Fine-grain Leakage Optimization in SRAM based FPGAs

Abstract

*FPGAs are evolving at a rapid pace with improved performance and logic density. At the same time, trends in technology scaling makes leakage power a serious concern for designers. In this paper, we propose a hierarchical look-up table (LUT) structure for FPGAs to improve leakage power consumption. We present a detailed analysis on the number of inputs actually used by LUTs, and we observe that on an average 47% LUTs do not use one or more inputs. In the proposed hierarchical LUT structure depending on the number of inputs used by the LUTs we shut off certain SRAM cells and transistors associated with the unused LUT inputs. Based on this technique, for 180nm technology, we report an average savings of **22.94**% (as high as **64.22%**) in leakage power per LUT. The savings will be even greater for technologies as low as 90nm currently in use for FPGA production as well as for future technologies.*

1. Introduction

Reconfigurable technologies have made remarkable progress over the last decade. Commercial FPGAs available today provide a wide range of functionalities along with the added benefits of low non recurring engineering (NRE) cost and higher flexibility. However, even with the tremendous improvement in system performance and logic density of FPGAs, power efficiency has continuously lagged behind these improved capabilities. Hence, several mainstream low-power application domains (e.g. mobile applications) restrict the use of FPGAs because of this prohibitive power consumption. Moreover, increased packaging and cooling costs, and decreased system reliability can also be attributed to high power dissipation. Hence, it is extremely important to improve the power efficiency of FPGAs.

CMOS devices have been scaled down for several years to achieve higher performance and logic density, and FPGAs at 90nm technology are now being developed. Various FPGA manufacturers have roadmaps to use the 65 nm technology in near future[1]. However, with each generation of technology scaling of supply voltage (V_{dd}), threshold voltage (V_t), channel length, and gate oxide thickness, there is a significant increase in leakage current. A reduction in V_{dd} is accompanied by a reduction in V_t as well to compensate for performance penalties, which results in an exponential increase in subthreshold leakage. On the other hand, thinning down gate oxides to improve driving capability leads to a substantial increase in gate leakage. These trends in technology scaling makes leakage power a dominant component in total power consumption. Therefore it is imperative to concentrate on leakage power optimization techniques.

Our aim is to accomplish this goal in this paper. Specifically, we investigated the impact on logic block leakage power by shutting down unused transistors in the look-up tables (LUTs). This opportunity comes from the fact that, the flexibility offered by an FPGA to target many applications, results in a large portion of the logic being left unused. In fact, prior studies show that typically 38% of the logic structures of an FPGA remain unused [1] and leakage power is consumed by both the used and the unused parts. In addition, leakage power is proportional to the total transistor count [2] and consequently shutting down unused transistors will lead to savings in leakage power. We performed a preliminary investigation on the variance of LUT utilization across several circuits. We observed that there are a significant number of LUTs for which one or more inputs remain unused. We present details of this analysis in Section 3. This motivated us to devise a *hierarchical* LUT structure, where the complexity of the LUT can be reduced incrementally based on the number of inputs required by the logic function it needs to implement. Reduction of LUT complexity is achieved by selectively shutting down transistors and SRAM cells that are associated with the unused inputs. This complexity reduction is done in hierarchical steps: from 16 cells SRAM array (4-input effective LUT size) to 8 cells SRAM array (3-input effective LUT size), from 8 cells SRAM array to 4 cells SRAM array (2-input effective LUT size), and so on.

There are pros and cons to employing a leakage control technique at the LUT level. The disadvantage is due to the overhead associated with the sleep transistors that we will need to employ to perform V_{dd} gating. First, these sleep transistors will bring some area overhead. However, since FPGA area is predominantly determined by routing area, an increase in logic area will not affect the total chip area significantly. In addition, leakage control techniques specifically target deep submicron technologies, 90 nm and below. The continuous downsizing of feature sizes will further reduce the impact of an increase in logic area. There will be some degradation on LUT delay due to the sleep transistors. The impact on performance in our hierarchical LUT structure is kept minimal. We will elaborate on this in Section 3.

The most important advantage of providing leakage control at the LUT level is the fact that we do not affect the packing, placement, and routing stages in any way. Each of these stages perform optimizations to improve important design metrics such as logic utilization, congestion/routability, wirelength, and delay. We do not want to impair their ability in reaching an optimal solution. Additional constraints imposed on packing and/or placement can affect routability adversely leading to infeasible, i.e. unroutable designs, or solutions with increased wirelength. This in turn will increase the interconnect power, which can overshadow the expected improvement in leakage power. Therefore, we assume that we cannot anticipate a priori, which LUTs in a design will allow complexity reduction and where these LUTs will be placed. This is the reason why we aim to provide a leakage reduction technique that is compatible with any placement result. Our specific contributions in this paper are:

- We introduce a low-penalty optimization technique to reduce leakage power consumption in FPGA logic blocks by

[1] Xilinx and IBM have a roadmap to produce chips at 65 nm. Lattice and Fujitsu are discussing the use of Fujitsu's forthcoming 65 nm technology in future Lattice products.

exploiting the variance in LUT utilization across different designs, and

- We analyze and evaluate the leakage power gain associated with this optimization technique.

The rest of this paper is organized as follows: Section 2 presents an overview of related work on leakage power optimization in FPGA technology. Section 3 briefly discusses the structures of the basic components used in an FPGA logic block, based on which our optimization technique is proposed. Section 4 illustrates our approach to optimize logic block leakage power. We also present statistical information on LUT utilization, which motivated such an optimization. Section 5 presents our results: leakage power savings based on actual power consumption, as well as savings in leakage power based on the number of transistors that can be shut down, and show that these estimates are consistent with each other. Section 6 summarizes our conclusions.

2. Related Work

Leakage power optimization techniques for ASICs have been extensively studied. A detailed study on leakage current mechanism and leakage reduction techniques for CMOS circuits is presented by Roy et al. [3]. Although a variety of leakage power optimization techniques have been proposed for ASICs and microprocessors in the past [4-9], reducing leakage power for FPGAs has been in focus only recently.

Until recently, most of the power optimization techniques for FPGAs primarily focused on dynamic power reduction. Sheng et al. [10] analyzed dynamic power consumption in Virtex II FPGA family. Li et al. [11] developed fpgaEVA-LP for power efficiency analysis of LUT table based FPGA architectures.

Several techniques for reducing leakage power were proposed in the past year. Gayasen et al. [1] proposed a technique for disabling unused portions of the FPGA through region constraint placement employing sleep transistors that control coarse grain regions of the FPGA. Our technique provides a finer grain leakage reduction capability. As explained earlier, we intend to avoid placing any constraint on the placement of logic blocks and we aim to provide a good leakage control solution for an arbitrary placement. In Section 3, we will discuss how we can tune our hierarchical LUT structures to achieve a solution that will have a more stringent control over the associated overhead of V_{dd} gating. Anderson et al. [2] proposed an optimization technique that selects polarities for logic signals at the inputs of LUTs so they spend the majority of their time in low leakage states. Our technique can be viewed complementary to this approach, where signal polarity assignment is performed on the active part of the logic blocks whereas inactive portions of the LUTs can benefit from our technique. Li et al. [12] proposed a scheme where the SRAM cells use a high V_t (which reduces 15× leakage power) with a 13% increase in configuration time. However, for deep submicron designs beyond 65 nm, V_t cannot be increased beyond a certain limit, since V_{dd} would be scaled down significantly. Rahman and Polavarapuv [13] evaluate several low-leakage design techniques for FPGAs and conclude that multiple V_t switch blocks are very effective in reducing leakage power dissipation. Our proposed optimization can again co-exist with both of these techniques. Calhoun et al [14] propose a design methodology using Multi-Threshold CMOS gates for leakage reduction and demonstrate the application of this design technique onto a reconfigurable architecture. Their approach is a general design technique for CMOS designs. Whereas our approach aims to introduce leakage optimization into LUTs without changing the LUT design methodology fundamentally.

3. FPGA Logic Block Structures

Before we proceed to describe our leakage reduction techniques, we will briefly discuss the structure of a logic block and its components that are commonly used in our target architecture.

Many modern FPGAs use island-style architecture, which consist of an array of logic blocks, I/O blocks, and programmable routing. The logic and the I/O blocks are connected through a programmable routing fabric. Logic is implemented using look-up tables (LUTs). In essence, a *k*-input LUT (*k*-LUT) is a small memory that can implement any function with at most *k* inputs. A *k*-LUT is built with 2^k SRAM cells and a 2^k:1 multiplexer, where the SRAM cells are programmed to be the truth table of the *k*-input function the LUT implements. Commercial FPGAs mostly use 4- LUTs, and previous work has shown that 4-LUTs have highest area efficiency [15].

As we will discuss in more detail in Section 5, we have performed our experiments using the VPR tool flow. Our architectural assumptions are closely related to the target architecture used in VPR and the diagrams depicting the representative architecture are based on descriptions in [16]. The 4-LUT shown in Figure 1 uses 16 SRAM cells, a 16-input pass transistor based multiplexer, and a set of buffers. Each SRAM cell consists of 6 minimum-width transistors. The total number of transistors for this LUT is 167 (96 for the SRAM cells, 30 for the multiplexer tree, and 41 for the input buffers and complementers) [16].

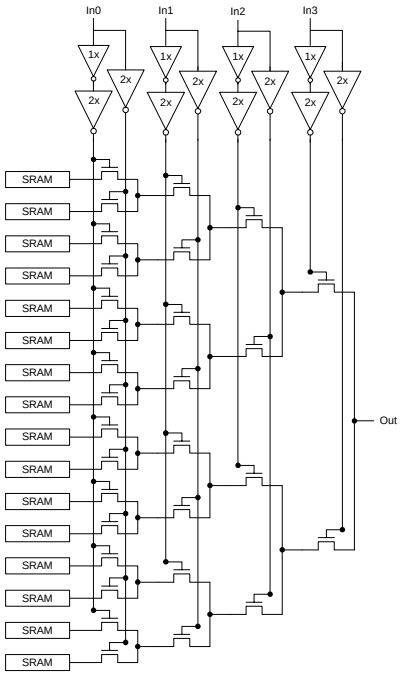

Figure 1. A 4-input LUT.

A LUT, a flip-flop, and a multiplexer are grouped together to form a *logic element* as shown in Figure 2. Logic blocks of modern FPGAs consist of a cluster of logic elements called Configurable Logic Blocks (CLBs), arranged in different hierarchical organizations. For instance, a few LUTs are grouped together to form a slice and several slices are grouped to form a CLB. Input multiplexers enable the communication between the

inputs to the logic clusters and the inputs of individual LUTs within the cluster. In our target architecture we assume that any input to the logic cluster can be routed to any LUT input.

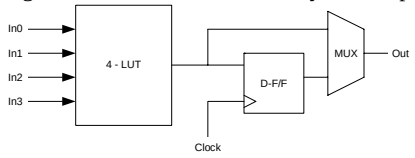

Figure 2. A logic element

4. Leakage Power Optimization

In this section, we will first present a preliminary analysis we performed on a set of benchmarks to assess the variance in LUT utilization across different designs. Next, we will introduce our proposed hierarchical LUT structure.

4.1 Variance in LUT Utilization

An analysis of the 20 MCNC benchmarks [17] after technology mapping using Flowmap [18] shows that if a circuit is mapped onto an FPGA containing 4-LUTs, there are many LUTs that do not use all 4 inputs. Table 1 shows the distribution of 2-, 3-, and 4-input LUTs (in addition to the unused LUTs) needed for each of these 20 MCNC benchmarks.

Table 1. Distribution of used LUT inputs

Circuit	# 2-LUT	# 3-LUT	# 4-LUT	Unused	Total
alu4	121	446	955	78	1600
apex2	117	589	1172	238	2116
apex4	24	538	700	182	1444
bigkey	350	4	1353	1209	2916
clma	546	2040	5797	453	8836
des	88	323	1180	2505	4096
diffeq	144	440	913	103	1600
dsip	10	4	1356	1546	2916
elliptic	435	1024	2145	240	3844
ex1010	190	1944	2464	302	4900
ex5p	45	227	792	92	1156
frisc	293	965	2298	44	3600
misex3	66	502	829	47	1444
pdc	84	979	3512	325	4900
s298	173	432	1326	185	2116
s38417	723	2508	3175	318	6724
s38584.1	1891	1167	3389	277	6724
seq	129	584	1037	186	1936
spla	53	892	2745	154	3844
tseng	133	283	631	109	1156

Figure 3 shows the distribution of the 2-, 3-, and 4-input LUTs shown as a percentage of the total number of LUTs actually present. From Figure 3 it is observed that on an average 53% of the 4-LUTs use all their inputs. Although using 4-LUTs yields high utilization rate, at the same time 47% of the LUTs do not use one or more inputs.

Based on this observation, we propose a technique that will save leakage power. Instead of having LUTs with fixed number of inputs we propose a *hierarchical look-up table*, which can yield LUTs with varying number of inputs. In this structure we employ V_{dd} gating to hierarchically cut-off power supply to one quarter or half of the original 4-input LUT. This will in effect yield a 3-input LUT or a 2-input LUT from a 4-input LUT selectively.

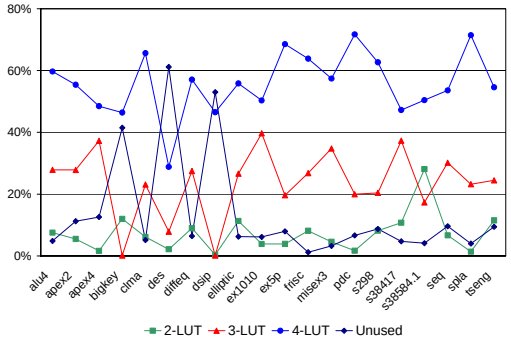

Figure 3. Distribution of LUT inputs

The hierarchical LUT structure will employ mechanisms to disable unused parts of the SRAM array and the output multiplexer as well as to de-activate multiplexers associated with unused LUT inputs. In the following we will elaborate on the hierarchical LUT structure.

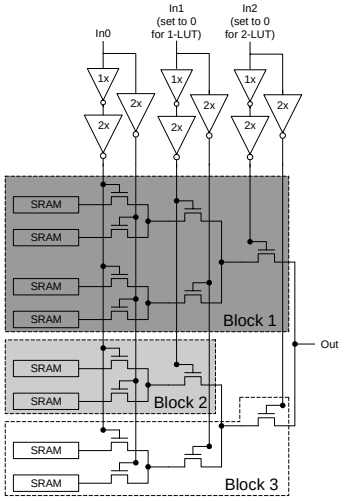

Figure 4. Implementation of hierarchical 3-input LUT (The LUT complexity can be reduced from 3-LUT configuration to 2-LUT configuration).

4.2 Hierarchical LUT

In this section we first describe how we construct a hierarchical LUT structure. Next, we present an analysis of the incurred overhead by employing the proposed scheme.

4.2.1 Hierarchical reduction of SRAM array

After the LUTs are packed into complex logic clusters we assess how many inputs each LUT will use based on the logic mapped to it. Depending on the number of inputs each LUT uses, a portion of the SRAM array and the associated output multiplexer will be deactivated.

For example, if a 3-LUT as shown in Figure 4 uses only two of its inputs then the SRAM cells inside the hashed block marked Block 1 can be shut off, and the LUT input In2 has to be set to 0. In this manner, input In2 controls the pass transistor at the third level of the multiplexer tree and disconnects the upper half of the LUT structure from the active lower half of the LUT structure.

Similarly, if the 3-LUT uses only one of its inputs, then Block 2, in addition to those in Block 1 will be shut off. In this case, both In2 and In1 have to be set to 0. If all the 3 inputs are unused, then the entire SRAM cell array along with the multiplexer can be shut down. We assume that the LUT inputs are so utilized such that the unused inputs are always the higher order inputs, i.e. only In2 is unused if the 3-LUT uses only two of the 3 inputs. This implies that we can enforce LUTs to use (hence not use) specific inputs. Some FPGAs are not built like that, which can affect the effectiveness of our approach. For our current work we make the aforementioned assumption of inputs being interchangeable. In the case of a more restricted architecture, techniques such as reorganizing the SRAM configuration to use specific LUT inputs could be applied.

Table 2. Transistor count for different LUT sizes

# LUT inputs	for SRAM	for MUX	for I/P Buffers	Total
2	24	8	41	73
3	48	15	41	104
4	96	30	41	167

The 4-LUT consists of 16 SRAM cells and 30 transistors for the multiplexer tree. In the hierarchical LUT structure 8 SRAM cells and 15 transistors of the multiplexer tree will be shut off when we configure the LUT as a 3-input LUT. Similarly for a 2-LUT configuration, 12 SRAM cells and 22 transistors can be shut off. Based on these observation and since leakage power is proportional to the transistor count [2], we hope to obtain a good amount of savings on leakage power. The total number of active transistors for 2-, 3-, and 4-LUT (including only the SRAM cells and the transistors contained in the output multiplexer trees) are shown in Table 2.

4.2.2 Elimination of active input multiplexers

Usually, complex logic clusters are designed such that any input to the logic cluster is accessible by any LUT input pin within the cluster as we described in Section 3. An input multiplexer is used for each LUT input to route any external cluster input to individual LUT inputs.

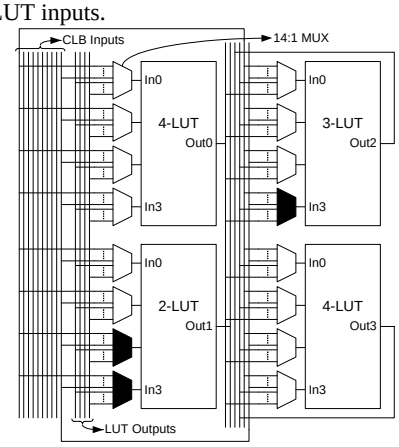

Figure 5. Disabling of input multiplexers based on the LUT configuration.

Generally, the number of inputs to a complex cluster is less than the total number of LUT inputs contained in the cluster. In other words, a cluster of four 4-input LUTs has less than 16 inputs. There have been studies to determine the best number of cluster inputs for a given cluster size. For example, for a logic cluster that has four 4-LUTs, 10 inputs is determined to be optimal in terms of logic utilization [16], which we have used for our experiments. In that case there are sixteen 14:1 input multiplexers, because each input of each of the four 4-LUTs needs one input multiplexer. The multiplexers are 14:1 because each LUT input can come from any of the 10 cluster inputs or 4 LUT outputs within the cluster. Now, if a 4-LUT is reduced to a 3-LUT, then the number of these input multiplexers is reduced by 1 per LUT.

In other architecture configurations a logic cluster can have a full set of 16 inputs. Then, there are sixteen 20:1 (16 external inputs to the cluster and 4 additional inputs generated through feedback from the 4 LUTs within the cluster) multiplexers. In such a case if the number of inputs to the LUTs can be decreased, that can add up to even larger savings in leakage power. Figure 5 depicts a CLB with 10 inputs, which contains four 4-LUTs, where two of these are configured as 4-LUTs, one is configured as a 3-LUT, and one is configured as a 2-LUT. The input multiplexers shown in black can be shut-off in this case.

4.2.3 V_{dd} Gating and Overhead Estimation

In this section we discuss how V_{dd} gating is performed to shut off the SRAM cells associated with the unused LUT inputs, and also discuss the associated overhead. We use a SRAM-controlled sleep transistor that cuts off the power supply for each of the blocks shown in Figure 4. A schematic for the V_{dd} gating for a group of two SRAM cells is shown in Figure 6, where each SRAM cell structure is shown inside the dashed box. The structure of the SRAM cell that controls the sleep transistor is the same as the other SRAM cells, but the details are not shown in the figure for clarity.

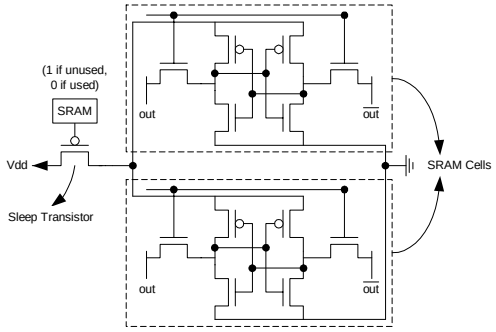

Figure 6. Vdd Gating for SRAM cells.

When a 3-LUT (as shown in Figure 4) uses only two inputs Block 1 is shut off by using the sleep transistor associated with Block 1. The sleep transistor for Block 1 in case of a 3-LUT can shut down power for the four SRAM cells and the seven transistors in the MUX tree. The SRAM cell associated with the sleep transistor has a value of 0 or 1 based on whether an input is unused or used. Similarly, when the 3-LUT uses only one input, both Block 1 and Block 2 can be shut off using the corresponding sleep transistors. When none of the inputs are used (i.e. the LUT is unused), we shut off all the SRAM cells and the entire MUX tree.

This technique uses *k* SRAM-controlled sleep transistors for a *k*-LUT, and each such sleep transistor uses 7 transistors (1 for the V_{dd} gating, and 6 for the additional SRAM cell). Therefore, for a k-LUT we need an extra $k \times 7$ transistors. Based on this, we will need 28 extra transistors for a 4-LUT in addition to the 167 transistors needed for a 4-LUT (as shown in Table 2). This addition of sleep transistors will hence increase the logic block

area by 16.8%. However, since FPGA area is predominantly determined by routing area, an increase in logic area will not affect the total chip area significantly.

5. Experimental Results

The effectiveness of the proposed leakage reduction technique is evaluated for 1.8V 180nm technology. We have used parameters at this technology due to their immediate availability and we performed power measurements based on this technology. We start by describing our methodology and subsequently present the experimental results.

5.1 Methodology

We start with estimating the leakage power for a LUT. For this we have used Power Model [19], an additional module integrated with Versatile Place and Route tool (VPR) [16]. Since VPR only allows defining architectures with a single LUT type, i.e. single LUT size, we have estimated the leakage power of LUTs of different sizes in the following manner. We first packed the logic of each benchmark using only one LUT per logic cluster. We repeated this for three different cases; using only 4-LUTs, 3-LUTs, and 2-LUTs. Then, for each implementation we have divided the total logic block leakage power by the number of logic blocks used. We obtained an average leakage power measure for different LUT sizes in this fashion, and the results are shown in Table 3. We use these values together with the data presented in Table 1 in Section 4 to estimate the savings in leakage power that can be achieved by using a hierarchical LUT structure. Moreover, since leakage power is proportional with the number of transistors, we also estimated the leakage power savings in terms of the number of transistors that could be shut off.

5.2 Results

We begin by presenting the leakage power consumption of a LUT shown in absolute and normalized form in Table 3. We have used the normalized value of the LUT leakage power to estimate the overall savings in leakage power using our optimization as compared to using all 4-LUTs. The results are shown in Table 4.

Table 3. Leakage power for different LUT sizes.

# LUT inputs	Absolute (nW)	Normalized
4	760.59	1.00
3	561.72	0.74
2	402.68	0.53

The second column of Table 4 shows the relative amount of leakage power consumed by all LUTs without any optimization. These values are equal to the number of 4-LUTs used to implement the circuit, since the normalized leakage power for a 4-LUT is 1. The third column shows the relative leakage power when the optimization is applied. If a circuit uses x 2-LUTs, y 3-LUTs, and z 4-LUTs, then the value for the optimized power is obtained as $x \times 0.53 + y \times 0.74 + z \times 1.00$. As seen from Table 4, savings in logic block leakage power of about 23% is possible with the proposed optimization technique.

Table 5 shows the achieved savings in terms of the number of transistors that could be shut off. Hence, we can conclude that by using our optimization technique about 26% of the logic block transistors can be shut off, which is consistent with the 23% power savings shown in Table 4.

We would like to emphasize once more that although our results show a leakage power savings close to 23% for 180nm technology, this savings in logic block leakage power will be substantially higher for smaller technologies.

Table 4. Savings in leakage power.

Circuit	Normalized Leakage Power		% Savings
	Unoptimized	Optimized	
alu4	1600.00	1349.17	15.68
apex2	2116.00	1669.87	21.08
apex4	1444.00	1110.84	23.07
bigkey	2916.00	1541.46	47.14
clma	8836.00	7595.98	14.03
des	4096.00	1465.66	64.22
diffeq	1600.00	1314.92	17.82
dsip	2916.00	1364.26	53.21
elliptic	3844.00	3133.31	18.49
ex1010	4900.00	4003.26	18.30
ex5p	1156.00	983.83	14.89
frisc	3600.00	3167.39	12.02
misex3	1444.00	1235.46	14.44
pdc	4900.00	4280.98	12.63
s298	2116.00	1737.37	17.89
s38417	6724.00	5414.11	19.48
s38584.1	6724.00	5254.81	21.85
seq	1936.00	1537.53	20.58
spla	3844.00	3433.17	10.69
tseng	1156.00	910.91	21.20
Average Leakage Power Savings			**22.94**

Table 5. Savings in terms of number of transistors that could be shut down.

Circuit	# Transistors Shut Off		% Savings
	Unoptimized	Optimized	
alu4	0	52498	19.65
apex2	0	87851	24.86
apex4	0	66544	27.59
bigkey	0	235055	48.27
clma	0	255495	17.31
des	0	446956	65.34
diffeq	0	58457	21.88
dsip	0	259374	53.26
elliptic	0	145482	22.66
ex1010	0	190766	23.31
ex5p	0	33895	17.56
frisc	0	95685	15.92
misex3	0	45679	18.94
pdc	0	123848	15.13
s298	0	74373	21.05
s38417	0	279072	24.85
s38584.1	0	297534	26.50
seq	0	79980	24.74
spla	0	86896	13.54
tseng	0	48534	25.14
Average % of CLB transistors shut down			**26.38**

Finally, we may choose to adopt hierarchical LUT structures in a selective manner, i.e. not every single LUT within a logic cluster needs to be hierarchical. We analyzed the distribution of number of inputs used per LUT across logic clusters. The distribution is presented in Table 6. For this benchmark set we observe that approximately 2 out of each 4 LUTs packed within a logic cluster use all 4 inputs. Similarly, 1 out of 4 LUTs use 3 inputs. To reduce the overheads associated with the hierarchical LUT structure, a logic cluster can be configured, where only 2 out of 4 LUTs are designed as hierarchical LUTs.

Table 6. Distribution of LUT sizes across logic clusters.

Circuit	# 2-LUT per cluster	# 3-LUT per cluster	# 4-LUT per cluster
alu4	0.30	1.12	2.39
apex2	0.22	1.11	2.22
apex4	0.07	1.49	1.94
bigkey	0.48	0.01	1.86
clma	0.25	0.92	2.62
des	0.09	0.32	1.15
diffeq	0.36	1.10	2.28
dsip	0.01	0.01	1.86
elliptic	0.45	1.07	2.23
ex1010	0.16	1.59	2.01
ex5p	0.16	0.79	2.74
frisc	0.33	1.07	2.55
misex3	0.18	1.39	2.30
pdc	0.07	0.80	2.87
s298	0.33	0.82	2.51
s38417	0.43	1.49	1.89
s38584.1	1.12	0.69	2.02
seq	0.27	1.21	2.14
spla	0.06	0.93	2.86
tseng	0.46	0.98	2.18
Average	**0.29**	**0.94**	**2.23**

6. Conclusions

The process technology trends in FPGA manufacturing indicate that leakage power will be an increasingly important design concern for future reconfigurable devices. In this paper, we investigated a fine grain leakage control technique, which relies on the observation that a significant amount of logic blocks are underutilized in practice. We addressed this aspect by introducing a hierarchical LUT structure, where depending on the level of utilization, the complexity of individual LUTs can be incrementally reduced via shutting off unused portions.

Variance in LUT utilization can be exploited in different ways. One opportunity is to utilize unused portions of LUTs to improve reliability. We can exploit variance in LUT utilization to embed redundancy into the logic in a systematic fashion.

7. References

[1] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing Leakage Energy in FPGAs Using Region-Constrained Placement," *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2004.

[2] J. Anderson, F. Najm, and T. Tuan, "Active Leakage Power Optimization for FPGAs," *International Symposium on Field-Programmable Gate Arrays*, 2004.

[3] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechnaisms and leakage reduction techniques in deep-submicrometer CMOS CIRCUITS," *Proceedings of the IEEE*, 2003.

[4] T. Kuroda, "Low Power CMOS Digital Design for Multimedia Processors," *Proceedings of ICVC*, 1999.

[5] S. Mutch, S. Shigematsu, Y. Matsuya, H. Fukuda, and J. Yamada, "A 1V Multi-Threshold Voltage CMOS DSP with an Efficient Power Management Technique for Mobile Phone Applications," *Proceedings of the International Solid-State Circuits Conference*, 1996.

[6] L. Clark, S. Demmons, N. Deutscher, and F. Ricci, "Standby Power Management for a 0.18um Microprocessor," *Proc. of ISPLED*, 2002.

[7] K. Roy and S. Prasad, *Low-Power CMOS VLSI Circuit Design*: Wiley-Interscience, 2000.

[8] J. P. Halter and F. N. Najm, "A Gate-Level Leakage Power Reduction Method for Ultra-Low-Power CMOS Circuits," *Proc. of CICC*, 1997.

[9] J. Chen, M. Johnson, L. Wei, and K. Roy, "Estimation of standby leakage power in CMOS circuit considering accurate modeling of transistor stacks," *Proc. of ISPLED*, 1998.

[10] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex™-II FPGA family," *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2002.

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

[12] F. Li, Y. Lin, L. He, and J. Cong, "Low-Power FPGA Using Pre-defined Dual-Vdd/Dual-Vt Fabrics," *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2004.

[13] A. Rahman and V. Polavarapuv, "Evaluation of Low-Leakage Design Techniques for Field Programmable Gate Arrays," *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2004.

[14] B. H. Calhoun, F. A. Honroe, and A. Chandrakasan, "Design methodology for fine-grained leakage control in MTCMOS," *Proceedings of the international symposium on Low power electronics and design*, 2003.

[15] J. Rose, R. Francis, D. Lewis, and P. Chow, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *Proc. of JSSC*, 1990.

[16] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*: Kluwer Academic Publishers, 1999.

[17] S. Yang, "Logic Synthesis and Optimization Benchmarks," Microelectronics Center of North Carolina 1991.

[18] J. Cong and Y. Ding, "FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1994.

[19] K. K. Poon, "Power Estimation for Field Programmable Gate Arrays," MS Thesis in *Dept. of Electrical and Computer Engg.*: University of British Colmbia, 1999.