A 90nm Low-Power FPGA for Battery-Powered Applications

Tim Tuan, Sean Kao, Arif Rahman, Satyaki Das, Steve Trimberger
Xilinx Research Labs
2100 Logic Dr.
San Jose, CA 95124
{tim.tuan, sean.kao, arif.rahman, satyaki.das, steve.trimberger}@xilinx.com

ABSTRACT

Programmable logic devices such as FPGAs are useful for a wide range of applications. However, FPGAs are not commonly used in battery-powered applications because they consume more power than ASICs and lack power management features. In this paper, we describe the design and implementation of *Pika*, a low-power FPGA core targeting battery-powered applications such as those in consumer and automotive markets. Our design uses the Xilinx Spartan-3 low-cost FPGA as a baseline and achieves substantial power savings through a series of power optimizations. The resulting architecture is compatible with existing commercial design tools. The implementation is done in a 90nm triple-oxide CMOS process. Compared to the baseline design, Pika consumes 46% less active power and 99% less standby power. Furthermore, it retains circuit and configuration state during standby mode, and wakes up from standby mode in approximately 100ns.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles - *Advanced technologies; Gate arrays; VLSI (very large scale integration)*.

General Terms

Design.

Keywords

Field-programmable gate arrays, FPGAs, programmable logic, low-power design.

1. INTRODUCTION

A key challenge in the IC scaling era is delivering high performance solutions while minimizing power and cost. Programmable logic devices such as FPGAs address this challenge by providing a cost-efficient solution for low to mid-volume applications due to low NRE. Also, with in-field programmability, FPGAs provide a platform solution with faster time to market and longer product lifetime. Despite its many advantages, FPGAs are not widely found in today's mobile applications.

Mobile applications generally have two power requirements, active power and standby power. A typical user application has extremely low duty-cycle, where the device is active for a short period of time (less than 1 hour), and then is inactive for a long period of time (days or weeks). During the periods of activity, the device must be energy efficient, that is, perform the necessary functions while consuming minimum energy. During the idle periods, the device must consume little or no power to extend battery life. The active power requirement for a typical mobile IC is on the order of 100's of mW, while its standby power requirement is on the orders of 10's to 100's of µW [2][3].

It is well known that FPGAs are less energy efficient than dedicated logic due to the overhead of reconfigurability. Thus, for applications that require maximum energy efficiency, it is necessary to use dedicated solutions such as standard cell ASICs or even custom IC. However, FPGAs have been shown to be more energy efficient than instruction-based processors such as microprocessors and DSPs [4], yet processors today are widely used in battery-operated applications while FPGAs are not. While ease of programming is a major reason, another reason is the extensive power management capabilities found in processors that enable very low power consumption during standby.

Because FPGAs have been primarily used for high-throughput data processing where there is virtually unlimited energy supply, they have little or no power management capability to minimize power during idle periods. The problem is exacerbated by the rapid increase in leakage power due to the scaling of processing technology. Current low-cost FPGAs consume up to 100's mW of standby power [21], while high-end FPGAs can consume over 1W [24]. Compared to current mobile ICs, the FPGA's standby power is at least 2 orders of magnitude higher than what is required.

In this paper, we present the design and implementation of *Pika*, a low-power FPGA core targeting battery-powered applications. We base our design on the Xilinx® Spartan™-3, a low-cost commercial FPGA. Due to practical concerns, we constrain the design to be compatible with existing software and process technology. Compared to the baseline Spartan-3 core, Pika

FPGA'06, February 22–24, 2006, Monterey, California, USA.

consumes 46% less active power and 99% less standby power. Furthermore, it retains circuit and configuration state during standby mode, and wakes up from standby mode in approximately 100ns.

The paper organization is as follows. In Section 2, we discuss some of the related past work. Section 3 presents the architecture and power consumption of our baseline FPGA. In Section 4, we discuss each of the power optimization techniques applied to the baseline architecture and various design challenges we encounter. In Section 5, we present and analyze our results on power, performance, area, and transient behaviors of the standby mode. Lastly, Section 6 concludes this paper.

2. RELATED WORK

One of the early applications of power gating was a multi-threshold CMOS (MTCMOS) DSP design that used high-Vt transistors to power gate low-Vt circuits to achieve a combination of high performance and low standby power [5]. The 0.5µm design achieved 1000X power reduction in standby mode. An example of aggressive voltage scaling is the XScale processor [6], a commercial 0.18µm microprocessor that operates down to 0.75V to achieve dramatic power reduction at low frequencies. It also uses body-biasing to provide a state-retaining standby mode with ~20X power reduction. More recently, power gating was applied to processor IP blocks in a 0.13µm process [7]. Test results showed 300X standby power reduction. One of the few results in 90nm was a DSP processor that achieved 40X leakage reduction using a combination of power gating and multi-threshold design [8]. In our work, we employ some of these concepts and apply them to an FPGA architecture in an efficient and flexible fashion.

Recently, low-power FPGA design has become an area of interest. Various CAD solutions for power reduction have been reported that address dynamic power as well as static power [9]-[12]. In the area of hardware design, a 0.25µm FPGA [13] with low dynamic energy was designed using low-swing interconnect, dual-edge triggered flip flops, and a novel interconnect architecture. Various low-power circuit techniques for reconfigurable logic have been studied in recent years, including body biasing [14], multi-Vt logic [14], gate biasing [14], fine-grain power gating [15], low-voltage interconnect [16], and heterogeneous interconnect [17]. Lastly, several dual-Vdd, dual-Vt FPGA architectures with corresponding CAD techniques have been proposed [18]-[20].

In contrast from previous work, our work focuses on a commercial FPGA architecture, which presents practical constraints on cost, performance, process choices, and software compatibility. Due to those constraints, we find that most of the previous work cannot be feasibly applied to our architecture. Also, by addressing a 90nm process, where both subthreshold leakage and gate leakage are significant, we face a more complex problem than in older processes, and the results are more indicative of future trends.

3. BASELINE ARCHITECTURE AND POWER

We use the Xilinx Spartan-3 [21] as our baseline architecture and apply a set of power optimizations to achieve the resulting low-power architecture. Spartan-3 is a low-cost FPGA built in a 1.2V, 90nm CMOS process. We choose a low-cost baseline architecture because many low-power applications are also cost-sensitive. To ensure our FPGA can be manufactured and programmed without substantial support on process technology and software design, we keep our design compatible with existing processes and CAD tools. This restriction also facilitates comparison of our results against the baseline design.

In this section, we describe the baseline architecture and its power consumption profile.

3.1. Baseline Architecture

Figure 1 shows a block diagram of the Spartan-3 FPGA core. The architecture comprises an array of configurable logic blocks (CLB), which are the basic element that can be programmed to perform various logic functions. Each CLB is coupled with a programmable interconnect switch matrix that connects the CLB to adjacent and nearby CLBs. We will refer to each CLB/switch matrix pair as a *tile*.

Each CLB has four logic slices. Each logic slice has two four-input lookup tables (4LUTs), two configurable flip-flops (FFs), and some additional circuitry for fast arithmetic operations and wide-input functions. The behavior of the CLB can be configured in many ways. For example, the 4LUTs can be configured to perform any logic functions with up to four inputs, or be combined to perform wider input functions. The FFs can be configured as level-sensitive latches or edge-triggered flops. Furthermore, in two of the four slices in each CLB, the 4LUTs can also be configured as 16-bit distributed RAM or 16-bit shift registers. All of these configurations are defined by control bits stored in local configuration SRAM cells (not shown).

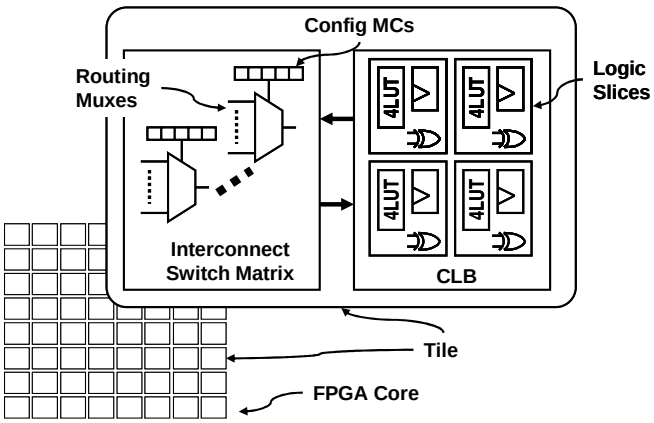


Figure 1: Architecture of the baseline Spartan-3 core.

Each interconnect switch matrix comprises numerous programmable switches that drive CLB outputs to other CLBs or select CLB inputs from signals in the interconnect. As shown in Figure 1, each programmable switch is a buffered multiplexer controlled by a set of configuration SRAM cells. The output switches are categorized by the distance of their output wire. Direct switches drive wires that span one CLB. Double and Hex switches drive wires that span two and six CLBs, respectively. Long switches reach the entire width or height of the array. Although the programmable switches are simple in structure,

they typically dominate the CLB power consumption because they make up a large portion of the total area.

In addition to the CLBs and the switch matrices, the Spartan-3 core also has a number of specialty blocks such as 18Kb block RAMs (BRAMs), 18-bit multipliers and digital clock managers (DCMs). They provide efficient implementation of complex functions common to many applications, such as multiplication and clock deskewing. These specialty blocks are beyond the scope of this paper. However, the optimizations we describe can be applied to the specialty blocks to achieve similar savings.

3.2. Baseline Power Consumption

The power consumption of the FPGA core is dependent on switching activity, configuration state, resource utilization, and temperature. In previous work [22][23], we had described detailed power simulation flows for FPGAs. In this section, we present the power consumption of the baseline architecture obtained by applying the previously published techniques.

Figure 2 shows the core power consumption of the Spartan-3 family excluding the specialty blocks. *Active* power is the power consumed during normal operation. It consists of both dynamic power and static power. *Standby* power is the power consumed when the part is idle and no signals are switching.

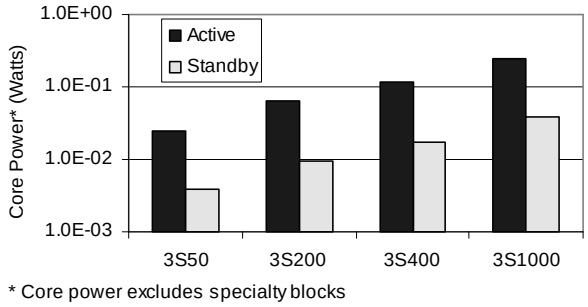


Figure 2: Typical power consumption of Spartan-3 cores (specialty blocks excluded).

Dynamic power is estimated at 150 MHz clock frequency, 12.5% average switching activity, and typical configuration and resource utilization as determined by user benchmarks. Static power, which includes both subthreshold leakage and gate leakage, is measured at 85C for active power to represent the warmer operating conditions due to higher power consumption, and 25C for standby power to represent the cooler idle conditions. Note that these definitions are intended to represent a typical scenario to simplify data analysis, and do not encompass all possible scenarios.

For the array sizes shown, we see that active power is on the order of 10's-100's of mW, while standby power is on the order of 1's-10's of mW. Compared to existing low-power solutions for mobile applications, the active power is comparable, while the standby power is several orders of magnitude too high. Therefore, we prioritize our efforts to target dramatic standby power reduction, and approach active power as a secondary goal.

We further break down total power to identify high power components (Figure 3). Routing switches make up the largest part of the total dynamic power. Both routing switches and configuration SRAM represent significant parts of the total static power.

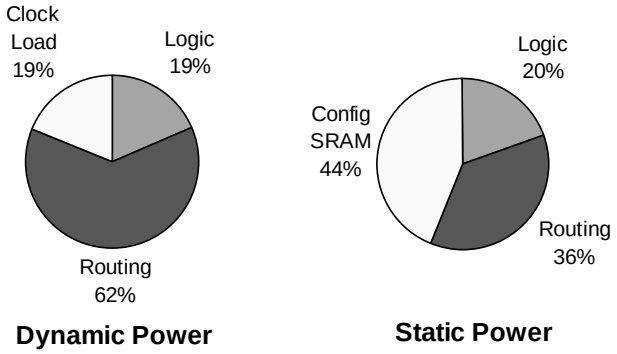


Figure 3: Breakdown of baseline Spartan-3 power consumption.

4. POWER OPTIMIZATIONS

In this section, we describe each of the power optimizations applied to the baseline architecture.

4.1. Voltage Scaling

Voltage scaling is particularly effective for reducing power because dynamic power and static power are quadratic and exponential functions of the supply voltage, respectively, while circuit speed is approximately a linear function of the supply voltage [1]. Therefore, designs that are optimized for performance, as is the case with conventional FPGAs, typically use a relatively high supply voltage for speed at the expense of higher power. Consequently, lowering the supply voltage of a high-speed design can often yield a more energy efficient solution.

Figure 4 shows the speed and the static power of a test circuit, a 4LUT driving two Double switches, at different voltages. Dynamic power is not shown because its quadratic dependency is well known. Looking purely at energy efficiency, one would typically set the supply voltage at a point where the power-delay product (PDP) is minimal. In the example in Figure 4, PDP continues to drop even below 0.8V, where performance degradation becomes prohibitively large. Also, at those low voltages, circuit reliability is compromised due to reduced noise margin, especially when considering the effects of process variation.

Consequently, considering performance, energy efficiency and reliability, we choose 1.0V as our core operating voltage. In the affected blocks (excluding the configuration SRAM), this gives approximately 30% reduction in active power and 40% reduction in standby power. The chip's peak performance is reduced by 15%. We do not scale the configuration SRAM's voltage because they are more effectively addressed via a different approach that is described in the next section.

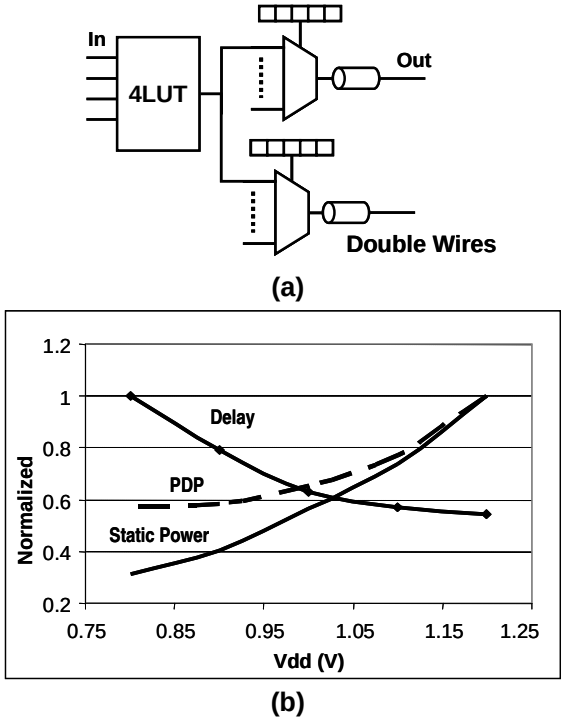


Figure 4: (a) A test circuit used to capture voltage scaling behavior. (b) Delay, static power, and PDP of the test circuit at various voltages.

4.2. Low-Leakage Configuration SRAM

As shown in Figure 3, configuration SRAM represents 44% of the core leakage power, making it a good candidate for power optimization.

Configuration SRAM cells do not switch during normal operation, therefore they can be made much slower with no impact on chip performance. It has been suggested to use high-Vt transistors for configuration SRAM to save leakage power [10][18]. However, high-Vt devices only address subthreshold leakage, while gate leakage, an increasingly large portion of the total leakage, is unaffected. Our analysis shows that gate leakage can be more than 50% of the total leakage power at low temperatures. Without addressing gate leakage, one cannot do better than cutting SRAM leakage in half.

The presence of significant gate leakage mandates that a low leakage device must have thicker gate oxide. Standard thick-oxide devices used for IO buffers are much too large to be used for millions of configuration SRAM cells. Alternatively, we use a mid-oxide, high-Vt device for the configuration SRAM cells. The mid-oxide transistor is available in the triple-oxide process used by the Virtex™-4 FPGA family [24]. Therefore, its use does not impose on us a new process technology.

Using mid-oxide, high-Vt transistors dramatically cuts both subthreshold leakage and gate leakage in the configuration SRAM. As shown in Figure 5, total SRAM leakage is reduced by nearly two orders of magnitude. This corresponds to a reduction of 43% to the total standby power of the FPGA core. The use of mid-oxide transistors does require additional mask cost and causes some die area increase (the latter will be covered in Section 5), but the power savings sufficiently justify the added cost.

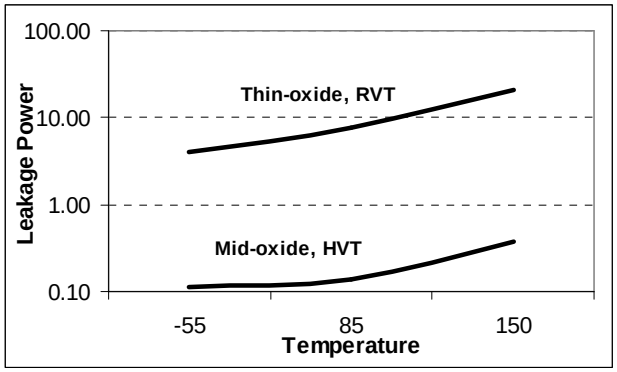


Figure 5: Leakage of thin-oxide, regular-Vt SRAM and mid-oxide, high-Vt SRAM.

4.3. Power Gating for Active Leakage Reduction

Power gating is a well known power reduction technique in ASICs and microprocessors [5][7][8]. To support power gating, one or more power transistors are inserted between a circuit block and its power and/or ground. When the power gates are turned on, active operation is unaffected except for a small performance penalty due to the on-resistance of the power gates. When the power gates are turned off, circuit current is limited to the leakage current of the power gates.

Typically, power gating is applied to coarse-grain functional blocks to reduce standby leakage when the block is temporarily idle. In FPGAs, power gating has been proposed to reduce the chip's active leakage power by power gating unused blocks [10]. Here we describe our implementation of power gating for active leakage reduction. Later in Section 4.4, we will describe the use of power gating for standby power reduction.

4.3.1. Granularity

One of the main design decisions of power gating is the granularity of the smallest block that can be independently power gated. At one extreme, each LUT, FF, and routing switch may be independently gated [15]. This approach leads to a larger number of unused blocks and hence greater power savings, but the area overhead is also larger. At the other extreme, clusters of 20 or more tiles may be controlled by a single power gate [10]. This approach has the benefit of less area overhead from power gating [7], but fewer clusters will be completely unused. As a compromise between the two approaches, we opt for power gating at the level of individual tiles. This granularity lends itself to a fairly efficient physical implementation. Across over 100 benchmarks, we find that on average, 25% of the tiles are left unused and can be power gated. A diagram of the power gating architecture is shown in Figure 6.

Although configuration SRAM cells are embedded within the switch matrices, we choose not to power gate the SRAM cells in our design for two reasons. First, they consume very little power because they use low-leakage transistors. Second, by leaving the configuration memory powered while the rest of the core is

power gated, we enable a state-retaining standby mode where all logic and routing are power gated. The details of standby mode will be described in Section 4.4.

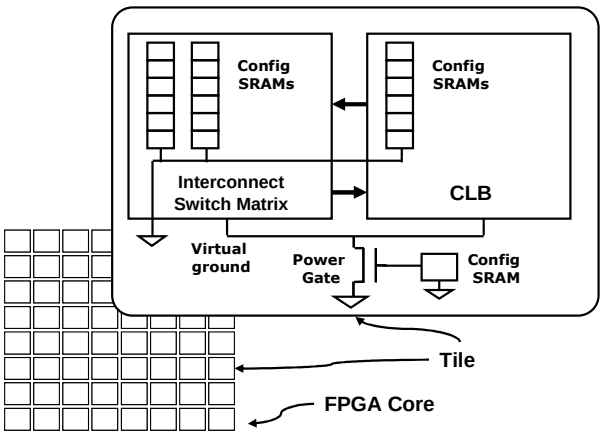
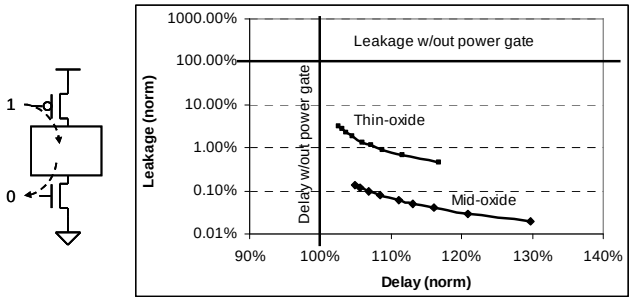


Figure 6: Proposed power gating architecture, where configuration SRAM cells are not power gated

4.3.2. Power Gate Design

Another design consideration is the implementation of the power gating transistors. The most straightforward power gating implementation is to use both PMOS and NMOS power gates, but by using only one or the other, one can achieve comparable power savings with much less area overhead. For the same size, NMOS power gates generally give better speed characteristics than PMOS power gates due to greater carrier mobility. In our simulations, we find the power savings are similar for both. Thus, we choose NMOS-only power gates.

Conventional power gating uses thin-oxide, high-Vt transistors [5], which is susceptible to gate leakage in a 90nm process. Figure 7a shows the gate leakage paths introduced by thin-oxide, high-Vt power gates. Figure 7b shows the power and delay behavior of thin-oxide and mid-oxide power gating. Each data point represents a different power gate size. Although thin-oxide power gating is potentially faster, it always consumes much higher power than mid-oxide power gating due to the additional gate leakage.



(a) **(b)**

Figure 7: (a) Sleep transistors introduce new gate leakage paths. (b) Sizing of thin-oxide and mid-oxide power gate produces leakage-delay trade-offs.

Consequently, we implement mid-oxide power gates, and size them at the point of 10% performance degradation. Note that although Figure 7b shows that mid-oxide power gates can reduce power by over 1000X, the existence of high-leakage paths at the input and output interfaces limits the actual power reduction. This is discussed further in the next Section.

4.3.3. Input/Output Shielding

While power gating limits all current paths to ground within an inactive block, care must be taken to avoid high-current paths at its inputs and outputs when interfacing to an active block. A high-current path may be a short-circuit path from supply to ground, or a high-leakage path (regular-Vt subthreshold leakage or thin-oxide gate leakage) that does not go through a power gate. In an FPGA, this problem is exacerbated because in addition to regular inputs, control signals from the configuration SRAM cells, which are always active, can create high-current paths as well. Figure 8 shows four examples of high-current paths in a power gated block (annotated by **a**, **b**, **c**, and **d)**. Here, all memory cells are standard 6T cells, and their outputs are directly tapped from the cross-coupled inverters inside the cells.

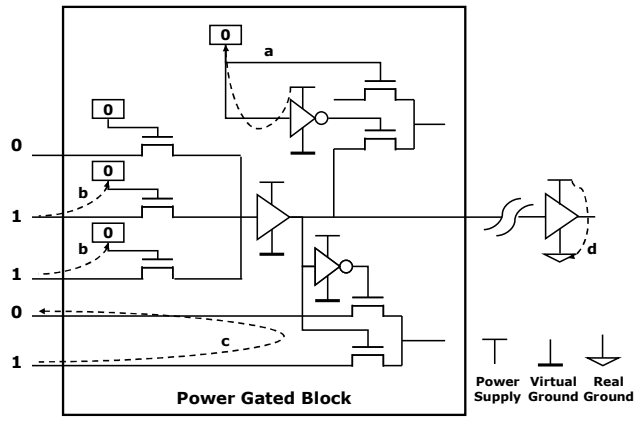


Figure 8: An example circuit demonstrating various high-current paths that may exist in a power gated block.

In example **a**, a memory cell storing 0 drives a power gated inverter, which subsequently controls a 2-1 mux. Here, a leakage path exists from the inverter's supply to the ground of the memory cell. Namely, it is the gate leakage of the thin-oxide PMOS of the inverter. In example **b**, externally driven inputs go into a 3-1 mux. When those inputs are 1, leakage paths exist between the drivers' supply, through the gate oxide of the pass transistors, into the control memory cells' ground. In example **c**, a 2-1 mux is controlled by outputs of gates that are power gated. Since these outputs are floating, the two inputs of the mux are shorted, creating a short-circuit path when the inputs are different. In example **d**, an output of a power gated driver, which is floating, drives an active buffer, causing a short-circuit path between the buffer's power and ground.

Figure 9 shows the same circuit with modifications that prevent the high-current paths highlighted in Figure 8. To remedy the problem in example **a**, we modify the configuration SRAM by adding a mid-oxide power gated output buffer to shield the memory cell from the incoming high-leakage path. This adds an area penalty in memory cell area, but eliminates many of the leaky paths at memory cell interfaces with no performance penalty. To remedy the problem in **b**, we modify those pass-gates to use mid-oxide transistors. This eliminates the gate

leakage paths, and the performance impact is minimal because these pass-gates do not switch. In example **c**, we insert shielding buffers at the inputs to eliminate the pass-gate short-circuit paths. To remedy the problem in **d**, we add a pull-up transistor at the output to prevent floating outputs driving other active blocks.

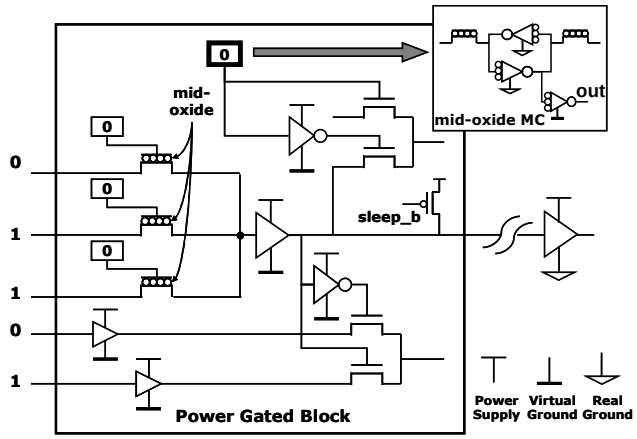


Figure 9: The example circuit modified to address the high-current paths.

4.4. Standby Modes

With infrastructures that support tile-level power gating in place, it takes a simple extension to support a low-power standby mode where all tiles are power gated while the circuit configuration is retained because the configuration memory remains powered. In order to retain the circuit state stored in FFs, we capture the FF content into dedicated configuration memory cells before entering the standby mode, and restore the FF states when coming out of the standby mode (these circuits exist in the Spartan-3 FPGA for the purpose of readback and FF initialization). A simple controller is designed to coordinate the necessary standby and wake up sequences. The controller also prevents contention during wake up by asserting a global reset, similar to the one used during the chip's initial power on.

4.4.1. Partial Standby Mode

Many applications call for a small amount of functionality during standby mode to detect events that trigger a wake up. For example, an automotive controller unit may reside on a local bus, and the wake up event may be in the form of a bus instruction. A small bus controller would be necessary to interpret the wake up event.

Having tile-based power gating enables a partial standby mode where a small amount of logic remains powered during the standby mode. Since each tile can be independently power gated, any combination of tiles may be selected to remain active during standby mode. This decision is programmable by setting the value of a single configuration bit per tile. Only a small amount of additional control logic in each tile is needed to support this feature (Figure 10). The extra configuration bits necessary for this feature are taken from previously unused bits in each tile.

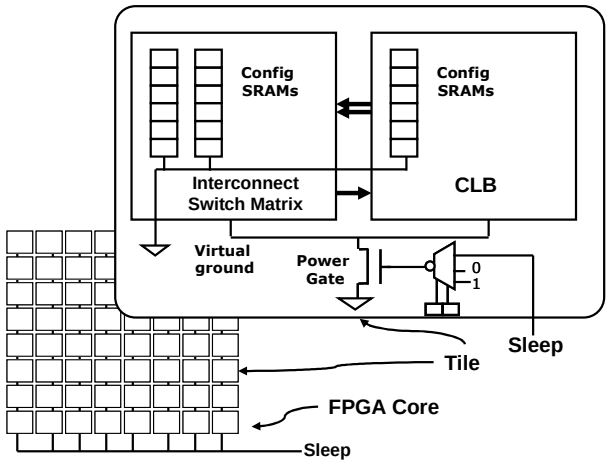


Figure 10: Simple changes to a power gating architecture enable a partial standby mode.

5. RESULTS

We implemented Pika in a 90nm dual-Vt, triple-oxide CMOS process. Figure 11 shows the physical layout of a small core with 8x16 tiles. In this section, we discuss the quantitative results in terms of power, area and performance.

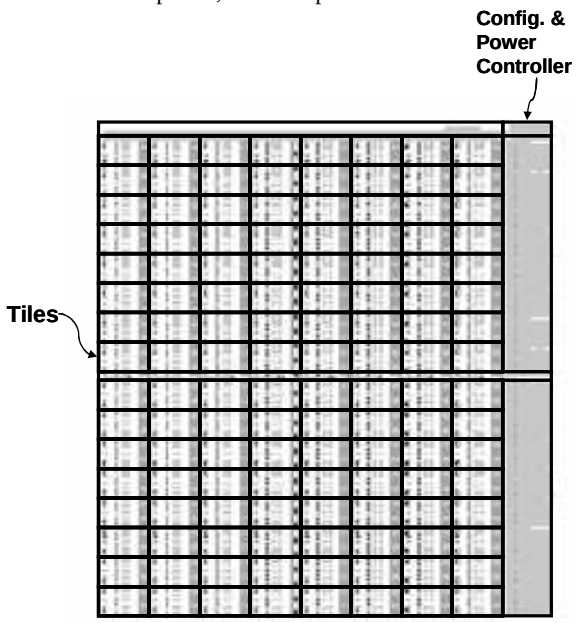


Figure 11: Layout of a core with 8x16 tiles.

5.1. Power

Figure 12 and Figure 13 show the power comparison between Pika and equivalent Spartan-3 cores, for active power and standby power, respectively. Both power measures are based on typical conditions for average designs (same as those used in Section 3.2). Since Pika does not have specialty blocks, those blocks are excluded from the Spartan-3 cores in this comparison.

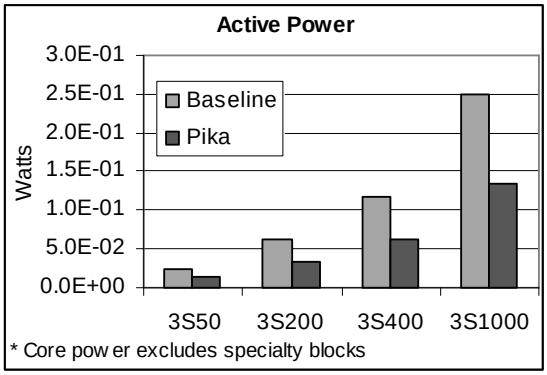


Figure 12: Active power comparison between baseline and Pika design for various size arrays.

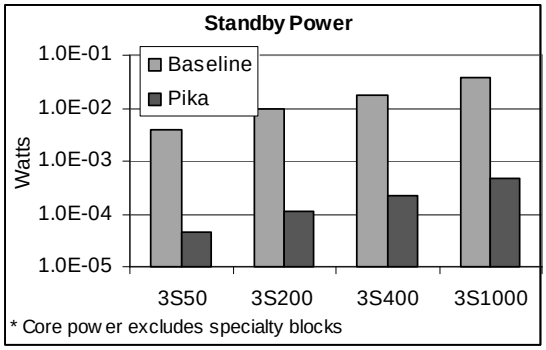


Figure 13: Standby power comparison between baseline and Pika design for various size arrays.

Overall, Pika's active power is 46% less than an equivalent Spartan-3 core, while its standby power is reduced by 99%. The active power improvement comes primarily from voltage scaling as well as static power improvements in the configuration SRAM design. The standby power reduction primarily comes from static power improvements and the standby mode. Because the implemented techniques are not circuit-specific, the power breakdown by resource type is approximately the same as that in Figure 3. Compared to the aforementioned mobile ICs standby power requirements of 10's-100's µW, the proposed low-power design falls within those ranges for the chip densities shown.

Table 1 shows a breakdown of the power reduction. For each technique, we estimate the power reduction we would achieve if we apply that technique alone. Since not all techniques are mutually exclusive, the parts do not add up to the sum.

Table 1: Summary of each technique's impact on total core power

	Active Power	Standby Power
Voltage Scaling	35%	23%
Low-Leakage Config SRAM	13%	43%
Power Gating Unused Blocks	7%	NA
Standby Mode	NA	51%
Total Power Reduction	46%	99%

5.2. Area

Compared to an equivalent Spartan-3 core, Pika is 40% larger in area. This area increase has a negative effect on chip power and performance, both of which are accounted for in our results. Area increase also negatively affects chip cost, although the relation is not one to one. When accounting for the costs of testing, assembly, and packaging, die cost is only a fraction of the total chip cost.

Table 2 shows the breakdown of the area increase. Of the 40% total area increase, only 16% is directly due to the design optimizations we have chosen, e.g. the introduction of power gates, mid-oxide transistor, and thus are unavoidable. Another part of the area increase is due to the differences between the low-cost dual-oxide process used in the baseline Spartan-3 and the high-performance triple-oxide process used in Pika. This part amounts to approximately 6%, and it can potentially be improved by tuning the process for low cost. Lastly, we estimate that 18% area increase can be salvaged through layout optimizations that were not done in this work.

Table 2: Breakdown of the area increase incurred from power optimizations.

Cause	Area Increase
Process Migration	6%
Midox SRAMs	8%
Power Gates	8%
Unoptimized floorplan & layout	18%
Total Area Increase	40%

5.3. Performance

We assess Pika's performance by looking at the delay of a typical data path consisting of both logic and routing resources. The ratio of routing to logic in this path is determined by profiling critical paths in benchmark designs. Because the architecture is unchanged, and the circuit-level changes are applied uniformly across all blocks, we expect the performance impact to be relatively independent of the user application.

As expected, the power reduction techniques adversely affect the FPGA's performance. We find the total performance impact in Pika is approximately 27%. Of that, approximately 7% is due to power gating, 5% is due to layout area increase, and the rest is from voltage scaling. Performance penalty from power gating is less than what we intended because in our physical layout, we are able to up size the power gates to fill in open spaces.

5.4. Transient Behavior during Mode Transitions

One of the objectives of this design is fast wake up time from standby mode. Figure 14 shows the power curve of a single tile entering and exiting standby mode. The exit or wake up time is shown to be approximately 100ns. This time is dominated by the time it takes to fully charge the gate of the power gate. When all tiles are woken in parallel, the entire core wakes up in approximately 100ns as well.

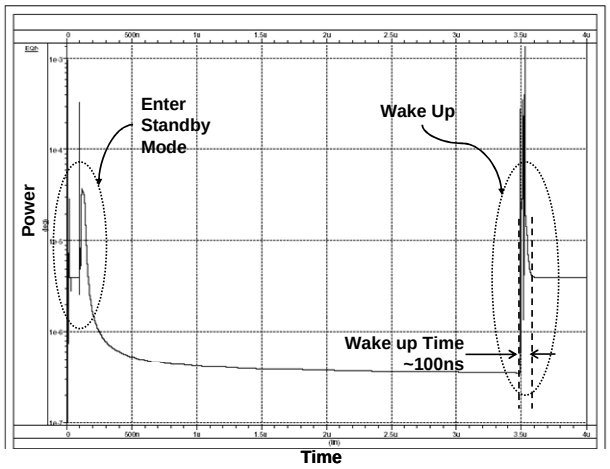


Figure 14: Power consumption of a single tile entering and exiting standby mode

It takes a certain amount of energy to enter and exit standby mode. For very short idle periods, it may require more energy to perform the mode transitions than to remain in active mode. We explored the minimum standby period for which it is beneficial to enter standby mode. Figure 15 shows two energy curves. The *Standby Mode* curve shows the energy of a tile transitioning into and out of the standby mode. The *Active Mode* curve shows the energy of a tile remaining in active mode. We see that after less than 2µs, the active mode tile consumes more energy than that consumed by the mode transitions. Therefore, for our design, the minimum standby period is less than 2µs.

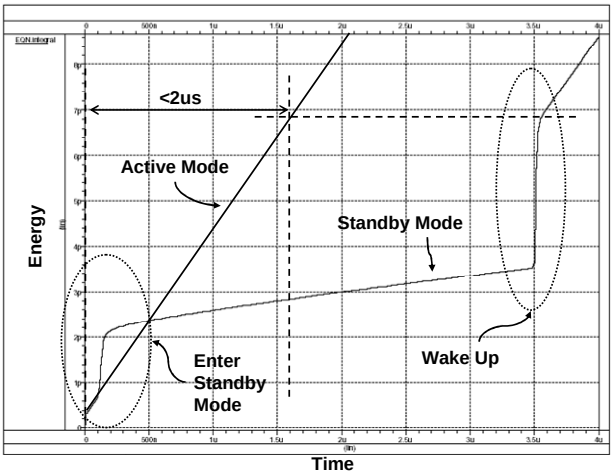


Figure 15: Energy consumption of using standby mode vs. remaining active.

6. CONCLUSION

FPGAs present numerous advantages in deep-submicron IC design. However, high power consumption, in particular high standby power, has thus far prevented FPGAs to be widely adopted in battery-powered applications.

We have presented the design and implementation of Pika, a 90nm low-power FPGA core based on a low-cost commercial FPGA architecture. The power-optimized design uses low-leakage memory cells, voltage scaling, and power-gating to achieve low active and standby power. In 90nm process and beyond, gate leakage is a significant and growing problem. In our design, we address this problem without compromising substantial performance and area by effectively leveraging a triple-oxide process.

Detailed simulations under typical use conditions show significant power reduction at the expense of moderate performance and area penalty. In operating mode, active power is reduced by 46%. When idle, using the proposed standby mode, standby power is reduced by 99%. The power optimizations incur a 27% performance penalty and 40% area increase. During the standby mode, circuit state and configuration state are both retained. Furthermore, the core wakes up from standby mode in approximately 100ns.

7. REFERENCES

[1] J. Rabaey., A. Chandrakasan, B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd Edition, Prentice Hall, 2003.

[2] Intel Corp., *PXA270 Processor Datasheet*, http://www.intel.com.

[3] Texas Instruments, *OMAP5910 Dual-Core Processor Data Manual*, http://www.ti.com.

[4] P. Schumacher, M. Paluszkiewicz, R. Ballantyne, and R. Turney, "An Efficient JPEG2000 Encoder Implemented on a Platform FPGA", SPIE Annual Meeting, 2003.

[5] S. Mutoh et al, "A 1V Multi-Threshold Voltage CMOS DSP with an Efficient Power Management Technique for Mobile Phone Application", Proceedings of ISSCC, 1996.

[6] L. Clark, N. Deutscher, F. Ricci, S. Demmons, "Standby Power Management for a 0.18um Microprocessor", Proc. of Int'l Symp. on Low Power Electronics and Design, 2002.

[7] R. Puri, L. Stok, S. Bhattacharya, "Keeping Hot Chips Cool", Proceedings of Design Automation Conference, 2005.

[8] P. Royannez, "90nm Low-Leakage SoC Design Techniques for Wireless Applications", Proc. of Int'l Solid-State Circuits Conference, 2005.

[9] J. Lamoureux, S. Wilton, "On the Interaction Between Power-Aware FPGA CAD Algorithms", Proc. of Int'l Conference on CAD, 2003.

[10] A. Gayasen et al, "Reducing Leakage Energy in FPGAs Using Region-Constrained Placement", Proc. of Int'l Symp. on FPGA, 2004.

[11] J. Anderson, F. Najm, "Power-Aware Technology Mapping for LUT-Based FPGAs", Proc of Int'l Conference on Field-Programmable Technology, 2002.

[12] J. Anderson, F. Najm, T. Tuan, "Active Leakage Power Optimization for FPGAs", Proc. of Int'l Symp. on FPGAs, 2004.

[13] V. George, H. Zhang, J. Rabaey, "The Design of Low Energy FPGA", Proc. of Int'l Symp. on Low Power Electronics and Design, 1999.

[14] Rahman, "Evaluation of LowLeakage Design Techniques for Field Programmable Gate Arrays", Proc. of Int'l Symp. on FPGA, 2004.

[15] B. Calhoun, F. Honore, A. Chandrakasan, "Design Methodology for Fine-Grained Leakage Control in MTCMOS", Proc. of ISLPED, 2003.

[16] J. Anderson, F. Najm, “Low-Power Programmable Routing Circuitry for FPGAs”, Proc. of Custom Integrated Circuits Conference, 2004.
[17] A. Rahman, S. Das, T. Tuan, A. Rahut, “Heterogeneous Routing Architecture for Low Power FPGA Fabric”, Proc. of Custom Integrated Circuits Conference, 2005.
[18] F. Li, Y. Lin, L. He, J. Cong, “Low-Power FPGA Using Pre-defined Dual-Vdd/Dual-Vt Fabrics”, Proc. of Int’l Symp. on FPGA, 2004.
[19] F. Li, Y. Lin, L. He, J. Cong, “FPGA Power Reduction Using Configurable Dual-Vdd”, Proc. of Design Automation Conference, 2004.
[20] F. Li, Y. Lin, L. He, “A Fully Vdd-Programmable Fabric for Low-Power FPGAs”, Proc of Int’l Conference on CAD, 2004.
[21] Xilinx Inc., Spartan-3 FPGA Family Datasheet, http://www.xilinx.com.
[22] T. Tuan, B. Lai, “Leakage Power Analysis of a 90nm FPGA”, Proc. of Custom Integrated Circuits Conference, 2003.
[23] V. Degalahal, T. Tuan, “Methodology for High Level Estimation of FPGA Power Consumption”, Proc. of ASP-DAC, 2005.
[24] Xilinx Inc, Virtex-4 FPGA Family Datasheet, http://www.xilinx.com.