A General Post-Processing Approach to Leakage Current Reduction in SRAM-based FPGAs

John Lach
ECE Department
University of Virginia
jlach@virginia.edu

Jason Brandon
ECE Department
University of Virginia
jmb2ac@virginia.edu

Kevin Skadron
CS Department
University of Virginia
skadron@cs.virginia.edu

Abstract

A negative effect of ever-shrinking supply and threshold voltages is the larger percentage of total power consumption that comes from leakage current. Several techniques have been developed to help reduce leakage in SRAM-based memory, in which the percent leakage power is especially acute. SRAM-based field programmable gate arrays (FPGAs) pose similar leakage problems, but their structure and function require different solutions. This paper introduces a low complexity post-processing approach to reducing FPGA leakage current by ground-gating off SRAM cells that are unused in a particular device configuration. The approach is general enough to apply to any device configuration, and results reveal that significant leakage current reduction can be achieved with no delay penalty and acceptable area overhead.

1. Introduction

As CMOS VLSI technology continues to scale, with ever-decreasing minimum feature sizes and increasing logic and memory densities, dynamic power consumption magnitude and density pose significant problems. Designers have historically addressed this issue by reducing the supply voltage. This in turn has led to a reduction in threshold voltages to maintain performance. However, due to the exponential dependence of subthreshold leakage current on threshold voltage, static power consumption has exploded and become a key area of concern and investigation. In fact, we are nearing a break-even point where it is no longer possible to reduce overall power consumption by scaling supply and threshold voltages due to the resulting increase in leakage current.

According to [7], leakage power will surpass active power and represent over 50% of total power in microprocessors at the 70nm technology node. To meet the ITRS roadmap restriction of static power comprising less than 10% of maximum power dissipation, the reduction in static power needed by circuit and architecture innovations reaches 98% by the end of the roadmap [14].

In order to keep power consumption in check, while still reaping the benefits of technology scaling, techniques for reducing static power consumption have recently been investigated. Most efforts have focused primarily on circuit-level techniques. In recent years, investigative work has begun to focus on static-power-hungry architecture-level structures such as SRAM-based cache memories. Because they are composed of a large number of leaky SRAM cells, these cache memories represent a prime target for static power reduction techniques.

SRAM-based FPGAs represent another opportunity for the development of leakage control techniques. Due to their reliance on SRAM cells for programmable logic and routing, FPGAs will require leakage control techniques to continue to benefit from technology scaling. With the increasing proliferation of FPGAs and the effort to utilize them in embedded low-power environments, the control of static power must be addressed in these devices.

In this paper, we present a methodology for leakage current reduction in SRAM-based FPGAs. For any given design mapped to an FPGA, a significant number of SRAM cells are unused, and leakage reduction techniques can be used to place these cells in a low leakage state. In this work, circuit gating (specifically ground-gating [1]) is used to turn off unused resources. The techniques presented are exclusively post-processing and do not restrict the CAD tool flow or require designer intervention. There is no effect on circuit performance, with the only penalty being the additional area required to implement ground-gating. Using a combined bottom-up circuit-level and top-down architecture-level approach, we have identified appropriate granularities for turning off SRAM cells in both FPGA logic and routing resources. Prior work has only identified the importance of the FPGA leakage problem [8,15]. To our knowledge, this paper represents the first effort to address the problem. While the post-processing techniques we present are straightforward, they provide significant leakage current reduction with minimal overhead.

Proceedings of the IEEE International Conference on Computer Design (ICCD'04)
1063-6404/04 $ 20.00 IEEE

In the next section, we present related work on FPGA leakage analysis and circuit-level techniques for leakage reduction. Section 3 introduces our general post-processing approach to leakage current reduction in FPGAs. Section 4 presents results for a theoretical 70nm FPGA employing our leakage reduction methodology. Section 5 gives our conclusions and direction of future work.

2. Background and related work

Two key areas of background and related work that we leverage concern leakage current analysis in FPGAs and circuit-level techniques to reduce leakage current.

2.1. Leakage current analysis in FPGAs

FPGAs have generally followed suit with the rest of the VLSI community and progressively scaled supply and threshold voltages with each technology generation to keep power consumption in check while maintaining performance. The resulting increase in leakage current is forcing designers to address this issue. Only recently have power evaluation tools and techniques for FPGAs, which give thorough consideration to the issue of leakage power, been developed.

[10] modified the VPR tool [2] to determine dynamic and static power consumption in a circuit placed-and-routed to a user-specified FPGA architecture. The model assumes that the gate-source voltages of inactive transistors are half of their threshold voltages and uses a formula to calculate leakage.

The work of [8] counters that this assumption is not usually valid. Their resulting method, fpgaEva-LP, models static leakage power using SPICE simulation of individual FPGA components with equal-probability input vectors. An important conclusion drawn by the authors is that up to 59% of the total power is attributed to leakage power.

In [15], leakage analysis is performed on a 90nm FPGA architecture. This work focuses on logic blocks as well as their access points to general-purpose routing. An analysis of the contribution of unused components to overall leakage power is also given. At 100% logic block utilization, unused resource leakage still accounts for 35% of total leakage. It is these unused resources that are the focus of the work presented here.

2.2. Circuit-level leakage current reduction

Both static and dynamic leakage reduction techniques in CMOS circuits have been studied. Static techniques utilize a multiple-threshold voltage (MTCMOS) fabrication process to selectively place faster low V_t transistors on the critical path, while employing less leaky high V_t transistors off the critical path. Such an approach is appropriate for ASICs, in which the function of the circuit is constant, but FPGAs require post-fabrication flexibility. Therefore, dynamic techniques are necessary. One such technique is the pre-characterization of a circuit's inputs in terms of minimal leakage power [16]. When the circuit is placed in standby mode, the input pattern with the lowest leakage can be applied.

A second post-fabrication leakage reduction technique is circuit gating. Circuit gating is the method of cutting off a circuit's path to V_{dd} or ground by insertion of 'sleep' transistors controlled by configuration signals or bits. When the sleep transistor is on, the circuit is in active mode. When it is off, the circuit is in a low-leakage mode. The gating approach may be single- or multiple-threshold-based. A higher V_t sleep transistor using MTCMOS provides increased leakage reduction, but the fabrication process is more expensive than single threshold [1]. In addition, single-V_t sleep transistors enable SRAM cells to maintain state [11]. In this paper, we explore both regular and high V_t transistors for ground-gating.

Ground-gating using NMOS sleep transistors has been successfully applied to SRAM-based cache memories. In [1], a single NMOS sleep transistor is inserted between the SRAM cells of each cache line and the ground plane. The technique utilizes the stacking effect of having two NMOS transistors connected in series [6]. The row decoder of the cache controls the sleep transistors, resulting in all lines being in sleep mode except when being accessed. Results show that compared to a conventional cache, their DRG-Cache leaks 32% less energy while the relative read time is only 2.8% slower.

Circuit gating has been employed in the FPGA domain as a proof of concept of an MTCMOS design methodology. [4] uses high-V_t local sleep transistors in FPGA logic blocks. These transistors give the capability to place individual regions of the logic block in sleep mode, including a group of four 4-input lookup tables (LUTs), a 4-bit adder, a 4-bit register, and the remaining control circuits. While this approach to ground-gating in an FPGA is similar to what we detail in this paper, the sleep transistor granularities considered in [4] were significantly more coarse, and only logic block leakage was considered. We explore a variety of granularities and show that the majority of SRAM cell leakage current savings can be derived from unused routing resources.

3. Methodology

A primary contribution of this paper is the derivation of appropriate granularities for turning off SRAM cells in FPGA logic and interconnect that are unused in a given configuration. While the ability to gate each SRAM cell

individually would ensure the maximum number of off cells regardless of the configuration, the overhead (in terms of not only area but also the additional leakage current introduced by the SRAM cells controlling the sleep transistors) would be unacceptable.

We therefore use a combined bottom-up/top-down approach for determining appropriate granularities. The bottom-up aspect considers circuit-level issues that affect the area and leakage overhead introduced at various granularities. The top-down portion explores architecture-level issues by considering cell groups left unused by real designs that are mapped onto the device. While bottom-up issues call for a coarse granularity (to minimize overhead), top-down pushes for a finer one (to maximize number of off cells). Finding the appropriate balance is the focus of this section.

3.1. Bottom-up analysis

In determining the appropriate granularity for sleep transistor insertion, several issues must be considered. It is necessary to determine the ability of a single sleep transistor to reduce leakage current for multiple SRAM cells and whether there is an effective limit to the number of cells that can be gated by a single transistor. The existence of a limit would dictate what granularities were feasible. The sizing ratio of the sleep transistor and whether regular or high V_t transistors should be used must also be explored. Finally, any potential delay introduced by the sleep transistor need be considered.

To answer these questions, we performed SPICE-level simulations of different sleep-transistor configurations applied to basic blocks of SRAM equivalent to those used in various FPGA resources. Berkeley BSIM device models from the Berkeley Predictive Technology Model (BPTM) 70nm process technology model [3,5] at V_{dd}=1.2V were used in these simulations, following the conservative approach discussed in [14].

SRAM leakage

Figure 1 gives the results for leakage reduction using regular V_t ($V_{t0\ (NMOS)}$ = 0.1902 V, $V_{t0\ (PMOS)}$ = -0.213 V) sleep transistors. We can see the general trend of decreased leakage reduction as the sleep transistor is sized up. We can also see that leakage savings continues to improve as the sleep transistor is shared amongst more SRAM cells. This is true up to the grouping of 64 cells, which is the upper limit appropriate for consideration given a target architecture of four 4-input LUTs.

Figure 2 shows the results of the same simulations performed using high V_t ($V_{t0\ (NMOS)}$ = 0.2402 V, $V_{t0\ (PMOS)}$ = -0.263 V) sleep transistors. It is clear that the high V_t results offer improved leakage savings across transistor widths and SRAM cell grouping sizes. This information can be used to make a design cost tradeoff. If the additional leakage savings is worth any additional process cost and the system does not require state-preserving gated cells, the high V_t approach is the obvious choice.

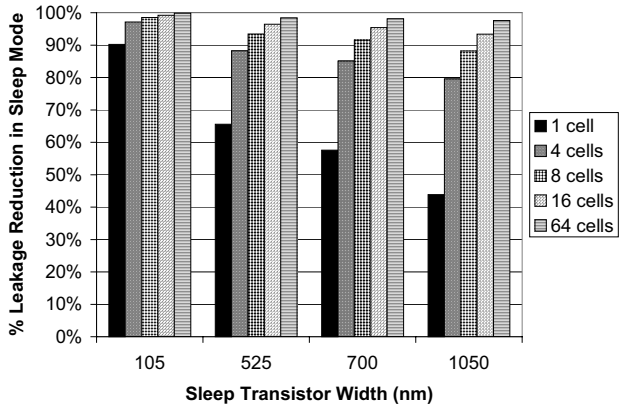

Figure 1. SRAM leakage reduction – regular V_t sleep transistors

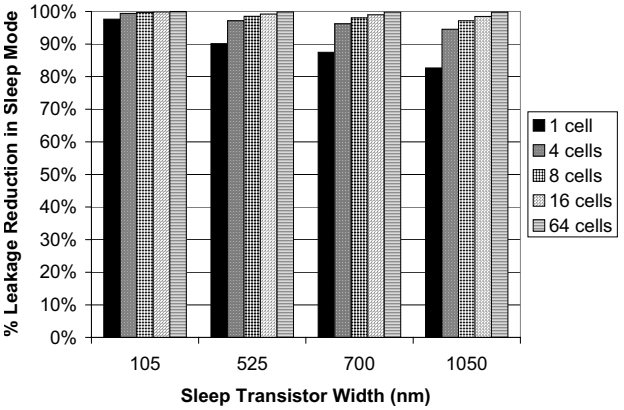

Figure 2. SRAM leakage reduction – high V_t sleep transistors

Sleep transistor delay

Potential circuit-level impact on delay of applying gated-ground to the SRAM cells used in FPGA logic and routing is an important consideration. When sleep transistors have been used to reduce leakage in caches, simulations revealed a delay penalty due to degraded bitline and sense amplifier delay [1].

However, circuit gating FPGA SRAM does not incur the same delay penalties as in caches due to the fact that reads from SRAM cells in FPGAs occur differently than in a cache, and the cells are not written to after configuration. FPGA resources using SRAM do not use bitlines and sense amplifiers to read the value of the cells. In an FPGA, only one of the two nodes is used to read the state of the cell. A connection is made directly from this node to the circuit it is used to control. SRAM cells thus continuously present their values (which remain constant

for a given FPGA configuration) directly to the circuitry they are used to control and therefore do not encounter the same delay issues as in a cache.

However, to verify that there is no delay penalty when gating SRAM-based resources in FPGAs we must consider the ability of gated SRAM cells in the active mode to provide sufficient drive strength to the FPGA circuits they are used to control. FPGA logic was evaluated by simulating LUTs and configurable logic blocks (CLBs) with various load capacitances at the outputs. These simulations showed no measurable increase in delay over non-gated circuits when sleep transistors are used to gate LUTs or CLBs. FPGA interconnect was also evaluated by examining the signal quality from the gated SRAM cells in active mode, as signal degradation on the gate input could cause a switch to misbehave or behave sub-optimally by not driving the transistor into saturation or cut-off. Simulations performed on active routing switches revealed no degradation in the quality of a logical one or zero signal observed. Finally, the effect of gating on inactive cells was also evaluated. Inactive ground-gated SRAM cells exhibited a node zero voltage rise, potentially causing switches gated by these cells to partially turn on, which could lead to unnecessary loading on interconnect segments. It is therefore necessary to restrict the cells that can be gated off to those where an active signal is not present anywhere on the affected segments. Future work will explore alternate gating techniques (e.g. V_{dd} gating) to address this issue.

3.2. Top-down analysis

Unused resources in FPGAs represent a particularly wasteful source of static power consumption. Given that FPGA family size increments are typically quite large and that the logic/interconnect ratio is set, there tends to be a significant number of unused logic and routing resources, regardless of the design configured onto the device. In the architectural analysis, we explore how many of the unused logic and routing resources can be gated off with various granularities of gating control.

Area overhead is calculated in terms of the equivalent number of minimum sized transistor areas introduced by the sleep transistors and the controlling SRAM cells at each granularity. This approach is process independent, and therefore the area overhead for each technique should stay relatively constant [12].

All of the techniques considered here are post-processing, in that they involve the evaluation of already placed-and-routed designs. There is therefore no impact on the design process and the resulting quality of implementation. While putting restrictions on the synthesis, mapping, and place-and-route tools may increase the percentage of SRAM cells that can be turned off with coarser sleep transistor granularities, this must be traded off against additional delay and area overhead. This tradeoff will be explored as part of future work.

Architecture model

The architecture we chose is modeled after the island-style SRAM-based Xilinx Spartan IIe [13]. While we model the Spartan IIe as closely as possible, the approach presented here is general for any SRAM-based island-style FPGA. Such FPGAs are composed of CLBs in a sea of programmable routing. CLBs in turn are composed of some number of LUTs, flip-flops, and additional architecture-specific logic and local interconnect. Following the Spartan IIe, each CLB in our architecture contains four 4-input LUTs and four flip-flops. CLBs are interconnected through routing channels running in rows and columns adjacent to each of their sides.

Multiple tracks exist in each channel and a CLB input or output pin could have a potential connection to any or all of these tracks. These multiple connections for a single pin could be controlled by either SRAM-gated pass-transistor switches or SRAM-controlled MUXes. The MUX approach is typically used for CLB input pins since these pins can only be driven by a single signal. However, output pins may fanout to more than one track and thus require individual connections to each accessible track in a channel. It was therefore assumed in this work that input pins employ the multiplexer style approach while output pins use individual pass transistor connections. In this paper, only unused CLB inputs are considered, as we found that the multiple-fanout nature of CLB outputs requires that constraints be placed on the router to enable coarse ground-gating of the SRAM cells controlling the CLB output pass transistors. This paper focuses only on post-place-and-route leakage reduction opportunities, but such tool alterations, and their resulting tradeoffs, will be considered as part of future work.

At the intersection of the routing channels, switch-boxes direct signals to adjacent channels. Several styles of switch-box exist, with each type having a different connection pattern to the tracks in adjacent channels. The "subset" style, in which a track in one channel can only connect to its corresponding track in each of the three other directions, is common. These connections are controlled by switch points, each containing six SRAM-gated pass transistors. Therefore, assuming symmetrical routing channels, each switch-box contains n switch-points and $6n$ SRAM-gated pass transistors, where n corresponds to the number of tracks in each channel routed through that switch-box. This subset style of switch-box is assumed in this work. Other switch-box

styles, such as the universal and Wilton switch-boxes, will be the subject of future work.

Following the Spartan IIe, we defined our architecture so that each routing channel contained 108 tracks. These segmented tracks consisted of 24 single-length segment, 72 hex-length segments, and 12 long lines. Assuming that each CLB input can connect to half of the tracks in its channel, a 54:1 MUX (with six SRAM control signals) is used at each input. The flexibility of the VPR tool allowed us to model this architecture by creating a custom architecture file.

The various gating granularities were applied to the set of sixteen MCNC benchmark circuits detailed in Table 1. Each circuit was targeted to the smallest equivalent Xilinx Spartan IIe device that would accommodate it.

Table 1. MCNC circuits used

Circuit	# Clusters	Spartan IIE Device
alu4	381	XC2S50E
apex4	316	XC2S50E
diffeq	375	XC2S50E
ex5p	266	XC2S50E
misex3	350	XC2S50E
apex2	470	XC2S100E
s298	483	XC2S100E
seq	438	XC2S100E
tseng	262	XC2S100E
elliptic	901	XC2S200E
ex1010	1150	XC2S200E
frisc	889	XC2S200E
pdc	1144	XC2S200E
spla	923	XC2S200E
clma	2096	XC2S400E
s38417	1602	XC2S400E

Logic granularity

The three granularities of LUT leakage control investigated for this architecture were the ability to gate a CLB, a LUT, and a single half of a LUT, turning off 64, 16, and 8 SRAM cells, respectively. A full CLB or LUT can be turned off only when completely unused, but gating half of a LUT puts to sleep half of the SRAM cells in a LUT when less than four inputs are used. In order to use only one sleep transistor per LUT, this technique requires that the input that is unused is known *a priori* and always appears in the same place. It is logical to speculate that most FPGA CAD tools use a consistent ordering or can be easily modified to do so. Indeed, the T-VPack tool used in this work follows such an approach [9]. Also, the unused input to the LUT must be set to insure that the asleep portion of the LUT is never accessed.

These three granularities were tested individually and in all possible combinations to determine the overall possible leakage savings in terms of gated SRAM cells. The T-VPack tool [9] was modified to gather logic resource utilization statistics for the benchmark circuits. The distributed version of the tool accepts as its input a blif file composed of LUTs and flip-flops representing the circuit and packs the logic resources into architecture specified CLBs. The modified version of the tool examines the internal representation of the circuit and determines the number of unused and partially used LUT resources that can be gated off at various granularities.

Table 2 gives the arithmetic average across the set of benchmark circuits for the various logic granularities, presented in terms of both logic SRAM asleep and total device SRAM asleep. Several combination strategies are nearly equivalent to the granularities presented and are omitted for space. Another important point to note is the dramatic difference between the percentages of logic SRAM asleep and total SRAM asleep at each granularity. This stems from the fact that, on average, logic SRAM represents less than 10% of total device SRAM, while routing SRAM occupies over 90%. The area overhead of each strategy is given in the last row of Table 2.

Table 2. % LUT SRAM cells asleep and area overheads for different granularities

Granularity	CLB	0.5 LUT	Full LUT	CLB+0.5LUT
% Logic SRAM Asleep	19.33%	23.41%	19.39%	33.07%
% Total SRAM Asleep	1.55%	1.97%	1.55%	2.75%
Area Overhead	0.16%	0.66%	0.66%	0.82%

The data presented in Table 2 yields some important insights. The CLB granularity provides a modest percentage of asleep cells but with a very small area overhead. The full LUT granularity does not provide much in additional sleep percentage over the CLB (as T-VPack packs an entire CLB before starting a new one), and the significantly higher area overhead dictates that the granularity be discarded. The half-LUT granularity provides better results than the CLB granularity, as there are a large number of utilized LUTs that use less than four inputs. The largest percentage of asleep cells is provided by the combination of CLB and half-LUT granularities. Overall, if maximum leakage savings is desired, the combined CLB/half-LUT granularity should be used, but if area overhead must be minimized, the pure CLB granularity still provides good savings.

Routing granularity

Even greater amounts of SRAM-based resources can be gated by targeting interconnect because FPGA area is dominated by routing resources. Given the need to accommodate a wide variety of potential signal routings, FPGAs are designed with an extensive flexible routing network. However, when a circuit is actually placed and routed, the vast majority of the routing network goes unused. Thus, another ideal opportunity is presented for the application of post place-and-route gating.

Our methodology examines the routing utilization characteristics of circuits to identify unused switch-points

and CLB inputs. By providing sleep transistors for the configuration SRAM cells of switch-point and MUX select line groups, we are able to gate a large amount of unused routing resources. Unused switch-points are identified by examining the four tracks connected to each switch-point. Due to the node zero voltage rise issue discussed in Section 3.1, only switch-points that connect to entirely signal-free tracks can be considered unused and be safely gated. Coarser granularities of control can be created by clustering switch-points and controlling each group with a single sleep transistor. The finest interconnect gating granularity is the individual switch-point while the coarsest granularity is that of gating a full switch-box. MUX select line control cells may also be successfully gated when a CLB input is unused.

In order to measure the desired routing resource utilization characteristics, the VPR place and route tool [2] was modified to track these statistics. The number of switch-boxes is easily determined based on the size of the CLB array and interconnect network. To determine the number of switch-points in the entire FPGA, the routing resource graph on which VPR is built is parsed to identify the true number of signals being routed through each switch-box. To identify switch-point usage statistics at various granularities, VPR's internal representation of the placed and routed netlist of a circuit is parsed to determine the track usage on each side of a given switch-box. Each switch-box was then analyzed to determine the number of unused switch-points that could be gated off. VPR was also modified to examine the placed-and-routed netlist and determine the pin usage for each CLB.

Table 3 gives the arithmetic mean across all of the benchmark circuits of the gating ability of the routing strategies and granularities. We consider the two extremes of switch-point granularity, full switch-box and individual switch-point, as well as CLB input gating. We also consider the combination of switch-point and input gating. Results are presented in terms of both routing only SRAM asleep and total device SRAM asleep. The area overhead introduced by each technique is also given.

Table 3. % Routing SRAM cells asleep and area overheads for different resources and granularities

Granularity	Switchbox	Switchpoint	Inputs	SP & Inputs
% Routing SRAM Asleep	4.50%	47.71%	7.72%	55.42%
% Total SRAM Asleep	4.14%	43.86%	7.09%	50.95%
Area Overhead	0.18%	11.13%	2.62%	13.75%

It is clear from these results that the majority of unused SRAM cells are in the interconnect network. We also note that the combination of switch-point gating and input gating enables almost 51% of device SRAM to be gated with a 13.75% area overhead. This gating percentage is a direct combination of the two individual gating percentages and results from the complimentary nature of the two gating strategies. It is clear from these results that there is significant opportunity for reducing SRAM leakage by gating unused routing resources.

4. Results

The analysis in Section 3 identified effective logic and routing granularities for sleep transistor gating. In this section, we analyze the leakage current reduction provided by these techniques and granularities. To gauge the leakage reduction obtained by the various gating techniques, the circuit-level and architecture-level results have been synthesized into a model of a 70nm FPGA. The detailed circuit-level leakage simulation results for the various blocks of the FPGA form basic units of leakage for each resource type. These leakage values are then combined linearly based on the architecture-level resource usage statistics and gating strategy to determine an overall leakage value for a given circuit and target architecture.

This total leakage is compared against a base leakage for each target architecture to determine leakage savings. When determining the base leakage, a conservative assumption was made that each resource is programmed in its lowest leakage state. The results presented can thus be viewed as the minimum of the attainable leakage savings.

The percentage leakage reduction results of applying the combination of the most effective gating techniques to each benchmark circuit are shown in Table 4. As derived in Section 3, the most effective gating granularities (considering both leakage reduction and area overhead) are CLBs, switch-points (SP), and CLB inputs. The additional savings of gating half LUTs and CLBs is also considered. The second column in the table shows the base SRAM leakage current for the device. The percentage leakage current reduction results presented here are given for regular and high V_t sleep transistors for both combinations of granularities. The average savings across the benchmark set and the area overheads of each strategy are also given. There is no delay penalty for any of the configurations due to the circuit-level (no SRAM switching and no load on active interconnect segments) and architecture-level (no CAD tool restrictions) factors discussed in Section 3.

As these results indicate, approximately 30-40% reductions in leakage can be achieved depending on the gating strategy. Given that [8] has shown that leakage represents 59% of the total power consumption in an SRAM-based FPGA, the leakage current reduction provided by our approach will result in significant power savings. It is clear that the choice of logic granularity has little effect on the overall leakage current savings. This is again due to the dominance of routing resources in FPGAs. It is also important to note the non-trivial leakage

differences between using regular vs. high V_t sleep transistors. The decision to use one over the other must be determined by process cost and power goals.

Table 4. SRAM cell leakage current reduction

Circuit	Base Ioff (mA)	CLB+SP+Inputs		CLB&0.5LUT+SP+Inputs	
		Reg Vt	High Vt	Reg Vt	High Vt
alu4	4.62	23.22%	38.55%	24.02%	39.44%
apex4	4.62	26.41%	40.17%	27.20%	41.05%
diffeq	4.62	25.28%	40.42%	26.12%	41.35%
ex5p	4.62	31.16%	42.81%	31.29%	42.98%
misex3	4.62	25.05%	39.28%	25.86%	40.17%
apex2	6.93	29.02%	40.27%	29.58%	40.90%
s298	6.93	34.10%	44.23%	34.50%	44.70%
seq	6.93	31.39%	42.34%	31.96%	42.98%
tseng	6.93	43.38%	51.32%	43.51%	51.49%
elliptic	12.93	31.84%	41.51%	32.49%	42.24%
ex1010	12.93	19.94%	34.21%	21.13%	35.52%
frisc	12.93	32.53%	41.47%	33.02%	42.02%
pdc	12.93	17.54%	31.01%	17.86%	31.40%
spla	12.93	28.75%	38.58%	28.98%	38.87%
clma	25.35	25.78%	35.71%	26.29%	36.29%
s38417	25.35	36.02%	43.14%	36.81%	44.01%
Average		28.84%	40.31%	29.41%	40.96%
Area Overhead		13.91%		14.57%	

5. Conclusions and future work

In this paper, we have shown how significant leakage current reduction can be achieved in SRAM-based FPGAs. Combining a bottom-up/top-down approach for determining effective ground-gating granularities for turning off SRAM cells in FPGA logic and routing resources, significant leakage current reductions were achieved with manageable area overhead and no delay penalty. In future FPGA generations, the use of these techniques can help control the inherent leakage problems induced by scaling. This work provides a straightforward yet effective start in addressing this significant problem.

The techniques presented here were static for a given FPGA configuration, and no restrictions were placed on the synthesis, mapping, and place-and-route tools. Once the circuits are placed-and-routed onto the device using VPR, our post-processing techniques identify which resources are unused, and the SRAM cells at the gate inputs of the sleep transistors can be programmed accordingly. Putting restrictions on the design tools may increase the percentage of SRAM cells that can be turned off with coarser sleep transistor granularities, but the impact on area and delay must be considered. Future work in this area will quantify this tradeoff. Finally, the tradeoffs associated with other gating techniques (e.g. V_{dd} gating) and multi-threshold processes will be explored, as will the impact of temperature on leakage.

6. Acknowledgements

This work is supported in part by the National Science Foundation under grant No. CCR-0105626.

7. References

[1] Agarwal, A., Roy, K., "A Single-Vt Low-Leakage Gated-Ground Cache for Deep Submicron," *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 2, pp, 319-328, February 2003.

[2] Betz, V., Rose, J., "VPR: A New Packaging, Placement and Routing Tool for FPGA Research," *International Conference on Field-Programmable Logic and Applications*, pp. 213-222, 1997.

[3] Berkeley Predictive Technology Model: http://www-device.eecs.berkeley.edu/~ptm

[4] Calhoun, B., Honore, F., Chandrakasan, A., "Design Methodology for Fine-Grained Leakage Control in MTCMOS," *International Symposium on Low Power Electronics and Design*, pp. 104-109, 2003.

[5] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," *Custom Integrated Circuits Conference*, pp. 201-204, 2000.

[6] Chen, Z., Johnson, M., Wei, L., Roy, K., "Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks," *International Symposium on Low Power Electronics and Design*, pp. 239-244, 1998.

[7] Kam, T., Rawat, S., Kirkpatrick, D., Roy, R., Spirakis, G., Sherwani, N., Peterson, C., "EDA Challenges Facing Future Microprocessor Design," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 12, pp. 1498-1506, December 2000.

[8] Li, F., Chen, D., He, L., Cong, J., "Architecture Evaluation for Power-Efficient FPGAs," *International Symposium on Field-Programmable Gate Arrays*, pp. 175-184, 2003.

[9] Marquardt, S., Betz, V., Rose, J., "Using Cluster-Based Logic Blocks and Timing Driven Packing to Improve FPGA Speed and Density," *International Symposium on Field-Programmable Gate Arrays*, pp. 37-46, 1999.

[10] Poon, K., Yan, A., Wilton, S., "A Flexible Power Model for FPGAs," *International Conference on Field-Programmable Logic and Applications*, pp. 312-321, 2002.

[11] Rabaey, J., Chandrakasan, A., Nikolic, B., *Digital Integrated Circuits*, 2nd Edition, Prentice Hall, New Jersey, 2003.

[12] Rose, J., Betz, V., Marquardt, S., *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, Boston, 1999.

[13] Spartan IIe 1.8 V FPGA Family: Complete Data Sheet.

[14] Sylvester, D., Kaul, H., "Future Performance Challenges in Nanometer Design," *Design Automation Conference*, pp. 3-8, 2001.

[15] Tuan, T., Lai, B., "Leakage Power Analysis of a 90nm FPGA," *Custom Integrated Circuits Conference*, pp. 57-60, 2003.

[16] Wolff, F., Knieser, M., Weyer, D., Papachristou, C., "High-Level Low Power FPGA Design Methodology," *National Aerospace Conference*, pp. 554-559, 2000.