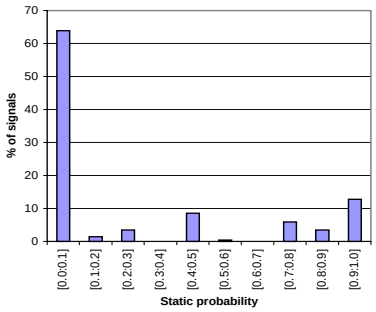
| Circuit | Unoptimized Interconnect ($\mu$W) | Other ($\mu$W) | Total ($\mu$W) | Optimized Interconnect ($\mu$W) (%) | Other ($\mu$W) (%) | Total ($\mu$W) (%) |
|---|---|---|---|---|---|---|
| alu4 | 547 | 193 | 740 | 400 (26.9) | 187 (3.1) | 587 (20.7) |
| apex4 | 1406 | 415 | 1821 | 929 (33.9) | 410 (1.2) | 1339 (26.5) |
| cps | 604 | 183 | 787 | 395 (34.6) | 180 (1.6) | 575 (26.9) |
| dalu | 372 | 126 | 498 | 266 (28.5) | 125 (0.8) | 391 (21.5) |
| ex1010 | 1606 | 427 | 2033 | 960 (40.2) | 424 (0.7) | 1384 (31.9) |
| ex5p | 709 | 210 | 919 | 372 (47.5) | 210 (0.0) | 582 (36.7) |
| misex3 | 254 | 99 | 353 | 190 (25.2) | 97 (2.0) | 287 (18.7) |
| pdc | 747 | 235 | 982 | 511 (31.6) | 230 (2.1) | 741 (24.5) |
| seq | 1660 | 453 | 2113 | 1151 (30.7) | 441 (2.6) | 1592 (24.7) |
| spla | 256 | 89 | 345 | 185 (27.7) | 87 (2.2) | 272 (21.2) |
| industry1 | 2997 | 1556 | 4553 | 1911 (36.2) | 1530 (1.7) | 3441 (24.4) |
| industry2 | 2030 | 1340 | 3370 | 1158 (43.0) | 1305 (2.6) | 2463 (26.9) |
| industry3 | 4394 | 1573 | 5967 | 3538 (19.5) | 1558 (1.0) | 5096 (14.6) |
| industry4 | 6176 | 1926 | 8102 | 3133 (49.3) | 1856 (3.6) | 4989 (38.4) |
| industry5 | 15425 | 6486 | 21911 | 11295 (26.8) | 6412 (1.1) | 17707 (19.2) |
| industry6 | 5553 | 3524 | 9077 | 3407 (38.6) | 3464 (1.7) | 6871 (24.3) |
| | | | | | **Average:** | **25.10%** |

the interconnect blocks. In the optimized circuits, 40% of leakage is attributable to non-interconnect blocks.

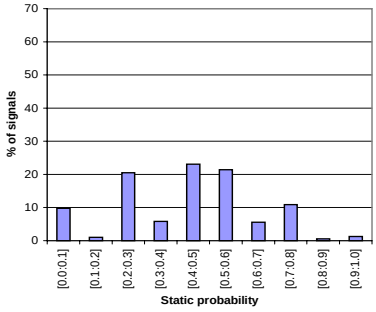
The results in Table 3 show there to be a wide variation in improvement across the circuits. This can be partially explained by considering the distribution of static probabilities amongst a circuit's signals. The proposed technique offers the greatest benefit in circuits having many signals with low static probability, and the least benefit in circuits having many signals with static probability $\geq 0.5$ (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 3, we see that the best results were achieved for the circuit *industry4*, with leakage reduced by 38%. Fig. 14(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 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. We 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. 14(b) shows the histogram for the circuit *industry3*, for which the worst results were observed. Here we see many signals having static probability close to 0.5. 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.

## 6. CONCLUSIONS

Trends in technology and voltage scaling have made leakage power a first class consideration in digital CMOS design. In this paper, we studied the leakage power characteristics of common FPGA hardware structures and found that their leakage depends strongly on the state of their inputs. We proposed 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. Our technique is based on a unique property of FPGA logic elements (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$n$m state-of-the-art commercial FPGA show the proposed approach reduces active leakage by 25%, on average.




a) circuit *industry4*




b) circuit *industry3*

**Figure 14: Histograms of static probability.**

## 7. REFERENCES


[1] A. Abdollahi, F. Fallah, and M. Pedram. Runtime mechanisms for leakage current reduction in CMOS VLSI circuits. In *International Symposium on Low-Power Electronics and Design*, pages 213–218, 2002.

[2] 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 *ACM/IEEE Design Automation Conference*, pages 480–485, 2002.

[3] M.A. Cirit. Estimating dynamic power consumption of CMOS circuits. In *IEEE International Conference on Computer-Aided Design*, pages 534–537, 1987.

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

[5] R.S. Guindi and F.N. Najm. Design techniques for gate-leakage reduction in CMOS circuits. In *IEEE International Symposium on Quality Electronic Design*, pages 61–65, 2003.

[6] J.P. Halter and F.N. Najm. A gate level leakage power reduction method for ultra-low-power CMOS circuits. In *IEEE Custom Integrated Circuits Conference*, pages 475–478, 1997.

[7] M.C. Johnson, D. Somasekhar, L.-Y. Choiu, and K. Roy. Leakage control with efficient use of transistor stacks in single threshold CMOS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(1):1–5, February 2002.

[8] J. Kao, S. Narendra, and A. Chandrakasan. Subthreshold leakage modeling and reduction techniques. In *IEEE International Conference on Computer-Aided Design*, pages 141–148, 2002.

[9] A. Keshavarzi, S. Ma, et. al. Effectiveness of reverse body bias for leakage control in scaled dual Vt CMOS ICs. In *ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 207–211, 2001.

[10] C.H. Kim, J.-J. Kim, S. Mukhopadhyay, and K. Roy. A forward body-biased low-leakage SRAM cache: Device and architecture considerations. In *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pages 6–9, 2003.

[11] C.H. Kim and K. Roy. Dynamic Vth scaling scheme for active leakage power reduction. In *IEEE Design, Automation and Test in Europe Conference*, pages 163–167, 2002.

[12] G. Lemieux. Design of interconnection networks for programmable logic devices. In *Ph.D. Thesis*. Department of Electrical and Computer Engineering, University of Toronto, 2003.

[13] G. Lemieux and D. Lewis. Circuit design of routing switches. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 19–28, 2002.

[14] D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, et. al. The Stratix routing and logic architecture. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 12–20, 2003.

[15] S.M. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *IEEE International Conference on Computer-Aided Design*, pages 721–725, 2002.

[16] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan. Scaling of stack effect and its application for leakage reduction. In *IEEE International Symposium on Low-Power Electronics and Design*, pages 195–200, 2001.

[17] M. Nemani and F. N. Najm. High-level area and power estimation for VLSI circuits. *IEEE Transactions on Computer Aided Design*, 18(6):697–713, June 1999.

[18] K.W. Poon, A. Yan, and S. J. E. Wilton. A flexible power model for FPGAs. In *International Conference on Field-Programmable Logic and Applications*, pages 312–321, 2002.

[19] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. In *Proceedings of the IEEE*, pages 305–327, February 2003.

[20] T. Sakurai. Minimizing power across multiple technology and design levels. In *IEEE International Conference on Computer-Aided Design*, pages 24–27, 2002.

[21] L. Shang, A. Kaviani, and K. Bathala. Dynamic power consumption the Virtex-II FPGA family. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 157–164, 2002.

[22] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda, and D. Blaauw. Duet: An accurate leakage estimation and optimization tool for dual-Vt circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(2):79–90, April 2002.

[23] T. Tuan and B. Lai. Leakage power analysis of a 90nm FPGA. In *IEEE Custom Integrated Circuits Conference*, pages 57–60, 2003.

[24] K. Usami, N. Kawabe, M. Koizumi, K. Seta, and T. Furusawa. Automated selective multi-threshold design for ultra-low standby applications. In *IEEE International Conference on Low-Power Electronics and Design*, pages 202–206, 2002.

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

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

[27] G. Yeap. *Practical Low Power Digital VLSI Design*. Kluwer Academic Publishers, Boston, MA, 1998.